

Machine Learning



MODULE

Machine Learning

Table of Contents



Hypothesis & Modeling



**Introduction to Supervised
Learning - Regression**



**Model Evaluation- Cross
Validation and Bootstrapping**



**Introduction to Unsupervised
Learning-Classification**

Deskripsi dan Tujuan Pembelajaran

Deskripsi Pembelajaran

Peserta akan belajar mengenai cara menggunakan aplikasi pendukung/*tools* dalam *Data Science*. Pada materi ini peserta akan mendapatkan pembelajaran mandiri melalui beberapa media seperti YouTube maupun modul dan akan dibantu nantinya dengan pelatihan online bersama expert. Kemudian peserta akan belajar mengenai pengubahan dan penggabungan bingkai data baik melalui pembelajaran mandiri, video pembelajaran maupun *Live Session*.

Tugas yang diperoleh oleh peserta berupa suatu *case study* sebuah fiktif *company* yang akan menyediakan data-data yang diperlukan sebagai latihan oleh peserta untuk setiap materinya.

Adapun sub materi yang akan dipelajari peserta ialah sebagai berikut:

1. Machine Learning
2. Hypothesis and Modelling
3. Introduction to Supervised Learning - Regression
4. Model Evaluation- Cross Validation and Bootstrapping
5. Introduction to Unsupervised Learning-Classification
6. Decision Trees
7. Random Forest
8. Gradient Boosting
9. Time Series
10. Forecasting
11. Hyperparameter Tuning
12. Feature Importance

Tujuan Pembelajaran

- Peserta mampu membuat hipotesis untuk model mereka
- Peserta mampu memahami konsep machine learning (supervised dan unsupervised serta mampu melakukan berbagai model machine learning dalam menyelesaikan suatu permasalahan data science
- Peserta dapat meningkatkan konsep pemahaman mengenai prediksi model mereka melalui penyetelan hyperparameter
- Peserta dapat menguraikan wawasan dan menentukan pentingnya fitur dari model yang mereka buat

Silabus

Silabus

3.0 Hypothesis and Modelling

3.1 Introduction to Supervised Learning - Regression

Memahami konsep dasar machine learning, mengetahui data train dan test, label, fitur, dan juga mengetahui masalah yang dapat diselesaikan dengan machine learning

3.2 Model Evaluation- Cross Validation and Bootstrapping

Memberikan detail kepada siswa tentang evaluasi model melalui validasi silang dan bootstrapping

3.3 Introduction to Unsupervised Learning-Classification

Memberikan pemahaman kepada mahasiswa tentang konsep unsupervised machine learning melalui klasifikasi

3.4 Decision Trees

3.5 Random Forest

3.6 Gradient Boosting

3.7 Time Series

3.8 Forecasting

3.9 Hyperparameter Tuning

3.10 Feature Importance

Memberikan detail kepada siswa tentang berbagai model pembelajaran mesin dan cara menggunakannya.

Learning Materials:

1. Books:

- a. Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming 2nd Edition [03 - Python Crash Course A Hands-On, Project-Based Introduction to Programming.pdf](#)
- b. Python Data Science Handbook: Essential Tools for Working with Data [03 - Python Data Science Handbook .pdf](#)
- c. Python For Data Analysis [03 - Python for Data Analysis.pdf](#)
- d. Pandas: Powerful Python Data Analysis Toolkit [03 - Pandas - Powerful Python Data Analysis Toolkit.pdf](#)

2. Articles:

- a. [Everything About Python — Beginner To Advanced](#)
- b. [Python Tutorial for Beginners](#)
- c. [Python Pandas Tutorial: A Complete Introduction for Beginners](#)
- d. [What is Data Analysis?](#)
- e. [The Ultimate Python Tutorial For Beginners :Covering All The Basic Topics, From Variables To Classes & Objects](#)

3. Video:

- a. [Data Analysis with Python: Part 1 of 6 \(Live Course\)](#)
- b. [Learn Python - Full Course for Beginners \[Tutorial\]](#)
- c. [Data Analysis with Python - Full Course for Beginners \(Numpy, Pandas, Matplotlib, Seaborn\)](#)

