# MACHINE LEARNING WITH PYTHON FOR BEGINNER

## Chapter 1

### Pengenalan Machine Learning

#### Pendahuluan

Akhirnya aku ada waktu untuk istirahat karena tinggal menunggu review dari Andra. Aku baru mau turun untuk jajan tapi terhenti saat menyadari kesibukan Senja di sampingku, “Lagi apa sih, Nja?”

“Oh, ini lagi siapin bahan tentang machine learning-nya Python. Kamu mau belajar bahannya?” Mumpung lagi lowong, enggak ada salahnya nambah ilmu baru. Aku mengangguk semangat.

#### Machine Learning? Apa Itu?

Pagi-pagi sekali, sudah kudapati Senja duduk di bangku di sebelah mejaku. Nyatanya, membawa modul pembelajaran baru. Apa ya?

“Aksara, hari ini kita mulai bateri baru seputar machine learning, karena proyek kamu kedepannya akan banyak berkaitan dengan machine learning. Pernah dengar sebelumnya?”tanya Senja padaku. “Sering dengar sih, tapi gak tau itu apa. Gak ngerti juga fungsinya apa dalam pengolahan data,” jawabku jujur.

“Oke, aku kasih penjelasan dikit yah sebelum aku kasih modul dan kita praktik. Machine Learning adalah teknik dimana komputer dapat mengekstraksi atau mempelajari pola dari suatu data, kemudian dengan pola yang telah dipelajari dari data historis, komputer mampu mengenali dan memprediksi trend, hasil atau kejadian di masa mendatang atau dari observasi baru tanpa perlu diprogram secara eksplisit,” jelas Senja sembari menyeruput kopi hangatnya.

“Contohnya gimana yah, Nja?”

“Sebagai contoh, pernah buka folder spam di email kamu? Di sana ada banyak email promosi dan sebagainya yang dikirim secara acak ke email kita padahal tidak kita inginkan. Lalu bagaimana, perusahaan seperti Google, Yahoo, Hotmail, Microsoft dan perusahaan penyedia lainnya secara otomatis memfilter pesan spam ini? Tentunya dengan machine learning.”

Aku mengangguk. Tampaknya jadi lebih sederhana dengan contoh kasus seperti ini.

“Selain machine learning, apakah ada cara lain utnuk memfilter spam seperti itu?”

“Bisa saja dengan cara lain, tetapi akan ada begitu banyak rules dan logic yang harus dibuat sehingga tidak fleksibel, serta kompleksitasnya akan sulit untuk dikelola. Terutama untuk data dan variabel yang cukup banyak serta berubah sewaktu-waktu, sangat sulit untuk dimodelkan secara kuantitatif atau matematis.”

Bisa kusimpulkan jika machine learning ini sangat mempermudah sistem perusahaan, pikirku. Aku jadi penasaran satu hal, “Senja, apakah machine learning bisa diimplementasikan selain pada email?”

“Tentu saja, Selain mengenali email sebagai spam atau ukan spam ada banyak contoh penggunaan machine learning lainnya, seperti memprediksi harga saham, pengenalan wajah (face recognition), mengenali tulisan tangan, mendeteksi fraud/scam kartu kredit, memprediksi cuaca, dan memprediksi permintaan barang,” tukas Senja.

Tampaknya ini akan seru. Siap untuk belajar lebih jauh!

#### Terminologi Machine Learning

“Oke, berarti kamu udah paham ya apa itu machine learning, Aksara? Karena kita butuh paham dasarnya dulu sebelum maju. Selanjutnya saya akan menjelaskan terkait istilah-istilah yang sering dipakai di machine learning agar kamu tidak bingung dalam project nantinya.” “Sudah paham, Nja. Lanjut aja.”  
  
“Oke, saya buka rangkumannya yah, kamu bisa sambil baca dan aku menerangkan sedikit,” ujar Senja. Aku memerhatikan presentasi berisi rangkuman machine learning yang ditampilkannya di layar laptop:

Dalam pembuatan model machine learning tentunya dibutuhkan data. Sekumpulan data yang digunakan dalam machine learning disebut **DATASET**, yang kemudian dibagi/di-split menjadi **training dataset** dan **test dataset**.

**TRAINING DATASET** digunakan untuk membuat/melatih model machine learning, sedangkan **TEST DATASET** digunakan untuk menguji performa/akurasi dari model yang telah dilatih/di-training.

Teknik atau pendekatan yang digunakan untuk membangun model disebut **ALGORITHM** seperti Decision Tree, K-NN, Linear Regression, Random Forest, dsb. dan output atau hasil dari proses melatih algorithm dengan suatu dataset disebut **MODEL**.

Umumnya dataset disajikan dalam bentuk tabel yang terdiri dari baris dan kolom. Bagian Kolom adalah **FEATURE** atau **VARIABEL** data yang dianalisa, sedangkan bagian baris adalah **DATA POINT/OBSERVATION/EXAMPLE**.

Hal yang menjadi target prediksi atau hal yang akan diprediksi dalam machine learning disebut **LABEL/CLASS/TARGET.** Dalam statistika/matematika, **LABEL/CLASS/TARGET** ini dinamakan dengan **Dependent Variabel**, dan **FEATURE** adalah **Independent Variabel**.

#### Supervised and Unsupervised Learning

“Nja, aku mau tanya deh. Aku pernah mendengar tentang supervised dan unsupervised learning di machine learning. Apa sih itu? Bedanya apa, Nja?”

”Jadi, Machine Learning itu terbagi menjadi 2 tipe yaitu supervised dan unsupervised Learning. Jika LABEL/CLASS dari dataset sudah diketahui maka dikategorikan sebagai supervised learning, dan jika Label belum diketahui maka dikategorikan sebagai unsupervised learning,” jelas Senja.

Tapi, aku masih menyimpan pertanyaan,”Kalau kasusnya yang email tadi, masuknya ke mana yah?”

“Mengenali email sebagai spam atau bukan spam tergolong sebagai supervised learning, karena kita mengolah dataset yang berisi data point yang telah diberi LABEL ”spam” dan “not spam”. Sedangkan jika kita ingin mengelompokkan customer ke dalam beberapa segmentasi berdasarkan variabel-variabel seperti pendapatan, umur, hobi, atau jenis pekerjaan, maka tergolong sebagai unsupervised learning,” tukas Senja.

Baiklah, kali ini semuanya jadi lebih jelas. Lanjut!

#### Pilih Algorithm yang Mana?

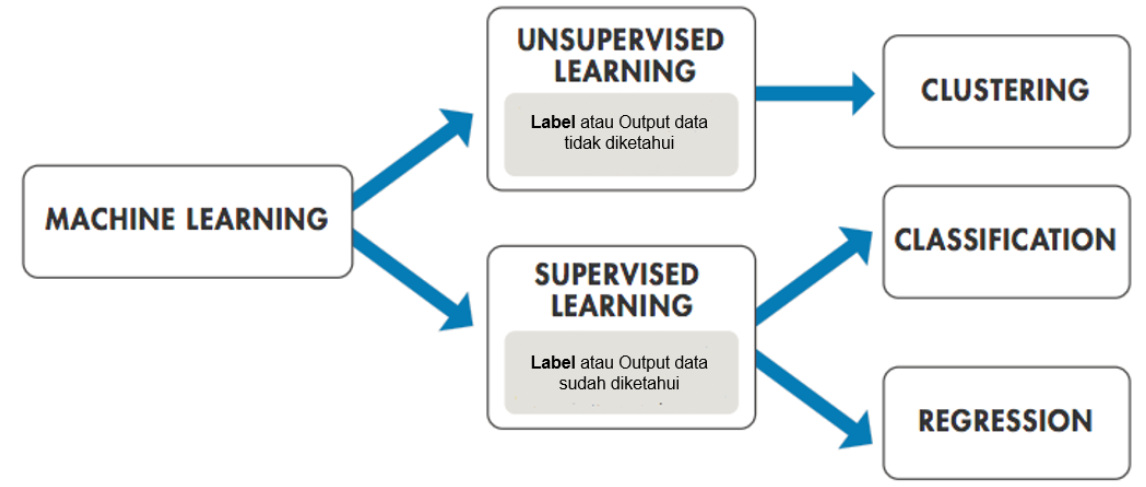
“Nja, aku sudah tahu bahwa proyek aku adalah tipe supervised learning, terus gimana cara menentukan algorithm yang cocok?” tanyaku.

“Satu lagi, Nja. Kalau sudah paham bahwa problem aku adalah tipe unsupervised learning, lalu apa algorithm yang tepat untuk kasus seperti ini?” tambahku lagi.

“Oke, kubahas satu persatu ya, Aksara. Jadi begini,  penting untuk diingat bahwa tidak ada ML algorithm yang cocok atau fit untuk diaplikasikan di semua problem. Oleh karena itu, proses ini terkadang memerlukan trial & error seperti research, bahkan experienced data scientist pun tidak akan tahu apakah algorithm itu akan tepat atau tidak jika tidak mencoba. Biasanya, data scientist akan mencoba beberapa algorithm dan membandingkan performansi dari algorithm - algorithm tersebut. Algorithm dengan performansi yang paling baiklah yang dipilih sebagai model,” jelas Senja.

“Jadi lebih banyak mencoba dan praktik ya untuk tahu yang tepat dan relevannya?”

“Iya, selain itu  untuk supervised learning, jika LABEL dari dataset kalian berupa numerik atau kontinu variabel seperti harga, dan  jumlah penjualan, kita memilih metode REGRESI dan jika bukan numerik atau diskrit maka digunakan metode KLASIFIKASI. Untuk unsupervised learning, seperti segmentasi customer, kita menggunakan metode CLUSTERING,” sahut Senja.



## Chapter 2

### Eksplorasi Data & Data Pre-processing

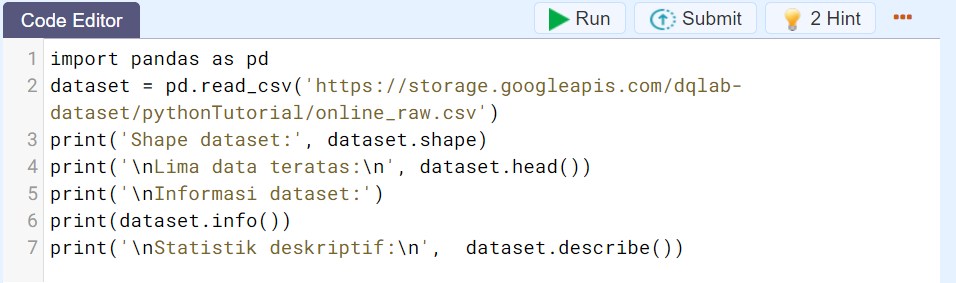
#### Eksplorasi Data: Memahami Data dengan Statistik - Part 1

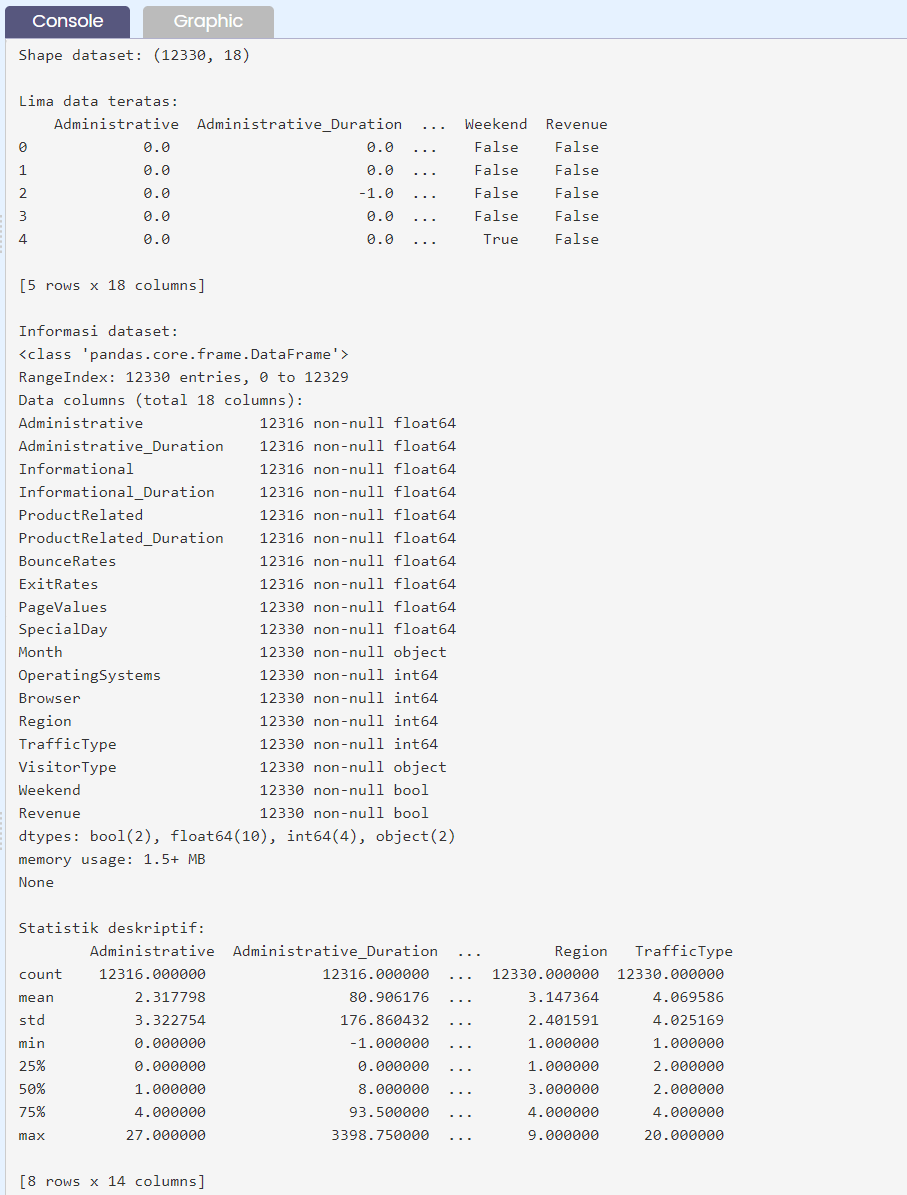
“Selanjutnya saya akan menjelaskan secara singkat tentang tahapan-tahapan dalam pembuatan model machine learning. Membuat model machine learning tidak serta-merta langsung modelling, ada tahapan sebelumnya yang penting untuk dilakukan sehingga kita menghasilkan model yang baik. Untuk penjelasan ini, kalian akan mempraktekkan langsung ya. Kita akan memanfaatkan **Pandas** library. Pandas cukup powerful untuk digunakan dalam menganalisa, memanipulasi dan membersihkan data. Siap, Aksara?”

Aku mengangguk, siap dengan laptopku.

“Oke, Pertama- tama,  kita check dimensi data kita terlebih dahulu. Aksara, silahkan load datanya dan gunakan **.shape**, .**head()**, **.info()**, dan **.describe()** untuk mengeksplorasi dataset secara berurut. Dataset ini adalah data pembeli online yang mengunjungi website dari suatu e-commerce selama setahun, yaitu **'https://storage.googleapis.com/dqlab-dataset/pythonTutorial/online\_raw.csv'**,” perintah Senja.

