IMPLEMENTASI ARTIFICIAL INTELIGENCE UNTUK MEMPREDIKSI HARGA PENJUALAN RUMAH MENGGUNAKAN METODE RANDOM FOREST DAN FLASK

(Studi kasus: Rohini, India)

TUGAS AKHIR

Putri Choirunisa
16611084

JURUSAN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2020

IMPLEMENTASI ARTIFICIAL INTELIGENCE UNTUK MEMPREDIKSI HARGA PENJUALAN RUMAH MENGGUNAKAN METODE RANDOM FOREST DAN FLASK

(Studi kasus: Rohini, India)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Jurusan Statistika

Putri Choirunisa
16611084

JURUSAN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2020

HALAMAN PESETUJUAN PEMBIMBING

TUGAS AKHIR

Judul : Implementasi *Artificial Inteligence* untuk Memprediksi Harga Penjualan Rumah Menggunakan Metode *Random Forest* dan *Flask* (Studi Kasus: Rohini, India)

Nama Mahasiswa : Putri Choirunisa

Nomor Mahasiswa : 16611084

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, 7 November 2020

Pembimbing

.....

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

.

HALAMAN PENGESAHAN
TUGAS AKHIR

IMPLEMENTASI ARTIFICIAL INTELIGENCE UNTUK MEMPREDIKSI HARGA PENJUALAN RUMAH MENGGUNAKAN METODE RANDOM FOREST DAN FLASK

(Studi Kasus: Rohini, India)

Nama Mahasiswa : Putri Choirunisa

NIM : 16611084

TUGAS AKHIR INI TELAH DI UJIKAN
PADA TANGGAL 7 November 2020

Nama Penguji	**Tanda Tangan**
1. Ayundyah Kesumawati, S.Si., M.Si.	.....
2. Tuti Purwaningsih, S.Stat., M.Si.	.....
3. Arum Handini Primandari, S.Pd.Si., M.Sc.	.....

Mengetahui,
Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

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

KATA PENGANTAR

Assalamualaikum Wr, Wb

Puji Syukur Kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayahnya sehingga tugas akhir yang berjudul "Implementasi *Artificial Inteligence* untuk Memprediksi Harga Penjualan Rumah Menggunakan Metode *Random Forest* dan *Flask*" dapat diselesaikan. Shalawat serta salam semoga selalu tercurah kepada junjungan Nabi Besar Muhammad SAW serta para sahabat dan pengikutnya sampai akhir jaman.

Penulis menyadari bahwa penulisan Tugas Akhir sebagai salah satu persyaratan yang harus dipenuhi dalam menyelesaikan jenjang strata satu di Jurusan Statistika, Universitas Islam Indonesia, ini banyak memperoleh bantuan dari berbagai pihak, baik yang berupa saran, kritik, bimbingan maupun bantuan lainnya. Oleh karena itu pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada:

1. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia
2. Bapak Dr. Edy Widodo, M.Si selaku Ketua Program Studi Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia.
3. Ibu Arum Handini Primandari, S.Pd.Si., M.Sc, selaku Dosen Pembimbing yang sudah membimbingan, memberi arahan, serta motivasi selama penyusunan tugas akhir.
4. Seluruh dosen dan staff karyawan Program Studi Statistika Universitas Islam Indonesia yang selalu memberikan ilmu baik dalam bidang akademik maupun non akademik
5. Kepada kedua orang tua, Bapak Agus Susanto, S.Pd dan Ibu Rosdianti, S.Pd yang selalu memberikan do'a, ridho, membimbing dan memberi semangat kepada penulis.
6. Kepada Adik penulis, Risky Dwi Ilfatmi yang senantiasa memberikan semangat dan do'a

7. Sahabat - sahabat yang selalu menemani penulis selama 8 semester menempuh perkuliahan , Indriyanti Ismayani, Nabila Wuri H, Rizka Yolanda W W ,Adelian Rahma S, dan Meiga Isyatan Mardiyah.
8. Sahabat – sahabat jalan dan juga bercerita tentang berbagai hal Barlinda Titania, Firli Windika dan Indira yang secara tidak langsung membantu penulis dalam menyelesaikan tugas akhir ini.
9. Teman-teman seperbimbingan, Maudi , Adhel dan teman-teman Jurusan Statistika lainnya yang selalu berjalan beriringan dalam menyelesaikan tugas akhir bersama.
10. Serta semua pihak lainnya yang tidak bisa dituliskan penulis satu per satu yang telah membatu selama pembuatan tugas akhir.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, karena keterbatasan ilmu dan pengetahuan penulis semata. Penulis menerima kritik dan saran yang membangun demi penyempurnaan tugas akhir ini agar lebih bermanfaat bagi penulis dan seluruh pihak yang membutuhkan.

Wassalamualaikum Wr. Wb

Yogyakarta, 7 November 2020

Penulis

Putri Choirunisa

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

HALAMAN PERNYATAAN

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

Yogyakarta, 7 November 2020

Putri Choirunisa

IMPLEMENTASI ARTIFICIAL INTELIGENCE UNTUK MEMPREDIKSI HARGA PENJUALAN RUMAH MENGGUNAKAN METODE RANDOM FOREST DAN FLASK

(Studi Kasus: Rohini, India)

Oleh: Putri Choirunisa

Program Studi Statistika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Islam Indonesia

ABSTRAK

Munculnya pandemi *Corona Virus Diseases* 2019 (COVID-19) memberikan efek negatif di segala aspek. Tak terkecuali bisnis properti, ditambah adanya kebijakan *lockdown* mengakibatkan pengurangan aktifitas diluar rumah. Akibatnya daya jual beli rumah mengalami penurunan. Penurunan penjualan rumah bisa menjadi salah satu faktor lambatnya pertumbuhan ekonomi di suatu negara. Kurangnya media informasi ditambah kebijakan larangan keluar rumah menyulitkan penjual dan pembeli dalam mendapatkan informasi harga rumah. Berdasarkan masalah tersebut maka dilakukan sebuah penelitian menggunakan metode *Random Forest* untuk memprediksi harga rumah untuk mempermudah penjual dan pembeli dalam bertransaksi. Metode *random forest* membangun suatu model dengan menggunakan beberapa pohon keputusan (*decision tree*) secara acak dan menggabungkan prediksi setiap pohon untuk mendapatkan hasil prediksi. Data yang digunakan dalam penelitian ini yaitu data sekunder yang bersumber dari website *magicbricks*. Hasil penelitian ini didapatkan pohon terbaik sebanyak 340 pohon, Faktor terbesar yang mempengaruhi prediksi harga rumah yaitu *Area*. Dengan hasil evaluasi model yang diperoleh untuk memprediksi harga rumah memiliki nilai *Mean Absolute Percent Error (MAPE)* sebesar 26.48% dan nilai *R-squared* sebesar 0.9423. Setelah mendapatkan model prediksi dilakukan perancangan *web* aplikasi menggunakan *flask* dan Heroku yang dapat digunakan untuk memprediksi harga rumah dengan kriteria yang diinginkan, sehingga calon penjual atau pembeli rumah bisa mendapatkan informasi dengan harga rumah yang mereka inginkan secara cepat dan mudah.

***Kata kunci:**, Harga Prediksi Rumah, Random Forest Regression, Features Importance, Web Aplikasi, Flask.*

A IMPLEMENTATION OF ARTIFICIAL INTELLIGENCE TO PREDICTING SALES PRICES OF THE HOUSES USING RANDOM FOREST AND FLASK

(Case Study: Rohini, India)

Author: Putri Choirunisa

Department of Statistics

Faculty of Mathematics and Natural Sciences

Islamic University of Indonesia

ABSTRACT

The advent of the corona virus diseases 2019 pandemic (covid-19) brings negative effects in all aspects. Property business is no exception, coupled with lockdown policy, resulting in a reduction in off-house activity. As a result, the house sales are down. The decline in home sales may be a factor in the slowness of economic growth in a country. The lack of information media plus the no-leave policy makes it difficult for sellers and buyers to obtain home price information. According to the problem, a study has been conducted using a random forest method to predict housing prices to make it easier for sellers and buyers to make transactions. Random forest's method builds a model using a few random decision trees and combines the predictions of each tree to get a predictable outcome. The data used in the research is a secondary data sourced from the magicbricks website. This study has yielded the finest trees of 340 and the biggest factor that affected the predictive value of the house was the area. the results of the model evaluation obtained to predict home prices have a value of mean absolute error (mape) 26.48% dan *R-squared value* 0.9423. After getting a predictive model done web designing applications using flask or drugs that could be used to predict home prices by desired criteria, so that prospective buyers or homeowners can get information at home prices quickly and easily

Keyword: *House prediction price, Random Forest, Features Importance, Web Application, Flask.*

BAB I
PENDAHULUAN

1.1. Latar Belakang

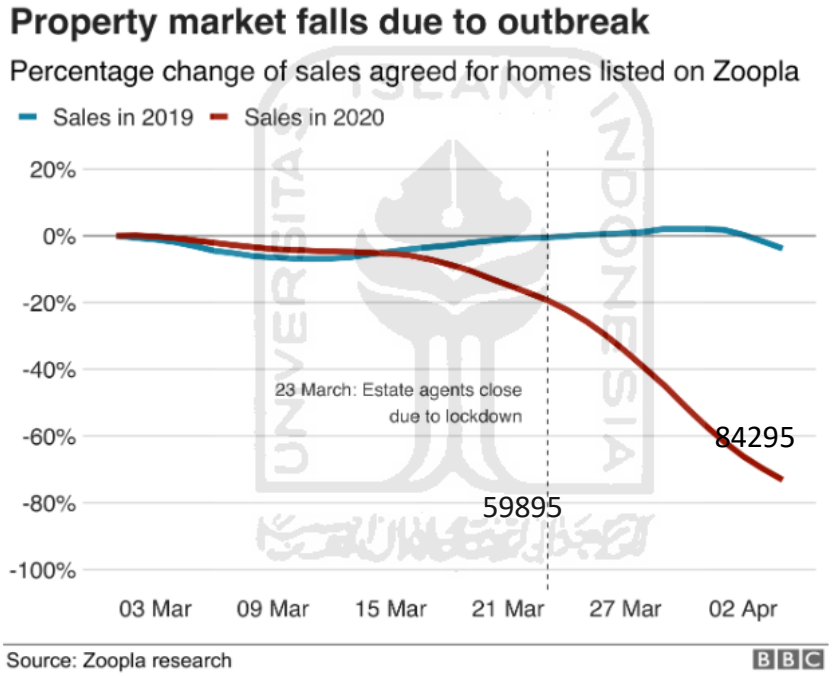
Pada era globalisasi saat ini perkembangan teknologi informasi dan komunikasi berkembang semakin pesat. Revolusi industry 4.0 sudah tidak asing lagi bagi masyarakat. Revolusi industry 4.0 merupakan sebuah revolusi industri dimana teknogi dan komunikasi dimanfaatkan sepenuhnya guna mencapai efisiensi setinggi – tingginya sehingga menghasilkan model bisnis baru berbasis digital. Revolusi industri 4.0 pada era digital ini juga menjanjikan keuntungan jangka panjang berupa efisiensi dan produktivitas Perkembangan ini kemudian menimbulkan disrupsi bagi dunia ekonomi, khususnya dibidang bisnis.

Paradigma bisnis pun bergeser dari penekanan *owning* menjadi *sharing*. Contoh nyata dapat dilihat pada perpindahan bisnis retail (toko fisik) kedalam *e-commerce* yang menawarkan kemudahan dalam berbelanja (Prasetyo & Trisyanti, 2018). Tidak hanya bisnis berbentuk barang saja yang mengalami disrupsi bisnis jasa pun mengalami perubahan seperti penjualan properti rumah , pembelian tiket dan penyewaan kamar hotel.

Salah satu bisnis dibidang jasa yang mengalami disrupsi yaitu penjualan properti rumah dan apartemen. Rumah merupakan kebutuhan primer bagi manusia. Setiap orang tentu punya rumah ideal yang mereka impikan. Apalagi dengan semakin lengkapnya fasilitas yang kini banyak ditawarkan oleh *developer* properti, semakin banyak juga alternatif rumah ideal yang bisa dipilih. Belum lagi jika mengingat harga rumah yang akan terus naik setiap tahunnya, itu membuat jual beli rumah menjadi salah satu bentuk investasi yang menguntungkan.

Munculnya pandemi *Corona Virus Diseases* 2019 (*COVID 19*) yang di laporkan oleh *World Health Organization (WHO*) di awal tahun 2020- hingga 13 September 2020 telah memiliki 28.637.952 kasus terinfeksi, 917.417 kasus kematian,

dan mewabah di 216 negara. Properti merupakan salah satu sektor yang memiliki harga jual beli yang sangat tinggi. Selain bisa digunakan sendiri, tak sedikit orang juga membeli properti untuk dijadikan sebagai investasi masa depan karena bisa mendapatkan keuntungan yang besar. Namun semenjak virus *COVID-19* mewabah hampir diseluruh dunia, bisnis properti terkena dampak. Daya jual beli masyarakat terhadap properti seperti apartmen dan rumah yang digunakan untuk hunian atau investasi mengalami penurunan. Terlebih lagi adanya sistem *lockdown* yang diterapkan beberapa negara untuk mengurangi penyebaran penyakit ini.



Gambar 1. 1 Data Penjualan Rumah di *Zoopla*

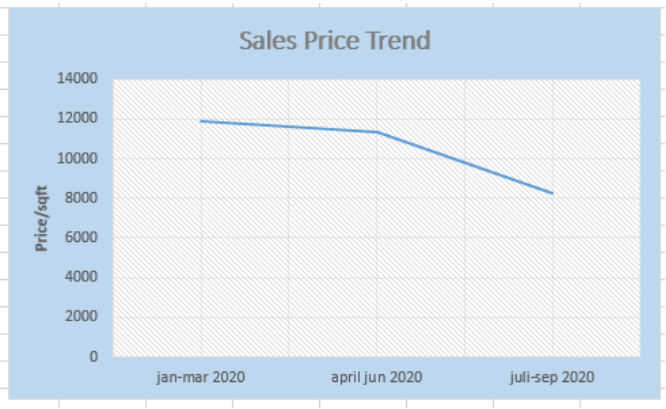
Berdasarkan Gambar 1.1 di UK dilaporkan jumlah orang yang ingin mengetahui tentang properti atau melihatnya mulai menurun pada awal Maret, dan telah turun lebih dari 60%. Selain itu dibeberapa Negara mengalami juga penurunan penjualan rumah di UK menurut laporan data *HM Revenue and Customs* (HMRC) turun sebanyak 56%. Negara jepang menurut data *Japan Properti Central* turun sebanyak 55.3% pada bulan April 2020 dan 52.6% dari tahun lalu. Menurut laporann

National Association of Realtors, US mengalami penurunan penjulan sebesar 17.8% di bulan Maret.

Name	Cases - cumulative total	Cases - newly reported in last 24 hours	Deaths - cumulative total	Deaths - newly reported in last 24 hours
Global	**28,637,952**	**307,930**	**917,417**	**5,537**
United States o…	6,386,832	45,523	191,809	1,022
India	4,754,356	94,372	78,586	1,114
Brazil	4,282,164	43,718	130,396	874
Russian Feder…	1,062,811	5,449	18,578	94
Peru	716,670	6,603	30,470	126

Gambar 1. 2 Data Kasus *COVID-19*

Berdasarkan Gambar 1.2 Negara India memiliki kasus terinfeksi COVID -19 terbanyak dengan urutan kedua dari seluruh dunia. Menurut konsultan properti *Knight Frank* selama kebijakan lockdown berlangsung sebanyak 8 kota di india mengalami penurunan penjualan rumah sebesar 54% dari periode Januari – Juni. Salah satu kota yang mengalami penurunan penjualan rumah yaitu Kota Rohini, India. Rohini merupakan salah satu kota padat pemukiman di India yang juga merupakan kota pertama yang di bangun berdasarkan project *Delhi Development Authority* (DDA) yang berfokus pada pemenuhan fasilitas umum. Berdasarkan Gambar 1.3 yang bersumber dari *website magickbricks* di laporkan bahwa adanya penurunan harga penjualan rumah di Rohini, India. Penuruan harga penjualan rumah terus terjadi selama pandemi COVID 19. Penurunan penjualan rumah bisa menjadi salah satu faktor lambatnya pertumbuhan ekonomi di suatu negara.



Gambar 1. 3 *Sales Price Trend*

Salah satu cara untuk meningkatkan penjualan rumah di tengah pandemi COVID-19 yaitu melalui *platform online* jual beli rumah. Perubahan transaksi jual beli terjadi di bidang properti. Jika dulu transaksi jual beli menggunakan bisnis tradisional yang mengharuskan penjual dan pembeli bertemu untuk mengetahui harga, model properti dan lain-lain. Namun pada masa sekarang bermunculan *website-website* jual beli properti seperti *Magicbricks, Trulia, Zillow , Redfin* dan lain-lain yang memudahkan jual-beli rumah. Salah satu website properti di India yaitu *Magicbricks*. *Magicbricks* adalah situs *web* yang menyediakan *platform* umum bagi pembeli & penjual properti untuk menemukan properti yang diminati di India, dan sumber informasi tentang semua masalah terkait properti. Diluncurkan pada tahun 2006, *Magicbricks* berhasil menjadi situs web properti *online* No. 1 yang paling disukai di India oleh berbagai survei independen. Situs *web* ini saat ini memiliki lebih dari 7000+ properti untuk dijual dan juga menampilkan sejumlah proyek baru. *Magicbricks* bertujuan untuk memberikan semua penggunanya informasi yang transparan tentang properti yang diminati hanya dengan beberapa klik. Situs *web* ini memungkinkan pengguna untuk memposting tentang properti secara gratis dan untuk mencari *real estate* di seluruh negeri. Penjualan rumah secara *online* merupakan cara yang paling efektif di tengah larangan untuk pergi keluar rumah.

Kurangnya media informasi ditambah kebijakan larangan keluar rumah menyulitkan penjual dan pembeli dalam mendapatkan informasi harga rumah.

Sehingga di perlukan model untuk memprediksi harga rumah untuk mempermudah penjual dan pembeli dalam bertransaksi. Bagi calon pembeli sangat konservatif mengenai *budget* dan tipe rumah yang diinginkan . Sehingga dibutuhkan model prediksi yang bisa digunakan untuk mengetahui kisaran harga rumah yang diinginkan pembeli sesuai kriteria tipe rumah yang mereka inginkan (Bhagat et al., 2016). Sedangkan bagi penjual model prediksi bisa membantu penjual rumah yang kesulitan untuk menentukan harga rumah dengan harga yang berada di pasaran

Salah satu metode yang bisa digunakan dalam memprediksi harga rumah yaitu metode *random forest*. Metode *random forest* merupakan bagian dari metode data mining dengan metode dasarnya pohon keputusan (*decision tree*). *Random forest* membangun suatu model dengan menggunakan beberapa pohon keputusan (*decision tree*) secara acak dan menggabungkan prediksi setiap pohon untuk mendapatkan hasil prediksi (Breiman, 2001).

Bagi seorang data analis atau data *scientist* penting untuk menyajikan dan menginterpretasikan model dengan mudah serta bisa dimanfaatkan langsung oleh masyarakat awam atau klien. Pengembangan sebuah perangkat lunak tidak terlepas dari peran seorang analis. Salah satu tugas analis adalah memodelkan masalah dalam tahapan analisis. Dalam memodelkan, analis biasanya menggunakan alat bantu berupa aplikasi pemodelan kebutuhan perangkat lunak. Aplikasi pemodelan yang tersedia di pasaran relatif komplek dan mahal. Hal tersebut menyebabkan analis mengalami masalah dalam memodelkan kebutuhan, khususnya untuk seorang analis pemula. Analis pemula biasanya kesulitan menerapkan pengetahuan mereka apabila aplikasi yang digunakan terlalu komplek dan sulit dipahami (Rosi Subhiyakto & Wahyu Utomo, 2017). *Flask* adalah sebuah alat yang membantu membuat kerangka untuk sebuah *web* dan dengan menggunakan *Flask*, pengembang pemula pun dapat menciptakan sebuah web yang bagus. *Flask* adalah sebuah *web framework* yang ditulis dengan bahasa *Python* dan tergolong sebagai jenis *microframework*.

