IMPLEMENTASI METODE *RANDOM FOREST* DAN *XGBOOST* PADA KLASIFIKASI *CUSTOMER CHURN*

TUGAS AKHIR

Adhelia Nurfira Rachmi

16611054

PROGRAM STUDI STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2020

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul	:	Implementasi Metode *Random Forest* dan *XGBoost* Pada Klasifikasi *Customer Churn*
Nama Mahasiswa	:	Adhelia Nurfira Rachmi
NIM	:	16611054

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, (07 November 2020)

Pembimbing

(Arum Handini Primandari, S.Pd.Si., M.Si.)

HALAMAN PENGESAHAN

TUGAS AKHIR

IMPLEMENTASI METODE *RANDOM FOREST* DAN *XGBOOST* PADA KLASIFIKASI *CUSTOMER CHURN*

Nama Mahasiswa : Adhelia Nurfira Rachmi

NIM : 16611054

TUGAS AKHIR INI TELAH DIUJIKAN
PADA TANGGAL: 07 November 2020

Nama Penguji:	**Tanda Tangan**
1. Muhammad Muhajir, S.Si.,M.Sc.	…….
2. Dina Tri Utari, S.Si.,M.Sc.	……..
3. Arum Handini Primandari, S.Pd.Si., M.Si.	……..

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

(Prof. Riyanto, S.Pd., M.Si., Ph.D.)

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Assalamu'alaikum Warahmatullaahi Wabarakaatuh

Alhamdulillahi Robbil 'Alamiin, Puji syukur atas kehadirat Allah SWT, Sang pencipta alam semesta yang telah melimpahkan rahmat, hidayah, serta kasih sayang-Nya sehingga penulis diberi nikmat sehat, kekuatan, serta kesabaran hingga penulis dapat menyusun dan menyelesaikan tugas akhir yang berjudul "**Implementasi Metode *Random Forest* dan *XGBoost* Pada Klasifikasi *Customer Churn***" untuk memenuhi salah persyaratan dalam menyelesaikan jenjang strata satu dan untuk mencapai gelar Sarjana Statistika di Jurusan Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Sholawat serta salam selalu tercurahkan kepada junjungan Nabi besar Muhammad SAW, keluarga dan sahabatnya yang syafaat-Nya kita nantikan di akhir zaman nanti.

Penulisan dan penyusunan tugas akhir ini tentunya tidak terlepas dari pihak-pihak yang memberikan bantuan baik moril maupun materiil, dukungan, arahan, serta bimbingan. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih kepada:

1. Kedua Orang Tua penulis, Bapak Maswan Jayadi dan ibu Ruliana yang telah banyak berjuang dan berkorban moril maupun materiil untuk dapat menyekolahkan penulis hingga di perguruan tinggi, doa, dukungan, serta motivasi untuk segala kelancaran hingga saat ini.
2. Kedua adik penulis Fadila Rizki Ananda dan Nabila Rizki Adinda yang memberikan saya motivasi serta semangat untuk menjadi orang yang berguna untuk kehidupannya kelak.
3. Keluarga besar H.M. Dawani H.Z. yang selalu mendoakan dan memberikan semangat dalam masa studi penulis serta membantu penulis dalam hal materiil.

4. Keluarga H. Mas'ud yang selalu mendoakan dan banyak membantu kedua orang tua penulis dalam menyekolahkan anak-anaknya hingga pada jenjang tertinggi.
5. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D., selaku dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
6. Bapak Dr.Edy Widodo, S.Si., M.Si., selaku ketua Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
7. Ibu Arum Handini Primandari, S.Pd.Si., M.Si. selaku Dosen Pembimbing yang selalu memberikan motivasi serta pelajaran kepada penulis dan telah sabar memberikan bimbingan dari awal hingga selesainya Tugas Akhir ini.
8. Bapak Ibu Dosen Program Studi Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia yang telah mendidik dan mengajarkan dari semester awal hingga akhir kepada penulis.
9. Penguruh inti IKS FMIPA UII Periode 2018/2019, Departemen Sosial Masyarakat LEM FMIPA UII Periode 2017/2018, Pengurus Inti ETIKA 2018, dan SC ETIKA 2017, selaku keluarga dalam bidang non akademik yang telah memberikan banyak pengalaman dan pelajaran baru tentang kepemimpinan dalam berorganisasi, berkepanitiaan, bersosialisasi, berbicara di depan umum, dan mementingkan kepentingan umat diatas kepentingan pribadi agar hidup jauh lebih bermanfaat.
10. Teman-teman seperjuangan Billa, Nadia, Akmal, Surya, Faisal, Iqbal, Iwik, Dinda, Cici, Indri, Mba Mey, Mba Rachel, Mba Fea, Mba Salwa yang sudah menemani, memberikan banyak motivasi, mengingatkan dalam hal kebaikan, dan *mood booster* selama kuliah.
11. Sahabat seperjuanggan bimbingan Tugas Akhir khususnya Maudi dan Putri yang telah bekerjasama dengan baik, saling menguatkan dan memberikan hiburan selama proses pengerjaan Tugas Akhir.
12. Sahabat Statistika 2016 yang telah banyak membantu dan menemani perjuangan selama empat tahun masa perkuliahan.

13. Semua pihak yang tidak dapat disebutkan satu persatu. Terima kasih atas doa, bantuan, serta dukungan kalian.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna dikarenakan keterbatasan pengetahuan dan kemampuan yang dimiliki oleh penulis. Oleh karena itu, penulis mengharapkan kritik serta saran yang bersifat membangun untuk menyempurnakan penulisan Tugas Akhir ini. Semoga Tugas Akhir ini dapat bermanfaat bagi penulis khususnya dan bagi semua yang membutuhkan. Akhir kata, semoga Allah SWT selalu melimpahkan karunia rahmat serta hidayah-Nya kepada kita semua. Akhirul kata, Wabillahi Taufik Walhidayah

Wassalamu'alaikum Warahmatullahi Wabarakaatuh

Yogyakarta, 07 November 2020

(Adhelia Nurfira Rachmi)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, (07 November 2020)

(Adhelia Nurfira Rachmi)

ABSTRAK

IMPLEMENTASI METODE *RANDOM FOREST* DAN *XGBOOST* PADA KLASIFIKASI *CUSTOMER CHURN*

Adhelia Nurfira Rachmi

Program Studi Statistika, Fakultas MIPA

Universitas Islam Indonesia

Teknologi komunikasi yang terus berkembang pesat mengakibatkan masyarakat konsumtif akan informasi dan komunikasi. Hal ini dimanfaatkan oleh penyedia jasa layanan telekomunikasi yang terus berinovasi untuk mempertahankan *customer*. Karena terbukanya persaingan antara penyedia jasa layanan telekomunikasi dapat mengakibatkan *customer churn*. Prediksi *churn* dapat dilakukan untuk mengidentifikasi *customer churn* sejak awal dan membantu sektor CRM (*Customer Relationship Management*) agar dapat mempertahankan *customer*, sehingga mengurangi potensi kerugian yang dialami perusahaan. Penelitian ini bertujuan untuk memprediksi *customer churn* dengan data data *telecommunications churn* pada *District of Columbia* menggunakan metode klasifikasi *Random Forest* dan *Extreme Gradient Boosting*, kedua metode ini merupakan bagian dari metode *ensemble*. Hasil analisis klasifikasi bahwa metode *Extreme Gradient Boosting* (XGBoost) lebih unggul dibandingkan metode *Random Forest* dilihat dari tingkat akurasi dan nilai AUC. Metode *Extreme Gradient Boosting* (XGBoost) mendapatkan nilai akurasi dan AUC sebesar 95.6% dan 0.876, sedangkan metode *Random Forest* mampu menghasilkan nilai akurasi dan AUC sebesar 93.5% dan 0.799.

Kata Kunci: *Customer Churn*, Klasifikasi, *Ensemble*, *Random Forest*, *Extreme Gradient Boosting*

ABSTRACT

IMPLEMENTATION OF RANDOM FOREST AND XGBOOST METHODS TO CLASSIFICATION OF CUSTOMER CHURN

Adhelia Nurfira Rachmi

Department of Statistics, Faculty of Mathematics and Natural Sciences

Islamic University of Indonesia

Communication technology is expanding exponentially which affects consumptive society in information and communication. It is exploited by the telecommunication service provider that keeps innovating in maintain the customer. Due to, the competition between the telecommunication service provider, it can affect the customer churn. Churn prediction able to identify the customer churn since the beginning and help the CRM (customer relationship management) sector, to maintain the customer as of reducing the company losses potential. This research intends to predict the customer churn with telecommunication churn data in the district of Columbia using classification random forest and extreme gradient boosting method. both of these methods are part of the ensemble method. The result of classification analysis is extreme gradient boosting (XGBoost) method is superior to the random forest method; it can be seen from the accuracy and AUC value. The extreme gradient boosting (XGboost) method gets the accuracy and the AUC 95.6% and 0.876, while the random forest method affords the accuracy and the AUC 93.5% and 0.799.

Keywords**:** *Customer Churn*, *Classification*, *Ensemble*, *Random Forest*, *Extreme Gradient Boosting*

BAB 1 PENDAHULUAN

1.1 Latar Belakang Masalah

Teknologi dari tahun ke tahun terus berkembang pesat, mengakibatkan kebutuhan akan informasi dan komunikasi semakin bertambah. Manusia dimudahkan dalam menyelesaikan masalah informasi dan komunikasi dengan menggunakan berbagai perangkat teknologi yang ada. Saat ini penyebaran informasi tidak hanya melalui media cetak, media elektronik dan media sosial merupakan media komunikasi paling populer dan paling banyak diakses.

Menurut data yang dihimpun dari survei *Wearesocial & Hootsuite* 2019, total populasi yang ada di dunia sebanyak 7,676 milyar, dengan total pengguna *unique mobile* sebanyak 5,112 milyar, pengguna internet sebanyak 4,388 milyar, pengguna media sosial aktif sebanyak 3,484 milyar, dan pengguna media sosial *mobile* sebanyak 3,256 milyar. Dengan hasil *survey* yang telah dipaparkan menunjukkan perkembangan komunikasi yang membuat masyarakat konsumtif terhadap alat telekomunikasi.

Banyak perusahaan yang bergerak pada industri telekomunikasi. Hal ini membuat perusahaan telekomunikasi terus melakukan berbagai inovasi untuk mendorong persaingan usaha yang sangat ketat. Terbukanya persaingan ini menjadi tantangan sangat penting bagi perusahaan (Hashmi et al., 2013). Dengan komunikasi yang mudah dan banyaknya perusahaan telekomunikasi membuat pelanggan berhak menentukan berlangganan pada penyedia jasa yang ingin digunakan, Selain itu, mengubah nomer ponsel bukan suatu halangan. yang mengakibatkan pelanggan pada suatu perusahaan telekomunikasi lebih mudah untuk berpindah ke perusahaan telekomunikasi lain atau bisa disebut *Customer Churn*.

Customer Churn atau yang dikenal juga dengan pindahnya pelanggan adalah pemutusan jasa suatu perusahaan oleh pelanggan karena pelanggan tersebut memilih menggunakan jasa layanan lain (Masarifoglu & Buyuklu, 2019). Hal ini menjadi perhatian utama pada sektor CRM (*Customer Relationship Management*) (Govindaraju et al., 2009). Dalam persaingan pasar saat ini

mayoritas pelanggan menginginkan produk yang sesuai kebutuhan, dan mendapatkan pelayanan yang lebih baik dengan harga yang lebih murah.

Dilihat dari pengalaman bahwa penyedia jasa layanan telekomunikasi mengalami 30-50% laju *customer churn* setiap tahunnya dan membutuhkan biaya 5-10 kali usaha dan biaya untuk menambah *customer* baru daripada mempertahankan yang sudah ada. *Customer Churn* juga dapat mengurangi keuntungan atau pendapatan perusahaan.(Govindaraju et al., 2009). Oleh karena itu penyedia jasa layanan telekomunikasi lebih memilih untuk mempertahankan *customer* daripada mendapatkan *customer* baru (Lu & Ph, 2002). Dalam hal mempertahankan pelanggan, perusahaan dapat meningkatkan layanan pelanggan, memperbaiki kualitas produk, dan dapat mengetahui sejak awal pelanggan mana yang kemungkinan akan meninggalkan perusahaan tersebut.

Dengan menggunakan prediksi *churn* dapat mengidentifikasi *customer churn* sejak awal sebelum mereka berpindah ke perusahaan lain, hal ini juga dapat membantu sektor CRM (*Customer Relationship Management*) agar dapat mempertahankan *customer*, sehingga mengurangi potensi kerugian yang dialami perusahaan. Dibutuhkan suatu metode *Data Mining* yaitu Klasifikasi untuk melakukan prediksi dengan jumlah dan bentuk data yang besar. Salah satu metode dalam klasifikasi adalah *XGBoost*.

XGBoost (*Extream Gradient Boosting*) merupakan pengembangan dari algoritma *Gradient Tree Boosting* yang berbasis algoritma *ensemble*, secara efisien dapat menangani permasalahan *machine learning* yang berskala besar. *XGBoost* dapat menyelesaikan berbagai fungsi klasifikasi, regresi, dan ranking. Metode *XGBoost* juga sukses menjadi salah satu metode yang banyak digunakan dalam berbagai kasus pada *machine learning*. *Higgs Boson Competition* pertama kali *XGBoost* diperkenalkan. Pada akhir kompetisi, metode yang paling banyak digunakan oleh sebagian besar tim yang mengikuti kompetisi yaitu *XGBoost*. Selain itu pada situs *Kaggle* selama tahun 2015, dalam kompetensi *machine learning* sebanyak 17 dari 29 *winning solution* menggunakan algoritma *XGBoost*.

Pada penelitian ini peneliti memilih metode *XGBoost* karena mengacu pada penelitian terdahulu yang dilakukan oleh (Pamina et al., 2019) metode *XGBoost*

lebih baik dibandingkan dengan metode lainnya seperti K-NN dan *Random Forest*, dilihat dari tingkat akurasi sebesar 79,8% dan nilai *F-Score* sebesar 58,2%. Peneliti juga ingin membandingkan metode *XGBoost* dengan metode *Random Forest* mengacu pada penelitian lainnya yaitu (S et al., 2019) melakukan pengklasifikasian terhadap data sekuens DNA dengan menggunakan metode *XGBoost* dan *Random Forest*.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang yang telah dipaparkan pada poin 1.1, maka terdapat beberapa rumusan masalah yaitu:

1. Bagaimana karakteristik *customer churn* berdasarkan metode klasifikasi *Random Forest* dan *XGBoost?*
2. Bagaimana hasil perbandingan metode klasifikasi *Random Forest* dan *XGBoost* dengan pengukuran tingkat ketepatan Akurasi dan *AUC*?

1.3 Batasan Masalah

Berikut ini batasan masalah yang digunakan dalam penelitian ini:

1. Data yang digunakan pada Tugas Akhir ini yaitu berupa data sekunder yang didapatkan dari situs dan *platform* kompetisi *Kaggle*. Data tersebut merupakan data *Telecommunication customer churn* pada suatu perusahaan telekomunikasi.
2. Metode Statistik yang digunakan adalah klasifikasi *Random Forest dan Extreme Gradient Boosting* (*XGBoost*).
3. *Software* yang digunakan untuk membantu penelitian adalah *Python* dan *Microsoft Excel*.

1.4 Tujuan Penelitian

Berikut ini merupakan tujuan dari penelitian yang akan dilakukan:

1. Mengetahui karakteristik *customer churn* berdasarkan metode klasifikasi *Random Forest* dan *XGBoost*.
2. Mengetahui hasil perbandingan metode klasifikasi *Random Forest* dan *XGBoost* dengan pengukuran tingkat ketepatan Akurasi dan *AUC*.

1.5 Manfaat Penelitian

Manfaat yang didapatkan dari penelitian ini yaitu, dengan dilakukannya prediksi *customer churn* menggunakan metode *Random Forest* dan *Extreme Gradient Boosting* (*XGBoost*), maka penulis dapat mengetahui implementasi klasifikasi dari kedua metode ini. Penelitian ini juga dapat digunakan sebagai bahan pertimbangan dalam dunia pelayanan jasa telekomunikasi, dilihat dari hasil metode terbaik yang didapatkan. Selain itu, penelitian ini dapat menjadi referensi penelitian selanjutnya terkait dengan prediksi *Customer Churn* maupun tentang penerapan metode *Random Forest* dan *XGBoost*.

BAB 2 TINJAUAN PUSTAKA

Tinjauan Pustaka ini dilakukan dengan tujuan menghindari terjadinya duplikasi penelitian dengan penelitian sebelumnya. Selain itu tinjauan pustaka ini dilakukan sebagai kajian litetatur untuk mengetahui keterkaitan antara penelitian yang sudah pernah dilakukan dengan penelitian yang akan dilakukan oleh peneliti. Berikut ini merupakan beberapa penelitian terdahulu yang dijadikan sebagai acuan oleh peneliti.

Penelitian pertama dengan judul "*A Hybrid Prediction Model for E-Commerce Customer Churn Based on Logistic Regression and Extreme Gradient Boosting Algorithm*" pada tahun 2019. Penelitian ini dilakukan oleh (Li & Li, 2019) dengan tujuan penelitian membandingkan metode klasifikasi Regresi Logistik dengan *Extreme Gradient Boosting* untuk memprediksi *customer churn* pada *platform e-commerce*. Digunakan 25 indeks yang berkaitan dengan *customer churn*, pada metode Regresi Logistik didapatkan tingkat akurasi mencapai 75,9%, *precision* 75,7%, dan *recall* 83,6%. Untuk metode *Extream Gradient Boosting* didapatkan hasil tingkat akurasi 76,6%, *precision* 76,3%, dan *recall* 84,2%. Hasil penelitian ini menunjukkan bahwa dari 25 indeks yang digunakan, dipilih 10 indeks penting untuk memprediksi *customer* churn dan metode *Extreme Gradient Boosting* lebih akurat daripada metode Regresi Logistik.

Penelitian kedua dengan judul "*Accurate Liver Disease Prediction with Extreme Gradient Boosting*" pada tahun 2019. Penelitian ini dilakukan oleh (Murty & Kumar, 2019) dengan tujuan penelitian untuk memprediksi penyakit hati menggunakan *Extreme Gradient Boosting*, dengan menyetel model *XGBoost* menggunakan *important hyperparameter* dan membandingkan metode *XGBoost* hasil *tuning hyperparameter* dengan beberapa metode klasifikasi antara lain Naïve Bayes dengan tingkat akurasi 71%, metode C4.5 dengan tingkat akurasi 97%, metode AD Tree tingkat akurasi 92%, metode SVM tingkat akurasi 75%, RBF *Network* tingkat akurasi 8%, *Multi-Layer Feed Forward Deep Neural Network* (MLFDNN) dengan tingkat akurasi 98%, dan metode *Extreme Gradient Boosting* (*XGBoost*) dengan tingkat akurasi 99%. Menggunakan metode *XGBoost* dengan

melakukan optimasi *tuning hyperparameter* didapatkan tingkat akurasi lebih unggul dibandingkan metode klasifikasi lainnya.

Penelitian ketiga yaitu penelitian yang berjudul "*An Effective Classifier for Predicting Churn in Telecommunication*" pada tahun 2019. Penelitian ini dilakukan oleh (Pamina et al., 2019) dengan tujuan penelitian membandingkan tiga metode klasifikasi yaitu *K-Nearest Neighbour* (K-NN), *Random Forest*, dan *Extreme Gradient Boosting* untuk memprediksi *customer churn* pada perusahaan telekomunikasi. Hasil penelitian dari ketiga metode klasifikasi didapatkan tingkat akurasi untuk K-NN senilai 75,4%, metode *Random Forest* dengan tingkat akurasi senilai 77,5%, dan metode *Extreme Gradient Boosting* (*XGBoost*) dengan tingkat akurasi 79,8%. Dari Ketika metode klasifikasi ini metode *XGBoost* lebih unggul.

Penelitian keempat yaitu penelitian yang berjudul "*Comparation Analysis of Ensemble Technique with Boosting(XGBoost) and Bagging (Random Forest) for Classify Splice Junction DNA Sequence Category*" pada tahun 2019. Penelitian ini dilakukan oleh (S et al., 2019) dengan tujuan penelitian mengklasifikasikan data sekuens DNA menggunakan metode *XGBoost* dan *Random Forest* sebagai representasi dari metode *boosting* dan *bagging*, penelitian ini juga melakukan optimasi dengan *tuning hyperparamter* secara *gridsearch* menggunakan pola tertentu, setelah melakukan *tuning hyperparameter* pada kedua metode klasifikasi *XGBoost* dan *Random Forest*, kemudian dilakukan klasifikasi pada data ekstraksi ciri frekuensi *k-means*. Dari penelitian ini didapatkan hasil penelitian bahwa metode *ensemble* secara *boosting* dan *bagging* mampu menangani klasifikasi dengan tingkat akurasi yang baik, kemudian dengan optimasi *tuning hyperparameter* yang tepat terbukti mampu meningkatkan nilai akurasi. Metode *XGBoost* mampu mencapai tingkat akurasi lebih baik dibandingkan *random forest*, baik itu sebelum *tuning hyperparameter* maupun setelah dilakukan *tuning hyperparamter*. Dengan tingkat akurasi *XGBoost* senilai 95,6% sebelum *tuning hyperparameter* dan 96,2% setelah *tuning hyperparameter*.

Penelitian kelima dengan judul "*Implementing Extreme Gradient Boosting (XGBoost) Classifier to Improve Customer Churn Prediction*" pada tahun 2019.

Penelitian ini dilakukan oleh (Hanif, 2019) dengan tujuan penelitian membuktikan atau menyangkal apakah metode *XGBoost* dapat memprediksi lebih baik dibandingkan dengan metode Regresi Logistik. Penelitian ini dilakukan dengan menggunakan sampel data pelanggan (pelanggan tetap maupun yang berpindah) tercatat selama 6 bulan dari bulan Oktober 2017 hingga Maret 2018, penelitian ini juga menerapkan *tuning hyperparameter* pada metode Regresi Logistik Maupun metode *XGBoost*. Hasil yang didapatkan dari penelitian ini bahwa metode *XGBoost* memiliki tingkat akurasi senilai 97,8% dan metode Regresi Logistik senilai 90,7%, untuk nilai kurva ROC-AUC metode *XGBoost* lebih unggul dengan nilai (0,99) dibandingkan metode Regresi Logistik dengan nilai kurva ROC-AUC (0,81). Maka dapat dikatakan bahwa metode *XGBoost* telah terbukti dapat memprediksi lebih baik dibandingkan dengan metode Regresi Logistik dilihat dari akurasi, spesifisitas, sensitivitas, maupun kurva ROC, metode *XGBoost* dapat menangani ketidakseimbangan data.

Penelitian keenam dengan judul "*Performance Evaluation of Machine Learning Approaches for Credit Scoring*" pada tahun 2018. Penelitian ini dilakukan oleh (Cao et al., 2018) yang mempunyai tujuan untuk menerapkan teknik *data mining* dalam memperoleh bukti yang digunakan untuk menilai metode klasifikasi mana yang bekerja lebih baik dalam menilai *Credit Scoring* pada model yang didapatkan. Penelitian ini menggunakan dua dataset, delapan metode klasifikasi diantaranya *Linear Discriminant Analysis* (LDA), *Logistic Regression* (LR), *Decision Tree* (DT), *Support Vector Machine* (SVM), *Random Forest* (RF), *Gradient Boosting Decision Tree* (GBDT), *Extreme Gradient Boosting* (XGBoost), dan *Multi-Layer Perceptron* (MLP), dipilih 3 indikator yaitu nilai akurasi, AUC, dan *Logistic Loss* untuk menganalisis kinerja dari setiap metode Klasifikasi. Hasil penelitian yang diperoleh bahwa metode *XGBoost* dengan tingkat akurasi 82% pada dataset pertama, dan 87% pada dataset kedua, lebih unggul dibandingkan dengan tujuh metode klasifikasi lainnya, begitu pula dengan nilai AUC dan *Logistic Loss* metode lebih unggul diantara tujuh metode klasifikasi yang digunakan.

Penelitian ketujuh dengan judul "*Machine-Learning Techniques for Customer Retention: A Comparative Study*" pada tahun 2018. Penelitian ini dilakukan oleh (Sabbeh, 2018), dengan tujuan untuk membandingkan dan menganalisis kinerja berbagai teknik *machine learning*, dengan menggunakan sepuluh teknik *machine learning* antara lain *Linear Discriminant Analysis* (LDA), *Decision Tree* (CART), *K-Nearest Neighbors* (KNN), *Support Vector Machines*, *Logistic Regression*, *Ensemble-based learning techniques* (*Random Forest*, *AdaBoost* , dan *Stochastic Gradient Boosting*), *Naïve Bayesian*, dan *Multi-Layer Perceptron*. Model yang diterapkan pada dataset telekomunikasi yang berisi 3,333 *records*. Hasil yang diperoleh dari penelitian ini adalah metode *Random Forest* dan *AdaBoost* unggul dari metode yang lainnya dengan akurasi yang hampir sama yaitu 96%. Untuk metode *Multi-Layer Perceptron* dan *Support Vector Machine* mendapatkan nilai akurasi sebesar 94%. Untuk metode *Decision Tree* mendapatkan nilai akurasi 90%, *Naïve Bayesian* 88%, dan metode *Logistic Regression* dan *Linear Discriminant Analysis* (LDA) degan nilai akurasi 86.7%.

Penelitian kedelapan dengan judul "Perbandingan Metode Random Forest dan Naïve Bayes untuk Klasifikasi Debitur Berdasarkan Kualitas Kredit" pada tahun 2019. Penelitian ini dilakukan oleh (Bawono & Wasono, 2019), dengan tujuan penelitian melakukan perbandingan terhadap metode *Random Forest* dan *Naïve Bayes* untuk klasifikasi debitur berdasarkan kualitas kredit dengan membandingkan nilai akurasi. Dataset yang digunakan adalah data sekudenr yang diperoleh dari salah satu Bank, dengan jumlah variabel 10. Hasil perbandingan klasifikasi dengan *Random Forest* dan *Naive Bayes* pada studi kasus klasifikasi debitur berdasarkan kualitas kredit menyatakan bahwa klasifikasi *Random Forest* merupakan metode klasifikasi yang memiliki tingkat akurasi paling tinggi untuk klasifikasi kualitas kredit yaitu mencapai 98,16% disusul *Naïve Bayes* 95,93%.

