Implementasi SMOTE untuk Klasifikasi Loyalitas Konsumen Telesales X

(Studi Kasus : Data Transaksi Telesales Perusahaan X)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Program Studi Statistika

Disusun Oleh:

Titania Tasya Wananda

19611014

PROGRAM STUDI STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2023

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul	:	Implementasi SMOTE untuk Klasifikasi Loyalitas Konsumen Telesales X (Studi Kasus : Data Transaksi Telesales Perusahaan X)
Nama Mahasiswa	:	Titania Tasya Wananda
NIM	:	19611014

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, Februrari

Pembimbing

(Ayundyah Kesumawati, S.Si., M.Si.)

HALAMAN PENGESAHAN

TUGAS AKHIR

Implementasi SMOTE untuk Klasifikasi Loyalitas Konsumen Telesales X

(Studi Kasus : Data Transaksi Telesales Perusahaan X)

Nama Mahasiswa : Titania Tasya Wananda

NIM : 19611014

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL : 24 Februari 2023

Nama Penguji	**Tanda Tangan**
1. Achmad Fauzan, S.Pd., M.Si.	.....
2. Sekti Kartika Dini, S.Si., M.Si.	.....
3. Ayundyah Kesumawati, S.Si., M.Si.	.....

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

UNIVERSITAS ISLAM INDONESIA YOGYAKARTA FAKULTAS MIPA

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

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, 23 Februari 2023

(Titania Tasya Wananda)

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Assalamu'alaikum Wr.Wb

Puji dan syukur kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir dalam rangka pemenuhan hasil akhir perkuliahan sebagai persyaratan dalam menyelesaikan jenjang S1 Program Studi Statistika, Universitas Islam Indonesia. Tugas akhir ini berjudul **"Implementasi SMOTE untuk Klasifikasi Loyalitas Konsumen Telesales X"**.

Dalam penyusunan tugas akhir ini, penulis mendapatkan dukungan dari berbagai pihak sehingga dapat menyelesaikan dengan baik. Untuk itu, penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu dalam penyusunan tugas akhir ini kepada:

1. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta.
2. Bapak Dr. Edy Widodo, S.Si., M.Si. selaku Ketua Jurusan Statistika beserta seluruh jajarannya.
3. Ibu Ayundyah Kesumawati, S.Si., M.Si. selaku dosen pembimbing yang telah memberikan arahan dan saran selama mengerjakan tugas akhir ini.
4. Kedua orang tua penulis Ayub Setyobudi dan Dariyati, serta keluarga yang telah memberikan dukungan serta memberikan doa yang terbaik kepada penulis selama ini.
5. Sahabat penulis sejak kuliah, Nadhira Ferita Kusuma, Laila Sirri Hayati, dan Rizka Putri Maulidya yang telah menemani dan membantu sejak awal perkuliahan hingga saat ini.
6. Sahabat penulis Riska Bangkit Nastiti, Meicha Putri Anantiasari, Nadhilah Nur Azizah yang selalu mendukung dan memberikan semangat kepada penulis.
7. Sahabat penulis Sabilla, Astrid, Febby, Tika, Yessy, Andin, Ayuni, Ela, dan Adisa yang selalu menemani dan memberikan dukungan sejak awal perkuliahan.

8. Serta semua pihak yang tidak bisa dituliskan satu per satu yang telah berkontribusi dalam penyelesaian tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih terdapat kekurangan atau kesalahan karena keterbatasan pengetahuan dan kemampuan. Penulis mengharapkan kritik dan saran yang membangun untuk dapat menyempurnakan tugas akhir ini. Penulis berharap laporan ini dapat memberikan manfaat dan menambah wawasan bagi seluruh pembacanya. Akhir kata, semoga Allah SWT selalu melimpahkan rahmat serta karunia-Nya kepada kita semua, *Aamiin*.

Yogyakarta, Februari

Titania Tasya Wananda

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya 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, Februari 2023

Penulis

INTISARI

Implementasi SMOTE untuk Klasifikasi Loyalitas Konsumen Telesales X
(Studi Kasus : Data Transaksi Telesales Perusahaan X)

Titania Tasya Wananda
Program Studi Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Islam Indonesia

Perkembangan teknologi pada saat ini telah merambah dunia pendidikan yang biasa disebut dengan *edutech.* Saat ini terdapat beberapa perusahaan *startup* yang bergerak dalam bidang *edutech* dengan memberikan layanan pendidikan secara online seperti *live class, tryoust,* dan *chat bot.* Setiap perusahaan tentunya memiliki strategi dalam meningkatkan penjualan, salah satunya dengan membuat konsumen melakukan pembelian ulang produk. Penelitian ini bertujuan untuk memprediksi apakah konsumen akan melakukan pembelian ulang atau tidak. Metode klasifikasi yang digunakan dalam penelitian ini adalah *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM)* dan dilakukan optimasi menggunakan *Synthetic Minority Oversampling Technique* (SMOTE). Jumlah data konsumen yang melakukan pembelian ulang cukup sedikit sehingga diperlukan metode SMOTE untuk menyeimbangkan data dengan membuat data sintetis secara acak. Penelitian ini membagi data *training* dan data *testing* dengan perbandingan 80% : 20%. Hasil penelitian menunjukkan bahwa klasifikasi dengan metode KNN k = 4 memiliki tingkat akurasi terbaik yaitu sebesar 91%, sedangkan pada KNN k =4 dengan SMOTE sebesar 89%.

Kata Kunci : Edutech, Ketidakseimbangan data, Klasifikasi, SMOTE, Telesales

ABSTRACT

Implementation of SMOTE for Customer Loyalty Classification in Telesales X

(Case Study : Company X Telesales Transaction Data)

Titania Tasya Wananda
Department of Statistics, Faculty of Matematics and Natural Sciences
Universitas Islam Indonesia

Technological developments at this time have penetrated the world of education which is commonly referred to as edutech. Currently there are several startup companies engaged in edutech by providing online education services such as live classes, try outs, and chat bots. Every company certainly has a strategy to increase sales, one of which is by getting consumers to repurchase products. This study aims to predict whether consumers will make repeat purchases or not. The classification method used in this study is Logistic Regression, K-Nearest Neighbor (KNN), Naïve Bayes, Support Vector Machine (SVM) and optimization is carried out using the Synthetic Minority Oversampling Technique (SMOTE). The amount of consumer data that makes repurchases is quite small, so the SMOTE method is needed to balance the data by randomly generating synthetic data. This study divides training data and testing data with a ratio of 80%: 20%. The results showed that the classification using the KNN k = 4 method had the best accuracy rate, which was 91%, while the KNN k = 4 with SMOTE was 89%.

Keywords: *Edutech, Classification,Imbalance data, SMOTE, Telesales*

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Teknologi pada saat ini telah memasuki era 5.0 yang mempengaruhi beberapa aspek seperti ekonomi, industri, komunikasi, serta pendidikan. Bidang pendidikan sendiri tidak terlepas dari perkembangan teknologi sehingga terdapat teknologi pendidikan. *Association for Educational Communication and Technology* (AECT) mendefinisikan teknologi pendidikan sebagai fasilitas pembelajaran untuk meningkatkan kinerja melalui studi dan praktek dengan memanfaatkan teknologi secara tepat (Widyastuti, et al., 2020). Terdapat beberapa contoh pemanfaatan teknologi dalam bidang pendidikan seperti *Computer Based Training* (CBT), *Computer Based Education* (CBE), perpustakaan digital, serta *e-learning*. Pembelajaran elektronik atau *e-learning* yang memanfaatkan aplikasi, website, dan internet sehingga siswa dapat mengakses materi pelajaran secara online dimana saja (Adawi, 2008).

Pada masa pandemik Covid-19, pemerintah meniadakan kegiatan belajar di sekolah sehingga pihak sekolah harus menggunakan *e-learning* untuk kegiatan belajar mengajar di rumah dapat berjalan. Dengan begitu, siswa di Indonesia sudah mengimplementasikan penggunaan *e-learning* seperti melakukan pertemuan online dengan menggunakan platform *zoom meeting* atau *google meet*, memberikan materi pembelajaran melalui *WhatsApp* atau *Google Drive*, dan penggunaan *e-book* secara efektif. *E-learning* mulai muncul pada tahun 1990 dimana pada era CBT terdapat aplikasi *e-learning* dalam bentuk CD-ROM dan pada 1997 merupakan era *Learning Management System* (LMS) yang memanfaatkan teknologi internet dan dari tahun ke tahun semakin berkembang (Agustina, Santosa, & Ferdiana, 2016). Di Indonesia sendiri terdapat beberapa perusahaan yang bergerak dalam bidang *education technology* seperti Zenius, Ruang Guru, Hacktiv8, CoLearn, Quipper, dan lainnya.

Perusahaan X merupakan perusahaan *startup* pendidikan berbasis teknologi di Indonesia. Perusahaan X memberikan layanan akses video pendidikan secara *online*, *try out*, *live class*, panduan belajar, *boot camp* melalui aplikasi dan *website*.

Fitur yang ditawarkan oleh Perusahaan X saat ini yaitu materi *online*, *live class, try out,* dan *chatbot*.

Setiap perusahaan *edutech* tentunya memiliki inovasi produknya masing-masing dan harga yang ditawarkan bervariasi. Persaingan antar perusahaan dalam meraih konsumen sangat umum terjadi karena daya beli masyarakat yang rendah serta persaingan dengan kompetitor lainnya membuat perusahaan harus memiliki strategi marketing untuk unggul dalam persaingan (Kartajaya, 2006).

Pemasaran merupakan proses menciptakan, memasarkan, membagikan barang dan jasa kepada pelanggan untuk membangun dan mempertahankan hubungan yang positif (Tjiptono & Diana, 2016). Salah satu strategi pemasaran yang digunakan oleh Perusahaan X untuk memasarkan produk kepada pengguna dengan melalui telemarketing.

Telemarketing merupakan proses terciptanya suatu hubungan pertukaran secara jarak jauh antar manusia atau organisasi. Telemarketing adalah strategi promosi untuk mensosialisasikan produk melalui teknologi komunikasi dan personal terlatih kepada konsumen yang telah ditargetkan. Metode promosi dengan telemarketing sebaiknya dilakukan riset terlebih dahulu untuk mengetahui identitas konsumen yang akan dihubungi (Subroto, 2011).

Telemarketing memiliki beberapa metode untuk melakukan panggilan kepada konsumen yaitu panggilan manual, panggilan otomatis, dan pesan siaran. Terdapat dua jenis panggilan dalam telemarketing yaitu *inbound* dan *outbound* (Kotler & Keller, 2012). Orang yang melakukan pekerjaan telemarketing disebut dengan telesales. Telesales dalam melakukan panggilan outbound pastinya memerlukan data konsumen untuk mengetahui informasi konsumen yang akan ditelepon sehingga dapat merekomendasikan produk yang cocok untuk konsumen. Data yang digunakan untuk melakukan panggilan telemarketing didapatkan ketika konsumen mendaftar melalui aplikasi atau website untuk dapat mengakses fitur gratis yang disediakan oleh Perusahaan X. Data tersebut memuat informasi konsumen seperti kelas, nama, nomor hp, dan informasi lainnya.

Umumnya tidak semua konsumen yang telah dihubungi oleh telesales membeli produk yang telah ditawarkan. Analisis data diperlukan untuk mengetahui kriteria konsumen yang paling banyak melakukan pembelian sehingga telesales

dapat memiliki skala prioritas untuk melakukan panggilan kepada konsumen. Data konsumen yang ada perlu diproses dengan *data mining.*

Data mining merupakan proses untuk mencari pola atau informasi pada data sehingga dapat menemukan informasi baru dengan menggunakan metode tertentu yang tidak dapat dilakukan secara manual. Pemilihan metode yang sesuai berdasarkan tujuan dan akan mempengaruhi proses *Knowledge Discovery in Database (KDD)* secara keseluruhan. Data mining memiliki beberapa fungsi seperti deskripsi, estimasi, prediksi, klasifikasi, pengklasteran, dan asosiasi (Mardi, 2016). Penelitian ini menggunakan klasifikasi yang dapat memprediksi kelas dari data baru yang belum diketahui kelasnya. Metode klasifikasi yang digunakan adalah *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree.* Dataset pada algoritma klasifikasi dibagi menjadi data *training* dan data *testing, rasio* pembagian data tersebut ditentukan oleh peneliti. Dari metode klasifikasi didapatkan beberapa nilai untuk mengevaluasi model menggunakan *confusion matrix* seperti *precision, recall, f1-score,* dan *accuration.*

Algoritma klasifikasi tidak mempertimbangkan adanya ketidakseimbangan data pada dataset yang dapat mempengaruhi performa prediksi dan dapat meningkatkan bias pada kelas mayoritas. Terdapat dua metode dalam menangani data tidak seimbang yaitu menggunakan oversampling dan undersampling. Oversampling bekerja menyeimbangkan kelas minoritas dengan cara menduplikasi kelas minoritas yang sama persis sehingga terjadi overfitting. Undersampling bekerja menyeimbangkan kelas minoritas dengan menghapus kelas mayoritas sampai distribusinya seimbang, kelemahannya adalah dapat menghapus informasi yang berguna (Wijayanti, Kencana, & Sumarjana, 2021). Sehingga peneliti menggunakan metode oversampling dalam menyeimbangkan data. Terdapat beberapa metode oversampling yang umum digunakan seperti ROS (*Random Over Sampling*), SMOTE, dan ADASYN. Peneliti menggunakan metode SMOTE yang dapat meningkatkan jumlah kelas minoritas melalui replikasi data secara acak menggunakan *k* tetangga terdekat, sehingga metode SMOTE dapat mengurangi overfitting dan mempertahankan data asli (Suryana, Pratiwi, & Prasetio, 2021)

1.2. Rumusan Masalah

Berdasarkan latar belakang maka rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut:

1. Bagaimana gambaran umum data telesales perusahaan X?
2. Bagaimana mengatasi data telesales yang tidak seimbang dengan menggunakan SMOTE?
3. Apa metode klasifikasi terbaik untuk data telesales?
4. Bagaimana hasil klasifikasi dari metode terbaik untuk data telesales?

1.3. Batasan Masalah

Peneliti memberikan batasan terkait ruang lingkup masalah sebagai berikut:

1. Data yang digunakan dalam penelitian ini merupakan data telesales Perusahaan X pada bulan Juli 2022 – Oktober 2022.
2. Metode yang digunakan dalam penelitian ini adalah klasifikasi yang terdiri dari metode *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree* untuk mengklasifikasikan data telesales dan SMOTE untuk mengatasi data yang tidak seimbang.
3. Data diolah dengan menggunakan *Google Spreadsheet* untuk melakukan *cleaning* dan *recode* data dan *Google Colab* dengan menggunakan Bahasa pemrograman *Python* untuk pembuatan model.

1.4. Jenis Penelitian dan Metode Analisis

Penelitian ini menggunakan jenis penelitian aplikatif. Penelitian ini menggunakan metode SMOTE untuk mengatasi ketidakseimbangan data dan dilanjutkan dengan melakukan analisis dengan metode klasifikasi menggunakan *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree*. Penelitian ini digunakan untuk mengetahui karakteristik pelanggan yang melakukan layanan berbayar pada Perusahaan X.

1.5. Tujuan Penelitian

Berdasarkan rumusan masalah maka didapatkan tujuan penelitian sebagai berikut :

1. Mengetahui gambaran umum data telesales.

2. Mengatasi data telesales yang tidak seimbang dengan menggunakan SMOTE.
3. Mengetahui metode klasifikasi terbaik untuk data telesales.
4. Mengetahui klasifikasi data telesales menggunakan metode klasifikasi terbaik.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai beberapa penelitian terdahulu yang berhubungan dengan penelitian yang akan dilakukan.

2.1. Penelitian Terdahulu

Penelitian ini menggunakan beberapa acuan dari hasil penelitian atau observasi yang dilakukan pada penelitian sebelumnya terkait SMOTE dan Klasifikasi.

Tabel 2.1 Tabel Penelitian Sebelumnya

Tahun	Nama	Judul	Hasil Penelitian
2022	Laila Qadrini	*Handling Unbalanced Data With Smote Adaboost*	Penelitian ini menggunakan metode SMOTE untuk mengatasi data tidak seimbang dan dilanjutkan dengan klasifikasi Adaboost. Data yang digunakan adalah kualitas wine dengan proporsi kelas mayoritas dan minoritas sebesar 0,86 : 0,14. Rasio pembagian data *training* dan *testing* adalah 70% : 30 %. Hasil analisis yang didapatkan nilai AUC untuk model SMOTE + Adaboost yaitu 0,784 lebih baik dibandingkan dengan model Adaboost yaitu 0,664.
2021	Nana Suryana, Pratiwi, Rizki Tri Prasetio	Penanganan Ketidakseimbangan Data pada Prediksi Customer Churn Menggunakan Kombinasi SMOTE dan Boosting	Penelitian ini bertujuan untuk memprediksi pelanggan yang berpindah layanan atau berhenti berlangganan dengan menggunakan SMOTE dan Boosting yang diterapkan kedalam beberapa metode klasifikasi seperti Random Forest, Naïve Bayes, Decision Tree, K-Nearest Neighboor, dan Deep Learning. Dari hasil analisis dapat diketahui metode SMOTE meningkatkan rata-rata akurasi sebesar 3% dan metode AdaBoost mampu meningkatkan rata-rata akurasi sebesar 8%. Klasifikasi menggunakan metode Random

Tahun	Nama	Judul	Hasil Penelitian
			Forest memiliki nilai akurasi tertinggi sebesar 89,19% dibandingkan dengan metode lain.
2021	Wilda Imama Sabilla, Candra Bella Vista	Implementasi SMOTE dan *Under Sampling* Pada *Imbalanced Dataset* untuk Prediksi Kebangkrutan Perusahaan.	Penelitian ini bertujuan untuk memprediksi kebangkrutan dengan menggunakan model klasifikasi. Metode SMOTE dan *under sampling* digunakan untuk melakukan resampling pada data sehingga didapatkan model klasifikasi yang optimal. Metode klasifikasi yang digunakan adalah *multilayer perceptron* (MLP) dan *complement naïve bayes* (CNB). Dataset yang digunakan adalah perusahaan yang dikumpulkan dari Taiwan Economic Journal tahun 1999-2009 yang memiliki 96 atribut. Data pada kelas minoritas sebanyak 220 data dan kelas mayoritas memiliki 6599 data. Hasil analisis yang didapatkan yaitu metode klasifikasi CNB dan *resampling* memiliki nilai *recall* dan ROC AUC terbaik yaitu sebesar 95,45% dan 87,80%.
2020	Amalia Anjani Arifiyanti, Eka Dyar Wahyuni	SMOTE : Metode Penyeimbangan Kelas Pada Klasifikasi Data Mining	Penelitian ini bertujuan untuk menyeimbangkan data menggunakan SMOTE dan membandingkan hasil klasifikasi metode Logistic Regression, K-Nearest Neighbor (KNN), Decision Tree, dan Gaussian Naïve Bayes. Dataset yang digunakan adalah data penipuan kartu kredit dengan 19 atribut prediksi dan 1 atribut klasifikasi yang terdiri dari kelas 0 yaitu transaksi penipuan sebanyak 97346 data dan kelas 1 yaitu bukan transaksi peniupan sebanyak 2654 data. Dari hasil analisis didapatkan hasil bahwa metode klasifikasi dengan decision tree menghasilkan model dengan

Tahun	Nama	Judul	Hasil Penelitian
			performa baik pada data seimbang sebesar 96,2% dan tidak seimbang sebesar 96,8%.
2020	Muhamad Syukron, Rukun Santoso, Tatik Widiharih	Perbandingan Metode SMOTE *Random Forest* dan SMOTE XGBoost untuk Klasifikasi Tingkat Penyakit Hepatitis C pada *Imbalance Class* Data	Penelitian ini bertujuan untuk melakukan klasifikasi pada pasien hepatitis C apakah mengalami gejala hepatitis atau tidak berdasarkan hasil rekam medis. Metode klasifikasi yang digunakan adalah SMOTE + *random forest* dan SMOTE + *XGBoost* karena terdapat imbalance data antara kelas 0 dan 1. Data yang digunakan sebanyak 1385 yang dilakukan pembagian data *training* dan data *testing* dengan rasio 70%:30%, 75%:25%, dan 80%:20%. Dari penelitian ini didapatkan bahwa SMOTE *Random Forest* memiliki tingkat akurasi yang lebih tinggi, tetapi SMOTE XGBoost memiliki nilai *recall* yang lebih tinggi. Variabel yang paling berpengaruh adalah jumlah sel darah, jumlah enzim ALT, dan jumlah HCV RNA pada pasien.
2019	Anis Nimatul Kasanah, Muladi, Utomo Pujianto	Penerapan Teknik SMOTE untuk Mengatasi *Imbalance Class* dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN	Penelitian ini bertujuan untuk mengembangkan teknik dalam pemilahan objektivitas berita secara otomatis berdasarkan isi berita dengan menggunakan algoritma KNN k = 1, 3, 5, 7, dan 9. Penelitian ini membandingkan performa KNN dan KNN + SMOTE. Data yang digunakan sebanyak 200 berita yang didapatkan dengan teknik *scraping* dan diberi 2 label yaitu objektif sebanyak 176 berita dan subjektif sebanyak 24 berita. Dari hasil penelitian dapat diketahui performa KNN dengan menggunakan SMOTE mengalami peningkatan nilai akurasi pada k=1 dan k=3. Sedangkan pada k 5, 7, dan 9 mengalami penurunan nilai akurasi.