Pada penelitian (Taylor et al., 2019) dengan judul "*A Model to Detect Heart Disease using Machine Learning Algorithm*" bertujuan untuk mendeteksi penyakit

jantung menggunakan algoritma pembelajaran mesin. Penelitian ini mendapatkan hasil akurasi sebesar 98.83% menggunakan metode klasifikasi *decision tree* dan menggunakan *python* dan *flask* untuk *web deployment*.

Berdasarkan penelitian tersebut model prediksi rumah bisa dijadikan model *prototype* untuk menyampaikan hasil analisis dengan mudah dan bisa dimanfaatkan langsung bagi pembeli dan penjual rumah. Variabel dependen yang akan digunakan dalam penelitian adalah *price*, sedangkan variabel independen antara lain, *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type.*

Dengan demikian, penelitian ini dilakukan dengan judul "**Implementasi *Artificial Inteligence* untuk Memprediksi Harga Penjualan Rumah Menggunakan Metode *Random Forest* dan *Flask***".

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimana hasil prediksi harga rumah dengan menggunakan metode *Random Forest Regression*?
2. Bagaimana membangun website untuk memprediksi harga rumah?

1.3. Batasan Masalah

Batasan masalah yang dapat diambil dari penelitian ini sebagai berikut:

1. Daerah yang digunakan dalam memprediksi harga rumah yaitu Rohini, India
2. Variabel yang digunakan antara lain *Price,Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type*
3. Analisis prediksi dilakukan menggunakan metode *Random Forest Regressor* dengan *Python* 3.6.4.
4. Melakukan *web deployment* untuk model *prototype* menggunakan *flask* dan *Heroku*.

1.4. Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah diuraikan, maka tujuan dilakukannya penelitian ini adalah:

1. Mengetahui hasil prediksi harga rumah menggunakan metode *Random Forest Regression*.
2. Mengetahui tahapan dalam membangun *website* untuk memprediksi harga rumah.

1.5. Manfaat Penelitian

Manfaat yang dapat diambil dari penelitian ini sebagai berikut:

1. Memberikan gambaran bagi penjual dan pembeli terkait faktor-faktor yang perlu diperhatikan dan dapat mempengaruhi harga rumah.
2. Memberikan informasi harga kisaran bagi penjual dan pembeli dalam melakukan transaksi jual beli rumah
3. Menambah dan memperluas ilmu pengetahuan terkait penerapan metode *Random Forest* dan pembuatan sebuah *prototype* menggunakan *Flask* serta pengaplikasianmya kedalam *website*..

BAB II
TINJAUAN PUSTAKA

Terkait dengan penelitian yang dilakukan penulis, maka penelitian terdahulu menjadi sangat penting agar dapat diketahui hubungan antara penelitian yang dilakukan sebelumnya dengan penelitian yang dilakukan pada saat ini, dan terjadinya suatu penjiplakan atau duplikasi dalam penelitian yang dilakukan tersebut mempunyai arti penting sehingga dapat diketahui kontribusi penelitian ini terhadap perkembangan ilmu pengetahuan.

Terdapat beberapa penelitian sebelumnya yang menggunakan metode *Random Forest Regression* untuk memprediksi harga. Berikut beberapa penelitian dan jurnal yang menjadi acuan peneliti dalam melakukan penelitian. Penelitian pertama dilakuan oleh (Mohd et al., 2019) yang berjudul "*Machine Learning Housing Price Prediction in Petaling Jaya, Selangor, Malaysia*" bertujuan untuk mengidentifikasi variabel apa yang memiliki kontribusi besar terhadap akurasi prediksi ketika diuji dengan metode *machine learning* yang dipilih serta melakukan verifikasi terhadap tingkat signifikansi dari setiap variabel yang mempengaruhi kinerja algoritma *machine learning*. Hasil penelitian ini menunjukkan akurasi tertinggi ditujukan pada metode *Random Forest Regressor* dan penggunaan variabel yang tidak relevan akan mempengaruhi nilai akurasi .

Penelitian kedua dilakukan oleh (Ma et al., 2018) yang berjudul "*Estimating Warehouse Rental Price using Machine Learning Techniques*" bertujuan untuk memprediksi harga penyewaan gudang di pasaran. Untuk mendapatkan model terbaik penelitan ini menggunakan empat metode yaitu Regresi Linear, *Random Forest Regressor, Regression Tree*, dan *Gradient Boosting Regression Trees*. Hasil penelitian ini menunjukan bahwa metode *Random Forest Regressor* memiliki akurasi tertinggi serta menunjukan hasil bahwa variabel jarak dari pusat kota memiliki pengaruh besar dengan nilai akurasi prediksi harga sewa gudang.

Penelitian ketiga dilakukan oleh (Hong et al., 2020) yang berjudul "*A House Price Valuation Based On The Random Forest Approach: The Mass Appraisal Of Residential Property In South Korea*". Penelitian ini bertujuan un tuk menyelidiki fitur prediktor harga rumah berdasarkan metode *Random Forest* (RF) dengan membandingkannya dengan model penetapan harga hedonis dengan metode konvensional yaitu *Ordinary Least Square* (OLS) dengan menggunakan dataset transaksi apartemen dari periode 2006 hingga 2017 di distrik Gangnam, salah satu daerah paling berkembang di Korea Selatan. Dengan menggunakan sampel 40% dari semua transaksi, menunjukkan bahwa keakuratan prediktor berbasis *machine learning* yaitu *random forest* memiliki akurasi yang lebih tinggi disbanding metode *Ordinary Least Square* (OLS). Nilai MAPE yang di tunjukkan pada metode OLS yaitu sebesar 20% dan metode *Random Forest* hanya 5.5%

Penelitian keempat dilakukan oleh (Čeh et al., 2018) yang berjudul "*Estimating the Performance of Random Forest versus Multiple Regression for Predicting Prices of the Apartments*". Penelitian ini bertujuan untuk menganalisis kinerja prediksi metode *machine learning random forest* dengan metode regresi berganda dalam memprediksi harga apartemen. Dataset yang digunakan yaitu data transaksi apartemen penjualan *real estate* di kota Ljubljana, Slovenia Tahun 2008-201. Dari penelitian tersebut hasil dari nilai *Rsquare*, rasio penjualan, rata-rata persentase kesalahan (MAPE), koefisien dispersi (COD)) mengungkapkan metode random forest memiliki hasil yang jauh lebih baik.

Penelitian kelima dilakukan oleh (Borde et al., 2017) yang berjudul "*Real Estate Investment Advising Using Machine Learning*" bertujuan untuk membandingkan beberapa metode yaitu metode *gradient descent, K-nearest neighbor regression* dan *random forest regression* dalam memprediksi harga *real estate*. Hasil penelitian menunjukan dari perhitungan *error* MAPE, RMSE dan MAE metode *random forest* memiliki nilai error terkecil.

Penelitian keenam dilakukan oleh (Park et al., 2019) yang berjudul "*A Study on Apartment Price Prediction Model Using Machine Learning : An Example from Busan*

Metropolitan Area" bertujuan untuk memprediksi harga apartemen dengan menggunakan dataset transaksi apartemen di kota Busan tahun 2013-2015. Metode yang digunakan pada penelitian ini yaitu *extreme gradient boost (XGboost*) , *random forest* , dan *support vector machine*. Hasil perbandingan ketiga metode tersebut metode *machine learning random forest* menunjukkan hasil kesalahan prediksi terendah.

Mengacu pada penelitian sebelumnya terdapat beberapa kesamaan yaitu meneliti harga prediksi *real estate* dan menggunakan metode *random forest*. Hal yang membedakan penelitian ini dengan penelitian sebelumnya yaitu sampel yang digunakan dan tahun penelitian serta penambahan *website app* menggunakan *flask* yang di publikasikan sehingga hasil dari penelitian ini dapat digunakan oleh masyarakat awam.

BAB III
LANDASAN TEORI

3.1. Rumah

Rumah merupakan kebutuhan primer bagi manusia yang berfungsi sebagai tempat perlindungan, tempat tinggal, dan berkumpul bersama. Rumah merupakan suatu bangunan, tempat manusia tinggal dan melangsungkan kehidupannya (Budihardjo, 1998). Di samping itu, rumah juga merupakan tempat berlangsungnya proses sosialisasi pada saat seorang individu diperkenalkan kepada norma dan adat kebiasaan yang berlaku di dalam suatu masyarakat. Perumahan memiliki beberapa jenis yaitu rumah tinggal tunggal, rumah tinggal Koppel, rumah kota (*town house*),rumah berpekarangan dalam (*patio house*), maisonet, rumah teras bertingkat (*terrace house),* rumah gandeng (*row houses*), dan rumah susun (apartemen) (Unterman & Small, 1983).

3.2. Apartemen

3.2.1. Pengertian Apartemen

1. Menurut KBBI, Apartemen adalah tempat tinggal yang terdiri atas ruang duduk, kamar tidur, kamar mandi, dapur, dan sebagainya yang berada pada satu lantai bangunan bertingkat yang besar dan mewah, dilengkapi dengan berbagai fasilitas (kolam renang, pusat kebugaran, toko, dan sebagainya).
2. Menurut Undang – Undang Republik Indonesia No 16 Tahun 1985 Apartemen (Rumah Susun) adalah bangunan gedung bertingkat yang dibangun dalam suatu lingkungan, yang terbagi dalam bagian-bagian yang distrukturkan secara fungsional dalam arah horizontal maupun vertikal dan merupakan satuan-satuan yang masing - masing dapat dimiliki dan digunakan secara terpisah, terutama untuk tempat hunian, yang dilengkapi dengan bagian-bersama, benda-bersama dan tanah-bersama

3. Apartemen adalah bangunan yang memuat beberapa grup hunian, yang berupa rumah flat atau rumah petak bertingkat yang diwujudkan untuk mengatasi masalah perumahan akibat kepadatan tingkat hunian dan keterbatasan lahan dengan harga yang terjangkau di perkotaan (Marlina, 2008)

3.2.2 Jenis – Jenis Apartemen

Apartemen memiliki beberapa jenis yaitu:

1. Berdasarkan pelayananan aparatmen terdiri dari beberapa jenis:
   a. Apartemen *Fully Service*

      Apartemen yang menyediakan layanan standar hotel bagi penghuninya, seperti laundry, catering, kebersihan, dan sebagainya.
   b. Apartemen *Fully Furnished*

      Apartemen yang menyediakan furniture dalam unit apartemen.
   c. Apartemen *Fully Furnished* dan *Fully Service*

      Apartemen jenis ini lebih lengkap dan lebih mahal karna gabungan dari apartemen *Fully Furnished* and *Fully Service*
   d. Apartemen *Building Only*

      Apartemen yang hanya menyediakan ruangannya saja.
2. Berdasarkan kategori jenis dan besar bangunan, apartemen terdiri dari:

a. High-Rise Apartment

Bangunan apartemen yang terdiri lebih dari sepuluh lantai. Dilengkapi area parkir bawah tanah, *system* keamanan dan servis penuh. Struktur apartemen lebih kompleks sehingga desain unit apartemen cenderung standard. Jenis ini banyak di bangun di pusat kota

b. Mid-Rise Apartment

Bangunan apartemen yang terdiri dari tujuh sampai dengan sepuluh lantai. Jenis apartemen ini lebih sering dibangun di kota .

c. Low-Rise Apartment

Apartemen dengan ketinggian kurang dari tujuh lantai dan menggunakn tangga sebagai alat transportasi vertikal. Biasanya untuk golongan menengah kebawah.

d. Walked-up Apartment

Bangunan apartemen yang terdiri atas tiga sampai dengan enam lantai. Apartemen ini kadang - kadang memiliki *lift*, tetapi dapat juga tidak menggunakan. Jenis apartemen ini disukai oleh keluarga yang lebih besar (keluarga inti ditambah orang tua). Gedung apartemen ini hanya terdiri atas dua atau tiga unit apartemen (Akmal, 2007).

3.3. Magicbricks

Magicbricks merupakan situs *web* yang menyediakan *platform* umum bagi penjual dan pembeli untuk mendapatkan sumber informasi mengenai property di India. Diluncurkan pada tahun 2006 oleh *Times Group, Magicbricks* dengan cepat menjadi Portal Properti No. 1 di India.

*Magicbrick*s menawarkan daftar properti residensial dan komersial untuk dijual dan disewakan di seluruh negeri. Desain *Magicbricks* didasarkan pada penelitian yang ketat, pengembangan produk yang unik, dan inisiatif inovatif yang telah diterima oleh pengguna. Dalam upaya untuk memberikan layanan terbaik kepada pengguna, *Magicbricks* secara konsisten terus melakukan inovasi dan evaluasi.

Hingga saat ini, *magicbricks* telah hadir di 75 daerah *domestic* di India dan lebih dari 15 Negara

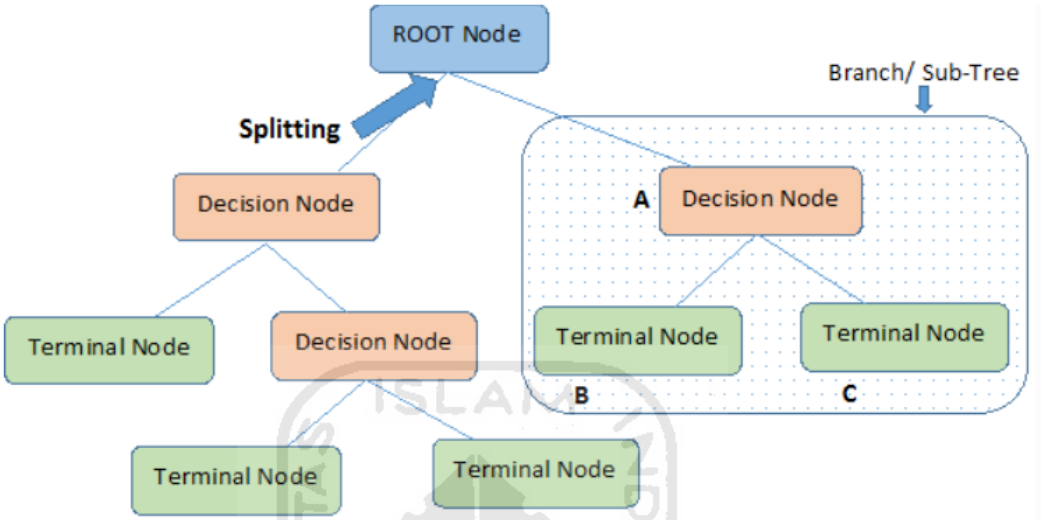
Sumber : magicbricks.com

Gambar 3. 1 Logo Perusahan Magicbricks

3.4. Pohon Keputusan *(Decision Tree)*

Pohon keputusan (*Decision Tree*) adalah salah satu model prediksi menggunakan struktur pohon atau struktur berhirarki, di mana masing-masing internal *node* menunjukkan tes pada atribut, setiap cabang (*branch*) menunjukkan hasil tes, dan

setiap simpul daun (*leaf node*) mempresentasikan label kelas. Bagian paling atas dari pohon keputusan adalah akar pohon (*root node*) (Wu et al., 2008).



Sumber : gdcoder.com

Gambar 3. 2 *Decision Tree*

Berdasarkan gambar 3.2, Simpul akar (*root node*) mewakili seluruh populasi atau sampel yang dibagi menjadi dua atau lebih. *Splititing* adalah proses membagi 2 simpul *(node*) atau lebih. Simpul keputusan *(decision node*) adalah ketika sebuah *sub-node* terpecah menjadi sub-node lebih lanjut. Cabang *(branch)* adalah Sebuah sub bagian dari seluruh pohon.

Decision Tree adalah sebuah diagram alir yang berbentuk seperti struktur pohon yang mana setiap *internal node* menyatakan pengujian terhadap suatu atribut, setiap cabang menyatakan *output* dari pegujian tersebut dan *leaf node* menyatakan kelas-kelas/ distribusi kelas. *Node* akar (*Root node)* memiliki beberapa *edge* keluar tetapi tidak memiliki *edge* masuk. *Internal node* akan memiliki satu *edge* masuk dan beberapa *edge* keluar, sedangkan *leaf node* hanya akan memiliki satu *edge* masuk tanpa memiliki *edge* keluar. *Leaf node* adalah hasil akhir yang mewakili label kelas dari kombinasi atribut yang terbentuk menjadi *rule* (Kasih, 2019).

Pohon keputusan *(decision tree*) memiliki dua jenis kegunaan didalam data mining yaitu klasifikasi dan regresi. Pohon keputusan membangun model regresi atau klasifikasi dalam bentuk struktur pohon. Ini memecah kumpulan data menjadi himpunan bagian yang lebih kecil dan lebih kecil sementara pada saat yang sama pohon keputusan terkait dikembangkan secara bertahap. Pohon klasifikasi (*classification tree)* digunakan ketika variabel target berupa data kategorik. Pohon regresi *(regression tree*) digunakan ketika variabel target berupa data numerik.

Algoritama untuk membangun pohon keputusan salah satunya metode ID3 yang dicetuskan oleh J. R. Quinlan yang menggunakan pencarian dari *top-down,* yang setiap cabang menggunakan algoritma *greedy search*. Algoritma *greedy search* merupakan algoritma optimasi yang memberikan solusi pada setiap langkah. Pada pohon keputusan (*decision tree*) optimasi dalam bentuk benar *(true*) atau salah *(false)* . Algoritma ID3 dapat digunakan untuk membangun pohon keputusan untuk regresi dengan mengganti *Information Gain* dengan *Standard Deviation Reduction.*

$$SDR\ (T,X) = S(T) - S(T,X) \qquad (3.1)$$

Pohon keputusan dibangun dari atas ke bawah dari simpul akar dan melibatkan partisi data menjadi subset yang berisi *instance* dengan nilai serupa (homogen) dengan menggunakan deviasi standar untuk menghitung homogenitas sampel numerik.

3.5. *Random Forest*

Classification and regression tree (CART) adalah istilah yang digunakan untuk menggambarkan algoritma pohon keputusan (*decision tree)* yang digunakan untuk klasifikasi dan regresi dalam *machine learning*. Metode CART diperkenalkan oleh Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone pada tahun 1984. Pohon yang digunakan untuk regresi dan pohon yang digunakan untuk klasifikasi memiliki beberapa kesamaan - tetapi juga memiliki beberapa perbedaan, seperti prosedur yang digunakan untuk membagi cabang. (Breiman et al., 1984).

CART akan menghasilkan pohon klasifikasi jika variabel respon mempunyai skala kategorik dan akan menghasilkan pohon regresi jika variabel respon berupa

data kontinu .Tujuan utama CART adalah untuk mendapatkan suatu kelompok data yang akurat sebagai penciri dari suatu pengklasifikasian (Sumartini & Purnami, 2015). Kelebihan metode CART yaitu interpretasi hasil dari pohon mudah untuk dijelaskan , perhitungan yang cepat, dan CART secara implisit melakukan seleksi fitur. Namun, CART memiliki kekurangan yaitu pohon yang dihasilkan *overfitting* sehingga hasil prediksi model tidak akurat, varians kecil dalam data dapat menyebabkan varians yang sangat tinggi dalam prediksi, sehingga mempengaruhi stabilitas hasil CART mungkin tidak stabil dalam *decision trees* (pohon keputusan) karena CART sangat sensitif dengan data baru. CART sangat bergantung dengan jumlah sampel. Jika sampel data *learning* dan *testing* berubah maka pohon keputusan yang dihasilkan juga ikut berubah. Tiap pemilihan bergantung pada nilai yang hanya berasal dari satu variabel penjelas (Pratiwi et al., 2014).

Berdasarkan hal itu metode baru untuk mengatasi kekurangan CART yaitu *ensemble tree*. *Ensemble tree* memanfaatkan beberapa pohon, bukan hanya satu pohon, untuk melakukan dugaan. Dengan kata lain dugaan dari suatu data tertentu, merupakan penggabungan dari dugaan-dugaan yang dihasilkan oleh beberapa pohon. Pada proses analisis, teknik ini menghasilkan beberapa pohon yang selanjutnya digunakan secara simultan untuk melakukan pendugaan (Sartono & Syafitri, 2010).