Jika kode telah dengan benar ditulis di code editor dan kemudian dijalankan melalui  maka diperoleh hasil seperti berikut:





“Nah, dengan mengetahui dimensi data yaitu jumlah baris dan kolom, kita bisa mengetahui apakah data kita terlalu banyak atau justru sangat sedikit. Jika data terlalu banyak, waktu melatih model akan lebih lama, sedangkan jika data terlalu sedikit, performansi model yang kita hasilkan mungkin tidak cukup bagus, karena tidak mampu mengenali pola dengan baik. Sudah lebih paham sekarang?”

Aku mengacungkan jempol. Kalau dibareng praktik, memang jadinya lebih jelas.

***Perhatian***:

Jika nanti mendapati DeprecationWarning di console seperti pesan berikut ini

/usr/local/lib/python3.5/site-packages/sklearn/ensemble/weight\_boosting.py:29: DeprecationWarning: numpy.core.umath\_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.  
from numpy.core.umath\_tests import inner1d

dapat diabaikan saja.

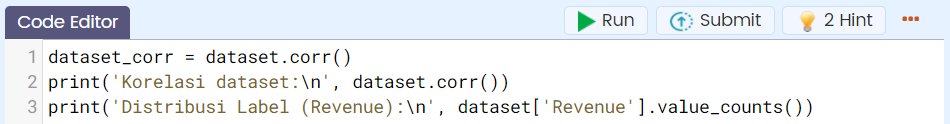
#### Eksplorasi Data: Memahami Data dengan Statistik - Part 2

Setelah selesai materi tadi, aku kembali diajak memahami eksplorasi data. Banyak sekali materi baru hari ini!

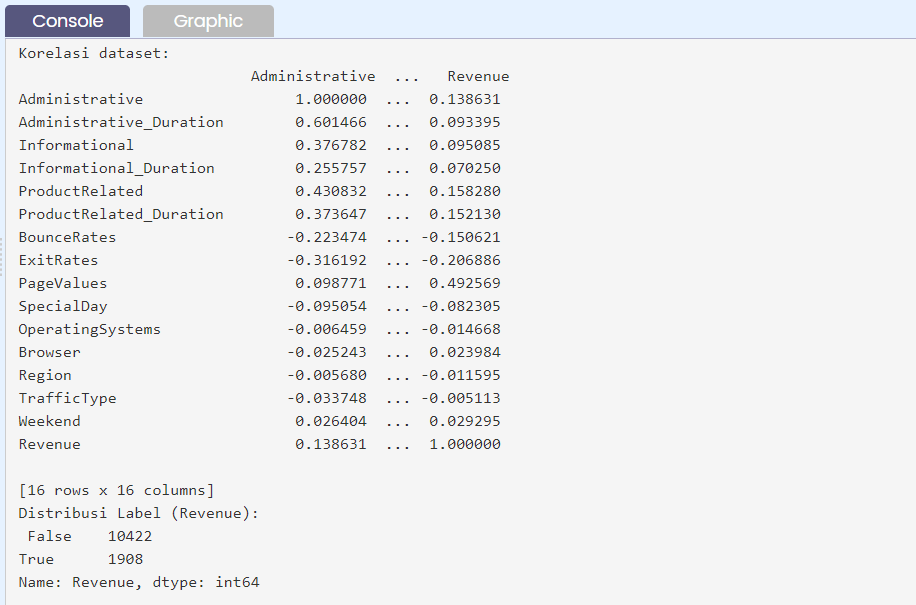
“Kita lanjut yah, Aksara. Data eksplorasi tidaklah cukup dengan mengetahui dimensi data dan statistical properties saja, tetapi kita juga perlu sedikit menggali tentang hubungan atau korelasi dari setiap feature, karena beberapa algorithm seperti linear regression dan logistic regression akan menghasilkan model dengan performansi yang buruk jika kita menggunakan feature/variabel saling dependensi atau berkorelasi kuat (multicollinearity). Jadi, jika kita sudah tahu bahwa data kita berkorelasi kuat, kita bisa menggunakan algorithm lain yang tidak sensitif terhadap hubungan korelasi dari feature/variabel seperti decision tree.”

”Aksara, coba sekarang lanjutkan eksplorasi data untuk melihat korelasi dan distribusi dataset.”

Aku segera berkutat dengan susunan kodenya:



dengan output berupa:



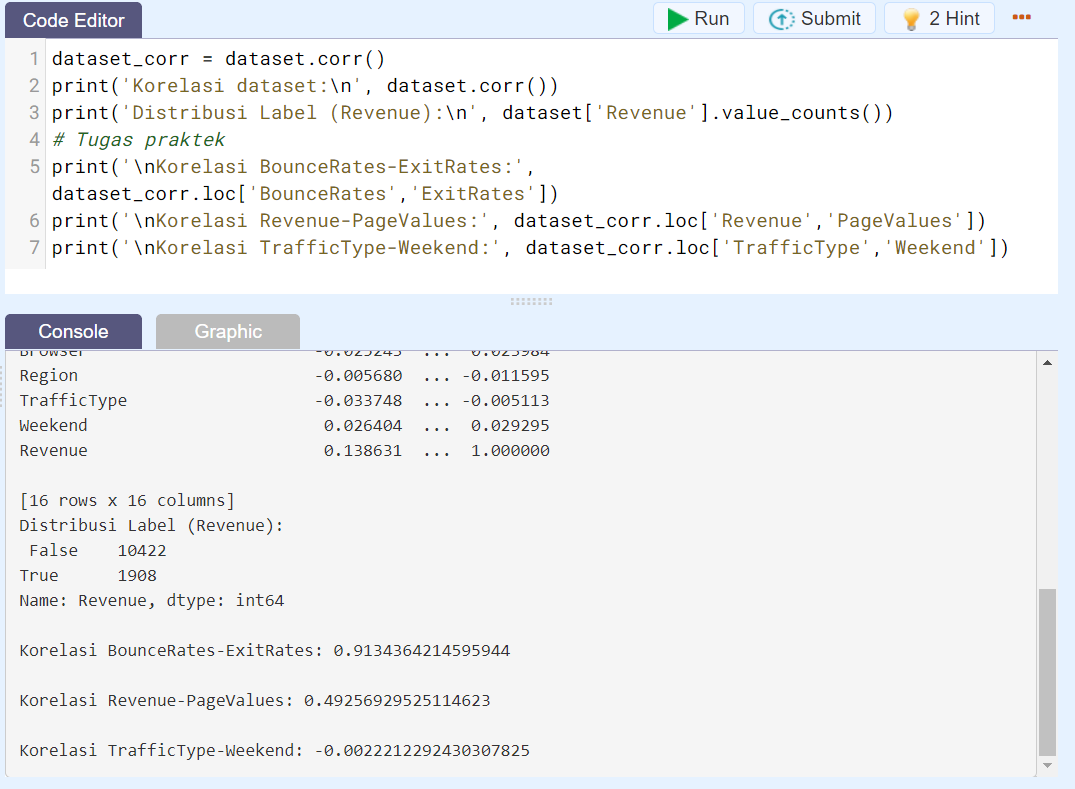
“Nja, kenapa mengetahui distribusi LABEL dari dataset itu penting?”

“Pertanyaan menarik, mengetahui distribusi label sangat penting untuk permasalahan klasifikasi, karena jika distribusi label sangat tidak seimbang (imbalanced class),  maka akan sulit bagi model untuk mempelajari pola dari LABEL yang sedikit dan hasilnya bisa misleading. Contohnya, kita memiliki 100 row data, 90 row adalah non fraud dan 10 row adalah fraud. Jika kita menggunakan data ini tanpa melakukan treatment khusus (handling imbalanced class), maka kemungkinan besar model kita akan cenderung mengenali observasi baru sebagai non-fraud, dan hal ini tentunya tidak diinginkan,” jelas Senja panjang lebar.

**Tugas Praktek:**

Sekarang coba inspeksi nilai korelasi dari fitur-fitur berikut pada dataset\_corr yang telah diberikan sebelumnya

* ExitRates dan BounceRates
* Revenue dan PageValues
* TrafficType dan Weekend



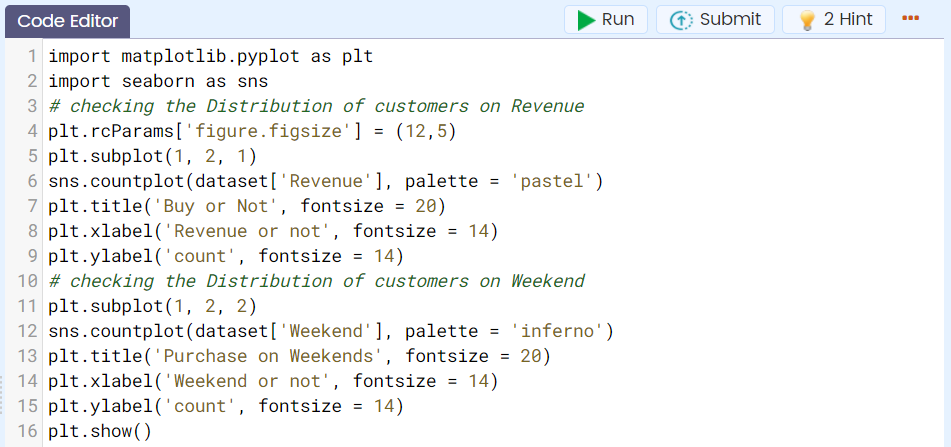
#### Eksplorasi Data: Memahami Data dengan Visual

“Aksara, satu lagi, dalam mengeksplorasi data, kita perlu untuk memahami data dengan visual.”

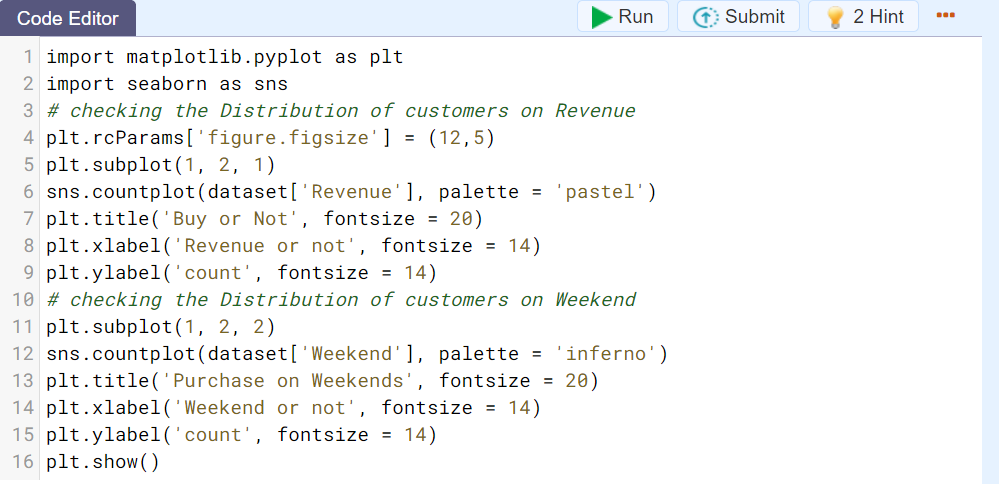
Aku tertarik, “Maksudnya?”

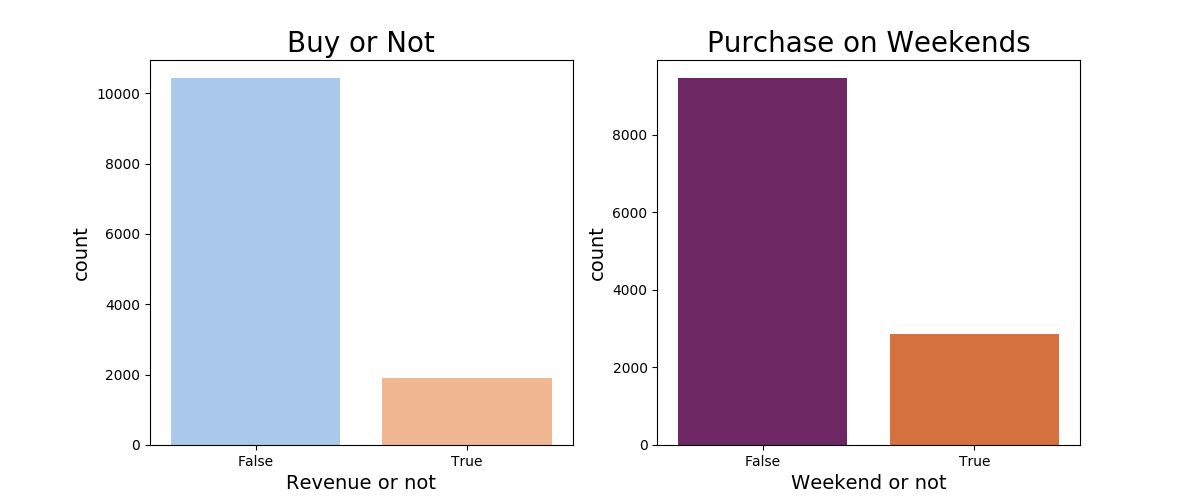
“Begini, selain dengan statistik, kita juga bisa melakukan eksplorasi data dalam bentuk visual. Dengan visualisasi kita dapat dengan mudah dan cepat dalam memahami data, bahkan dapat memberikan pemahaman yang lebih baik terkait hubungan setiap variabel/ features.

Misalnya kita ingin melihat distribusi label dalam bentuk visual, dan jumlah pembelian saat weekend. Kita dapat memanfaatkan **matplotlib** library untuk membuat chart yang menampilkan perbandingan jumlah yang membeli (1) dan tidak membeli (0), serta perbandingan jumlah pembelian saat weekend,” tambah Senja sembari menampilkan contoh kodenya:



Tuliskanlah kembali kode tersebut dengan benar dan kemudian dijalankan dengan , selanjutkan akan mendapati count\_plot berikut ini



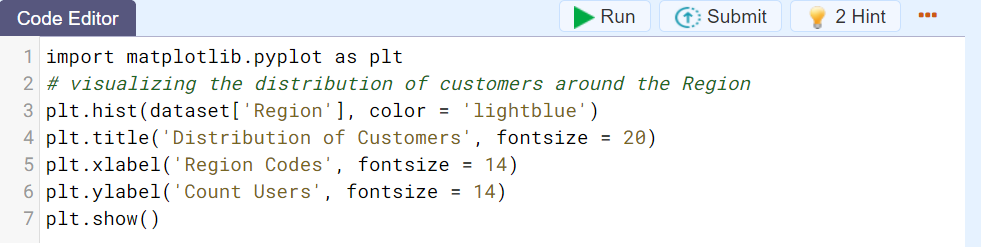


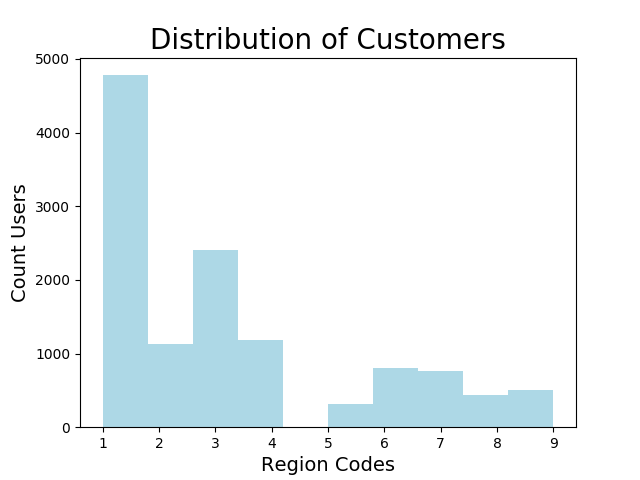
#### Tugas Praktek

Aku kemudian diminta Senja untuk membuat visualisasi berupa histogram yang menggambarkan jumlah customer untuk setiap Region.