Jurnal penelitian pada Tabel 2.1 dijadikan referensi dalam penulisan penelitian ini. Penelitian ini meneliti tentang penerapan klasifikasi menggunakan beberapa metode yaitu *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree* dengan menggunakan SMOTE dan tanpa SMOTE. Hal yang membedakan penelitian ini dengan penelitian sebelumnya adalah objek yang diteliti. Penelitian ini menggunakan data transaksi telesales dari Perusahaan X. Hal yang sama antara penelitian ini dengan sebelumnya adalah klasifikasi dengan menggunakan SMOTE.

BAB III
LANDASAN TEORI

3.1. Telesales

Telemarketing merupakan kegiatan menggunakan telepon untuk menjual produk serta memberikan layanan kepada pelanggan secara langsung yang bertujuan untuk mendapatkan keuntungan. Telemarketing termasuk dalam pemasaran langsung (*direct marketing*) tanpa menggunakan pihak ketiga dalam bertransaksi. Telemarketing terbagi kedalam dua jenis yaitu *inbound* dan *outbound*. *Inbound* merupakan panggilan telepon dari konsumen untuk memesan dan bertanya terkait suatu produk atau umumnya disebut layanan konsumen. *Outbound* merupakan penggunaan telepon keluar kepada konsumen untuk menjual produk secara langsung (Kotler & Keller, 2012).

3.2. Statistika Deskriptif

Statistik secara garis besar dibagi menjadi statistik deskriptif dan statistik inferensia. Statistik deskriptif adalah analisis data dengan cara mendeskripsikan atau menggambarkan data tanpa membuat kesimpulan secara umum dengan menggunakan data sampel atau populasi (Sugiyono, 2015). Analisis deskriptif bertujuan untuk mengumpumpulkan, mengolah, dan menyajikan data sehingga dapat dipahami dengan mudah dengan visualisasi yang baik. Nilai yang termasuk dalam statistika deskriptif adalah rata-rata, modus, median, varians, simpangan baku, dan ukuran lainnya (Ghozali, 2016). Statistika deskriptif umumnya disajikan dengan ukuran pemusatan data dan divisualisasikan dalam bentuk tabel dan diagram (Hadi, 1993).

3.3. *Data Mining*

Data Mining merupakan proses dalam mendapatkan suatu informasi dari basis data yang besar dan perlu diolah sehingga menjadi informasi baru dan didapatkan informasi yang baru untuk membantu dalam pengambilan keputusan (Witten, Frank, Hall, & Pal, 2011). *Data mining* memiliki beberapa fungsi seperti berikut:

a. *Classification* : penemuan model untuk dapat menentukan kelas dari suatu data yang tidak diketahui kategorinya.
b. *Regression* : pemetaan data pada nilai prediksi
c. *Clustering* : pengelompokkan data yang memiliki karakteristik yang mirip.
d. *Association* : pemodelan untuk menemukan aturan asosiasi pada kombinasi item dalam waktu tertentu.
e. *Sequence* : pemodelan untuk menemukan aturan asosiasi pada kombinasi item dalam waktu tertentu dan diterapkan dalam beberapa periode.
f. *Forecasting* : peramalan suatu nilai berdasarkan pola yang terdapat pada suatu kelompok data.
g. *Solution* : proses penemuan dasar masalah dan penyelesaian dari permasalahan bisnis sebagai dasar pengambilan keputusan.

3.4. Klasifikasi

Klasifikasi merupakan salah satu analisis data untuk mendapatkan model untuk menggambarkan dan membedakan kelas data sehingga model dapat digunakan untuk memprediksi kelas dari suatu data yang belum diketahui kelasnya. Proses klasifikasi terdiri dari dua langkah yaitu *learning* (*fase training*) dan *testing. Learning* merupakan pembuatan algoritma klasifikasi untuk menganalisis data *training* dan dibentuk dalam model klasifikasi. *Fase testing* merupakan model klasifikasi yang telah terbentuk diuji pada data untuk mengetahui akurasi dari model (Putri & Setiadi, 2014). Terdapat empat komponen pada proses klasifikasi (Iriandi, 2013):

a. Kelas : variabel respon dengan tipe kategori yang mempresentasikan label pada objek.
b. *Prediktor* : variabel prediktor yang mempresentasikan karakteristik dari data.
c. *Training* dataset : *dataset* yang berisi kelas dan *prediktor* untuk menentukan kelas yang cocok berdasarkan *prediktor*.
d. *Testing* dataset : data baru yang akan diklasifikasikan menggunakan model yang telah dibuat dan menghasilkan tingkat akurasi model.

Tingkat performa algoritma data mining pada setiap kasus bergantung kepada kualitas dataset, jika data *training* berkualitas rendah menyebabkan tingkat akurasi pada klasifikasi lemah (Ardiyansyah, Rahayuningsih, & Maulana, 2018). Terdapat enam metode klasifikasi antara lain *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree.*

Evaluasi pada algoritma klasifikasi menggunakan *confusion matrix* yang terdiri dari *True Positive (TP), False Positive (FP), False Negative (FN),* dan *True Negative (TN). True Positive (TP)* merupakan data yang seharusnya berada di kelas positif dan diprediksi benar. *False Positive (FP)* merupakan data yang seharusnya berada di pada kelas positif tetapi diprediksi menjadi kelas negatif. *False Negative (FN)* merupakan data yang seharusnya berada di kelas negative diprediksi menjadi kelas positif. *True Negative (TN)* merupakan data yang berada pada kelas negative dan diprediksi secara benar (Prakasa & Lhaksamana, 2018).

Tabel 3.1 Confusion Matrix

Predictive|Actual	Negatif	Positif
Negatif	*True Negative (TN)*	*False Positive (FP)*
Positif	*False Negative (FN)*	*True Positive (TP)*

Terdapat beberapa parameter untuk mengukur kinerja algoritma dengan menggunakan *confusion matrix* seperti *precision, sensitivity* atau *recall, spesificity, f1-score, accuracy,* dan AUC. *Precision* merupakan parameter untuk mengetahui tingkat ketepatan dari algoritma yang dihitung dengan rasio jumlah label positif yang diprediksi benar dengan total data yang diprediksi memiliki label positif.

$$precision = \frac{TP}{TP + FP} \quad (3.1)$$

Accuracy merupakan parameter untuk mengetahui seberapa akurat kinerja dari sebuah algoritma untuk melakukan klasifikasi.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.2)$$

Sensitivity atau *Recall* merupakan parameter untuk mengetahui proporsi kasus positif atau klasifikasi benar pada kelas minoritas yang diidentifikasi dengan benar.

$$Sensitivity = \frac{TP}{TP + FN} \quad (3.3)$$

Spesificity merupakan parameter untuk mengetahui proporsi kasus negatif atau klasifikasi benar pada kelas mayoritas yang diidentifikasi dengan benar. Nilai sensitivitas dan spesivitas yang mendekati 100% menyatakan indikator baiknya suatu sistem klasifikasi dalam mengklasifikasikan data sesuai kelasnya.

$$Spesifity = \frac{TN}{TN + FP} \quad (3.4)$$

F1-Score atau F-Measure merupakan parameter untuk mengetahui perbandingan rata-rata precision dan recall dengan nilai terbesar adalah 1 dan nilai terkecil 0. Semakin tinggi nilai *f1-score* menunjukkan nilai *recall* dan *precision* yang baik. Jika nilai f1-score rendah menunjukkan jumlah false positif atau false negative yang tinggi, jika f1-score tinggi maka menunjukkan model memiliki jumlah false positif dan false negative cukup rendah.

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall} \quad (3.5)$$

AUC (*Area Under Curve*) merupakan parameter untuk mengukur kinerja menggunakan probabilitas dari sampel acak kelas positif dan negatif. Nilai AUC berkisar 0 sampai 1, semakin mendekati nilai 1 maka model klasifikasi akan semakin baik.

$$AUC = \left[\frac{1 + sensitifity - FPR}{2}\right] \quad (3.6)$$

$$FPR\ (False\ Positif\ Rate) = 1 - specificity \quad (3.7)$$

Contoh :

Data telesales suatu perusahaan diklasifikasikan menggunakan metode *Decision Tree* dengan pembobotan antara data *training* dan data *testing* sebesar 80% : 20%, sehingga didapatkan *confusion matrix* seperti berikut.

	Negatif	Positif
Negatif	1203 *(TN)*	28 *(FP)*
Positif	104 *(FN)*	11 *(TP)*

Confusion matrix diatas dapat digunakan untuk mengetahui kinerja algoritma dengan KNN k=2 menggunakan *precision, recall, f1-score,* dan *accuration.*

- *Precision*

$$precision = \frac{TP}{TP + FP} = \frac{11}{11 + 28} = 0.28$$

- *Accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{11 + 1203}{11 + 1203 + 28 + 104} = 0.90$$

- *Sensitivity*

$$Sensitivity = \frac{TP}{TP + FN} = \frac{11}{11 + 104} = 0.10$$

- *Spesificity*

$$Spesificity = \frac{TN}{TN + FP} = \frac{1203}{1203 + 28} = 0.98$$

- *F1-Score*

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall} = \frac{2 \times 0.28 \times 0.10}{0.28 + 0.10} = 0.14$$

- AUC

$$AUC = \left[\frac{1 + sensitifity - FPR}{2}\right] = \left[\frac{1 + 0.10 - 0.02}{2}\right] = 0.54$$

Hasil klasifikasi menggunakan *Decision Tree* didapatkan nilai *accuracy* sebesar 0.90 atau 90% algoritma dapat melakukan klasifikasi dengan akurat, nilai *precision* sebesar 0.28 atau 28% ketepatan algoritma dalam menghitung label positif, nilai *sensitifity* sebesar 0.10 atau 10% algoritma dapat memprediksi kasus positif yang diidentifikasi dengan benar, nilai *spesificity* sebesar 0.98 atau 98% algoritma dapat memprediksi kasus negatif yang diidentifikasi dengan benar, nilai *f1-score* sebesar 0.14 atau 14% yang cukup rendah sehingga kurang baik dalam melakukan prediksi kelas positif, dan nilai AUC sebesar 0.54 atau 54% yang menandakan model klasifikasi tidak cukup baik.

3.4.1 *Regresi Logistic*

Regresi Logistic adalah metode analisis data yang bertujuan untuk mengetahui hubungan antara beberapa variabel dimana variabel responnya bersifat kategorik. Regresi *logistic* biner merupakan pendekatan model matematika yang digunakan untuk menguji hubungan antara variabel prediktor yang terdiri dari beberapa faktor dengan variabel respon. Prosedur yang digunakan untuk mengestimasi nilai parameter menggunakan *Maximum Likelihood Estimation* (MLE) (Ramli, Yuniarti, & Goejantoro, 2013).

Variabel respon terdiri dari 2 kategori yaitu nilai 1 yang menyatakan keberadaan dan nilai 0 menyatakan ketidakberadaan suatu karakteristik, maka fungsi probabilitas untuk setiap observasi adalah (Damanik, Ispriyanti, & Sugito, 2015)

$$P = (Y = 1|X = x_i) = \pi(x_i) \text{ , untuk Y= 1} \quad \text{(3.8)}$$

dan

$$P = (Y = 0|X = x_i) = 1 - \pi(x_i) \text{ , untuk Y = 0}$$

X : Variabel prediktor

Y : Variabel respon

Diperoleh definisi model regresi logistik

$$\pi(x_i) = \left(\frac{e^{g(x)}}{1 + e^{g(x)}}\right) \quad \text{(3.9)}$$

$\pi(x_i)$: Peluang terjadinya respon pada variabel *i*

$g(x)$: Logit yang merupakan fungsi linier dari variabel prediktor

Logit dari $\pi(x_i)$ adalah

$$g(x) = \ln\left(\frac{\pi(x)}{1 - \pi(x)}\right) = \beta_0 + \beta_1 \mathrm{x}_1 + \beta_2 \mathrm{x}_2 + \cdots + \beta_\mathrm{p} \mathrm{x}_\mathrm{p} \quad \text{(3.10)}$$

β_0 : Konstanta dari model

$\beta_1, \beta_2, \ldots, \beta_p$: Parameter koefisien regresi

Menentukan estimasi parameter menggunakan fungsi likelihood sebagai berikut.

$$\ell(\beta) = \prod_{i=1}^{n} (\pi(x_i))^{y_i} (1 - \pi(x_i))^{1-y_i} \quad \text{(3.11)}$$

Fungsi likelihood akan lebih mudah dimaksimalkan dalam bentuk log $\ell(\beta)$ yang ditanyakan dengan $\mathrm{L}(\beta)$.

$$L(\beta) = \ln \ell(\beta) = \ln \prod_{i=1}^{n} (\pi(x_i))^{y_i} (1 - \pi(x_i))^{1-y_i} \quad \text{(3.12)}$$

$$= \sum_{i=1}^{n} y_i(\beta_0 + \beta_1 \mathrm{x}_\mathrm{i}) - \sum_{i=1}^{n} \ln(1 + \exp(\beta_0 + \beta_1 x_i)$$

Model MLE yang telah terbentuk kemudian diuji signifikansi dengan menggunakan Uji Rasio Likelihood dan Uji Wald. Uji Rasio Likelihood digunakan untuk mengetahui kesesuaian model dengan variabel prediktor secara keseluruhan. Statistik uji yang digunakan mengikuti distribusi *Chi Square*, dimana *Chi Square*

tabel sebagai perbandingan dalam pengambilan keputusan (Faramudhita, Ruswandi, & Saidi, 2017).

- Hipotesis

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_p = 0$$
$$H_1: \text{minimal ada satu } \beta_j \neq 0, j = 1,2,\ldots,p \quad (3.13)$$

- Statistik Uji

$$G = -2\ln\left[\frac{L_0}{L_1}\right] = -2\ln\left[\frac{\left(\frac{n_1}{n}\right)^{n_1}\left(\frac{n_0}{n}\right)^{n_0}}{\prod_{i=1}^{n}\pi_1{}^{y_i}(1-\pi_1)^{(1-y_i)}}\right] \quad (3.14)$$

n_1 = jumlah kategori 1

n_0 = jumlah kategori 0

n = jumlah $n_1 + n_0$

L_1 = Likelihood tanpa variabel bebas

L_0 = Likelihood dengan variabel bebas

- Kriteria Uji

Tolak H_0 jika $G > \chi^2_{(db,a)}$ atau p-value < a

Uji Wald dilakukan untuk mengetahui apakah variabel prediktor mempengaruhi variabel respon sehingga diketahui kelayakan suatu variabel prediktor dalam model. Statistik uji yang digunakan mengikuti distribusi normal baku sehingga tabel Z sebagai perbandingan dalam pengambilan keputusan.

- Hipotesis

$$H_0: \beta_i = 0$$
$$H_1: \beta_i \neq 0, \mathrm{i} = 1,2,\ldots,\mathrm{n} \quad (3.15)$$

- Statistik Uji

$$W = \left[\frac{\hat{\beta}_i}{SE(\hat{\beta}_i)}\right]^2 \quad (3.16)$$
$$SE(\hat{\beta}_i) = \sqrt{Var(\hat{\beta}_i)}$$

$\hat{\beta}_i$: Penaksir parameter β_i

$SE\hat{\beta}_i$: Standar Error $\hat{\beta}$

- Kriteria Uji

Tolak H_0 jika $Wr > Z_{\alpha/2}$ atau p-value < a

Odds ratio merupakan ukuran risiko atau kecenderungan untuk mengalami kejadian sukses (y=1) pada suatu kategori dengan kategori lainnya (y=0). Rasio dari odds untuk y=1 dan y=0 dituliskan pada persamaan berikut.

$$\varphi = \frac{\pi(1)/[1-\pi(1)]}{\pi(0)/[1-\pi(0)]} \quad (3.17)$$

Nilai odds ratio

$$\varphi = \frac{e^{\beta_0+\beta_1}}{e^{\beta_0}} = e^{\beta_1} \quad (3.18)$$

Setiap kenaikan satu satuan nilai X akan mengakibatkan perubahaan nilai *odds* (resiko) terjadinya Y=1 sebesar exp (β_1) kali (Risdayanti & Aidid, 2020).