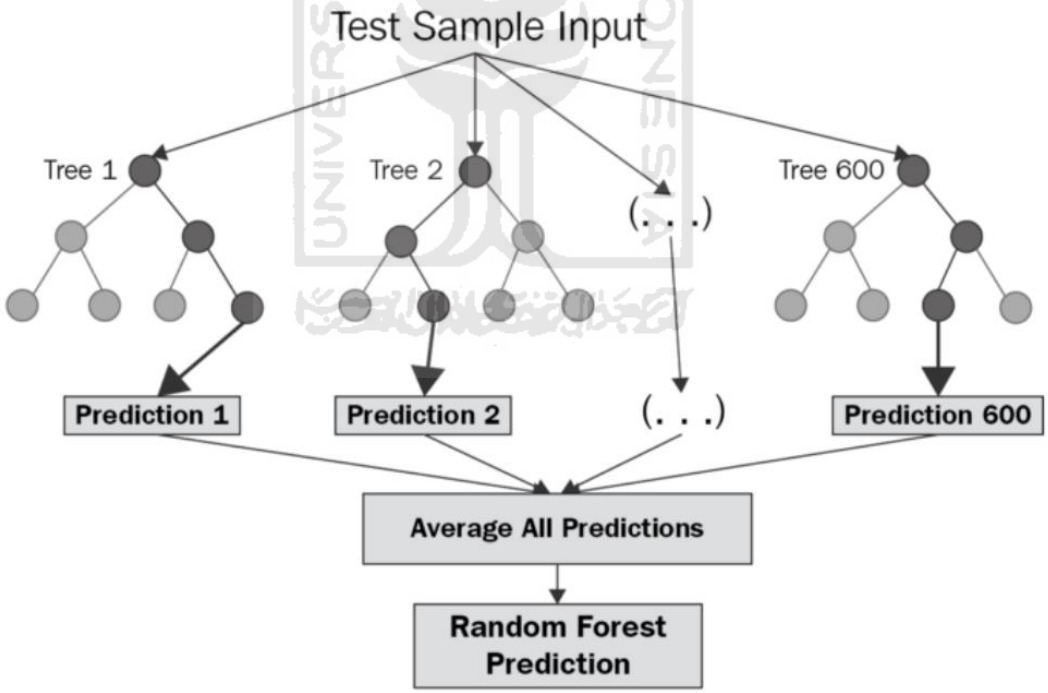
Salah satu *ensemble learning* yang paling popular yaitu *bagging*. *Bagging* merupakan singkatan dari *bootstrap aggregating*. Berdasarkan namanya, maka dapat diperkirakan ada dua tahapan utama dalam analisis ini, yaitu *bootstrap* yang tidak lain adalah pengambilan sampel dari data sampling yang dimiliki (resampling) dan *aggregating* yaitu menggabungkan banyak nilai dugaan menjadi satu nilai dugaan. Penggunaan *bagging* ini sangat membantu terutama mengatasi sifat ketidakstabilan pohon klasifikasi dan regresi tunggal seperti yang telah disinggung sebelumnya.

Metode klasifikasi yang menggunakan algoritma *ensemble tree bagging* yaitu *random forest*. *Random forest* di usulkan pertama kali oleh Tin Kam Ho dari Bell Labs pada tahun 1995 dan di kembangkan lagi oleh breiman pada tahun 2001. Kata "*random"* pada *random forest* memiliki dua arti yaitu:

1. Pengambilan sampel acak dari data observasi saat membangun pohon.
2. Setiap subset memiliki feature yang berbeda (*random feature selection)* untuk memisahkan node

Dengan kata lain, *Random forest* membangun suatu model dengan menggunakan beberapa pohon keputusan (*decision tree)* dan menggabungkan prediksinya untuk mendapatkan prediksi yang lebih akurat dan stabil daripada mengandalkan satu pohon keputusan *decision tree.* Perhitungan prediksi menggunakan voting jika ingin melakukan klasifikasi dan menghitung rata-rata untuk prediksi regresi.

Metode *random forest* berupaya untuk memperbaiki proses pendugaan yang dilakukan menggunakan metode *bagging.* Perbedaan utama dari metode ini terletak pada penambahan tahapan random sub-setting sebelum di setiap kali pembentukan pohon (Breiman, 2001).



Sumber : towardsdatascience.com

Gambar 3. 3 *Random Forest*

Berdasarkan gambar 3. Langkah- langkah *random forest* sebagai berikut :

1. Tahap awal membentuk subset:

a. Mengambil sampel secara acak dengan pengembalian berukuran n dari gugus data training
b. Membuat subset acak dengan menyusun pohon berdasarkan data tersebut,
c. ulangi langkah a-b sebanyak k kali sehingga diperoleh k buah pohon acak;

2. Setelah pembentukan subset melakukan proses splitting dengan menggunakan algoritma terbaik untuk pemisahan node

   Melakukan prediksi dengan menggabungkan semua hasil prediksi K pohon. Kasus klasifikasi menggunakan *majority vote* dan kasus regresi menggunakan rata-rata.

Pada tabel 3. Diberikan contoh data *random forest* kasus regresi dengan kasus apakah variabel *windy, depth, temperature, outlook, humidity, waves* mempengaruhi hasil tangkapan ikan (*number of fish)*

Tabel 3.1 Contoh Kasus Regresi

No	Windy	Depth	Temperature	Outlook	Humidity	Waves	Number of fish
1.	False	30	Hot	Rainy	High	Yes	25
2	True	40	Hot	Rainy	High	Yes	30
3	False	23	Hot	Overcast	High	No	46
4	True	39	Cool	Rainy	Normal	No	50
5	False	45	Mild	Sunny	High	No	45
6	False	21	Cool	Sunny	Normal	No	52
7	True	54	Cool	Sunny	Normal	Yes	23
8	True	12	Cool	Overcast	Normal	No	43
9	False	36	Mild	Rainy	High	Yes	35
10	True	32	Hot	Overcast	High	No	49
11	False	32	Cool	Rainy	Normal	No	38
12	False	35	Mild	Sunny	Normal	No	48
13	True	30	Mild	Rainy	Normal	Yes	48
14	False	35	Hot	Sunny	High	No	38
15	True	40	Mild	Overcast	High	Yes	52
16	False	40	Hot	Overcast	Normal	No	44
17	False	31	Cool	Mild	Normal	Yes	31
18	True	20	Mild	Sunny	High	No	30

Langkah – langkah penyelesaian sebagai berikut:

Membuat *subset* dengan cara mengambil sampel dan fitur secara acak. Misalkan sampel *subset* pohon ke-1 sebagai berikut:

Tabel 3.2 Subset Acak

Windy	Temperature	Outlook	Humidity	Number of fish
False	Hot	Rainy	High	25
True	Hot	Rainy	High	30
False	Hot	Overcast	High	46
False	Mild	Sunny	High	45
False	Cool	Sunny	Normal	52
True	Cool	Sunny	Normal	23
True	Cool	Overcast	Normal	43
False	Mild	Rainy	High	35
False	Cool	Rainy	Normal	38
False	Mild	Sunny	Normal	48
True	Mild	Rainy	Normal	48
True	Mild	Overcast	High	52
False	Hot	Overcast	Normal	44
True	Mild	Sunny	High	30

Lakukan tahap ini sebanyak k pohon

1. Selanjutnya melakukan proses *splitting* pohon keputusan *(decision node*) menggunakan algoritma ID3 dengan mengganti informasi *gain* dengan *standar deviation reduction.*
   a. Standar deviasi 1 atribut

Pohon keputusan di bangun dengan proses *top-down* dari akar simpul *(root node)* sampai ke simpul daun (*leaf node).* Setelah mendapatkan sampel subset selajutnya menentukan akar simpul *(root node).*

Tabel 3.3 Data *Number of fish*

Number of fish
25
30
46
45
52
23
43
35
38
48
48
52
44
30

Count = n =14

Rata=rata = $\bar{x}$ = 39.8

Standar deviasi = S = $\sqrt{\frac{\sum(x-\bar{x}^2}{n}}$=9.32

Coefficient of Deviation = CV = $\frac{S}{\bar{x}}$ x 100 % = 23 %

- Standar deviasi digunakan untuk membangun pohon
- *Coefficient of deviation* (CV) di gunakan untuk menghentikan percabangan
- Rata-rata adalah nilai (value) simpul daun *(leaf nodes)*

b. Standar deviasi 2 atribut (target dan *predicto*r)

$$S(T,X) = \sum_{c \in X} P(c)\, S(x) \qquad (3.2)$$

Tabel 3.4 Menghitung Standar Deviasi

		Number of fish (standar deviasi)	count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	sunny	10.87	5
			14

S(Number of fish, Outlook) = P(sunny) * S (sunny)+ P (overcast) * S (overcast)+ P (rainy) * S (rainy)

= (5/14)*10.87 + (4/14)*3.49 +(5/14)*7.78

= 7.66

a. *Standar deviation reduction*

Standar deviation reduction didasarkan pada penurunan deviasi standar setelah kumpulan data dipisahkan pada atribut. Pemilihan cabang di pilih berdasarkan *standar deviation reduction* terbesar.

Langkah pertama : menghitung standar deviasi target

Standar deviasi (*number of fish*) = 9.32

Langkah kedua: menghitung standar deviasi setiap variabel *predictor* dengan variabel target) kemudian dilakukan pengurangan standar deviasi. Tahap ini untuk menentukan variabel mana yang akan menjadi *root node*.

$$SDR\ (T,X) = S(T) - S(T,X) \quad (3.3)$$

		Number of fish (standar deviasi)
Outlook	Overcast	3.49
	Rainy	7.78
	sunny	10.87
SDR = 1.66		

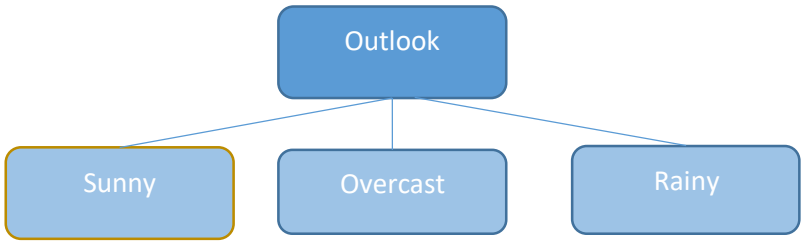
		Number of fish (standar deviasi)
Temp	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR = 0.17		

		Number of fish (standar deviasi)
Humidity	High	9.36
	Normal	8.37
SDR = 0.28		

		Number of fish (standar deviasi)
Windy	False	7.87
	True	10.59
SDR = 0.29		

Gambar 3. 4 Menghitung SDR Setiap Variabel

Berdasarkan Gambar 3.4 Memiliki selisih standar deviasi terbesar maka variabel *outlook* terpilih menjadi akar simpul *(root node)*



Gambar 3. 5 Proses *splitting*

Dari proses ini data terbagi-bagi sesuai cabang

Tabel 3.5 Data *Sunny*

Outlook	Temp	Humidity	Windy	Number of fish
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Sunny	Mild	Normal	False	45
Sunny	Mild	High	True	30

Tabel 3.6 Data *Overcast*

Outlook	Temp	Humidity	Windy	Number of fish
Overcast	Hot	High	False	46
Overcast	Cool	Normal	True	43
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44

Tabel 3.7 Data *Rainy*

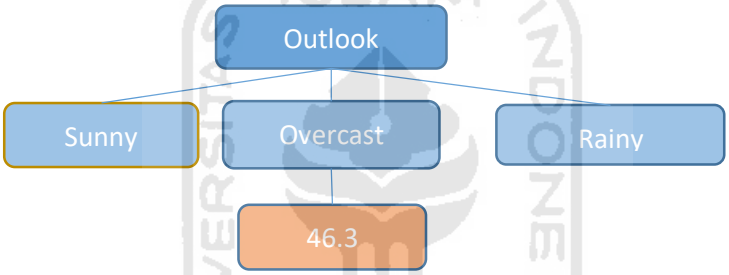
Outlook	Temp	Humidity	Windy	Number of fish
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Rainy	Mild	Normal	True	48

Langkah ketiga : Pemberhentian split. Kriteria pemberhentian cabang yaitu jika nilai CV setiap variabel kurang dari CV target, jika nilai CV variabel lebih kecil dari ambang tertentu misalnya *threshold* 10% atau banyak sampel pada cabang sedikit.

Tabel 3.8 Menentukan Pemberhentian Cabang

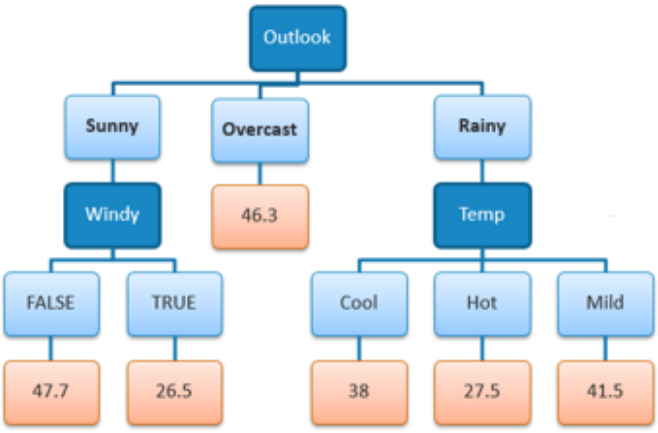
		Number of fish (standar deviasi)	*Number of fish* (rata-rata)	*Number of fish* (CV)	*Count*
Outlook	*Overcast*	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	sunny	10.87	39.2	28%	5

Berdasarkan tabel 3. Cabang *overcas*t memiliki CV kurang dari 10% dan kurang dari CV target maka cabang *overcas*t dilakukan pemberhentian cabang dengan nilai value sebesar 46.3.



Gambar 3. 6 Pemberhentian Cabang

Selanjutnya di lakukan proses s*plitting* lagi pada cabang *rainy* dan *sunny* dengan tahapan proses yang sama seperi langkah pertama sampai ketiga. Setelah melakukan tahap *splitting* di dapatkan pohon seperti gambar berikut:



Gambar 3. 7 *Decision Tree*

Menghitung nilai prediksi dari semua pohon dengan menghitung rata-rata untuk kasus regresi dan melakukan *majority vote* untuk kasus klasifikasi

3.6. *Tuning Parameter* dengan *Random Search CV*

Beberapa metode *machine learning*, terdapat nilai parameter yang diatur guna mendapatkan model yang optimal, yang disebut *hyperparameter*. *Hyperparameter* adalah titik pilihan atau konfigurasi yang memungkinkan model pembelajaran mesin disesuaikan untuk tugas atau kumpulan data tertentu. *Hyperparameter* digunakan untuk mengatur berbagai macam aspek dalam *machine learning* yang sangat berpengaruh pada performa dan model yang dihasilkan. Pencarian *hyperparameter* dilakukan secara manual atau dengan menguji kumpulan *hyperparameter* pada parameter yang ditentukan sebelumnya (Claesen, M., & De Moor, n.d.).

Salah satu metode *hyperparameter* yang dapat digunakan yaitu adalah *random search*. *Random search* merupakan metode alternatif yang digunakan untuk menemukan parameter terbaik dalam suatu model, sehingga metode yang digunakan secara akurat memprediksi data yang digunakan.

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

Tabel 3.9 Parameter pada Metode *Random Forest*

Sumber: (scikit-learn.org)

Parameter	Keterangan
n_estimators	Jumlah pohon pada *tree*
max_depth	Kedalaman maksimum pada *tree*
min_samples_split	Jumlah minimum sampel yang diperlukan untuk memisahkan node internal
min_samples_leaf	Pengukuran untuk kualitas *split*
max_features	Jumlah fitur yang dipertimbangkan saat mencari *split* terbaik

3.7. *Feature Importance*

Berbeda halnya dengan CART, *Random forest* memiliki banyak pohon keputusan yang di bentuk sehingga hasil interpretasinya sulit untuk di pahami. *Feature Importance* digunakan untuk melihat seberapa kuat variabel input mempengaruhi hasil prediksi. Ukuran yang bisa digunakan saat mengukur *feature selection* yaitu *Mean Decrease Accuracy* dimana MDA menampilkan besar tambahan observasi yang mengalami misklasifikasi jika satu persatu tidak diikutsertakan dalam pengujian. Semakin besar nilai MDG maka peubah tersebut dianggap semakin penting. Menurut Sandri dan Zuccolotto dalam penelitian berjudul "*A bias correction algorithm for the gini variable importance measure in classification trees*" , *Mean Decrease Gini* (MDG) merupakan salah satu ukuran tingkat kepentingan peubah penjelas yang dihasilkan oleh metode *random forest* (Sandri & Zuccolotto, 2008). Misalkan terdapat p peubah penjelas dengan h = 1,2,3,...,p. Maka MDG mengukur tingkat kepentingan peubah penjelas X_h dengan cara:

$$MDG_h = \frac{1}{k}\sum_t [d(h,t)I(h,t)] \qquad (3.4)$$

Dimana :

d(h,t) : Besar penurunan indeks gini untuk peubah penjelas X_h

I(h,t) : Memilih simpul t

k : Banyaknya pohon dalam model *random forest*

3.8. Evaluasi Model

Menurut Carlo Vercellis dalam bukunya yang berjudul "*Business Intelligence: Data Mining and Optimization for Decision Making"* , ada dua alasan utama untuk mengukur akurasi prediksi model. Pertama, pada tahap evaluasi model digunakan untuk mengidentifikasi model prediksi yang paling akurat, masing-masing model dianggap diterapkan pada data masa lalu, dan model dengan total error minimum dipilih. Kedua, perlu untuk secara berkala menilai keakuratan, untuk mendeteksi kelainan dan kekurangan dalam model yang mungkin timbul di lain waktu (Vercellis, 2009). Ukuran yang digunakan untuk mengevaluasi hasil prediksi dari model adalah

ukuran *R squared, Mean Absolute Percent Error (MAPE)* dan *Root Mean Square Error (RMSE*). Semakin kecil ukuran akurasi ini, maka semakin baik hasil prediksi dari model yang digunakan. Berbeda dengan RMSE dan MAE yang memberikan hasil berupa bilangan desimal, MAPE akan memberikan hasil berupa persentase (%), sehingga lebih mudah untuk digunakan sebagai tolak ukur bagus atau tidaknya suatu model.(Montaño Moreno et al., 2013) memberikan interpretasi dari nilai MAPE pada tabel berikut:

Tabel 3.10 Interpretasi MAPE

Interpretasi dari nilai MAPE	
<10%	*Highly accurate forecasting*
10-20%	*Good forecasting*
20-50%	*Reasonable forecasting*
>50%	*Inaccurate forecasting*
Source. : (Lewis, 1982)	

$$MSE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)}{n} \quad (3.5)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \times 100 \quad (3.6)$$

$$R^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \quad (3.7)$$

Dimana :

n : banyak data

y_i : data aktual

$\hat{y}_i$: data prediksi

$\bar{y}_i$: rata- rata

3.9. *Web Scraping*

Web scraping mengacu pada ekstraksi data dari situs web. Informasi ini dikumpulkan dan kemudian diekspor ke dalam format yang lebih berguna bagi pengguna. Format tersebut bisa disimpan dalam bentuk s*preadsheet* atau API. Menurut (Turland, 2010) *Web Scraping* adalah proses pengambilan sebuah dokumen semi-

terstruktur dari *internet*, umumnya berupa halaman-halaman *web* dalam bahasa *markup* seperti HTML atau XHTML, dan menganalisis dokumen tersebut untuk diambil data tertentu dari halaman tersebut untuk digunakan bagi kepentingan lain.

3.10. *Flask*

Web development adalah sebuah kegiatan pengembangan halaman-halaman *web* yang ada di *internet* Perkembangan teknologi saat ini sangat pesat terdapat banyak alat untuk membantu kegiatan *web development*, salah satunya adalah *web framework* yang dibuat menggunakan bahasa *python* yaitu *flask.* F*lask* adalah sebuah alat yang membantu membuat kerangka untuk sebuah *web* dan dengan menggunakan *Flask*, pengembang pemula pun dapat menciptakan sebuah *web* yang bagus. *Flask* adalah sebuah *web framework* yang ditulis dengan bahasa *Python* dan tergolong sebagai jenis *microframework* . *Flask* berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu *web.* Dengan menggunakan *Flask* dan bahasa *Python*, pengembang dapat membuat sebuah *web* yang terstruktur. *Flas*k termasuk pada jenis *microframework* karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi *form, database,* dan sebagainya tidak terpasang secara *default* di *Flask.* Meskipun Flask disebut sebagai *microframework,* bukan berarti *Flask* mempunyai kekurangan dalam hal fungsionalitas. *Microframework* disini berarti bahwa *Flask* bermaksud untuk membuat *core* dari aplikasi ini sesederhana mungkin tapi tetap dapat dengan mudah ditambahkan . Dengan begitu, fleksibilitas serta skalabilitas dari *Flask* dapat dikatakan cukup tinggi dibandingkan dengan *framework* lainnya (Irsyad, 2018).