Tabel 2. 1 Perbedaan Penelitian ini dengan Penelitian Sebelumnya

Nama Peneliti	Judul Penelitian	Tahun	Hasil
Xueling Li & Zhen Li	*A Hybrid Prediction Model for E-Commerce*	2019	Penelitian ini menggunakan algoritma Regresi Logistik dan *Extreme Gradient*

Nama Peneliti	Judul Penelitian	Tahun	Hasil
	Customer Churn Based on Logistic Regression and Extreme Gradient Boosting Algorithm		*Boosting* (XGBoost) untuk memprediksi *customer churn* pada *platform E-commerce*. Data yang digunakan dalam penelitian ini adalah 25 indeks yang berkaitan dengan *customer churn* pada *platform* tersebut.
Sivala Vishnu Murty & R Kiran Kumar	*Accurate Liver Disease Prediction with Extreme Gradient Boosting*	2019	Penelitian ini bertujuan untuk memprediksi penyakit hati mengunakan algoritma *Extreme Gradient Boosting* dengan pengaturan *important hyperparameter*. Algoritma klasifikasi lain yang digunakan antara lain Naïve Bayes, metode C4.5, metode AD Tree, RBF *Network*, dan *Multi-Layer Feed Forward Deep Neural Network* (MLFDNN). Data yang digunakan pada penelitian ini adalah data penyakit hati berasal dari Rumah Sakit Amrutha Group, Srikakulam, dan Andhra Pradesh di Negara India.
J. Pamina, J. Beschi Raja, S. Sathya Bama, S. Soundarya, M.S. Sruthi, S. Kiruthika, V.J.	*An Effective Classifier for Predicting Churn in Telecommunication*	2019	Penelitian yang dilakukan ini bertujuan untuk membandingkan tiga metode klasifikasi yaitu *K-Nearest Neighbour* (K-NN), *Random*

Nama Peneliti	Judul Penelitian	Tahun	Hasil
Aiswaryadevi & G. Priyank			*Forest*, dan *Extreme Gradient Boosting* untuk memprediksi *customer churn* pada perusahaan telekomunikasi.
Iqbal Hanif	*Implementing Extreme Gradient Boosting (XGBoost) Classifier to Improve Customer Churn Prediction*	2019	Tujuan dari penelitian ini adalah ingin membuktikan atau menyangkal apakah metode *XGBoost* dapat memprediksi lebih baik dibandingkan dengan metode Regresi Logistik. Dengan menggunakan data yang berasal dari perusahaan Telkom Indonesia dari bulan Oktober 2017 hingga Maret 2018
Sahar F.Sabbeh	*Machine-Learning Techniques for Customer Retention: A Comparative Study*	2018	Tujuan dari penelitian ini adalah membandingkan dan menganalisis kinerja berbagai teknik *machine learning*, dengan menggunakan sepuluh metode antara lain. *Linear Discriminant Analysis* (LDA), *Decision Tree* (CART), *K-Nearest Neighbors* (KNN), *Support Vector Machines*, *Logistic Regression*, *Ensemble-based learning techniques* (*Random Forest*, *AdaBoost* , dan *Stochastic Gradient*

Nama Peneliti	Judul Penelitian	Tahun	Hasil
			Boosting), *Naïve Bayesian*, dan *Multi-Layer Perceptron*. Dengan menggunakan dataset perusahaan telekomunikasi.
Bonggo Bawono & Rochdi Wasono	Perbandingan Metode *Random Forest* dan *Naïve Bayes* untuk Klasifikasi Debitur Berdasarkan Kualitas Kredit	2019	Penelitian ini bertujuan untuk membandingkan nilai akurasi dua metode klasifikasi yaitu metode *Random Forest* dan *Naïve Bayes*. Didapatkan hasil metode *Random Forest* memiliki hasil akurasi tertinggi yakni sebesar 98,16% disusul *Naïve Bayes* 95,93%.
Penelitian yang penulis lakukan			
Adhelia Nurfira Rachmi	Implementasi Metode Random Forest Dan Xgboost Pada Klasifikasi Customer Churn	2020	Perbedaan penelitian ini dengan beberapa penelitian sebelumnya yakni terletak pada metode yang digunakan pada penelitian ini peneliti menggunakan teknik *ensemble* metode *Random Forest* dan *XGBoost* dengan menggunakan teknik optimasi parameter menggunakan metode *Grid Search CV*. Dataset yang digunakan yaitu data telekomunikasi pada salah satu perusahaan. Kemudian hasil dari kedua metode

Nama Peneliti	Judul Penelitian	Tahun	Hasil
			tersebut dibandingkan untuk mengetahui metode yang paling baik dalam memprediksi *Customer Churn*.

Pada penelitian terdahulu yang dipaparkan sebelumnya, terdapat perbedaan dan persamaan dengan penelitian yang akan dilakukan. Hal yang membedakan penelitian ini dengan penelitian sebelumnya yaitu dengan menggunakan teknik *ensemble* metode *Random Forest* dan *XGBoost* dengan menggunakan teknik optimasi parameter menggunakan metode *Grid Search CV*. Dataset yang digunakan yaitu data telekomunikasi pada salah satu perusahaan. Kemudian hasil dari kedua metode tersebut dibandingkan untuk mengetahui metode yang paling baik dalam memprediksi *Customer Churn*.

BAB 3 LANDASAN TEORI

3.1 Telekomunikasi

Berdasarkan pasal 1 Undang-undang telekomunikasi Nomor 36 tahun 1999 Bab I, telekomunikasi adalah setiap pemancaran, pengiriman, dan atau penerimaan dari setiap informasi dalam bentuk tanda-tanda, isyarat, tulisan, gambar, suara dan bunyi melalui system kawat, optik, radio, atau system elektromagnetik lainnya.

3.2 *Customer Churn*

'*Churn*' adalah kata yang berasal dari perubahan dan putaran atau dapat diartikan dengan penghentian kontrak (Lazarov & Capota, 2007). *Customer churn* merupakan perubahan atau pergeseran pelanggan dari satu penyedia layanan ke pesaing lainnya di pasar penyedia layanan, hal ini merupakan tantangan utama di pasar penyedia layanan yang kompetitif dan banyak diamati pada sektor telekomunikasi. *Customer churn* adalah pelanggan yang ditargetkan telah memutuskan untuk meninggalkan penyedia jasa, produk, bahkan perusahaan dan beralih ke pesaing dipasaran (Amin et al., 2016).

Menurut (Amin et al., 2016) jenis *customer churn* terbagi menjadi tiga yaitu:

1. *Active churner* (sukarela) merupakan pelanggan yang ini berhenti dari kontrak dan berpindah ke penyedia layanan berikutnya.
2. *Passive churner* (non sukarela) yaitu ketika sebuah perusahaan menghentikan layanan kepada pelanggan.
3. *Rotational churner* (diam) merupakan penghentian kontrak tanpa sepengetahuan kedua belah pihak (pelanggan dan perusahaan), dimana masing-masing pihak tiba-tiba dapat mengakhiri kontrak tanpa pemberitahuan.

Beberapa dampak pada kinerja perusahaan disebabkan perilaku *customer churn* antara lain:

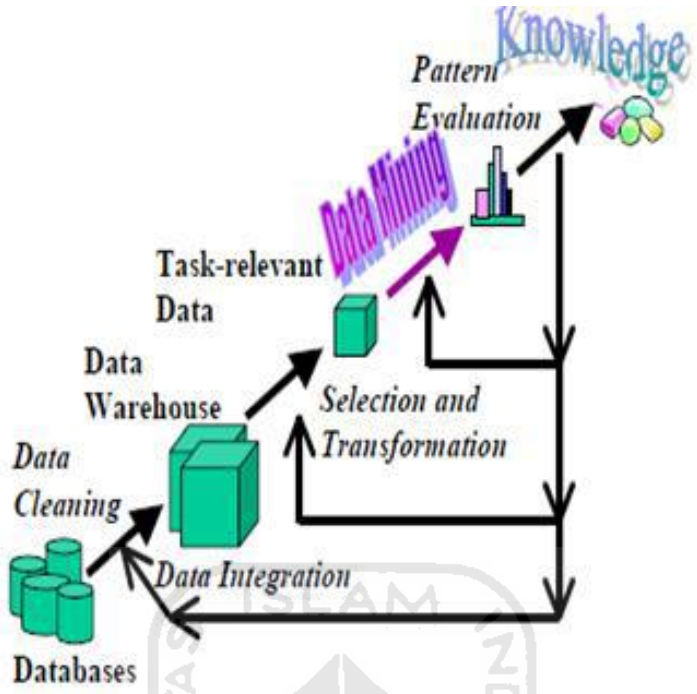
1. Dampak negatif terhadap kinerja keseluruhan perusahaan.

2. Penyebab potensial untuk penjualan rendah karena pelanggan baru atau jangka pendek membeli layanan lebih sedikit.
3. Membantu pesaing untuk mendapatkan pelanggan yang tidak puas dengan promosi bisnis
4. Menyebabkan kerugian pendapatan.
5. Berdampak negatif pada pelanggan jangka panjang
6. Meningkatkan ketidakpastian yang mengurangi kemungkinan pelanggan baru.
7. Menarik pelanggan baru lebih mahal daripada mempertahankan posisi
8. Risiko terhadap citra perusahaan di pasar yang kompetitif dengan hilangnya basis pelanggan.

3.3 *Data Mining*

Data mining adalah penemuan informasi yang tidak diketahui dari database. Mengumpulkan data, mendapatkan data, dan menganalisis data mengacu pada penggunaan metode *data mining* untuk mengambil sampel dari kumpulan data dengan populasi yang besar untuk dibuat kesimpulan statistik yang digunakan untuk memvalidasi pola yang ditemukan (Thankachan & Suchithra, 2017). *Data mining* digunakan untuk mengarahkan seluruh proses yang dilakukan untuk menggabungkan metodologi dan teknik dari berbagai bidang, seperti statistic, *database*, *machine learning*, dan visualisasi(Zanin et al., 2016). *Data mining* juga dikenal sebagai *Knowledge Discovery in Database* (KDD) yang merujuk pada ekstraksiinformasi implisit, yang tidak diketahui sebelumnya dapat berpotensi bermanfaat dari data di dalam *database*.

Data mining dalam proses *Knowledge Discovery in Database* (KDD) dilakukan dengan langkah (Sahu et al., 2008):



Gambar 3. 1 *Data Mining* pada Proses *Knowledge Discovery in Database* (Sahu et al., 2008)

3.4 *Machine Learning*

Machine learning adalah bidang ilmu komputer dimana berevolusi dari mempelajari pola dan teori pembelajaran komputasi dalam *Artificial Intelligence*. *Machine Learning* merupakan proses pembelajaran dan membangun algoritma yang dapat belajar dan membuat prediksi pada dataset (Simon et al., 2015). *Machine learning* berhubungan erat dengan bidang ilmu statistik dan data, yang menyediakan berbagai alat dan metode yang digunakan untuk analisis dan menarik kesimpulan dari data. *Machine learning* juga berkaitan dengan robotika dan kecerdasan otomatisasi. Berbagai bidang yang berkaitan dengan *machine learning* membantu membentuk konteks dimana orang-orang dapat berhubungan dengan banyak aplikasi *machine learning*, dan menginformasikan peluang serta tantangan yang terkait. *Machine learning* juga mendukung kemajuan dalam bidang ini, sebagai teknologi yang mendasari *Artificial Intelligence* dan ilmu data. (The Royal Society, 2017).

Terdapat tiga cabang utama dalam *machine learning* menurut (The Royal Society, 2017) diantaranya:

1. *Suvervised machine learning*

   Dalam pengawasan *machine learning*, suatu sistem dilatih dengan data yang telah diberi label, kemudian label tersebut mengkategorikan setiap titik data dalam satu atau beberapa kelompok. Sistem mempelajari bagaimana data tersebut dikenal sebagai data *training* terstruktur, dan menggunakan data *training* tersebut untuk memprediksi data *test* atau data uji.

2. *Unsupervised learning*

   Pembelajaran tanpa pengawasan artinya pembelajaran tanpa label. Hal ini bertujuan untuk mendeteksi karakteristik yang membuat titik suatu data kurang lebih mirip satu sama lain, misalnya dengan membuat *cluster* dan menetapkan data pada *cluster* tersebut.

3. *Reinforcement learning*

   Pembelajaran untuk memperkuat belajar dari pengalaman, yang berada antara *supervised* dan *unsupervised learning*. Dalam pengaturan *reinforcement learning*, suatu agen berinteraksi dengan lingkungannya, kemudian diberi fungsi imbalan yang digunakan untuk mencoba mengoptimalkan, sebagai contoh suatu sistem kemungkinan dihargai untuk memenangkan permainan, agen memiliki tujuan untuk mempelajari konsekuensi keputusannya, seperti gerakan yang cukup penting dalam memenangkan permainan, dan menggunakan pembelajaran ini untuk menemukan strategi yang dapat memaksimalkan keuntungannya.

3.5 Klasifikasi

Klasifikasi adalah suatu bentuk analisis data dalam menentukan model atau fungsi yang membedakan atau menggambarkan suatu konsep atau kelas data (Han et al., 2011). Tujuan dari klasifikasi yaitu memprediksi secara akurat kategori pada data yang tidak diketahui dalam setiap kasus. Algoritma klasifikasi dapat

diterapkan untuk data kategorikal, apabila target data numerik, model prediksi yang digunakan adalah algoritma regresi (Sumathi et al., 2016).

Menurut (Gupta et al., 2015) terdapat langkah-langkah dalam proses klasifikasi yaitu:

1. Langlah pertama membangun model dari data *training* dengan nilai label kelas diketahui. Algoritma klasifikasi digunakan untuk membuat model dari dataset *training*.
2. Langkah kedua melakukan pemeriksaan akurasi model dari data *train*, jika kebenaran model memuaskan maka model digunakan untuk mengklasifikasikan data dengan label kelas yang tidak diketahui.

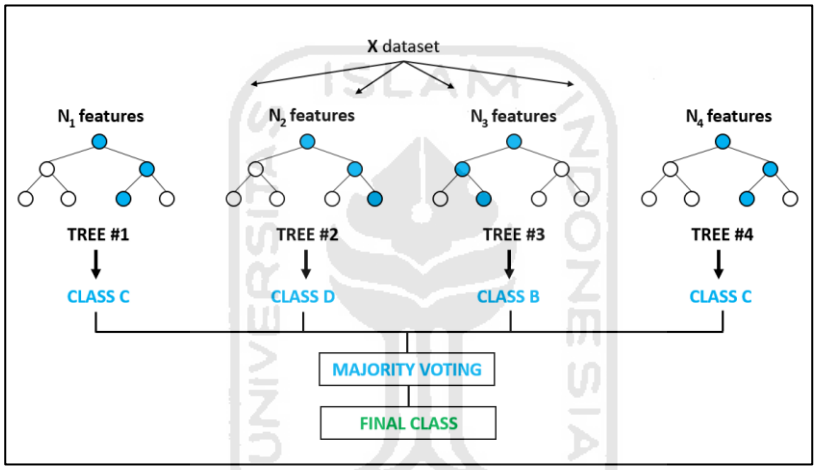
3.6 *Bagging*

Metode *Bagging* pertama kali diperkenalkan oleh Leo Breiman untuk mengurangi perbedaan prediktor. Metode *Bagging* secara berurutan menggabungkan pembelajaran yang lemah untuk mengurangi kesalahan prediksi (Wu et al., 2018). *Bagging* atau *boostrap aggregation* merupakan salah satu metode *ensemble* yang paling sederhana namun sukses dalam meningkatkan masalah klasifikasi yang tidak stabil. Metode *Bagging* biasanya diterapkan pada algoritma *Decision Tree*, dapat juga diaplikasikan pada algoriman klasifikasi lain seperti *Random Forest*, *Naïve Bayes*, *Nearest Neighbour*, dan lain sebagainya. Metode *Bagging* sangat berguna untuk data besar dan berdimensi tinggi (Syarif et al., 2012).

Metode *Bagging* melatih sejumlah pembelajaran dasar dari setiap sampel boostrap yang berbeda dengan memanggil algoritma pembelajaran dasar. Sampel boostrap diperoleh dari subsample yang sama dengan ukuran kumpulan data *training*. Untuk sampel boostrap, beberapa contoh *training* kemungkinan muncul namun beberapa kemungkinan tidak, dimana probabilitas bahwa contoh muncul sekali setidaknya sekitar 0,632. Setelah mendapatkan pelajaran dasar, metode *Bagging* akan menggabungkan seluruh percobaan dengan suara terbanyak dan kelas yang paling banyak diprediksi (Tuv, 2006).

3.6.1 *Random Forest*

Random Forest adalah salah satu dari sekian banyak metode *Ensemble* yang dikembangkan oleh Leo Breiman pada tahun 2001. *Random Forest* merupakan pengembangan metode *Classification and Regression Tree* (CART) dengan menerapkan metode *Boostrap Aggregating* (*Bagging*), dan *Random Feature Selection*. Metode *Random Forest* sendiri memiliki beberapa kelebihan antara lain, menghasilkan hasil klasifikasi yang baik, menghasilkan*error* yang lebih rendah, secara efisien dapat mengatasi data *training* dengan jumlah data yang sangat besar (Breiman, 2001).

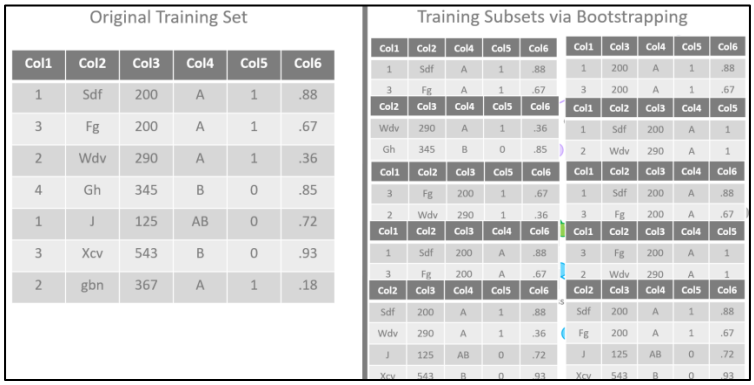


Gambar 3. 2 Skema Kinerja Algoritma *Random Forest*
Sumber : (medium.com)

Metode *Random Forest* menghasilkan satu set pohon acak. Kelas yang dihasilkan berasal dari proses klasifikasi yang dipilih dari kelas yang paling banyak (modus) yang dihasilkan oleh pohon keputusan yang ada (Biau, 2012). Algoritma atau prosedur dalam membangun *Random Forest* pada gugus data yang terdiri dari *n* amatan dan terdiri atas *p* peubah penjelas (*predictor*), berikut dibawah merupakan tahapan penyusunan dan pendugaan menggunakan *Random Forest* (Breiman & Cutler, 2003) :

1. Tahapan *Bootstrapping*

   Untuk mulai membangun *Random Forest* langkah pertama yang dilakukan adalah pengambilan sampel secara acak berukuran *n* dari kumpulan data asli dengan pengembalian.



Gambar 3. 3 Contoh Pengambilan Sampel dengan *Bootstrapping*
Sumber: (dataversity.net)

2. Tahapan *Random Feature Selection*

Pada tahapan ini pohon dibangun hingga mencapai ukuran maksimum (tanpa pemangkasan). Pada proses pemilah pemilih variabel prediktor *m* dipilih secara acak, dimana *m*<<*p*, kemudian pemilah terbaik dipilih berdasarkan *m* prediktor. Berikut dibawah ini merupakan contoh dalam membangun *Decision Tree* (Sidhu, 2019):

Dalam menentukan pohon keputusan langkah awal yang dilakukan yaitu menghitung nilai *entropy* sebagai penentu tingkat ketidakmurnian atribut dan nilai *information gain*. Menghitung nilai *entropy* dapat menggunakan rumus seperti persamaan (3.1) untuk satu atribut, persamaan (3.2) untuk dua atribut menggunakan tabel frekuensi, dan menentukan nilai *information gain* menggunakan persamaan (3.3):

$$Entropy\ (S) = \sum_{i=1}^{C} -p_i\ log_2 p_i \tag{3.1}$$

Dimana:

- S = Himpunan dataset
- C = Jumlah kelas
- p_i = Probabilitas frekuensi kelas ke-i dalam dataset

$$Entropy\ (T, X) = \sum_{c \in X} P(c)E(c) \tag{3.2}$$

Dimana:

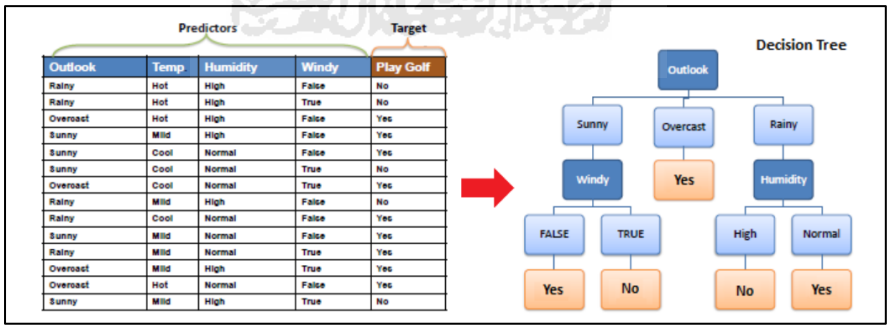
- (T,X) = Atribut T dan Atribut X
- P(c) = Probabilitas kelas atribut
- E(c) = Nilai *Entropy* kelas atribut

$$Gain\ (A) = \ Entropy\ (S) - \sum_{i=1}^{k} \frac{|S_i|}{|S|} x\ Entropy(S_i) \qquad (3.3)$$

Dimana:

- S = Himpunan dataset
- A = Atribut
- $|S_i|$ = Jumlah sampel untuk nilai i
- $|S|$ = Jumlah seluruh sampel data
- $Entropy(S_i)$ = *Entropy* untuk sampel yang memiliki nilai *i*

Diberikan contoh dataset yang digunakan dalam membentuk *decision tree* dimana dataset tersebut memiliki atribut-atribut seperti *Outlook*, *Temperature*, *Humidity*, dan *Windy*. Untuk kelas ada pada atribut *Play Golf* yaitu "No" dan "Yes".



Gambar 3. 4 Contoh Pembentukan *Decision Tree*
Sumber: (saedsayad.com)

Berikut dibawah ini langkah-langkah dalam membangun *decision tree*:

a. Langkah pertama yang dilakukan adalah menghitung nilai *Entropy* dari seluruh data, dimana terdapat 14 data dengan total "No" sebanyak 5 dan "Yes" sebanyak 9.

Tabel 3. 1 Tabel Atribut *Play Golf*

Play Golf	
Yes	*No*
9	5

$$\begin{aligned} Entropy\,(PlayGolf) &= Entropy(5,9) \\ &= Entropy\,(0.36,0.64) \\ &= -0.36\,log_2 0.36 + (-0.64 log_2 0.64) \\ &= 0.53 + 0.64 \\ &= 0.94 \end{aligned}$$

b. Setelah mendapatkan nilai *entropy* dari atribut target, langkah selanjutnya yaitu menghitung nilai *entropy* dari masing-masing *feature*. Sebagai contoh menghitung Nilai *Entropy Play Golf* dan atribut *Outlook*.

Tabel 3. 2 Tabel Frekuensi dari Dua Atribut

		Play Golf		
		Yes	No	
Outlook	*Sunny*	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

$$\begin{aligned} &Entropy\,(PlayGolf, Outlook) \\ &\quad = (P(Sunny)x\,E(3,2)) + (P(Overcast)x\,E(4,0)) \\ &\quad + (P(Rainy)x\,E(2,3)) \end{aligned}$$

$$\begin{aligned} Entropy\,(PlayGolf, Outlook) &= \left(\frac{5}{14}x\,0.971\right) + \left(\frac{4}{14}x\,0.0\right) + \left(\frac{5}{14}x\,0.917\right) \\ &= 0.693 \end{aligned}$$

Lakukan langkah pada poin (b) untuk atribut yang lain.

c. Setelah menghitung nilai *Entropy* langkah selanjutnya menghitung nilai *Information Gain* dari setiap variabel.

Sebagai contoh menghitung nilai *information gain* dari atribut *Outlook*

$$Gain\,(Outlook) \quad = E(PlayGolf) - E(PlayGolf, Outlook)$$

$$= 0.94 - 0.693$$

$$= 0.247$$

Lakukan langkah pada poin (c) untuk atribut yang lain:

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

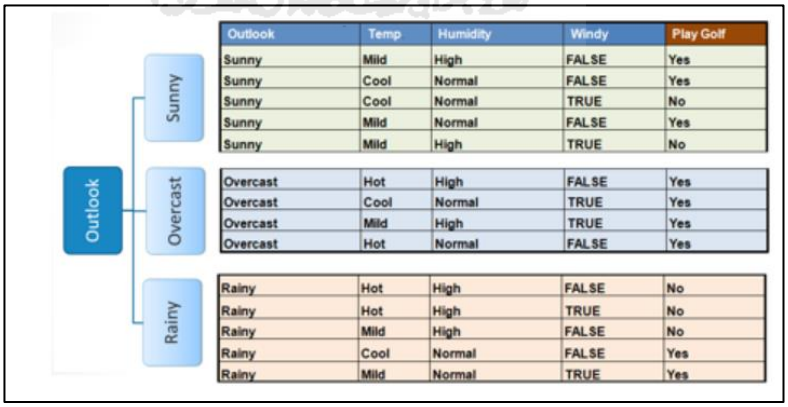
		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Gambar 3. 5 Contoh Tabel Frekuensi dan Nilai *Information Gain* dari masing-masing Atribut
Sumber: (saedsayad.com)

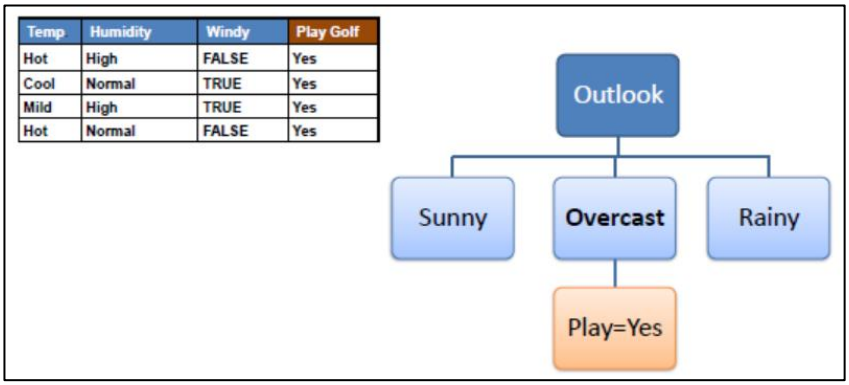
d. Pilihlah atribut dengan nilai *information gain* terbesar menjadi *root node*, bagi dataset dengan cabangnya dan ulangi proses yang sama pada setiap cabang.

   Atribut *Outlook* dipilih menjadi *root node* karena memiliki nilai *information gain* terbesar.



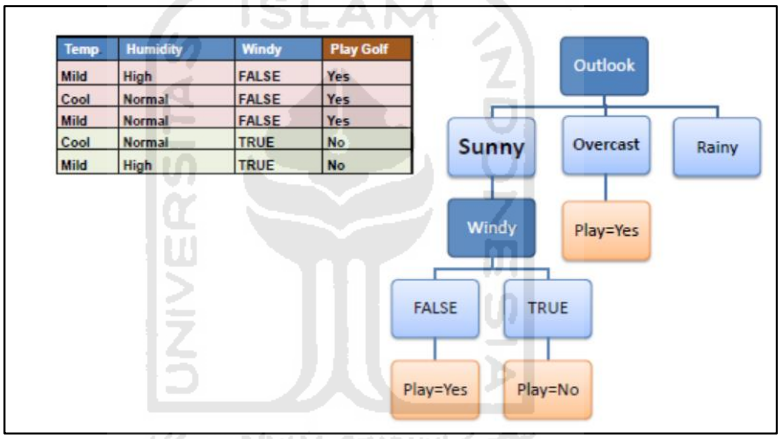
Gambar 3. 6 Contoh Pembentukan *Root Node* dan *Terminal Node*
Sumber: (saedsayad.com)

e. Selanjutnya analisis *node* hasil percabangan *root node*, cabang dengan nilai *entropy* 0 adalah *leaf node*.



Gambar 3. 7 Contoh Pembentukan *Leaf Node*
Sumber: (saedsayad.com)

f. Lakukan pemisahan cabang seperti langkah langkah sebelumnya hingga semua *node* berbentuk *leaf node*.



Gambar 3. 8 Contoh Pembentukan *Iinternal Node* dan *Leaf Node*
Sumber: (saedsayad.com)

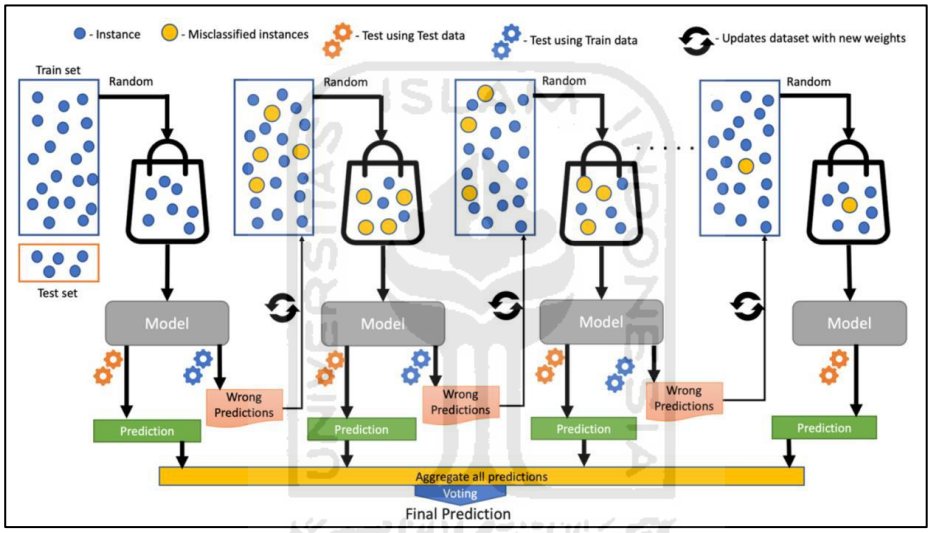
3. Ulangi langkah (1) dan (2) sebanyak *k* kali hingga diperoleh *k* buah *decision tree*.
4. Lakukan pendugaan gabungan terhadap *k decision tree* yang dibangun menggunakan *majority voting*.