3.4.2 *K-Nearest Neighbor (KNN)*

KNN merupakan metode yang digunakan untuk mengklasifikasikan objek berdasarkan data *training* yang memiliki jarak terdekat dari objek tersebut untuk menentukan kelas dari data tersebut. KNN termasuk kedalam algoritma *supervised learning* yang menentukan kelas berdasarkan jarak terpendek dari data *testing* dan data *training*. Kelas yang paling banyak muncul akan menjadi kelas hasil dari klasifikasi. Metode KNN memiliki beberapa kelebihan seperti tahan terhadap data yang memiliki *noise* atau pencilan dan efektif terhadap data *training* dalam jumlah besar serta memiliki performa yang baik (Kasanah, Muladi, & Pujianto, 2019).

Klasifikasi menggunakan metode KNN perlu menentukan nilai *k* terlebih dahulu. Nilai *k* adalah banyaknya tetangga terdekat yang akan digunakan untuk melakukan klasifikasi data baru (Isman, Ahmad, & Latief, 2021). Nilai *k* terbaik tergantung pada data yang digunakan. Secara umum, jika nilai *k* tinggi maka akan mengurangi efek *noise* pada klasifikasi dan membuat kelas antara setiap klasifikasi semakin kabur (Prakasa & Lhaksamana, 2018)

Setelah menentukan nilai *k,* maka dilanjutkan dengan mengukur jarak antar data menggunakan *hamming distance* untuk menghitung jarak antar titik data ketika data penelitianya kategorik. Berikut merupakan rumus yang digunakan untuk menghitung nilai *hamming distance* (Nayak, Bhat, Reddy, & Rao, 2022).

$$d = \sum_{i=1}^{n} (x_i - y_i) \qquad (3.19)$$

d = Jarak

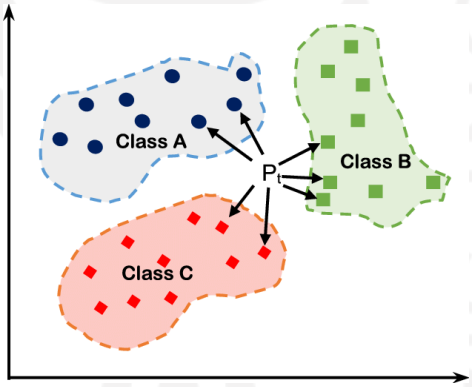
x_i = Data *training*

y_i = Data *testing*

i = Atribut

n = Jumlah atribut

Semakin besar nilai jarak maka akan semakin jauh tingkat kesamaan antara dua objek dan semakin kecil nilai jarak maka kedua objek akan semakin memiliki banyak kesamaan (Prakasa & Lhaksamana, 2018). Nilai *hamming distance* yang diperoleh kemudian diurutkan kedalam kelas yang memiliki jarak terkecil sampai urutan *k* dan memasangkan kelas yang bersesuaian. Selanjutnya mencari jumlah kelas dari tetangga yang terdekat dan menetapkan kelas sebagai kelas yang akan dievaluasi (D.N, Auliasari, & Pranoto, 2020).



Gambar 3.1 Ilustrasi KNN k=6

3.4.3 *Naïve Bayes*

Klasifikasi naïve bayes merupakan pengklasifikasian statistik untuk memprediksi probabilitas keanggotaan pada suatu kelas yang didasarkan pada teorema bayes. Metode naïve bayes memiliki tingkat akurasi yang tinggi ketika diaplikasikan kedalam *database* yang besar (Annur, 2018). Metode naïve bayes termasuk kedalam pembelajaran *supervised,* sehingga dibutuhkan data *training* pada tahap pembelajaran untuk dapat mengambil keputusan. Pada tahap klasifikasi akan dihitung nilai probabilitas pada setiap label kelas yang ada dan label kelas

yang memiliki nilai probabilitas terbesar akan dijadikan label kelas pada data awal. Teorema naïve bayes

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (3.20)$$

X = Data dengan kelas yang belum diketahui

C_i = Hipotesis data X merupakan suatu kelas spesifik

$P(C_i|X)$ = Probabilitas kelas C_i dengan data X

$P(X|C_i)$ = Probabilitas data X dengan label kelas C_i

$P(C_i)$ = Probabilitas kelas C_i

Teorema Bayes yang digunakan disesuaikan dengan teorema naïve bayes.dimana memerlukan sejumlah informasi untuk menentukan kelas yang cocok bagi sampel yang dianalisis.

$$P(C|F_1 \dots F_n) = \frac{P(C)P(F_1 \dots F_n|C)}{P(F_1 \dots F_n)} \quad (3.21)$$

Rumus (3.17) menjelaskan bahwa peluang masuknya sampel yang memiliki karakteristik tertentu dalam kelas *C* yang disebut dengan *posterior,* peluang munculnya kelas *C* sebelum masuknya sampel dengan karakteristik tertentu disebut dengan *prior,* peluang kemunculan karakteristik sampel pada kelas *C* disebut dengan *likelihood.*

$$Posterior = \frac{Prior\ x\ Likelihood}{Evidence} \quad (3.22)$$

Evidence merupakan peluang munculnya karakteristik sampel secara global yang memiliki nilai tetap untuk setiap kelas pada sampel yang sama. Nilai posterior yang diperoleh pada setiap kelas akan dibandingkan untuk menentukan suatu sampel akan diklasifikasikan pada kelas yang mana.

Jika rumus bayes (3.17) $P(C|F_1 \dots F_n)$ dijabarkan maka akan menyebabkan semakin banyak dan kompleksnya faktor-faktor yang mempengaruhi nilai probabilitas. Sehingga digunakan asumsi prediktorsi dimana masing-masing petunjuk $F_1, F_2, \dots, F_n$ saling bebas dan berlaku kesamaan berikut.

$$P(F_i|F_j) = \frac{P(F_i \cap F_j)}{P(F_i)} = \frac{P(F_i)P(F_j)}{P(j)} = P(F_i) \quad (3.23)$$

Untuk i ≠ j, maka

$$P(F_i|C, F_j) = P(F_i|C) \quad (3.24)$$

Persamaan (Aliady, Tuasikal, & Widodo, 2018) didapatkan bahwa asumsi prediktorsi dapat membuat syarat peluang menjadi sederhana, sehingga $P(C|F_1 \dots F_n)$ dapat disederhanakan menjadi (Saleh, 2015).

$$P(F_1 \dots F_n|C) = P(F_1|C)P(F_2|C), \dots, P(F_n|C) = \prod_{i=1}^{n} P(F_i|C) \quad (3.25)$$

Contoh

Seorang sales akan mengklasifikasikan siswa yang telah melakukan pembelian paket belajar di perusahaan X akan melakukan pembelian ulang atau tidak. Dimana telah didapatkan himpunan data dengan 4 atribut.

No	Tipe Produk	Status Cicilan	Harga Produk	Pembelian Ulang
1	Murah	Tidak	Murah	Tidak
2	Mahal	Ya	Sedang	Tidak
3	Murah	Tidak	Murah	Ya
4	Mahal	Ya	Mahal	Ya
5	Murah	Ya	Murah	Tidak
6	Mahal	Tidak	Mahal	Ya
7	Murah	Ya	Murah	Ya
8	Murah	Ya	Murah	Tidak
9	Murah	Tidak	Murah	Ya
10	Murah	Tidak	Murah	Tidak
11	Mahal	Tidak	Sedang	Tidak
12	Murah	Tidak	Murah	Ya
13	Murah	Tidak	Murah	Tidak
14	Murah	Ya	Murah	Ya
15	Mahal	Ya	Mahal	Ya
16	Mahal	Tidak	Sedang	Ya
17	Mahal	Ya	Mahal	Tidak

Probabilitas kemunculan setiap nilai untuk Tipe Produk

Tipe Produk	Jumlah		Probabilitas	
	Ya	Tidak	Ya	Tidak
Mahal	4	3	4/9	3/8
Murah	5	5	5/9	5/8
Jumlah	9	8	1	1

Probabilitas kemunculan setiap nilai untuk Status Cicilan

Status Cicilan	Jumlah		Probabilitas	
	Ya	Tidak	Ya	Tidak
Ya	4	4	4/9	4/8
Tidak	5	4	5/9	4/8
Jumlah	9	8	1	1

Probabilitas kemunculan setiap nilai untuk Harga Produk

Harga Produk	Jumlah		Probabilitas	
	Ya	Tidak	Ya	Tidak
Mahal	3	1	3/9	1/8
Sedang	1	2	1/9	2/8
Murah	5	5	5/9	5/8
Jumlah	9	8	1	1

Probabilitas kemunculan setiap nilai untuk Pembelian Ulang

Pembelian Ulang	Jumlah		Probabilitas	
	Ya	Tidak	Ya	Tidak
Jumlah	9	8	9/17	8/17

Berdasarkan dataset tersebut, seorang siswa melakukan pembelian dengan tipe produk mahal, tidak menggunakan cicilan, dan harga produk yang dibeli sedang.

- Likelihood Ya

$$P(\text{Ya}|\text{Pembelian Ulang}) = \frac{P(\text{Tipe produk mahal}|\text{Ya}) \times P(\text{Status cicilan tidak}|\text{Ya}) \times P(\text{Harga produk sedang}|\text{Ya}) \times P(\text{Ya})}{P(\text{Pembelian ulang})}$$

$$= \frac{4}{9} \times \frac{5}{9} \times \frac{1}{9} \times \frac{9}{17} = 0.015$$

- Likelihood Tidak

$$P(\text{Tidak}|\text{Pembelian Ulang}) = \frac{P(\text{Tipe produk mahal}|\text{Tidak}) \times P(\text{Status cicilan tidak}|\text{Tidak}) \times P(\text{Harga produk sedang}|\text{Tidak}) \times P(\text{Tidak})}{P(\text{Pembelian ulang})}$$

$$= \frac{3}{8} \times \frac{4}{8} \times \frac{2}{8} \times \frac{8}{17} = 0.022$$

Nilai probabilitas dapat dihitung dengan melakukan normalisasi terhadap nilai likelihood yang telah didapatkan, dengan jumlah peluang kejadian adalah 1.

- Probabilitas Ya

$$\text{P(Ya|Pembelian Ulang)} = \frac{\text{P(Ya|Pembelian ulang)}}{\text{P(Ya|Pembelian ulang)} + \text{P(Tidak|Pembelian ulang)}}$$

$$= \frac{0.015}{0.015 + 0.022} = 0.397$$

- Probabilitas Tidak

$$\text{P(Tidak|Pembelian Ulang)} = \frac{\text{P(Tidak|Pembelian Ulang)}}{\text{P(Ya|Pembelian Ulang)} + \text{P(Tidak|Pembelian Ulang)}}$$

$$= \frac{0.022}{0.015 + 0.022} = 0.603$$

$$\text{P(Tidak|Pembelian Ulang)} > \text{P(Ya|Pembelian Ulang)}$$

Berdasarkan perhitungan diatas, dapat disimpulkan bahwa siswa tersebut tidak akan melakukan pembelian ulang.

3.4.4 *Support Vector Machine (SVM)*

Support Vector Machine (SVM) merupakan metode untuk melakukan prediksi yang dapat digunakan untuk klasifikasi dan regresi. SVM mendefinisikan batas antara dua kelas menggunakan jarak maksimal dari data yang terdekat. Batas maksimal antar kelas didapatkan dengan membentuk *hyperplane* (garis pemisah) terbaik yang diperoleh dari mengukur *margin hyperplane* yang merupakan jarak antara *hyperplane* dari titik terdekat setiap kelas dan mencari titik maksimalnya (Rizal, Girsang, & Prasetyo, 2019). Klasifikasi dengan metode SVM awalnya dikembangkan hanya untuk dua kelas, tetapi dikembangkan kembali untuk klasifikasi pada multikelas. Klasifikasi multikelas menggunakan metode *Structure Learning Approach* (SLA) yang membentuk *k hyperplane* dimana *k* adalah banyak kelas (Octaviani, Wilandari, & Ispriyanti, 2014).

Hyperplane untuk klasifikasi linier SVM dinotasikan sebagai berikut.

$$f(x) = Wx + b \qquad (3.26)$$

Hyperplane membagi dataset menjadi 2 kelas, dengan kelas pertama bernilai +1 dan kelas lainnya bernilai -1.

$$X_i . W + b \geq 1 \text{ untuk } Y_i = +1 \qquad (3.27)$$

$$X_i . W + b \leq -1 \text{ untuk } Y_i = -1 \qquad (3.28)$$

X_i = data ke-i

W = nilai bobot support vector

b = nilai bias

Y_i = kelas data ke-i

Bobot *vector* (*w*) merupakan garis vektor yang tegak lurus antara titik pusat koordinat dengan garis *hyperplane.* Sedangkan bias (*b*) merupakan garis relatif terhadap titik koordinat (Parapat, Furqon, & Sutrisno, 2018).

$$b = -\frac{1}{2}(w.x^{+} + w.x^{-}) \quad (3.29)$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (3.30)$$

b = nilai bias

$w.x^{+}$ = bobot nilai data kelas positif

$w.x^{+}$ = bobot nilai data kelas positif

w = bobot vector

α_i = bobot nilai data ke-*i*

y_i = kelas data ke-*i*

x_i = data ke-*i*

Hyperlane terbaik terletak diantara dua bidang pembatas kelas atau dapat dengan memaksimalkan *margin* antara dua set objek dari data kelas yang berbeda (Octaviani, Wilandari, & Ispriyanti, 2014). Margin dapat dihitung dengan $\frac{2}{||w||}$. Metode *Quadratic Programming (QP)* digunakan untuk mencari *hyperplane* terbaik dengan persamaan berikut.

$$\min \frac{1}{2}||w||^2 \quad (3.31)$$

Dengan syarat

$$y_i(wx_i + b) \geq 1, i = 1, 2, \ldots, n \quad (3.32)$$

Persamaan (3.27) berfungsi untuk meminimalkan $\frac{1}{2}||w||^2$ dengan memperhatikan pembatas. Persamaan (3.28) mewakili batasan bahwa titik data ke-*i* harus berada di sisi yang benar pada *hyperplane.* Jika data $y_i = +1$ maka pembatas menjadi $(wx_i + b) \geq 1$, sedangkan jika data $y_i = -1$ maka pembatas menjadi $(wx_i + b) \leq -1$.

Metode SVM mengasumsikan kedua kelas terpisah sempurna oleh *hyperlane,* namun pada kenyataannya kedua kelas tidak selalu terpisah dengan sempurna. Untuk mengatasi masalah ini SVM dapat dirumuskan ulang menggunakan teknik *soft margin* dengan menambahkan parameter *C* dan variabel slack (Nugroho, 2003).

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\varepsilon_i \qquad (3.33)$$

Pada persamaan (3.30) ε_i merupakan variabel slack yang digunakan untuk meminimalkan kesalahan klasifikasi dari batasan $y_i(wx_i + b) \geq 1$ dengan memberikan penalti untuk data yang tidak memenuhi batasan tersebut. Pinalti ini diterapkan dengan menambahkan parameter C yang berperan dalam menyeimbangkan antara *margin* dan kesalahan klasifikasi. Semakin besar nilai C maka pinalti yang dikenakan untuk setiap kesalahan klasifikasi semakin besar (Nugroho, 2003).

SVM merupakan *linear classifier* yaitu kasus klasifikasi yang hanya dapat dipisahkan secara linier, tetapi SVM telah berkembang pada kasus non linier dengan menggunakan konsep *kernel* pada ruang kerja berdimensi tinggi sehingga dapat memaksimalkan *margin* antar kelas (Octaviani, Wilandari, & Ispriyanti, 2014).

Tabel 3.2 Fungsi Kernel (Nugroho, 2003)

Kernel	**Definisi**
Linear K	$K(x_i, x_j) = x_i, x_j$
Polynomial	$K(x_i, x_j) = (x_i, x_j + 1)^2$
Gaussian RBF	$K(x_i, x_j) = \exp\left(-\frac{\left\|\left\|x_i, x_j\right\|\right\|^2}{2\sigma^2}\right)$
Sigmoid	$K(x_i, x_j) = \tanh(\alpha x_i, x_j + \beta)$

3.4.5 *Decision Tree*

Decision Tree atau pohon keputusan merupakan algoritma yang digunakan untuk pengambilan keputusan pada klasifikasi. *Decision Tree* dapat digunakan untuk mengeksplorasi data, serta mengetahui hubungan antara calon variabel input

dengan variabel target (Achmad & Slamat, 2012). Decision Tree menggunakan contoh pohon untuk metode klasifikasinya seperti *root node* yang berada di bagian paling atas, *internal node* merupakan percabangan yang membutuhkan satu *input* dan dapat mengeluarkan hingga dua *output* yang menggambarkan nilai dari setiap kelas, dan *leaf node* terletak pada ujung pohon dimana memiliki satu input dan tidak terdapat output yang menggambarkan setiap kelas (Harryanto & Hansun, Penerapan Algoritma C4.5 untuk Memprediksi Penerimaan Calon Pegawai Baru di PT WISE, 2017).

Algoritma dalam menentukan pohon keputusan menggunakan *decision tree C4.5* terlebih dahulu menentukan data *training* yang akan digunakan dan dilanjutkan dengan menghitung dan menentukan atribut yang digunakan sebagai akar dari pohon keputusan. Dilakukan perhitungan *entropy* dan *gain* untuk menentukan *lead node* dari pohon keputusan. *Entropy* adalah jumlah data yang tidak relevan berdasarkan informasi dari suatu dataset. *Gain* adalah informasi yang didapatkan dari *entropy* pada suatu dataset. Berdasarkan nilai *gain* yang telah dihitung, maka pohon keputusan dapat dibentuk. Semakin tinggi nilai *gain* pada suatu atribut maka akan memiliki kedudukan yang lebih tinggi pada pohon keputusan (Harryanto & Hansun, 2017).

Rumus *Entropy*

$$Entropy(S) = \sum_{i=1}^{n} -p_i * log_2 p_i \quad (3.34)$$

S = Himpunan dataset (kasus)

n = Banyaknya partisi S

i = Jumlah partisi S

p_i = Jumlah kasus pada partisi ke-i

Setelah didapatkan nilai *entropy* maka dapat digunakan untuk menghitung *gain* yang menentukan atribut sebagai akar dengan melihat nilai *gain* tertinggi.

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i) \quad (3.35)$$

$|S_i|$ = Jumlah kasus pada partisi ke-i

$|S|$ = Jumlah kasus dalam S

3.5. Data Tidak Seimbang

Ketidak seimbangan data merupakan jumlah data satu kelas terdapat perbedaan jumlah kelas yang signifikan dengan kelas lainnya. Kelas data yang memiliki lebih banyak objek disebut dengan kelas mayoritas dan kelas yang memiliki jumlah sedikit disebut kelas minor. Suatu data dapat dikatakan tidak seimbang ketika perbandingan jumlah data lebih besar atau sama dengan 1:4 (Krawczyk, 2016). Pada penelitian (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) menggunakan sembilan dataset yang tidak seimbang dengan proporsi yang berbeda, dimana rasio rasio terkecilnya pada dataset Pina yaitu 500:268 atau 65% : 35%.