3.11. *Heroku*

Heroku adalah layanan *PaaS (Platform as a Service)* yang memungkinkan pengembang untuk membangun, menjalankan, dan mengoperasikan aplikasi sepenuhnya di *cloud. Heroku* menyediakan *web service* bagi para pengembang aplikasi untuk menempatkan aplikasi mereka di ruang publik agar semua orang dapat mengaksesnya. (Wijaya et al., 2019).

BAB IV
METODOLOGI PENELITIAN

Bab ini berisikan rancangan penelitian yang dilakukan oleh penulis meliputi populasi dan sampel penelitian, sumber data, variabel penelitian, metode pengambilan data, tahapan penelitian, dan diagram alir.

4.1. Populasi dan Sampel

Penelitian ini menggunakan data sekunder yang didapatkan dari *website magicbricks*. Populasi yang digunakan dalam penelitian ini adalah semua harga rumah di India. Sedangkan Sampel yang digunakan disini adalah harga rumah di daerah kota Rohini, India.

4.2. Jenis dan Sumber data

Jenis data yang digunakan dalam penelitian ini merupakan data sekunder yaitu data atau informasi yang diambil dari *website magicbricks* menggunakan metode *web scraping*. Data yang digunakan adalah data penjualan rumah di Rohini, India sebanyak 1005 data. Berikut adalah linknya (*https://www.magicbricks.com/property-for-sale-in-rohini-new-delhi-pppfs/page*-1) diakses pada tanggal 8 Februari 2020.

4.3. Variabel Penelitian

Variabel yang digunakan dalam penelitian ini adalah :

a. Variabel Prediktor atau variabel bebas, yaitu *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type, per_sqft*

b. Variabel Respon atau variabel terikat, yaitu *Price*

Definisi operasional variabel pada penelitian ini sebagai berikut:

Tabel 4.1 Definisi Operasional Variabel (DOV)

No	Nama Variabel	Kategori	DOV	Satuan/ skala
1.	*Area*		Ukuran luas rumah dengan satuan square feet	Numerik

No	Nama Variabel	Kategori	DOV	Satuan/ skala
2.	*Price*		Nilai barang atau uang yang harus dibayarkan oleh konsumen atas	Numerik
3.	*BHK*		Banyak kamar tidur, ruang keluarga dan dapur dalam satu rumah	Numerik
4.	*Bathroom*		Banyak kamar mandi dalam satu rumah	Numerik
5.	*Per sqft*		Harga rumah berdasarkan luas *square feet*	Numerik
6.	*Furnishing*	0 = *Furnished* 1= *Unfurnished* 2 = *Semi Furnished*	Suatu perlengkapan rumah yang dilengkapi perabotan	Kategorik
7.	*Parking*		Banyak parkir mobil yang tersedia	Numerik
8.	*Status*	0 = *Ready to Move* 1 = *Under construction*	Keadaan atau kondisi pembangunan rumah saat di jual	Kategorik
9.	*Transaction*	0 = *New Properti* 1 = *Resold*	Status kepemilikan rumah saat di jual	Kategorik
10.	*Type*	0 = *Apartment* 1 = *Builder Floor*	Tipe atau jenis yang digunakan untuk membedakan suatu rumah dengan rumah lainnya	Kategorik

4.4. Metode Analisis Data

1. Analisis menggunakan metode *Random Forest* untuk mengetahui pemodelan dalam prediksi harga rumah serta keakuratan model yang terbentuk dengan menggunakan *Python 3.6.4.*
2. Model prediksi yang telah di analisis dalam *machine learning* di *deploy* menjadi sebuah *prototype* menggunakan *flask* agar terhubung dengan *localhost* proses ini menggunakan *Notepad++* dan *Command Prompt.*
3. Hasil model *prototype* dikembangkan menjadi sebuah *web* yang dapat diakses oleh masyarakat awam. Proses ini dilakukan menggunakan *Git, Command Prompt, Notepad++,* dan Heroku.

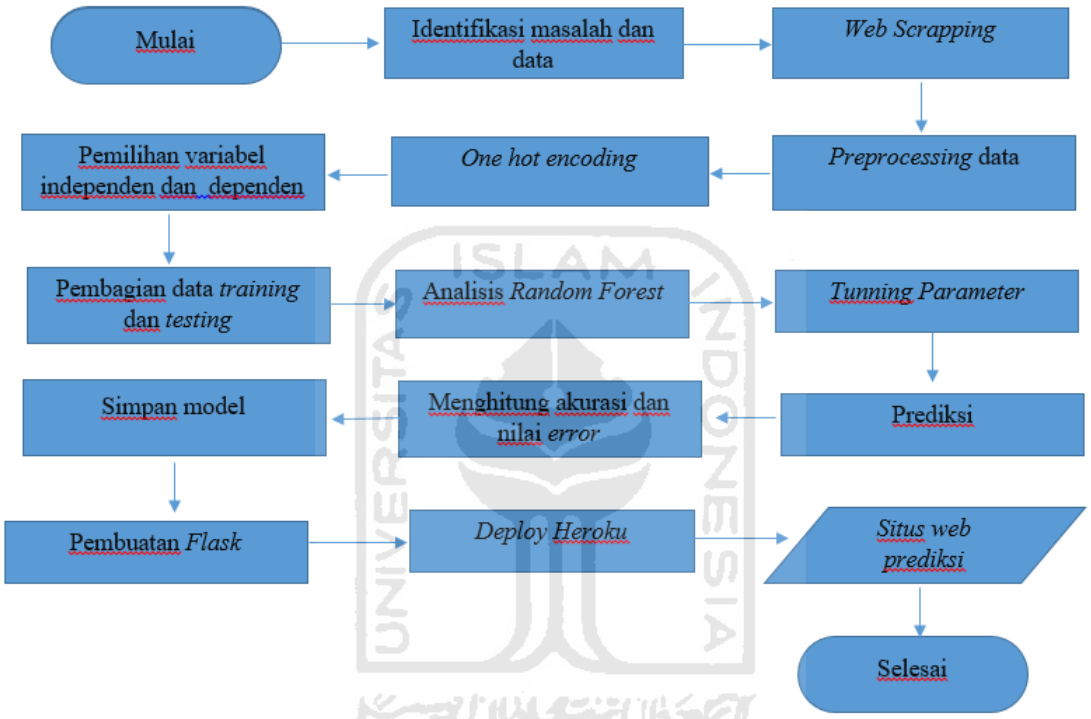
4.5. Alur Analisis Data

Penelitian ini dilakukan dengan beberapa tahapan sesuai diagram alur yang telah terbentuk pada gambar 4.1. Tahapan-tahapan tersebut antara lain:

1. Mengidentifikasi topic masalah dan objek penelitian yang akan digunakan.
2. Melakukan pengambilan data yang diperlukan, yaitu data dari *website magicbricks.com*.
3. Melakukan *Pre-processing* terhadap data
4. Melakukan *label encoder* untuk menjadikan data kualitatif yang awalnya dalam bentuk kata menjadi bilangan angka dengan skala nominal.
5. Melakukan seleksi dan mendefinisikan data yang menjadi variabel dependen dan independen. Pada penelitian ini variabel dependen yang digunakan adalah *price*, sedangkan *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type*.
6. Membagi data menjadi 2 bagian yaitu data *training* dan data *testing*, dimana proporsi data *training* lebih besar dibandingkan data *testing*. Data *training* digunakan melatih data dalam membentuk model, sedangkan data *testing* digunakan untuk memprediksi dan melihat keakuratan model yang terbentuk.
7. Melakukan analisis *Random Forest* untuk memperoleh *feature importance* dan model yang digunakan untuk memprediksi harga rumah.
8. Melakukan prediksi harga rumah data *testing* dan seluruh data yang digunakan dalam penelitian.
9. Menghitung nilai keakuratan model yang terbentuk. Nilai kesalahan model diperoleh berdasarkan nilai *Mean Absolute Percentage Error* (MAPE) dan *R-squared*
10. Menyimpan model prediksi yang telah dibentuk .py.
11. Mengembangkan model prediksi menjadi sebuah *prototype* menggunakan *flask* melalui *notepad++* dengan beberapa *script* yaitu *app.py* yang berisikan pengolahan data, dan *home.html* sebagai tampilan dalam *website*. *Script-script*

tersebut dijalankan menggunakan *command promt* kemudian dihubungkan kesalah satu *browser*.

12. Mengembangkan *prototype* sistem prediksi menjadi sebuah situs yang dapat diakses oleh khalayak menggunakan Heroku.

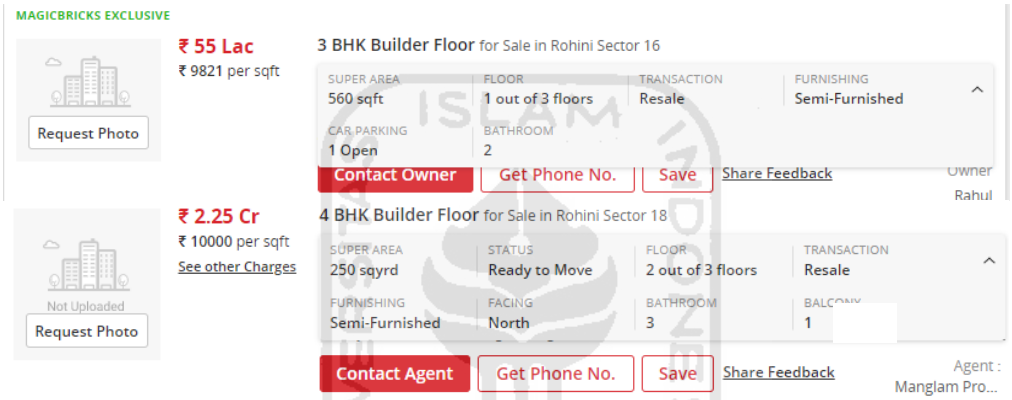


Gambar 4.1 *Flowchart* Penelitian

BAB V
HASIL DAN PEMBAHASAN

5.1 *Web Scrapping*

Pengumpulan data penjualan rumah pada penelitian ini dilakukan menggunakan metode *web scraping* menggunakan *python* dengan modul *BeautifulSoup*.



Gambar 5.1 Website *Magicbricks*

Gambar 5.1 merupakan gambaran data pada website *Magicbricks* yang terdiri atas harga rumah, tipe rumah, serta spesifikasi rumah lainnya. Berdasarkan Gambar 5.1 dilakukan *web scrapping* untuk melakukan ekstrasi data yang digunakan dari *website magicbricks.*

```
Localities = ['rohini']
for loc in Localities:
    Data = {}
    s = random.randrange(2,10)
    time.sleep(s)
    URL = 'https://www.magicbricks.com/property-for-sale-in-rohini-new-delhi-pppfs/page-1'
    r = requests.get(URL)
    soup = BeautifulSoup(r.content, 'html.parser')
    small = soup.findAll(class_='flex relative clearfix m-srp-card__container')
    for part in small:
        got = part.get_text().split()
        #print(got)
        for i in range(len(got)):
            ############ Price ###########
            if got[i]=='Cr':
                Data['Price']=float(got[i-1])*10000000
            elif got[i]=='Lac':
                Data['Price']=float(got[i-1])*100000
            ############ Area per Sqft/yard ############
            if got[i]=='per':
                if got[i+1]=='sqft':
                    Data['Per_Sqft']=float(got[i-1])
                elif got[i+1]=='sqyrd':
                    Data['Per_Sqft']=float(got[i-1])*9
                elif got[i+1]=='sqm':
                    Data['Per_Sqft']=float(got[i-1])*10.7639
```

Gambar 5.2 *Scrapping data*

Berdasarkan gambar 5.2 di lakukan pengampilan data dari *website* dan dilakukan penyamaan satuan harga dan satuan luas. Pada *website magicbricks* satuan harga rumah biasanya *lakh (lac)* atau *crore (cr)*. *Lac* dan *crore* merupakan sistem penomoran di India untuk menyatakan angka besar. 1 lac berarti 100.000 sedangkan 1 cr berarti 10.000.000. Berdasarkan harga rumah di *website magicbricks* 1 *lac* berarti 100.000 rupee sedangkan 1 *cr* berarti 10.000.000 rupee.

Terdapat satuan yang berbeda-beda pula yaitu *square feet (sqft), square meter (sqm),* dan *square yard (sqyrd)*. Pada tahap ini juga dilakukan penyamaan satuan luas yaitu menggunakan *square feet (sqft)*. 1 *square yard* sama dengan 9 *square feet* sedangkan 1 *square meter* sama dengan 10,7639 *square feet.* Berdasarkan luas rumah di *website magicbricks* jika terdapat luas rumah 2 *sqyrd* berarti rumah tersebut memiliki luas 18 *sqrt* sedangkan jika terdapat luas rumah 2 *sqm* berarti rumah tersebut memiliki luas 21,5278 *sqft.*

Setelah proses *web scraping* selesai dilakukan maka akan diperoleh data sebagai berikut:

Tabel 5. 1 Data Harga Rumah

Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
750	2	2	Semi-Furnished	1	Ready to move	New Property	Apartment	6667	5,000,000
950	2	2	Furnished	1	Ready to move	Resale	Apartment	6667	15,500,000
600	2	2	Semi-Furnished	1	Ready to move	Resale	Builder Floor	6667	4,200,000
650	2	2	Semi-Furnished	1	Ready to move	New Property	Builder Floor	6667	6,200,000
1300	4	3	Semi-Furnished	1	Ready to move	New Property	Builder Floor	6667	15,500,000

5.2 Analisis *Random Forest*

Metode *Random Forest Regressor* merupakan metode yang digunakan utuk membuat model prediksi rumah . Setelah mendapatkan data dengan teknik *web scrapping* di *website magicbricks* didapatkan 10 variabel yaitu *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type, Per_sqft* , dan *Price*. *Price* merupakan variabel dependen dengan tipe numerik yang akan diprediksi nilainya berdasarkan

pengaruh dari beberapa 9 variabel lainnya yang bersifat kualitatif. Data yang digunakan dalam penelitian ini sebanyak 1005 data.

Sebelum pembentukan model *Random Forest*, terdapat data kategorik di beberapa variabel yaitu *Furnishing, Status Transaction,* dan *Type*. Tahapan yang harus dilakukan adalah pelabelan data kategorik menjadi bilangan menggunakan skala nominal. Pelabelan ini dilakuan karena saat melakukan pemodelan *machine learning* mesin hanya membaca data berupa angka. Pelabelan data kategorik bisa dilakukan dengan metode *one hot enconding*. Metode *one hot enconding* dapat digunakan ketika terdapat variabel yang memiliki data nominal dan fitur data pelabelannya sedikit. Pada penelitian ini variabel yang memiliki data nominal yaitu variabel *type* Seperti yang telah dijelaskan pada tabel 4.1 bahwa data kategorik yang digunakan dalam penelitian terdiri atas 9 variabel yang merupakan variabel independen.

```
X = a[['Area', 'BHK', 'Bathroom', 'Furnishing', 'Parking','Status','Transaction','Type','Per_Sqft']]
Y = a.Price
X = pd.get_dummies(data=X)
```

urnishing_Furnished	Furnishing_Semi-Furnished	Furnishing_Unfurnished	Status_Almostready	Status_Readytomove	Transaction_New_Property	Transaction_Resale
0	1	0	0	1	1	0
0	1	0	0	1	1	0
1	0	0	0	1	0	1
0	1	0	0	1	0	1
0	0	1	0	1	1	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1

Gambar 5. 3 *Label Encod*

Gambar 5.5 merupakan hasil dari pelabelan data terlihat bahwa adanya variabel baru yang dibentuk. Proses pelabelan data menggunakan *one hot encoding* yaitu membuat kolom baru dari variabel kategorik dari data yang dimiliki dimana setiap kolom baru berisi nilai 0 atau 1 yang memiliki arti 0 berarti tidak ada dan 1 berarti ada. Berdasarkan gambar tersebut dapat diketahui bahwa variabel status, *transaction* dan *type* terdiri atas 2 kategorik dan variabel *furnishing* terdiri atas 3 kategorik. Setelah dilakukan pelabelan data terdapat 14 variabel independen baru yang terbentuk yaitu *Area, BHK, Bathroom, Per_sqft, Furnishing_semifurnished, Furnishing_furnished, Furnishing_unfurnished, Parking, Status_Almostready, Status_Readytomove,*

Transaction_newproperty, *Transaction_resale*, *Type_apartement*, dan *Type_builderfloor*.

Sebelum melakukan pemodelan menggunakan *random forest* tahap yang harus dilakukan adalah membagi data menjadi dua bagian yaitu data *training* dan data *testing*. Tahap ini dilakukan untuk mengukur kinerja model dengan menghitung segala bentuk kesalahan prediksi model. Data *training* digunakan untuk melatih algoritma dalam membentuk sebuah model, sedangkan data *testing* digunakan untuk mengevaluasi keakuratan model yang telah terbentuk. Apabila performa yang dihasilkan tinggi maka model tersebut dapat digunakan untuk menggambarkan prediksi suatu nilai dengan data yang baru. Data *training* dan data *testing* dibagi dengan proporsi 80% untuk data *training* dan 20% dari data *testing* dari total dataset

Tabel 5. 2 Proporsi Data *Training* dan *Testing*

Keterangan	**Data *Training***	**Data *Testing***	**Total**
Proporsi	80%	20%	100%
Jumlah	804	201	1005

Pada umumnya, data *training* memiliki proporsi yang lebih besar dibandingkan data *testing*. Berdasarkan tabel 5.2 dapat diketahui dari 1005 dataset yang ada, pembagian data untuk data *training* sebanyak 804 data, dan untuk data *testing* sebanyak 201 data. Pembagian data *training* dan *testing* pada dataset dilakukan secara *random* dengan bantuan *software Python*.

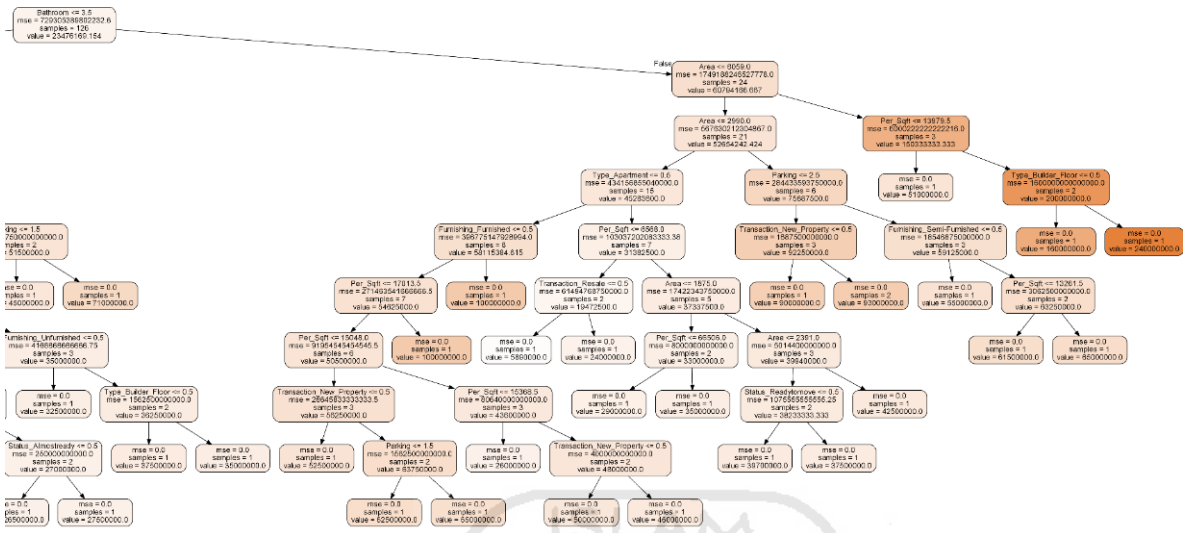
Tahapan berikutnya adalah melakukan analisis *Random Forest*. Langkah pertama yang dilakukan yaitu memilih parameter terbaik secara optimal untuk melakukan analisis *random forest*. *Tunning parameter* pada penelitian ini menggunakan metode *random search CV*. Penentuan parameter menggunakan *random search* yaitu memilih titik data pada setiap *features* secara acak. Hasil dari *tuning* parameter ditampilkan pada tabel berikut:

Tabel 5. 3 Hasil *Tuning Parameter*

Parameter	*Random Search Value*	*Best Parameter*
n_estimators	10,120, 230, 340, 450, 560, 670, 780, 890, 1000	340
max_depth	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110	100
min_samples_leaf	1, 2, 4	1
max_features	auto, sqrt	auto
min_samples_split	2, 5, 10	2

Tabel 5.3 merupakan hasil *tuning* parameter yang didapatkan dari proses *random searchCV* dengan melakukan pencarian secara acak terhadap parameter yang di ujikan. Penelitian ini menggunakan 3-*fold cross validation* yang digunakan untuk mengevaluasi kinerja model sebanyak tiga kali perulangan dalam proses *random search* dari setiap parameter. Nilai parameter terbaik dari proses *random search* digunakan dalam penentuan model prediksi.