Dalam membuat visualisasi ini aku akan menggunakan **dataset['region']** untuk membuat histogram, dan berikan judul 'Distribution of Customers' pada title, 'Region Codes' sebagai label axis-x dan 'Count Users' sebagai label axis-y.

Berikut tampilan dari histogram yang dimaksud setelah aku menjalankan kode dengan benar.





#### Data Pre-processing: Handling Missing Value - Part 1

“Kita sedang mengejar materi, Aksara. Semoga materi yang padat ini bisa kamu pahami dengan cepat ya. Kita akan move ke materi lanjutan,” ujar Senja padaku sembari membuka lembaran modul.

Aku ikut memperhatikan. Tampaknya akan ada proyek besar hingga aku harus cepat mempelajari soal machine learning ini.

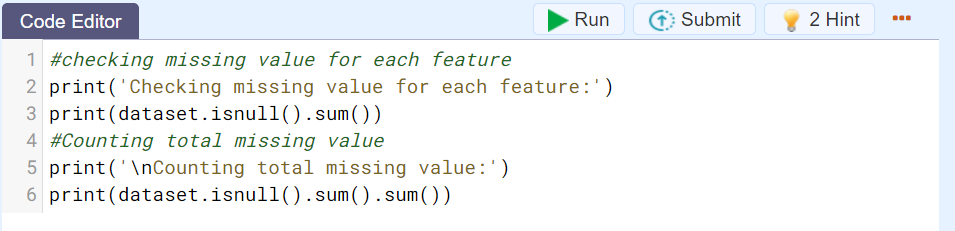
“Setelah kita melakukan eksplorasi data, kita akan melanjutkan ke tahap data pre-processing. Seperti yang saya jelaskan sebelumnya, raw data kita belum tentu bisa langsung digunakan untuk pemodelan. Jika kita memiliki banyak missing value, maka akan mengurangi performansi model dan juga beberapa algorithm machine learning tidak dapat memproses data dengan missing value. Oleh karena itu, kita perlu mengecek apakah terdapat missing value dalam data atau tidak. Jika tidak, maka kita tidak perlu melakukan apa-apa dan bisa melanjutkan ke tahap berikutnya. Jika ada, maka kita perlu melakukan treatment khusus untuk missing value ini,” jelas Senja.

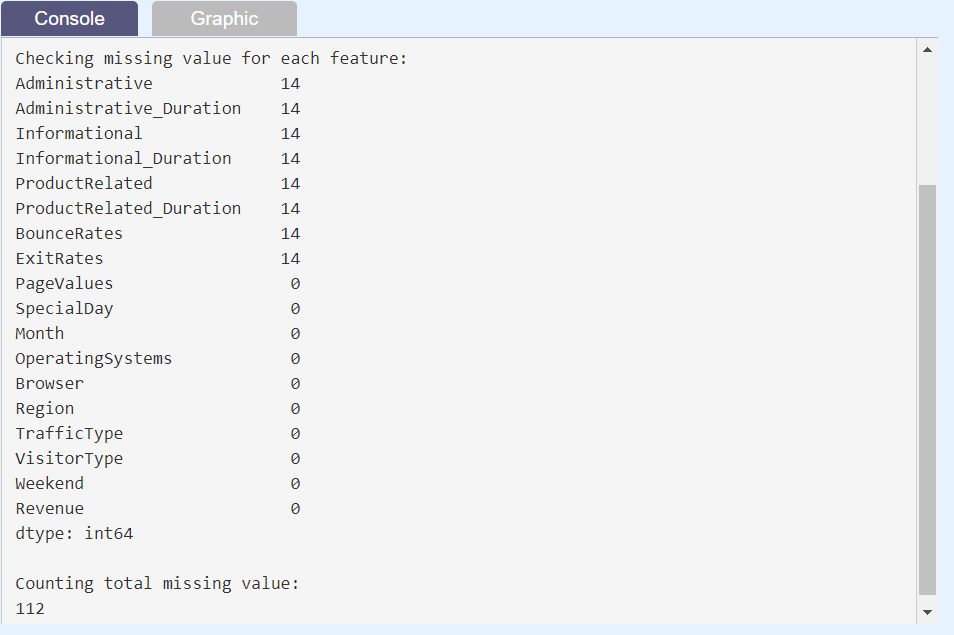
“Caranya bagaimana?”

“Pengecekan missing value dapat dilakukan dengan code berikut,” ujar Senja sambil mengarahkan layar laptopnya padaku.

Melalui layar dapat ku lihat Senja menggunakan metod .isnull pada dataset dan kemudian men-chaining-nya dengan method sum. Untuk jumlah keseluruhan missing value digunakan chaining method sum sekali lagi.

Jika dengan benar kodenya telah ditulis dan kemudian dijalankan dengan menekan , diperoleh tampilan di console seperti berikut ini





#### Data Pre-processing: Handling Missing Value - Part 2

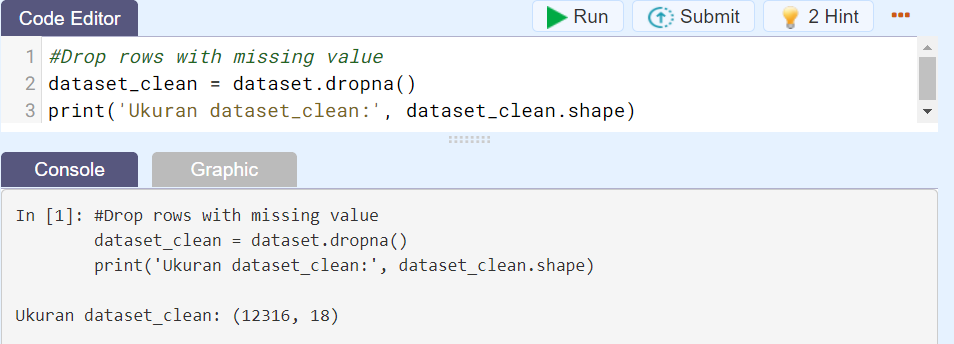
“Wah, ternyata ada missing value di dataset kita. Apakah data point-nya bisa dihapus saja?” tanyaku.

“Ada beberapa metode yang dapat kita lakukan untuk menangani missing value. Pilihanmu tepat, Aksara,  menghapus data adalah salah satunya. Tetapi, metode ini tidak dapat serta merta diimplementasikan. Kita juga perlu menganalisis penyebaran missing value, dan berapa persen jumlah missing value dalam data kita,” jawab Senja lengkap.  Tapi ada beberapa bagian yang masih membingungkan buatku.

“Aku masih agak bingung terutama penerapan metodenya,” sahutku jujur.

“Metode ini dapat diterapkan jika tidak banyak missing value dalam data, sehingga walaupun data point ini dihapus, kita masih memiliki sejumlah data yang cukup untuk melatih model Machine Learning. Tetapi jika kita memiliki banyak missing value dan tersebar di setiap variabel, maka metode menghapus missing value tidak dapat digunakan. Kita akan kehilangan sejumlah data yang tentunya mempengaruhi performansi model. Kita bisa menghapus data point yang memiliki missing value dengan fungsi **.dropna( )** dari pandas library. Fungsi dropna( ) akan menghapus data point atau baris yang memiliki missing value."

Silakan diketikkan di console untuk membuang baris yang memiliki missing value ini. Kemudian jalankan code dengan  sehingga diperoleh hasil seperti berikut ini

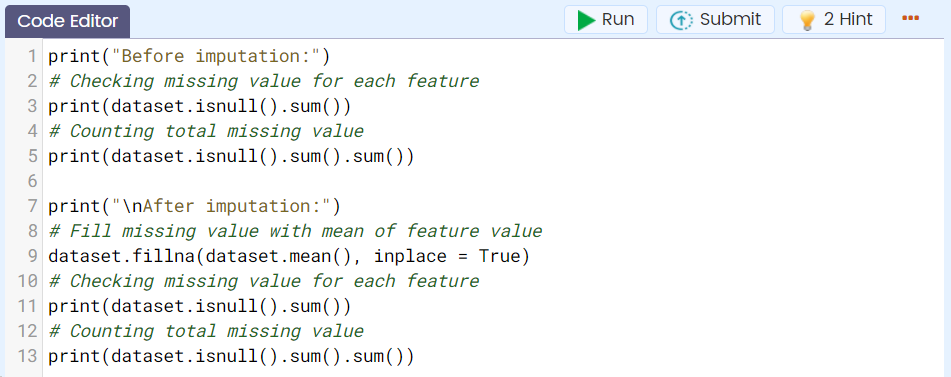


#### Data Pre-processing: Handling Missing Value - Part 3

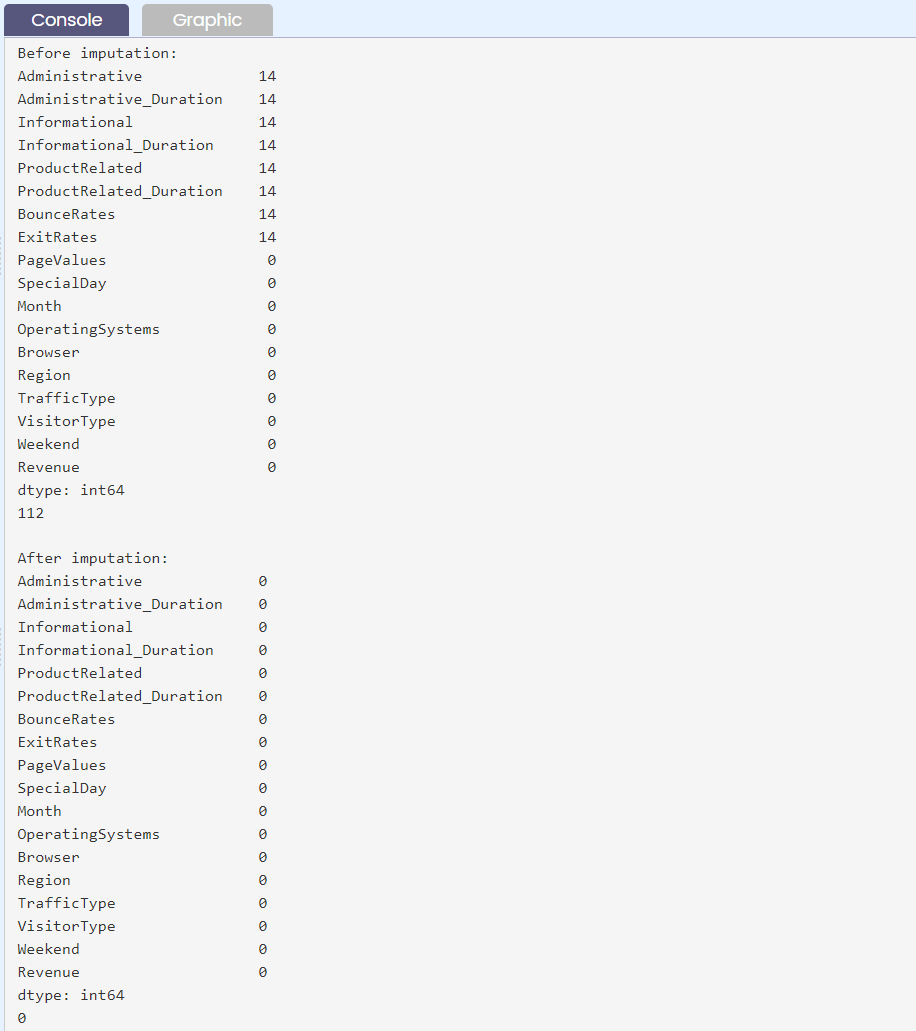
“Kalau tidak dihapus, ada metode lain yang bisa dipakai?”

“Kita bisa menggunakan metode **impute missing value**, yaitu mengisi record yang hilang ini dengan suatu nilai. Ada berbagai teknik dalam metode imputing, mulai dari yang paling sederhana yaitu mengisi missing value dengan nilai mean, median, modus, atau nilai konstan, sampai teknik paling advance yaitu dengan menggunakan nilai yang diestimasi oleh suatu predictive model. Untuk kasus ini, kita akan menggunakan imputing sederhana yaitu menggunakan nilai rataan atau mean,” jelas Senja.

Imputing missing value sangat mudah dilakukan di Python, cukup memanfaatkan fungsi **.fillna()** dan **.mean()** dari Pandas, seperti berikut:

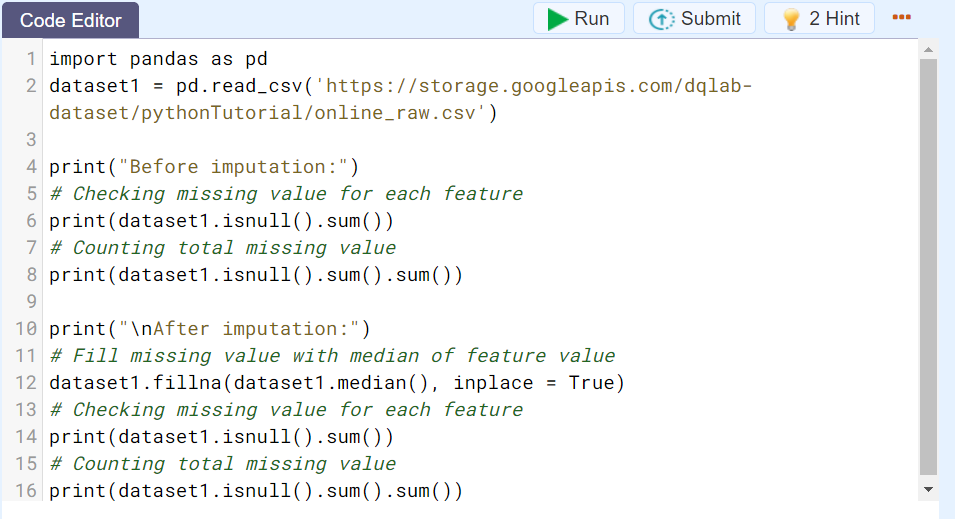


Jika dijalankan dengan  diperoleh

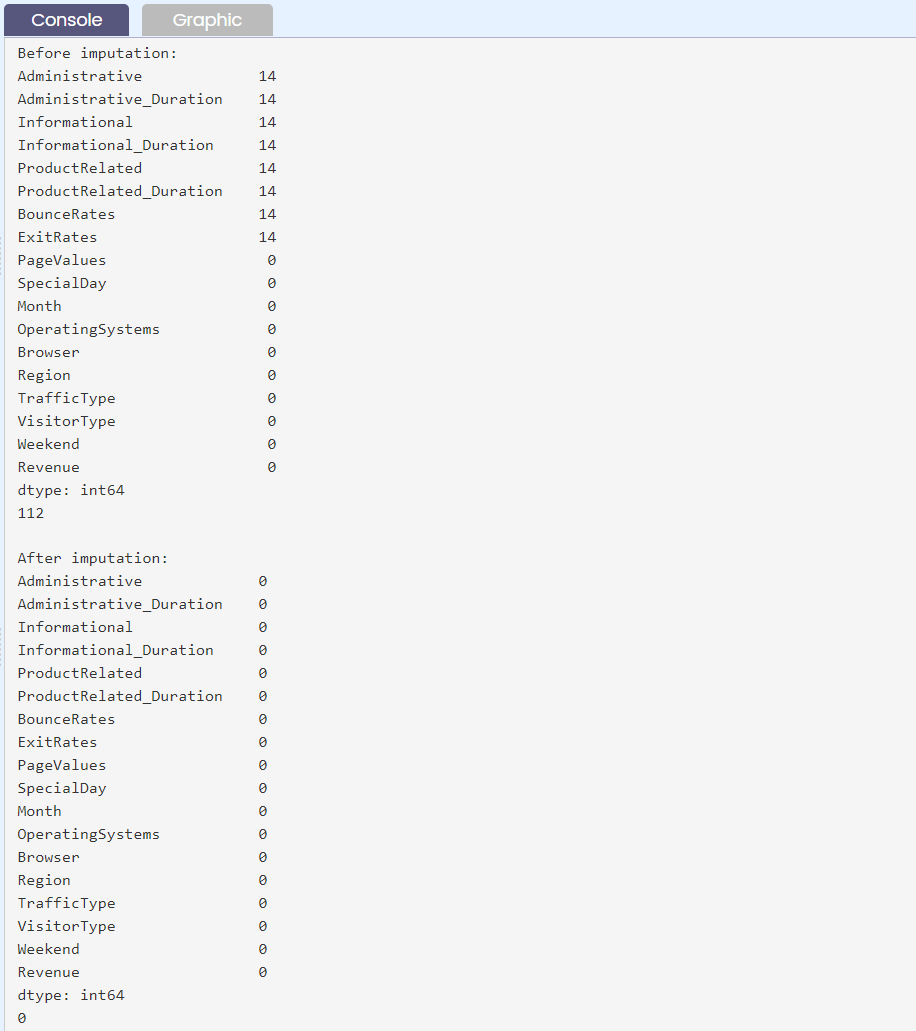


#### Tugas Praktek

Praktekkan metode imputing missing value dengan menggunakan nilai median.