Data yang tidak seimbang dapat menyebabkan model klasifikasi lebih mudah mengklasifikasikan kelas mayoritas dibandingkan kelas minoritas, dan akan menyebabkan nilai akurasi yang lebih tinggi untuk kelas mayoritas atau akan mempengaruhi kinerja model dalam melakukan kelasifikasi sehingga hasilnya akan bias (Hairani, Saputro, & Fadli, K-Means SMOTE untuk Menangani Ketidakseimbangan Kelas dalam Klasifikasi Penyakit Diabetes dengan C4.5, SVM, dan Naive Bayes, 2020). Pada data yang tidak seimbang, umumnya nilai sensitivitas akan rendah karena model lebih condong memprediksi kelas mayoritas. Sedangkan nilai spesifitas akan memiliki nilai yang lebih tinggi karena lebih mudah memprediksi kelas mayoritas dengan benar (Permatasari, Rizki, & Debataraja, 2020).

Data tidak seimbang dapat diatasi dengan menggunakan *oversampling* dan *undersampling*. *Oversampling* bekerja menyeimbangkan kelas minoritas dengan cara menduplikasi kelas minoritas yang sama persis sehingga terjadi *overfitting*. *Undersampling* bekerja menyeimbangkan kelas minoritas dengan menghapus kelas mayoritas sampai distribusinya seimbang, kelemahannya adalah dapat menghapus informasi yang berguna (Hairani, Saputro, & Fadli, 2020). Terdapat beberapa metode *oversampling* yang umum digunakan seperti ROS (*Random Over Sampling*), SMOTE, dan ADASYN. Metode ROS tidak tahan terhadap *overfitting* dan metode ADASYN tidak dapat mempertahankan karakteristik data asli. Sedangkan metode SMOTE tahan terhadap *overfitting* dan dapat mempertahankan karakteristik data asli (Sir & Soepranoto, 2022). Sehingga peneliti menggunakan metode *oversampling* untuk menyeimbangkan data.

3.6. *Synthetic Minority Oversampling Technique* (SMOTE)

SMOTE merupakan metode yang digunakan untuk mengatasi ketidakseimbangan kelas data pada klasifikasi (Hairani, Saputro, & Fadli, K-Means SMOTE untuk Menangani Ketidakseimbangan Kelas dalam Klasifikasi Penyakit Diabetes dengan C4.5, SVM, dan Naive Bayes, 2020). Penggunaan metode SMOTE dapat menyebabkan *overfitting* pada data karena duplikasi pada data minoritas sehingga memungkinkan adanya data latih yang sama (Kasanah, Muladi, & Pujianto, 2019). Rasio data tidak seimbang yang drastis sebesar 1:100 dan 1:1000 atau dapat lebih besar. Beberapa penelitian dengan rasio data kelas 1:10 dapat membuat beberapa metode tidak dapat membangun model dengan baik.

Metode SMOTE dikembangkan untuk menangani data berskala campuran yaitu nominal dan kontinu (Chawla, Bowyer, Hall, & Kegelmeyer, 2002). Metode SMOTE akan menambah data sintetis menggunakan *oversampling* berdasarkan *k* tetangga terdekat pada kelas minoritas, sehingga jumlah data pada kelas minoritas menjadi sama dengan jumlah data kelas mayoritas. Jumlah *k* tetangga terdekat dipilih secara acak dan data sintetis tersebut hanya ditambahkan pada data latih. Rumus SMOTE untuk membangkitkan data sintetis sebagai berikut (Wijayanti, Kencana, & Sumarjana, 2021).

$$X_{new} = X_i + \left(\hat{X}_k - X_i\right) \delta \qquad (3.36)$$

X_{new} = Data sintetis baru

X_i = Data ke-*i* kelas minoritas

$\hat{X}_k$ = Data *k* tetangga terdekat dari X_i

δ = Bilangan acak (0 dan 1)

3.7. Python

Python merupakan Bahasa pemrograman yang bersifat *interpreter, interactive, object oriented* yang dapat dikembangkan untuk membuat perangkat lunak, seperti *internet scripting, systems programming, user interface, product customization,* dan lainnya. *Python* dapat berjalan pada berbagai *platform* seperti *Windows, Mac, Linux,* dan ponsel (Prasetya & Nurviyanto, 2012). Bahasa *python* merupakan Bahasa pemrograman yang bersifat open source sehingga dapat digunakan secara bebas dan gratis yang telah disetujui oleh *Open Source Initiatives (OSI). Python*

memiliki banyak algoritma yang dapat mempermudah perhitungan secara besar dan dapat digunakan untuk mengurutkan, menemukan, mengiris, dan menggabungkan data. *Python* memiliki isi kumpulan modul yang dapat menangani setiap domain masalah dan dapat bekerja dengan Bahasa lain seperti C atau C++. (Santoso, 2010).

3.8. Flask

Flask merupakan *website* framework yang menggunakan Bahasa *python* berfungsi sebagai kerangka kerja aplikasi dan tampilan *website.* Flask termasuk kedalam *microframework* yang tidak memerlukan alat atau pustaka tertentu sehingga komponen seperti form, database, dan sebagainya tidak terpasang *default* pada Flask. Flask merupakan aplikasi yang sangat sederhana tetapi dapat dengan mudah ditambahkan komponen melalui pihak ketiga atau menggunakan ekstensi. Untuk menggunakan Flask harus melakukan instalasi *package management system* yaitu PIP pada *python* yang dapat digunakan untuk mengatur dan meng*install package* berisi modul *python* (Irsyad, 2018).

BAB IV
METODOLOGI PENELITIAN

4.1. Populasi Penelitian

Populasi penelitian ini adalah data transaksi telesales dari perusahaan X. Sampel dari penelitian ini adalah data transaksi telesales selama 4 bulan yang mulai dari 1 Juli 2022 sampai 31 Oktober 2022. Penelitian ini menggunakan data sekunder yang berasal dari database telesales perusahaan X dengan jumlah data transaksi telesales sebanyak 6.116 transaksi dengan 9 variabel.

4.2. Jenis dan Sumber Data

Penelitian ini menggunakan data sekunder yang diambil dari rekapan data penjualan melalui telesales pada perusahaan X. Data yang digunakan merupakan data transaksi konsumen selama 4 bulan yang dikelola oleh departemen bisnis oleh perusahaan X, dimana data transaksi tersebut berisi tentang data konsumen dan data terkait produk yang dibeli.

4.3. Tempat dan Waktu Penelitian

Penelitian ini dilakukan di perusahaan X dimulai dari Oktober 2022 sampai Desember 2022.

4.4. Variabel penelitian

Penelitian ini menggunakan 9 variabel yang dijelaskan dalam table berikut.

Tabel 4.1 Definisi Operasional Variabel

No	Variabel	Definisi Operasional Variabel	Jenis Data
1	*Month*	Menunjukkan transaksi terjadi pada bulan apa	Kategorik
2	*Grade*	Tingkatan kelas sekolah konsumen, dengan kategori berikut 1 = SD 2 = Kelas 1 SMP 3 = Kelas 2 SMP 4 = Kelas 3 SMP 5 = Kelas 1 SMA 6 = Kelas 2 SMA	Kategorik

No	Variabel	Definisi Operasional Variabel	Jenis Data
		7 = Kelas 3 SMA 8 = Sudah menempuh pendidikan formal	
3	SKU	Paket yang dijual oleh telesales. Terdiri dari 28 kategori.	Kategorik
4	SKU *Type*	Tipe paket yang terdiri dari 0 = murah dan 1 = mahal	Kategorik
5	SKU *Group*	Kelompok dari SKU. Terdiri dari 8 kategori.	Kategorik
6	*Voucher*	Kode promo yang digunakan oleh konsumen. Terdiri dari 10 kategori.	Kategorik
7	Status Cicilan	Keterangan konsumen melakukan pembayaran dengan cicilan atau tidak dengan 0 = lunas dan 1 = cicilan.	Kategorik
8	Harga	Rentang harga paket yang dibeli oleh konsumen yang terdiri dari 1 = murah (<1.047.441), 2 = sedang (1.047.442 – 2.054.883) , dan 3 = mahal (>2.054.884)	Kategorik
9	*Class*	Kelas dari konsumen yang terdiri dari *Class* 0 = tidak melakukan pembelian ulang dan *Class* 1 = melakukan pembelian ulang	Kategorik

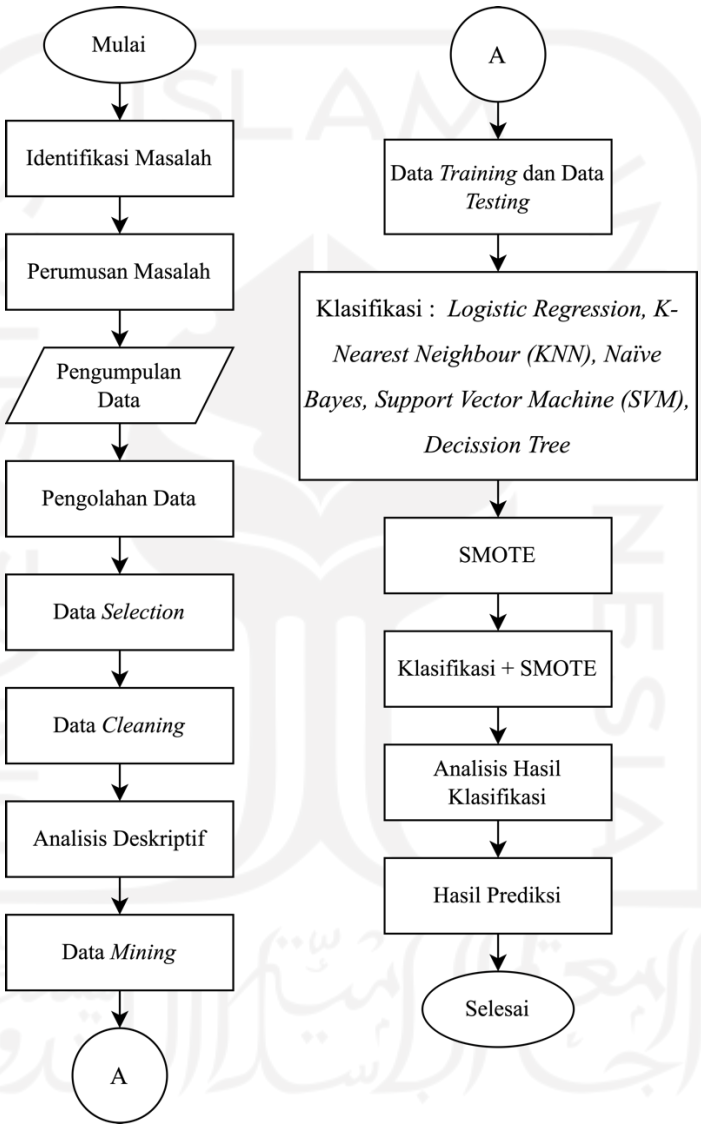
4.5. Metode Analisis

Metode analisis yang digunakan dalam penelitian ini adalah klasifikasi yang terdiri dari beberapa metode yaitu *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree* dengan menggunakan Bahasa pemrograman *Python* pada *Google Colab* yang digunakan untuk mengklasifikasikan konsumen berdasarkan variabel yang ada dan

menggunakan metode SMOTE untuk mengatasi ketidakseimbangan data. Selain itu *software* yang digunakan pada penelitian ini adalah *Microsoft Excel* untuk melakukan pembersihan data.

4.6. Alur Penelitian

Tahapan yang dilakukan pada penelitian ini digambarkan melalui diagram alir atau *flowchart* pada Gambar 4.1 berikut.



Gambar 4.1 Flow Chart Penelitian

Berikut tahapan yang dilakukan pada penelitian :

1. Identifikasi dan Perumusan Masalah

   Dalam penelitian ini terlebih dahulu melakukan identifikasi masalah dan merumuskan masalah yang akan digunakan sebagai latar belakang penelitian.

2. Pengumpulan Data

   Observasi lapangan dilakukan untuk melihat keadaan nyata dari produk yang dijadikan objek penelitian sehingga didapatkan data transaksi.

3. Pengolahan Data

   a. Data *Selection*

      Data yang didapatkan kemudian diambil pada rentang waktu tertentu sehingga dapat membantu proses pengolahan lebih baik.

   b. Data *Cleaning*

      Data *cleaning* merupakan proses pengurangan variabel yang tidak dibutuhkan dalam pengolahan data.

   c. Analisis Deskriptif

      Mengetahui gambaran umum terkait penjualan produk melalui telesales selama empat bulan.

   d. Data *Mining*

      Pengolahan data transaksi telesales menggunakan *software python* dengan tahapan seperti berikut.

      - Data *Training* dan Data *Testing*

        Melakukan pembagian dataset kedalam data *training* sebesar 80% dan data *testing* sebesar 20%.

      - Klasifikasi

        Melakukan klasifikasi menggunakan beberapa metode antara lain *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree* untuk mendapatkan nilai *accuracy, recall, precision*, dan *f1-score* dari setiap model yang terbentuk.

      - SMOTE

        Melakukan *balancing* data untuk menyeimbangkan kelas 0 dan 1 menggunakan metode SMOTE.

      - Klasifikasi + SMOTE

        Melakukan klasifikasi ulang setelah dilakukan SMOTE menggunakan lima metode sebelumnya.

   e. Analisis Hasil

Didapatkan hasil *accuracy, recall, precision*, dan *f1-score* yang selanjutnya dipilih model terbaik untuk melakukan prediksi.

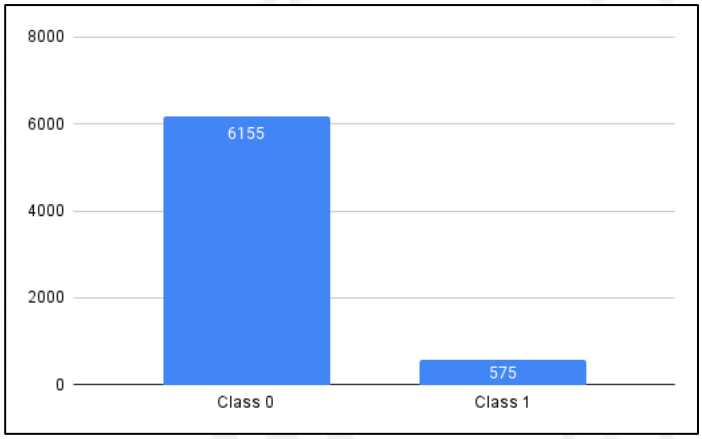
f. Hasil Prediksi

Melakukan prediksi data baru menggunakan model yang telah dipilih.

BAB V
HASIL DAN PEMBAHASAN

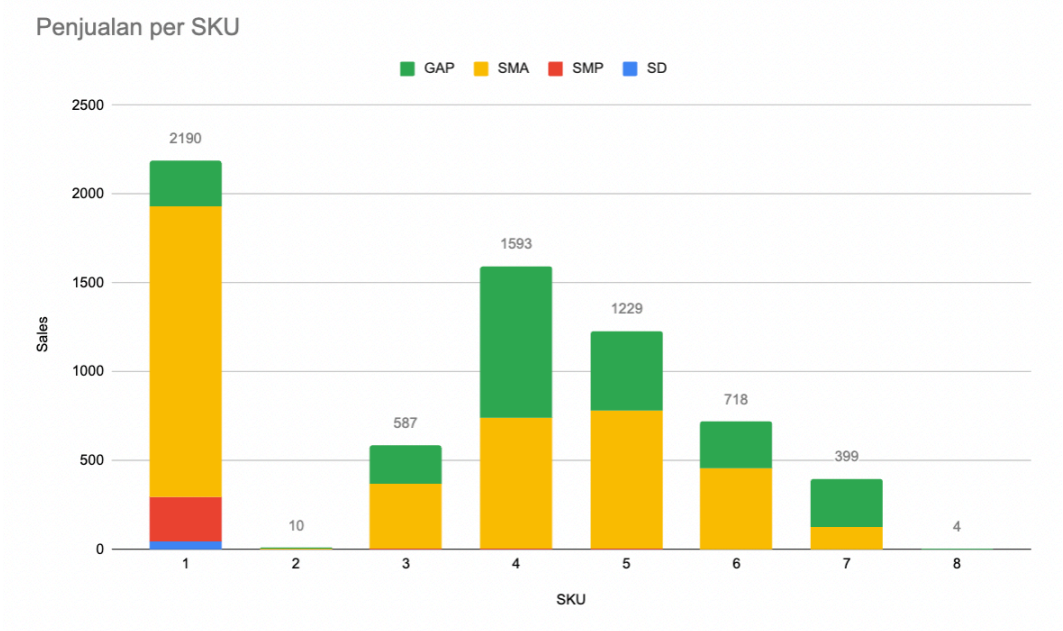
5.1. Statistika Deskriptif

Penelitian ini menggunakan analisis deskriptif untuk mengetahui gambaran umum terhadap data pelanggan perusahaan X. Analisis deskriptif yang digunakan menggunakan grafik histogram dan crosstab. Data yang digunakan pada penelitian ini sebanyak 6.730 transaksi yang terdiri dari Class 1 yang menunjukkan konsumen melakukan pembelian ulang dan Class 0 yang menunjukkan konsumen tidak melakukan pembelian ulang.



Gambar 5.1 Perbandingan Class 0 dan Class 1

Gambar 5.1 merupakan grafik batang yang menunjukkan banyaknya *class* 0 sebanyak 6.155 atau 91.5% dan *class* 1 sebanyak 575 atau 8.5% dimana frekuensi antara *class* 0 dan *class* 1 terdapat perbedaan yang sangat signifikan. Selanjutnya dilakukan analisis deskriptif menggunakan *crosstab* untuk mengetahui frekuensi dari beberapa variabel yang dianggap berhubungan.



Gambar 5.2 Diagram Penjualan per SKU

Gambar 5.2 menunjukkan frekuensi antara variabel Grade dan Group SKU, dapat diketahui dimana Grade yang paling banyak melakukan pembelian adalah SMA yang terdiri dari *grade* 10, 11, dan 12 sebanyak 4.104 konsumen dan *gap year* atau grade 0 sebanyak 2.315 konsumen, sedangkan konsumen pada tingkat SD melakukan pembelian yang paling sedikit. Group SKU yang paling laku adalah nomor 1 yang telah terjual sebanyak 2.190 produk, sedangkan group SKU 2 dan 8 memiliki penjualan yang sangat rendah. Dilakukan analisis crosstab pada variabel status cicilan dan tipe SKU sehingga didapatkan hasil pada tabel berikut.