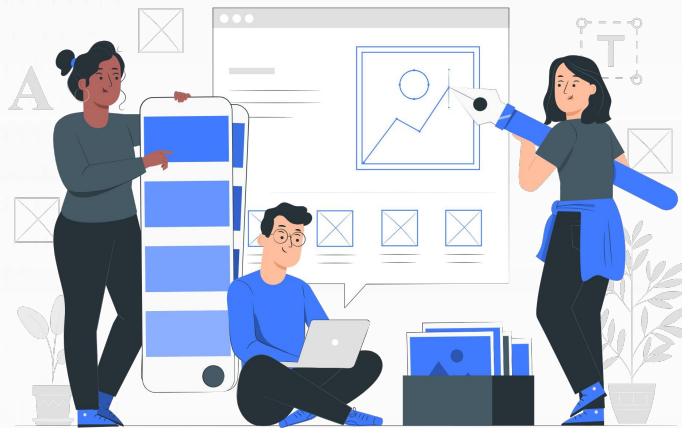
4. Hands-on:

- a. [HackerRank - Python](#)
- b. [Python Tutorial - W3Schools](#)
- c. [Data Analysis with Python](#)

5. Bonus:

- a. [Python for Data Science - A Cheat Sheet for Beginners](#)

MACHINE LEARNING



Pada topik ketiga kita akan mempelajari tahapan pembuatan model dalam proses siklus Data Science. Yang akan dipelajari adalah

- I. Machine Learning
- II. Hypothesis and Modelling
- III. Introduction to Supervised Learning - Regression
- IV. Model Evaluation- Cross Validation and Bootstrapping
- V. Introduction to Unsupervised Learning-Classification
- VI. Decision Trees
- VII. Random Forest
- VIII. Gradient Boosting
- IX. Time Series
- X. Forecasting
- XI. Hyperparameter Tuning
- XII. Feature Importance

Harapannya setelah selesai mempelajari topik ketiga ini kalian bisa memahami cara dan penggunaan tools dalam proses pembuatan model data science.

Tujuan Pembelajaran



- Peserta mampu membuat hipotesis untuk model mereka
- Peserta mampu memahami konsep machine learning (supervised dan unsupervised serta mampu melakukan berbagai model machine learning dalam menyelesaikan suatu permasalahan data science
- Peserta dapat meningkatkan evaluasi model mereka melalui penyetelan hyperparameter
- Peserta dapat menguraikan wawasan dan menentukan pentingnya fitur dari model yang mereka buat

Silabus

3.0 Hypothesis and Modelling

3.1 Introduction to Supervised Learning - Regression

Memahami konsep dasar machine learning, mengetahui data train dan test, label, fitur, dan juga mengetahui masalah yang dapat diselesaikan dengan machine learning

3.2 Model Evaluation- Cross Validation and Bootstrapping

Memberikan detail kepada siswa tentang evaluasi model melalui validasi silang dan bootstrapping

3.3 Introduction to Unsupervised Learning-Classification

Memberikan pemahaman kepada mahasiswa tentang konsep unsupervised machine learning melalui klasifikasi

3.4 Decision Trees

3.5 Random Forest

3.6 Gradient Boosting

3.7 Time Series

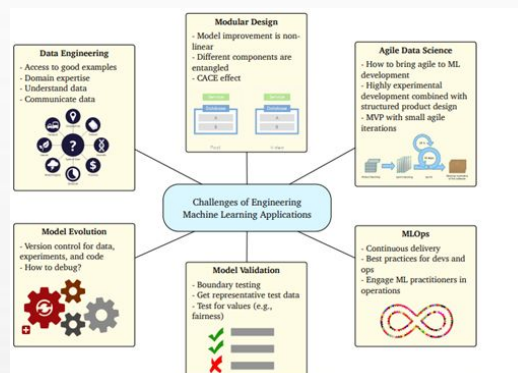
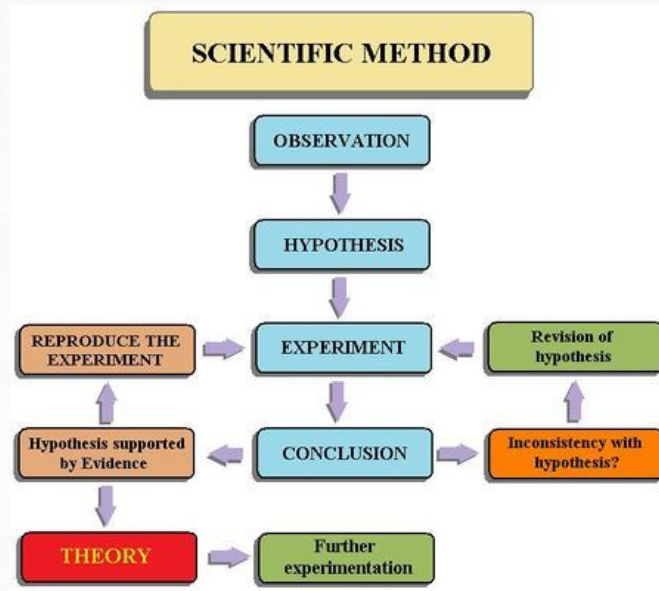
3.8 Forecasting