3.7 *Boosting*

Boosting yang diperkenalkan oleh Robert E. Schapire pada tahun 1998 merupakan salah satu metode *ensemble* yang dapat meningkatkan kinerja dari beberapa hasil klasifikasi yang lemah agar dapat menjadi proses klasifikasi yang kuat. Teknik *Boosting* dapat dilihat sebagai metode model rata-rata yang awalnya

dirancang untuk metode klasifikasi tetapi dapat juga diaplikasikan pada metode regresi (Syarif et al., 2012).

Seperti halnya metode *bagging* yang memanfaatkan voting untuk tujuan mengklasifikasikan atau menghitung rata-rata estimasi numerik untuk *output* model individu tunggal. Persamaan lainnya adalah menyatukan model yang memiliki jenis yang sama, seperti *decision tree*. *Bagging* menggunakan model individu yang dibangun secara terpisah, untuk *boosting* dengan menggunakan model baru yang dipengaruhi oleh performa model yang dibangun sebelumnya (Yaman & Subasi, 2019).



Gambar 3. 9 Kerja *Internal* dari Algoritma *Boosting*
Sumber: (Malik et al., 2020)

Pada awalnya *boosting* dilakukan pada klasifikasi biner untuk mendapatkan nilai berupa $\in \{-1, 1\}$ dengan pengklasifikasi

$$D(x) = sign(f(x)) \tag{3.3}$$

Langkah-langkah yang dilakukan pada teknik *gradient boosting*, yaitu dengan memuat model ke dalam data dengan persamaan:

$$F_1(x) = y \tag{3.4}$$

Kemudian, terapkan model untuk menghitung hasil *residual* pada proses sebelumnya menggunakan persamaan:

$$h_1(x) = y - F_1(x) \tag{3.5}$$

Selanjutnya, membuat model baru dengan persamaan:

$$F_2(x) = F_1(x) + h_1(x) \quad (3.6)$$

Sehingga didapatkan model final hasil dari gabungan dari kumpulan model-model sebanyak n iterasi sampai menghasilkan nilai *error* terkecil dari *residual*. Model final dari *boosting* didefinisikan dengan persamaan berikut:

$$\hat{F}(x) = F_1(x) \rightarrow F_2(x) = F_1(x) + h_1(x) \ldots \rightarrow F_M(x) = F_{M-1}(x) + h_{M-1}(x) \quad (3.7)$$

Model akhir yang didapatkan untuk metode *boosting* didefinisikan dengan persamaan berikut:

$$f(x) = \sum_{m-1}^{M} f_m(x) \quad (3.8)$$

Atau dapat juga didefinisikan dengan persamaan berikut:

$$f(x) = \gamma_0 \sum_{m-1}^{M} \gamma_m h_m(x) \quad (3.9)$$

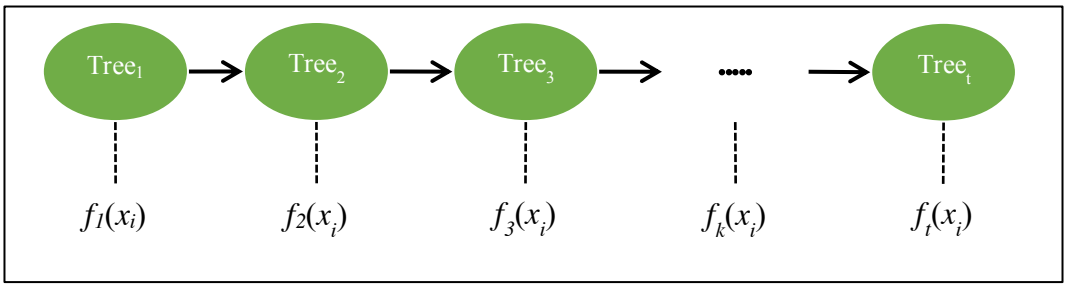
Dimana:

- f_x = γ_0
- $f_m(x)$ = $\gamma_m h_m(x)$, untuk m = 1,2,3, ..., M

Dengan nilai $h_m(x) \in \{-1,1\}$. $\gamma_m(x)$ merupakan pengklasifikasian lemah sedangkan γ_m adalah bobot pada tiap pengklasifikasi (Syahrani, 2019).

3.7.1 *Extreme Gradient Boosting*

Extreme Gradient Boosting (XGBoost) merupakan salah satu teknik dalam *machine learning* untuk analisis regresi dan klasifikasi berdasarkan *Gradient Boosting Decision Tree* (GBDT). Metode (XGBoost) pertama kali di diperkenalkan oleh (Friedman, 2001), dalam penelitiannya Friedman menghubungkan antara *boosting* dan optimasi dalam membangun *Gradient Boosting Machine* (GBM). Membangun model baru untuk memprediksi *error* dari model sebelumnya digunakan dalam metode *boosting*. Penambahan model baru dilakukan hingga tidak ada lagi perbaikan *error* yang dapat dilakukan. Dengan menggunakan *gradient descent* untuk memperkecil *error* saat membuat model baru, algoritma tersebut dinamakan *gradient boosting*. Proses komputasi algoritma *XGBoost* ditunjukkan pada **Gambar 3.10** berikut (Mo et al., 2019):



Gambar 3. 10 Diagram Skema dari Algoritma *XGBoost*

Nilai prediksi pada langkah t diumpamakan $\hat{y}_i^{(t)}$ dengan:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i)$$

$f_k(x_i)$ menggambarkan model pohon. Untuk y_i diperoleh dari perhitungan berikut:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_1) = \hat{y}_i^{(0)} + f_1(x_1) \\
\hat{y}_i^{(2)} &= f_1(x_1) + f_2(x_2) = \hat{y}_i^{(1)} + f_2(x_2) \\
&\vdots \\
\hat{y}_i^{(t)} &= \hat{y}_i^{(t-1)} + f_t(x_i) \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i)
\end{aligned} \tag{3.11}$$

Dimana:

- $\hat{y}_i^{(t)}$ = *Final tree* model
- $\hat{y}_i^{(t-1)}$ = Model pohon yang dihasilkan sebelumnya
- $f_t(x_i)$ = Model baru yang dibangun
- t = Jumlah total model dari *base tree models*

Untuk algoritma *Extreme Gradient Boosting*, penentuan jumlah pohon dan *depth* merupakan hal penting. Permasalahan dalam menemukan algoritma yang optimum dapat diubah dengan pencarian klasifikasi baru yang dapat mengurangi *loss function*, dengan target fungsi kerugian ditunjukkan pada persamaan (3.11) berikut:

$$Obj^{(t)} = \sum_{i=1}^{t} l(y_i, \hat{y}_i{}^{(t)}) + \sum_{i=i}^{t} \Omega(f_i) \tag{3.11}$$

Dimana:

- $\hat{y}_i{}^{(t)}$ = Nilai prediksi
- y_i = Nilai aktual
- $l(\hat{y}_i{}^{(t)}, y_i)$ = *lost function*
- $\Omega(f_i)$ = istilah regularisasi

Karena model *ensemble tree* pada persamaan (3.11) merupakan fungsi sebagai parameter dan tidak dapat dioptimalkan menggunakan metode pengoptimalan tradisional pada ruang *Euclidean*. Sehingga digantikan dengan model dilatih dengan cara aditif, dengan menggunakan $\hat{y}_i{}^{(t)}$ pada prediksi ke-*i* dan iterasi ke-*t* (Chen & Guestrin, 2016). Dalam meminimalkan *loss function* maka ditambahkan f_t sehingga didapatkan persamaan (3.12) sebagai berikut:

$$Obj^{(t)} = \sum_{i=1}^{t} l(y_i, \hat{y}_i{}^{(t-1)} + (f_t(x_i)) + \Omega(f_t) + constant \tag{3.12}$$

Selanjutnya target akhir dari *loss function* diubah menjadi persamaan (3.13), kemudian dilatih sesuai dengan target *loss function* berikut:

$$Obj^{(t)} = \sum_{i=1}^{t} \left[g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i) \right] + \Omega(f_t) \tag{3.13}$$

Dimana:

- g_i = $\partial_{y_i{}^{(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)}$
- h_i = $\partial^2_{y_i{}^{(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)}$

g_i dan h_i merupakan urutan pertama dan kedua statistik *gradient* pada *loss function*

Istilah regularisasi $\Omega(f_i)$ dapat dihitung menggunakan persamaan (3.14) yang digunakan untuk mengurasi kompleksitas model dan dapat meningkatkan kegunaan pada dataset yang lainnya.

$$\Omega(f_i) = \gamma T + \frac{1}{2} \lambda \|\omega\|^2 \tag{3.14}$$

Dimana:

- T = Jumlah *leaf*
- ω = Bobot *leaf*
- λ dan γ = Koefisien, dengan nilai *default* ditetapkan untuk λ=1 dan γ=0

Deret Taylor *second order* dapat digunakan untuk pendekatan fungsi tujuan. Misalkan $I_j = \{i|q(x_i) = j\}$ satu set contoh *leaf* j dengan $q(x)$ struktur tetap. Bobot optimal w_j^* *leaf* j dan nilai optimal yang sesuai dapat diperoleh dengan persamaan berikut.

$$w_j^* = -\frac{g_j}{h_j + \lambda}$$

dan

$$\mathcal{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \lambda T \tag{3.15}$$

Dimana:

g_i dan h_i = *Gradient* orde pertama dan kedua dari *loss function* $\mathcal{L}$

Loss function $\mathcal{L}$ dapat digunakan sebagai skor kualitas struktur pohon q. semakin kecil nilainya, maka modelnya akan semakin baik. Karena tidak mungkin untuk menghitung semua struktur pohon, *greedy* Algoritma dapat menyelesaikan masalah dengan memulai dari satu *leaf* dan menambahkan cabang pada pohon secara berulang. Misalkan I_R dan I_L adalah satu set contoh dari *node* kanan dan kiri setelah *split*. Dengan asumsi $= I_R \cup I_L$, maka setelah dilakukan *split* maka penurunan *loss* menggunakan formula sebagai berikut.

$$\mathcal{L}_{split} = -\frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{3.16}$$

Formula ini biasanya digunakan dalam praktik untuk mengevaluasi kandidat *split*. Model *XGBoost* menggunakan banyak pohon sederhana dan skor *leaf node* selama *split*. Tiga suku pertama dari persamaan masing-masing mewakili skor kiri, kanan, dan *leaf* asli. Selain itu istilah γ adalah regularisasi pada *leaf* tambahan dan akan digunakan dalam proses *training*.

Berikut merupakan tahapan dalam penyusunan Algoritma *Extreme Gradient Boosting* (Dayananda, 2020):

Diberikan suatu dataset dengan dua variabel yaitu variabel dosis obat, dan efektifitas.

Tabel 3. 3 Dataset untuk Membangun Pohon *XGBoost*
Sumber: (medium.com)

X	Y
2	0
8	1
12	1
18	0

Dengan dosis obat = X, efektifitas = Y

Parameter-parameter yang digunakan yaitu *n_estimators* atau model yang dibangun sebanyak 2, *max_depth* = 2, *learning_rate* = 1, *gamma* = 2, *min_child_weight* = 0, *reg_lambda* = 0, dan *base_score* = 0.5.

1. Prediksi awal

   Diberikan prediksi awal atau *base_score* sebesar 0.5 untuk seluruh titik data dalam dataset dimana,

   $$f_0(x) = h_0(x) = 0.5$$

2. Perhitungan *error* atau *residuals*

   Hitunglah *residuals* $\hat{Y}$ untuk semua titik data dari prediksi sebelumnya dimana

   $$\hat{Y} = y - f_0(x)$$

Tabel 3. 4 Perhitungan Nilai *Error* atau *Residuals*
Sumber: (medium.com)

X	Y	$f_0(x)$	$\hat{Y} = y - f_0(x)$
2	0	0.5	-0.5
8	1	0.5	0.5

X	Y	$f_0(x)$	$\hat{Y} = y - f_0(x)$
12	1	0.5	0.5
18	0	0.5	-0.5

3. Model latih

Latih model pertama (pelatihan model merujuk pada pembangunan pohon) yaitu M1 dengan menggunakan [X, $\hat{Y}$] data dan modelnya adalah pohon *XGBoost* khusus yang dibangun berbeda jika dibandingkan dengan *decision tee*. Terdapat beberapa istilah sebelum membangun pohon *XGBoost* dengan menurunkan rumus dalam menyelesaikan masalah optimasi pada *XGBoost* sebagai berikut.

$$Gain = \left(Left_{similarity} + Right_{similarity}\right) - Root_{similarity}$$

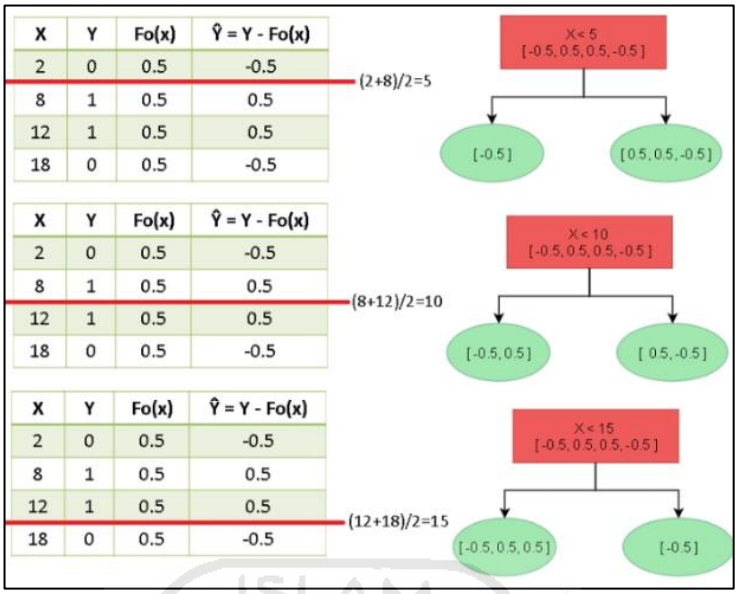
$$Similarity\ Score = \frac{\left(\sum \hat{Y}_i\right)^2}{\sum[Previous\ f_i(x)\ .(1 - Previous\ f_i(x)\ ] + \lambda}$$

$$Output\ Value = \frac{\left(\sum \hat{Y}_i\right)}{\sum[Previous\ f_i(x)\ .(1 - Previous\ f_i(x)\ ] + \lambda}$$

Dimana:

- $\hat{Y}_i$ = Residual ke-i
- λ = *reg_lambda*
- $Previous\ f_i(x)$ = probabilitas$_i$ sebelumnya
- Perhitungan nilai gain hanya untuk menghitung akar pohon
- Perhitungan *similarity* untuk semua *node*
- Perhitungan *output value* hanya untuk *leaf node*
- *Lambda* merupakan parameter, apabila nilai *lambda* meningkat akan menghasilkan *pruning* lebih banyak *node* pada pohon yang dibangun.

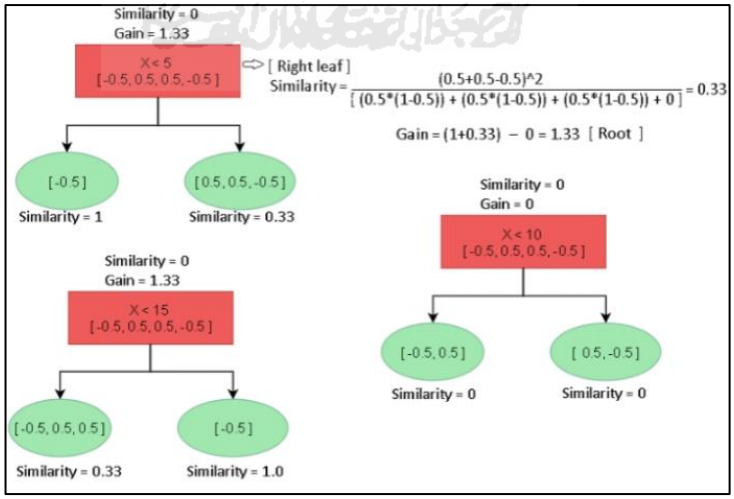
i. Pohon dibangun dengan membagi data menjadi dua partisi dari berbagai kemungkikan pemisahan atau *split*.



Gambar 3. 11 Contoh Membangun Pohon *XGBoost*
Sumber: (medium.com)

Menentukan batasan untuk *root* dihitung dengan mengambil nilai rata-rata antara dua titik percabangan atau *splits* dan sisanya menuju pada masing-masing *leaf node*. Jika dataset berisi n data, maka sejumlah n-1 pohon dapat dibangun.

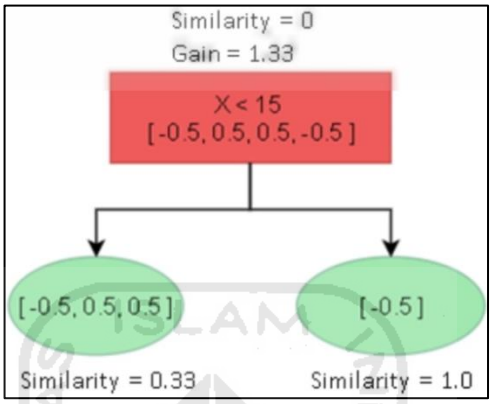
ii. Hitunglah nilai *similarity* dan *gain* pada seluruh pohon yang dibangun untuk menemukan pohon dengan *split* yang optimal.



Gambar 3. 12 Contoh Perhitungan *Similarity* dan *Gain*
Sumber: (medium.com)

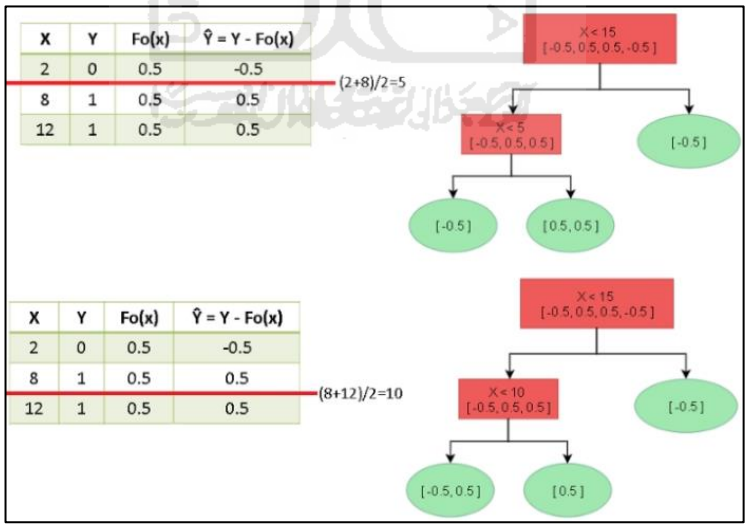
Pilihlah nilai dari suatu pohon yang memiliki nilai *gain* optimal, pada gambar 3.12 pohon dengan x<15 memiliki nilai *gain* maksimal sebesar 1.33.

iii. Lakukan pemisahan kembali untuk pohon yang memiliki nilai *gain* maksimal hingga *max_depth* untuk konstruksi pohon lengkap.



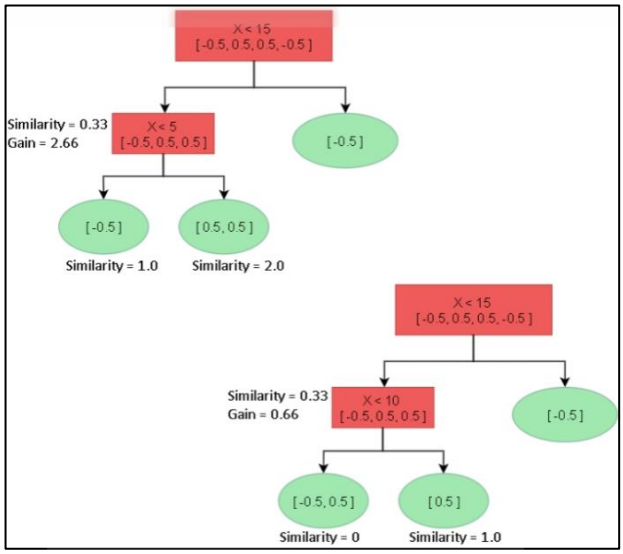
Gambar 3. 13 Contoh *Split* pada Pohon *XGBoost*
Sumber: (medium.com)

Karena nilai *max_depth* yang digunakan sebanyak dua, bangun pohon kembali dengan memisahkan data yang ada pada *leaf* sebelah kiri pada **Gambar 3.13.**



Gambar 3. 14 Contoh *Split* pada Turunan Percabangan
Sumber: (medium.com)

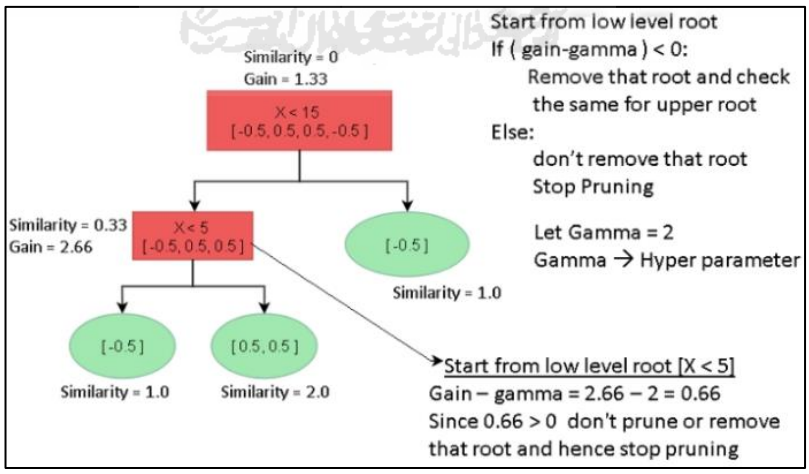
Hitung kembali nilai *similarity* dan *gain* untuk *split* baru pada percabangan lanjutan untuk memilih *internal root* yang memiliki nilai *gain* maksimal.



Gambar 3. 15 Contoh Perhitungan *Similarity* dan *Gain* pada *Split* Lanjutan
Sumber: (medium.com)

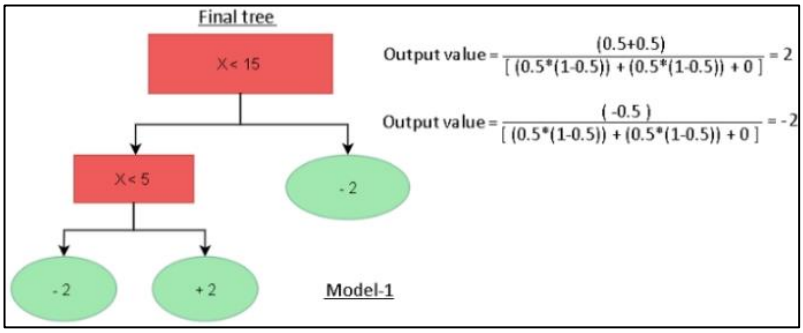
Iinternal root X<5 terpilih untuk konstruksi pohon *XGBoost* karena menghasilkan nilai gain maksimal sebesar 2.66.

iv. Setelah membangun pohon, langkah selanjutnya adalah melakukan *pruning* atau pemangkasan yang bertujuan untuk memperkecil ukuran pohon keputusan dengan menghilangkan bagian pohon yang memiliki kekuatan yang kurang untuk mengklasifikasikan kejadian.



Gambar 3. 16 Proses *Prunning* pada Pohon
Sumber: (medium.com)

v. Menghitung nilai *output* untuk semua *leaf* untuk mendapatkan pohon terakhir pada model 1, karena beberapa *leaf* memiliki lebih dari satu *residual*.



Gambar 3. 17 Contoh Perhitungan *Output Value*
Sumber: (medium.com)

vi. Menghitung prediksi dari model 1, seluruh titik data melewati pohon terakhir untuk mendapatkan $h_1(x)$ dan hitung prediksi $f_1(x)$ dan nilai *resisdual*.

Diberikan nilai *learning_rate* = 1.0

Maka

$$f_1(x) = \sigma\left[\left(\frac{h_0(x)}{1-h_0(x)}\right) + \left(\eta\ x\ h_1(x)\right)\right]$$

Memecahkan $f_1(x)$ pada klasifikasi didapatkan:

$f_1(x) = \sigma(0 + 1\ x(h_1(x)))$

Fungsi sigmoid:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

Tabel 3. 5 Perhitungan Nilai Prediksi pada Model-1
Sumber: (medium.com)

X	Y	$h_1(x)$	$f_1(x) = \sigma(0 + 1\ x(h_1(x)))$	$\hat{Y} = y - f_1(x)$
2	0	-2	-0.5	-0.11
8	1	2	0.5	0.12
12	1	2	0.5	0.12
18	0	-2	-0.5	-0.11

4. Ulangi Langkah 3 untuk membangun pohon yang lain.

Salah satu faktor yang paling penting dibalik kesuksesan *XGBoost* adalah lebih efisien dan *scalable* pada berbagai skenario, karena dapat menyelesaikan berbagai fungsi seperti, regresi, klasifikasi, maupun ranking. *Scalable* atau *scalability* sendiri disebabkan karena optimasi pada algoritma sebelumnya.

Algoritma *XGBoost* melakukan optimasi 10 kali lebih cepat dibandingkan dengan implementasi GBM lainnya. Kesuksesan tersebut dibuktikan dengan metode *XGBoost* menjadi metode yang banyak digunakan pada kompetisi *machine learning* (Chen & Guestrin, 2016).

3.8 *Tuning* Parameter dengan *Grid Search CV*

Beberapa metode *machine learning*, terdapat nilai parameter yang diatur guna mendapatkan model yang optimal, yang disebut *hyperparameter*. *hyperparameter* digunakan untuk mengatur berbagai macam aspek dalam *machine learning* yang sangat berpengaruh pada performa dan model yang dihasilkan. Pencarian *hyperparameter* dilakukan secara manual atau dengan menguji kumpulan *hyperparameter* pada parameter yang ditentukan sebelumnya (Claesen & De Moor, 2015).

Salah satu metode *hyperparameter* yang dapat diaplikasikan adalah *grid search*. *Grid search* merupakan metode alternatif yang digunakan untuk menemukan parameter terbaik dalam suatu model, sehingga metode yang digunakan secara akurat memprediksi data yang digunakan. *Grid search* dikategorikan sebagai metode yang teliti, karena dalam menentukan parameter terbaik dilakukan eksplorasi masing masing parameter dengan mengatur jenis nilai prediksi terlebih dahulu. Kemudian metode tersebut akan menampilkan skor untuk masing-masing nilai parameter. *Grid Search* dapat diterapkan secara maksimum, apabila batas atas dan batas bawah dari masing masing parameter diketahui (Ramadhan et al., 2017).

Parameter yang digunakan untuk melakukan *hyperparameter* pada metode *Random Forest* sebagai berikut:

Tabel 3. 6 Parameter pada Metode *Random Forest*
Sumber: (scikit-learn.org)

Parameter	Keterangan
n_estimators	Jumlah pohon pada *tree*
max_depth	Kedalaman maksimum pada *tree*
criterion	Pengukuran untuk kualitas *split*
min_samples_leaf	Jumlah sampel minimum yang dibutuhkan *leaf node*

Parameter	Keterangan
max_features	Jumlah fitur yang dipertimbangkan saat mencari *split* terbaik

Parameter pada metode *Extreme Gradient Boosting* yang digunakan untuk klasifikasi dan *hyperparameter* sebagai berikut:

Tabel 3. 7 Parameter pada Metode *XGBoost*
Sumber: (Xgboost.readthedocs.io)

Parameter	Keterangan
n_estimators	Jumlah pohon pada *tree*
max_depth	Kedalaman maksimum pada *tree*
min_child_weight	Jumlah bobot minimum pada *child node*
eta (*learning_rate*)	Penyusutan ukuran yang digunakan untuk mencegah *overfitting*
gamma	Nilai minimum *loss reduction*
subsample	Jumlah sampel yang digunakan saat proses pelatihan sebelum membangun *tree*
colsample_bylevel	Rasio subsample kolom pada setiap tingkatan

3.9 Pengukuran Kinerja Algoritma Klasifikasi

Setelah melakukan analisis pada metode klasifikasi dan didapatkan hasil prediksi. Langkah selanjutnya adalah melakukan kinerja algoritma untuk membandingkan nilai prediksi yang paling baik. Secara umum pengukuran kinerja algoritma klasifikasi dilakukan dengan membandingkan antara nilai prediksi algoritma klasifikasi dengan nilai target variabel data *testing* sebagai data sebenarnya.