Jika dengan benar ditulis dan dijalankan diperoleh tampilan output seperti berikut ini



#### Data Preprocessing: Scaling

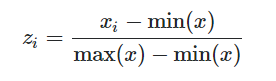
“Setelah berhasil menangani missing value, sekarang mari kita mempelajari tahapan preprocessing selanjutnya. Aksara, tolong tampilkan kembali 5 dataset teratas dan deskripsi statistik dari dataset. Coba perhatikan, rentang nilai dari setiap feature cukup bervariasi. Misalnya, ProductRelated\_Duration vs BounceRates.  ProductRelated\_Duration memiliki rentang nilai mulai dari 0 - 5000; sedangkan BounceRates rentang nilainya 0 - 1. Bisa kamu lihat?”

Aku mengangguk. Senja pun melanjutkan,

“Beberapa machine learning seperti K-NN dan gradient descent mengharuskan semua variabel memiliki rentang nilai yang sama, karena jika tidak sama, feature dengan rentang nilai terbesar misalnya ProductRelated\_Duration otomatis akan menjadi feature yang paling mendominasi dalam proses training/komputasi, sehingga model yang dihasilkan pun akan sangat bias. Oleh karena itu, sebelum memulai training model, kita terlebih dahulu perlu melakukan data rescaling ke dalam rentang 0 dan 1, sehingga semua feature berada dalam rentang nilai tersebut, yaitu nilai max = 1 dan nilai min = 0. Data rescaling ini dengan mudah dapat dilakukan di Python menggunakan **.MinMaxScaler( )** dari Scikit-Learn library.”

 “Kenapa ke range 0 - 1, tidak menggunakan range yang lain?” tanyaku penasaran.

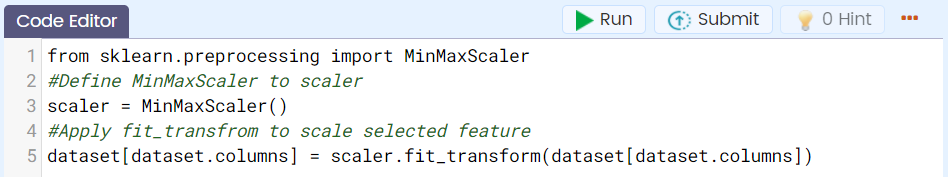
“Karena rumus dari rescaling adalah



dengan rumus ini, nilai max data akan menjadi 1 dan nilai min menjadi 0; dan nilai lainnya berada di rentang keduanya. Rumus ini tidak memungkinkan adanya rentang nilai selain 0 – 1,” ujar Senja sembari menggambarkan rumusnya pada buku catatanku.

#### Tugas Praktek

”Aksara, silakan praktikkan code untuk data scaling berikut,” pinta Senja.



“Kok enggak bisa ya. Jadi eror seperti ini, Nja,” kataku sambil memperlihatkan kode erornya:

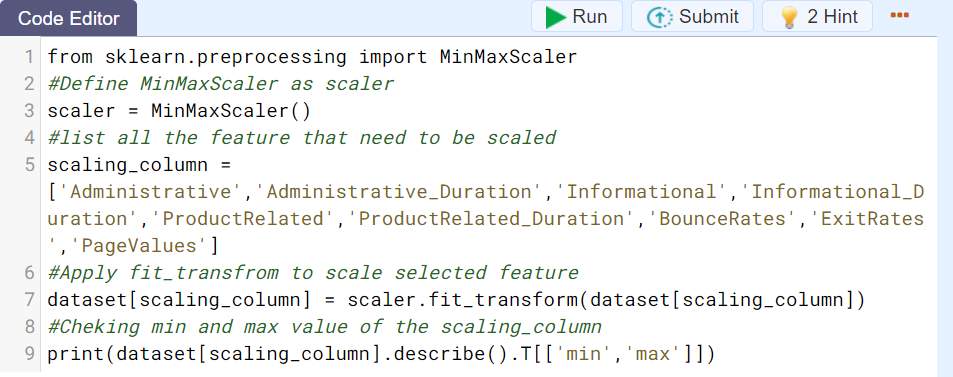


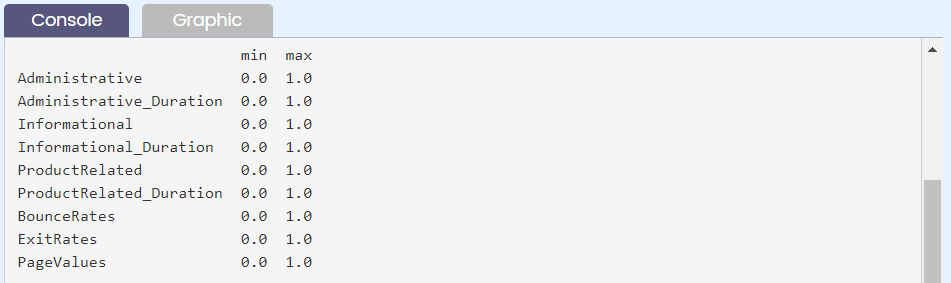
“Ya, code diatas merupakan basic code untuk proses scaling dengan asumsi bahwa semua feature adalah numerik. Tetapi, ketika menjalankan code tersebut untuk dataset online\_raw, pasti akan terjadi error. Proses scaling hanya bisa dilakukan untuk feature dengan tipe numerik, sedangkan dalam dataset online\_raw, terdapat feature dengan tipe string atau karakter dan categorical, seperti Month, VisitorType, Region. Oleh karena itu, kita tidak dapat langsung menggunakan code di atas, tetapi kita perlu terlebih dahulu menyeleksi feature - feature dari dataset yang bertipe numerik.”

Senja pun membagikan catatan berisi  langkah - langkah untuk proses scaling dengan dataset yang memiliki feature dengan tipe data yang berbeda:

1. Import **MinMaxScaler** dari **sklearn.preprocessing**
2. Deklarasikan fungsi **MinMaxScaler()** ke dalam variabel **scaler**
3. List semua feature yang akan di-scaling dan beri nama **scaling\_column** yaitu : ['Administrative', 'Administrative\_Duration', 'Informational', 'Informational\_Duration', 'ProductRelated', 'ProductRelated\_Duration', 'BounceRates', 'ExitRates', 'PageValues']
4. Berdasarkan contoh code yang dipraktekkan oleh Aksara, ganti **dataset.columns** dengan **scaling\_column**.

Jika dengan benar dituliskan dan kemudian dijalankan dengan  diperolehlah hasil seperti berikut ini





#### Data Pre-processing: Konversi string ke numerik

"Aksara, kita memiliki dua kolom yang bertipe object yang dinyatakan dalam tipe data str, yaitu kolom 'Month' dan 'VisitorType'. Karena setiap algoritma machine learning bekerja dengan menggunakan nilai numeris, maka kita perlu mengubah kolom dengan tipe pandas object atau str ini ke bertipe numeris. Untuk itu, kita list terlebih dahulu apa saja label unik di kedua kolom ini," jelas Senja. Lalu Senja pun mulai mempraktikkan sambil menunjukkannya padaku.

Label unik kolom 'Month':

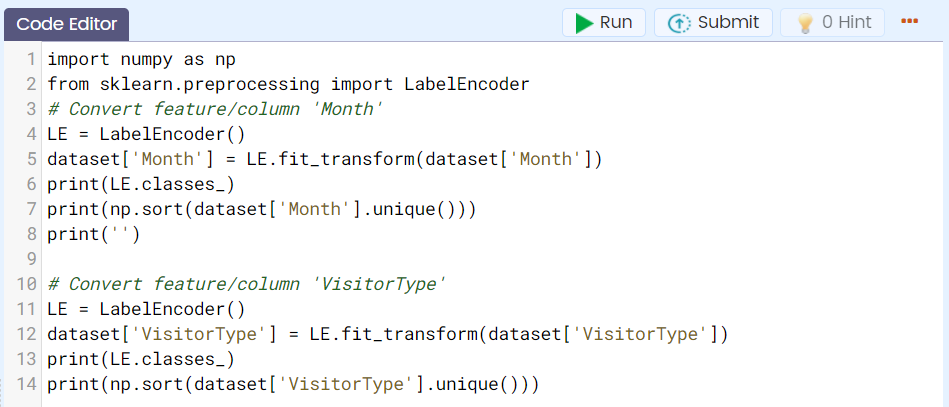
['Feb' 'Mar' 'May' 'Oct' 'June' 'Jul' 'Aug' 'Nov' 'Sep' 'Dec']

dan label unik kolom 'VisitorType':

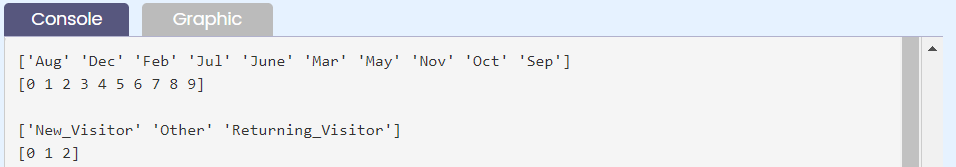
['Returning\_Visitor' 'New\_Visitor' 'Other']

"Nja, bagaimana ya cara menrubah tipe pandas object ini ke numerik (int, float) ya?," tanyaku penasaran.

"Ok, kita dapat menggunakan **LabelEncoder** dari **sklearn.preprocessing** untuk merubah kedua kolom ini seperti ini," Senja pun menjawab sambil praktik dengan kode yang diketikkan sebagai berikut



Jika dijalankan dengan menggunakan  diperoleh



"Bisa dilihat, kan, Aksara bahwa LabelEncoder akan mengurutkan label secara otomatis secara alfabetik, posisi/indeks dari setiap label ini digunakan sebagai nilai numeris konversi pandas objek ke numeris (dalam hal ini tipe data int). Dengan demikian kita telah membuat dataset kita menjadi dataset bernilai numeris seluruhnya yang siap digunakan untuk pemodelan dengan algoritma machine learning tertentu," tutup Senja.

## Chapter 3

### Pemodelan dengan Scikit-Learn

#### Pendahuluan

“Akhirnya, setelah data eksplorasi dan preprocessing, datasetnya sudah siap untuk digunakan dalam proses modelling. Kalau dipikir-pikir, preprocessing ini panjang juga dan ribet, aku lebih senang langsung modelling aja,” komentarku. ”Sebenarnya tidak panjang, ini karena kamu  masih tahap belajar sehingga kamu  perlu mengerti konsepnya dan tidak asal membuat model, tetapi ketika kamu sudah  mulai implementasi proses ini akan otomatis dilakukan sehingga tidak terasa panjang lagi. Semangat yah!”

“Sip, masih semangat kok. Jadi selanjutnya apa?”

“Oke, saya lanjutkan. Pertama-tama saya akan mengenalkan kamu pada library Scikit - Learn. Scikit-learn adalah library untuk machine learning bagi para pengguna python yang memungkinkan kita melakukan berbagai pekerjaan dalam Data Science, seperti regresi (regression), klasifikasi (classification), pengelompokkan/penggugusan (clustering), data preprocessing, dimensionality reduction, dan model selection (pembandingan, validasi, dan pemilihan parameter maupun model).”

Sembari Senja menjelaskan, aku membuka halaman demi halaman modul:

Ada beberapa library machine learning di Python seperti Keras, tetapi Scikit - Learn adalah yang paling basic sehingga jika kita menguasai scikit-learn, kita dapat dengan mudah mempelajari library machine learning yang lain.

#### Features & Label

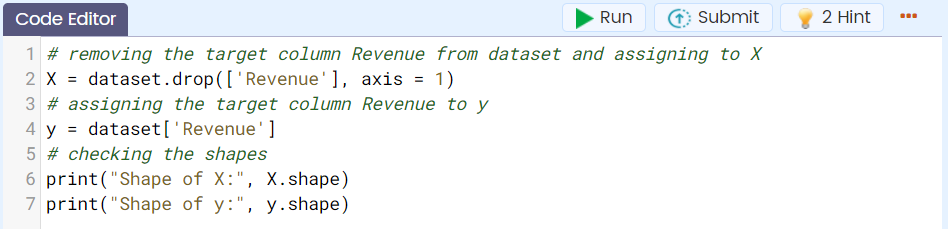
Dalam dataset user online purchase, label target sudah diketahui, yaitu kolom **Revenue** yang bernilai 1 untuk user yang membeli dan 0 untuk yang tidak membeli, sehingga pemodelan yang dilakukan ini adalah klasifikasi. Nah, untuk melatih dataset menggunakan Scikit-Learn library, dataset perlu dipisahkan ke dalam **Features** dan **Label/Target**. Variabel Feature akan terdiri dari variabel yang dideklarasikan sebagai **X** dan**[Revenue]** adalah variabel Target yang dideklarasikan sebagai **y**. Gunakan fungsi **drop()** untuk menghapus kolom [Revenue] dari dataset.

“Sudah baca petunjuknya, Aksara?”

“Sudah, ini bisa langsung aku coba jalankan?” tanyaku menunggu lampu hijau dari Senja.

“Silakan.”

Aku pun mulai bekerja untuk mengubah dataset ke dalam format yang diminta oleh Scikit - Learn sesuai arahan Senja dengan kode berikut:



Jika dijalankan dengan  akan menghasilkan output berikut



#### Training dan Test Dataset

“Well done, Aksara! Nah, sebelum kita melatih model dengan suatu algorithm machine , seperti yang saya jelaskan sebelumnya, dataset perlu kita bagi ke dalam training dataset dan test dataset dengan perbandingan 80:20. **80%** digunakan untuk training dan**20%** untuk proses testing.”