3.9 Hyperparameter Tuning

3.10 Feature Importance

Memberikan detail kepada siswa tentang berbagai model pembelajaran mesin dan cara menggunakannya.

Hypothesis and Modelling



Hipotesis adalah pernyataan sementara/praduga peneliti terhadap masalah yang dihadapi dalam penelitian. Namun, hipotesis belum bisa dianggap suatu kebenaran apabila belum melalui proses testing. Oleh karena itu, dalam hypothesis testing, sebuah hipotesis dapat ditolak atau gagal ditolak (bukan berarti diterima karena masih belum terbukti kebenarannya).

Prosedur dalam melakukan hypothesis testing antara lain:

1. Merumuskan hipotesis penelitian
2. Menentukan significance level yang akan digunakan
3. Menentukan metode statistik yang digunakan
4. Menentukan kriteria yang digunakan untuk menolak atau tidak menolak hipotesis nol (H_0) sesuai dengan significance level yang ada
5. Membuat kesimpulan berdasarkan uji statistik

Hypothesis and Modeling

Sedangkan modelling atau pemodelan adalah pembangunan representasi dari hal - hal yang ada di dunia nyata, sehingga ide - ide yang terdapat di dalamnya dapat diselidiki. Model adalah sarana untuk membantu pemahaman terhadap masalah yang ada dan komponen dari suatu metode yang digunakan dalam aktivitas pengembangan. Model juga dapat berfungsi sebagai alat bantu komunikasi yang dapat digunakan oleh peneliti untuk menjelaskan sesuatu.

Dalam konteks data, modelling adalah representasi visual dari suatu sistem informasi. Tujuan data modeling adalah untuk mengilustrasikan tipe data yang digunakan dan disimpan dalam sistem, menemukan hubungan antara tipe data, mengelompokkan data, serta mengatur data dari segi format dan atributnya. Sebagai contoh, data modelling dapat diterapkan pada sistem reservasi hotel. Hotel dapat didaftarkan dalam sistem dan pelanggan dapat memesan kamar di hotel pilihan mereka. Aktivitas berikutnya yang meliputi tipe kamar, reservasi, layanan, dan tagihan dapat direpresentasikan melalui diagram sederhana.

3.1 Introduction to Supervised Learning - Regression

Secara literal, supervised learning berarti pembelajaran yang terarah. Supervised learning dapat diilustrasikan seperti gambar di bawah. Siswa melambangkan mesin/model, angka yang ditulis di papan tulis melambangkan input, kemudian cara membaca melambangkan desired output. Pada supervised learning, kita memberikan desired output kepada model, agar model dapat mempelajari tiap pasang pasangan input - desired output. Input dan desired output terdapat dalam training data.

Secara umum, terdapat 2 jenis data dalam supervised learning, yaitu training dataset dan testing dataset. Training dataset adalah himpunan data yang digunakan untuk melatih atau membangun model machine learning. Sedangkan testing dataset adalah himpunan data yang digunakan untuk menguji model setelah proses training selesai. Testing dataset bersifat unseen, artinya model maupun manusia tidak boleh melihat sampel data ini, karena dapat mengakibatkan bias.