Tabel 5.1 Crosstab Status Cicilan dan Tipe SKU

		Status Cicilan		
		0	**1**	**Total**
Tipe SKU	**0**	3678	1336	5015
	1	1031	684	1715
Total		4710	2020	6730

Tabel 5.2 menunjukkan bahwa konsumen yang membeli paket dengan harga murah sebanyak 5.015 konsumen dan paket dengan harga mahal sebanyak 1.715 konsumen. Sehingga, konsumen cenderung membeli paket dengan harga yang murah dibandingkan dengan paket yang mahal. Konsumen memiliki kecenderungan membeli paket yang murah dengan menggunakan cicilan sebanyak

3.678 atau 55% konsumen. Konsumen secara keseluruhan lebih memilih menggunakan cicilan untuk melakukan pembelian paket sebanyak 4.710 atau 70% konsumen dan sebanyak 2.020 atau 30% konsumen memilih melakukan pembayaran langsung.

5.2. *Preprocessing*

Preprocessing data merupakan proses mengolah data sebelum dilakukan analisis atau pemodelan untuk mengatasi masalah data missing, data duplikat, data tidak berkualitas sehingga data dapat dilakukan analisis lanjutan dengan baik. Proses preprocessing yang dilakukan terdiri dari beberapa tahapan yaitu variabel *selection*, *labeling* data, menghapus data yang tidak lengkap.

5.2.1 Variabel *Selection*

Pada proses variabel *selection* adalah memilih beberapa variabel yang akan dilakukan analisis lanjutan yang dapat memberikan hasil prediksi yang lebih akurat. Proses ini mengambil 9 dari 18 variabel pada data transaksi. Tabel menunjukkan variabel yang akan digunakan dalam penelitian ini.

Tabel 5.2 Daftar variabel

No	Variabel
1	*Month*
2	*Grade*
3	SKU
4	SKU *Type*
5	SKU *Group*
6	*Voucher*
7	Status Cicilan
8	Harga
9	*Class*

Setelah dilakukan pemilihan variabel yang digunakan maka dapat dilakukan *labelling* data untuk menyembunyikan data asli dan memberikan kategori pada data sehingga dapat mempermudah proses analisis.

5.2.2 *Labeling* Data

Pada proses *labeling* data merupakan tahap memberikan label atau kategori pada setiap data untuk menyembunyikan data asli seperti dalam data transaksi sebuah perusahaan agar tetap terjaga privasi konsumen. Selain itu proses *labeling* data juga dapat digunakan untuk mengkategorikan data sehingga dapat memudahkan dalam proses analisis data.

Tabel 5.3 *Labeling* Data

No	Variabel	Label
1	*Month* (X1)	Nomor 1 sampai 12
2	*Grade* (X2)	1 = SD 2 = Kelas 1 SMP 3 = Kelas 2 SMP 4 = Kelas 3 SMP 5 = Kelas 1 SMA 6 = Kelas 2 SMA 7 = Kelas 3 SMA 8 = Sudah menempuh pendidikan formal
3	SKU (X3)	Nomor 1 sampai 28.
4	SKU *Type* (X4)	0 = murah 1 = mahal
5	SKU *Group* (X5)	Nomor 1 sampai 8.
6	*Voucher* (X6)	Nomor 1 sampai 10.
7	Status Cicilan (X7)	0 = lunas 1 = cicilan.
8	Harga (X8)	1 = murah (<1.047.441) 2 = sedang (1.047.442 – 2.054.883) 3 = mahal (>2.054.884)
9	*Class* (Y)	0 = tidak melakukan pembelian ulang 1 = melakukan pembelian ulang

Setelah proses *labeling* dilakukan maka data sudah siap untuk dilakukan analisis lanjutan dengan terlebih dahulu membagi data yang ada kedalam data *training* dan data *testing*.

5.3. Data *Training* dan Data *Testing*

Peneliti menggunakan perbandingan data *training* sebesar 80% dan data *testing* sebesar 20%. Penggunaan rasio tersebut dikarenakan peneliti telah menguji data dalam beberapa rasio seperti 80%:20%, 70%:30, 85%:15%, 75%:25%, dan didapatkan bahwa nilai akurasi terbaik dengan menggunakan rasio 80%:20%. Didapatkan hasil pembagian data *training* dan *testing* sebagai berikut.

Data *Training* = 80% x 6730 = 5384 transaksi

Data *Testing* = 20% x 6730 = 1346 transaksi

Tabel 5.4 Data *Testing* dan *Training*

Class	Data *Training* (80%)	Data *Testing* (20%)	Total
1	2692	673	3365
0	2692	673	3365
Total	5384	1346	6730

Data yang digunakan dalam penelitian ini sebanyak 6.730 transaksi yang terdiri dari *class* 1 dan *class* 0. Data yang ada dibagi kedalam data *training* sebesar 80% atau 5.384 transaksi dan data *testing* sebesar 20% atau 1.346 transaksi. Data *training* dan *testing* kemudian dibagi lagi kedalam kategori *class* 1 dan *class* 0. Setelah melakukan pembagian data *training* dan *testing* kemudian dilakukan klasifikasi dengan menggunakan beberapa metode.

5.4. Klasifikasi

Penelitian ini menggunakan beberapa metode klasifikasi untuk mengetahui metode klasifikasi manakah yang paling baik. Metode yang digunakan yaitu *Logistic Regression, K-Nearest Neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM),* dan *Decision Tree.* Klasifikasi dengan menggunakan metode KNN dilakukan beberapa kali dengan menggunakan nilai *k* yang berbeda-beda yaitu 2, 3, 4, dan 5. Metode klasifikasi yang baik dapat dilihat dari nilai *accuracy, precision, f1-score, sensitifity,* dan *specificity.* Nilai *accuracy* dan *f1-score* memiliki fungsi yang sama yaitu mengukur kinerja model, nilai *accuracy* sendiri mengukur nilai kebenaran secara keseluruan yang dihasilkan oleh prediksi model tanpa peduli

positif atau negatif. Sedangkan nilai *f1-score* mempertimbangkan nilai positif dan negatif dari prediksi model dan memberikan keseimbangan antara keduanya.

Tabel 5.5 Nilai Klasifikasi

Metode Klasifikasi	Class	Accuracy	Specificity	Sensitivity	AUC	Precision	F1-Score
Regresi Logistic	0	0.91	1.00	0	0.50	0.91	0.96
	1					0	0
KNN k=12	0	0.91	1.00	0	0.51	0.92	0.96
	1					0.50	0.05
KNN k=2	0	0.90	0.98	0.07	0.52	0.92	0.95
	1					0.24	0.11
KNN k=3	0	0.88	0.96	0.09	0.52	0.92	0.94
	1					0.16	0.11
KNN k=4	**0**	**0.91**	**0.99**	**0.05**	**0.52**	**0.92**	**0.95**
	1					**0.27**	**0.09**
KNN k=5	0	0.90	0.98	0.09	0.53	0.92	0.95
	1					0.25	0.13
Naïve Bayes	0	0.91	1	0	0.5	0.91	0.96
	1					0	0
SVM	0	0.91	1	0	0.5	0.91	0.96
	1					0	0
Decisionn Tree	0	0.90	0.97	0.10	0.54	0.92	0.97
	1					0.26	0.10

Dari hasil klasifikasi pada tabel 5.4 dapat diketahui metode yang memiliki nilai *accuracy* yang tinggi adalah Regresi Logistic, KNN, Naïve Bayes, dan SVM sebesar 0.91. Keempat model tersebut dapat memprediksi dengan benar terhadap keseluruhan data sebesar 91%.

Nilai evaluasi diatas didapatkan dengan melakukan perhitungan pada *confusion matrix,* untuk metode KNN *k*=4 didapatkan *confusion matrix* seperti berikut dan dilakukan perhitungan untuk *precision*, *recall*, dan *F1-score* pada setiap kelas.

KNN k=4	Negatif	Positif
Negatif	1215 (TN)	16 (FP)
Positif	109 (FN)	6 (TP)

- *Accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{6 + 1215}{6 + 1215 + 16 + 109} = 0.91$$

- *Sensitivity*

$$Sensitivity = \frac{TP}{TP + FN} = \frac{6}{6 + 109} = 0.05$$

- *Spesificity*

$$Spesificity = \frac{TN}{TN + FP} = \frac{1215}{1215 + 16} = 0.99$$

- AUC

$$FPR = \frac{FP}{FP + TN} = \frac{16}{16 + 1215} = 0.01$$

$$AUC = \left[\frac{1 + sensitifity - FPR}{2}\right] = \left[\frac{1 + 0.05 - 0.01}{2}\right] = 0.52$$

Class 0 (Negatif)

- *Precision*

$$precision = \frac{TN}{TN + FN} = \frac{1215}{1215 + 109} = 0.92$$

- *F1-Score*

$$F1 - Score = \frac{2 \times precision \times spesificity}{precision + spesificity} = \frac{2 \times 0.92 \times 0.99}{0.92 + 0.99} = 0.95$$

Class 1 (Positif)

- *Precision*

$$precision = \frac{TP}{TP + FP} = \frac{6}{6 + 16} = 0.27$$

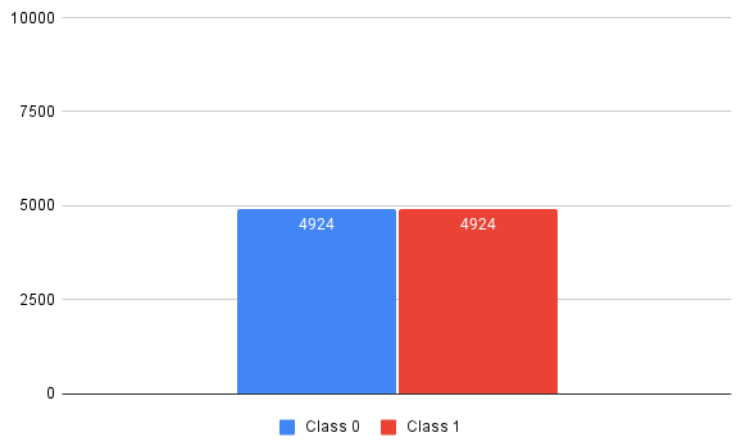
- *F1-Score*

$$F1 - Score = \frac{2 \times precision \times sensitivity}{precision + sensitivity} = \frac{2 \times 0.27 \times 0.05}{0.27 + 0.05} = 0.09$$

Model secara keseluruhan memiliki nilai *precision*, dan *F1-score* yang tinggi pada *class* 0 tetapi pada *class* 1 memiliki nilai yang rendah. Hal ini dapat terjadi karena dataset yang digunakan tidak seimbang, dimana *class* 0 memiliki jumlah data yang banyak sehingga dapat memiliki nilai *precision*, dan *F1-score* yang cukup tinggi. Sedangkan pada data pada *class* 1 cukup sedikit sehingga didapatkan nilai *precision*, dan *F1-score* yang rendah. Selain itu, nilai sensitifitas cukup rendah karena model lebih condong memprediksi kelas mayoritas. Sedangkan nilai spesivitas akan memiliki nilai yang lebih tinggi karena lebih mudah memprediksi kelas mayoritas dengan benar. Karena data yang digunakan merupakan data yang tidak seimbang, sehingga model yang dihasilkan hanya dapat memprediksi kelas mayoritas (*class* 0) dengan baik tetapi tidak dengan kelas minoritas (*class* 1). Untuk mengatasi perbedaan kelas yang tidak seimbang maka dilakukan *balancing* data dengan menggunakan metode SMOTE.

5.5. Klasifikasi + SMOTE

SMOTE merupakan metode untuk menyeimbangkan jumlah kelas minoritas dengan membuat data sintesis secara acak sehingga memiliki jumlah yang sama dengan kelas mayoritas.



Gambar 5.3 Perbandingan Class 0 dan Class 1 dengan SMOTE

Setelah dilakukan SMOTE pada Gambar 5.2 dapat diketahui bahwa *class* 0 dan *class* 1 memiliki data yang seimbang yaitu sebesar 4924 data sehingga didapatkan perbandingan jumlah data antara *class* 0 dan *class* 1 sebesar 50%:50%.

Data yang telah seimbang dapat dilakukan klasifikasi ulang dengan menggunakan metode sebelumnya.

Tabel 5.6 Nilai Klasifikasi+SMOTE

Metode Klasifikasi	Class	Accuracy	Specificity	Sensitivity	AUC	Precision	F1-Score
Regresi Logistic	0	0.61	0.61	0.54	0.58	0.93	0.74
	1					0.12	0.19
KNN k=12	0	0.86	0.92	0.19	0.56	0.92	0.92
	1					0.19	0.19
KNN k=2	0	0.89	0.97	0.08	0.52	0.92	0.94
	1					0.20	0.11
KNN k=3	0	0.87	0.94	0.12	0.53	0.92	0.93
	1					0.17	0.14
KNN k=4	**0**	**0.89**	**0.96**	**0.12**	**0.54**	**0.92**	**0.94**
	1					**0.22**	**0.16**
KNN k=5	0	0.88	0.96	0.10	0.53	0.92	0.94
	1					0.18	0.13
Naïve Bayes	0	0.62	0.63	0.55	0.58	0.94	0.75
	1					0.12	0.20
SVM	0	0.60	0.60	0.58	0.59	0.94	0.73
	1					0.12	0.20
Decisionn Tree	0	0.72	0.74	0.46	0.60	0.94	0.83
	1					0.14	0.22

Dari hasil klasifikasi dengan menggunakan SMOTE maka dapat diketahui nilai *accuracy* pada semua metode klasifikasi mengalami penurunan. Tetapi, terdapat kenaikan yang tidak terlalu signifikan pada nilai *precision, specificity, sensitifity,* dan *f1-score* pada *class* 0 meskipun telah dilakukan SMOTE sehingga proporsi data antara *class* 0 dan *class* 1 seimbang yaitu 50% : 50%. Hal ini dapat terjadi karena data *noise* dan *overfitting*. Data *noise* disebabkan karena sampel sintetis yang dihasilkan oleh SMOTE tidak sepenuhnya mempresentasikan kelas minoritas, dikarenakan terdapat kesalahan yang dimasukkan ke dalam sampel sintetis tersebut. Sedangkan dalam penggunaan SMOTE juga dapat menyebabkan *overfitting* karena sampel sintetis yang dihasilkan terlalu besar. Selain itu juga dapat

disebabkan oleh beberapa faktor seperti pada tahap *preprocessing* menghilangkan data terlalu banyak sehingga dapat menurunkan tingkat akurasi karena menghilangkan informasi penting yang ada pada data, pemilihan algoritma klasifikasi yang tepat, dan pemilihan variabel yang sesuai. Pemilihan algoritma yang tepat akan mempengaruhi tingkat akurasi pada suatu model seperti pada algoritma KNN memiliki nilai akurasi yang paling baik dibandingkan dengan algoritma lainnya. Semakin kecil nilai *k* maka tetangga terdekat yang digunakan dalam pengambilan keputusan semakin sedikit sehingga lebih sensitif terhadap data latih dan dapat meningkatkan keakuratan hasil prediksi. Nilai *precision, recall,* dan *f1-score* untuk KNN k=4 dapat dihitung dari confusion matriks berikut

KNN k=4	Negatif	Positif
Negatif	1180 (TN)	51 (FP)
Positif	101 (FN)	14(TP)

- *Accuracy*

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{14 + 1180}{14 + 1180 + 51 + 101} = 0.89$$

- *Sensitivity*

$$Sensitivity = \frac{TP}{TP + FN} = \frac{14}{14 + 101} = 0.12$$

- *Spesificity*

$$Spesificity = \frac{TN}{TN + FP} = \frac{1180}{1180 + 51} = 0.96$$

- AUC

$$FPR = \frac{FP}{FP + TN} = \frac{51}{51 + 1180} = 0.04$$

$$AUC = \left[\frac{1 + sensitifity - FPR}{2}\right] = \left[\frac{1 + 0.12 - 0.04}{2}\right] = 0.54$$

Class 0 (Negatif)

- *Precision*

$$precision = \frac{TN}{TN + FN} = \frac{1180}{1180 + 101} = 0.92$$

- *F1-Score*

$$F1 - Score = \frac{2 \times precision \times spesificity}{precision + spesificity} = \frac{2 \times 0.92 \times 0.96}{0.92 + 0.96}$$
$$= 0.94$$

Class 1 (Positif)

- *Precision*

$$precision = \frac{TP}{TP + FP} = \frac{14}{14 + 51} = 0.22$$

- *F1-Score*

$$F1 - Score = \frac{2 \times precision \times sensitivity}{precision + sensitivity} = \frac{2 \times 0.22 \times 0.12}{0.22 + 0.12}$$
$$= 0.16$$

Algoritma KNN *k=4* memiliki nilai akurasi yang stabil dibandingkan dengan algoritma lainnya. Dimana nilai akurasi KNN *k*=4 pada klasifikasi tanpa SMOTE sebesar 0.91 dan pada klasifikasi dengan SMOTE sebesar 0.89. Model yang akan digunakan dalam klasifikasi konsumen pada perusahaan X adalah KNN *k*=4 karena memiliki nilai akurasi yang lebih tinggi sehingga dapat mengidentifikasi konsumen yang memiliki potensi dalam pembelian ulang lebih banyak, meskipun nilai *recall* dan *precision* yang rendah.

5.6. Hasil Klasifikasi

Penelitian ini menggunakan beberapa metode klasifikasi untuk mengetahui metode terbaik dalam melakukan prediksi hasil klasifikasi. Beberapa metode klasifikasi memiliki model yang dapat digunakan secara langsung sehingga dapat memprediksi secara manual, seperti regresi logistik dan *decision tree*.

5.6.1 Model Regresi Logistik

Klasifikasi menggunakan metode regresi logistik didapatkan model klasifikasi sebelum SMOTE seperti berikut.

Tabel 5.7 Model Regresi Logistik

Variabel	B	P-Value	Odds Ratio
Intercept	-2.7495	0.000	-
X1	0.2443	0.000	1.2767
X2	0.1616	0.000	1.1754
X3	-0.0305	0.007	0.9700
X4	0.9286	0.000	2.5310
X5	0.1538	0.003	1.1663

Variabel	B	P-Value	Odds Ratio
X6	-0.0515	0.011	0.9498
X7	-0.1225	0.288	0.8847
X8	-0.7274	0.000	0.4832

$$\pi(x) = \left(\frac{e^{-2.7495+0.2443(X1)+0.1616(X2)-0.0305(X3)+0.9286(X4)+0.1538(X5)-0.0515(X6)-0.7274(X8)}}{1+e^{-2.7495+0.2443(X1)+0.1616(X2)-0.0305(X3)+0.9286(X4)+0.1538(X5)-0.0515(X6)-0.7274(X8)}}\right)$$

Model logit

$$g(x) = -2.7495 + 0.2443(X1) + 0.1616(X2) - 0.0305(X3) + 0.9286(X4) + 0.1538(X5) - 0.0515(X6) - 0.7274(X8)$$

Model klasifikasi regresi logistik tanpa SMOTE terdapat satu variabel yang tidak signifikan yaitu variabel Status Cicilan karena karena memiliki nilai signikansi lebih dari 0.05. Berikut model klasifikasi regresi logistik dengan SMOTE.