Aku kembali menyimak.

“Perbandingan lain yang biasanya digunakan adalah 75:25. Hal penting yang perlu diketahui adalah scikit-learn tidak dapat memproses dataframe dan hanya mengakomodasi format data tipe Array. Tetapi kalian tidak perlu khawatir, fungsi **train\_test\_split( )** dari Scikit-Learn, otomatis mengubah dataset dari dataframe ke dalam format array. Apakah kamu paham. Aksara? Atau ada pertanyaan?”

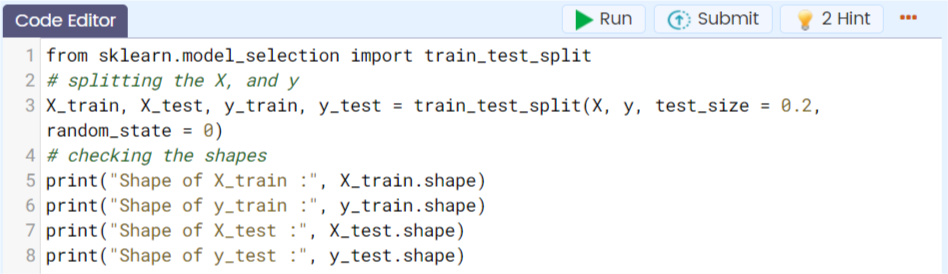
“Kenapa perlu ada Training dan Testing, Nja?”

  “Fungsi Training adalah melatih model untuk mengenali pola dalam data, sedangkan testing berfungsi untuk memastikan bahwa model yang telah dilatih tersebut mampu dengan baik memprediksi label dari new observation dan belum dipelajari oleh model sebelumnya. Lebih baik kita praktik saja ya, tampaknya kalau praktik kamu lebih paham.”

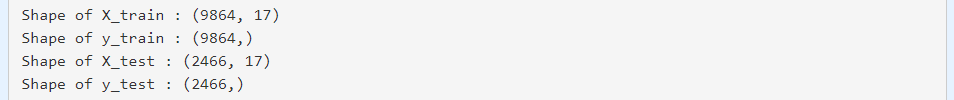
Senja menarik laptopnya, sepertinya sedang membuat dokumen untuk praktik. Dan, benar saja…

“Aksara silahkan bagi dataset ke dalam Training dan Testing dengan melanjutkan coding yang  sudah kukerjakan ini. Gunakan **test\_size = 0.2** dan tambahkan argumen **random\_state = 0,** pada fungsi **train\_test\_split( ).**Dicoba saja dulu yah, saya yakin kamu bisa.”

“Siap,” aku segera menyusun kode sesuai penjelasan Senja tadi:



Jika dijalankan dengan  akan menghasilkan output berikut



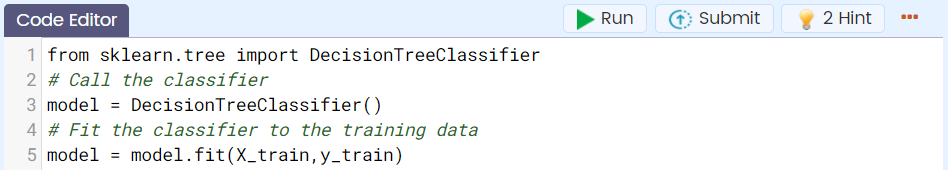
#### Training Model: Fit

“Good Job, Aksara! Sekarang saatnya kita melatih model atau training. Dengan Scikit-Learn, proses ini menjadi sangat sederhana. Kita cukup memanggil nama algorithm yang akan kita gunakan, biasanya disebut classifier untuk problem klasifikasi, dan regressor untuk problem regresi.”

Aku selalu suka ketika Senja mengapresiasiku sesederhana apapun itu, karena selalu berhasil mendorong semangat belajarku. Aku jadi lebih berani untuk mencoba dan menyimak hal baru. Dan satu lagi, Senja selalu mau aku repotkan dengan meminta contoh.

“Boleh kasih contohnya, Nja?”

“Begini, sebagai contoh, kita akan menggunakan Decision Tree. Kita hanya perlu memanggil fungsi **DecisionTreeClassifier()** yang kita namakan **“model”**. Kemudian menggunakan fungsi **.fit()** dan **X\_train, y\_train** untuk melatih classifier tersebut dengan training dataset, seperti ini:”



Jika tidak ada error yang muncul silakan dilanjutkan.

#### Training Model: Predict

“Yang tadi sudah cukup saya rasa. Setelah model/classifier terbentuk, selanjutnya kita menggunakan model ini untuk memprediksi LABEL dari testing dataset (**X\_test**), menggunakan fungsi **.predict()**. Fungsi ini akan mengembalikan hasil prediksi untuk setiap data point dari **X\_test** dalam bentuk array. Proses ini kita kenal dengan TESTING,” sambung Senja.

Benar-benar membutuhkan konsentrasi penuh untuk materi modul ini. Aku pun melanjutkan proses testing menggunakan fungsi **.predict()**seperti ini:



Dengan menekan diperoleh



ukuran y\_pred sama dengan ukuran y\_test.

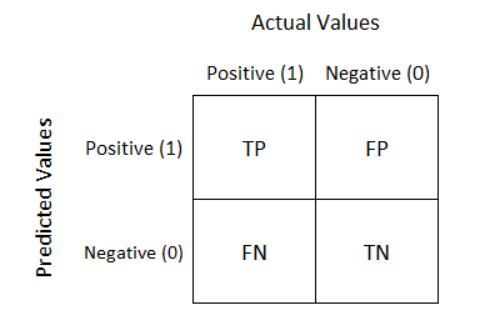
#### Evaluasi Model Performance - Part 1

Aku menelusuri ulang susunan kodeku, aku merasa ini sudah lengkap dan siap. “Nja, ini semua sudah selesai menurutku, ada tahap akhir khusus kah?”

“Tentu saja. sekarang kita melanjutkan di tahap terakhir dari modelling yaitu evaluasi hasil model. Untuk evaluasi model performance, setiap algorithm mempunyai metrik yang berbeda-beda. Sekarang saya akan menjelaskan sedikit metrik apa saja yang umumnya digunakan. Metrik paling sederhana untuk mengecek performansi model adalah accuracy.”

“Gimana caranya?” tanyaku bingung karena awalnya kupikir ini sudah tuntas.

“Kita bisa munculkan dengan fungsi **.score( )**. Tetapi, di banyak real problem, accuracy saja tidaklah cukup. Metode lain yang digunakan adalah dengan Confusion Matrix. Confusion Matrix merepresentasikan perbandingan prediksi dan real LABEL dari test dataset yang dihasilkan oleh algoritma ML,” tukas Senja sambil membuka  template dari confusion Matrix untukku:



**True Positive (TP)**: Jika user diprediksi (Positif) membeli ([Revenue] = 1]), dan memang benar(True) membeli.

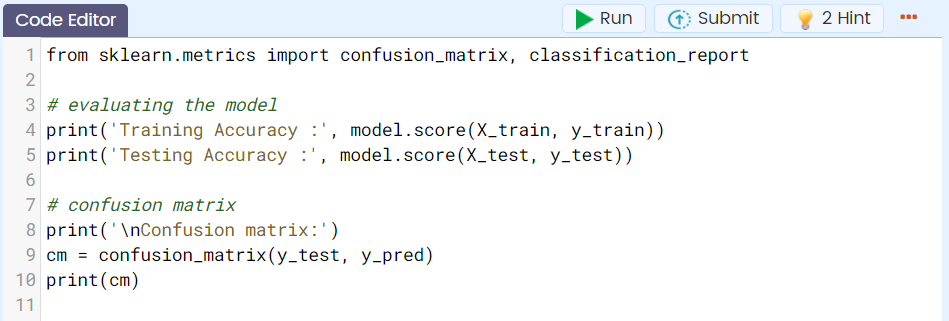
**True Negative (TN)**: Jika user diprediksi tidak (Negatif) membeli dan aktualnya user tersebut memang (True) membeli.

**False Positive (FP)**: Jika user diprediksi Positif membeli, tetapi ternyata tidak membeli (False).

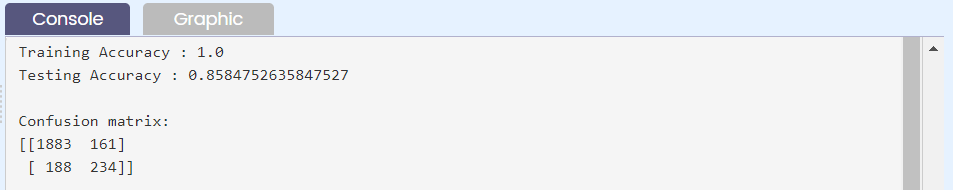
**False Negatif (FN)**: Jika user diprediksi tidak membeli (Negatif), tetapi ternyata sebenarnya membeli.

#### Evaluasi Model Performance - Part 2

Untuk menampilkan confusion matrix cukup menggunakan fungsi **confusion\_matrix()** dari Scikit-Learn



dengan output



Berdasarkan confusion matrix, dapat mengukur metrik - metrik berikut :

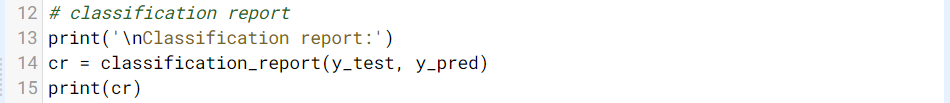
Accuracy = (TP + TN ) / (TP+FP+FN+TN)

Precision = (TP) / (TP+FP)

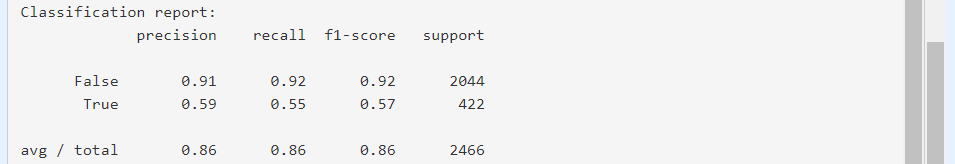
Recall = (TP) / (TP + FN)

F1 Score = 2 \* (Recall\*Precission) / (Recall + Precission)

Tidak perlu menghitung nilai ini secara manual. Cukup gunakan  fungsi **classification\_report()** untuk memunculkan hasil perhitungan metrik - metrik tersebut.



dengan output



#### Pakai Metrik yang Mana?

Jika dataset memiliki jumlah data False Negatif dan False Positif yang seimbang (Symmetric), maka bisa gunakan **Accuracy**, tetapi jika tidak seimbang, maka sebaiknya menggunakan **F1-Score**.

Dalam suatu problem, jika lebih memilih False Positif lebih baik terjadi daripada False Negatif, misalnya: Dalam kasus Fraud/Scam, kecenderungan model mendeteksi transaksi sebagai fraud walaupun kenyataannya bukan, dianggap lebih baik, daripada transaksi tersebut tidak terdeteksi sebagai fraud tetapi ternyata fraud. Untuk problem ini sebaiknya menggunakan **Recall**.

Sebaliknya, jika lebih menginginkan terjadinya True Negatif dan sangat tidak menginginkan terjadinya False Positif, sebaiknya menggunakan **Precision**.

Contohnya adalah pada kasus klasifikasi email SPAM atau tidak. Banyak orang lebih memilih jika email yang sebenarnya SPAM namun diprediksi tidak SPAM (sehingga tetap ada pada kotak masuk email kita), daripada email yang sebenarnya bukan SPAM tapi diprediksi SPAM (sehingga tidak ada pada kotak masuk email).

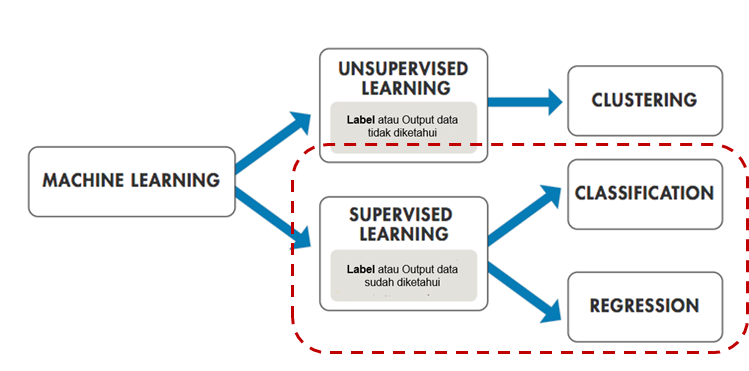
## Chapter 4

### Supervised Learning - Algorithm

#### Pendahuluan

Setelah pemahaman dengan prosedur machine learning modelling. Selanjutnya materi akan membahas mengenai machine learning algorithm.

Sebagai dasar, akan dipelajari beberapa algorithm machine learning yaitu Logistic Regression, dan Decision Tree untuk classification problem, dan Linear regression untuk regression problem.



#### Classification - Logistic Regression

Logistic Regression merupakan salah satu algoritma klasifikasi dasar yang cukup popular. Secara sederhana, Logistic regression hampir serupa dengan linear regression tetapi linear regression digunakan untuk Label atau Target Variable yang berupa numerik atau continuous value, sedangkan Logistic regression digunakan untuk Label atau Target yang berupa **categorical/discrete value**.

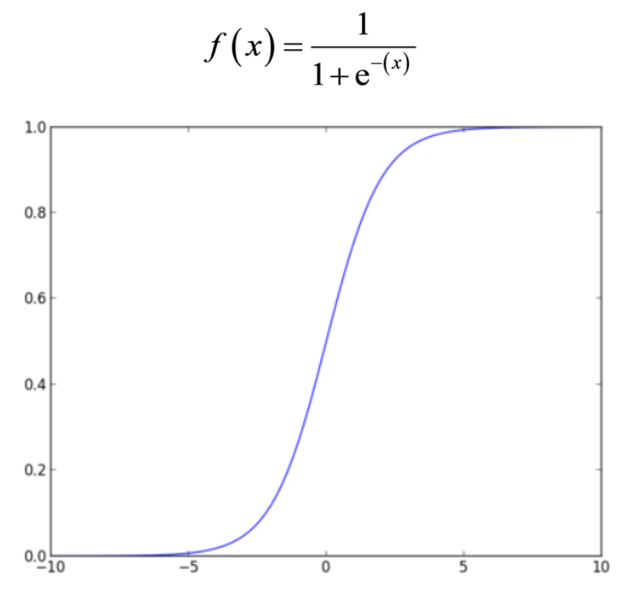
Contoh continuous value adalah harga rumah, harga saham, suhu, dsb; dan contoh dari categorical value adalah prediksi SPAM or NOT SPAM (1 dan 0) atau prediksi customer SUBSCRIBE atau UNSUBSCRIBED (1 dan 0).

Umumnya Logistic Regression dipakai untuk binary classification (1/0; Yes/No; True/False) problem, tetapi beberapa data scientist juga menggunakannya untuk multiclass classification problem. Logistic regression adalah salah satu linear classifier, oleh karena itu, Logistik regression juga menggunakan rumus atau fungsi yang sama seperti linear regression yaitu:



yang disebut Logit, dimana Variabel 𝑏₀, 𝑏₁, …, 𝑏ᵣ adalah koefisien regresi, dan 𝑥₁, …, 𝑥ᵣ adalah explanatory variable/variabel input atau feature.