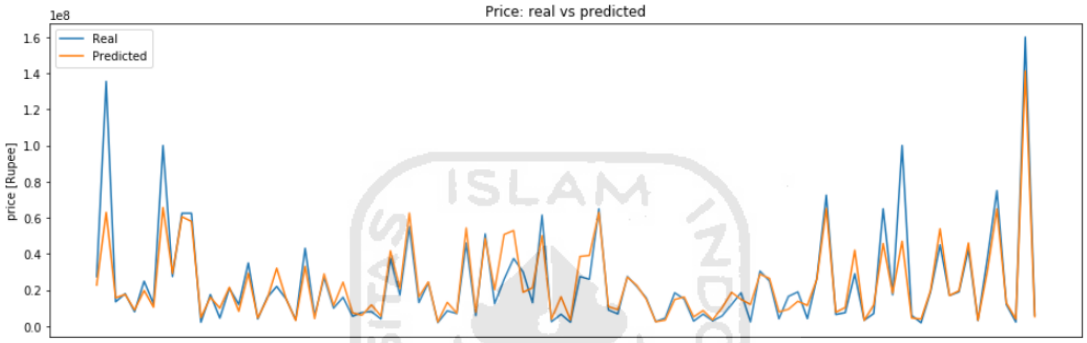
Pada analisis *Random Forest* terdapat 340 pohon yang digunakan, kemudian setiap pohon membuat maksimal seratus percabangan. Banyak fitur digunakan saat mencari *split* terbaik yaitu semua fitur yang terdapat pada data. Kemudian untuk mengukur kualitas *split* pada pohon digunakan nilai *mse*. Jumlah sampel minimum yang digunakan pada setiap *leaf node* sebanyak satu Jumlah sampel minimum yang digunakan pada saat pemberhentian *split* yaitu sebanyak satu. Setelah melakukan pembagian data dan pemilihan banyak pohon selanjutnya membangun pohon dengan hasil sebagai berikut:



Gambar 5.4 Hasil Pohon

Berdasarkan gambar 5.4 merupakan salah satu hasil pohon yang telah di bangun. Hasil keseluruhan pohon pada gambar 5.4 dilampirkan pada lampiran 4. Terdapat 3 *output* yang terlihat dari setiap *node* yang terbentuk yaitu *mse, samples*, dan *values. Mean square error* (MSE) berfungsi untuk mengukur kualitas *split*. Pada *root node* didapat nilai mse sebesar 729305389802232,6. *samples* merupakan banyak data yang terdapat pada *node*. Banyak sampel yang terdapat pada *root node* yaitu 124 sampel. *Value* merupakan hasil prediksi yang didapatkan dari nilai rata – rata sampel pada *node.* Nilai *value* yang terdapat pada *root node* yaitu 23476169,154. Algoritma pohon keputusan di bangun dengan proses *top-down* dari akar simpul *(root node)* sampai ke simpul daun (*leaf node)* dengan hasil keputusan pada setiap node yaitu *false* atau *true*. Interpretasi pohon dari gambar 5.4 bisa digambarkan sebagai berikut. Jika seorang calon pembeli rumah ingin menjual rumah baru dengan spesifikasi rumah memiliki *Area* seluas 7000 sqft , *4 BHK, 4 Bathroom,* dengan perobatan rumah, memiliki 3 parkir mobil, *Status ready to move, Type* rumah *builder floor* dan ingin menjual rumah 22000 rupee/*sqft* maka harga rumah yang di prediksi sebesar 160.000.000 rupee. Hasil ini didapat berdasarkan pohon pada gambar 5.4. dimana calon pembeli rumah ingin rumah dengan bathroom sebanyak 4 yang kemudian masuk ke *leaf node false*. Pada *leaf node false* muncul pertanyaan apakah rumah yang ingin

dibeli calon pembeli memiliki luas area rumah dari kurang 6059 sqft atau tidak. Pada *node* hasil yang dipilih yaitu *leaf node false* karena pembeli ingin membeli rumah dengan luas area 7000 *sqft.* Selanjutnya pada *leaf node* ini muncul pertanyaan apakah rumah yang ingin dibeli bertipe *builder floor* atau tidak. Hasil pada node ini yaitu *true* karena rumah yang ingin dibeli bertipe *builder floor*. Sehingga, hasil prediksi yang didapatkan pada *terminal node* yaitu sebesar 160.000.000 rupee.



Gambar 5.5 Perbandingan Data Prediksi dan Data Aktual

Tujuan utama regresi adalah untuk memprediksi respon dari model yang telah dihasilkan. Semakin dekat data prediksi dengan data sesungguhnya maka akan baik modelnya. Gambar 5.6 menunjukkan grafik perbandingan antara prediksi data *testing* dan data *testing* aktual. Dilihat dari gambar tersebut, nilai prediksi yang dihasilkan oleh model memiliki nilai yang hampir mendekati nilai aktual, meskipun hasil prediksi yang didapatkan dari model berada diatas atau dibawah nilai aktual. Ketepatan algoritma ini dapat menghasilkan model yang baik dengan akurasi yang tinggi, sehingga nilai prediksi semakin mendekati nilai aktual. Keakuratan yang tinggi dilihat dari nilai *error* dari model tersebut. Semakin kecil nilai *error*nya maka semakin akurat hasil yang didapatkan.

Tabel 5.4 Nilai Prediksi dan Nilai Aktual

Data ke-i	Harga Aktual (Rupee)	Harga Prediksi (Rupee)
1	19.000.000	18.718.656,7
2	3.700.000	3.900.835,82
3	3.000.000	3.105.567,16
4	82.500.000	74.148.880,6

Data ke-i	Harga Aktual (Rupee)	Harga Prediksi (Rupee)
5	3.800.000	4.286.950
6	5.600.000	6.505.950
7	8.600.000	9.744.550
8	1.700.000	4.956.650
9	14.200.000	15.077.500
…	…	…
201	6.200.000	5.313.650

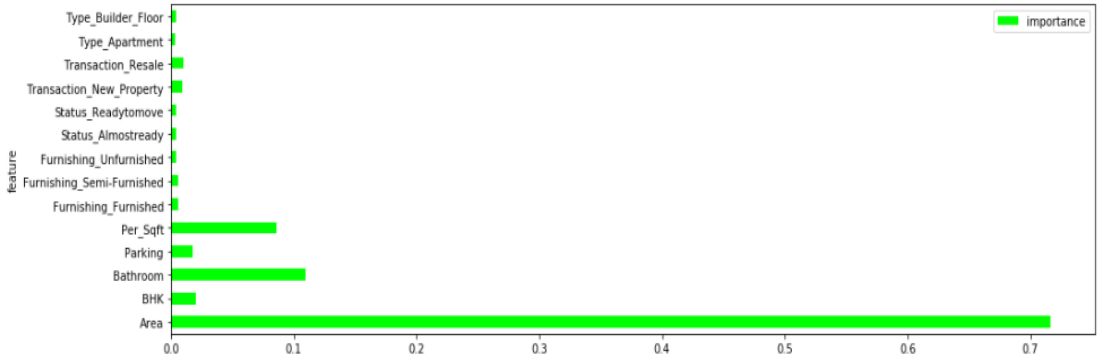
Berdasarkan tabel 5.4 didapatkan nilai prediksi dari data *testing* yang akan di bandingkan dengan nilai aktual harga rumah . Seperti grafik pada gambar 5.4. hasil prediksi harga rumah yang dihasilkan oleh model tidak berbeda jauh dari nilai aslinya. Pada data ke-1 pada tabel 5.4 harga rumah aktual sebesar 19.000.000 Rupee sedangkan model mendapatkan hasil prediksi sebesar 18.718.656,7Rupee. Hasil prediksi model didapatkan dengan cara memasukan semua variabel prediktor kesemua pohon. Hasil dari prediksi setiap pohon kemudian di rata-ratakan.

Tabel 5.5 Hasil Akurasi

RMSE	MAPE	R^2
11080543,63	26,48%	0,9423

Setelah mendapatkan model dilakukan evaluasi model. Tahap ini digunakan untuk mengidentifikasi keakuratan suatu model. Ukuran yang digunakan untuk mengevaluasi hasil prediksi dari model adalah ukuran *R squared, Mean Absolute Percent Error (MAPE)* dan *Root Mean Square Error (RMSE*). Berdasarkan tabel 5.5 diketahui nilai RMSE sebesar 11080543,63. Nilai RMSE digunakan untuk menggambarkan tingkat *error* data model yang digunakan. Semakin kecil nilai RMSE maka semakin tinggi nilai akurasi sistem. Sedangkan, nilai MAPE merupakan nilai rata-rata dari persentase seluruh selisih antara data akual dengan data dari hasil prediksi. Model prediksi yang terbentuk memiliki nilai MAPE sebesar 26,48%. Semakin rendah nilai MAPE maka semakin kecil tingkat kesalahan yang dihasilkan oleh model. Kemudian nilai *R-square* yang dihasilkan model sebesar 0,9423 atau

94,23%. Semakin nilai *R-square* mendekati angka 1 berarti model yang dihasilkan memberikan keakuratan hasil yang baik.



Gambar 5.6 *Features Importance*

Tahapan selanjutnya adalah mengukur kepentingan variabel independen terhadap harga rumah berdasarkan nilai *features importance*. Berdasarkan Gambar 5.5 menunjukkan *features importance* yang berguna untuk menggambarkan pemahaman data terhadap variabel yang lebih penting dalam pembentukan model dan penentuan prediksi. *feature importance* menunjukan hubungan variable itu sendiri dalam mempengaruhi hasil analisis/prediksi. Semakin besar angka *feature importance* menunjukan semakin besar juga peran variable tersebut dalam mempengaruhi hasil analisis. Nilai *feature importance* didapatkan dari perhitungan *mean decrease in impurity* (MDI). Berdasarkan Gambar 5.5 diketahui bahwa terdapat 5 variabel teratas yang memiliki pengaruh besar terhadap model prediksi harga rumah yaitu variabel *area* sebesar 0,708, variabel *bathroom* sebesar 0,109 , variabel *per_sqft* sebesar 0,089 , variabel BHK sebesar 0,021, dan variabel *parking* sebesar 0,02.

```
In [238]: from sklearn.externals import joblib

          joblib.dump(regressor, 'model.pkl')

Out[238]: ['model.pkl']
```

Gambar 5.7 Menyimpan Model

Setelah melakukan analisis, maka model yang diperoleh akan disimpan dan digunakan untuk merancang *web* aplikasi menggunakan fungsi *pickle* seperti gambar

5.6. Model disimpan dalam bentuk *pickle* agar bisa diproses untuk tahap *web deployment*.

5.3. Pembuatan *Web Deployment* Menggunakan *Flask* dan Heroku

Setelah mendapatkan model yang tepat dan disimpan dalam bentuk *pickle*, maka selanjutnya model tersebut akan dikembangkan menjadi sebuah *prototype web deployment* yang dapat digunakan untuk melakukan prediksi harga rumah yang bisa digunakan oleh masyarakat awam. Alat bantu untuk melakukan *web deployment* pada penelitian ini adalah *flask. Flask* adalah sebuah *web framework* yang ditulis dengan bahasa *Python* dan tergolong sebagai jenis *microframework web*. Berdasarkan model yang telah dibuat terdapat variabel – varaiabel prediktor yaitu *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type, per_sqft.* Maka dari itu dibutuhkan suatu HTML *form* yang berisikan seluruh variabel prediksi. *Form* tersebut dapat dibuat sesederhana mungkin ataupun dikembangkan agar lebih menarik. Seperti gambar 5.7 dan gambar 5.8 dibawah ini merupakan desain yang akan dibangun dalam tampilan *web* menggunakan *notepad+*. HTML *form* yang dibuat pada penelitian ini yaitu sebanyak dua. *Form* pertama yaitu *form* untuk memasukan nilai variabel independen yang disimpan dengan format *'index.html'*. *Form* kedua yaitu *form* berisi hasil prediksi rumah yang disimpan dengan format *'result..html'*. Kedua HTML *form* tersebut disimpan didalam sebuah folder bernama *templates*.

```
<div class="container-contact100">
    <div class="wrap-contact100">
        <form id="form_pred" class="contact100-form validate-form" method="post" action="/api">
            <span class="contact100-form-title">
                House Price Prediction
            </span>

            <div class="wrap-input100 validate-input" data-validate="Area is required">
                <span class="label-input100">Area</span>
                <input class="input100" type="text" id="Area" name="Area" placeholder="145364">
                <span class="focus-input100"></span>
            </div>

            <div class="wrap-input100 validate-input" data-validate="BHK  is required">
                <span class="label-input100">BHK  </span>
                <input class="input100" type="text" id="BHK" name="BHK" placeholder="2">
                <span class="focus-input100"></span>
            </div>

            <div class="wrap-input100 validate-input" data-validate="Bathroom is required">
                <span class="label-input100">Bathroom</span>
                <input class="input100" type="text" id="Bathroom" name="Bathroom" placeholder="1">
                <span class="focus-input100"></span>
            </div>

            <div class="wrap-input100 validate-input" data-validate="Parking is required">
                <span class="label-input100">Parking</span>
                <input class="input100" type="text" id="Parking" name="Parking" placeholder="2">
                <span class="focus-input100"></span>
            </div>
```

Gambar 5. 8 *index.html*

```
<script src="{{ url_for('static',filename='js/main.js') }}"></script>
<!-- include sweetAlert plugin -->
<script src="{{ url_for('static',filename='js/sweetalert2.all.js') }}"></script>
<script type="text/javascript">
    $(function () {

        $("button#predict").click(function(e){
            e.preventDefault();
            /*Get for variabes*/
            var Area = $("#Area").val(), BHK = $("BHK").val(), Bathroom = $("#Bathroom").val();
            var Parking = $("#Parking").val(), Per_Sqft = $("#Per_Sqft").val(), Furnishing = $("#Furnishing").val();
            var  Status= $("Status").val(), Type = $("#Type").val(), Transaction = $("#Transaction").val();
            /*create the JSON object*/
            var data = {"Area
            :Area, "BHK":BHK, "Bathroom":Bathroom, "Parking":Parking, "Per_Sqft":Per_Sqft, "Furnishing":Furnishing, "Sta
            /*send the ajax request*/
            $.ajax({
                method : "POST",
                url : window.location.href + 'api',
                data : $('form').serialize(),
                success : function(result){
                    var json_result = JSON.parse(result);
                    var price = json_result['Price'];
                    swal('Estimated price is '+Price+' DH', '','success')
                },
                error : function(){
                    console.log("error")
                }
            })
        })

    });
</script>
```

Gambar 5. 9 *result.html*

Selain itu, untuk membaca kode *template* tampilan *web* dan model prediksi harus dibangun sebuah *script* seperti gambar 5.9. *Script* tersebut berisikan fungsi *flask* yang digunakan untuk menghubungkan *python* yang berisikan model prediksi dengan tampilan *web* agar hasil prediksi dapat ditampilkan dan aplikasi dapat digunakan.. *Script* ini disimpan dengan format '*app.py*'.

```
from flask import Flask, abort, jsonify, request, render_template
from sklearn.externals import joblib
import numpy as np
import json

regressor = joblib.load('model.pkl')

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')


#@app.route('/api', methods=['POST'])
#def make_prediction():
    #data = request.get_json(force=True)
##convert our json to a numpy array
   # one_hot_data = input_to_one_hot(data)
   # predict_request = regressor.predict([one_hot_data])
   # output = [predict_request[0]]
   # print(data)
   # return jsonify(results=output)

def input_to_one_hot(a):
    # initialize the target vector with zero values
    enc_input = np.zeros(14)
    # set the numerical input as they are
    enc_input[0] = a['Area']
    enc_input[1] = a['BHK']
    enc_input[2] = a['Bathroom']
    enc_input[3] = a['Parking']
    enc_input[4] = a['Per_Sqft']
```