Di dalam sebuah data, terdapat fitur, atribut, dan kelas. Fitur adalah variabel independen yang mempengaruhi output. Setiap fitur memiliki atribut nilai dengan tipe data dan range tertentu. Sedangkan kelas atau label adalah variabel dependen atau output.

Salah satu jenis supervised learning adalah regresi. Regresi adalah metode yang dapat digunakan untuk melihat hubungan sebab-akibat antar variabel. Beberapa kegunaan dari regresi adalah untuk mengetahui variabel-variabel yang memiliki pengaruh terhadap variabel dependen. Selain itu, regresi juga bisa digunakan untuk memprediksi output berupa bilangan kontinu. Terdapat beberapa jenis regresi seperti regresi linear, regresi linear berganda, regresi polinomial, dll.

Dalam machine learning, pembuatan model regresi dapat dilakukan menggunakan bahasa pemrograman Python. Secara garis besar, berikut adalah tahapan - tahapan pembuatan model regresi menggunakan Python menggunakan dataset nilai dan IQ:

Introduction to Supervised Learning - Regression

1. Mengimport package dan dataset

```
[16] import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import math

general_data = pd.read_csv('/content/drive/MyDrive/Startup Campus/tes.csv')
general_data.head()
```

	A	B
1	IQ	Nilai
2	100	92
3	110	86
4	90	78
5	105	82
6	100	81
7	100	81
8	102	81
9	100	81
10	101	81
11	121	88
12	97	52
13	98	55

- Setelah package berhasil di import, maka kita akan mengetahui berapa banyak baris pada tabel yang terdapat di data set
- `general_data.shape` : Melihat jumlah dataset ketikkan pada script program

```
[17] general_data.shape
```

```
(12, 2)
```

- Mengkonfirmasi jumlah dari dataset yang digunakan pada data set

```
[18] print('#jumlah dataset saya : ' +str(len(general_data)))
```

```
#jumlah dataset saya : 12
```

- `general_data.describe()` : Melakukan analisis deskriptif secara otomatis. Lalu, akan muncul mean, standard deviasi, min, max, dll

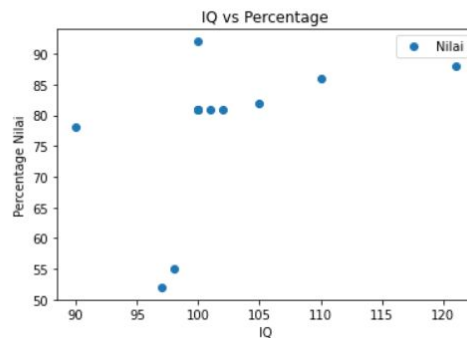
```
[19] general_data.describe()
```

	IQ	Nilai
count	12.000000	12.000000
mean	102.000000	78.166667
std	7.603827	12.156804
min	90.000000	52.000000
25%	99.500000	80.250000
50%	100.000000	81.000000
75%	102.750000	83.000000
max	121.000000	92.000000

Introduction to Supervised Learning - Regression

6. Script ini digunakan untuk menampilkan plot. Koordinat X adalah Jumlah IQ sedangkan koordinat Y adalah Jumlah Nilai

```
[20] general_data.plot(x='IQ',y='Nilai',style='o')  
      plt.title('IQ vs Percentage')  
      plt.xlabel('IQ')  
      plt.ylabel('Percentage Nilai')  
      plt.show()
```



7. Sebelum ke tahap analisis regresi linear, import sklearn pada program, kemudian ketik script program. Data dibagi menjadi 2 bagian data 80% untuk data training, dan 20% untuk data test, sehingga padatest_size=0.2

```
[21] x = general_data.iloc[:, :-1].values  
      y = general_data.iloc[:, 1].values
```

```
[22] from sklearn.model_selection import train_test_split  
      x_train, x_test, y_train, y_test = train_test_split(  
          x, y, test_size=0.2, random_state=0  
      )
```

8. Setelah data di split menjadi 80% training set dan 20% test set, selanjutnya masuk ke analisis regresi linear, ketikan script seperti dibawah. Selanjutnya akan muncul intercept dan coef, 2 parameter ini digunakan untuk menentukan model matematika yang baru

```
[23] from sklearn.linear_model import LinearRegression  
      regressor = LinearRegression()  
      regressor.fit(x_train, y_train)
```

```
LinearRegression()
```

```
[24] print(regressor.intercept_)  
      print(regressor.coef_)
```

```
23.018155410312282  
[0.55610022]
```