Tabel 5.8 Model Regresi Logistik SMOTE

Variabel	B	P-Value	Odds Ratio
X1	0.2052	0.000	1.2278
X2	0.1321	0.000	1.1412
X3	-0.0158	0.002	0.9843
X4	0.9191	0.000	2.5070
X5	0.1115	0.000	1.1180
X6	-0.0063	0.000	0.9937
X7	-0.5657	0.000	0.5680
X8	-0.7218	0.000	0.4859

$$\pi(x) = \left(\frac{e^{0.2052(X1)+0.1321(X2)-0.0158(X3)+0.9191(X4)+0.1115(X5)-0.0063(X6)-0.5657(X7)-0.7218(X8)}}{1+e^{0.2052(X1)+0.1321(X2)-0.0158(X3)+0.9191(X4)+0.1115(X5)-0.0063(X6)-0.5657(X7)-0.7218(X8)}}\right)$$

Model logit

$$g(x) = 0.2052(X1) + 0.1321(X2) - 0.0158(X3) + 0.9191(X4) + 0.1115(X5) - 0.0063(X6) - 0.5657(X7) - 0.7218(X8)$$

Model klasifikasi regresi logistik setelah dilakukan SMOTE didapatkan bahwa seluruh variabel telah signifikan yang dapat dilihat dari seluruh variabel memiliki nilai p-value <0.05. Nilai *odds ratio* merupakan kecenderungan pada kejadian sukses antara kategori satu dengan kategori lainnya yang menyatakan kecenderungan kategori dengan X=1 adalah berapa kali lipat dibandingkan dengan kategori X=0. Jika nilai *odds ratio* sama dengan 1, maka peluang sukses antar kategori sama. Jika nilai *odds ratio* >1, maka peluang pada kategori pertama (0)

lebih besar daripada kategori kedua (1). Jika nilai *odds ratio* <1, maka peluang pada kategori pertama (0) lebih kecil daripada kategori kedua (1).

Berdasarkan nilai *odds ratio* maka dapat dilakukan untuk setiap variabel prediktor. Berikut beberapa interpretasi berdasarkan nilai *odds ratio* yang telah didapatkan. Pada varibel *SKU Type* (X4) konsumen yang melakukan pembelian paket murah akan melakukan pembelian ulang sebesar 2.5 kali dibandingkan konsumen yang melakukan pembelian paket mahal. Pada variabel status cicilan (X7) menunjukkan konsumen yang melakukan pembayaran cicilan akan melakukan pembelian ulang sebesar 0.5 kali dibandingkan konsumen yang melakukan pembayaran lunas. Dari model yang didapatkan diatas kita dapat melakukan prediksi dalam menentukan kelas seperti pada contoh berikut.

Contoh

Seorang siswa SMA kelas 12 melakukan pembelian pada bulan Oktober dengan membeli produk nomor 26 yang merupakan group produk nomor 4 dan termasuk tipe murah. Siswa tersebut melakukan pembayaran lunas dengan menggunakan nomor voucher 9 dengan harga pada rentang 1. Apakah siswa tersebut akan melakukan pembelian ulang?

Regresi logistik

$$g(x) = -2.7495 + 0.2443(4) + 0.1616(7) - 0.0305(26) + 0.9286(4) + 0.1538(4) - 0.0515(9) - 0.7274(1) = 1.7061$$

$$P(Y = 1|X) = \left(\frac{1}{1+e^{(1.7061)}}\right) = 0.1537$$

$$P(Y = 0|X) = 1 - \mathrm{P}(Y = 1|X) = 1 - 0.1537 = 0.8463$$

Proposi pada kelas 0 lebih tinggi yaitu 0.84 dibandingkan kelas 1, sehingga siswa tersebut tidak melakukan pembelian ulang.

Regresi logistik dengan SMOTE

$$g(x) = 0.2052(4) + 0.1321(7) - 0.0158(26) + 0.9191(0) + 0.1115(4) - 0.0063(9) - 0.5657(0) - 0.7218(1) = 0.4443$$

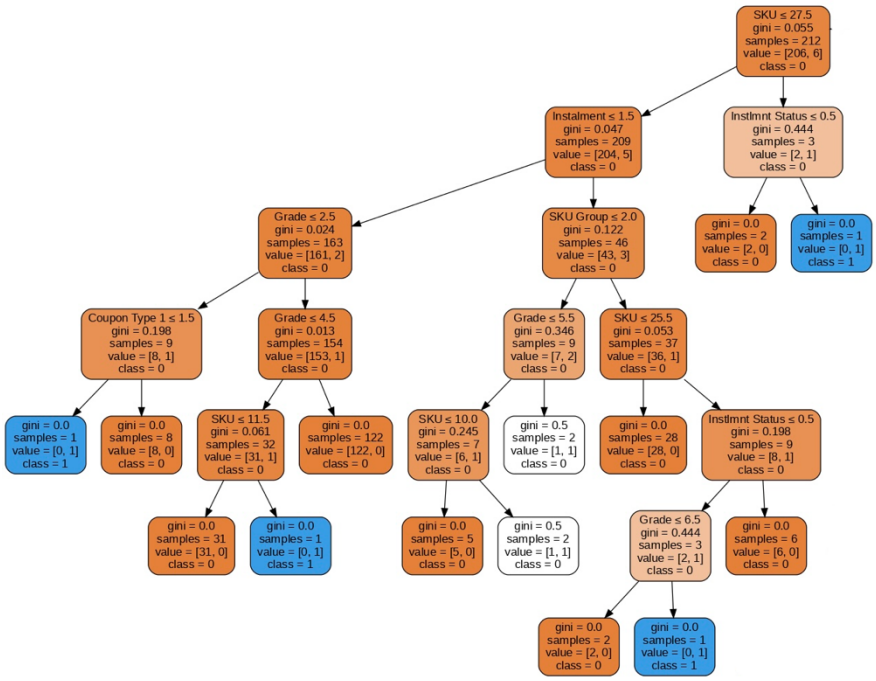
$$P(Y = 1|X) = \left(\frac{1}{1+e^{(0.4443)}}\right) = 0.3907$$

$$P(Y = 0|X) = 1 - \mathrm{P}(Y = 1|X) = 1 - 0.3907 = 0.609$$

Proposi pada kelas 1 lebih tinggi yaitu 0.609 dibandingkan kelas 0, sehingga siswa tersebut akan melakukan pembelian ulang.

5.6.2 Model Decision Tree

Klasifikasi menggunakan metode *decision tree* akan didapatkan hasil berupa model berbentuk pohon untuk membantu dalam mengambil keputusan. Model *decision tree* sebelum SMOTE didapatkan hasil seperti berikut.

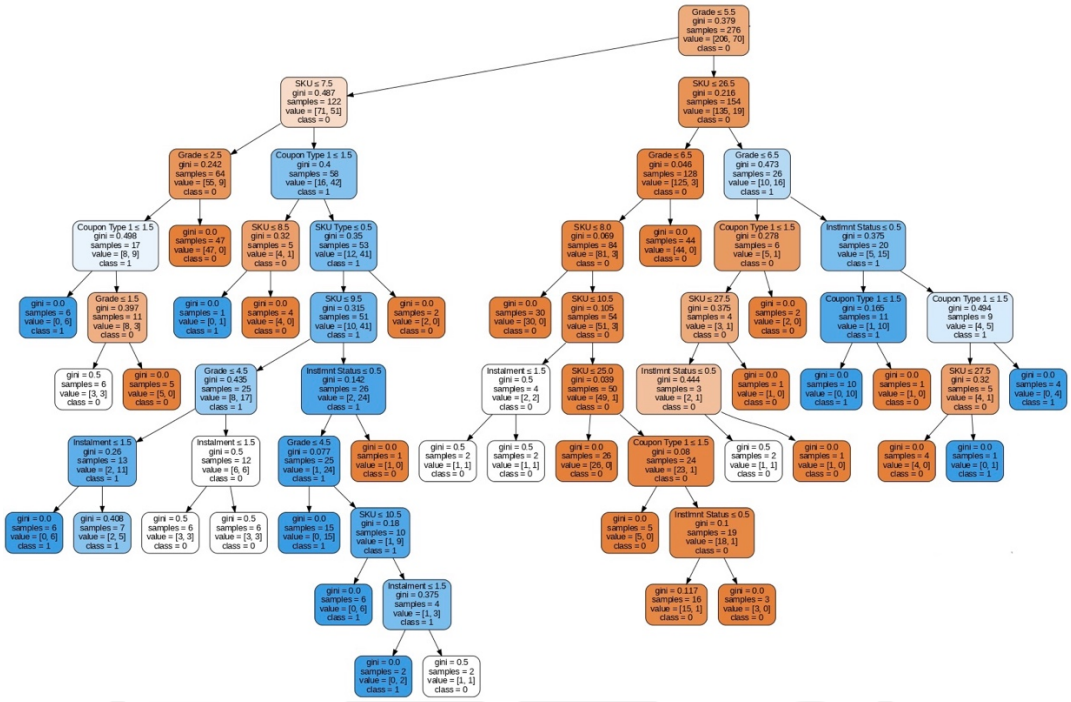


Gambar 5.4 Model Decision Tree

Pada model diatas node dengan warna orange menunjukkan class 0 dan node berwarna biru menunjukkan class 1. Nilai Rasio Gini digunakan untuk mengukur ketidakmurnian atau impurity dalam sebuah set kelas. Rasio Gini memiliki nilai antara 0 dan 1, dengan nilai 0 menunjukkan bahwa semua contoh pada set kelas tersebut adalah anggota kelas yang sama dan nilai 1 menunjukkan bahwa contoh pada set kelas tersebut dibagi sama rata di antara dua kelas.

Dalam mengambil keputusan prediksi, pada node akan terdapat suatu kondisi dan terdapat dua panah. Panah ke kiri menunjukkan bahwa kondisi tersebut benar dan penah ke kanan menunjukkan bahwa kondisi tersebut salah.

Model klasifikasi *decision tree* pada gambar 5.4 lebih banyak memiliki node berwarna orange (*class* 0) dibandingkan node berwarna biru (*class* 1) yang disebabkan oleh ketidak seimbangan data sehingga model lebih mundah memprediksi *class* 0 dibandingkan *class* 1. Berikut model klasifikasi *decision tree* setelah dilakukan *balancing* data menggunakan SMOTE.



Gambar 5.5 Moden Decision Tree SMOTE

Model decision tree setelah dilakukan SMOTE didapatkan node berwarna orange (*class* 0) dan node berwarna biru (*class* 1) tidak setimpang model decision tree sebelumnya. Berikut contoh klasifikasi dengan menggunakan model *decision tree* dengan SMOTE.

Seorang siswa SMA kelas 12 (kategori 7) melakukan pembelian pada bulan Oktober dengan membeli produk nomor 26 yang merupakan group produk nomor 4 dan termasuk tipe murah. Siswa tersebut melakukan pembayaran lunas dengan menggunakan nomor voucher 9 dengan harga pada rentang 1. Apakah siswa tersebut akan melakukan pembelian ulang?

Untuk mengklasifikasikan siswa tersebut dengan terlebih dahulu mengecek kondisi pada node tertinggi. Siswa tersebut berada pada kelas 12 (kategori 7) sehingga tidak memenuhi kondisi Grade ≤ 5.5, jika tidak terpenuhi maka menggunakan panah sebelah kanan sehingga bertemu node dengan kondisi baru yaitu SKU ≤ 26.5. Siswa tersebut melakukan pembelian SKU nomor 26 sehingga kondisi tersebut terpenuhi, jika terpenuhi maka menggunakan panah kiri sehingga bertemu node Grade ≤6. Kondisi tersebut tidak terpenuhi karena siswa berada pada kelas 12 (kategori 7) sehingga menggunakan panah kanan dan bertemu node

terakhir dengaan class 0. Sehingga siswa tersebut diprediksi tidak melakukan pembelian ulang.

5.7. Flask

Model klasifikasi terbaik yang diperoleh kemudian diimplementasikan dalam website dengan menggunakan *Flask*. *Flask* merupakan web *framework* yang ditulis dengan menggunakan Bahasa *python*. Model klasifikasi yang diimplementasikan dengan flask adalah model klasifikasi dengan SMOTE yaitu KNN K=2. Model terbaik yang didapatkan kemudian disimpan dengan menggunakan fungsi *pickle* untuk dapat dirancang dalam *web framework* yang disimpan dalam file 'model.pkl'. Model tersebut dapat digunakan setiap saat untuk mengklasifikasikan data baru tanpa harus melakukan tahapan analisis dari awal.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
import pickle


# Load the csv file
df = pd.read_csv("DATA TA! - 2.csv")

print(df.head())

# Select independent and dependent variable
X = df[["Month", "Grade", "SKU", "SKU Type", "SKU Group", "Coupon Type 1", "Instlmnt Status", "Instalment"]]
y = df["Class"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

#SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(X_train, y_train)

# Instantiate the model
classifier = KNeighborsClassifier(n_neighbors=4)

# Fit the model
classifier.fit(X_resampled, y_resampled)

# Make pickle file of our model
pickle.dump(classifier, open("model.pkl", "wb"))
```

Gambar 5.6 *save* model

Klasifikasi pada pengguna dipengaruhi oleh variabel respon seperti kelas, SKU, kupon, harga, dan lainnya. Oleh karena itu diperlukan wadah yang berisi opsi data tersebut dalam bentuk *form* html. Desain website dibangun dengan menggunakan *sublime,* dimana variabel variabel respon dirancang dalam bentuk *form*. File rancangan tersebut diberi nama '*index.html'* dan diletakkan pada folder *templates.*

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Clasification</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

</head>

<body>
 <div class="login">
    <h1>Class Prediction</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">

        <select name="Month" id="month" required> ...
        </select>

        <select name="Grade" id="grade" required> ...
        </select>

        <select name="SKU" id="sku" required> ...
        </select>

        <select name="SKU Type" id="sku_type" required> ...
        </select>

        <select name="SKU Group" id="sku_group" required> ...
        </select>

        <select name="Coupon Type 1" id="coupon_type_1" required> ...
        </select>

        <select name="Instlmnt Status" id="instlmnt_status" required> ...
        </select>

        <select name="Instalment" id="instalment" required> ...
        </select>

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

   {% if prediction == 1 %}
   <p>User Melakukan Pembelian Ulang</p>
   {% elif prediction == 0 %}
   <p>User Tidak Melakukan Pembelian Ulang</p>
   {% endif %}
```

Gambar 5.7 *index.html*

Model klasifikasi dalam Bahasa *python* dapat dihubungkan dengan website agar dapat ditampilkan dan diaplikasikan dengan menggunakan *script* yang berisi fungsi *flask* yang diberi nama '*app.py*'. Jika *form* pada website telah diisi *form* akan menyimpan nilai tersebut dan melakukan klasifikasi dengan menggunakan model yang telah ditentukan lalu menampilkan nilai hasil klasifikasi pada halaman website.

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html",prediction=prediction, prediction_text = "user classification {}".format(prediction))

if __name__ == "__main__":
    flask_app.run(debug=True)
```

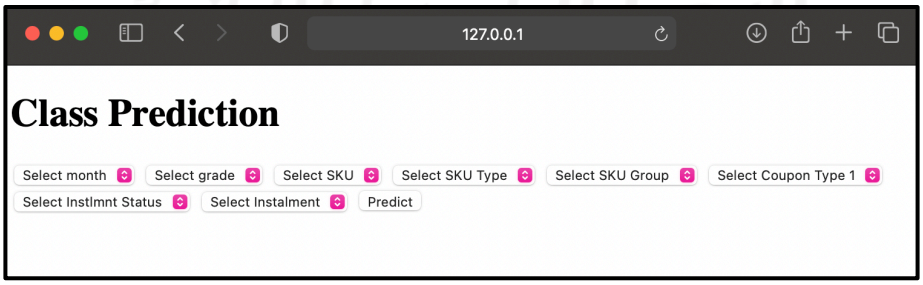
Gambar 5.8 *app.py*

Script yang dibutuhkan untuk melakukan prediksi sudah dirancang maka *prototype* dapat dijalankan di dalam *localhost* dengan mengaktifkan script App.py menggunakan *command prompt* hingga muncul *output* seperti pada gambar 5.6 yang menandakan bahwa sudah berhasil diaktifkan.