3.9.1 *Confusion Matrix*

Confusion matrix adalah tabulasi dari perhitungan yang didasari pada evaluasi kinerja model klasifikasi berdasarkan jumlah objek penelitian yang diprediksi dengan benar dan salah. Secara singkat *confusion matrix* memberikan perincian terkait kesalahan klasifikasi (Gorunescu, 2010).

Tabel 3. 8 Tabel *Confusion Matrix*
Sumber: (Doreswamy & Hemanth, 2011)

Classification	*Predicted Positive*	*Predicted Negative*
Actual Positive	*True Positive (TP)*	*False Negative (FN)*
Actual Negative	*False Positive (FP)*	*True Negative (TN)*

Dalam pengukuran kinerja *confusion matrix* terdapat empat bagian untuk mengidentifikasi suatu prediksi, berikut diantaranya (Doreswamy & Hemanth, 2011):

1. TP (*True Positive*) adalah jumlah data dengan nilai aktual positif dan nilai prediksi positif
2. TN (*True Negative*) adalah jumlah data dengan nilai actual positif dan nilai prediksi negatif
3. FP (*False Positive*) adalah jumlah data dengan nilai actual negatif dan nilai prediksi positif
4. FN (*False Negative*) adalah jumlah data dengan nilai actual negatif dan nilai prediksi negatif

Pada klasifikasi biner terdapat beberapa nilai evaluasi yang sering digunakan. Dapat dilihat berdasarkan nilai *confusion matrix* (Sokolova & Lapalme, 2009):

1. *Accuracy* (ACC) adalah efektivitas keseluruhan dari hasil klasifikasi

$$accuracy\ (\%) = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3.17}$$

2. *Precision* (PREC) merupakan presentase dari label data dengan label positif yang diberikan oleh klasifikasi

$$Precision\ (\%) = \frac{FN}{(FP + FN)} \tag{3.18}$$

3. *Recall* (REC) atau *sensitivity* adalah efektivitas dari pengklasifikasi dalam mengidentifikasi label positif

$$Recall\ (\%) = \frac{TP}{(TP + FN)} \tag{3.19}$$

3.9.2 *Area Under the Curve* (AUC)

Area Under Curve merupakan metode umum yang digunakan untuk menghitung *under the ROC curve*. AUC dapat diartikan sebagai probabilitas, jika memilih satu satu contoh positif dan negatif secara acak, metode klasifikasi akan memberikan skor lebih tinggi pada contoh positif daripada contoh negatif. Oleh karena itu, nilai AUC yang lebih tinggi menunjukkan metode klasifikasi yang lebih baik, yang menjadikan nilai AUC sebagai tujuan dalam pemaksimalan (Gorunescu, 2010). Berikut dibawah ini formula dalam mencari nilai AUC dilihat dari hasil *confusion matrix* (Sokolova & Lapalme, 2009):

$$AUC = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{3.20}$$

Nilai *Area under Curve* (AUC) akan selalu berada pada range 0-1, karena bagian dari luas persegi satuan dengan sumbu x dan sumbu y memiliki nilai dari 0 sampai 1. Nilai diatas 0,5 dikatakan nilai yang menarik karena prediksi acak menghasilkan garis diagonal antara (0,0) dan (1,1) yang memiliki luas 0,5. Kualitas klasifikasi keakuratan dari tes diagnostik menggunakan nilai AUC ditunjukkan pada **Tabel 3.6** (Gorunescu, 2010).

Tabel 3. 9 Keakuratan hasil klasifikasi berdasarkan nilai AUC
Sumber : (Gorunescu, 2010)

Nilai AUC	Kategori
0.90 - 1.00	Sangat Baik
0.80 – 0.90	Baik
0.70 – 0.80	Cukup Baik
0.60 – 0.70	Kurang Baik
0.50 – 0.60	Buruk

BAB 4 METODOLOGI PENELITIAN

4.1 Jenis dan Sumber Data

Penelitian ini menggunakan data sekunder. Data tersebut diperoleh dari situs dan *platform* kompetisi *Kaggle* dengan URL (https://www.kaggle.com/nikkitha8/telecom-churn). Data tersebut berupa data *customer churn* pada salah satu perusahaan telekomunikasi dengan jumlah data sebanyak 3,333.

4.2 Variabel Penelitian

Adapun variabel-variabel yang digunakan pada penelitian ini adalah sebanyak 13 variabel. Berikut tabel dibawah memuat variabel penelitian dan penjelasan definisi operasional dari masing-masing variabel:

Tabel 4. 1 Penelitian Operasional Variabel

No	Nama Variabel	Definisi Operasional	Tipe Data
1	*AccountLength*	Jangka waktu akun telah aktif	*Integer*
2	*International Plan*	Rencana atau mengaktifkan panggilan internasional	*Categorical*
3	*Number Vmail Plan*	Pesan Suara	*Integer*
4	*Total Day Minutes*	Total durasi panggilan siang hari yang digunakan	
5	*Total Day Calls*	Total berapa kali *customer* melakukan panggilan pada siang hari	
6	*Total Eve Minutes*	Total durasi panggilan sore hari yang digunakan	
7	*Total Eve Calls*	Total berapa kali *customer* melakukan panggilan pada sore hari	

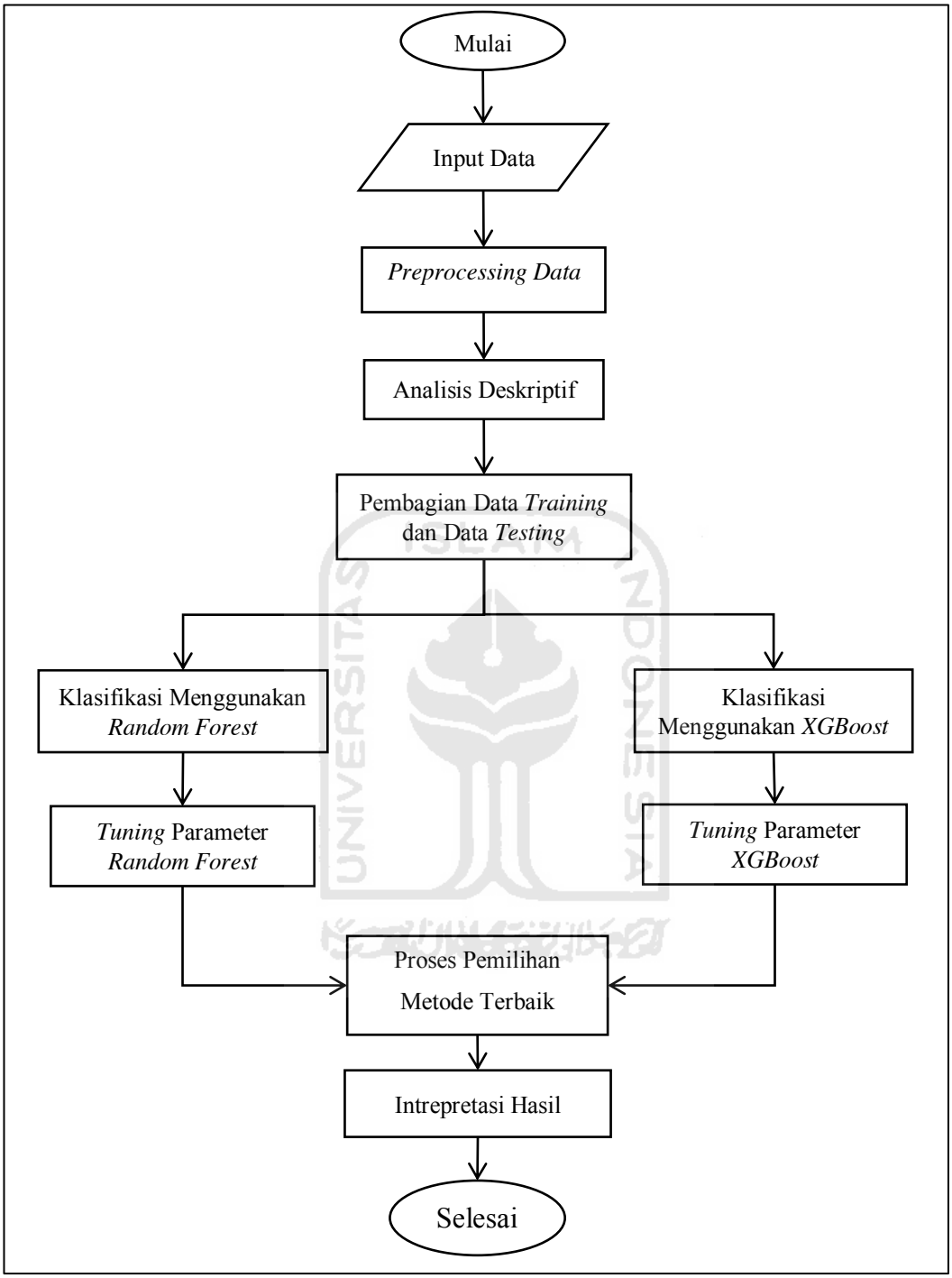
No	Nama Variabel	Definisi Operasional	Tipe Data
8	*Total Night Minutes*	Total durasi panggilan malam hari yang digunakan	
9	*Total Night Calls*	Total berapa kali *customer* melakukan panggilan pada malam hari	
10	*Total Intl Minutes*	Total durasi panggilan internasional yang digunakan	
11	*Total Intl Calls*	Total berapa kali *customer* melakukan panggilan internasioanl	
12	*Customer Service Calls*	Jumlah panggilan *customer* kepada layanan pelanggan (*customer service*)	
13	*Churn*	Status *churn*	*Categorical*

4.3 Metode Analisis Data

Pada penelitian ini metode analisis data yang digunakan adalah dua metode klasifikasi yaitu *Random Forest* dan *Extreme Gradient Boosting* (XGBoost). Kedua analisis ini digunakan untuk mengklasifikasikan *Churn* pada pelanggan di salah satu industri telekomunikasi di Negara bagian Columbia, dengan menggunakan 13 variabel, dengan menggunakan bantuan *software Microsoft Excel* dan *Python* 3.8.3.

4.4 Tahapan Analisis Data

Berikut ini adalah tahapan yang dilakukan pada penelitian yang disajikan dalam bentuk *flowchart* berikut ini:



Gambar 4. 1 *Flow Chart* Penelitian

1. Tahapan awal yang dilakukan oleh peneliti melakukan *input* data pada *software* yang digunakan dalam proses analisis data.
2. Langkah selanjutnya adalah *preprocessing* data dengan membagi data numerik dan kategorik. Untuk data kategorik diberi label menggunakan

metode *label encoding*, dan memilih variabel yang akan dilakukan untuk analisis.

3. Selanjutnya melakukan analisis deskriptif, guna melihat bagaimana gambaran data yang digunakan.
4. Proses selanjutnya adalah pembagian data *training* dan data *testing*, dengan jumlah data *training* sebesar 80%, dan data *testing* 20% dari total keseluruhan data.
5. Selanjutnya menganalisis data *training* menggunakan metode klasifikasi *Random Forest* kemudian melakukan *tuning* parameter untuk menentukan nilai parameter optimum, parameter yang digunakan yaitu *n_estimators, max_depth, criterion, min_samples_leaf,* dan *max_features*.
6. Untuk klasifikasi pada metode *XGBoost tuning* parameter yang digunakan yaitu *n_estimators*, *max_depth*, *min_child_weight*, *eta* (*learning_rate*), *gamma*, *subsample*, *colsample_bylevel*.
7. Setelah mendapatkan hasil klasifikasi dari masing-masing metode langkah selanjutnya yaitu, membandingkan metode yang memiliki nilai akurasi terbaik dalam memprediksi data *testing*.
8. Kemudian melakukan interpretasi dari model terbaik yang didapatkan.

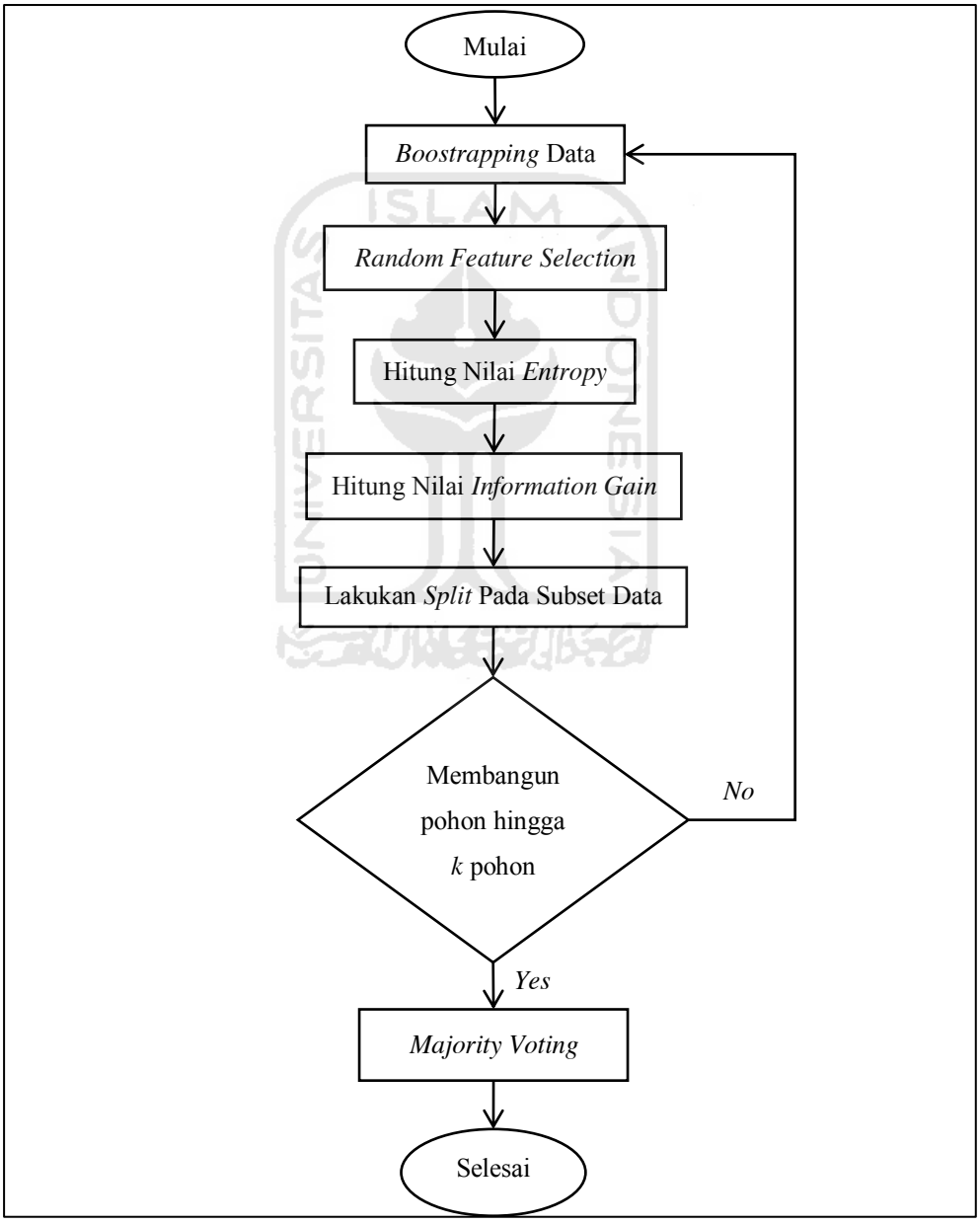
4.4.1 *Tahapan Analisis* Metode *Random Forest*

Berikut adalah langkah-langkah dalam melakukan klasifikasi menggunakan metode *Random Forest*.

1. Tahapan pertama yaitu *boostrapping* data yaitu melakukan pengambilan sampel secara acak dari dataset data *training* dengan pengembalian.
2. Langkah selanjutnya yaitu *Random Feature Selection* dimana pada tahap ini pohon dibangun hingga meenapai ukuran maksimum (tanpa dilakukan *pruning*).
3. Selanjutnya menghitung nilai *entropy* dari seluruh *feature* yang digunakan untuk membangun pohon.
4. Setelah mendapatkan nilai *entropy* dari masing-masing *features*, dilakukan perhitungan *information gain* dari niali *entropy* yang

didapatkan. *Feature* yang memiliki nilai *information gain* paling besar akan dijadikan *root node* pada pohon yang dibangun.

5. Selanjutnya dilakukan *split* hingga semua *node* berbentuk *leaf node* atau tidak ada *split* yang dapat dilakukan kembali.
6. Apabila belum selesai membangun pohon sebanyak *k* pohon ulangi langkah 1-5, apabila pohon yang dibangun sudah mencapai *k* pohon dilakukan *majority voting* untuk prediksi akhir.

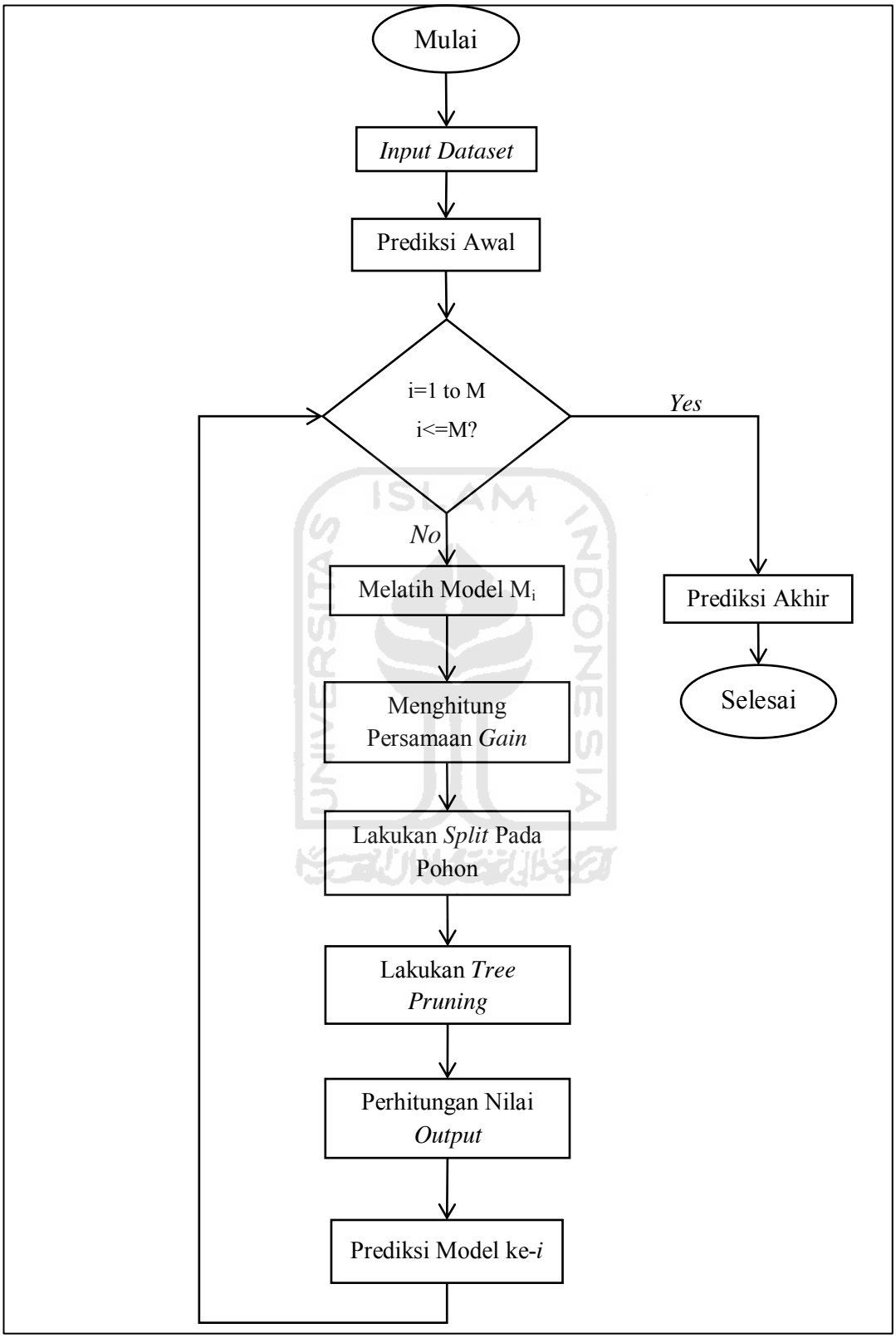


Gambar 4. 2 *Flowchart* Metode *Random Forest*

4.4.2 Tahapan Analisis Metode *Extreme Gradient Boosting*

Berikut merupakan langkah-langkah yang dilakukan dalam klasifikasi menggunakana metode *Extreme Gradient Boosting*.

1. Langkah pertama yang dilakukan adalah meng*input*kan data yang akan digunakan.
2. Kemudian melakukan prediksi awal dengan menghitung residuals $\hat{Y}$ pada seluruh titik data dari prediksi sebelumnya.
3. Apabila pohon ke-*i* sudah dibangun hingga mencapai pohon ke-M maka dilakukan prediksi akhir *i*, tetapi apabila pohon ke-*i* belum mencapai M pohon dilakukan pelatihan pada model M_i.
4. Langkah selanjutnya menghitung nilai *similarity* dan *gain* pada selirij pohon yang dibangun untuk menemukan pohon dengan *split* yang optimal.
5. Melakukan *split* kembali untuk *node* yang memiliki nilai *gain* maksimal hingga kontruksi pohon lengkap.
6. Setelah pohon yang dibangun sudah lengkap, dilakukan proses *pruning* atau pemangkasan dengan tujuan untuk memperkecil ukuran pohon dan menghilangkan bagian pohon yang memiliki kekuatan yang kurang untuk mengklasifikasikan.
7. Menghitung nilai *output* untuk semua *leaf* guna mendapatkan pohon terakhir dari model M_i
8. Menghitung prediksi dari model M_i.
9. Setelah itu kembali pada langkah ke-3 apabila seluruh pohon M sudah dibangun dilakukan prediksi akhir untuk pohon *Extreme Gradient Boosting*.



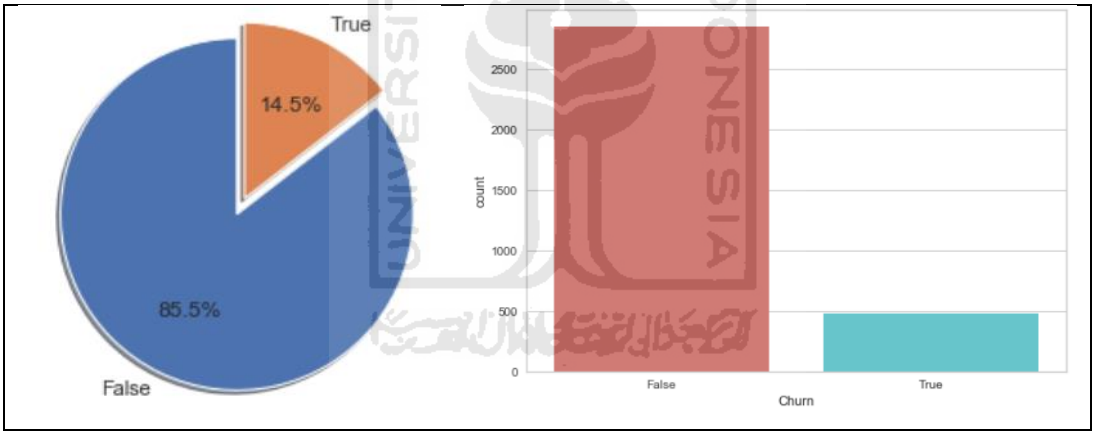
Gambar 4. 3 *Flowchart* Metode *Extreme Gradient Boosting*

BAB 5 HASIL DAN PEMBAHASAN

Berdasarkan kajian teori dan hasil penelitian sebelumnya, maka pada bab ini peneliti akan menjelaskan hasil-hasil yang telah didapatkan dari hasil komputasi klasifikasi menggunakan metode *Random Forest* dan *Extreme Gradient Boosting*. Pembahasan pada bab ini meliputi analisis deskriptif, *preprocessing* data, klasifikasi menggunakan metode *Random Forest* dan *Extreme Gradient Boosting* serta mengevaluasi kedua metode klasifikasi yang digunakan.

5.1 Analisis Deskriptif

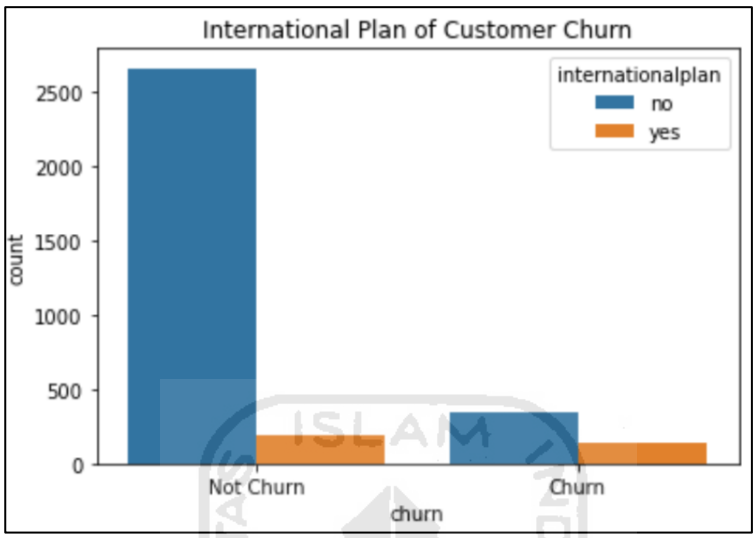
Salah satu tahapan yang dilakukan sebelum melakukan analisis data adalah analisis deskriptif yang digunakan untuk menggambarkan atau mendeskripsikan masing-masing variabel.



Gambar 5. 1 *Pie Chart* dan *Barplot Customer Churn*

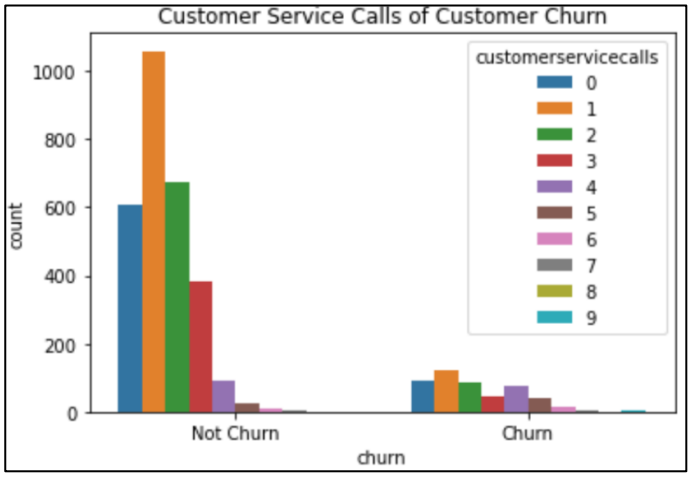
Pada **Gambar 5.1** (i) terdapat *piechart* yang menunjukan presentase *customer churn* di salah satu perusahaan telekomunikasi. Sebanyak 14.5% *customer* mengalami *churn* atau meninggalkan perusahaan dan berpindah ke perusahaan lain dan 85.5% sisanya bertahan menggunakan layanan perusahaan tersebut. Dari **Gambar 5.1** (ii) dapat dilihat bahwa dari 3,333 *customer* yang melakukan *churn* kurang dari 500 *customer* dan *customer* yang bertahan atau *not churn* lebih dari 2,500. Hal ini menunjukan bahwa perusahaan memiliki kualitas yang baik karena jumlah *customer* yang *not churn* lebih banyak dibandingkan kasus *churn*. Peneliti ingin mengetahui apa saja faktor-faktor yang mempengaruhi

customer melakukan *churn* untuk meminimalisir terjadinya *churn* pada penyedia jasa telekomunikasi. Berikut dibawah ini merupakan faktor-faktor atau variabel yang dapat menyebabkan terjadinya *churn*.