Output dari Logistic Regression adalah 1 atau 0; sehingga real value dari fungsi logit ini perlu ditransfer ke nilai di antara 1 dan 0 dengan menggunakan fungsi sigmoid.



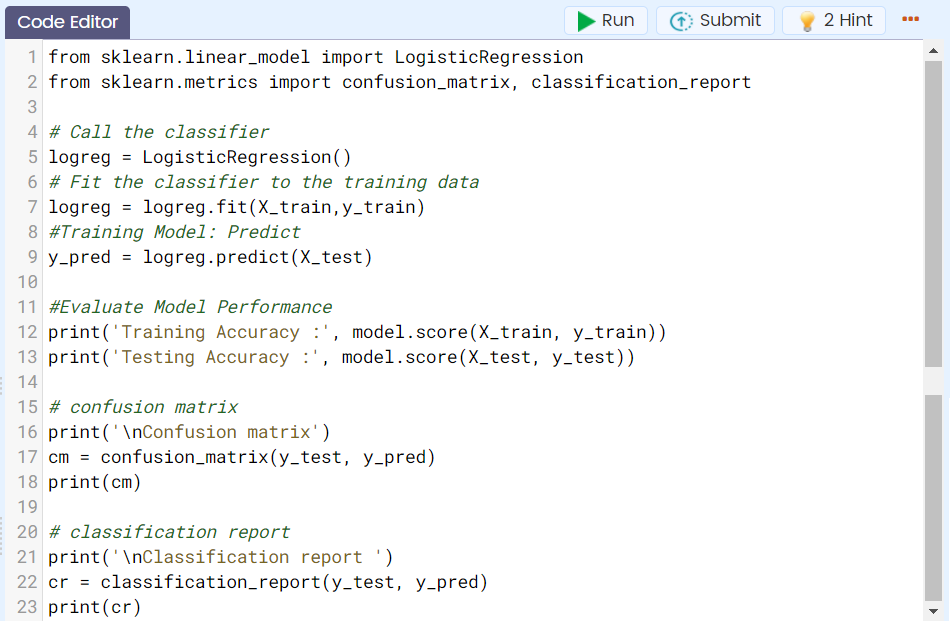
Jadi, jika output dari fungsi sigmoid bernilai lebih dari 0.5, maka data point diklasifikasi ke dalam label/class: 1 atau YES; dan kurang dari 0.5, akan diklasifikasikan ke dalam label/class: 0 atau NO.

Logistic Regression hanya dapat mengolah data dengan tipe numerik. Pada saat preparasi data, pastikan untuk mengecek tipe variabel yang ada dalam dataset dan pastikan semuanya adalah numerik, lakukan data transformasi jika diperlukan.

#### Pemodelan Permasalahan Klasifikasi dengan Logistic Regression

Pemodelan Logistic Regression dengan memanfaatkan Scikit-Learn sangatlah mudah. Dengan menggunakan dataset yang sama yaitu **online\_raw**, dan setelah dataset dibagi ke dalam Training Set dan Test Set, cukup menggunakan modul **linear\_model** dari Scikit-learn, dan memanggil fungsi **LogisticRegression()**yang diberi nama **logreg**.

Kemudian, model yang sudah ditraining ini  bisa digunakan untuk memprediksi output/label dari test dataset sekaligus mengevaluasi model performance dengan fungsi **score()**, **confusion\_matrix()**dan **classification\_report()**.

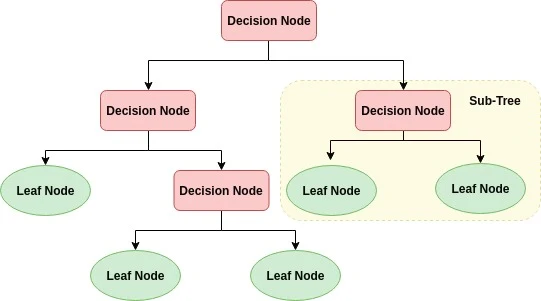


Dengan menekan  diperoleh hasil berikut ini



#### Classification - Decision Tree

Decision Tree merupakan salah satu metode klasifikasi yang populer dan banyak diimplementasikan serta mudah diinterpretasi. Decision tree adalah model prediksi dengan struktur pohon atau struktur berhierarki. Decision Tree dapat digunakan untuk classification problem dan regression problem. Secara sederhana, struktur dari decision tree adalah sebagai berikut:



Decision tree terdiri dari :

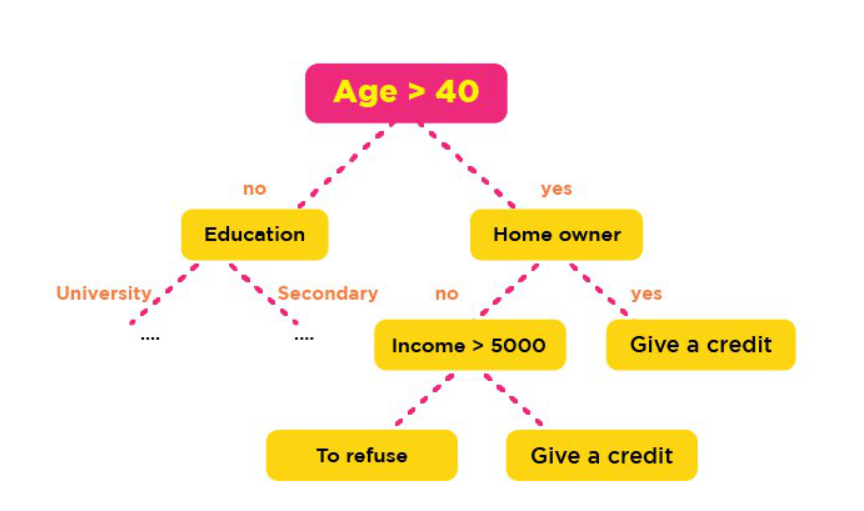
Decision Node yang merupakan feature/input variabel;

Branch yang ditunjukkan oleh garis hitam berpanah, yang adalah rule/aturan keputusan, dan

Leaf yang merupakan output/hasil.

Decision Node paling atas dalam decision tree dikenal sebagai akar keputusan, atau feature utama yang menjadi asal mula percabangan. Jadi, decision tree membagi data ke dalam kelompok atau kelas berdasarkan feature/variable input, yang dimulai dari node paling atas (akar), dan terus bercabang ke bawah sampai dicapai cabang akhir atau leaf.

Misalnya ingin memprediksi apakah seseorang yang mengajukan aplikasi kredit/pinjaman, layak untuk mendapat pinjaman tersebut atau tidak. Dengan menggunakan decision tree, dapat membreak-down kriteria-kriteria pengajuan pinjaman ke dalam hierarki seperti gambar berikut :



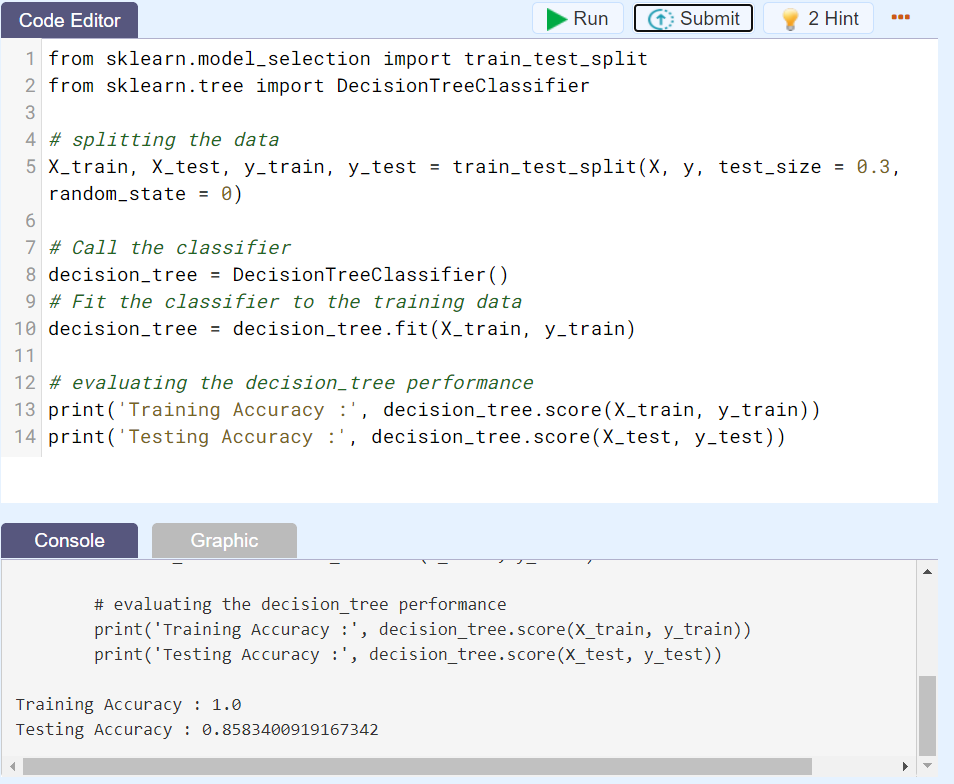
Seumpama, orang yang mengajukan berumur lebih dari 40 tahun, dan memiliki rumah, maka aplikasi kreditnya dapat diluluskan, sedangkan jika tidak, maka perlu dicek penghasilan orang tersebut. Jika kurang dari 5000, maka permohonan kreditnya akan ditolak. Dan jika usia kurang dari 40 tahun, maka selanjutnya dicek jenjang pendidikannya, apakah universitas atau secondary. Nah, percabangan ini masih bisa berlanjut hingga dicapai percabangan akhir/leaf node.

Seperti yang sudah dilakukan dalam prosedur pemodelan machine learning, selanjutnya dapat dengan mudah melakukan pemodelan decision tree dengan menggunakan scikit-learn module, yaitu **DecisionTreeClassifier**.

#### Tugas Praktek

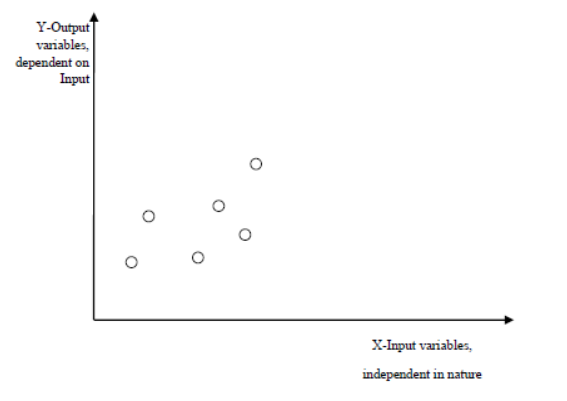
Dengan menggunakan dataset **online\_raw.csv** dan diasumsikan sudah melakukan EDA dan pre-processing, aku akan membuat model machine learning dengan menggunakan **decision tree** :

1. Import **DecisionTreeClassifier** dan panggil fungsi tersebut dengan nama **decision\_tree**
2. Split dataset ke dalam training & testing dataset dengan perbandingan **70:30**, dengan **random\_state = 0**
3. Latih model dengan training feature **(X\_train)** dan training target **(y\_train)** menggunakan **.fit()**
4. Evaluasi hasil model **decision\_tree** yang sudah dilatih dengan testing feature **(X\_test)** dan print nilai akurasi dari **training dan testing** dengan fungsi **.score()**



#### Regression: Linear Regression - Part 1

Regression merupakan metode statistik dan machine learning yang paling banyak digunakan. Seperti yang dijelaskan sebelumnya, regresi digunakan untuk memprediksi output label yang berbentuk numerik atau continuous value. Dalam proses training, model regresi akan menggunakan variabel input (features) dan variabel output (label) untuk mempelajari bagaimana hubungan/pola dari variabel input dan output.



Model regresi terdiri atas 2 tipe yaitu :

1. Simple regression model → model regresi paling sederhana, hanya terdiri dari satu feature (univariate) dan 1 target.
2. Multiple regression model → sesuai namanya, terdiri dari lebih dari satu feature (multivariate).

Adapun model regresi yang paling umum digunakan adalah Linear Regression.

#### Regression: Linear Regression - Part 2

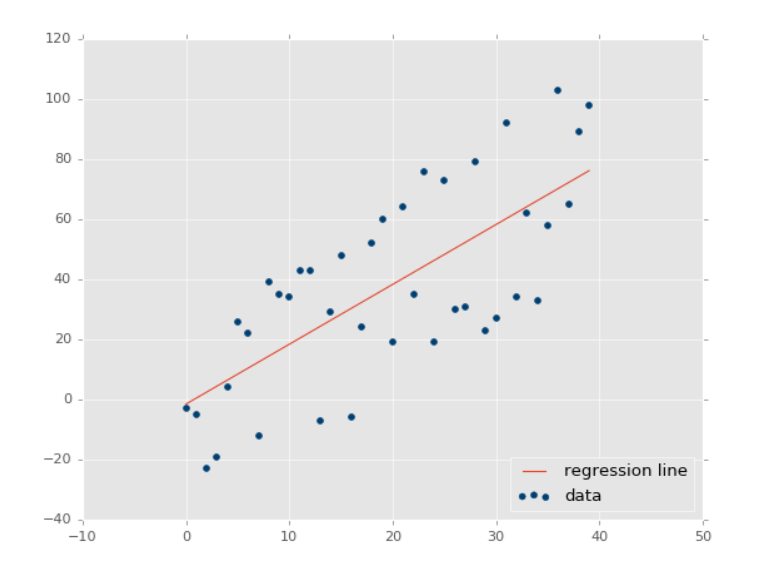
Linear regression digunakan untuk menganalisis hubungan linear antara dependent variabel (feature) dan independent variabel (label). Hubungan linear disini berarti bahwa jika nilai dari independen variabel mengalami perubahan baik itu naik atau turun, maka nilai dari dependen variabel juga mengalami perubahan (naik atau turun). Rumus matematis dari Linear Regression adalah:



untuk simple linear regression, atau



untuk multiple linear regression dengan, **y** adalah target/label, **X** adalah feature, dan **a**,***b*** adalah model parameter (intercept dan slope).



Perlu diketahui bahwa tidak semua problem dapat diselesaikan dengan linear regression. Untuk pemodelan dengan linear regression, terdapat beberapa asumsi yang harus dipenuhi, yaitu :

1. Terdapat hubungan linear antara variabel input (feature) dan variabel output(label). Untuk melihat hubungan linear feature dan label, dapat menggunakan chart seperti scatter chart. Untuk mengetahui hubungan dari variabel umumnya dilakukan pada tahap eksplorasi data.
2. Tidak ada multicollinearity antara features. Multicollinearity artinya terdapat dependency antara feature, misalnya saja hanya bisa mengetahui nilai feature B jika nilai feature A sudah diketahui.
3. Tidak ada autocorrelation dalam data, contohnya pada time-series data.

Pemodelan Linear regression menggunakan scikit-learn tidaklah sulit. Secara prosedur serupa dengan pemodelan logistic regression. Cukup memanggil **LinearRegression** dengan terlebih dahulu meng-import fungsi tersebut :

from sklearn.linear\_model import LinearRegression

“Setelah memahami konsep dasar dari regression, kita akan berlatih membuat model machine learning dengan Linear regression. Untuk pemodelan ini kita akan menggunakan data ‘Boston Housing Dataset’. Setelah pembelajaran kamu sampai di sini, tahu tidak   mengapa kita tidak bisa menggunakan data “online purchase”, Aksara?”