Introduction to Supervised Learning - Regression

9. Dengan menggunakan model matematika yang baru, model tersebut dapat digunakan untuk memprediksi hasil nilai berdasarkan jumlah IQ

```
[25] y_pred = regressor.predict(x_test)
```

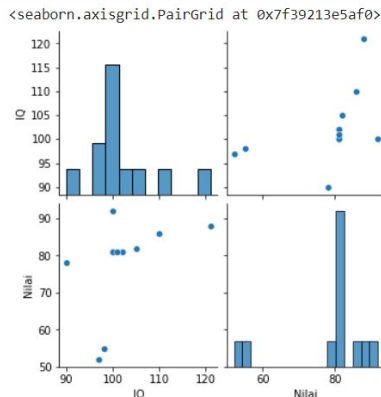
10. Untuk menaruh hasil prediksi IQ dan nilai dapat menggunakan script seperti dibawah ini

```
[26] df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

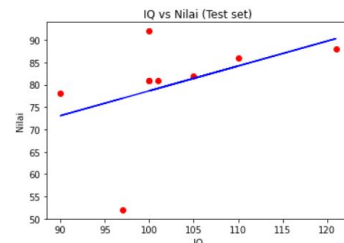
	Actual	Predicted
0	81	79.740378
1	55	77.515977
2	81	78.628177

11. Tampilkan hasil test set dan training set serta garis regresi dalam satu plot