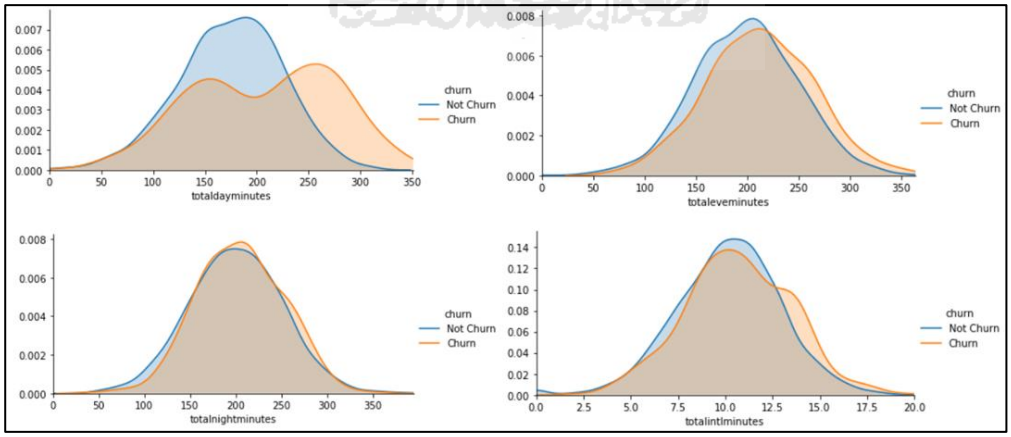
Gambar 5. 2 *Bar Chart* Variabel *International Plan*

Berdasarkan **Gambar 5.2** variabel *International Plan* atau rencana *customer* untuk mengaktifkan panggilan internasional. Pada *bar chart* tersebut rencana mengaktifkan panggilan internasional untuk *customer* yang melakukan *churn* maupun yang tidak melakukan *churn* lebih sedikit dibandingkan dengan *customer* yang tidak melakukan rencana pengaktifkan panggilan internasional. Untuk customer yang tidak *churn* lebih dari 2500 tidak melakukan rencana pengaktifan panggilan internasional, dibandingkan dengan customer yang melakukan *churn* kurang dari 500 *customer*. Berikut dibawah ini merupakan analisis deskriptif variabel

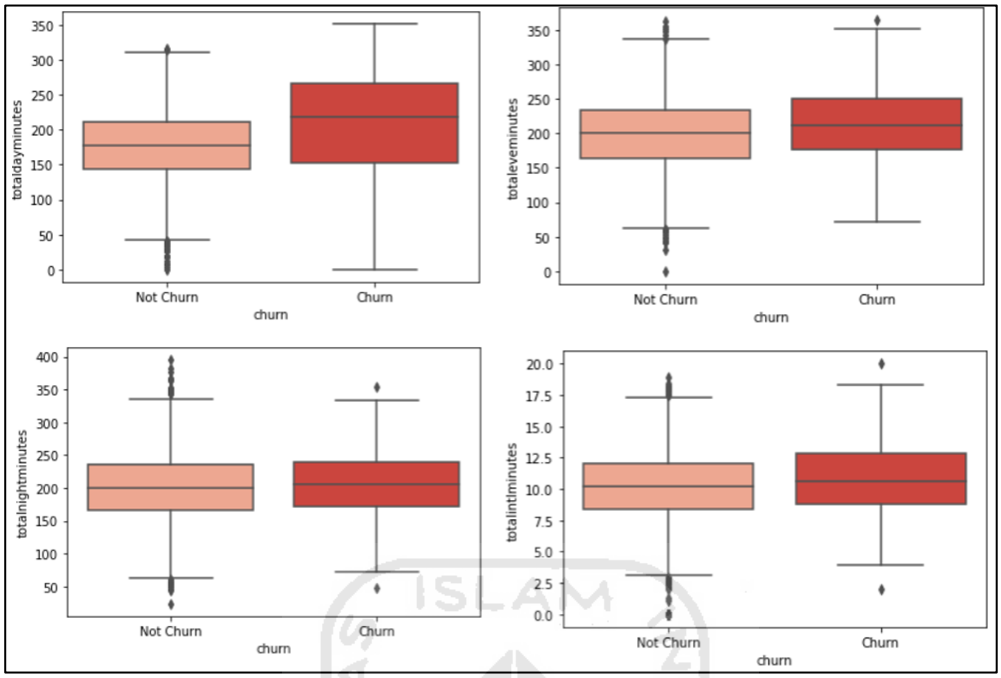


Gambar 5. 3 *Bar Chart* Variabel *Customer Service Calls*

Pada **Gambar 5.3** diatas menggambarkan berapa kali *customer* melakukan panggilan pada *customer service provider* yang digunakan. Dapat dilihat untuk kasus *churn* maupun tidak *churn* paling banyak *customer* melakukan panggilan sebanyak 1 kali. *Customer* yang tidak melakukan *churn* kebanyakan melakukan panggilan sebanyak 1 sampai 3 kali, bahkan tidak sedikit yang tidak melakukan panggilan sama sekali. Hampir sama dengan *customer* yang melakukan *churn* tetapi banyak pelanggan yang melakukn panggilan sebanyak 4 kali, angka ini lebih banyak dibandingkan *customer* yang melakukan panggilan sebanyak 3 kali. Berikut di bawah merupakan gambaran dari variabel-variabel durasi panggilan.



Gambar 5. 4 Distribusi Variabel-variabel Durasi Panggilan



Gambar 5. 5 *Boxplot* Variabel-variabel Durasi Panggilan

Tabel 5. 1 Tabel *Crosstab* Variabel-variabel Durasi Panggilan

Variabel		***Churn***	
		Churn	***Not Churn***
Total Day Minutes	***Max***	350.8	315.6
	Min	0	0
	Mean	206.91	175.18
Total Eve Minutes	***Max***	363.7	361.8
	Min	70.9	0
	Mean	212.41	199.04
Total Night Minutes	***Max***	354.9	395
	Min	47.4	23.2
	Mean	205.23	200.13
Total Intl Minutes	***Max***	20	18.9
	Min	2	0
	Mean	10.7	10.16

Dari **Gambar 5.4** (i) dapat dilihat bahwa sebaran data untuk variabel *total day minutes*, *customer* yang mengalami *churn* dan *not churn* berbeda dan tidak berdistribusi normal. **Gambar 5.5** (i) menampilkan *boxplot* dari variabel *total day minutes*. Untuk *customer* yang melakukan panggilan pagi hari lebih dari 250 menit cenderung mengalami *churn* atau berpindah ke operator lain. Dari tabel *crosstab* dapat dilihat bahwa durasi rata-rata *customer* yang melakukan panggilan pagi hari selama 206.91 menit akan melakukan *churn*, hal ini bisa saja karena

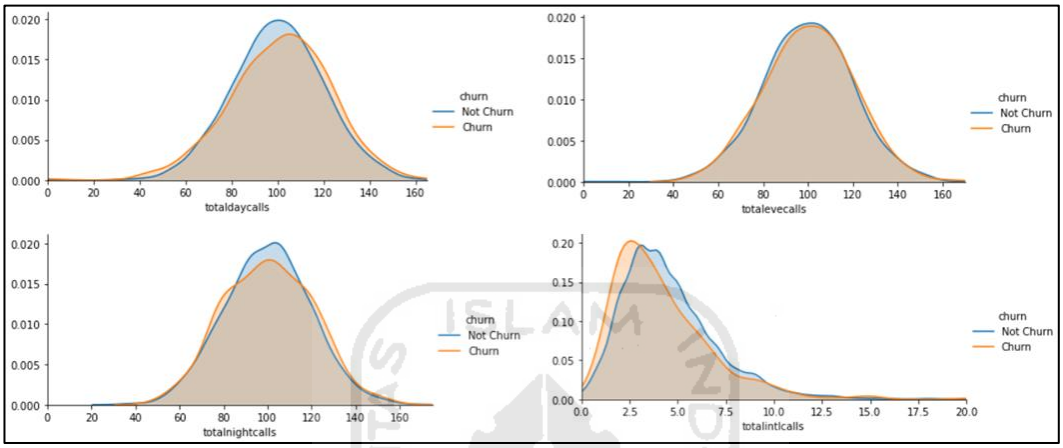
faktor jaringan yang tidak stabil di pagi hari yang menyebabkan *customer* berpindah ke operator lain.

Gambar 5.4 (ii) merupakan sebaran data untuk variabel *total eve minutes*, sebaran data *customer* yang mengalami *churn* dan *not churn* membentuk distribusi normal, dapat dikatakan data cenderung seimbang. **Gambar 5.5** (ii) ditampilkan *boxplot* dari variabel *total eve minutes* terdapat *outlier* dikarenakan *range* nilai yang terlalu jauh. Untuk *customer* yang melakukan panggilan sore hari lebih dari 200 menit cenderung mengalami *churn* atau berpindah ke operator lain. Dari tabel *crosstab* dapat dilihat bahwa durasi rata-rata *customer* yang melakukan panggilan sore hari selama 212.41 menit melakukan *churn*. Perbandingan durasi panggilan sore hari *customer* yang melakukan *churn* dan *not churn* tidak jauh.

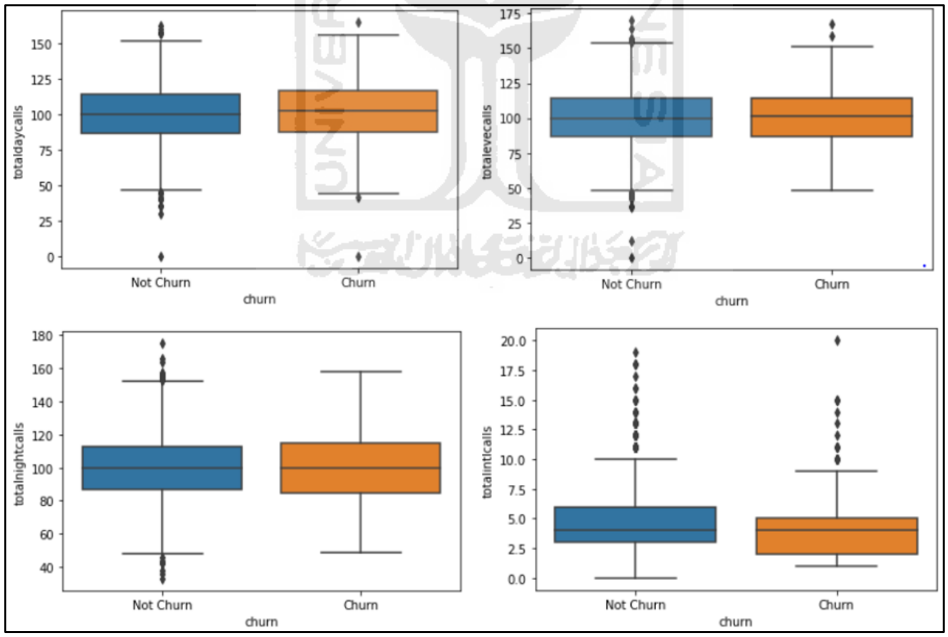
Gambar 5.4 (iii) merupakan sebaran data untuk variabel *total night minutes*, sebaran data *customer* yang mengalami *churn* dan *not churn* membentuk distribusi normal, dapat dikatakan data cenderung seimbang. **Gambar 5.5** (iii) ditampilkan *boxplot* dari variabel *total night minutes* terdapat *outlier* yang disebabkan oleh *range* nilai yang terlalu jauh. Rata-rata durasi panggilan malam hari untuk *customer* yang melakukan *churn* dan *not churn* sangat kecil dapat dilihat pada visualisasi *boxplot* yang hampir setara. Terbukti dari hasil tabel *crosstab* selisih antara rata-rata durasi panggilan malam hari *customer* yang *churn* dan *not churn* hanya 5.1 menit, hal ini sedikit menyulitkan perusahaan karena perbedaan antara *customer* yang melakukan panggilan malam hari sangat tipis, tetapi durasi panggilan malam hari lebih lama dibandingkan durasi pada waktu pagi maupun sore hari. Hal ini dapat menjadi pertimbagan operator untuk menarik *customer* dengan memberikan promo atau memperbaiki jaringan dari operator tersebut.

Gambar 5.4 (iv) merupakan sebaran data untuk variabel *total intl minutes*, dimana sebaran data *customer* yang mengalami *churn* dan *not churn* cenderung sama dan membentuk distribusi normal, dapat dikatakan data cenderung seimbang. Kemudian pada **Gambar 5.5** (iv) ditampilkan *boxplot* dari variabel *total intl minutes* terdapat beberapa data *outlier* yang disebabkan oleh *range* data yang terlalu jauh. Dari *boxplot* tersebut dapat dilihat bahwa *customer* yang

melakukan panggilan internasional lebih dari 12.5 menit akan melakukan *churn*. Dari tabel *crosstab* dapat dilihat bahwa durasi rata-rata *customer* yang melakukan panggilan sore hari selama 10.3 menit melakukan *churn*. Perbandingan durasi panggilan sore hari *customer* yang melakukan *churn* dan *not churn* tidak jauh. Selanjutnya gambaran sebaran data untuk variabel-variabel jumlah panggilan.



Gambar 5. 6 Distribusi Variabel-variabel Total Panggilan



Gambar 5. 7 *Boxplot* Variabel-variabel Total Panggilan

Tabel 5. 2 Tabel *Crosstab* Variabel-variabel Total Panggilan

Variabel		***Churn***	
		Churn	***Not Churn***
Total Day Minutes	***Max***	165	163
	Min	0	0
	Mean	101	100

Total Eve Minutes	***Max***	168	170
	Min	48	0
	Mean	101	100
Total Night Minutes	***Max***	158	178
	Min	49	33
	Mean	100	100
Total Intl Minutes	***Max***	20	19
	Min	1	0
	Mean	4	5

Dari **Gambar 5.6** (i) dapat dilihat bahwa sebaran data untuk variabel *total day calls*, *customer* yang mengalami *churn* dan *not churn* cenderung sama dan membentuk distribusi normal, dapat dikatakan data cenderung seimbang. **Gambar 5.7** (i) ditampilkan *boxplot* dari variabel *total day minutes*. Untuk *customer* yang melakukan panggilan pagi hari lebih dari 100 kali cenderung mengalami *churn* atau berpindah ke operator lain sama halnya dengan *customer not churn*. Dari tabel *crosstab* dapat dilihat bahwa selisih rata-rata total panggilan yang dilakukan oleh *customer churn* maupun *not churn* hanya satu panggilan saja.

Gambar 5.6 (ii) merupakan sebaran data untuk variabel *total eve calls*, sebaran data *customer* yang mengalami *churn* dan *not churn* membentuk distribusi normal, dapat dikatakan data cenderung seimbang. **Gambar 5.7** (ii) ditampilkan *boxplot* dari variabel *total eve calls* terdapat *outlier* dikarenakan *range* nilai yang terlalu jauh. Untuk *customer* yang melakukan panggilan sore hari lebih dari 100 kali cenderung mengalami *churn* atau berpindah ke operator lain sama halnya dengan *customer not churn*. Dari tabel *crosstab* dapat dilihat bahwa selisih rata-rata total panggilan yang dilakukan oleh *customer churn* maupun *not churn* hanya satu panggilan saja.

Gambar 5.6 (iii) merupakan sebaran data untuk variabel *total night calls*, sebaran data *customer* yang mengalami *churn* dan *not churn* membentuk distribusi normal, dapat dikatakan data cenderung seimbang. **Gambar 5.7** (iii) ditampilkan *boxplot* dari variabel *total night calls* terdapat *outlier* yang disebabkan oleh *range* nilai yang terlalu jauh. Untuk *customer* yang melakukan panggilan sore hari lebih dari 100 kali cenderung mengalami *churn* atau berpindah ke operator lain sama halnya dengan *customer not churn*. Dari tabel

crosstab dapat dilihat bahwa rata-rata total panggilan *customer churn* maupun *not churn* memiliki nilai sama, hal ini membuat perusahaan kesulitan untuk mengidentifikasi perilaku *customer* yang melakukan panggilan di malam hari.

Gambar 5.6 (iv) merupakan sebaran data untuk variabel *total intl calls*, dimana sebaran data *customer* yang mengalami *churn* dan *not churn* cenderung sama dan tidak membentuk distribusi normal atau distribusi miring negatif. Kemudian pada **Gambar 5.7** (iv) ditampilkan *boxplot* dari variabel *total intl calls* terdapat banyak data *outlier* yang disebabkan oleh *range* data yang terlalu jauh. Dari *boxplot* tersebut dapat dilihat bahwa *customer* yang melakukan panggilan internasional kurang dari 5 kali akan melakukan *churn*. Dari tabel *crosstab* dapat dilihat bahwa total panggilan *customer churn* sebanyak 4 kali panggilan, total panggilan internasional *customer churn* lebih kecil dibandingkan dengan yang *not churn*.

5.2 *Pre-Processing* Data

Sebelum melakukan analisis data, diperlukan ekstaksi data dengan cara mengubah dan menjadikan bentuk data yang lebih mudah dipahami melalu proses eliminasi data yang tidak sesuai dan dibutuhkan. Pertama-tama peneliti men*drop* variabel *state*, *phone number*, dan *area code*. Kemudian membagi data kategorik dan numerik. Variabel-variabel dengan data kategorik yaitu *interational plan*, *voice mail plan*, dan *churn*.

Tabel 5. 3 Pelabelan pada Data Kategorik

No	Variabel	Label	Keterangan
1	*International Plan*	0	No
		1	Yes
2	*Voice Mail Plan*	0	No
		1	Yes
3	*Churn*	0	No
		1	Yes

Proses pelabelan data dapat dilihat pada **Tabel 5.3** menunjukkan hasil dari pelabelan data kategorik menjadi skala nominal, pelabelan ini digunakan hanya untuk membedakan kategori berdasarkan variasi data. Terdapat tiga variabel yang memiliki 3 kategori antara lain variabel *international plan*, *voice mail plan*, dan *churn* dengan nilai kategorik 0 dan 1, 0 untuk data "*no*" dan 1 untuk data "*yes*". Selanjutnya melakukan *feature selection* data dengan metode korelasi *Pearson* dengan matriks korelasi.

Gambar 5. 8 *Correlation Matrix*

Gambar 5.8 menunjukkan matriks korelasi dari seluruh variabel yang digunakan. Terdapat beberapa variabel yang sangat berkorelasi antara lain, variabel *number vmail messages*, *voice mail plan*, *total day minutes*, *total day charge*, *total eve minutes*, *total eve charge*, *total night minutes*, *total night charge*, *total intl charge*, *total intl minutes*. Oleh karena itu peneliti hanya memilih satu antara dua variabel yang berkorelasi tinggi. Pada tahap ini variabel *voice mail plan*, *total day charge*, *total eve charge*, *total night charge*, dan *total intl charge* tidak dipilih dalam analisis. Pertimbangan yang peneliti lakukan untuk mengurangi variabel yang saling berkorelasi yaitu untuk memungkinkan algoritma *machine learning* dalam proses *training* bekerja lebih cepat, mengurangi model yang kompleks agar *output* yang dihasilkan mudah diinterpretasi, dan dapat meningkatkan nilai akurasi dalam model yang dibangun.

5.3 Pembagian Data *Training* dan *Data Testing*

Sebelum melakukan analisis klasifikasi dilakukan pembagian data menjadi dua bagian, yaitu data *training* dan data *testing*. Data *training* berguna melatih algoritma untuk pembentukan sebuah model, dan data *testing* digunakan untuk mengukur sejauh mana tingkat keakuratan dan performa yang didapatkan dari data *training*. Data *training* dan data *testing* dibagi dengan proporsi 80% untuk data *training* dan 20% dari data *testing* dari total dataset.

Tabel 5. 4 Proporsi Data *Training* dan Data *Testing*

Keterangan	**Data *Training***	**Data *Testing***	**Total**
Proporsi	80%	20%	100%
Jumlah	2,666	667	3,333

Berdasarkan **Tabel 5.4** dapat diketahui dari 3,333 dataset yang ada, pembagian data untuk data *training* sebanyak 2,666 data, dan untuk data *testing* sebanyak 667 data. Pembagian data *training* dan *testing* pada dataset dilakukan secara *random* dengan bantuan *software Python*.

5.4 Analisis Klasifikasi *Random Forest*

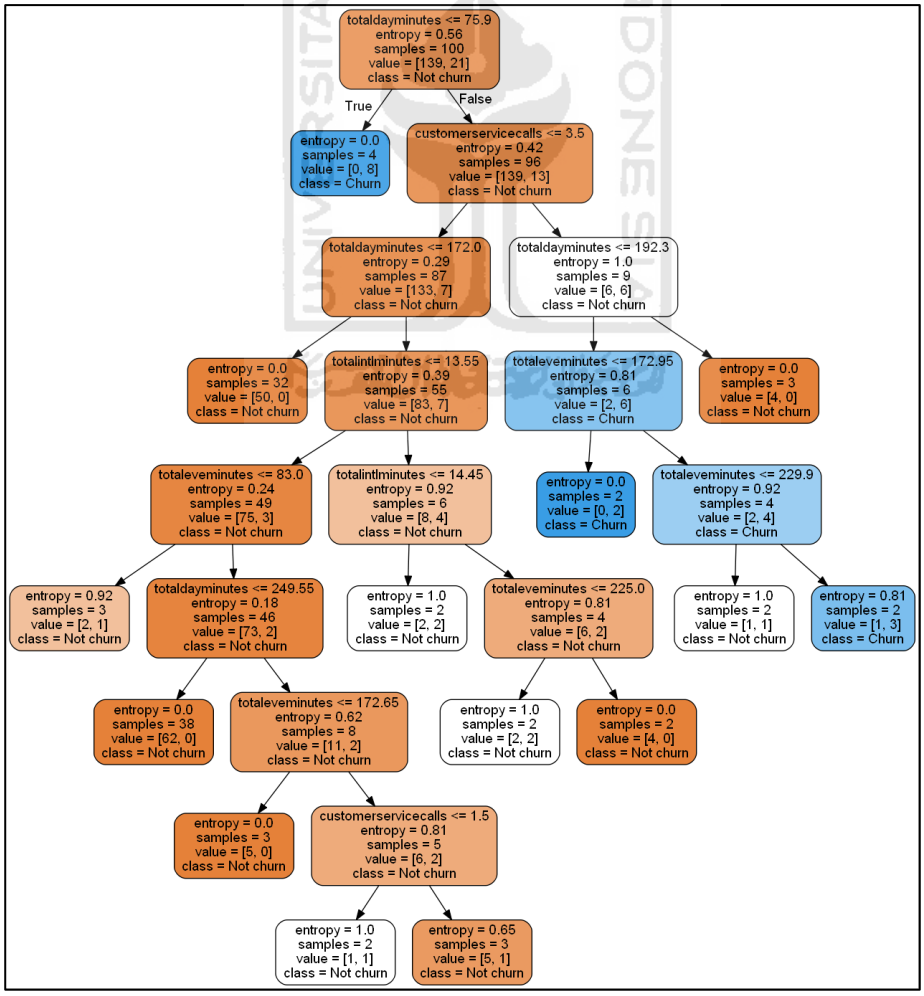
Setelah melakukan pembagian data *training* dan data *testing* tahapan selanjutnya yaitu melakukan analisis klasifikasi *Random Forest*. Langkah pertama yang dilakukan adalah melakukan *tuning* parameter menggunakan fungsi pada *scikit-learn* yaitu *gridsearchCV*. Hasil dari *tuning* parameter ditampilkan pada tabel berikut.

Tabel 5. 5 Hasil *Tuning* Parameter Metode *Random Forest*

Parameter	***Grid Search Values***	***Best* Parameter**
n_estimators	100,200,300	100
max_depth	4,5,6,7,8	8
criterion	*Gini, Entropy*	*Entropy*
min_samples_leaf	2,3,4,5,6,7,8,9,10	2
max_features	*auto,sqrt,log2*	*sqrt*

Tabel 5.5 merupakan hasil *tuning* parameter yang didapatkan dari proses *grid searchCV* dengan melakukan pencarian secara menyeluruh terhadap parameter yang di ujikan. Penelitian ini menggunakan 5-*fold cross validation* yang digunakan untuk mengevaluasi kinerja model sebanyak lima kali perulangan dalam proses *grid search* dari setiap parameter. Nilai parameter terbaik dari proses *grid search* digunakan dalam penentuan model klasifikasi.

Pada klasifikasi *Random Forest* terdapat seratus pohon yang digunakan, kemudian setiap pohon membuat delapan percabangan. Akar kuadrat dari total fitur digunakan saat mencari *split* terbaik. Kemudian untuk mengukur kualitas *split* pada pohon digunakan nilai *entropy*. Jumlah sampel minimum yang digunakan pada setiap *leaf node* sebanyak dua. Berikut dibawah ini merupakan contoh dari pohon *Random Forest*.



Gambar 5. 9 Contoh Pohon *Random Forest*

Dari **Gambar 5.9** didapatkan hasil pohon *Decision Tree* pada *Random Forest* dengan *variable importance* yang mempengaruhi *churn* ada pada *root node* yaitu variabel *total day minutes*. Kemudian pada *split node* kedua dilakukan oleh variabel *customer service calls*. *Split* node ketiga dilakukan oleh variabel *total day minutes*. Untuk *node* keempat pada *split true* dilakukan oleh variabel *total intl minutes*, dan *split false* dilakukan oleh variabel *total eve minutes*. *Split node* kelima dilakukan oleh variabel *total eve minutes* dan variabel *total intl minutes*. Selanjutnya untuk *split* pada *node* keenam dilakukan oleh variabel *total day minutes*, dan *total eve minutes*. Pada *split node* ketujuh dilakukan oleh variabel *total eve minutes*, dan untuk *split node* kedelapan dilakuka oleh variabel *customer service calls*. Pada salah satu pohon keputusan pada *Random Forest* terdapat delapan kali percabangan, sesuai dengan parameter yang sudah di *tuning* dengan nilai *max_depth* sebanyak 8.

Dapat dilihat pada *decision tree* pada **Gambar 5.9** *customer* akan melakukan *churn* jika:

1. *Customer* melakukan *total day minutes* kurang dari atau sama dengan 75.9 menit
2. *Customer* melakukan *total day minutes* lebih dari 75.9 menit, *customer service calls* lebih dari 3.5, *total day minutes* kurang dari atau sama dengan 192.3 menit, dan *total eve minutes* kurang dari atau sama dengan 172.95 menit.
3. *Customer* melakukan *total day minutes* lebih dari 75.9 menit, *customer service calls* lebih dari 3.5, *total day minutes* kurang dari atau sama dengan 192.3 menit, *total eve minutes* lebih dari 172.95 menit dan kurang dari atau sama dengan 229.9 menit.

Kemudian evaluasi model dilakukan dengan melakukan prediksi terhadap data *testing* menggunakan model pohon keputusan pada *Random Forest* yang terbentuk. Dengan mengevaluasi model digunakan nilai akurasi, *precision*, dan *recall*, serta nilai *AUC*.