```
(base) titania@MacBook-Air-Titania FIX_DEPLOY % python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with fsevents reloader
 * Debugger is active!
 * Debugger PIN: 186-715-650
```

Gambar 5.9 Mengaktifkan *prototype*

Rancangan website aplikasi dapat dibuka pada *browser* menggunakan alamat http://127.0.0.1:5000/ sehingga didapatkan tampilan website yang telah terhubung dengan model klasifikasi dalam bentuk python seperti berikut.



Gambar 5.10 Tampilan aplikasi

Tampilan website yang dirancang terdiri dari judul dan 8 *form* yang berisi pilihan sesuai dengan variabelnya. Terdapat tombol *predict* yang berfungsi untuk menampilkan hasil klasifikasi sesuai data yang dimasukkan kedalam *form* yang

akan muncul hasil user melakukan pembelian ulang atau user tidak melakukan pembelian ulang.

BAB VI
PENUTUP

Berdasarkan hasil dan pembahasan yang telah dipaparkan pada bab sebelumnya, maka diperoleh kesimpulan sebagai berikut.

6.1. Kesimpulan

1. Data yang digunakan merupakan data transaksi telesales perusahaan X yang terdiri dari 6730 transaksi selama 4 bulan. Dari data tersebut terbagi dalam dua kelas, yaitu kelas 1 merupakan konsumen yang melakukan pembelian ulang sebanyak 6155 atau 91.5% dan kelas 0 merupakan konsumen yang tidak melakukan pembelian ulang sebanyak 575 atau 8.5%. Data yang digunakan mengalami ketidakseimbangan data antar kelas, sehingga dilakukan *balancing* data untuk menyeimbangkan jumlah data antara kedua kelas dengan menggunakan SMOTE.
2. Data telesales yang tidak seimbang dapat diatasi dengan *balancing* data menggunakan SMOTE. Metode SMOTE akan menambah data sintetis menggunakan *oversampling* berdasarkan *k* tetangga terdekat pada kelas minoritas, sehingga jumlah data pada kelas minoritas menjadi sama dengan jumlah data kelas mayoritas.
3. Klasifikasi tanpa SMOTE dengan nilai *accuracy* yang tinggi adalah Regresi Logistic, KNN, Naïve Bayes, dan SVM sebesar 0.91. Keempat model tersebut dapat memprediksi dengan benar terhadap keseluruhan data sebesar 91%. Sedangkan nilai akurasi klasifikasi dengan SMOTE pada setiap metode mengalami penurunan dengan nilai akurasi tertinggi yaitu KNN *k*=4 sebesar 89%. Metode KNN k=4 memiliki nilai akurasi yang cukup stabil dengan dan tanpa SMOTE. Sehingga metode terbaik yang untuk data telesales adalah metode KNN *k*=4 SMOTE dengan akurasi 89%.
4. Model KNN K=4 SMOTE memiliki nilai *accuracy* paling tinggi sebesar 89% dan AUC sebesar 54%. Pada *class* 0 memiliki nilai *precision* sebesar 92%, *f1-score* sebesar 94%. Sedangkan pada *class* 1 memiliki nilai

precision sebesar 22%, *f1-score* sebesar 16%. Selain itu didapatkan nilai specificity sebesar 96% dan sensitivity 0.12%.

6.2. Saran

Saran yang dapat diberikan dari hasil penelitian ini antara lain :

1. Menambah jumlah dataset dan memperhatikan keseimbangan jumlah data antar kelas agar mendapatkan nilai akurasi yang lebih baik.
2. Melakukan *split data* pada data *testing* dan data *training* dengan beberapa rasio agar mendapatkan rasio data yang tepat.
3. Melakukan *preprocessing* data secara efektif dan cermat.

DAFTAR PUSTAKA

(t.thn.).

Achmad, B. D., & Slamat, F. (2012). Klasifikasi Data Karyawan Untuk Menentukan Jadwal Kerja Menggunakan Metode Decision Tree. *Jurnal IPTEK, 16*(1), 17-23.

Adawi, R. (2008). Pembelajaran Berbasis E-Learning. *Jurnal Bahasa Unimed*, 53.

Agustina, R., Santosa, P. I., & Ferdiana, R. (2016). Sejarah, Tantangan, Dan Faktor Keberhasilan Dalam Pengembangan E-Learning. *Seminar Nasional Sistem Informasi Indonesia*, 209-218.

Aliady, H., Tuasikal, N. J., & Widodo, E. (2018). IMPLEMENTASI SUPPORT VECTOR MACHINE (SVM) DAN RANDOM FOREST PADA DIAGNOSIS KANKER PAYUDARA. *Seminar Nasional Teknologi Informasi dan Komunikasi 2018 (SENTIKA 2018)* (hal. 278-285). Yogyakarta: SENTIKA 2018.

Annur, H. (2018). Klasifikasi Masyarakat Miskin Menggunakan Metode Naive Bayes. *ILKOM, 10*(2), 160-165.

Ardiyansyah, Rahayuningsih, P. A., & Maulana, R. (2018). Analisis Perbandingan Algoritma Klasifikasi Data Mining untuk Dataset Blogger dengan Rapid Miner. *Jurnal Khatulistiwa Informatika, VI*(1), 20-28.

Barro, R. A., Sulvianti, I. D., & Afendi, F. M. (2013). Penerapan Synthetic Minority Oversampling Technique (SMOTE) Terhadap Data Tidak Seimbang Pada Pembuatan Model Komposisi Jamu. *Xplore, 1*(9), 1-6.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE : Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 321-357.

D.N, A. R., Auliasari, K., & Pranoto, Y. A. (2020). Implementasi Metode K-Nearest Neighbor (KNN) untuk Seleksi Calon Karyawan Baru. *JATI*, 14-20.

Damanik, S. S., Ispriyanti, D., & Sugito. (2015). Klasifikasi Lama Studi Mahasiswa FSM Universitas Diponegoro Menggunakan Regresi Logistik Biner dan Support Vector Machine (SVM). *GAUSSIAN*, 123-132.

Dharmawan, W. S. (2021). Komparasi Algoritma Klasifikasi SVM-PSO dan C4.5-PSO Dalam Prediksi Penyakit Jantung. *Informatika*, 31-41.

Faramudhita, S., Ruswandi, R., & Saidi, S. (2017). Analisis Klasifikasi Menggunakan Metode Regresi Logistik Ordinal dan Klasifikasi Naive Bayes Pada Data Alumni UNILA Tahun 2016. *Seminar Nasional Metode Kuantitatif* (hal. 251-262). Lampung: UNILA.

Ghozali, I. (2016). *Aplikasi Analisis Multivariate dengan Program IBM SPSS 23.* Semarang: Badan Penerbit Universitas Diponegoro.

Hadi, S. (1993). *Metodologi Research.* Yogyakarta: Andi Offset.

Hairani, Saputro, K. E., & Fadli, S. (2020). K-Means SMOTE untuk Menangani Ketidakseimbangan Kelas dalam Klasifikasi Penyakit Diabetes dengan C4.5, SVM, dan Naive Bayes. *Jurnal Teknologi dan Sistem Komputer, 8*(2), 89-93.

Hairani, Saputro, k. E., & Fadli, S. (2020). K-means-SMOTE untuk menangani ketidakseimbangan kelas dalamklasifikasi penyakit diabetes dengan C4.5, SVM, dan naive Bayes. *Jurnal Teknologi dan Sistem Komputer*, 89-93.

Harryanto, F. F., & Hansun, S. (2017). Penerapan Algoritma C4.5 untuk Memprediksi Penerimaan Calon Pegawai Baru di PT WISE. *Jatisi, 3*(2), 95-103.

Harryanto, F. F., & Hansun, S. (2017). Penerapan Algoritma C4.5 untuk Memprediksi Penerimaan Calon Pegawai Baru di PT WISE. *Jatisi*, 95-103.

Iriandi, N. (2013). Komparasi Algoritma Klasifikasi Data Mining Dalam Penentuan Resiko Kredit Pada Koperasi Serba Usaha. *Paradigma, XV*(2), 192-204.

Irsyad, R. (2018). Penggunaan Python Web Framework Flask Untuk Pemula. *OSF Preprints*.

Isman, Ahmad, A., & Latief, A. (2021). Perbandingan Metode KNN Dan LBPH Pada Klasifikasi Daun Herbal. *Jurnal Resti, 5*(3), 557-564.

Kartajaya, H. (2006). *Hermawan Kartajaya on Segmentation Seri 9 Elemen Marketing.* Bandung: PT. Mizan Pustaka.

Kasanah, A. N., Muladi, & Pujianto, U. (2019). Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN. *Jurnal RESTI, 3*(2), 196-201.

Kasanah, A. N., Muladi, & Pujianto, U. (2019). Penerapan Teknik SMOTE untuk Mengatasi Imbalance Class dalam Klasifikasi Objektivitas Berita Online Menggunakan Algoritma KNN. *Jurnal Resti, 3*(2), 196-201.

Kotler, P., & Keller, K. (2012). *Marketing Management.* England: Pearson Education Limited.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Springer*, 221-232.

Mardi, Y. (2016). Data Mining : Klasifikasi Menggunakan Algoritma C4.5. *Jurnal Edik Informatika, 2*(2), 213-219.

Nayak, S., Bhat, M., Reddy, N. S., & Rao, B. A. (2022). Study of distance metrics on k-nearest neighboor algorithm for star categorization. *AICECS 2021*, 1-8.

Nugroho, A. S. (2003). *Pengantar Support Vector Machine.*

Octaviani, P. A., Wilandari, Y., & Ispriyanti, D. (2014). Penerapan Metode Klasifikasi Support Vector Machine (SVM) Pada Data Akreditasi Sekolah Dasar (SD) di Kabupaten Magelang. *Jurnal Gaussian, 3*(4), 811-820.

Parapat, I. M., Furqon, M. T., & Sutrisno. (2018). Penerapan Metode Support Vector Machine (SVM) Pada Klasifikasi Penyimpangan Tumbuh Kembang Anak. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3163-3169.

Permatasari, R. D., Rizki, S. W., & Debataraja, N. N. (2020). Penerapam Synthetic Minority Oversampling Technique dalam Mengatasi Data Tidak Seimbang Pada Metode Classification and Regression Tree. *Bimaster*, 231-238.

Prakasa, O. S., & Lhaksamana, K. M. (2018). Klasifikasi Teks Dengan Menggunakan Algoritma K-Nearest Neighbor pada Kasus Kinerja Pemerintah di Twitter. *e-Proceeding of Engineering, 5*(3), 8237-8248.

Prasetya, D. A., & Nurviyanto, I. (2012). Deteksi Wajah Metode Viola Jones Pada OpenCV Menggunakan Pemrograman Python. *Simposium Nasional, XI*, 18-23.

Putri, E. K., & Setiadi, T. (2014). Penerapan Text Mining Pada Sistem Klasifikasi Email Spam Menggunakan Naive Bayes. *Jurnal Sarjana Teknik Informatika, 2*(3), 73-83.

Ramli, Yuniarti, D., & Goejantoro, R. (2013). Perbandingan Metode Klasifikasi Regresi Logistik Dengan Jaringan Saraf Tiruan. *Jurnal Eksponensial, 4*(1), 17-24.

Risdayanti, & Aidid, M. K. (2020). Analisis Regresi Logistik Biner untuk Menentukan Model Pengguna KB di Kelurahan Langnga Kabupaten Pinrang. *VARIANSI*, 1-8.

Rizal, R. A., Girsang, I. S., & Prasetyo, S. A. (2019). Klasifikasi Wajah Menggunakan Support Vector Machine (SVM). *Remik, 3*(2), 1-4.

Saleh, A. (2015). *Klasifikasi Gejala Depresi Pada Manusia dengan Metode Naive Bayes Menggunakan Java.* Yogyakarta.

Santoso, B. (2010). Bahasa Pemrograman Python di Platform GNU/Linuc. *Ultimatics*, 43-51.

Sir, Y. A., & Soepranoto, A. H. (2022). Pendekatan Resampling Data untuk Menangani Masalah Ketidakseimbangan Kelas. *Jurnal Komputer dan Informatika*, 31-38.

Subroto, B. (2011). *Pemasaran industri (Business to Business Marketing).* Yogyakarta: Andi.

Suryana, N., Pratiwi, & Prasetio, R. T. (2021). Penanganan Ketidak Seimbangan Data pada Prediksi Customer Churn Menggunakan Kombinasi SMOTE dan Boosting. *IJCIT, 6*(1), 31-37.

Tjiptono, F., & Diana, A. (2016). *Pemasaran.* Yogyakarta: Ansi Offset.

Widyastuti, A., Mawati , A. T., Yuniwati, I., Simarmata, J., Pakpahan, A. F., Ardiana, D. Y., . . . Inayah, A. N. (2020). *Pengantar Teknologi Pendidikan.* Yayasan Kita Menulis.

Wijayanti, N. P., Kencana, E. N., & Sumarjana, I. W. (2021). SMOTE: Potensi dan Kekurangannya Pada Survey. *E-Jurnal Matematika*, 235-240.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2011). *Data Mining Practical Machine learning Tools and Techniques.* Burlington: Morgan Kaufmann Publisher.

LAMPIRAN

Link Google Drive : https://bit.ly/19611014_LAMPIRAN

Lampiran 1 Dataset

Month	Grade	SKU	SKU Type	SKU Group	Coupon Type	Instalment Status	Instalment	Class
1	7	19	1	3	9	0	2	0
1	7	17	1	6	9	0	2	0
1	7	21	1	7	9	0	3	0
1	6	21	1	7	9	0	3	0
1	7	27	1	6	9	0	2	0
1	8	19	1	3	9	0	2	0
1	7	19	1	3	9	0	2	0
1	7	28	1	6	9	0	2	0
1	7	9	1	2	9	0	1	0
1	8	27	1	6	5	0	3	0
1	6	19	1	3	9	0	2	0
1	6	28	1	6	9	0	2	0
1	8	28	1	6	9	0	2	0
1	7	28	1	6	9	0	2	0
1	7	19	1	3	9	0	2	0
1	8	19	1	3	9	0	2	0
1	7	27	1	6	9	0	2	0
1	7	19	1	3	9	0	2	0
1	7	28	1	6	9	0	2	0
1	8	9	1	2	9	0	1	0
1	7	27	1	6	9	0	2	0
1	7	18	1	3	9	0	1	0
1	8	9	1	2	9	0	1	1
1	7	19	1	3	9	0	2	0
1	7	9	1	2	9	0	1	0
1	6	20	1	7	10	0	3	0
1	7	27	1	6	9	0	2	0
1	7	27	1	6	9	0	2	0
1	7	21	1	7	9	0	3	0
1	6	20	1	7	9	0	3	0
1	7	19	1	3	9	0	2	0
1	7	28	1	6	6	0	2	0
1	7	16	1	6	7	0	2	0

1	7	17	1	6	9	0	2	0
1	7	19	1	3	9	1	2	0
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
1	7	21	1	7	6	0	3	0
1	6	19	1	3	6	0	2	0
1	8	22	1	7	6	1	3	0
1	7	27	1	6	5	0	3	0
1	8	21	1	7	9	0	3	0
1	6	19	1	3	9	0	2	0
1	7	19	1	3	9	0	2	0
1	8	22	1	7	9	1	3	0
1	7	27	1	6	9	0	3	0
1	7	18	1	3	6	0	2	0
1	8	19	1	3	9	0	2	0
1	7	19	1	3	9	0	2	0
1	8	21	1	7	9	0	3	0
1	7	21	1	7	4	0	1	0

Lampiran 2 Program Python

Split

In [148…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, stratify=data.Class)
```

In [149…

```
# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[774 457]
 [ 58  57]]
              precision    recall  f1-score   support

           0       0.93      0.63      0.75      1231
           1       0.11      0.50      0.18       115

    accuracy                           0.62      1346
   macro avg       0.52      0.56      0.47      1346
weighted avg       0.86      0.62      0.70      1346
```

In [150…

```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[780 451]
 [ 59  56]]
              precision    recall  f1-score   support

           0       0.93      0.63      0.75      1231
           1       0.11      0.49      0.18       115

    accuracy                           0.62      1346
   macro avg       0.52      0.56      0.47      1346
weighted avg       0.86      0.62      0.70      1346
```

In [151…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, stratify=data.Class)
```

In [152…

```
# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[1172  675]
 [  78   94]]
              precision    recall  f1-score   support

           0       0.94      0.63      0.76      1847
           1       0.12      0.55      0.20       172

    accuracy                           0.63      2019
   macro avg       0.53      0.59      0.48      2019
weighted avg       0.87      0.63      0.71      2019
```

In [153…

```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[1146  701]
 [  78   94]]
              precision    recall  f1-score   support

           0       0.94      0.62      0.75      1847
           1       0.12      0.55      0.19       172

    accuracy                           0.61      2019
   macro avg       0.53      0.58      0.47      2019
weighted avg       0.87      0.61      0.70      2019
```

In [154…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, stratify=data.Class)
```

In [155…

```
# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[930 609]
 [ 62  82]]
              precision    recall  f1-score   support

           0       0.94      0.60      0.73      1539
           1       0.12      0.57      0.20       144

    accuracy                           0.60      1683
   macro avg       0.53      0.59      0.47      1683
weighted avg       0.87      0.60      0.69      1683
```

In [156…

```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[933 606]
 [ 62  82]]
              precision    recall  f1-score   support

           0       0.94      0.61      0.74      1539
           1       0.12      0.57      0.20       144

    accuracy                           0.60      1683
   macro avg       0.53      0.59      0.47      1683
weighted avg       0.87      0.60      0.69      1683
```

In [157…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.15, stratify=data.Class)
```

In [160…

```
# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[566 358]
 [ 43  43]]
              precision    recall  f1-score   support

           0       0.93      0.61      0.74       924
           1       0.11      0.50      0.18        86

    accuracy                           0.60      1010
   macro avg       0.52      0.56      0.46      1010
weighted avg       0.86      0.60      0.69      1010
```

In [161…

```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[549 375]
 [ 40  46]]
              precision    recall  f1-score   support

           0       0.93      0.59      0.73       924
           1       0.11      0.53      0.18        86

    accuracy                           0.59      1010
   macro avg       0.52      0.56      0.45      1010
weighted avg       0.86      0.59      0.68      1010
```

DATA

In [ ]:
```
# Import the data
import pandas as pd
data = pd.read_csv('/content/DATA TA! - 2.csv')
data.head()
```

Out[90]:

	Month	Grade	SKU	SKU Type	SKU Group	Coupon Type 1	Instlmnt Status	Instalment	Class
0	1	6	19	1	3	9	0	2	0
1	1	6	17	1	6	9	0	2	0
2	1	6	21	1	7	9	0	3	0
3	1	5	21	1	7	9	0	3	0
4	1	6	27	1	6	9	0	2	0

In [ ]:
```
x = data.drop(['Class'], axis = 1)
x.head()
```

Out[91]:

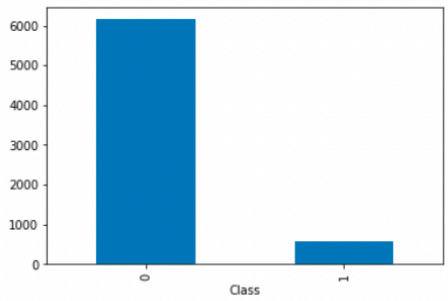
	Month	Grade	SKU	SKU Type	SKU Group	Coupon Type 1	Instlmnt Status	Instalment
0	1	6	19	1	3	9	0	2
1	1	6	17	1	6	9	0	2
2	1	6	21	1	7	9	0	3
3	1	5	21	1	7	9	0	3
4	1	6	27	1	6	9	0	2

In [ ]:
```
y = data['Class']
y.head()
```

Out[92]:
```
0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int64
```

In [ ]:
```
data.pivot_table(index='Class', aggfunc='size').plot(kind='bar')
```

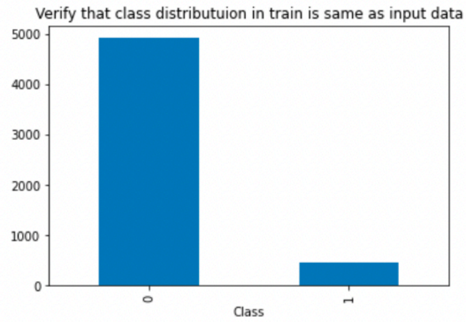
Out[93]:
```
<AxesSubplot:xlabel='Class'>
```



In [ ]:
```
# Create a stratified train/test split.
# Test set will be 30% of the data.
# Class distribution will be equal for train test and original data
#mengaktifkan package sklearn
#membagi data ke dalam train dan test