Pertanyaan Senja padaku terdengar seperti ujian. Aku berpikir sejenak sebelum menjawab,

“Hmm, karena untuk linear regression target/label harus berupa numerik, sedangkan target dari online purchase data adalah categorical. Apakah benar?” jawabku ragu-ragu.

Senyum Senja cukup melegakanku.

“Tepat sekali, Senja. Kalau begitu kita bisa lanjut ke pemodelan. Tujuan dari pemodelan ini adalah memprediksi harga rumah di Boston berdasarkan feature - feature yang ada. Asumsikan saja bahwa kita sudah melakukan data eksplorasi dan data pre-processing. Jadi, data yang akan digunakan adalah data yang siap untuk diproses ke tahap pemodelan.”

“Siap, praktik langsung saja,” usulku percaya diri dengan mengklik tombol **Next**.

#### Tugas Praktek

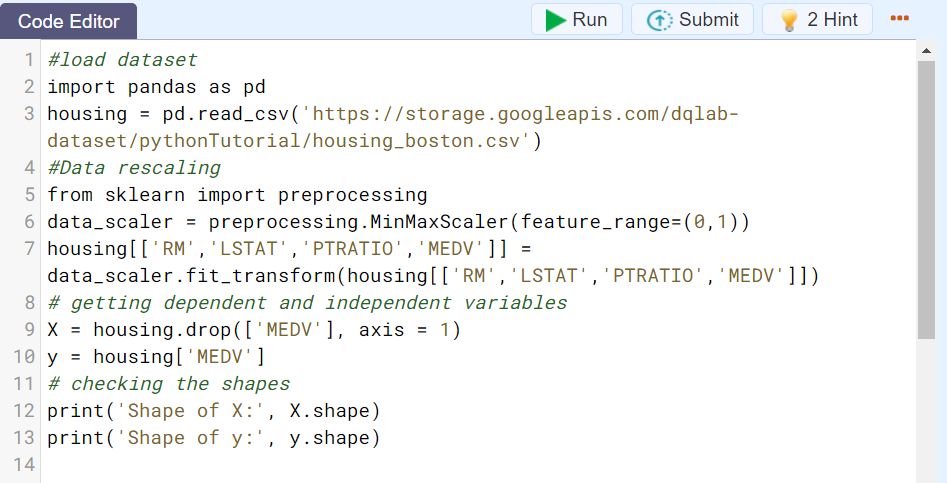
“Oke, saya tahu kamu sudah enggak sabar. Sebelumnya kamu membuat modellingnya, saya jelaskan prosedur dan library yang tepat untuk digunakan ya, nanti saya akan email detail intruksinya.”

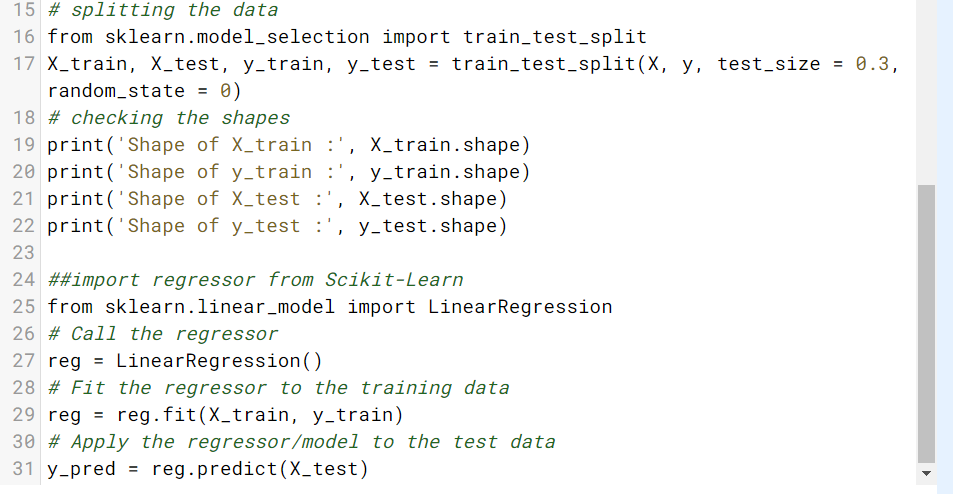
Aku pun menunggu email dari Senja. Setelah beberapa menit, pesan yang kutunggu akhirnya muncul:

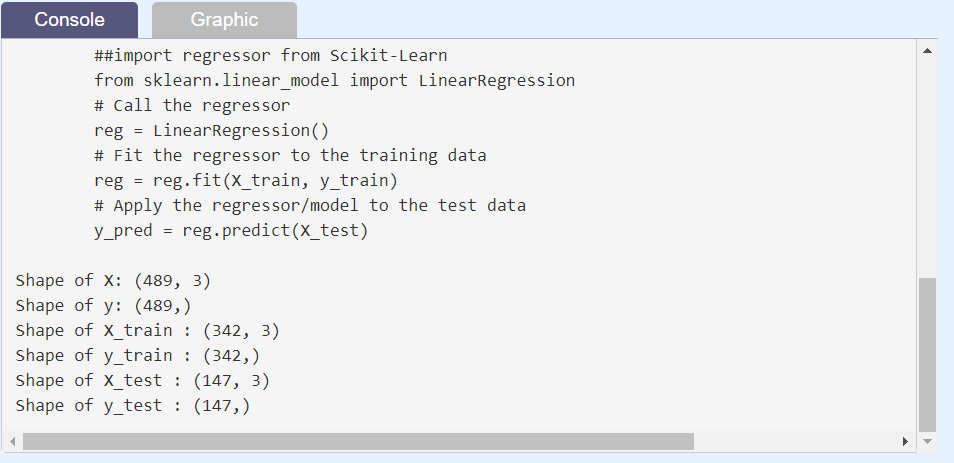
1. Pisahkan dataset ke dalam Feature dan Label, gunakan fungsi **.drop()**. Pada dataset ini, label/target adalah variabel **MEDV**
2. Checking dan print jumlah data setelah Dataset pisahkan ke dalam Feature dan Label, gunakan **.shape()**
3. Bagi dataset ke dalam Training dan test dataset, 70% data digunakan untuk training dan 30% untuk testing, gunakan fungsi **train\_test\_split() ,** dengan **random\_state** = 0
4. Checking dan print kembali jumlah data dengan fungsi **.shape()**
5. Import **LinearRegression** dari **sklearn.linear\_model**
6. Deklarasikan  **LinearRegression** regressor dengan nama **reg**
7. Fit regressor ke training dataset dengan **.fit()**, dan gunakan **.predict()** untuk memprediksi nilai dari testing dataset.

“Kalau dibaca aja kelihatannya membingungkan, tapi kalau sudah dicoba pasti bisa, Aksara. Semangat!”

Baiklah, aku akan mulai mengerjakan di code editor.



Hasil :



#### Regression Performance Evaluation

Aku sudah sampai tahap evaluasi. Sudah sejauh ini tapi ada bagian membingungkan yang kutemukan. Berhubung Senja masih duduk di sebelahku untuk memantau proses kerjaku, aku pun bertanya,

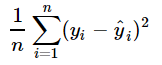
“Kalau mengevaluasi perfoma dari model klasifikasi, aku pakai akurasi dan confusion matrix. Nah, kalau modenya regression, metode evaluasinya bagaimana yah. Nja?”

Senja yang sedang fokus di depan layar laptop akhirnya menoleh oleh sahutanku.

“Untuk model regression, kita menghitung selisih antara nilai aktual (y\_test) dan nilai prediksi (y\_pred) yang disebut **error**, adapun beberapa metric yang umum digunakan. Coba kamu ke mari, aku jelaskan langkah-langkahnya.”

Keuntungan bertanya pada Senja adalah ia selalu menyempatkan waktunya untuk menjelaskan secara maksimal! Aku fokus memperhatikan:

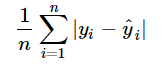
**Mean Squared Error** (MSE) adalah rata-rata dari squared error:



**Root Mean Squared Error** (RMSE) adalah akar kuadrat dari MSE:



**Mean Absolute Error** (MAE) adalah rata-rata dari nilai absolut error:



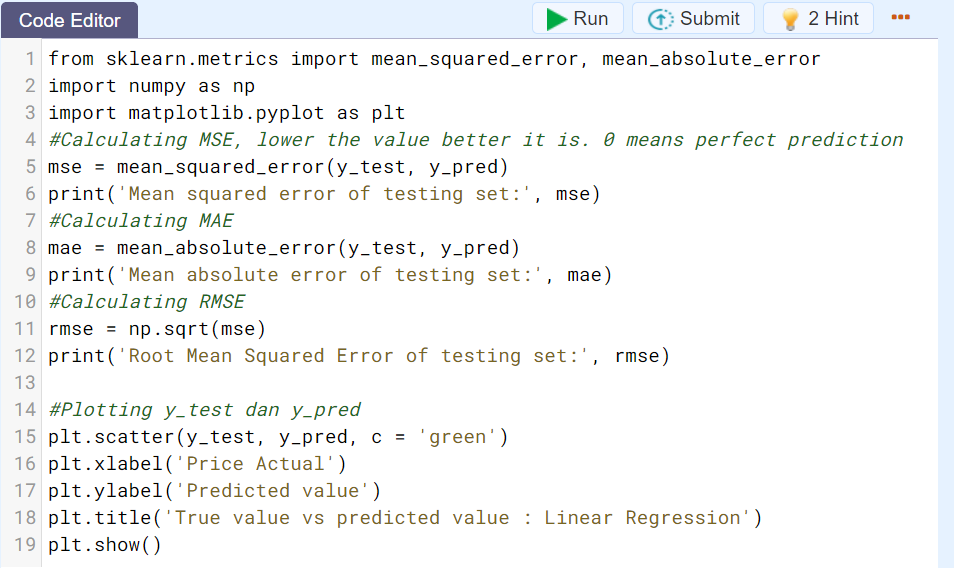
Semakin kecil nilai MSE, RMSE, dan MAE, semakin baik pula performansi model regresi. Untuk menghitung nilai MSE, RMSE dan MAE dapat dilakukan dengan menggunakan fungsi **mean\_squared\_error ()** ,  **mean\_absolute\_error ()** dari **scikit-learn.metrics** dan untuk RMSE sendiri tidak terdapat fungsi khusus di scikit-learn tapi dapat dengan mudah kita hitung dengan terlebih dahulu menghitung MSE kemudian menggunakan **numpy** module yaitu, **sqrt()**untuk memperoleh nilai akar kuadrat dari MSE.

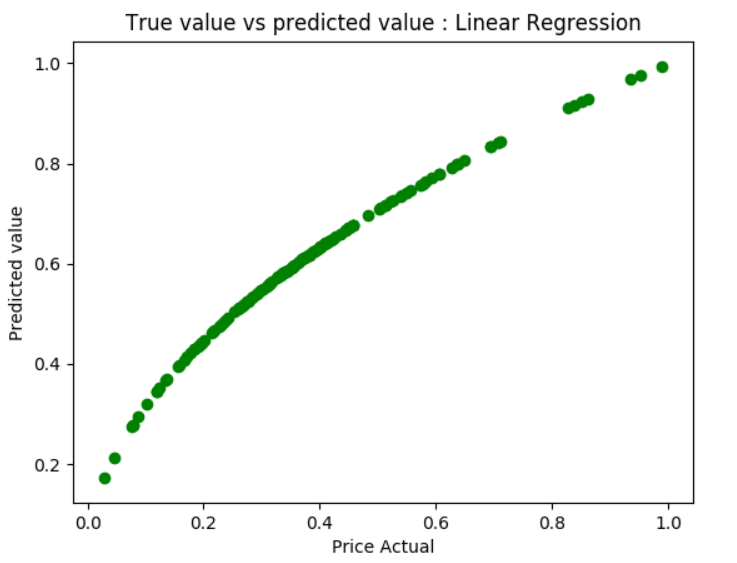
#### Tugas Praktek

"Kalau kamu sudah paham evaluasi performa model regresi, sekaligus kamu coba hitung nilai MSE, MAE, dan RMSE dari linear modelnya, Aksara,” pinta Senja

Aku baru mau mengutak-atik kodenya ketika email dari Senja muncul berisi intruksi:

1. Import library yang digunakan: **mean\_squared\_error**, **mean\_absolute\_error** dari **sklearn.metrics** dan **numpy** sebagai aliasnya yaitu **np**. Serta, import juga matplotlib.pyplot sebagai aliasnya, plt.
2. Hitung dan print nilai MSE dan RMSE dengan menggunakan argumen **y\_test** dan **y\_pred**, untuk rmse gunakan **np.sqrt()**
3. Buat scatter plot yang menggambarkan hasil prediksi **(y\_pred)** dan harga actual **(y\_test)**





## Chapter 5

### Unsupervised Learning - Algorithm

#### Pendahuluan

Tak seperti kemarin, hari ini Senja memintaku datang ke ruangannya.

“Silakan duduk. Aksara. Kita bakal full belajar materi baru hari ini, jadi bakal lebih intens. Makanya aku minta kamu ke ruanganku.”

Aku diam menyimak.

“Seperti yang sudah dijelaskan sebelumnya, Machine Learning terdiri atas 2 tipe yaitu supervised dan unsupervised learning. Kita telah banyak membahas tentang supervised learning yaitu Klasifikasi model dan Regression Model. Sekarang kita akan mempelajari dasar- dasar terkait unsupervised learning,” jelas Senja.

Kulihat dengan cepat ia membuka laptop dan menampilkan layar presentasi di depan yang menampilkan rangkuman materi sekaligus slide conton gambar:

**Unsupervised Learning** adalah teknik machine learning dimana tidak terdapat label atau output yang digunakan untuk melatih model. Jadi, model dengan sendirinya akan bekerja untuk menemukan pola atau informasi dari dataset yang ada. Metode unsupervised learning yang dikenal dengan clustering. Sesuai dengan namanya, Clustering memproses data dan mengelompokkannya atau mengcluster objek/sample berdasarkan kesamaan antar objek/sampel dalam satu kluster, dan objek/sample ini cukup berbeda dengan objek/sample di kluster yang lain. Contohnya pada gambar berikut:

Chart, diagram, scatter chart

Description automatically generated

"Nja, mau tanya. Kita tahu dari mana bentuk polanya?”

“Pada awalnya kita tidak mengetahui bagaimana pola dari objek/sample, termasuk juga tidak mengetahui bagaimana kesamaan maupun perbedaan antara objek yang satu dengan objek yang lain. Setelah dilakukan clustering, baru dapat terlihat bawah objek/sample tersebut dapat dikelompokkan ke dalam 3 kluster. Untuk menjelaskan tentang metode Clustering, kita akan menggunakan metode clustering yang sangat populer, yaitu K-Means Algorithm yang akan kita praktikkan nanti.”

#### K-Means Clustering

"Jadi, Algorithm K-Means itu apa dan bagaimana cara kerjanya?” tanyaku antusias.

 “K-Means merupakan tipe clustering dengan centroid based (titik pusat). Artinya kesamaan dari objek/sampel dihitung dari seberapa dekat objek itu dengan centroid atau titik pusat.”

Aku masih penasaran. “Jadi, bagaimana kita mengukur kedekatan objek dan centroid?”

 “Untuk menghitung kedekatan, digunakan perhitungan jarak antar 2 buah data atau jarak Minkowski. Saya share yah rumusnya,” ujar Senja.