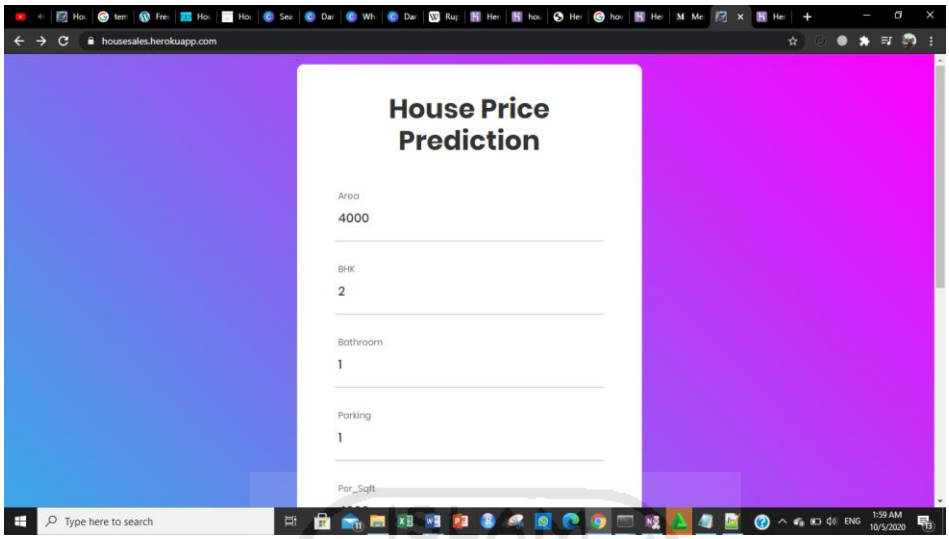
Gambar 5.10 *App.py*

Setelah membangun seluruh kode *script* yang dibutuhkan untuk *web deployment.* Kode *script flask* dapat dijalankan didalam *localhost*. *Script* tersebut dapat dijalankan dengan mengaktifkan *script flask app.py* didalam *command prompt* sehingga akan muncul pemberitahuan bahwa *flask* telah berjalan di *localhos*t sepeti gambar 5.10. Selanjutnya, buka aplikasi browser di komputer dan akses IP 127.0.0.1:5000 atau *localhost* di komputer tersebut.

```
* Restarting with stat
* Debugger is active!
* Debugger PIN: 314-291-864
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

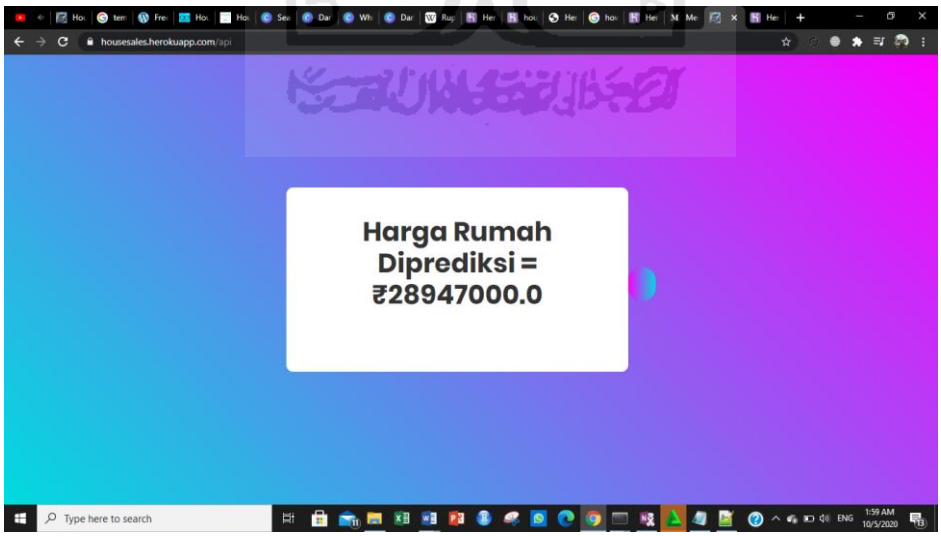
Gambar 5.11 Mengaktifkan *flask*

Setelah berhasil membuat model *flask*, selanjutnya rancangan tersebut akan di-*deploy* ke internet menggunakan Heroku agar dapat diakses oleh masyarakat umum . Proses *deploy* ini dilakukan pada halaman *command prompt* dan dibantu oleh Github serta beberapa *package*. Setelah proses *deploy* menggunakan Heroku selesai maka aplikasi prediksi harga rumah yang dirancang dapat diakses oleh masyarakat awam dengan alamat *https://housesales.herokuapp.com/* Hasil *web deployment* ditampikan sebagai berikut:



Gambar 5.12 Tampilan *Web*

Gambar 5.11 merupakan hasil dari *web deployment* dapat dilihat tampilan *web* aplikasi yang dirancang terdiri atas judul 9 variabel independen. Kemudian tombol prediksi "*how much does it cost"* berguna untuk menampilkan prediksi harga rumah berdasarkan kriteria variabel independen yang diinginkan. Hasil prediksi di tampilkan pada gambar 5.12



Gambar 5.13 Tampilan Hasil Prediksi

BAB VI
PENUTUP

6.1. KESIMPULAN

Berdasarkan hasil analisis yang dilakukan peneliti pada bab sebelumnya yakni hasil dan pembahasan, maka diperoleh kesimpulan sebagai berikut:

1. Prediksi harga rumah dilakukan menggunakan metode *random forest regression.* Berdasarkan dari hasil analisis *random forest regression* didapatkan pohon terbaik sebanyak 320 pohon, Faktor terbesar yang mempengaruhi prediksi harga rumah yaitu *Area*. Dengan demikian akurasi model yang diperoleh untuk memprediksi harga rumah memiliki akurasi sebesar 94,23% dan nilai *Mean Absolute Percent Error (MAPE)* sebesar 26,48%.
2. Website prediksi harga rumah dibangun melalui proses *web deployment* dengan bantuan *flask* dan *Heroku.* Alamat website yaitu *https://housesales.herokuapp.com*. Terdapat 9 variabel *input* pada *website* tersebut yaitu variabel *Area, BHK, Bathroom, Furnishing, Parking, Status, Transaction, Type,* dan *Per sqft.*

6.2. SARAN

Berdasarkan penelitian yang dilakukan, adapun saran yang diberikan oleh penulis sebagai berikut:

1. Pada penelitian selanjutnya dapat menambahkan variabel baru yang lebih berpengaruh untuk memprediksi harga jual beli rumah agar nilai akurasi model meningkat.
2. Dapat mengembangkan desain *website* hasil *deployment* menjadi lebih baik dan komunikatif.

DAFTAR PUSTAKA

Akmal, I. (2007). *Menata Apartemen*. Gramedia Pustaka Utama.

Bhagat, N., Mohokar, A., & Mane, S. (2016). House Price Forecasting using Data Mining. *International Journal of Computer Applications*. https://doi.org/10.5120/ijca2016911775

Borde, S., Rane, A., Shende, G., & Shetty, S. (2017). Real Estate Investment Advising Using Machine Learning. *International Research Journal of Engineering and Technology(IRJET)*.

Breiman, L. (2001). Random forests. *Machine Learning*. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. In *Classification and Regression Trees*. https://doi.org/10.1201/9781315139470

Budihardjo, E. (1998). *Sejumlah Masalah Pemukiman Kota*. Alumni.

Čeh, M., Kilibarda, M., Lisec, A., & Bajat, B. (2018). Estimating the Performance of Random Forest versus Multiple Regression for Predicting Prices of the Apartments. *ISPRS International Journal of Geo-Information*. https://doi.org/10.3390/ijgi7050168

Claesen, M., & De Moor, B. (n.d.). *Hyperparameter Search in Machine Learning*.

Hong, J., Choi, H., & Kim, W. S. (2020). A house price valuation based on the random forest approach: The mass appraisal of residential property in south korea. *International Journal of Strategic Property Management*. https://doi.org/10.3846/ijspm.2020.11544

Irsyad, R. (2018). *Penggunaan Python Web Framework Flask Untuk Pemula*. https://doi.org/10.31219/osf.io/t7u5r

Kasih, P. (2019). Pemodelan Data Mining Decision Tree Dengan Classification Error Untuk Seleksi Calon Anggota Tim Paduan Suara. *Innovation in Research of*

Informatics (INNOVATICS), *2*, 63–69.

Lewis. (1982). Industrial and Business Forecasting Methods: A Practical Guide to Exponential Smoothing and Curve Fitting. *Butterworth Scientific*. https://doi.org/10.1002/for.3980010202

Ma, Y., Zhang, Z., Ihler, A., & Pan, B. (2018). Estimating warehouse rental price using machine learning techniques. *International Journal of Computers, Communications and Control*. https://doi.org/10.15837/ijccc.2018.2.3034

Marlina, E. (2008). *Panduan Perancangan Bangunan Komersial* (D. Hardjono (ed.)). Andy Offset.

Mohd, T., Masrom, S., & Johari, N. (2019). Machine learning housing price prediction in petaling jaya, Selangor, Malaysia. *International Journal of Recent Technology and Engineering*, *8*(2 Special Issue 11), 542–546. https://doi.org/10.35940/ijrte.B1084.0982S1119

Montaño Moreno, J. J., Palmer Pol, A., Sesé Abad, A., & Cajal Blasco, B. (2013). Using the R-MAPE index as a resistant measure of forecast accuracy. *Psicothema*. https://doi.org/10.7334/psicothema2013.23

Park, S., Lee, K., Sung, S., Park, B., Kim, D., & Kim, H. (2019). A Study on an Apartment Price Prediction Model Using Machine Learning : An Example from Busan Metropolitan area. *Int'l Conf. Artificial Intelligence*, 79–85.

Prasetyo, B., & Trisyanti, U. (2018). Revolusi Industri 4.0 dan Tantangan Perubahan Sosial. *2018*.

Pratiwi, F. E., Pratiwi, F. E., & Zain, I. (2014). Klasifikasi Pengangguran Terbuka Menggunakan CART (Classification and Regression Tree) di Provinsi Sulawesi Utara. *Jurnal Sains Dan Seni ITS*, *3*(1), D54–D59. http://www.ejurnal.its.ac.id/index.php/sains_seni/article/view/6129

Rosi Subhiyakto, E., & Wahyu Utomo, D. (2017). Analisis Dan Perancangan Aplikasi Pemodelan Kebutuhan Perangkat Lunak Menggunakan Metode Prototyping. *Prosiding Seminar Nasional Multi Disiplin Ilmu & Call For Papers UNISBANK Ke-3(SENDI_U 3) 2017*, *207*, 57–62.

https://media.neliti.com/media/publications/174414-ID-analisis-dan-perancangan-aplikasi-pemode.pdf

Sandri, M., & Zuccolotto, P. (2008). A bias correction algorithm for the gini variable importance measure in classification trees. *Journal of Computational and Graphical Statistics*. https://doi.org/10.1198/106186008X344522

Sartono, B., & Syafitri, U. D. (2010). Metode Pohon Gabungan: Solusi Pilihan Untuk Mengatasi Kelemahan Pohon Regresi Dan Klasifikasi Tunggal. *Forum Statistika Dan Komputasi*, *15*(1), 1–7.

Sumartini, S. H., & Purnami, S. W. (2015). Penggunaan Metode Classification and Regression Trees (CART) untuk Klasifikasi Rekurensi Pasien Kanker Serviks di RSUD Dr. Soetomo Surabaya. *Jurnal Sains Dan Seni ITS*, *4*(2), 211–216. https://www.neliti.com/publications/15687/penggunaan-metode-classification-and-regression-trees-cart-untuk-klasifikasi-rek

Taylor, O. E., Ezekiel, P. S., & Okuchaba, F. B. D. (2019). A Model to Detect Heart Disease using Machine Learning Algorithm. *International Journal of Computer Sciences and Engineering*, *7*(11), 1–5. https://doi.org/10.26438/ijcse/v7i11.15

Turland, M. (2010). *Phparchitect's Guide to Web Scraping*. musketeers.me, LLC.

Unterman, R., & Small, R. (1983). *Perencanaan Tapak untuk Perumahan*. Intermedia.

Vercellis, C. (2009). Business Intelligence: Data Mining and Optimization for Decision Making. In *Business Intelligence: Data Mining and Optimization for Decision Making*. https://doi.org/10.1002/9780470753866

Wijaya, M., Junaedy, & Arfandy, H. (2019). Perancangan chatbot untuk informasi penerimaan mahasiswa baru pada stmik kharisma makassar. *Jurnal Ilmu Komputer*, 1–11.

Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*. https://doi.org/10.1007/s10115-007-0114-2

ISLAM
UNIVERSITAS
INDONESIA

RINGKASAN TUGAS AKHIR

LAMPIRAN

Lampiran 1 Data Harga Rumah

No	Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
1	750	2	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	6667	5000000
2	950	2	2	Furnished	1	Readytomove	Resale	Apartment	6667	15500000
3	600	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6667	4200000
4	650	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6667	6200000
5	1300	4	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6667	15500000
6	1350	4	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6667	10000000
7	650	2	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	6154	4000000
8	985	3	3	Unfurnished	1	Almostready	New_Property	Builder_Floor	6154	6800000
9	1300	4	4	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	15000000
10	1100	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	6200000
11	870	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	7700000
12	630	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	5500000
13	660	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6154	5000000
14	344.4448	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6154	3310000
15	660	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	4700000
16	550	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	4500000
17	1100	4	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	6154	17000000
18	1150	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	6154	25000000
19	650	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6154	6000000
20	850	2	2	Semi-Furnished	1	Readytomove	Resale	Apartment	6154	11000000
21	900	3	2	Semi-Furnished	5	Readytomove	Resale	Builder_Floor	6154	4500000
22	430.556	1	1	Furnished	2	Readytomove	Resale	Apartment	6154	5300000
23	900	2	2	Semi-Furnished	1	Readytomove	Resale	Apartment	6154	10500000
24	914.9315	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6154	12000000
25	1100	3	3	Furnished	1	Readytomove	New_Property	Builder_Floor	6364	7000000
26	800	3	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	8750	7000000
27	500	1	1	Semi-Furnished	1	Readytomove	Resale	Apartment	8750	3400000
28	28	2	2	Furnished	1	Readytomove	New_Property	Apartment	8750	5000000
29	2160	4	4	Furnished	1	Readytomove	New_Property	Builder_Floor	9722	21000000
30	900	2	2	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	14444	13000000
31	1000	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	22000	22000000

No	Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
32	900	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	22000	15800000
33	2925	4	4	Unfurnished	2	Readytomove	New_Property	Builder_Floor	22000	57500000
34	900	2	2	Unfurnished	2	Readytomove	Resale	Builder_Floor	22000	10000000
35	1600	3	3	Unfurnished	2	Readytomove	New_Property	Builder_Floor	22000	30000000
36	1143	3	3	Unfurnished	2	Readytomove	New_Property	Builder_Floor	22000	22000000
37	1150	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	22000	24000000
38	1500	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	22000	31500000
39	850	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	22000	14600000
40	8000	4	5	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	22000	240000000
41	890	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	22000	8300000
42	1100	3	3	Unfurnished	1	Readytomove	New_Property	Builder_Floor	22000	22500000
43	850	2	1	Semi-Furnished	2	Readytomove	New_Property	Apartment	2235	1900000
44	1350	3	2	Unfurnished	1	Readytomove	New_Property	Apartment	3800	5130000
45	1387	4	4	Unfurnished	1	Readytomove	New_Property	Apartment	4247	5890000
46	1050	3	2	Unfurnished	1	Readytomove	New_Property	Apartment	3524	3700000
47	1500	3	2	Unfurnished	1	Readytomove	Resale	Apartment	3524	17400000
48	1200	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	15500000
49	700	2	1	Semi-Furnished	1	Almostready	New_Property	Apartment	3524	2400000
50	1700	3	2	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	17200000
51	1550	3	2	Furnished	2	Readytomove	Resale	Apartment	3524	14200000
52	1400	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	13500000
53	1900	4	4	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	30000000
54	2160	4	3	Unfurnished	1	Readytomove	Resale	Apartment	3524	21000000
55	1700	4	3	Unfurnished	1	Readytomove	Resale	Apartment	3524	20500000
56	1850	3	4	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	24000000
57	1800	4	2	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	16500000
58	1620	3	2	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	14900000
59	1800	4	3	Furnished	1	Readytomove	Resale	Apartment	3524	19200000
60	1540	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	14500000
61	1600	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	15600000
62	1400	3	2	Unfurnished	1	Readytomove	Resale	Apartment	3524	11400000
63	2300	4	4	Furnished	1	Readytomove	Resale	Apartment	3524	31500000
64	1560	3	2	Furnished	2	Readytomove	Resale	Apartment	3524	15800000
65	1500	3	3	Semi-Furnished	2	Readytomove	Resale	Apartment	3524	16000000
66	1540	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	14500000

No	Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
67	1050	2	2	Unfurnished	1	Readytomove	Resale	Apartment	3524	10500000
68	1800	3	2	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	14800000
69	400	1	1	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	3524	1850000
70	1150	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	3524	6800000
71	1525	3	3	Semi-Furnished	1	Readytomove	Resale	Apartment	3524	14200000
72	1500	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	26666	40000000
73	2880	4	5	Unfurnished	1	Readytomove	New_Property	Builder_Floor	15972	46000000
74	2700	4	4	Semi-Furnished	2	Readytomove	Resale	Builder_Floor	183333	55000000
75	1400	3	3	Unfurnished	1	Readytomove	Resale	Builder_Floor	183333	24500000
76	1800	3	3	Unfurnished	1	Readytomove	Resale	Apartment	183333	24500000
77	1400	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	183333	32500000
78	1000	2	2	Furnished	1	Readytomove	Resale	Apartment	20000	20000000
79	8000	4	5	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	20000	240000000
80	1000	2	2	Furnished	1	Readytomove	Resale	Apartment	20000	20000000
81	1700	3	3	Furnished	2	Readytomove	Resale	Builder_Floor	20000	19000000
82	1600	3	3	Unfurnished	2	Readytomove	Resale	Builder_Floor	20000	26000000
83	1350	3	3	Unfurnished	1	Readytomove	Resale	Builder_Floor	15556	21000000
84	1700	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	16176	27500000
85	1400	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	16176	32500000
86	1450	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	16176	45000000
87	1350	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	15556	21000000
88	1700	3	3	Unfurnished	1	Readytomove	Resale	Builder_Floor	15556	28000000
89	1600	3	3	Unfurnished	2	Readytomove	Resale	Builder_Floor	15556	29500000
90	1350	2	2	Furnished	4	Readytomove	Resale	Builder_Floor	9259	12500000
91	1500	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	9259	26000000
92	750	2	2	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	9259	11500000
93	850	2	2	Semi-Furnished	2	Readytomove	Resale	Builder_Floor	9259	14600000
94	1575	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	9259	25500000
95	1675	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	18507	31000000
96	1350	3	3	Semi-Furnished	2	Readytomove	Resale	Builder_Floor	18507	22000000
97	900	2	2	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	18507	13400000
98	1800	3	3	Semi-Furnished	2	Almostready	New_Property	Builder_Floor	18507	35000000
99	900	2	2	Furnished	2	Readytomove	Resale	Builder_Floor	18507	11000000
100	900	2	2	Unfurnished	2	Readytomove	Resale	Builder_Floor	7000	6300000
101	900	2	2	Furnished	2	Readytomove	Resale	Builder_Floor	7000	12500000

No	Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
\102	800	2	2	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	14500000
103	1100	2	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	22500000
104	810	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	15500000
105	1500	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	26000000
106	900	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	15800000
107	825	2	2	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	15000000
108	900	2	2	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	13000000
109	875	2	2	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	15000000
110	1350	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	25000000
111	750	2	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	16500000
112	900	2	2	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	10300000
113	1350	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	7000	26000000
114	900	2	2	Unfurnished	1	Readytomove	Resale	Builder_Floor	7000	11000000
115	1800	3	3	Unfurnished	1	Readytomove	New_Property	Builder_Floor	7000	28500000
116	900	2	2	Unfurnished	1	Readytomove	Resale	Builder_Floor	7000	11000000
117	450	2	1	Unfurnished	1	Readytomove	Resale	Builder_Floor	72000	3600000
118	100	3	2	Furnished	1	Readytomove	New_Property	Builder_Floor	43000	4300000
119	900	3	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	4444	4000000
120	500	2	2	Semi-Furnished	1	Readytomove	Resale	Apartment	3300	1650000
121	450	1	1	Furnished	1	Readytomove	Resale	Builder_Floor	4444	2000000
122	75	2	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	40000	3000000
123	270	2	1	Semi-Furnished	1	Readytomove	Resale	Apartment	5185	1400000
124	500	2	1	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	4000	2000000
125	444	2	2	Unfurnished	1	Readytomove	Resale	Apartment	4000	2000000
126	400	1	1	Furnished	1	Readytomove	Resale	Builder_Floor	4000	2000000
127	729	3	4	Furnished	1	Readytomove	Resale	Builder_Floor	3567	2600000
128	4050	6	3	Furnished	1	Readytomove	Resale	Builder_Floor	1259	5100000
129	2400	6	4	Furnished	1	Readytomove	Resale	Builder_Floor	2500	6000000
130	450	2	2	Furnished	1	Readytomove	Resale	Builder_Floor	2500	4000000
131	405	2	3	Furnished	1	Readytomove	Resale	Builder_Floor	2500	6200000
132	666	1	2	Furnished	1	Readytomove	Resale	Builder_Floor	5255	3500000
133	1000	2	2	Semi-Furnished	1	Readytomove	Resale	Apartment	5255	7600000
134	1360	4	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	13500000
135	1550	4	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	12500000
136	850	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	6500000
137	2170	4	4	Unfurnished	2	Readytomove	New_Property	Builder_Floor	5255	22000000

No	Area	BHK	Bathroom	Furnishing	Parking	Status	Transaction	Type	Per_Sqft	Price
138	720	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	6000000
139	850	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	7300000
140	950	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	7600000
141	2300	4	4	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	20000000
142	1300	4	4	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	15000000
143	1100	3	2	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	5255	6200000
144	1030	3	2	Furnished	1	Readytomove	New_Property	Builder_Floor	5255	6200000
145	750	2	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	6667	5000000
146	720	2	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	6528	4700000
147	800	2	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	6875	5500000
148	350	1	1	Semi-Furnished	1	Readytomove	New_Property	Apartment	5714	2000000
149	980	3	2	Semi-Furnished	1	Readytomove	New_Property	Apartment	7449	7300000
….	……	….	…	……	….	…	……	….	…	……
985	1200	3	3	Semi-Furnished	1	Readytomove	New_Property	Builder_Floor	18333	22200000
986	1450	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	18333	22500000
987	1000	2	2	Semi-Furnished	1	Readytomove	Resale	Apartment	18333	7500000
988	2430	4	4	Furnished	5	Readytomove	Resale	Builder_Floor	18333	30000000
989	945	2	1	Unfurnished	5	Readytomove	Resale	Apartment	6878	6500000
990	1650	3	3	Unfurnished	5	Readytomove	New_Property	Builder_Floor	6878	35000000
991	1350	3	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	6878	17500000
992	2625	4	5	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	6878	60000000
993	1800	4	4	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	6878	40000000
994	125	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	6878	11500000
995	1440	3	3	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	6878	22500000
996	1900	3	4	Semi-Furnished	2	Readytomove	New_Property	Builder_Floor	6878	28500000
997	1800	3	3	Semi-Furnished	2	Readytomove	Resale	Builder_Floor	6878	29000000
998	1200	3	2	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	12916	15500000
999	1800	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	12916	26000000
1000	1200	3	3	Semi-Furnished	1	Readytomove	Resale	Builder_Floor	12916	16500000
1001	4118	4	5	Unfurnished	3	Readytomove	New_Property	Builder_Floor	12916	55000000
1002	1050	3	2	Semi-Furnished	3	Readytomove	Resale	Builder_Floor	12916	12500000
1003	875	3	3	Semi-Furnished	3	Readytomove	New_Property	Builder_Floor	12916	17500000
1004	990	2	2	Unfurnished	1	Readytomove	Resale	Builder_Floor	12916	11500000
1005	11050	3	3	Unfurnished	1	Readytomove	New_Property	Builder_Floor	12916	18500000

Lampiran 2 Pengambilan Data

9/27/2020 Untitled2

In [1]:
```
import requests
import pandas as pd
import time
import random
from bs4 import BeautifulSoup
```

In [2]:
```
Output = pd.DataFrame()
```

In [3]:
```
Localities = ['rohini']
for loc in Localities:
    Data = {}
    s = random.randrange(2,10)
    time.sleep(s)
    URL = 'https://www.magicbricks.com/property-for-sale-in-rohini-new-delhi-pppfs
    r = requests.get(URL)
    soup = BeautifulSoup(r.content, 'html.parser')
    small = soup.findAll(class_='flex relative clearfix m-srp-card__container')
    for part in small:
        got = part.get_text().split()
        #print(got)
        for i in range(len(got)):
            ############## Price ###########
            if got[i]=='Cr':
                Data['Price']=float(got[i-1])*10000000
            elif got[i]=='Lac':
                Data['Price']=float(got[i-1])*100000
            ############ Area per Sqft/yard ############
            if got[i]=='per':
                if got[i+1]=='sqft':
                    Data['Per_Sqft']=float(got[i-1])
                elif got[i+1]=='sqyrd':
                    Data['Per_Sqft']=float(got[i-1])*9
                elif got[i+1]=='sqm':
                    Data['Per_Sqft']=float(got[i-1])*10.7639
            ############ BHK apartment ####################
            if got[i]=='BHK':
                Data['BHK']=int(got[i-1])
            ########### Type of House:- Builder/Apartment ####
            if got[i]=='Apartment':
                Data['Type']='Apartment'
            elif got[i]=='Builder':
                Data['Type']='Builder_Floor'
            ############ Area ##############################
            if got[i]=='super' or got[i]=='carpet':
                if got[i+3]=='sqft':
                    Data['Area']=int(got[i+2])
                elif got[i+3]=='sqyrd':
                    Data['Area']=int(got[i+2])*9
                elif got[i+3]=='sqm':
                    Data['Area']=int(got[i+2])*10.7639
```