Tabel 5. 6 *Confussion Matrix* dari Data *Testing Random Forest*

Data *Test*	Prediksi	
	Churn	*Not Churn*
Churn	61	40
Not Churn	3	563

Pada **Tabel 5.6** dapat dilihat hasil dari *confussion matrix* data *testing*. Dengan ketepatan klasifikasi memprediksi *customer churn* sebanyak 61, dan yang memprediksi salah sebanyak 3 dengan jumlah data untuk kelas *churn* sebanyak 64. Untuk *customer* yang *not churn* dengan prediksi benar sebanyak 563, sedangkan 40 lainnya tidak tepat dalam memprediksi dengan total kelas *not churn* sebanyak 603. Dari tabel *confusion matrix* didapatkan hasil perhitungan pengukuran kinerja algoritma klasifikasi sebagai berikut:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$= \frac{(61 + 563)}{(61 + 563 + 3 + 40)}$$

$$= 0.935532$$

$$= 93.5\%$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$= \frac{61}{(61 + 3)}$$

$$= 0.953$$

$$= 95.3\%$$

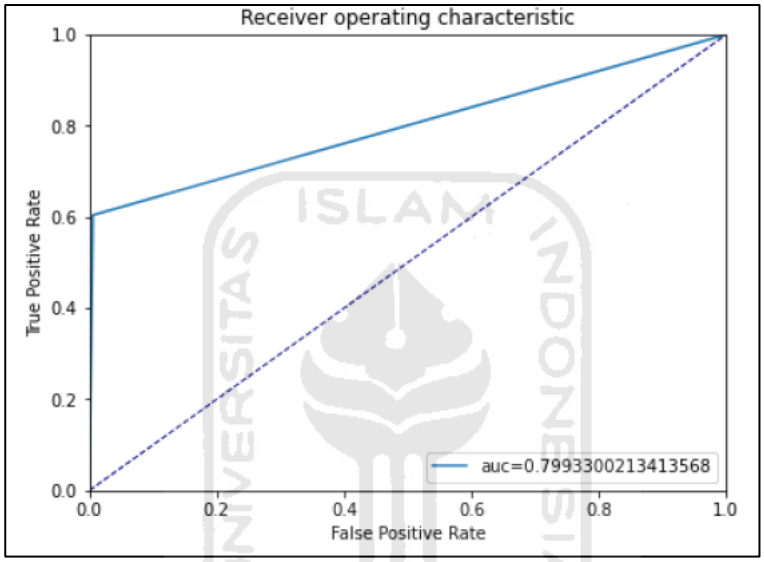
$$Recall = \frac{TP}{(FN + TP)}$$

$$= \frac{61}{(40 + 61)}$$

$$= 0.603$$

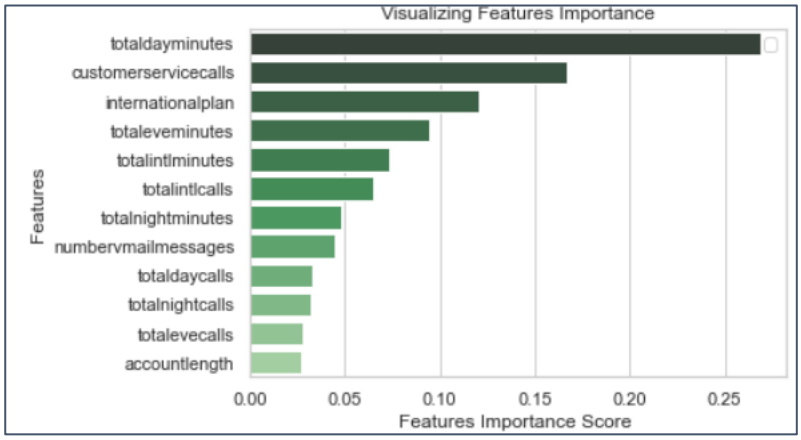
$$= 60.3\%$$

Dari ketiga hasil kinerja algoritma klasifikasi tersebut didapatkan bahwa 93.5% model dapat mengklasifikan *customer* yang melakukan *churn* maupun tidak. Dari total *customer* yang melakukan *churn* sebanyak 95.3% *customer* diidentifikasi benar melakukan *churn*. Untuk nilai *recall* atau pengukuran proporsi *churn* yang diidentifikasi benar sebanyak 60.3%. Berdasarkan hasil tersebut maka klasifikasi sudah baik. Disisi lain, kinerja algoritma dapat dilihat dari kurva *ROC-AUC* seperti yang ditampilkan pada gambar berikut.



Gambar 5. 10 Kurva *ROC Random Forest*

Gambar 5.10 menunjukkan kurva ROC yang menggambarkan hubungan antara data *testing* dan data prediksi. Dari kurva tersebut didapatkan nilai *Area Under the Curve* (AUC) yaitu luas daerah dibawah kurva ROC. Nilai AUC yang diperoleh sebesar 0.799. Dapat dikatakan bahwa hasil klasifikasi yang didapatkan cukup baik. Selanjutnya adalah mengukur kepentingan variabel independen terhadap *customer* berdasarkan nilai *features importance*.



Gambar 5. 11 *Features Importance* Metode *Random Forest*

Gambar 5.11 menunjukkan *features importance* dari variabel-variabel yang digunakan untuk memprediksi *customer churn*. *Feature importance* menunjukkan hubungan variabel yang digunakan dalam mempengaruhi hasil prediksi. Variabel *total day minutes* merupakan variabel yang sangat penting terhadap prediksi *customer churn* dilihat dari nilai *feature importance* yang paling besar, dan variabel *total day minutes* memiliki nilai *information gain* paling besar dilihat dari **Gambar 5.9** variabel *total day minutes* menjadi *root node* pada salah satu pohon keputusan *Random Forest*. Untuk variabel yang memiliki pengaruh paling kecil terhadap prediksi *customer churn* yaitu variabel *account length*. Penentuan *feature importances* didapatkan dari hasil perhitungan *information gain* dari masing-masing variabel, nilai *information gain* dapat dihitung menggunakan nilai *entropy* atau *gini* dari pohon yang dihasilkan. **Tabel 5.7** dibawah ini menampilkan nilai *feautes importance* dari masing-masing variabel.

Tabel 5. 7 Nilai *Features Importance* Metode *Random Forest*

Variabel	***Feature Importance***
totaldayminutes	0.268856
customerservicecalls	0.167072
internationalplan	0.120531
totaleveminutes	0.094122
totalintlminutes	0.073288
totalintlcalls	0.064444
numbervmailmessages	0.048152

Variabel	*Feature Importance*
totalnightminutes	0.044384
totaldaycalls	0.032762
totalnightcalls	0.031602
totalevecalls	0.027718
accountlength	0.027068

5.5 Analisis Klasifikasi *Extreme Gradient Boosting*

Klasifikasi menggunakan metode *Extreme Gradient Boosting* (XGBoost) sama halnya dengan proses klasifikasi menggunakan metode *Random Forest*. Langkah pertama yang dilakukan adalah melakukan *tuning* parameter menggunakan fungsi pada *scikit-learn* yaitu *gridsearchCV*. Hasil dari *tuning* parameter ditampilkan pada tabel berikut.

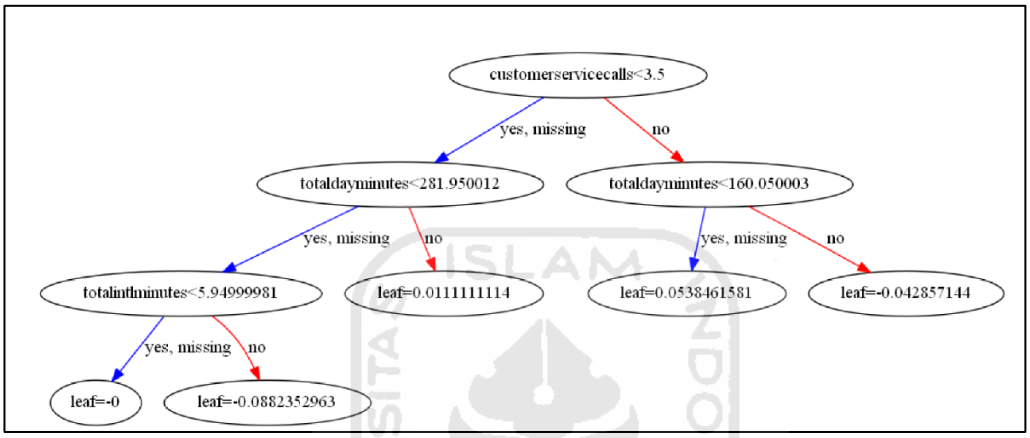
Tabel 5. 8 Hasil *Tuning* Parameter Metode *Extreme Gradient Boosting*

Parameter	*Grid Search Values*	*Best Parameter*
n_estimators	100,200,300	100
max_depth	4,5,6,7,8	6
min_child_weight	1,2,3,4,5,6,7	1
eta (*learning_rate*)	0.025,0.05,0.1,0.2,0.3	0.05
gamma	0,0.1,0.2,0.3,0.4,1.0,1.5, 2.0	1.5
subsample	0.15, 0.5, 0.75, 1.0	1.0
colsample_bylevel	log_2, *sqrt*, *0.25*, *1.0*	1.0

Tabel 5.8 merupakan hasil *tuning* parameter yang didapatkan dari proses *grid searchCV* dengan melakukan pencarian secara menyeluruh terhadap parameter yang di ujikan. Penelitian ini menggunakan 5-*fold cross validation* yang digunakan untuk mengevaluasi kinerja model sebanyak lima kali perulangan dalam proses *grid search* dari setiap parameter. Nilai parameter terbaik dari proses *grid search* digunakan dalam penentuan model klasifikasi.

Pada klasifikasi *Extreme Gradient Boosting* terdapat seratus pohon yang digunakan, kemudian setiap pohon membuat enam percabangan. Untuk jumlah minimum berat yang dibutuhkan pada *child node* bernilai 1. *Eta* digunakan

sebagai tingkat pembelajaran yang mempengaruhi algoritma XGBoost dalam membuat klasifikasi yang berbentuk pohon, dengan menggunakan nilai 0.05. Kemudian minimum *loss* yang dibutuhkan untuk membuat partisi *node* pada *tree* digunakan nilai 1.5. Banyaknya bagian sampel yang dipilih dalam membangun pohon sebanyak satu. Rasio *subsample* kolom untuk setiap level bernilai 1. Berikut dibawah ini merupakan contoh dari pohon *Extreme Gradient Boosting*.



Gambar 5. 12 Contoh Pohon *Extreme Gradient Boosting*

Dari **Gambar 5.12** didapatkan hasil salah satu contoh pohon *XGBoost*. Interpretasi dari pohon diatas adalah sebagai berikut:

1. Jika *customer service calls* kurang dari 3.5, *total day minutes* kurang dari 281.95, dan *total intl minutes* kurang dari 5.949, maka akan mengembalikan -0 untuk pohon selanjutnya, dan yang lain akan mengembalikan -0.0882352963.
2. Jika *customer service call* kurang dari 3.5 dan *total day minutes* lebih dari 281.95 maka akan mengembalikan 0.0111111114.
3. Jika *customer service calls* lebih dari 3.5, dan *total day minutes* lebih dari 160.05 maka akan mengembalikan -0.042857144 dan yang lain akan mengembalikan 0.0538461581.

Kemudian evaluasi model dilakukan dengan melakukan prediksi terhadap data *testing* menggunakan model pohon keputusan pada *Random Forest* yang terbentuk. Dengan mengevaluasi model digunakan nilai akurasi, *precision*, dan *recall*, serta nilai *AUC*.

Tabel 5. 9 *Confusion Matrix* dari Data *Testing Extreme Gradient Boosting*

Data *Test*	Prediksi	
	Churn	*Not Churn*
Churn	77	24
Not Churn	5	561

Pada **Tabel 5.9** dapat dilihat hasil dari *confussion matrix* data *testing* dengan ketepatan klasifikasi memprediksi *customer customer* yang melakukan *churn* dengan prediksi benar sebanyak 77 sedangkan 5 lainnya tidak tepat dalam memprediksi *customer churn* dengan total kelas *churn* sebanyak 82. Untuk *customer* yang *not churn* dengan prediksi benar sebanyak 561, dan yang memprediksi salah sebanyak 24 dengan jumlah data untuk kelas *not churn* sebanyak 585. Dari tabel *confussion matrix* didapatkan hasil untuk perhitungan pengukuran kinerja algoritma klasifikasi sebagai berikut:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$= \frac{(77 + 561)}{(77 + 561 + 5 + 24)}$$

$$= 0.956522$$

$$= 95.6\%$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$= \frac{77}{(77 + 5)}$$

$$= 0.939$$

$$= 94\%$$

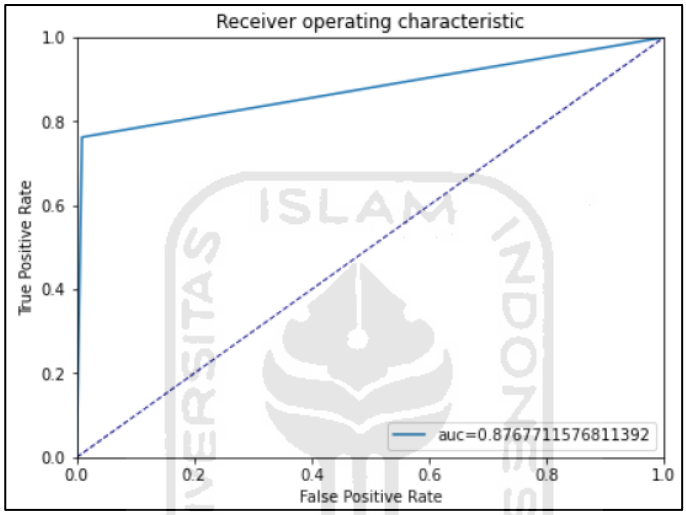
$$Recall = \frac{TP}{(FN + TP)}$$

$$= \frac{77}{(24 + 77)}$$

$$= 0.762$$

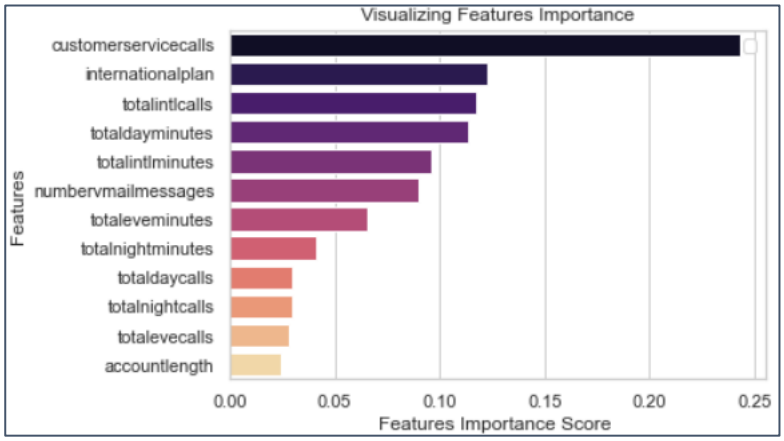
$$= 76.2\%$$

Dari ketiga hasil kinerja algoritma klasifikasi tersebut didapatkan bahwa 95.6% model dapat mengklasifikan *customer* yang melakukan *churn* maupun tidak. Dari total *customer* yang melakukan *churn* 94% *customer* diidentifikasi benar. Untuk nilai *recall* atau pengukuran proporsi *not churn* yang diidentifikasi benar sebanyak 76.2%. Berdasarkan hasil tersebut dapat dikatakan bahwa klasifikasi sudah baik. Disisi lain, kinerja algoritma dapat dilihat dari kurva *ROC-AUC* seperti yang ditampilkan pada gambar berikut.



Gambar 5. 13 Kurva *ROC Extreme Gradient Boosting*

Gambar 5.13 menunjukkan kurva ROC yang menggambarkan hubungan antara data *testing* dan data prediksi. Dari kurva tersebut didapatkan nilai *Area Under the Curve* (AUC) yaitu luas daerah dibawah kurva ROC. Nilai AUC yang diperoleh sebesar 0.876. Dapat dikatakan bahwa hasil klasifikasi yang didapatkan baik. Selanjutnya adalah mengukur kepentingan variabel independen terhadap *customer* berdasarkan nilai *features importance*.



Gambar 5. 14 *Features Importance Extreme Gradient Boosting*

menunjukkan *features importance* dari variabel-variabel yang digunakan untuk memprediksi *customer churn*. *Feature importance* menunjukkan hubungan variabel yang digunakan dalam mempengaruhi hasil prediksi. Variabel *customer service calls* merupakan variabel yang sangat penting terhadap prediksi *customer churn* untuk metode *XGBoost* dilihat dari nilai *feature importance* yang paling besar, dan variabel *customer service calls* memiliki nilai *gain* paling besar dilihat dari **Gambar 5.9** variabel *customer service calls* menjadi *root node* pada salah satu pohon *XGBoost* yang dibangun. Untuk variabel yang memiliki pengaruh paling kecil terhadap prediksi *customer churn* yaitu variabel *account length* sama dengan *feature importances* pada metode *Random Forest*. **Tabel 5.10** dibawah ini menampilkan nilai *feautes importance* dari masing-masing variabel.

Tabel 5. 10 Nilai *Feature Importance Extreme Gradient Boosting*

Variabel	***Feature Importance***
internationalplan	0.243112
customerservicecalls	0.122557
totalintlcalls	0.117296
totaldayminutes	0.113212
totalintlminutes	0.095841
numbervmailmessages	0.089657
totaleveminutes	0.065699
totalnightminutes	0.040940
totaldaycalls	0.029837

Variabel	*Feature Importance*
totalnightcalls	0.029418
totalevecalls	0.027991
accountlength	0.024440

5.6 Perbandingan Metode *Random Forest* dan *Extreme Gradient Boosting*

Setelah melakukan proses analisis menggunakan metode *Random Fores* dan *Extreme Gradient Boosting* dan mendapatkan hasil dari kedua klasifikasi tersebut, maka langkah selanjutnya dilakukan perbandingan antar kedua metode untuk menentukan metode terbaik dengan melihat beberapa nilai pengukuran kinerja dari kedua metode yang digunakan.

Tabel 5. 11 Nilai Akurasi dan AUC Metode *Random Forest* dan *XGBoost*

Metode Klasifikasi	Nilai Akurasi	Nilai AUC
Random Forest	93.5%	0.799
Extreme Gradient Boosting	95.6%	0.876

Berdasarkan **Tabel 5.11** dapat diketahui bahwa kedua metode klasifikasi *Random Forest* dan *Extreme Gradient Boosting*, metode *Extreme Gradient Boosting* mendapatkan nilai akurasi dan AUC lebih tinggi dibandingkan dengan metode *Random Forest* yaitu sebesar 95.6% dan 0.876 yang artinya bahwa klasifikasi data *customer churn* dapat diklasifikasikan dengan baik.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan pada bab sebelumnya dapat ditarik kesimpulan sebagai berikut.

1. Klasifikasi *customer churn* menggunakan metode *Random Forest* dilakukan dengan *tuning parameter* menggunakan *Grid Search CV*, dengan membangun pohon sebanyak 100. Dari 100 pohon yang dibangun dilakukan *majority voting* untuk mendapatkan hasil klasifikasi. Model yang didapatkan bahwa *customer* melakukan *churn* jika *total day minutes* kurang dari atau sama dengan 75.9 menit; *total day minutes* lebih dari 75.9 menit, *customer service calls* lebih dari 3.5, *total day minutes* kurang dari atau sama dengan 192.3 menit, dan *total eve minutes* kurang dari atau sama dengan 172.95 menit; *total day minutes* lebih dari 75.9 menit, *customer service calls* lebih dari 3.5, *total day minutes* kurang dari atau sama dengan 192.3 menit, *total eve minutes* lebih dari 172.95 menit dan kurang dari atau sama dengan 229.9 menit. Untuk klasifikasi menggunakan metode *Extreme Gradient Boosting* dengan melakukan *tuning parameter* menggunakan *Grid Search CV* untuk mendapatkan nilai parameter yang optimal, kemudian membangun 100 pohon untuk klasifikasi *XGBoost* dengan memperhatikan *residuals* pada setiap pohon yang dibangun. Pada metode *XGBoost* dilakukan tahapan *pruning* atau pemangkasan pohon untuk meminimalkan nilai *residuals* di setiap pohonnya hingga didapatkan nilai *residuals* paling kecil. Dari model pohon yang didapatkan hasil *customer* akan *churn* jika *customer service calls* kurang dari 3.5, *total day minutes* kurang dari 281.95, dan *total intl minutes* kurang dari 5.949, maka akan mengembalikan -0 untuk pohon selanjutnya, dan yang lain akan mengembalikan -0.0882352963; *customer service call* kurang dari 3.5 dan *total day minutes* lebih dari 281.95 maka akan mengembalikan 0.0111111114; *customer service calls* lebih dari 3.5,

dan *total day minutes* lebih dari 160.05 maka akan mengembalikan -0.042857144 dan yang lain akan mengembalikan 0.0538461581.

2. Dengan membandingkan metode *Random Forest* dan *XGBoost* didapatkan hasil bahwa metode *Extreme Gradient Boosting* memiliki performa yang lebih baik dibandingkan dengan *bagging Random Forest*. Model *Extreme Gradient Boosting* menghasilkan nilai akurasi sebesar 95.6% dan nilai AUC sebesar 0.876, sedangkan metode *Random Forest* mendapatkan nilai akurasi sebesar 93.5% dan nilai *AUC* sebesar 0.799, dapat diartikan bahwa kasus *churn* pada *customer* di salah satu perusahaan telekomunikasi dapat diklasifikasikan dengan baik atau tepat menggunakan metode *Extreme Gradient Boosting*.

6.2 Saran

Saran yang dapat diberikan pada hasil penelitian ini sebagai bahan perbaikan dan pengembangan untuk penelitian selanjutnya sebagai berikut:

1. Mengimplementasikan kedua metode pada data telekomunikasi dari perusahaan yang ada di Indonesia.
2. Menggunakan teknik *boosting* dan *bagging* lainnya sebagai bahan perbandingan.
3. Menggunakan metode-metode untuk menangani *imbalance idata* seperti SMOTE, *Oversampling*, *Undersampling*, dan lain sebagainya.

DAFTAR PUSTAKA

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Alawfi, K., Hussain, A., & Huang, K. (2016). *Customer Churn Prediction in The Telecommunication Sector Using a Rough Set Approach*. *Neurocomputing*, *237*, 242–254. https://doi.org/10.1016/j.neucom.2016.12.009

Bawono, B., & Wasono, R. (2019). *Perbandingan Metode Random Forest dan Naïve Bayes*. *Seminar Nasional Edusaintek*, 343–348.

Biau, G. (2012). *Anlysis of a Random Forests Model*. *Journal of Machine Learning Research*, *13*, 1063–1095.

Breiman, L. (2001). *Random Forests*. *Machine Learning*, *45*, 5–23. https://doi.org/10.1023/A:1010933404324

Breiman, L., & Cutler, A. (2003). *Manual--Setting Up, Using, and Understanding Random Forest V4.0*. https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf

Cao, A., He, H., Chen, Z., & Zhang, W. (2018). *Performance Evaluation of Machine Learning Approaches for Credit Scoring*. *International Journal of Economics, Finance and Management Sciences*, *6*(6), 255–260. https://doi.org/10.11648/j.ijefm.20180606.12

Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 785–794. https://doi.org/10.1145/2939672.2939785

Claesen, M., & De Moor, B. (2015). *Hyperparameter Search in Machine Learning*. *February*. http://arxiv.org/abs/1502.02127

Dayananda. (2020). *How does XGBoost Work*. https://towardsdatascience.com/how-does-xgboost-work-748bc75c58aa

Doreswamy, & Hemanth, K. S. (2011). *Performance Evaluation of Predictive Engineering Materials Data Sets*. *Artificial Intelligent Systems Ans Machine Learning*, *3*(3), 1–8.

Friedman, J. H. (2001). *Greedy Function Approximation: A Gradient Boosting*

Machine. *Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Gorunescu, F. (2010). *Data Mining Concepts, Models and Techiniques*. Springer International Publishing. https://doi.org/10.1007/978-3-642-19721-5

Govindaraju, R., Simatupang, T., & Samadhi, T. A. (2009). *Perancangan Sistem Prediksi Churn Pelanggan PT*. *Telekomunikasi Seluler Dengan Memanfaatkan Proses Data Mining*. *Jurnal Informatika*, *9*(1). https://doi.org/10.9744/informatika.9.1.33-42

Gupta, S. K., Rajeev, B., & Kaur, D. (2015). *Review of Decision Tree Data mining Algorithms*: *ID3 and C4.5*. *Proceeding of International Conference on Information Technology and Computer Science*.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Data Mining Concepts And Techniques* (Third). Morgan Kaufmann Publishers.

Hanif, I. (2019). *Implementing Extreme Gradient Boosting (XGBoost) Classifier to Improve Customer Churn Prediction*. *International Conference on Statistics and Analytics*. https://doi.org/10.4108/eai.2-8-2019.2290338

Hashmi, N., Butt, N. A., & Iqbal, M. (2013). *Customer Churn Prediction in Telecommunication: A Decade Review and Classification*. *IJCSI International Journal of Computer Science Issues*, *10*(5), 271–282.

Lazarov, V., & Capota, M. (2007). *Employee Churn Prediction*. *Expert Systems with Applications*, *38*(3).

Li, X., & Li, Z. (2019). *A Hybrid Prediction Model for E-Commerce Customer Churn Based on Logistic Regression and Extreme Gradient Boosting Algorithm*. *Ingenierie Des Systemes d'Information*, *24*(5), 525–530. https://doi.org/10.18280/isi.240510

Lu, J., & Ph, D. (2002). *Predicting Customer Churn in the Telecommunications Industry*. *An Application of Survival Analysis Modeling Using SAS*, *114–27*, 114–27. http://www2.sas.com/proceedings/sugi27/p114-27.pdf

Malik, S., Harode, R., & Kunwar, A. S. (2020). *XGBoost: A Deep Dive into Boosting*. *February*. https://doi.org/10.13140/RG.2.2.15243.64803

Masarifoglu, M., & Buyuklu, A. H. (2019). *Applying Survival Analysis to*

Telecom Churn Data. *American Journal of Theoretical and Applied Statistics*, *8*(6), 261–275. https://doi.org/10.11648/j.ajtas.20190806.18

Mo, H., Sun, H., Liu, J., & Wei, S. (2019). *Developing Window Behavior Models for Residential Buildings Using XGBoost Algorithm*. *Energy and Buildings*, *205*, 109564. https://doi.org/10.1016/j.enbuild.2019.109564

Murty, S. V., & Kumar, R. K. (2019). *Accurate Liver Disease Prediction with Extreme Gradient Boosting*. *International Journal of Engineering and Advanced Technology (IJEAT)*, *8*(6), 2288–2295. https://doi.org/10.35940/ijeat.F8684.088619

Pamina, J., Beschi Raja, J., Sathya Bama, S., Soundarya, S., Sruthi, M. S., Kiruthika, S., Aiswaryadevi, V. J., & Priyanka, G. (2019). A*n Effective Classifier for Predicting Churn in Telecommunication*. *Journal of Advanced Research in Dynamical and Control Systems*, *11*(1 Special Issue), 221–229.

R, A. (2018). *Applying Random Forest (Classification) — Machine Learning Algorithm from Scratch with Real Datasets*. https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57

Ramadhan, M. M., Sitanggang, I. S., Nasution, F. R., & Ghifari, A. (2017). *Parameter Tuning in Random Forest Based on Grid Search Method for Gender Classification Based on Voice Frequency*. *DEStech Transactions on Computer Science and Engineering*, 625–629. https://doi.org/10.12783/dtcse/cece2017/14611

S, I. M., Kusuma, W. A., & Wahjuni, S. (2019). *Comparation Analysis of Ensemble Technique with Boosting(Xgboost) and Bagging(Randomforest) for Classify Splice Junction DNA Sequence Category*. *Jurnal Penelitian Pos Dan Informatika*, *9*(1), 27–36. https://doi.org/10.17933/jppi.2019.090

Sabbeh, S. F. (2018). *Machine-Learning Techniques for Customer Retention: A Comparative Study*. *International Journal of Advanced Computer Science and Applications*, *9*(2), 273–281. https://doi.org/10.14569/IJACSA.2018.090238

Sahu, H., Shrma, S., & Gondhalakar, S. (2008). *A Brief Overview on Data Mining*

Survey. *Ijctee*, *1*(3), 114–121.

Sayad, S. (n.d.). *Decision Tree - Classification*. https://www.saedsayad.com/decision_tree.htm

scikit-learn.org. (n.d.). *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier*. Retrieved September 6, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sidhu, R. (2019). *Building Intuition for Random Forests*. https://medium.com/x8-the-ai-community/building-intuition-for-random-forests-76d36fa28c5e

Silipo, R. (2019). *From a Single Decision Tree to a Random Forest*. https://www.dataversity.net/from-a-single-decision-tree-to-a-random-forest/

Simon, A., Deo, M. S., Venkatesan, S., & Babu, R. D. R. (2015). An Overview of *Deep Learning and Its Applications*. *International Journal of Electrical Sciences & Engineering (IJESE)*, *1*(1), 22–24.

Sokolova, M., & Lapalme, G. (2009). *A Systematic Analysis of Performance Measures for Classification Tasks*. *Information Processing and Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Sumathi, K., Kannan, S., & Nagarajan, K. (2016). *Data Mining: Analysis of Student Database Using Classification Techniques*. *International Journal of Computer Applications*, *141*(8), 22–27. https://doi.org/10.5120/ijca2016909703

Syahrani, I. M. (2019). Analisis Pembandingan Teknik *Ensemble* Secara *Boosting(XGBoost)* Dan *Bagging (Random Forest)* Pada Klasifikasi Kategori Sambatan *Sekuens* DNA. *Jurnal Penelitian Pos Dan Informatika*, *9*(1), 27. https://doi.org/10.17933/jppi.2019.090103

Syarif, I., Zaluska, E., Prugel-Bennett, A., & Wills, G. (2012). *Application Of Bagging, Boosting and Stacking to Intrusion Detection*. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7376 LNAI*, 593–602. https://doi.org/10.1007/978-3-642-31537-4_46

Thankachan, S., & Suchithra. (2017). *Data Mining & Warehousing Algorithms*

and its Application in Medical Science - A Survey. *International Journal of Computer Science and Mobile Computing*, *6*(3), 160–168.

The Royal Society. (2017). *Machine Learning: The Power and Promise of Computers That Learn by Example*. *In Report By The Royal Society*.