Aku menyimak isi rumus yang dibagikan Senja di slide presentasinya:



xi , xj adalah dua buah data yang akan dihitung jaraknya, dan p = dimensi/jumlah dari data

Terdapat beberapa tipe perhitungan jarak yang dapat digunakan, yaitu :

Jarak Manhattan di mana g = 1

Jarak Euclidean di mana g = 2

Jarak Chebychev di mana g = ∞

 “Nja, aku masih bingung, cara menentukan centroid bagaimana caranya?”

 “Untuk menentukan centroid, pada awalnya kita perlu mendefinisikan jumlah centroid (K) yang diinginkan, semisalnya kita menetapkan jumlah K = 3; maka pada awal iterasi, algorithm akan secara random menentukan 3 centroid. Setelah itu, objek/sample/data point yang lain akan dikelompokkan sebagai anggota dari salah satu centroid yang terdekat, sehingga terbentuk 3 cluster data. Sampai sini cukup dipahami?”

“Yup, boleh lanjut, Nja,” sahutku mempersilakan Senja kembali menjelaskan.

“Iterasi selanjutnya, titik-titik centroid diupdate atau berpindah ke titik yang lain, dan jarak dari data point yang lain ke centroid yang baru dihitung kembali, kemudian dikelompokkan kembali berdasarkan jarak terdekat ke centroid yang baru. Iterasi akan terus berlanjut hingga diperoleh cluster dengan error terkecil, dan posisi centroid tidak lagi berubah.”

Chart

Description automatically generated

“Kamu sudah bisa lihat di layar ya, Aksara. Menurutmu, apakah ada perbedaan prosedur antara unsupervised learning dan supervised learning?”

Aku tahu ini pertanyaan untuk menguji pemahamanku.

“Secara prosedur, tahap eksplorasi data untuk memahami karakteristik data, dan tahap preprocessing tetap dilakukan. Tetapi dalam unsupervised learning, kita tidak membagi dataset ke feature dan label; dan juga ke dalam training dan test dataset, karena pada dasarnya kita tidak memiliki informasi mengenai label/target data,” jawabku mantap.

“Tampaknya kamu sudah paham. Saatnya kita mulai praktik membuat programnya.”

#### Tugas Praktek

“Untuk praktik  ini, kita akan menggunakan dataset ‘Mall Customer Segmentation’,” ujar Senja.

Aku membaca detail latihan yang sudah ia catatkan untukku:

Dataset ini merupakan data customer suatu mall dan berisi basic informasi customer berupa : CustomerID, age, gender, annual income, dan spending score.  Adapun tujuan dari clustering adalah untuk memahami customer - customer mana saja yang sering melakukan transaksi sehingga informasi ini dapat diberikan kepada marketing team untuk membuat strategi promosi yang sesuai dengan karakteristik customer.

“Kita akan melakukan segmentasi customer, dengan memanfaatkan fungsi KMeans dari Scikit-Learn.cluster. Silakan berlatih dengan intruksi di catatan tadi ya, Aksara.”

Aku membuka kembali catatan yang berisi intruksi Senja:

1. Import **pandas** sebagai aliasnya dan **KMeans** dari **sklearn.cluster**.
2. Load dataset **'https://storage.googleapis.com/dqlab-dataset/pythonTutorial/mall\_customers.csv'** dan beri nama **dataset**
3. Diasumsikan EDA dan preprocessing sudah dilakukan, selanjutnya kita memilih feature yang akan digunakan untuk membuat model yaitu **annual\_income** dan **spending\_score**. Assign dataset dengan feature yang sudah dipilih ke dalam **'X'**. Pada dasarnya terdapat teknik khusus yang dilakukan untuk menyeleksi feature - feature (Feature Selection) mana saja yang dapat digunakan untuk machine learning modelling, karena tidak semua feature itu berguna. Beberapa feature justru bisa menyebabkan performansi model menurun. Tetapi untuk problem ini, secara default kita akan menggunakan **annual\_income** dan **spending\_score.**
4. Deklarasikan  **KMeans( )**  dengan nama **cluster\_model**dan gunakan **n\_cluster = 5**. n\_cluster adalah argumen dari fungsi KMeans( ) yang merupakan jumlah cluster/centroid (K).  random\_state = 24.
5. Gunakan fungsi .**fit\_predict( )** dari **cluster\_model** pada 'X'  untuk proses clustering.

Setelah dijalankan dengan  dan tidak menghasilkan error maka kode yang ditulis sudah benar.

Graphical user interface, text, application

Description automatically generated

#### Tugas Praktek

Inspect & Visualizing the Cluster

“Satu lagi, Aksara kalau sudah membuat cluster, tolong  visualisasikan hasil dari clustering yang telah kamu lakukan sebelumnya ya. Langkah-langkahnya sudah saya email,” tambah Senja lagi tepat saat aku sedang membuka pesan berisi intruksi tambahan darinya:

1. Pertama - tama, import **matplotlib.pyplot** dan beri inisial **plt**.
2. Gunakan fungsi **.values** untuk mengubah tipe ‘X’ dari dataframe menjadi array
3. Pisahkan X kedalam xs dan ys, di mana xs adalah Kolom index [0] dan ys adalah kolom index [1]
4. Buatlah scatter plot **plt.scatter()** dari xs dan ys, kemudian tambahkan c = labels untuk secara otomatis memberikan warna yang berbeda pada setiap cluster, dan alpha = 0.5 ke dalam scatter plot argumen.
5. Hitunglah koordinat dari centroid menggunakan **.cluster\_centers\_** dari cluster\_model, deklarasikan ke dalam variabel centroids.
6. Pisahkan centroids kedalam **centroids\_x** dan **centroids\_y,** di mana **centroids\_x** adalah kolom index [0] dan **centroids\_y** adalah kolom index [1]
7. Buatlah scatter plot dari **centroids\_x** dan **centroids\_y** , gunakan ‘D’ (diamond) sebagai marker parameter, dengan ukuran 50, s = 50

Jika dijalankan dengan  akan diperoleh grafik seperti berikut ini

Graphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

#### Measuring Cluster Criteria

“Segmentasinya udah jadi nih, Nja. Tapi, bagaimana kita tahu bahwa membagi segmentasi ke dalam 5 cluster adalah segmentasi yang paling optimal? Karena jika dilihat pada gambar beberapa data point masih cukup jauh jaraknya dengan centroidnya.”

 “Clustering yang baik adalah cluster yang data point-nya saling rapat/sangat berdekatan satu sama lain dan cukup berjauhan dengan objek/data point di cluster yang lain. Jadi, objek dalam satu cluster tidak tersebut berjauhan. Nah, untuk mengukur kualitas dari clustering, kita bisa menggunakan inertia,” jawab Senja langsung.

Aku kembali bertanya karena rasanya masih ada yang janggal. “Memang apa fungsi inertia, Nja?”

“Inertia sendiri mengukur seberapa besar penyebaran object/data point data dalam satu cluster, semakin kecil nilai inertia maka semakin baik. Kita tidak perlu bersusah payah menghitung nilai inertia karena secara otomatis, telah dihitung oleh KMeans( ) ketika algorithm di fit ke dataset. Untuk mengecek nilai inertia cukup dengan **print** fungsi **.inertia\_** dari model yang sudah di fit ke dataset.”

“Kalau begitu,   bagaimana caranya mengetahui nilai K yang paling baik dengan inertia yang paling kecil? Apakah harus trial Error dengan mencoba berbagai jumlah cluster?”

“Benar, kita perlu mencoba beberapa nilai, dan memplot nilai inertia-nya. Semakin banyak cluster maka inertia semakin kecil. Sini deh, saya tunjukkan gambarnya.”

Chart, line chart

Description automatically generated

Meskipun suatu clustering dikatakan baik jika memiliki inertia yang kecil tetapi secara praktikal in real life, terlalu banyak cluster juga tidak diinginkan. Adapun rule untuk memilih jumlah cluster yang optimal adalah dengan memilih jumlah cluster yang terletak pada “elbow” dalam intertia plot, yaitu ketika nilai inertia mulai menurun secara perlahan. Jika dilihat pada gambar maka jumlah cluster yang optimal adalah K = 3.

#### Tugas Praktek

“Kamu coba latihan saja. Aksara. Coba kamu membuat inertia plot untuk melihat apakah K = 5 merupakan jumlah cluster yang optimal. Saya kirim ya berkas latihannya.”

Aku mengangguk pada Senja. Seperti yang tadi ia bilang, materi hari ini benar-benar intens! Email dari Senja sudah masuk dengan cepat:

Untuk membuat inertia plot, silakan memanfaatkan fungsi looping (for):

1. Pertama - tama, buatlah sebuah list kosong yang dinamakan '**inertia'**. List ini akan kita gunakan untuk menyimpan nilai inertia dari setiap nilai K.
2. Gunakan for untuk membuat looping dengan range 1-10. Sebagai index looping gunakan k
3. Di dalam fungsi looping, deklarasikan  **KMeans()**  dengan nama **cluster\_model**dan gunakan n\_cluster = k, dan random\_state = 24
4. Gunakan fungsi .**fit()** dari **cluster\_model** pada 'X'
5. Dari dari **cluster\_model** yang sudah di-fit ke dataset, dapatkan nilai inertia menggunakan **inertia\_** dan deklarasikan sebagai **inertia\_value**
6. Append **inertia\_value**ke dalam list **'inertia'**
7. Setelah iterasi/looping selesai plotlah list **'inertia'** tadi sebagai ordinat-nya dan absica-nya adalah **range(1, 10)**.

Jika berhasil dijalankan setelah menekan  maka akan diperoleh grafik seperti berikut ini

Graphical user interface, text, application

Description automatically generated

Chart, line chart

Description automatically generated

## Chapter 6

### Mini Project

#### Pendahuluan

“Praktikmu cukup memuaskan. Jadi saya sudah bisa melepasmu untuk mengerjakan proyek lagi ya,” pancing Senja.

“Proyek apa nih? tanyaku.

“Ini saya baru dapat kiriman data dari e-commerce kita. Divisi e-commerce kita ingin memprediksi apakah user- user yang sedang mengunjungi halaman website yang baru akan mengklik banner promo (ads) di halaman tersebut atau tidak berdasarkan feature yang ada. Tolong buatkan machine learning model untuk menyelesaikan permasalahan dari e-commerce kita ini ya.”

"Coba cek email yang baru saja saya kirimkan, Aksara," kata Senja.

"Ok!" aku pun segera membuka kotak masuk dari Senja.

#### Case Study: Promos for our e-commerce - Part 1

Aku akan membuat machine learning model untuk menyelesaikan permasalahan dari e-commerce divisi kantor.

Adapun feature - feature dalam dataset ini adalah :

1. 'Daily Time Spent on Site' : lama waktu user mengunjungi site (menit)
2. 'Age' : usia user (tahun)
3. 'Area Income' : rata - rata pendapatan di daerah sekitar user
4. 'Daily Internet Usage' : rata - rata waktu yang dihabiskan user di internet dalam sehari (menit)
5. 'Ad Topic Line' : topik/konten dari promo banner
6. 'City' : kota dimana user mengakses website
7. 'Male' : apakah user adalah Pria atau bukan
8. 'Country' : negara dimana user mengakses website
9. 'Timestamp' : waktu saat user mengklik promo banner atau keluar dari halaman website tanpa mengklik banner
10. 'Clicked on Ad' : mengindikasikan user mengklik promo banner atau tidak (0 = tidak; 1 = klik).

Di proyek ini, aku diharapkan untuk membuat machine learning model sesuai dengan prosedur machine learning yang sudah disharing sebelumnya. Jadi, tahap - tahap yang perlu dilakukan adalah **(langkah ke-1)** terlebih dahulu

1. Data eksplorasi dengan **head()**, **info()**, **describe()**, **shape**

Graphical user interface, text, application, email

Description automatically generated

#### Case Study: Promos for our e-commerce - Part 2

Sekarang mari melanjutkan dengan ekplorasi data untuk langkah ke-2 dan ke-3:

1. Data eksplorasi dengan dengan mengecek korelasi dari setiap feature menggunakan fungsi **corr()**
2. Data eksplorasi dengan mengecek distribusi label menggunakan fungsi **groupby()** dan **size()**

Text

Description automatically generated

#### Case Study: Promos for our e-commerce - Part 3

Di proyek ini, aku akan melanjutkan mengeksplorasi data dengan visualisasi dengan tahap - tahap yang perlu dilakukan adalah (**langkah ke-4**):

1. Data eksplorasi dengan visualisasi:

* Jumlah user dibagi ke dalam rentang usia menggunakan histogram (**hist()**), gunakan **bins = data.Age.nunique()**sebagai argumen. **nunique()** adalah fungsi untuk menghitung jumlah data untuk setiap usia (Age).
* Gunakan **pairplot()** dari seaborn modul untuk menggambarkan hubungan setiap feature.

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated with low confidence

Chart, histogram

Description automatically generated

#### Case Study: Promos for our e-commerce - Part 4

Di bagian proyek (langkah ke-5) ini aku akan mengecek apakah terdapat missing value dari data, jika terdapat missing value dapat dilakukan treatment seperti didrop atau diimputasi dan jika tidak maka dapat melanjutkan ke langkah berikutnya.

1. Cek missing value

Graphical user interface, text, application

Description automatically generated

#### Case Study: Promos for our e-commerce - Part 5

Pada langkah ke-6 ini aku akan melakukan pemodelan dengan Logistic Regression dengan cara seperti berikut:

1. Lakukan pemodelan dengan **Logistic Regression**, gunakan perbandingan 80:20 untuk training vs testing :
   1. Deklarasikan data ke dalam **X**dengan mendrop feature/variabel yang bukan numerik, (type = object) dari **data** (Logistic Regression hanya dapat memproses numerik variabel). Assign Target/Label feature dan assign sebagai **y**
   2. Split **X** dan **y** ke dalam training dan testing dataset, gunakan perbandingan 80:20 dan **random\_state = 42**
   3. Assign classifier sebagai **logreg**, kemudian fit classifier ke **X\_train** dan predict dengan **X\_test**. Print evaluation score.

Text

Description automatically generatedText

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

#### Case Study: Promos for our e-commerce - Part 6

Di langkah terakhir ini atau langkah ke-7 aku akan melihat performansi model dengan menggunakan confusion matrix dan classification report.

1. Print Confusion matrix dan classification report

Graphical user interface, text, application, email

Description automatically generated

A picture containing table

Description automatically generated

#### Penutup/Kesimpulan

**Congratulations!** Akhirnya selesai satu lagi modul **Machine Learning With Python for Beginner**. Berdasarkan materi-materi yang telah kupelajari dan praktekkan dalam modul ini, aku telah mendapatkan pengetahuan (knowledge) dan praktek (skill) yang diantaranya

* Memahami apa itu machine learning dengan jenisnya untuk pemodelan
* Memahami dan mampu melakukan Eksplorasi Data & Data Pre-processing
* Memahami dan mampu melakukan proses-proses Pemodelan dengan Scikit-Learn
* Memahami dan mampu melakukan proses-proses pemodelan dengan menggunakan algoritma pada Supervised Learning
* Memahami dan mampu melakukan proses-proses pemodelan dengan menggunakan algoritma pada Unsupervised Learning
* Mengerjakan mini project yang merupakan integrasi keseluruhan materi dan tentunya materi-materi pada modul-modul sebelumnya untuk menyelesaikan persolan bisnis.

**Keep fighting!**