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, stratify=data.Class
```

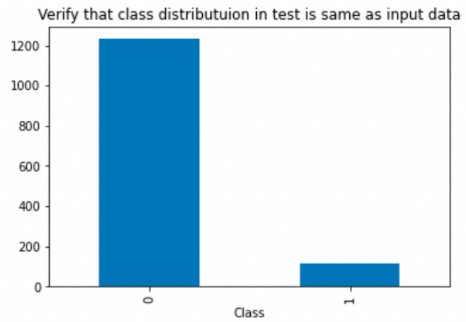
In [ ]:
```
x_train.pivot_table(index=y, aggfunc='size').plot(kind='bar', title='Verify that class distribu
```

Out[95]:
```
<AxesSubplot:title={'center':'Verify that class distributuion in train is same as input dat
a'}, xlabel='Class'>
```



In [ ]:
```
x_test.pivot_table(index=y, aggfunc='size').plot(kind='bar', title='Verify that class distribut
```

Out[96]:
```
<AxesSubplot:title={'center':'Verify that class distributuion in test is same as input dat
a'}, xlabel='Class'>
```



Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score

# Instantiate the Logistic Regression with only default settings
my_log_reg = LogisticRegression()

# Fit the logistic regression on the independent variables of the train data with buy as depend
my_log_reg.fit(x_train, y_train)

# Make a prediction using our model on the test set
preds = my_log_reg.predict(x_test)
```

In [ ]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds))
print(classification_report(y_test, preds))
```

```
[[1231    0]
 [ 115    0]]
              precision    recall  f1-score   support

           0       0.91      1.00      0.96      1231
           1       0.00      0.00      0.00       115

    accuracy                           0.91      1346
   macro avg       0.46      0.50      0.48      1346
weighted avg       0.84      0.91      0.87      1346

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetr
icWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

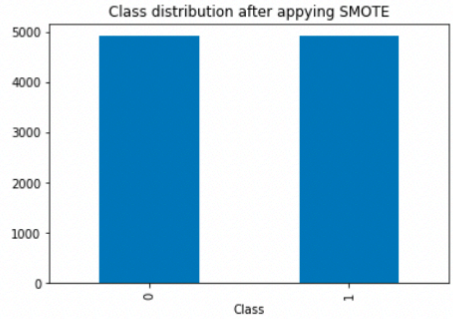
In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, preds)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, preds)
auc = roc_auc_score(y_test, preds)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 1.0
Sensitivity: 0.0
AUC: 0.5
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)
```

In [ ]:
```
pd.Series(y_resampled).value_counts().plot(kind='bar', title='Class distribution after appying
```

Out[112]:
```
<AxesSubplot:title={'center':'Class distribution after appying SMOTE'}, xlabel='Class'>
```



In [ ]:
```
# Instantiate the new Logistic Regression
log_reg_2 = LogisticRegression()

# Fit the model with the data that has been resampled with SMOTE
log_reg_2.fit(X_resampled, y_resampled)

# Predict on the test set (not resampled to obtain honest evaluation)
preds2 = log_reg_2.predict(x_test)
```

In [ ]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preds2))
print(classification_report(y_test, preds2))
```

```
[[756 475]
 [ 53  62]]
              precision    recall  f1-score   support

           0       0.93      0.61      0.74      1231
           1       0.12      0.54      0.19       115

    accuracy                           0.61      1346
   macro avg       0.52      0.58      0.47      1346
weighted avg       0.86      0.61      0.69      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, preds2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, preds2)
auc = roc_auc_score(y_test, preds2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.6141348497156783
Sensitivity: 0.5391304347826087
AUC: 0.5766326422491436
```

KNN

In [ ]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier


# Menentukan daftar nilai k yang akan diuji
k_list = list(range(1, 31))

# Membuat list untuk menyimpan akurasi pada setiap k
cv_scores = []

# Melakukan cross-validation untuk setiap nilai k
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, x, y, cv=31, scoring='accuracy')
    cv_scores.append(scores.mean())

# Menampilkan nilai k dengan akurasi tertinggi
best_k = k_list[cv_scores.index(max(cv_scores))]
print("Best k = ", best_k)


print("k = ", scores)
```

```
Best k =  12
k =  [0.91284404 0.91284404 0.91284404 0.91705069 0.91705069 0.9124424
 0.91705069 0.91705069 0.91705069 0.91705069 0.91705069 0.91705069
 0.9124424  0.91705069 0.91705069 0.91705069 0.91705069 0.9124424
 0.9124424  0.9124424  0.9124424  0.9124424  0.9124424  0.9124424
 0.9124424  0.91705069 0.9124424  0.9078341  0.9124424  0.9124424
 0.9124424 ]
```

KNN K=12

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.spatial.distance import hamming

classifier = KNeighborsClassifier(n_neighbors=12, metric=hamming)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1228    3]
 [ 112    3]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      1231
           1       0.50      0.03      0.05       115

    accuracy                           0.91      1346
   macro avg       0.71      0.51      0.50      1346
weighted avg       0.88      0.91      0.88      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9975629569455727
Sensitivity: 0.02608695652173913
AUC: 0.5118249567336559
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.spatial.distance import hamming

classifier2 = KNeighborsClassifier(n_neighbors=12, metric=hamming)
classifier2.fit(X_resampled, y_resampled)
y_pred2 = classifier2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
[[1135   96]
 [  93   22]]
              precision    recall  f1-score   support

           0       0.92      0.92      0.92      1231
           1       0.19      0.19      0.19       115

    accuracy                           0.86      1346
   macro avg       0.56      0.56      0.56      1346
weighted avg       0.86      0.86      0.86      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred2)
auc = roc_auc_score(y_test, y_pred2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9220146222583265
Sensitivity: 0.19130434782608696
AUC: 0.5566594850422067
```

KNN K=2

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.spatial.distance import hamming

classifier = KNeighborsClassifier(n_neighbors=2, metric=hamming)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1205   26]
 [ 107    8]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      1231
           1       0.24      0.07      0.11       115

    accuracy                           0.90      1346
   macro avg       0.58      0.52      0.53      1346
weighted avg       0.86      0.90      0.88      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9788789601949635
Sensitivity: 0.06956521739130435
AUC: 0.5242220887931339
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.spatial.distance import hamming

classifier2 = KNeighborsClassifier(n_neighbors=2, metric=hamming)
classifier2.fit(X_resampled, y_resampled)
y_pred2 = classifier2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
[[1194   37]
 [ 106    9]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      1231
           1       0.20      0.08      0.11       115

    accuracy                           0.89      1346
   macro avg       0.56      0.52      0.53      1346
weighted avg       0.86      0.89      0.87      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred2)
auc = roc_auc_score(y_test, y_pred2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9699431356620634
Sensitivity: 0.0782608695652174
AUC: 0.5241020026136404
```

KNN = 3

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=3, metric=hamming)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1178   53]
 [ 105   10]]
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      1231
           1       0.16      0.09      0.11       115

    accuracy                           0.88      1346
   macro avg       0.54      0.52      0.52      1346
weighted avg       0.85      0.88      0.87      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9569455727051178
Sensitivity: 0.08695652173913043
AUC: 0.5219510472221242
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn.neighbors import KNeighborsClassifier

classifier2_3 = KNeighborsClassifier(n_neighbors=3, metric=hamming)
classifier2_3.fit(X_resampled, y_resampled)
y_pred2 = classifier2_3.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
[[1161   70]
 [ 101   14]]
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      1231
           1       0.17      0.12      0.14       115

    accuracy                           0.87      1346
   macro avg       0.54      0.53      0.54      1346
weighted avg       0.86      0.87      0.86      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred2)
auc = roc_auc_score(y_test, y_pred2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9431356620633631
Sensitivity: 0.12173913043478261
AUC: 0.5324373962490728
```

KNN = 4

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=4, metric=hamming)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1215   16]
 [ 109    6]]
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      1231
           1       0.27      0.05      0.09       115

    accuracy                           0.91      1346
   macro avg       0.60      0.52      0.52      1346
weighted avg       0.86      0.91      0.88      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9870024370430545
Sensitivity: 0.05217391304347826
AUC: 0.5195881750432663
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn.neighbors import KNeighborsClassifier

classifier2 = KNeighborsClassifier(n_neighbors=4, metric=hamming)
classifier2.fit(X_resampled, y_resampled)
y_pred2 = classifier2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
[[1180   51]
 [ 101   14]]
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      1231
           1       0.22      0.12      0.16       115

    accuracy                           0.89      1346
   macro avg       0.57      0.54      0.55      1346
weighted avg       0.86      0.89      0.87      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred2)
auc = roc_auc_score(y_test, y_pred2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.958570268074736
Sensitivity: 0.12173913043478261
AUC: 0.5401546992547592
```

KNN K=5

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.spatial.distance import hamming


classifier = KNeighborsClassifier(n_neighbors=5,metric=hamming )

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1201   30]
 [ 105   10]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      1231
           1       0.25      0.09      0.13       115

    accuracy                           0.90      1346
   macro avg       0.58      0.53      0.54      1346
weighted avg       0.86      0.90      0.88      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9756295694557271
Sensitivity: 0.08695652173913043
AUC: 0.5312930455974287
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
from scipy.spatial.distance import hamming


X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn.neighbors import KNeighborsClassifier

classifier2 = KNeighborsClassifier(n_neighbors=4,  metric=hamming)
classifier2.fit(X_resampled, y_resampled)
y_pred2 = classifier2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
[[1177   54]
 [ 103   12]]
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      1231
           1       0.18      0.10      0.13       115

    accuracy                           0.88      1346
   macro avg       0.55      0.53      0.54      1346
weighted avg       0.86      0.88      0.87      1346
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_pred2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_pred2)
auc = roc_auc_score(y_test, y_pred2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
Specificity: 0.9561332250203087
Sensitivity: 0.10434782608695652
AUC: 0.5302405255536327
```

Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
modelnb = GaussianNB()
modelnb.fit(x_train, y_train)
y_prednb = modelnb.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_prednb))
print(classification_report(y_test, y_prednb))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_prednb)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_prednb)
auc = roc_auc_score(y_test, y_prednb)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[1231    0]
 [ 115    0]]
              precision    recall  f1-score   support

           0       0.91      1.00      0.96      1231
           1       0.00      0.00      0.00       115

    accuracy                           0.91      1346
   macro avg       0.46      0.50      0.48      1346
weighted avg       0.84      0.91      0.87      1346

Specificity: 1.0
Sensitivity: 0.0
AUC: 0.5
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)
from sklearn.naive_bayes import GaussianNB
modelnb2 = GaussianNB()
modelnb2.fit(X_resampled, y_resampled)
y_prednb2 = modelnb2.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_prednb2))
print(classification_report(y_test, y_prednb2))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, y_prednb2)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, y_prednb2)
auc = roc_auc_score(y_test, y_prednb2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[770 461]
 [ 52  63]]
              precision    recall  f1-score   support

           0       0.94      0.63      0.75      1231
           1       0.12      0.55      0.20       115

    accuracy                           0.62      1346
   macro avg       0.53      0.59      0.47      1346
weighted avg       0.87      0.62      0.70      1346

Specificity: 0.6255077173030057
Sensitivity: 0.5478260869565217
AUC: 0.5866669021297637
```

SVM

In [ ]:
```
from sklearn import svm

modelsvm = svm.SVC(kernel="linear")
modelsvm.fit(x_train, y_train)
predsvm = modelsvm.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix((y_test), predsvm))
print(classification_report(y_test, predsvm))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score
conf = confusion_matrix(y_test, predsvm)
tn, fp, fn, tp = conf.ravel()
specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, predsvm)
auc = roc_auc_score(y_test, predsvm)
print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[1231    0]
 [ 115    0]]
              precision    recall  f1-score   support

           0       0.91      1.00      0.96      1231
           1       0.00      0.00      0.00       115

    accuracy                           0.91      1346
   macro avg       0.46      0.50      0.48      1346
weighted avg       0.84      0.91      0.87      1346

Specificity: 1.0
Sensitivity: 0.0
AUC: 0.5
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn import svm
modelsvm2 = svm.SVC(kernel="linear")
modelsvm2.fit(X_resampled, y_resampled)
predsvm2 = modelsvm2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix((y_test), predsvm2))
print(classification_report((y_test), predsvm2))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score
conf = confusion_matrix(y_test, predsvm2)
tn, fp, fn, tp = conf.ravel()
specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, predsvm2)
auc = roc_auc_score(y_test, predsvm2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[743 488]
 [ 48  67]]
              precision    recall  f1-score   support

           0       0.94      0.60      0.73      1231
           1       0.12      0.58      0.20       115

    accuracy                           0.60      1346
   macro avg       0.53      0.59      0.47      1346
weighted avg       0.87      0.60      0.69      1346

Specificity: 0.6035743298131601
Sensitivity: 0.5826086956521739
AUC: 0.593091512732667
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn import svm
modelsvm2 = svm.SVC(kernel="sigmoid")
modelsvm2.fit(X_resampled, y_resampled)
predsvm2 = modelsvm2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix((y_test), predsvm2))
print(classification_report((y_test), predsvm2))
```

```
[[628 603]
 [ 58  57]]
              precision    recall  f1-score   support

           0       0.92      0.51      0.66      1231
           1       0.09      0.50      0.15       115

    accuracy                           0.51      1346
   macro avg       0.50      0.50      0.40      1346
weighted avg       0.84      0.51      0.61      1346
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn import svm
modelsvm2 = svm.SVC(kernel="rbf")
modelsvm2.fit(X_resampled, y_resampled)
predsvm2 = modelsvm2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix((y_test), predsvm2))
print(classification_report((y_test), predsvm2))
```

```
[[684 547]
 [ 38  77]]
              precision    recall  f1-score   support

           0       0.95      0.56      0.70      1231
           1       0.12      0.67      0.21       115

    accuracy                           0.57      1346
   macro avg       0.54      0.61      0.45      1346
weighted avg       0.88      0.57      0.66      1346
```

In [ ]:
```
from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn import svm
modelsvm2 = svm.SVC(kernel="poly")
modelsvm2.fit(X_resampled, y_resampled)
predsvm2 = modelsvm2.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix((y_test), predsvm2))
print(classification_report((y_test), predsvm2))
```

```
[[729 502]
 [ 46  69]]
              precision    recall  f1-score   support

           0       0.94      0.59      0.73      1231
           1       0.12      0.60      0.20       115

    accuracy                           0.59      1346
   macro avg       0.53      0.60      0.46      1346
weighted avg       0.87      0.59      0.68      1346
```

Decission Tree

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
modeldt = DecisionTreeClassifier(random_state=42)
modeldt.fit(x_train, y_train)
preddt = modeldt.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preddt))
print(classification_report(y_test, preddt))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score

conf = confusion_matrix(y_test, preddt)
tn, fp, fn, tp = conf.ravel()

specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, preddt)
auc = roc_auc_score(y_test, preddt)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[1196   35]
 [ 103   12]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.95      1231
           1       0.26      0.10      0.15       115

    accuracy                           0.90      1346
   macro avg       0.59      0.54      0.55      1346
weighted avg       0.86      0.90      0.88      1346

Specificity: 0.9715678310316815
Sensitivity: 0.10434782608695652
AUC: 0.537957828559319
```

In [ ]:
```
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image

# Visualize tree
dot_data = export_graphviz(modeldt, out_file=None, feature_names=x_train.columns, class_names=
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.92985 to fit
```

Out[163]:

In [ ]:
```
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image

from imblearn.over_sampling import SMOTE
X_resampled, y_resampled = SMOTE().fit_resample(x_train, y_train)

from sklearn.tree import DecisionTreeClassifier
modeldt2 = DecisionTreeClassifier(random_state=0)
modeldt2.fit(X_resampled, y_resampled)
preddt2 = modeldt2.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, preddt2))
print(classification_report(y_test, preddt2))

from sklearn.metrics import confusion_matrix, recall_score, roc_auc_score
conf = confusion_matrix(y_test, preddt2)
tn, fp, fn, tp = conf.ravel()
specificity = tn / (tn + fp)
sensitivity = recall_score(y_test, preddt2)
auc = roc_auc_score(y_test, preddt2)

print("Specificity:", specificity)
print("Sensitivity:", sensitivity)
print("AUC:", auc)
```

```
[[916 315]
 [ 62  53]]
              precision    recall  f1-score   support

           0       0.94      0.74      0.83      1231
           1       0.14      0.46      0.22       115

    accuracy                           0.72      1346
   macro avg       0.54      0.60      0.52      1346
weighted avg       0.87      0.72      0.78      1346

Specificity: 0.7441104792851341
Sensitivity: 0.4608695652173913
AUC: 0.6024900222512627
```

In [ ]:
```
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image

# Visualize tree
dot_data = export_graphviz(modeldt2, out_file=None, feature_names=X_resampled.columns, class_na
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.677004 to fit
```

Out[167]:

MODEL REG LOGISTIC

In [ ]:
```
import statsmodels.api as sm
x_train = sm.add_constant(x_train)
logit_model=sm.Logit(y_train,x_train)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.282403
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Class   No. Observations:                 5384
Model:                          Logit   Df Residuals:                     5375
Method:                           MLE   Df Model:                            8
Date:                Wed, 08 Mar 2023   Pseudo R-squ.:                 0.03239
Time:                        14:28:32   Log-Likelihood:                -1520.5
converged:                       True   LL-Null:                       -1571.3
Covariance Type:            nonrobust   LLR p-value:                 1.849e-18
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.7495      0.332     -8.277      0.000      -3.401      -2.098
Month          0.2443      0.034      7.165      0.000       0.178       0.311
Grade          0.1616      0.042      3.806      0.000       0.078       0.245
SKU           -0.0305      0.011     -2.715      0.007      -0.053      -0.008
SKU Type       0.9286      0.227      4.087      0.000       0.483       1.374
SKU Group      0.1538      0.051      2.994      0.003       0.053       0.254
Coupon Type 1 -0.0515      0.020     -2.528      0.011      -0.091      -0.012
Instlmnt Status -0.1225    0.115     -1.063      0.288      -0.349       0.103
Instalment    -0.7274      0.150     -4.841      0.000      -1.022      -0.433
==============================================================================
```

In [ ]:
```
import statsmodels.api as sm
x_train = sm.add_constant(X_resampled)
logit_model=sm.Logit(y_resampled,X_resampled)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.664731
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Class   No. Observations:                 9848
Model:                          Logit   Df Residuals:                     9840
Method:                           MLE   Df Model:                            7
Date:                Wed, 08 Mar 2023   Pseudo R-squ.:                0.04100
Time:                        14:28:32   Log-Likelihood:                -6546.3
converged:                       True   LL-Null:                       -6826.1
Covariance Type:            nonrobust   LLR p-value:                1.161e-116
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Month          0.2052      0.016     12.971      0.000       0.174       0.236
Grade          0.1321      0.014      9.605      0.000       0.105       0.159
SKU           -0.0158      0.005     -3.071      0.002      -0.026      -0.006
SKU Type       0.9191      0.095      9.702      0.000       0.733       1.105
SKU Group      0.1115      0.024      4.560      0.000       0.064       0.159
Coupon Type 1 -0.0663      0.008     -8.065      0.000      -0.082      -0.050
Instlmnt Status -0.5657    0.050    -11.230      0.000      -0.664      -0.467
Instalment    -0.7218      0.058    -12.442      0.000      -0.836      -0.608
==============================================================================
```