Tuv, E. (2006). *Ensemble learning*. *Studies in Fuzziness and Soft Computing*, *207*, 187–204. https://doi.org/10.4249/scholarpedia.2776

Wearesocial, & Hootsuite. (n.d.). *Digital 2019 Indonesia*. https://wearesocial.com/global-digital-report-2019

Wu, Z., Li, N., Peng, J., Cui, H., Liu, P., Li, H., & Li, X. (2018). *Using An Ensemble Machine Learning Methodology-Bagging to Predict Occupants' Thermal Comfort in Buildings*. *Energy and Buildings*, *173*(January 2019), 117–127. https://doi.org/10.1016/j.enbuild.2018.05.031

Xgboost.readthedocs.io. (n.d.). *XGBoost Documentation*. https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Yaman, E., & Subasi, A. (2019). *Comparison of Bagging and Boosting Ensemble Machine Learning Methods for Automated EMG Signal Classification*. *BioMed Research International*, *2019*. https://doi.org/10.1155/2019/9152506

Zanin, M., Papo, D., Sousa, P. A., Menasalvas, E., Nicchi, A., Kubik, E., & Boccaletti, S. (2016). *Combining Complex Networks and Data Mining: Why and How*. *Physics Reports*, *635*, 1–44. https://doi.org/10.1016/j.physrep.2016.04.005

LAMPIRAN

Lampiran 1 *Data Telecommunication Churn*

No	*State*	*Account Length*	*Area Code*	*Phone Number*	*International Plan*	*Voicemail Plan*	*Number Vmail Messages*	*Total Day Minutes*	*Total Day Calls*	*Total Day Charge*
1	KS	128	415	382-4657	no	yes	25	265.1	110	45.07
2	OH	107	415	371-7191	no	yes	26	161.6	123	27.47
3	NJ	137	415	358-1921	no	no	0	243.4	114	41.38
4	OH	84	408	375-9999	yes	no	0	299.4	71	50.9
5	OK	75	415	330-6626	yes	no	0	166.7	113	28.34
6	AL	118	510	391-8027	yes	no	0	223.4	98	37.98
7	MA	121	510	355-9993	no	yes	24	218.2	88	37.09
8	MO	147	415	329-9001	yes	no	0	157	79	26.69
9	LA	117	408	335-4719	no	no	0	184.5	97	31.37
10	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96
11	IN	65	415	329-6603	no	no	0	129.1	137	21.95
12	RI	74	415	344-9403	no	no	0	187.7	127	31.91
13	IA	168	408	363-1107	no	no	0	128.8	96	21.9

14	MT	95	510	394-8006	no	no	0	156.6	88	26.62
15	IA	62	415	366-9238	no	no	0	120.7	70	20.52
:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:
3319	OK	52	415	397-9928	no	no	0	124.9	131	21.23
3320	WY	89	415	378-6924	no	no	0	115.4	99	19.62
3321	GA	122	510	411-5677	yes	no	0	140	101	23.8
3322	VT	60	415	400-2738	no	no	0	193.9	118	32.96
3323	MD	62	408	409-1856	no	no	0	321.1	105	54.59
3324	IN	117	415	362-5899	no	no	0	118.4	126	20.13
3325	WV	159	415	377-1164	no	no	0	169.8	114	28.87
3326	OH	78	408	368-8555	no	no	0	193.4	99	32.88
3327	OH	96	415	347-6812	no	no	0	106.6	128	18.12
3328	SC	79	415	348-3830	no	no	0	134.7	98	22.9
3329	AZ	192	415	414-4276	no	yes	36	156.2	77	26.55
3330	WV	68	415	370-3271	no	no	0	231.1	57	39.29
3331	RI	28	510	328-8230	no	no	0	180.8	109	30.74
3332	CT	184	510	364-6381	yes	no	0	213.8	105	36.35
3333	TN	74	415	400-4344	no	yes	25	234.4	113	39.85

No	*Total Eve Minutes*	*Total Eve Calls*	*Total Eve Charge*	*Total Night Minutes*	*Total Night Calls*	*Total Night Charge*	*Total Intl Minutes*	*Total Intl Calls*	*Total Intl Charge*	*Customer Service Calls*	*Churn*
1	197.4	99	16.78	244.7	91	11.01	10	3	2.7	1	no
2	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7	1	no
3	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29	0	no
4	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	no
5	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	no
6	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7	0	no
7	348.5	108	29.62	212.6	118	9.57	7.5	7	2.03	3	no
8	103.1	94	8.76	211.8	96	9.53	7.1	6	1.92	0	no
9	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35	1	no
10	222	111	18.87	326.4	97	14.69	11.2	5	3.02	0	no
11	228.5	83	19.42	208.8	111	9.4	12.7	6	3.43	4	yes
12	163.4	148	13.89	196	94	8.82	9.1	5	2.46	0	no
13	104.9	71	8.92	141.1	128	6.35	11.2	2	3.02	1	no
14	247.6	75	21.05	192.3	115	8.65	12.3	5	3.32	3	no

No	*Total Eve Minutes*	*Total Eve Calls*	*Total Eve Charge*	*Total Night Minutes*	*Total Night Calls*	*Total Night Charge*	*Total Intl Minutes*	*Total Intl Calls*	*Total Intl Charge*	*Customer Service Calls*	*Churn*
15	307.2	76	26.11	203	99	9.14	13.1	6	3.54	4	no
:	:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:	:
3319	300.5	118	25.54	192.5	106	8.66	11.6	4	3.13	300.5	118
3320	209.9	115	17.84	280.9	112	12.64	15.9	6	4.29	209.9	115
3321	196.4	77	16.69	120.1	133	5.4	9.7	4	2.62	196.4	77
3322	85	110	7.23	210.1	134	9.45	13.2	8	3.56	85	110
3323	265.5	122	22.57	180.5	72	8.12	11.5	2	3.11	265.5	122
3324	249.3	97	21.19	227	56	10.22	13.6	3	3.67	249.3	97
3325	197.7	105	16.8	193.7	82	8.72	11.6	4	3.13	197.7	105
3326	116.9	88	9.94	243.3	109	10.95	9.3	4	2.51	116.9	88
3327	284.8	87	24.21	178.9	92	8.05	14.9	7	4.02	284.8	87
3328	189.7	68	16.12	221.4	128	9.96	11.8	5	3.19	189.7	68
3329	215.5	126	18.32	279.1	83	12.56	9.9	6	2.67	215.5	126
3330	153.4	55	13.04	191.3	123	8.61	9.6	4	2.59	153.4	55
3331	288.8	58	24.55	191.9	91	8.64	14.1	6	3.81	288.8	58

No	*Total Eve Minutes*	*Total Eve Calls*	*Total Eve Charge*	*Total Night Minutes*	*Total Night Calls*	*Total Night Charge*	*Total Intl Minutes*	*Total Intl Calls*	*Total Intl Charge*	*Customer Service Calls*	*Churn*
3332	159.6	84	13.57	139.2	137	6.26	5	10	1.35	159.6	84
3333	265.9	82	22.6	241.4	77	10.86	13.7	4	3.7	265.9	82

Lampiran 2 *Script* Penelitian

```
#Importing libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from math import * # module math
import matplotlib.pyplot as plt # visualization
import plotly.offline as py # visualization
import plotly.express as px
from matplotlib import rcParams
import seaborn as sns # visualization
import itertools
import io
import plotly.figure_factory as ff # visualization
import warnings
warnings.filterwarnings("ignore")
import os
%matplotlib inline
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, log_loss, fbeta_score
from sklearn.metrics import auc, roc_curve, roc_auc_score,
precision_recall_curve
from sklearn.preprocessing import LabelEncoder

from sklearn import tree
from sklearn.tree import export_graphviz
from IPython.display import Image
from xgboost import plot_tree
from matplotlib.pylab import rcParams

telcom=pd.read_csv('deskriptif.csv', delimiter=";")
telcom.head()

telcom['churn'].value_counts()

chroma = ["#BDFCC9","#FFDEAD"]
plt.pie(telcom["churn"].value_counts(),explode=(0,0.1), autopct='%1.1f%%',
        shadow=True, colors=chroma,
startangle=90,labels=telcom["churn"].unique())
plt.title('Percentage of Churn')
plt.axis('equal') ;

sns.countplot(data=telcom,
                  x = 'churn',
                  hue = 'internationalplan')
plt.title('International Plan of Customer Churn')
plt.show()

p = sns.countplot(data=telcom,
                  hue = 'customerservicecalls',
                  x = 'churn')
```

```
plt.title('Customer Service Calls of Customer Churn')
#PREPOCESSING DATA
from sklearn import preprocessing
#Removing correlated and unneccessary columns
col_to_drop = ['state','phonenumber','areacode']
telcom = telcom.drop(columns = col_to_drop, axis = 1)
#target column
target_col = ["churn"]


#number of levels in feature to be a categorical feature
nlevels = 2

#Separating categorical and numerical columns
#categorical columns
cat_cols =
list(set(telcom.nunique()[telcom.nunique()<nlevels].keys().tolist()
                    +
telcom.select_dtypes(include='object').columns.tolist()))
cat_cols = [x for x in cat_cols if x not in target_col]
#numerical columns
num_cols = [x for x in telcom.columns if x not in cat_cols + target_col]
#Binary columns with 2 values
bin_cols = telcom.nunique()[telcom.nunique() == 2].keys().tolist()

#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols:
    telcom[i] = le.fit_transform(telcom[i])
    #defining the studied or used independent features (columns) as well the
target

#get correlations of each features in dataset
corrmat = telcom.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(15,7))
#plot heat map
g=sns.heatmap(telcom[top_corr_features].corr(),annot=True,cmap="RdYlGn")

#Removing correlated and unneccessary columns
col_to_drop =
['totaldaycharge','totalevecharge','totalnightcharge','totalintlcharge','voic
emailplan']
telcom = telcom.drop(columns = col_to_drop, axis = 1)

telcom.to_csv('terakhir1.csv',index=False)
telcom=pd.read_csv('terakhir1.csv')
#defining the studied or used independent features (columns) as well the
target
cols = [i for i in telcom.columns if i not in target_col]
target_col = ['churn']

#membagi data testing dan training
X_train, X_test, y_train, y_test = train_test_split(telcom[cols],
telcom[target_col],
                                                    test_size = .20,
random_state = 42)
```

```
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))

def apply_classifier(clf,xTrain,xTest,yTrain,yTest):

    clf.fit(xTrain, yTrain)
    predictions = clf.predict(xTest)
    probabilities = clf.predict_proba(xTest)

    print('Algorithm:', type(clf).__name__)
    print("\nClassification report:\n", classification_report(yTest,
predictions))
    print("Accuracy Score:", accuracy_score(yTest, predictions))

    conf_mtx = confusion_matrix(yTest,predictions)
    f, axes = plt.subplots(ncols=2, figsize=(15, 5))
    sns.heatmap(conf_mtx,annot=True,cmap='tab20c',cbar = False,fmt = "g",ax =
axes[0])
    axes[0].set_xlabel('Predicted labels')
    axes[0].set_ylabel('True labels')
    axes[0].set_title('Confusion Matrix');
    axes[0].xaxis.set_ticklabels(['Not Churn', 'Churn']);
    axes[0].yaxis.set_ticklabels(['Not Churn', 'Churn']);


    roc_auc = roc_auc_score(yTest,predictions)
    print ("Area under ROC curve : ",roc_auc,"\n")

    fpr, tpr,_ = roc_curve(yTest, predictions)
    axes[1].plot(fpr,tpr,label= "auc="+str(roc_auc));
    axes[1].plot([0, 1], [0, 1], color='navy', lw=1, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

##KLASIFIKASI RANDOM FOREST
#tuning parameter
rf=RandomForestClassifier(random_state=42)

RF2_parameters={'max_features': ['auto', 'sqrt', 'log2']}
grid_obj = GridSearchCV(rf,RF2_parameters,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_

RF1_parameters={'n_estimators': [100,200,300]}
grid_obj = GridSearchCV(rf,RF1_parameters,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_

RF3_parameters={'max_depth' : [4,5,6,7,8]}
grid_obj = GridSearchCV(rf,RF3_parameters,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
```

```
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_

RF4_parameters={"min_samples_leaf" :[1,2,3,4,5,6,7,8,9,10]}
grid_obj = GridSearchCV(rf,RF4_parameters,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_

RF5_parameters={"criterion" :['entropy','gini']}
grid_obj = GridSearchCV(rf,RF5_parameters,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_

#model klasifikasi RF
RF_grid=RandomForestClassifier(n_estimators=100, criterion='entropy',
                               max_depth=8,max_features='sqrt',
min_samples_leaf=2)
apply_classifier(RF_grid,X_train, X_test, y_train, y_test)

#membangun pohon
RF_grid.estimators_
print(len(RF_grid.estimators_))
estimator_limited=RF_grid.estimators_[0]
export_graphviz(estimator_limited, out_file=akhirend.dot', feature_names =
cols,
                class_names =["Not churn","Churn"],
                rounded = True, proportion = False, precision = 2, filled =
True)
os.system('dot -Tpng akhirend.dot -o akhirend.png')
Image(filename = akhirend.png')

#feature importance
feature_imp =
pd.Series(RF_grid.feature_importances_,index=cols).sort_values(ascending=Fals
e)
feature_imp

# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score ')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()

#KLASIFIKASI XGBOOST
#optimasi parameter
xg_boost = XGBClassifier()

Xgboost_parameters8 = {"n_estimators":[100,200,300]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters8,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
Xgboost_parameters2 = { "max_depth" : [4,5,6,7,8]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters2,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

Xgboost_parameters3 = {"min_child_weight" : [0,1,2,3,4,5,6,7]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters3,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

Xgboost_parameters4 = {"learning_rate" : [0.025, 0.05, 0.1, 0.2, 0.3]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters4,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

Xgboost_parameters5 = {"gamma": [0, 0.1, 0.2, 0.3, 0.4, 1.0, 1.5, 2.0]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters5,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

Xgboost_parameters7 = { "colsample_bylevel" : ['log2', 'sqrt', 0.25, 1.0]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters7,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

Xgboost_parameters8 = {"subsample":[0.15, 0.5, 0.75, 1.0]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters8,n_jobs=-
1,cv=5,verbose=3,scoring='roc_auc')
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_

#model xgboost
XGBGrid = XGBClassifier(n_estimators=100,colsample_bylevel= 1,
                        gamma= 1.5, learning_rate= 0.05, max_depth= 6,
min_child_weight=1,
                        subsample=1.0)
apply_classifier(XGBGrid,X_train, X_test, y_train, y_test)

#visualisasi tree
plot_tree(XGBGrid)
fig = plt.gcf()
fig.set_size_inches(100, 150)
plt.savefig('endbanget.png')

#feature importance
feature_XGB =
pd.Series(XGBGrid.feature_importances_,index=cols).sort_values(ascending=Fals
e)
feature_XGB

# Creating a bar plot
```

```
sns.barplot(x=feature_XGB, y=feature_XGB.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

Lampiran 3 *Output* Analisis Deskriptif

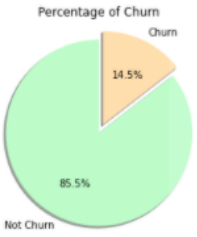
In [2]:
```
telcom=pd.read_csv('deskriptif.csv', delimiter=";")
telcom.head()
```

Out[2]:

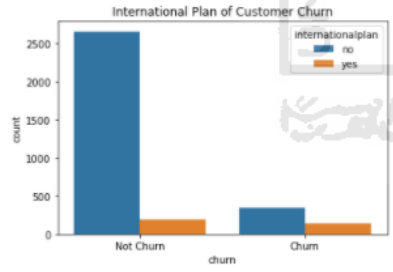
	state	accountlength	areacode	phonenumber	internationalplan	voicemailplan	numbervmailmessages	totaldayminutes	totaldaycalls	totaldaycharge	...	tc
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	

5 rows × 21 columns

In [4]:
```
chroma = ["#BDFCC9","#FFDEAD"]
plt.pie(telcom["churn"].value_counts(),explode=(0,0.1), autopct='%1.1f%%',
        shadow=True, colors=chroma, startangle=90,labels=telcom["churn"].unique())
plt.title('Percentage of Churn')
plt.axis('equal') ;
```

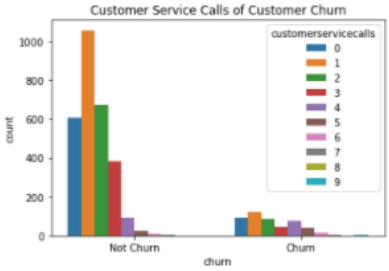


In [5]:
```
sns.countplot(data=telcom,
              x = 'churn',
              hue = 'internationalplan')
plt.title('International Plan of Customer Churn')
plt.show()
```



In [6]:
```
p = sns.countplot(data=telcom,
                  hue = 'customerservicecalls',
                  x = 'churn')
plt.title('Customer Service Calls of Customer Churn')
```

Out[6]: `Text(0.5, 1.0, 'Customer Service Calls of Customer Churn')`



In [9]:
```
summary = (telcom[[i for i in telcom.columns]].
           describe().transpose().reset_index())

summary = summary.rename(columns = {"index" : "feature"})
summary = np.around(summary,3)

val_lst = [summary['feature'], summary['count'],
           summary['mean'],summary['std'],
           summary['min'], summary['25%'],
           summary['50%'], summary['75%'], summary['max']]

trace  = go.Table(header = dict(values = summary.columns.tolist(),
                                line = dict(color = ['#506784']),
                                fill = dict(color = ['#119DFF']),
                               ),
                  cells  = dict(values = val_lst,
                                line = dict(color = ['#506784']),
                                fill = dict(color = ["lightgrey",'#F5F8FF'])
                               ),
                  columnwidth = [200,60,100,100,60,60,80,80,80])
layout = go.Layout(dict(title = "Variable Summary"))
figure = go.Figure(data=[trace],layout=layout)
py.iplot(figure)
```

Variable Summary

feature	count	mean	std	min	25%	50%	75%	max
accountlength	3333	101.065	39.822	1	74	101	127	243
internationalplan	3333	0.097	0.296	0	0	0	0	1
numbervmailmessages	3333	8.099	13.688	0	0	0	20	51
totaldayminutes	3333	179.775	54.467	0	143.7	179.4	216.4	350.8
totaldaycalls	3333	100.436	20.069	0	87	101	114	165
totaleveminutes	3333	200.98	50.714	0	166.6	201.4	235.3	363.7
totalevecalls	3333	100.114	19.923	0	87	100	114	170
totalnightminutes	3333	200.872	50.574	23.2	167	201.2	235.3	395
totalnightcalls	3333	100.108	19.569	33	87	100	113	175
totalintlminutes	3333	10.237	2.792	0	8.5	10.3	12.1	20
totalintlcalls	3333	4.479	2.461	0	3	4	6	20
customerservicecalls	3333	1.563	1.315	0	1	1	2	9
churn	3333	0.855	0.352	0	1	1	1	1

Lampiran 4 *Output* Klasifikasi *Random Forest*

```
In [64]:
grid_obj = GridSearchCV(rf,RF2_parameters,n_jobs=-1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  15 | elapsed:    3.8s remaining:
5.7s
[Parallel(n_jobs=-1)]: Done  12 out of  15 | elapsed:    4.5s remaining:
1.1s
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:    4.5s finished
```

```
Out[64]:

({'mean_fit_time': array([0.65495849, 0.63333616, 0.56232123]),
  'std_fit_time': array([0.01181137, 0.04911255, 0.01408942]),
  'mean_score_time': array([0.03561649, 0.0436914 , 0.02894773]),
  'std_score_time': array([0.01316107, 0.01703193, 0.0023748 ]),
  'param_max_features': masked_array(data=['auto', 'sqrt', 'log2'],
               mask=[False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'max_features': 'auto'},
   {'max_features': 'sqrt'},
   {'max_features': 'log2'}],
  'split0_test_score': array([0.94130268, 0.94249623, 0.93590327]),
  'split1_test_score': array([0.88765404, 0.89894046, 0.89214557]),
  'split2_test_score': array([0.92106703, 0.91448808, 0.9137107 ]),
  'split3_test_score': array([0.83885178, 0.84665438, 0.85622769]),
  'split4_test_score': array([0.91322055, 0.91283607, 0.90793746]),
  'mean_test_score': array([0.90041921, 0.90308304, 0.90118494]),
  'std_test_score': array([0.03525667, 0.03155931, 0.02649799]),
  'rank_test_score': array([3, 1, 2])},
 {'max_features': 'sqrt'},
 0.9030830445419298)
```

```
In [65]:
grid_obj = GridSearchCV(rf,RF1_parameters,n_jobs=-1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  15 | elapsed:    1.6s remaining:
2.4s
[Parallel(n_jobs=-1)]: Done  12 out of  15 | elapsed:    3.1s remaining:
0.7s
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:    3.6s finished
```

Out[65]:

```
({'mean_fit_time': array([0.82034001, 1.64551153, 2.05642381]),
  'std_fit_time': array([0.07142178, 0.12833212, 0.18052724]),
  'mean_score_time': array([0.05562534, 0.07879171, 0.06521282]),
  'std_score_time': array([0.01145028, 0.0377317 , 0.01952058]),
  'param_n_estimators': masked_array(data=[100, 200, 300],
               mask=[False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'n_estimators': 100},
   {'n_estimators': 200},
   {'n_estimators': 300}],
  'split0_test_score': array([0.93428344, 0.94421552, 0.9384893 ]),
  'split1_test_score': array([0.90259703, 0.89257745, 0.88556662]),
  'split2_test_score': array([0.9190228 , 0.91356674, 0.92043361]),
  'split3_test_score': array([0.84524358, 0.84541633, 0.84509962]),
  'split4_test_score': array([0.91253702, 0.90951811, 0.9085925 ]),
  'mean_test_score': array([0.90273678, 0.90105883, 0.89963633]),
  'std_test_score': array([0.03053509, 0.03243001, 0.03223198]),
  'rank_test_score': array([1, 2, 3])},
 {'n_estimators': 100},
 0.9027367752981019)
```

In [66]:

```
RF3_parameters={'max_depth' : [4,5,6,7,8]}
grid_obj = GridSearchCV(rf,RF3_parameters,n_jobs=-1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  19 out of  25 | elapsed:    1.6s remaining:
0.4s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    2.0s finished
```

Out[66]:

```
({'mean_fit_time': array([0.57292333, 0.55409117, 0.48303347, 0.52719822, 0.
47834644]),
  'std_fit_time': array([0.01331663, 0.08979203, 0.0132213 , 0.00709516, 0.0
6326343]),
  'mean_score_time': array([0.03830204, 0.03214955, 0.03865838, 0.02768173,
0.01956177]),
  'std_score_time': array([0.01331603, 0.0082121 , 0.01111408, 0.00102191,
0.00398913]),
  'param_max_depth': masked_array(data=[4, 5, 6, 7, 8],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'max_depth': 4},
   {'max_depth': 5},
   {'max_depth': 6},
   {'max_depth': 7},
   {'max_depth': 8}],
  'split0_test_score': array([0.92193583, 0.92406718, 0.93043281, 0.9350649
4, 0.93628691]),
  'split1_test_score': array([0.91563976, 0.91860532, 0.91854774, 0.9130484
9, 0.91641714]),
  'split2_test_score': array([0.88402626, 0.8902741 , 0.8977312 , 0.9055338
, 0.90757803]),
  'split3_test_score': array([0.84541633, 0.84118392, 0.8469423 , 0.8461937
1, 0.85451457]),
  'split4_test_score': array([0.91005924, 0.9156129 , 0.91643882, 0.9092902
7, 0.91592618]),
  'mean_test_score': array([0.89541548, 0.89794868, 0.90201857, 0.90182624,
0.90614456]),
  'std_test_score': array([0.02812977, 0.03066605, 0.02946287, 0.02964856,
0.02748628]),
  'rank_test_score': array([5, 4, 2, 3, 1])},
 {'max_depth': 8},
 0.9061445637168314)
```

In [67]:

```
RF4_parameters={"min_samples_leaf" :[1,2,3,4,5,6,7,8,9,10]}
grid_obj = GridSearchCV(rf,RF4_parameters,n_jobs=-1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_ , grid_fitRF.best_params_ , grid_fitRF.best_score_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    1.5s
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed:    4.6s finished
```

Out[67]:

```
({'mean_fit_time': array([0.80080166, 0.669736  , 0.61203866, 0.71071334, 0.
68380842,
        0.57632942, 0.64098144, 0.69300447, 0.61182895, 0.53529758]),
  'std_fit_time': array([0.03419261, 0.05530361, 0.02404333, 0.05367761, 0.0
5070835,
        0.01241426, 0.05590531, 0.03566588, 0.01183903, 0.05829783]),
  'mean_score_time': array([0.03789792, 0.03151693, 0.05211058, 0.03841329,
0.03183446,
        0.03705611, 0.03860078, 0.03785591, 0.03482113, 0.02752061]),
  'std_score_time': array([0.00950232, 0.00474527, 0.01054403, 0.00760487,
0.00314214,
        0.00773218, 0.0056044 , 0.00566689, 0.00596488, 0.01055281]),
  'param_min_samples_leaf': masked_array(data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 1
0],
               mask=[False, False, False, False, False, False, False, False,
                     False, False],
        fill_value='?',
             dtype=object),
  'params': [{'min_samples_leaf': 1},
   {'min_samples_leaf': 2},
   {'min_samples_leaf': 3},
   {'min_samples_leaf': 4},
   {'min_samples_leaf': 5},
   {'min_samples_leaf': 6},
   {'min_samples_leaf': 7},
   {'min_samples_leaf': 8},
   {'min_samples_leaf': 9},
   {'min_samples_leaf': 10}],
  'split0_test_score': array([0.93775043, 0.94343403, 0.94245361, 0.9435903
3, 0.94029384,
        0.94430078, 0.93969706, 0.93478076, 0.93594589, 0.93983915]),
  'split1_test_score': array([0.89336923, 0.89836462, 0.89668029, 0.8933980
2, 0.90815386,
        0.89873892, 0.90752044, 0.89254866, 0.90115743, 0.91267419]),
  'split2_test_score': array([0.91699297, 0.9201169 , 0.90867212, 0.9045980
7, 0.90829782,
        0.90705977, 0.90933433, 0.91322124, 0.91273177, 0.90622481]),
  'split3_test_score': array([0.8444518 , 0.85838708, 0.86418864, 0.8459633
8, 0.84366003,
        0.85517678, 0.84771968, 0.85653   , 0.84826673, 0.85111713]),
  'split4_test_score': array([0.91126965, 0.914374  , 0.91245158, 0.9094896
3, 0.91629642,
        0.90883459, 0.9117111 , 0.90963203, 0.90900547, 0.91245158]),
  'mean_test_score': array([0.90076682, 0.90693532, 0.90488925, 0.89940788,
0.9033404 ,
        0.90282217, 0.90319652, 0.90134254, 0.90142146, 0.90446137]),
  'std_test_score': array([0.03151981, 0.02825467, 0.02532393, 0.0315376 ,
0.03207151,
        0.02850031, 0.03012918, 0.02612826, 0.02899266, 0.02909158]),
  'rank_test_score': array([ 9,  1,  2, 10,  4,  6,  5,  8,  7,  3])},
 {'min_samples_leaf': 2},
 0.9069353249345763)
```

In [68]:

```
grid_obj = GridSearchCV(rf,RFS_parameters,n_jobs=-1,cv=5,verbose=3,scoring='roc_auc')
grid_fitRF = grid_obj.fit(X_train,y_train)
grid_fitRF.cv_results_, grid_fitRF.best_params_, grid_fitRF.best_score_
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  10 | elapsed:    0.8s remaining:
2.0s
[Parallel(n_jobs=-1)]: Done   7 out of  10 | elapsed:    0.8s remaining:
0.3s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    1.2s finished
```

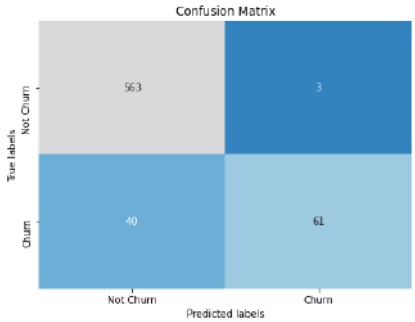
Out[68]:

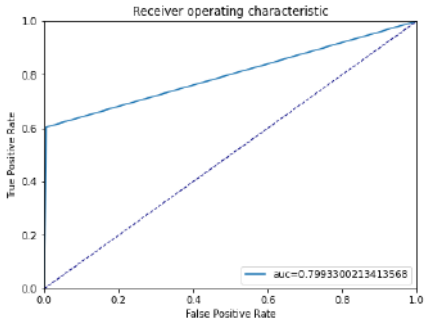
```
({'mean_fit_time': array([0.88757663, 0.64700689]),
  'std_fit_time': array([0.02357159, 0.18934468]),
  'mean_score_time': array([0.03733249, 0.02571001]),
  'std_score_time': array([0.01057452, 0.00750304]),
  'param_criterion': masked_array(data=['entropy', 'gini'],
               mask=[False, False],
         fill_value='?',
              dtype=object),
  'params': [{'criterion': 'entropy'}, {'criterion': 'gini'}],
  'split0_test_score': array([0.94667368, 0.94175737]),
  'split1_test_score': array([0.90549061, 0.89938673]),
  'split2_test_score': array([0.92083669, 0.9165323 ]),
  'split3_test_score': array([0.85602614, 0.84121271]),
  'split4_test_score': array([0.90742481, 0.91071429]),
  'mean_test_score': array([0.90729039, 0.90192068]),
  'std_test_score': array([0.02954937, 0.03337564]),
  'rank_test_score': array([1, 2])},
 {'criterion': 'entropy'},
 0.9072903875463474)
```

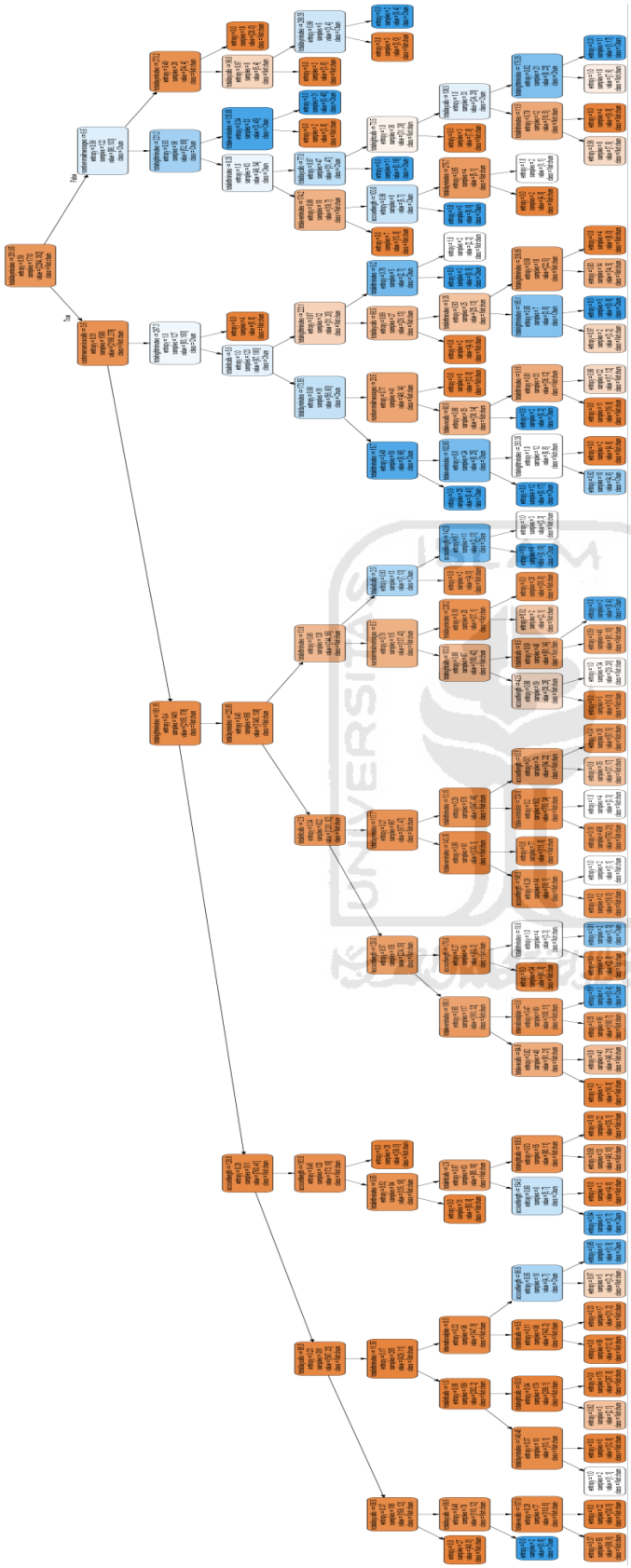
In [69]:

```
                              max_depth=8,max_features='sqrt', min_samples_leaf=2)
apply_classifier(RF_grid,X_train, X_test, y_train, y_test)
```

```
Algorithm: RandomForestClassifier

Classification report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       566
           1       0.95      0.60      0.74       101

    accuracy                           0.94       667
   macro avg       0.94      0.80      0.85       667
weighted avg       0.94      0.94      0.93       667

Accuracy Score: 0.9355322338830585
Area under ROC curve :  0.7993300213413568
```





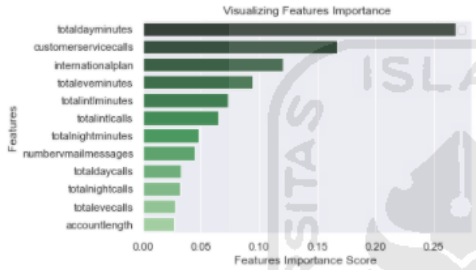


In [28]:
```
import pandas as pd
feature_imp = pd.Series(RF_grid.feature_importances_,index=cols).sort_values(ascending=False)
feature_imp
```

Out[28]:
```
totaldayminutes         0.268856
customerservicecalls    0.167072
internationalplan       0.120531
totaleveminutes         0.094122
totalintlminutes        0.073288
totalintlcalls          0.064444
totalnightminutes       0.048152
numbervmailmessages     0.044384
totaldaycalls           0.032762
totalnightcalls         0.031602
totalevecalls           0.027718
accountlength           0.027068
dtype: float64
```

In [42]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index, palette="Greens_d")
# Add labels to your graph
plt.xlabel('Features Importance Score ')
plt.ylabel('Features')
plt.title("Visualizing Features Importance")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



Lampiran 5 *Output* Klasifikasi Metode *Extreme Gradient Boosting*

In [81]:

```
Xgboost_parameters2 = { "max_depth" : [4,5,6,7,8]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters2,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid fitXGB.cv results , grid fitXGB.best params , grid fitXGB.best score
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  19 out of  25 | elapsed:    3.7s remaining:
1.1s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    4.0s finished
```

Out[81]:

```
({'mean_fit_time': array([1.39496012, 1.05116343, 0.45059052, 0.50885172, 0.
47855949]),
  'std_fit_time': array([0.01940891, 0.53236747, 0.00671633, 0.00933547, 0.0
5875118]),
  'mean_score_time': array([0.01795998, 0.01057029, 0.01117067, 0.00910516,
0.00678205]),
  'std_score_time': array([0.00260161, 0.00214781, 0.00230787, 0.00073015,
0.00074591]),
  'param_max_depth': masked_array(data=[4, 5, 6, 7, 8],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'max_depth': 4},
   {'max_depth': 5},
   {'max_depth': 6},
   {'max_depth': 7},
   {'max_depth': 8}],
  'split0_test_score': array([0.93034755, 0.92105488, 0.92622695, 0.9319957
9, 0.93151269]),
  'split1_test_score': array([0.90334562, 0.89971784, 0.89634919, 0.9012438
1, 0.9005528 ]),
  'split2_test_score': array([0.90317287, 0.91981458, 0.92269377, 0.9165611
, 0.9169066 ]),
  'split3_test_score': array([0.85344927, 0.87011977, 0.88071519, 0.8681907
2, 0.86496603]),
  'split4_test_score': array([0.91262247, 0.91074277, 0.91732171, 0.9061859
2, 0.90655616]),
  'mean_test_score': array([0.90058755, 0.90428997, 0.90866136, 0.90483547,
0.90409886]),
  'std_test_score': array([0.02556093, 0.01872015, 0.01739562, 0.02113166,
0.02220607]),
  'rank_test_score': array([5, 3, 1, 2, 4])},
 {'max_depth': 6},
 0.9086613614800877)
```

In [82]:

```
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters3,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    2.4s finished
```

Out[82]:

```
({'mean_fit_time': array([0.57765336, 0.4936903 , 0.46735411, 0.46056209,
0.44146914,
        0.43228822, 0.42485409, 0.40574307]),
  'std_fit_time': array([0.0296596 , 0.01797896, 0.01685909, 0.0168261 ,
0.00711651,
        0.00528586, 0.00723899, 0.00389988]),
  'mean_score_time': array([0.01635699, 0.00936308, 0.0157608 , 0.0099746
7, 0.01161222,
        0.01198606, 0.01139197, 0.00727782]),
  'std_score_time': array([0.00430543, 0.00100145, 0.00385393, 0.0035143
9, 0.00340553,
        0.00245545, 0.00354514, 0.00123007]),
  'param_min_child_weight': masked_array(data=[0, 1, 2, 3, 4, 5, 6, 7],
               mask=[False, False, False, False, False, False, False, Fal
se],
         fill_value='?',
              dtype=object),
  'params': [{'min_child_weight': 0},
   {'min_child_weight': 1},
   {'min_child_weight': 2},
   {'min_child_weight': 3},
   {'min_child_weight': 4},
   {'min_child_weight': 5},
   {'min_child_weight': 6},
   {'min_child_weight': 7}],
  'split0_test_score': array([0.93461025, 0.92622695, 0.93165478, 0.91608
173, 0.92310097,
        0.91758788, 0.92679531, 0.92577226]),
  'split1_test_score': array([0.89989059, 0.89634919, 0.89154094, 0.90138
777, 0.90504434,
        0.90510192, 0.89871012, 0.89056202]),
  'split2_test_score': array([0.92252102, 0.92269377, 0.92574571, 0.90950
708, 0.91235748,
        0.91552459, 0.91322124, 0.90947829]),
  'split3_test_score': array([0.87694345, 0.88071519, 0.87613728, 0.86099
274, 0.87052286,
        0.86306576, 0.8707244 , 0.86692387]),
  'split4_test_score': array([0.90874915, 0.91732171, 0.90464798, 0.89863
864, 0.90772386,
        0.90493279, 0.90362269, 0.894509  ]),
  'mean_test_score': array([0.90854289, 0.90866136, 0.90594534, 0.8973215
9, 0.9037499 ,
        0.90124259, 0.90261475, 0.89744909]),
  'std_test_score': array([0.01973993, 0.01739562, 0.02073991, 0.0191754
1, 0.01772037,
        0.01978447, 0.01860351, 0.01967114]),
  'rank_test_score': array([2, 1, 3, 8, 4, 6, 5, 7])},

 {'min_child_weight': 1},
 0.9086613614800877)
```

In [83]:

```
Xgboost_parameters4 = {"learning_rate" : [0.025, 0.05, 0.1, 0.2, 0.3]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters4,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Done  19 out of  25 | elapsed:    1.5s remaining:
0.4s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    1.8s finished
```

Out[83]:

```
({'mean_fit_time': array([0.54317565, 0.54717188, 0.49870481, 0.49379253, 0.
43435483]),
  'std_fit_time': array([0.0097981 , 0.02200379, 0.00890666, 0.00755654, 0.0
8179701]),
  'mean_score_time': array([0.01393285, 0.02092113, 0.01686482, 0.01199384,
0.00867138]),
  'std_score_time': array([0.00732763, 0.00464818, 0.00655151, 0.00227623,
0.00239233]),
  'param_learning_rate': masked_array(data=[0.025, 0.05, 0.1, 0.2, 0.3],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'learning_rate': 0.025},
   {'learning_rate': 0.05},
   {'learning_rate': 0.1},
   {'learning_rate': 0.2},
   {'learning_rate': 0.3}],
  'split0_test_score': array([0.93165478, 0.93219472, 0.93037597, 0.9315979
4, 0.92622695]),
  'split1_test_score': array([0.91883566, 0.92427732, 0.91178164, 0.8972993
2, 0.89634919]),
  'split2_test_score': array([0.91612922, 0.92252102, 0.91592767, 0.9209374
6, 0.92269377]),
  'split3_test_score': array([0.85359323, 0.86672233, 0.86228838, 0.8649372
3, 0.88071519]),
  'split4_test_score': array([0.93328492, 0.91578378, 0.91492937, 0.9145021
6, 0.91732171]),
  'mean_test_score': array([0.91069956, 0.91229983, 0.90706061, 0.90585483,
0.90866136]),
  'std_test_score': array([0.02934631, 0.02338027, 0.02328514, 0.02329127,
0.01739562]),
  'rank_test_score': array([2, 1, 4, 5, 3])},
 {'learning_rate': 0.05},
 0.9122998340290751)
```

In [84]:

```
Xgboost_parameters5 = {"gamma": [0, 0.1, 0.2, 0.3, 0.4, 1.0, 1.5, 2.0]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters5,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 8 candidates, totalling 40 fits

[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    2.6s finished
```

Out[84]:

```
({'mean_fit_time': array([0.52569766, 0.52817602, 0.49564123, 0.50915942, 0.
50432115,
        0.50252771, 0.49190617, 0.48512726]),
  'std_fit_time': array([0.00948119, 0.0295358 , 0.00961672, 0.01059025, 0.0
0712498,
        0.00961269, 0.00695687, 0.01987466]),
  'mean_score_time': array([0.01339135, 0.01540785, 0.01671476, 0.01263123,
0.01419363,
        0.0116498 , 0.00900936, 0.00611973]),
  'std_score_time': array([0.00311755, 0.00364386, 0.00476531, 0.00302764,
0.00338734,
        0.0035159 , 0.00411905, 0.001753  ]),
  'param_gamma': masked_array(data=[0, 0.1, 0.2, 0.3, 0.4, 1.0, 1.5, 2.0],
               mask=[False, False, False, False, False, False, False, Fals
e],
         fill_value='?',
              dtype=object),
  'params': [{'gamma': 0},
   {'gamma': 0.1},
   {'gamma': 0.2},
   {'gamma': 0.3},
   {'gamma': 0.4},
   {'gamma': 1.0},
   {'gamma': 1.5},
   {'gamma': 2.0}],
  'split0_test_score': array([0.92622695, 0.92634062, 0.93259257, 0.9266248
, 0.92190741,
        0.93370087, 0.93370087, 0.92778993]),
  'split1_test_score': array([0.89634919, 0.90265461, 0.90311528, 0.8981054
9, 0.91132097,
        0.90605206, 0.90809628, 0.90642635]),
  'split2_test_score': array([0.92269377, 0.91745365, 0.9183174 , 0.9208510
9, 0.90409421,
        0.9140562 , 0.91984337, 0.90896004]),
  'split3_test_score': array([0.88071519, 0.87461131, 0.87314292, 0.8645053
6, 0.86387193,
        0.86361281, 0.87584936, 0.86951514]),
  'split4_test_score': array([0.91732171, 0.91008772, 0.90672704, 0.9043062
2, 0.91097061,
        0.92469811, 0.90615744, 0.91367624]),
  'mean_test_score': array([0.90866136, 0.90622958, 0.90677904, 0.90287859,
0.90243303,
        0.90842401, 0.90872946, 0.90527354]),
  'std_test_score': array([0.01739562, 0.01765114, 0.01971946, 0.02183675,
0.020104  ,
        0.02428645, 0.01915833, 0.01934516]),

 'rank_test_score': array([2, 5, 4, 7, 8, 3, 1, 6])},
{'gamma': 1.5},
0.9087294648995673)
```

In [85]:

```
Xgboost_parameters6 = {"reg_alpha":[1e-5, 1e-2, 0.1, 1, 5e-3,5e-3]}
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters6,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 6 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Done  26 out of  30 | elapsed:    2.0s remaining:
0.2s
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    2.0s finished
```

Out[85]:

```
({'mean_fit_time': array([0.54807911, 0.53999968, 0.52399902, 0.47869678,
0.50741811,
        0.40702982]),
  'std_fit_time': array([0.02618285, 0.03073224, 0.01266716, 0.02615445,
0.01883864,
        0.05311571]),
  'mean_score_time': array([0.01386886, 0.01381893, 0.01792397, 0.0112735
7, 0.01260581,
        0.00740356]),
  'std_score_time': array([0.00286147, 0.00398888, 0.00306704, 0.0022100
6, 0.00639527,
        0.00079636]),
  'param_reg_alpha': masked_array(data=[1e-05, 0.01, 0.1, 1, 0.005, 0.00
5],
               mask=[False, False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'reg_alpha': 1e-05},
   {'reg_alpha': 0.01},
   {'reg_alpha': 0.1},
   {'reg_alpha': 1},
   {'reg_alpha': 0.005},
   {'reg_alpha': 0.005}],
  'split0_test_score': array([0.92622695, 0.92651113, 0.923783  , 0.92747
734, 0.92659638,
        0.92659638]),
  'split1_test_score': array([0.89634919, 0.90432454, 0.90311528, 0.89709
778, 0.90564897,
        0.90564897]),
  'split2_test_score': array([0.92269377, 0.92266498, 0.92090867, 0.92090
867, 0.91837499,
        0.91837499]),
  'split3_test_score': array([0.88180928, 0.87795117, 0.87671312, 0.87101
232, 0.86447656,
        0.86447656]),
  'split4_test_score': array([0.91732171, 0.90746753, 0.9196571 , 0.90966
051, 0.92011278,
        0.92011278]),
  'mean_test_score': array([0.90888018, 0.90778387, 0.90883543, 0.9052313
2, 0.90704194,
        0.90704194]),
  'std_test_score': array([0.01704608, 0.01717099, 0.01761173, 0.0199868
8, 0.02234015,
        0.02234015]),
  'rank_test_score': array([1, 3, 2, 6, 4, 4])},

 {'reg_alpha': 1e-05},
 0.9088801798608317)
```

In [86]:

```
grid_obj = GridSearchCV(xg_boost,xgboost_parameters7,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Done  12 out of  20 | elapsed:    0.3s remaining:
0.2s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    0.8s finished
```

Out[86]:

```
({'mean_fit_time': array([0.04012575, 0.02691069, 0.29899426, 0.48987031]),
  'std_fit_time': array([0.00396159, 0.01170451, 0.035662  , 0.07320427]),
  'mean_score_time': array([0.        , 0.        , 0.01196704, 0.0107694
6]),
  'std_score_time': array([0.        , 0.        , 0.0010921 , 0.00396098]),
  'param_colsample_bylevel': masked_array(data=['log2', 'sqrt', 0.25, 1.0],
               mask=[False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'colsample_bylevel': 'log2'},
   {'colsample_bylevel': 'sqrt'},
   {'colsample_bylevel': 0.25},
   {'colsample_bylevel': 1.0}],
  'split0_test_score': array([       nan,        nan, 0.93037597, 0.9262269
5]),
  'split1_test_score': array([       nan,        nan, 0.90204998, 0.8963491
9]),
  'split2_test_score': array([       nan,        nan, 0.90708856, 0.9226937
7]),
  'split3_test_score': array([       nan,        nan, 0.86427502, 0.8807151
9]),
  'split4_test_score': array([       nan,        nan, 0.90911939, 0.9173217
1]),
  'mean_test_score': array([       nan,        nan, 0.90258179, 0.9086613
6]),
  'std_test_score': array([       nan,        nan, 0.0214628 , 0.01739562]),
  'rank_test_score': array([3, 4, 2, 1])},
 {'colsample_bylevel': 1.0},
 0.9086613614800877)
```

In [87]:

```
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters8,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Done  12 out of  20 | elapsed:    0.8s remaining:
0.5s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    1.3s finished
```

Out[87]:

```
({'mean_fit_time': array([0.33171158, 0.51354833, 0.52275524, 0.44060044]),
  'std_fit_time': array([0.03255995, 0.01453974, 0.00940107, 0.04353682]),
  'mean_score_time': array([0.0105721 , 0.01158953, 0.01021767, 0.0080152
]),
  'std_score_time': array([0.00119675, 0.00160808, 0.00188762, 0.00143797]),
  'param_subsample': masked_array(data=[0.15, 0.5, 0.75, 1.0],
               mask=[False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'subsample': 0.15},
   {'subsample': 0.5},
   {'subsample': 0.75},
   {'subsample': 1.0}],
  'split0_test_score': array([0.87118134, 0.92278837, 0.92364091, 0.9262269
5]),
  'split1_test_score': array([0.87207762, 0.88966947, 0.89565818, 0.8963491
9]),
  'split2_test_score': array([0.88811471, 0.91181043, 0.91719452, 0.9226937
7]),
  'split3_test_score': array([0.835224  , 0.87101232, 0.84855465, 0.8807151
9]),
  'split4_test_score': array([0.86933242, 0.90567327, 0.90971748, 0.9173217
1]),
  'mean_test_score': array([0.86718602, 0.90019077, 0.89895315, 0.9086613
6]),
  'std_test_score': array([0.01734404, 0.01809865, 0.02686622, 0.01739562]),
  'rank_test_score': array([4, 2, 3, 1])},
 {'subsample': 1.0},
 0.9086613614800877)
```

In [88]:

```
grid_obj = GridSearchCV(xg_boost,Xgboost_parameters8,n_jobs=-1,cv=5,verbose=3,scoring='roc_
grid_fitXGB = grid_obj.fit(X_train,y_train)
grid_fitXGB.cv_results_, grid_fitXGB.best_params_, grid_fitXGB.best_score_
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

Fitting 5 folds for each of 3 candidates, totalling 15 fits

[Parallel(n_jobs=-1)]: Done   6 out of  15 | elapsed:    1.0s remaining:
1.5s
[Parallel(n_jobs=-1)]: Done  12 out of  15 | elapsed:    1.8s remaining:
0.4s
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:    2.0s finished
```

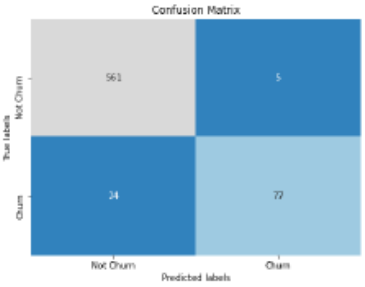
Out[88]:

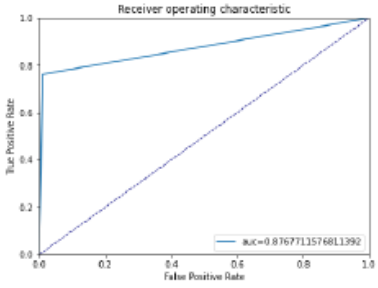
```
({'mean_fit_time': array([0.58836722, 0.98914876, 1.1021276 ]),
  'std_fit_time': array([0.03549485, 0.05125616, 0.11824156]),
  'mean_score_time': array([0.01179109, 0.01391125, 0.01141009]),
  'std_score_time': array([0.00147282, 0.00067109, 0.00119809]),
  'param_n_estimators': masked_array(data=[100, 200, 300],
               mask=[False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'n_estimators': 100},
   {'n_estimators': 200},
   {'n_estimators': 300}],
  'split0_test_score': array([0.92622695, 0.92892665, 0.92889824]),
  'split1_test_score': array([0.89634919, 0.90026489, 0.89876771]),
  'split2_test_score': array([0.92269377, 0.9232984 , 0.92519866]),
  'split3_test_score': array([0.88071519, 0.87613728, 0.87590694]),
  'split4_test_score': array([0.91732171, 0.90869219, 0.90772386]),
  'mean_test_score': array([0.90866136, 0.90746388, 0.90729908]),
  'std_test_score': array([0.01739562, 0.01868728, 0.01921119]),
  'rank_test_score': array([1, 2, 3])},
 {'n_estimators': 100},
 0.9086613614800877)
```

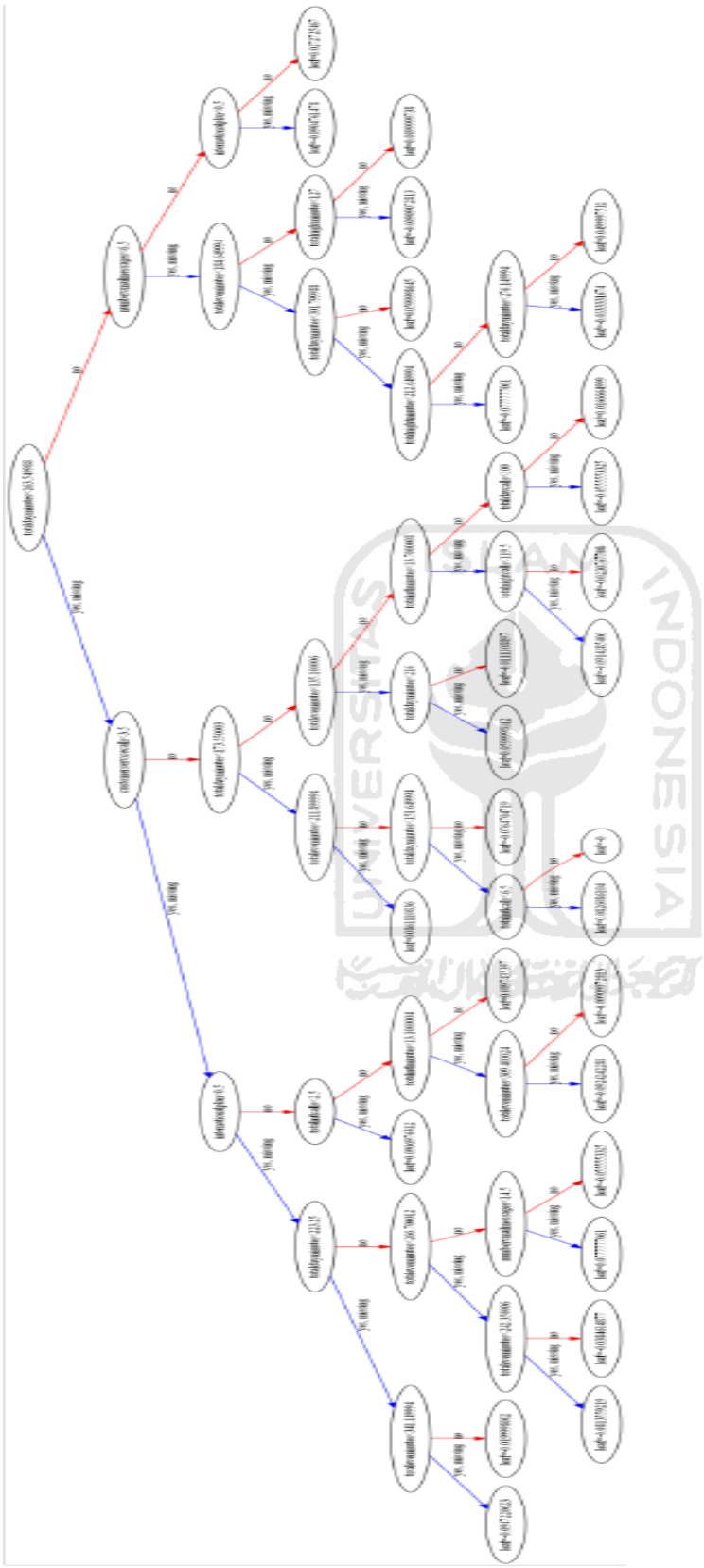
In [89]:

```
XGBGrid = XGBClassifier(n_estimators=100,colsample_bylevel= 1.0,
                        gamma= 1.5, learning_rate= 0.05, max_depth= 6, min_child_weight=1,
                        reg_alpha = 1e-05,
                        subsample=1.0)
apply_classifier(XGBGrid,X_train, X_test, y_train, y_test)
```

```
Algorithm: XGBClassifier

Classification report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       566
           1       0.94      0.76      0.84       101

    accuracy                           0.96       667
   macro avg       0.95      0.88      0.91       667
weighted avg       0.96      0.96      0.95       667

Accuracy Score: 0.9565217391304348
Area under ROC curve :  0.8767711576811392
```







In [37]:
```
feature_XGB = pd.Series(XGBGrid.feature_importances_,index=cols).sort_values(ascending=False)
feature_XGB
```

Out[37]:
```
customerservicecalls    0.243112
internationalplan       0.122557
totalintlcalls          0.117296
totaldayminutes         0.113212
totalintlminutes        0.095841
numbervmailmessages     0.089657
totaleveminutes         0.065699
totalnightminutes       0.040940
totaldaycalls           0.029837
totalnightcalls         0.029418
totalevecalls           0.027991
accountlength           0.024440
dtype: float32
```

In [40]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot

sns.barplot(x=feature_XGB, y=feature_XGB.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```