```
############ Status ########################
        if got[i]=='status':
            if got[i+1]=='Ready':
                Data['Status']='Ready_to_move'
            elif got[i+1]=='Possession':
                Data['Status']='Almost_ready'
############ Transaction ########################
        if got[i]=='transaction':
            if got[i+1]=='New':
                Data['Transaction']='New_Property'
            elif got[i+1]=='Resale':
                Data['Transaction']='Resale'
############ Furnishing ########################
        if got[i]=='furnishing':

            if got[i+1]=='Unfurnished':
                Data['Furnishing']='Unfurnished'
            elif got[i+1]=='Furnished':
                Data['Furnishing']='Furnished'
            elif got[i+1]=='Semi-Furnished':
                Data['Furnishing']='Semi-Furnished'
############ Parking ########################
        if got[i]=='parking':
            Data['Parking']=int(got[i+1])
############ Locality ########################
        if got[i]=='in':
            j=i+1
            string =got[j]
            j+=1
            while(j<len(got)):
                if got[j]=="What's":
                    break
                else :
                    string +=(' '+got[j])
                j+=1
            Data['Locality']=string


############ Bathroom ########################
        if got[i]=='bathroom':
            Data['Bathroom']=int(got[i+1])
            break

    Output = Output.append(Data,ignore_index=True)

print(r.status_code)
```

```
200
```

In [4]:
```
Output
```

Out[4]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	
0	1100.0000	3.0	2.0	Semi-Furnished	Sunrise Apartment, Rohini Sector 13 carpet are...	1.0	26000000.0	Ready_to_move	
1	1100.0000	3.0	2.0	Semi-Furnished	Rohini Sector 3 carpet area 1100 sqft status R...	1.0	22500000.0	Ready_to_move	N
2	1100.0000	3.0	2.0	Semi-Furnished	other new and old sectors:::home loan availab...	1.0	8300000.0	Ready_to_move	
3	1100.0000	3.0	2.0	Semi-Furnished	Rohini Sector 3 carpet area 1100 sqft status R...	1.0	18000000.0	Ready_to_move	N
4	1480.0000	3.0	2.0	Semi-Furnished	Kadambari Apartments, Rohini Sector 9 carpet a...	1.0	19200000.0	Ready_to_move	
5	1100.0000	3.0	2.0	Semi-Furnished	Rohini Sector 3 carpet area 1100 sqft status R...	1.0	17000000.0	Ready_to_move	N
6	1100.0000	3.0	2.0	Semi-Furnished	Rohini Sector 3 carpet area 1100 sqft status R...	1.0	16000000.0	Ready_to_move	N
7	1250.0000	3.0	2.0	Furnished	Overseas Apartment, Rohini Sector 9 carpet are...	1.0	25000000.0	Ready_to_move	
8	700.0000	2.0	2.0	Semi-Furnished	Rohini Sector 22 super area 700 sqft floor Gro...	1.0	4500000.0	Ready_to_move	
9	780.0000	3.0	3.0	Semi-Furnished	Rohini carpet area 780 sqft status Ready to Mo...	1.0	7000000.0	Ready_to_move	
10	1100.0000	3.0	2.0	Furnished	Oriental Apartment, Rohini Sector 9 carpet are...	1.0	17000000.0	Ready_to_move	
11	500.0000	2.0	2.0	Furnished	Rohini Sector 24 carpet area 500 sqft status R...	1.0	4000000.0	Ready_to_move	
12	1200.0000	3.0	2.0	Semi-Furnished	Rohini Sector 22 super area 1200 sqft status R...	1.0	8600000.0	Ready_to_move	

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	
13	312.1531	1.0	1.0	Unfurnished	DDA Pocket G 2, Rohini Sector 15 carpet area 2...	1.0	2600000.0	Ready_to_move	
14	4843.7550	6.0	6.0	Unfurnished	Rohini Sector 11 carpet area 450 sqm status Re...	3.0	74000000.0	Ready_to_move	
15	720.0000	2.0	2.0	Unfurnished	Vijay Vihar, Rohini super area 80 sqyrd status...	3.0	3500000.0	Ready_to_move	
16	452.0838	1.0	1.0	Unfurnished	Rohini Sector 28 super area 42 sqm status Read...	1.0	3500000.0	Ready_to_move	
17	882.0000	3.0	2.0	Semi-Furnished	Prashant Vihar, Rohini carpet area 98 sqyrd st...	1.0	16000000.0	Ready_to_move	
18	516.6672	2.0	2.0	Semi-Furnished	Rohini Sector 20 super area 48 sqm status Read...	1.0	4000000.0	Ready_to_move	
19	1800.0000	3.0	3.0	Semi-Furnished	Rohini Sector 7 super area 1800 sqft status Re...	1.0	22500000.0	Ready_to_move	
20	378.0000	1.0	1.0	Unfurnished	Rohini Sector 29 carpet area 42 sqyrd status R...	1.0	2500000.0	Ready_to_move	N
21	900.0000	2.0	2.0	Semi-Furnished	Chandra Priya Apartment, Rohini Sector 8 carpe...	1.0	8600000.0	Ready_to_move	
22	1100.0000	2.0	1.0	Unfurnished	Rohini Sector 21 super area 1100 sqft status R...	1.0	8500000.0	Ready_to_move	
23	630.0000	2.0	2.0	Semi-Furnished	Rohini Sector 24 carpet area 630 sqft status R...	1.0	4900000.0	Ready_to_move	
24	340.0000	1.0	1.0	Semi-Furnished	Rohini Sector 34 super area 340 sqft status Re...	1.0	2000000.0	Ready_to_move	
25	300.0000	3.0	1.0	Unfurnished	Rohini Sector 17 carpet area 300 sqft status R...	1.0	2000000.0	Ready_to_move	
26	20.0000	1.0	1.0	Furnished	DDA LIG Flats Rohini, Rohini carpet area 20 sq...	1.0	2500000.0	Ready_to_move	N

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status
27	20.0000	1.0	1.0	Furnished	near future. For details on the plot please co...	1.0	1250000.0	Ready_to_move
28	400.0000	1.0	1.0	Unfurnished	Rohini Sector 28 super area 400 sqft status Re...	1.0	2000000.0	Ready_to_move
29	585.0000	2.0	1.0	Unfurnished	Avantika Enclave, Rohini carpet area 585 sqft ...	1.0	3500000.0	Ready_to_move

In [6]:
```
Output.to_csv('MagicBricks pages .csv')
```

Lampiran 3 *Syntax* Analisis *Random Forest* Menggunakan *Python*

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display
from IPython import get_ipython
import seaborn as sns
import warnings
import pickle
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
# In[2]:
import os # accessing directory structure
# In[3]:
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_squared_error as mse
```

In [2]:
```
df1 = pd.read_csv('MagicBricks.csv', delimiter=',')
```

In [3]:
```
df1.head()
```

Out[3]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Status	Transaction	Type
0	800.0	3	2.0	Semi-Furnished	Rohini Sector 25	1.0	Readytomove	New_Property	Builder_Floo
1	750.0	2	2.0	Semi-Furnished	J R Designers Floors, Rohini Sector 24	1.0	Readytomove	New_Property	Apartmen
2	950.0	2	2.0	Furnished	Citizen Apartment, Rohini Sector 13	1.0	Readytomove	Resale	Apartmen
3	600.0	2	2.0	Semi-Furnished	Rohini Sector 24	1.0	Readytomove	Resale	Builder_Floo
4	650.0	2	2.0	Semi-Furnished	Rohini Sector 24 carpet area 650 sqft status R...	1.0	Readytomove	New_Property	Builder_Floo

In [4]:
```
df1=df1.dropna()
```

In [5]:
```
a=df1
```

In [6]:
```
a=a.dropna()
```

In [7]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1005 entries, 1 to 1258
Data columns (total 11 columns):
Area           1005 non-null float64
BHK            1005 non-null int64
Bathroom       1005 non-null float64
Furnishing     1005 non-null object
Locality       1005 non-null object
Parking        1005 non-null float64
Status         1005 non-null object
Transaction    1005 non-null object
Type           1005 non-null object
Per_Sqft       1005 non-null float64
Price          1005 non-null int64
dtypes: float64(4), int64(2), object(5)
memory usage: 94.2+ KB
```

In [8]:
```
del a["Locality"]
```

In [9]:
```
X = a[['Area', 'BHK', 'Bathroom', 'Furnishing', 'Parking','Status','Transaction','
Y = a.Price
X = pd.get_dummies(data=X)
```

In [10]:
```
X.head(5)
```

Out[10]:

	Area	BHK	Bathroom	Parking	Per_Sqft	Furnishing_Furnished	Furnishing_Semi-Furnished	Furnishing_
1	750.0	2	2.0	1.0	6667.0	0	1	
2	950.0	2	2.0	1.0	6667.0	1	0	
3	600.0	2	2.0	1.0	6667.0	0	1	
4	650.0	2	2.0	1.0	6667.0	0	1	
5	1300.0	4	3.0	1.0	6667.0	0	1	

In [11]:
```
# now we use the train_test_split function already available in sklearn library to
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = .20, random_
```

In [12]:
```
X_train.head(5)
```

Out[12]:

	Area	BHK	Bathroom	Parking	Per_Sqft	Furnishing_Furnished	Furnishing_Semi-Furnished	Furnis
109	8000.0000	4	5.0	1.0	20000.0	0	1	
47	900.0000	2	2.0	2.0	14444.0	0	1	
327	495.1394	1	1.0	1.0	41304.0	1	0	
581	810.0000	2	2.0	1.0	14818.0	0	1	
815	1350.0000	3	3.0	1.0	3294.0	0	0	

In [14]:
```
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 1000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

```
{'n_estimators': [10, 120, 230, 340, 450, 560, 670, 780, 890, 1000], 'max_feat
ures': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 10
0, 110, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4],
 'bootstrap': [True, False]}
```

Out[14]:
```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

In [15]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
# Fit the random search model
rf_random.fit(X_train, Y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  2.7min
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:  4.7min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:  6.9min finished
```

Out[15]:
```
RandomizedSearchCV(cv=3, error_score='raise',
          estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_
depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False),
          fit_params=None, iid=True, n_iter=100, n_jobs=-1,
          param_distributions={'n_estimators': [10, 120, 230, 340, 450, 560, 67
0, 780, 890, 1000], 'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30,
40, 50, 60, 70, 80, 90, 100, 110, None], 'min_samples_split': [2, 5, 10], 'min_
samples_leaf': [1, 2, 4], 'bootstrap': [True, False]},
          pre_dispatch='2*n_jobs', random_state=42, refit=True,
          return_train_score='warn', scoring=None, verbose=2)
```

In [16]:
```
rf_random.best_params_
```

Out[16]:
```
{'n_estimators': 340,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 100,
 'bootstrap': True}
```

In [47]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=340, random_state=0, min_samples_sp
regressor.fit(X_train, Y_train)
y_pred = regressor.predict(X_test)
```

In [18]:
```
regressor
```

Out[18]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=100,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=670, n_jobs=1,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [48]:
```
regressor.fit(X_train, Y_train)

# Score model
regressor.score(X_train, Y_train)
```

Out[48]: 0.9791027156718145

In [49]:
```
regressor.fit(X_test, Y_test)

# Score model
regressor.score(X_test, Y_test)
```

Out[49]: 0.9423611983292544

In [24]:
```
Y_test
```

Out[24]:

1159	19000000
775	3700000
856	3000000
588	82500000
418	3800000
818	7000000
819	5600000
609	8600000
373	1700000
89	14200000
1107	13500000
424	20000000
953	2600000
528	5500000
288	70000000
756	1500000
880	1800000
1233	47500000
347	30000000
135	14500000
1212	35000000
747	14000000
251	90000000
278	20000000
710	16500000
328	7000000
430	16000000
335	5000000
441	12000000
1100	4600000
	...
553	30500000
565	25000000
312	4100000
144	16500000
540	19000000
917	4200000
605	26500000
297	72500000
307	6500000
624	7500000
466	29000000
489	3300000
773	7000000
575	65000000
493	17400000
912	100000000
173	6000000
410	1900000
1161	19000000
259	45000000
939	17000000
711	19000000
663	42500000
416	3200000

```
597      37500000
246      75000000
981      12000000
386       2300000
248     160000000
403       6200000
Name: Price, Length: 201, dtype: int64
```

In [50]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_test, y_pre
print('MAPE :', np.mean(np.abs((Y_test - y_pred) / y_pred))*100)
```

```
Mean Absolute Error: 5767885.5965271685
Mean Squared Error: 122778447254294.77
Root Mean Squared Error: 11080543.635322897
MAPE : 26.48008888127927
```

In [27]:
```
to_pred = [800, 2, 2, 'Semi-Furnished', 1, 'Readytomove', 'New_Property','Apartmen
```

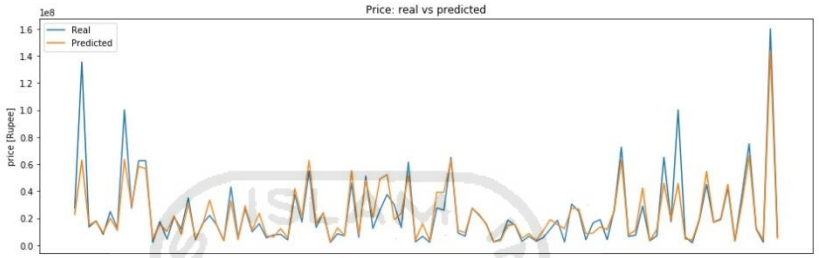
In [28]:
```
A = Y_test.values.reshape(-1, 1)
```

In [29]:
```
B = y_pred.reshape(-1, 1)
```

In [53]:
```
B
```

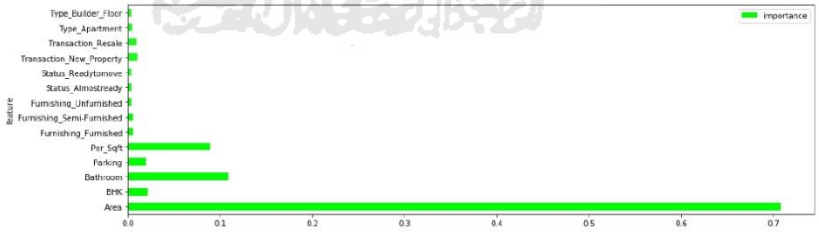
```
[1.16424627e+07],
[2.55128955e+07],
[6.31167164e+07],
[7.82534328e+06],
[1.09136269e+07],
[4.23571642e+07],
[3.24880597e+06],
[1.15975771e+07],
[4.58801493e+07],
[1.89122388e+07],
[4.56510448e+07],
[4.61304975e+06],
[3.95210448e+06],
[1.95534328e+07],
[5.45728358e+07],
[1.71501493e+07],
[1.95534328e+07],
[4.50313433e+07],
[3.26313433e+06],
[3.15442090e+07],
```

In [30]:
```
plt.rcParams['figure.figsize'] = 16,5
plt.figure()
plt.plot(A[-100:], label="Real")
plt.plot(B[-100:], label="Predicted")
plt.legend()
plt.title("Price: real vs predicted")
plt.ylabel("price [Rupee]")
plt.xticks(())
plt.show()
```



In [31]:
```
#Feature Importance
# Feature importances into a dataframe
features = list(X_train.columns)
feature_importances = pd.DataFrame({'feature': features, 'importance': regressor.f
feature_importances .plot(x ='feature', y='importance', kind = 'barh', color="lime
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1ef6190e470>



In [32]:
```
importance = regressor.feature_importances_
importance_df = pd.DataFrame(importance, index=X_train.columns,
                             columns=["Importance"])
```

In [33]:
```
importance_df
```

Out[33]:

	Importance
Area	0.708216
BHK	0.021261
Bathroom	0.109251
Parking	0.020251
Per_Sqft	0.089053
Furnishing_Furnished	0.005531
Furnishing_Semi-Furnished	0.005582
Furnishing_Unfurnished	0.004270
Status_Almostready	0.003922
Status_Readytomove	0.004403
Transaction_New_Property	0.010344
Transaction_Resale	0.009293
Type_Apartment	0.004597
Type_Builder_Floor	0.004026

In [34]:
```
user_input = {'Area':750, 'BHK':2, 'Bathroom':2.0,'Parking':1.0,'Per_Sqft':6667.0,
def input_to_one_hot(a):
    # initialize the target vector with zero values
    enc_input = np.zeros(14)
    # set the numerical input as they are
    enc_input[0] = a['Area']
    enc_input[1] = a['BHK']
    enc_input[2] = a['Bathroom']
    enc_input[3] = a['Parking']
    enc_input[4] = a['Per_Sqft']
    ##################### Mark ########################
    # get the array of marks categories
    furnishing = df1.Furnishing.unique()
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Furnishing_'+a['Furnishing']
    # search for the index in columns name list
    furnishing_column_index = X.columns.tolist().index(redefinded_user_input)
    #print(mark_column_index)
    # fullfill the found index with 1
    enc_input[furnishing_column_index] = 1
    ##################### Fuel Type ####################
    # get the array of fuel type
    status = df1.Status.unique()
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Status_'+a['Status']
    # search for the index in columns name list
    status_column_index = X.columns.tolist().index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[status_column_index] = 1
    # get the array of fuel type
    type_types = df1.Type.unique()
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Type_'+a['Type']
    # search for the index in columns name list
    type_column_index = X.columns.tolist().index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[type_column_index] = 1
    # get the array of fuel type
    transaction_types = df1.Transaction.unique()
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Transaction_'+a['Transaction']
    # search for the index in columns name list
    transaction_column_index = X.columns.tolist().index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[transaction_column_index] = 1

    return enc_input
```

In [35]:
```
print(input_to_one_hot(user_input))
```

```
[7.500e+02 2.000e+00 2.000e+00 1.000e+00 6.667e+03 0.000e+00 1.000e+00
 0.000e+00 0.000e+00 1.000e+00 1.000e+00 0.000e+00 1.000e+00 0.000e+00]
```

In [36]:
```
b = input_to_one_hot(user_input)
```

In [37]:
```
b
```

Out[37]:
```
array([7.500e+02, 2.000e+00, 2.000e+00, 1.000e+00, 6.667e+03, 0.000e+00,
       1.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 1.000e+00, 0.000e+00,
       1.000e+00, 0.000e+00])
```

In [38]:
```
price_pred = regressor.predict([b])
```

In [39]:
```
price_pred[0]
```

Out[39]:
```
13229761.19402985
```

In [40]:
```
estimator = regressor.estimators_[100]
```

In [41]:
```
estimator
```

Out[41]:
```
DecisionTreeRegressor(criterion='mse', max_depth=100, max_features='auto',
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=301492857, splitter='best')
```

In [42]:
```
from IPython.display import Image
from sklearn import tree
import pydotplus
import os
os.environ["PATH"] += os.pathsep + 'C:\Program Files (x86)\Graphviz2.38/bin/'
```

In [43]:
```
from sklearn.tree import export_graphviz
# Export as dot file
export_graphviz(estimator, out_file='tree.dot',
                feature_names = X.columns,
                class_names = Y,
                rounded = True, proportion = False,
                precision = 3, filled = True)
```

In [44]:
```
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])
```

Out[44]:
```
0
```

In [45]:
```
# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'tree.png')
```

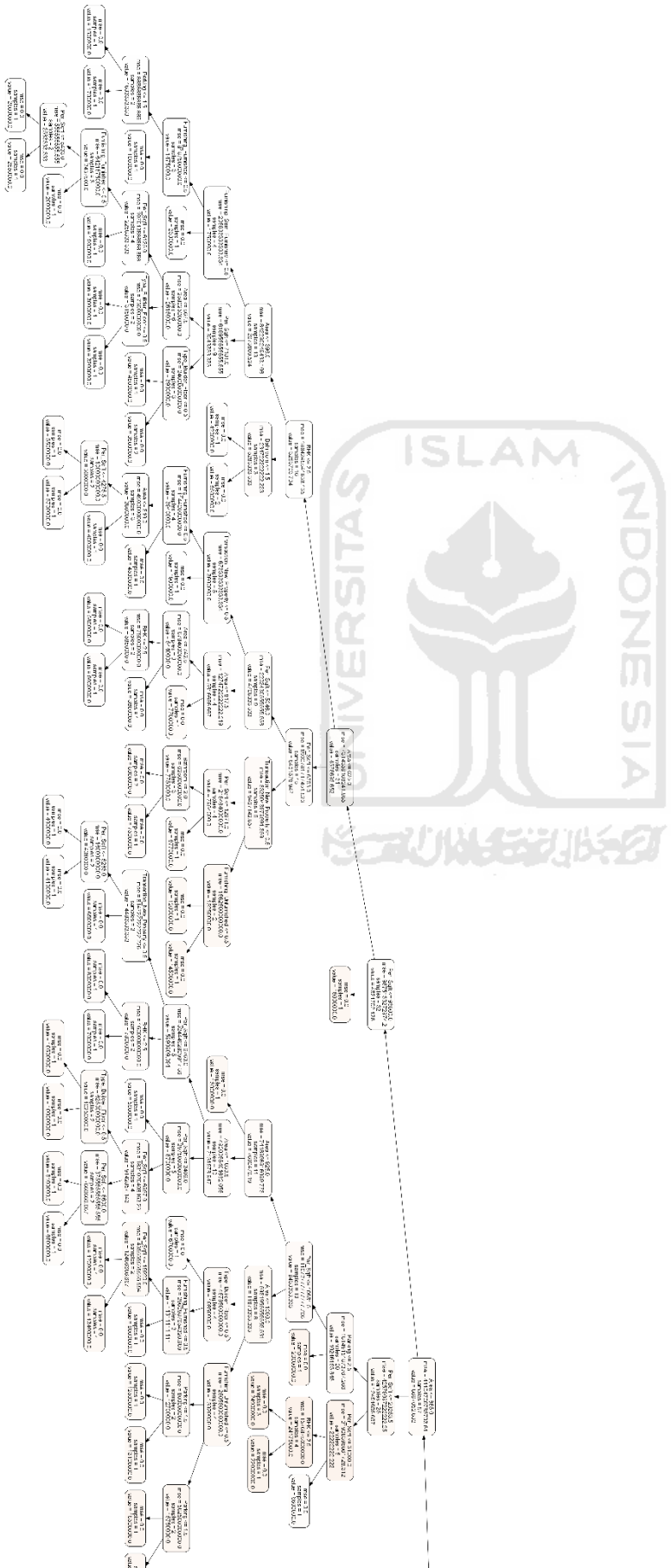
Out[45]:

In [ ]:
```
from sklearn.externals import joblib

joblib.dump(regressor, 'model.pkl')
```

In [ ]:
```

```



Lampiran 4 *Deploy Flask Script* (*index.html*)

```
<!DOCTYPE html>
<html lang="en">
<head>
      <title>House Price Predictor</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1">
<!--
===============================================================
============================-->
      <link rel="icon" type="image/png" href="{{
url_for('static',filename='images/icons/car_icon.png') }}" />
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/bootstrap/css/bootstrap.min.css')
}}" >
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='fonts/font-awesome-4.7.0/css/font-
awesome.min.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/animate/animate.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/css-
hamburgers/hamburgers.min.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/animsition/css/animsition.min.css
') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/select2/select2.min.css') }}">
<!--
===============================================================
============================-->
```

```
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/daterangepicker/daterangepicker.c
ss') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/util.css') }}">
      <link rel= "stylesheet" type= "text/css" href="{{
url_for('static',filename='css/main.css') }}">
<!--
===============================================================
============================-->

</head>
<body>


      <div class="container-contact100">
            <div class="wrap-contact100">
                  <form id="form_pred" class="contact100-form
validate-form" method="post" action="/api">
                        <span class="contact100-form-title">
                              House Price Prediction
                        </span>

                        <div class="wrap-input100 validate-input"
data-validate="Area is required">
                              <span class="label-
input100">Area</span>
                              <input class="input100" type="text"
id="Area" name="Area" placeholder="145364">
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 validate-input"
data-validate="BHK  is required">
                              <span class="label-input100">BHK
</span>
                              <input class="input100" type="text"
id="BHK" name="BHK" placeholder="2">
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 validate-input"
data-validate="Bathroom is required">
                              <span class="label-
input100">Bathroom</span>
                              <input class="input100" type="text"
id="Bathroom" name="Bathroom" placeholder="1">
                              <span class="focus-input100"></span>
                        </div>
```

```
                        <div class="wrap-input100 validate-input"
data-validate="Parking is required">
                              <span class="label-
input100">Parking</span>
                              <input class="input100" type="text"
id="Parking" name="Parking" placeholder="2">
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 validate-input"
data-validate="Per_Sqft is required">
                              <span class="label-
input100">Per_Sqft</span>
                              <input class="input100" type="text"
id="Per_Sqft" name="Per_Sqft" placeholder="6667">
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 input100-select">
                              <span class="label-
input100">Furnishing</span>
                              <div>
                                    <select class="selection-2"
id="Furnishing" name="Furnishing">
                                          <option value="Semi-
Furnished">Semi-Furnished </option>
                                          <option
value="Furnished">Furnished </option>
                                          <option
value="Unfurnished">Unfurnished</option>
                                    </select>
                              </div>
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 input100-select">
                              <span class="label-
input100">Status</span>
                              <div>
                                    <select class="selection-2"
id="Status" name="Status">
                                          <option
value="Readytomove" selected="">Readytomove </option>
                                          <option
value="Almostready">Almostready</option>
                                    </select>
                              </div>
                              <span class="focus-input100"></span>
                        </div>

                        <div class="wrap-input100 input100-select">
```

```
                                    <span class="label-
input100">Type</span>
                                    <div>
                                          <select class="selection-2"
id="Type" name="Type">
                                                <option value="Apartment"
selected="">Apartment </option>
                                                <option
value="Builder_Floor">Builder_Floor</option>
                                          </select>
                                    </div>
                                    <span class="focus-input100"></span>
                              </div>

                              <div class="wrap-input100 input100-select">
                                    <span class="label-
input100">Transaction</span>
                                    <div>
                                          <select class="selection-2"
id="Transaction" name="Transaction">
                                                <option
value="New_Property" selected="">New_Property </option>
                                                <option
value="Resale">Resale</option>
                                          </select>
                                    </div>
                                    <span class="focus-input100"></span>
                              </div>
                              <div class="container-contact100-form-btn">
                                    <div class="wrap-contact100-form-
btn">
                                          <div class="contact100-form-
bgbtn"></div>
                                          <button id="predict"
class="contact100-form-btn">
                                                <span>
                                                      How much does it
cost !!
                                                      <i class="fa fa-
long-arrow-right m-l-7" aria-hidden="true"></i>
                                                </span>
                                          </button>
                                    </div>
                              </div>
                        </form>
                  </div>
            </div>


            <div id="dropDownSelect1"></div>
```

```
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/jquery/jquery-3.2.1.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/animsition/js/animsition.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/bootstrap/js/popper.js')
}}"></script>
      <script src="{{
url_for('static',filename='vendor/bootstrap/js/bootstrap.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/select2/select2.min.js')
}}"></script>
      <script>
            $(".selection-2").select2({
                  minimumResultsForSearch: 20,
                  dropdownParent: $('#dropDownSelect1')
            });
      </script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/daterangepicker/moment.min.js')
}}"></script>
      <script src="{{
url_for('static',filename='vendor/daterangepicker/daterangepicker.j
s') }}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/countdowntime/countdowntime.js')
}}"></script>
<!--
===============================================================================
============================-->
```

```
      <script src="{{ url_for('static',filename='js/main.js')
}}"></script>
      <!-- include sweetAlert plugin -->
      <script src="{{
url_for('static',filename='js/sweetalert2.all.js') }}"></script>
      <script type="text/javascript">
            $(function () {

              $("button#predict").click(function(e){
                  e.preventDefault();
                  /*Get for variabes*/
                  var Area = $("#Area").val(), BHK =
$("BHK").val(), Bathroom = $("#Bathroom").val();
                  var Parking = $("#Parking").val(), Per_Sqft =
$("#Per_Sqft").val(), Furnishing = $("#Furnishing").val();
                  var  Status = $("Status").val(), Type =
$("#Type").val(), Transaction = $("#Transaction").val();
                        /*create the JSON object*/
                  var data = {"Area":Area, "BHK":BHK,
"Bathroom":Bathroom, "Parking":Parking, "Per_Sqft":Per_Sqft,
"Furnishing":Furnishing, "Status":Status, "Type":Type,
"Transaction":Transaction}
                  /*send the ajax request*/
                  $.ajax({
                        method : "POST",
                        url : window.location.href + 'api',
                        data : $('form').serialize(),
                        success : function(result){
                              var json_result = JSON.parse(result);
                              var Price = json_result['Price'];
                              swal('Estimated price is '+Price+'
DH', '','success')
                        },
                        error : function(){
                              console.log("error")
                        }
                  })
              })

            });
      </script>
</body>
</html>
```

Lampiran 5 *Deploy Flask Script* (*result..html*)

```
<!DOCTYPE html>
<html lang="en">
<head>
      <title>House Price Predictor</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1">
<!--
===============================================================
============================-->
      <link rel="icon" type="image/png" href="{{
url_for('static',filename='images/icons/car_icon.png') }}" />
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/bootstrap/css/bootstrap.min.css')
}}" >
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='fonts/font-awesome-4.7.0/css/font-
awesome.min.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/animate/animate.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/css-
hamburgers/hamburgers.min.css') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/animsition/css/animsition.min.css'
) }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/select2/select2.min.css') }}">
<!--
===============================================================
============================-->
```

```
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='vendor/daterangepicker/daterangepicker.cs
s') }}">
<!--
===============================================================
============================-->
      <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/util.css') }}">
      <link rel= "stylesheet" type= "text/css" href="{{
url_for('static',filename='css/main.css') }}">
<!--
===============================================================
============================-->

</head>
<body>


      <div class="container-contact100">
            <div class="wrap-contact100">

                        <span class="contact100-form-title">
                        {{prediction_text}}
                        </span>


                              </div>
                              <div class="container-contact100-form-
btn">
                              <div class="wrap-contact100-form-btn">
                                    <div class="contact100-form-
bgbtn"></div>
                                    <button  href="/"
class="contact100-form-btn">
                                          <span>
                                                Home !!

                                          </span>
                                    </button>
                              </div>
                        </div>
                        </div>

            </div>
      </div>



      <div id="dropDownSelect1"></div>
```

```
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/jquery/jquery-3.2.1.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/animsition/js/animsition.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/bootstrap/js/popper.js')
}}"></script>
      <script src="{{
url_for('static',filename='vendor/bootstrap/js/bootstrap.min.js')
}}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/select2/select2.min.js')
}}"></script>
      <script>
            $(".selection-2").select2({
                  minimumResultsForSearch: 20,
                  dropdownParent: $('#dropDownSelect1')
            });
      </script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/daterangepicker/moment.min.js')
}}"></script>
      <script src="{{
url_for('static',filename='vendor/daterangepicker/daterangepicker.js
') }}"></script>
<!--
===============================================================================
============================-->
      <script src="{{
url_for('static',filename='vendor/countdowntime/countdowntime.js')
}}"></script>
<!--
===============================================================================
============================-->
```

```
      <script src="{{ url_for('static',filename='js/main.js')
}}"></script>
      <!-- include sweetAlert plugin -->
      <script src="{{
url_for('static',filename='js/sweetalert2.all.js') }}"></script>
      <script type="text/javascript">
            $(function () {

              $("button#predict").click(function(e){
                  e.preventDefault();
                  /*Get for variabes*/
                  var Area = $("#Area").val(), BHK = $("BHK").val(),
Bathroom = $("#Bathroom").val();
                  var Parking = $("#Parking").val(), Per_Sqft =
$("#Per_Sqft").val(), Furnishing = $("#Furnishing").val();
                  var  Status= $("Status").val(), Type =
$("#Type").val(), Transaction = $("#Transaction").val();
                        /*create the JSON object*/
                  var data = {"Area
                        :Area, "BHK":BHK, "Bathroom":Bathroom,
"Parking":Parking, "Per_Sqft":Per_Sqft, "Furnishing":Furnishing,
"Status":Status, "Type":Type, "Transaction":Transaction}
                  /*send the ajax request*/
                  $.ajax({
                        method : "POST",
                        url : window.location.href + 'api',
                        data : $('form').serialize(),
                        success : function(result){
                              var json_result = JSON.parse(result);
                              var price = json_result['Price'];
                              swal('Estimated price is '+Price+'
DH', '','success')
                        },
                        error : function(){
                              console.log("error")
                        }
                  })
              })

            });
      </script>
</body>
</html>
```

Lampiran 6 *Deploy Flask Script* (*app.py*)

```
from flask import Flask, abort, jsonify, request, render_template
from sklearn.externals import joblib
import numpy as np
import json

regressor = joblib.load('model.pkl')

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

#@app.route('/api', methods=['POST'])
#def make_prediction():
    #data = request.get_json(force=True)
##convert our json to a numpy array
   # one_hot_data = input_to_one_hot(data)
   # predict_request = regressor.predict([one_hot_data])
   # output = [predict_request[0]]
   # print(data)
   # return jsonify(results=output)
def input_to_one_hot(a):
    # initialize the target vector with zero values
    enc_input = np.zeros(14)
    # set the numerical input as they are
    enc_input[0] = a['Area']
    enc_input[1] = a['BHK']
    enc_input[2] = a['Bathroom']
    enc_input[3] = a['Parking']
    enc_input[4] = a['Per_Sqft']
    ##################### Mark ########################
    # get the array of marks categories
    furnishing = ['Semi-Furnished', 'Furnished', 'Unfurnished']
    cols = ['Area','BHK','Bathroom','Parking','Per_Sqft'
,'Furnishing_Furnished',
    'Furnishing_Semi-
Furnished','Furnishing_Unfurnished','Status_Almostready',

'Status_Readytomove','Transaction_New_Property','Transaction_Resale
','Type_Apartment',
    'Type_Builder_Floor']

    redefinded_user_input = 'Furnishing_'+a['Furnishing']
    # search for the index in columns name list
    furnishing_column_index = cols.index(redefinded_user_input)
    #print(mark_column_index)
    # fullfill the found index with 1
    enc_input[furnishing_column_index] = 1
```

```
    ##################### Fuel Type ####################
    # get the array of fuel type
    status = ['Readytomove', 'Almostready']
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Status_'+a['Status']
    # search for the index in columns name list
    status_column_index = cols.index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[status_column_index] = 1
    # get the array of fuel type
    type_types = ['Apartment', 'Builder_Floor']
    # redefine the the user inout to match the column name
    redefinded_user_input = 'Type_'+a['Type']
    # search for the index in columns name list
    type_column_index = cols.index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[type_column_index] = 1
    transaction_types = ['New_Property', 'Resale']
    redefinded_user_input = 'Transaction_'+a['Transaction']
    # search for the index in columns name list
    transaction_column_index = cols.index(redefinded_user_input)
    # fullfill the found index with 1
    enc_input[transaction_column_index] = 1
    return enc_input
@app.route('/api',methods=['POST'])
def get_delay():
    result=request.form
    Area = result['Area']
    BHK = result['BHK']
    Bathroom = result['Bathroom']
    Parking = result['Parking']
    Per_Sqft = result['Per_Sqft']
    Furnishing = result['Furnishing']
    Status = result['Status']
    Type = result['Type']
    Transaction = result['Transaction']

    user_input = {'Area':Area, 'BHK':BHK, 'Bathroom':Bathroom,
'Parking':Parking, 'Per_Sqft':Per_Sqft, 'Furnishing':Furnishing,
'Status':Status, 'Type':Type, 'Transaction':Transaction}

    print(user_input)
    b = input_to_one_hot(user_input)
    price_pred = regressor.predict([b])[0]
    price_pred = round(price_pred, 2)
    #return json.dumps({'Price':price_pred});
    #return render_template('index.html',prediction=price_pred)
    return render_template('result.html',prediction_text='Harga
Rumah Diprediksi = Rp{}'.format(price_pred))

if __name__ == '__main__':
    app.run(port=8080, debug=True)
```

Lampiran 7 *Deploy* Aplikasi Menggunakan Heroku

```
C:\Users\HP\Model Delhi\Nyoba>pip install virtualenv

C:\Users\HP\Model Delhi\Nyoba>virtualenv venv

C:\Users\HP\Model Delhi\Nyoba>cd venv

C:\Users\HP\Model Delhi\Nyoba\venv>cd Scripts

C:\Users\HP\Model Delhi\Nyoba\venv\Scripts>.\activate

(venv) C:\Users\HP\Model Delhi\Nyoba\Deploy1\venv\Scripts>cd..

(venv) C:\Users\HP\Model Delhi\Nyoba\venv>cd..

(venv) C:\Users\HP\Model Delhi\Nyoba>pip install flask gunicorn
jinja2 numpy sklearn

(venv) C:\Users\HP\Model Delhi\Nyoba>pip freeze requirements

(venv) C:\Users\HP\Model Delhi\Nyoba>pip freeze> requirements.txt

#membuka halaman command prompt baru
C:\Users\HP\Model Delhi\Nyoba>git init

C:\Users\HP\Model Delhi\Nyoba>git add .

C:\Users\HP\Model Delhi\Nyoba>git status

C:\Users\HP\Model Delhi\Nyoba>git commit -m "langkah pertama"

C:\Users\HP\Model Delhi\Nyoba>heroku login

C:\Users\HP\Model Delhi\Nyoba>heroku create housesalespredict

C:\Users\HP\Model Delhi\Nyoba>git remote –v

C:\Users\HP\Model Delhi\Nyoba>git push heroku master
```