```
[27] sns.pairplot(general_data)
```



```
[28] plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color = 'blue')
plt.title('IQ vs Nilai (Test set)')
plt.xlabel('IQ')
plt.ylabel('Nilai')
plt.show()
```



12. Berdasarkan hasil diatas. Maka dapat didapatkan rumus sebagai berikut

$$IQ = 23,018 + 0,556 * Nilai$$

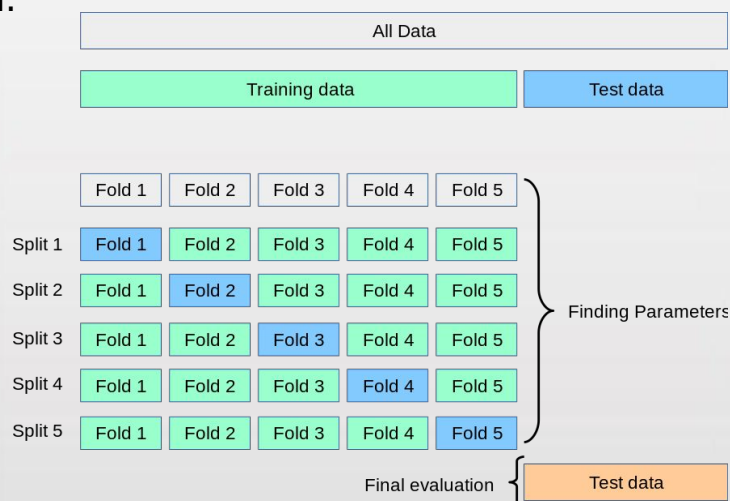
[Google Collab](#)

3.2 Model Evaluation- Cross Validation and Bootstrapping

Dalam machine learning, tujuan utama suatu model machine learning adalah mengoptimalkan class-assignment probability atau probabilitas masing - masing kelas untuk suatu input. Kemampuan model untuk mengoptimalkan class-assignment probability ini diukur dalam evaluasi kinerja model. Terdapat berbagai metode evaluasi model, beberapa diantaranya adalah cross validation dan bootstrapping.

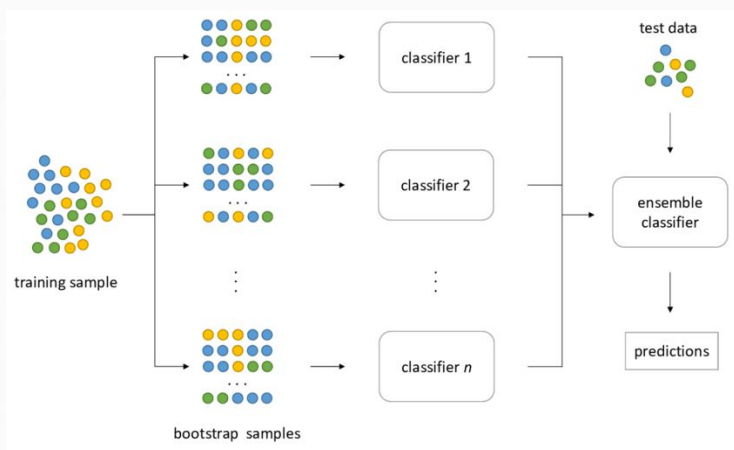
Pada saat mengevaluasi parameter - parameter dalam model supervised learning, terdapat resiko overfitting pada testing dataset karena parameter - parameter tersebut dapat terus ditingkatkan hingga model bekerja secara optimal. Hal ini dapat menyebabkan informasi di dalam testing dataset 'bocor' ke dalam model dan metrik evaluasi tidak lagi menampilkan performa model secara general. Untuk mengatasi masalah ini, diperlukan adanya tambahan dataset baru bernama validation dataset. Sehingga setelah model dikembangkan menggunakan training dataset, evaluasi pertama akan dilaksanakan menggunakan validation dataset, dan evaluasi terakhir dilaksanakan menggunakan testing dataset. Akan tetapi, pembagian data menjadi 3 dataset dapat menyebabkan penurunan jumlah sampel training secara drastis, terutama pada dataset dengan ukuran yang tidak terlalu besar.

Solusi atas permasalahan tersebut adalah sebuah prosedur bernama cross validation (CV). Testing dataset tetap akan diperlukan untuk evaluasi model terakhir. Akan tetapi, validation dataset tidak lagi diperlukan. Melalui k - fold CV, training dataset akan dipisah menjadi k set yang lebih kecil. Pertama - tama, model akan dibangun menggunakan k-1 data fold sebagai training dataset. Kemudian, model divalidasi menggunakan bagian lain dari dataset. Ukuran performa dari k-fold CV adalah rata - rata dari nilai yang didapatkan selama pengulangan.



Model Evaluation- Cross Validation and Bootstrapping

Bootstrapping adalah metode evaluasi yang mirip dengan *cross validation*. Perbedaannya terletak pada bagaimana *dataset* dapat terbentuk. Pada *bootstrapping*, sampel dilakukan dengan menggunakan metode *sampling with replacement*. Sampel didapatkan dari *original dataset* dalam satu waktu dan dikembalikan setelah sampel tersebut terpilih. Hal ini memungkinkan sampel yang sama dapat muncul dalam beberapa dataset.

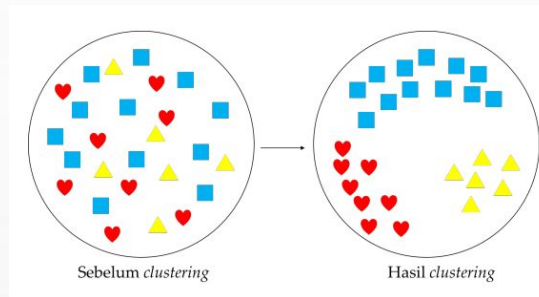


Penjelasan mengenai *cross bootstrapping* dapat dilihat pada link [berikut](#).

3.3 Introduction to Unsupervised Learning-Classification

Unsupervised Learning adalah cabang lainnya dari machine learning. Jika pada supervised learning terdapat 'guru' yang mengajar, pada unsupervised learning, tidak ada 'guru' yang mengajar. Hal ini berarti tidak ada desired output yang diberikan terhadap model. Sehingga model dibangun untuk dapat mengenali pola yang ada di dalam dataset tanpa adanya desired output yang diberikan. Berbeda dengan supervised learning, pada unsupervised learning tidak terdapat training dataset dan testing dataset.

Terdapat beberapa kegunaan dari unsupervised learning, salah satunya adalah klasifikasi. Secara umum, klasifikasi memang termasuk dalam supervised learning. Proses klasifikasi tanpa desired output yang dilakukan oleh unsupervised learning disebut clustering (klasterisasi). Dalam clustering, setiap fitur tidak berkorespondensi dengan kelas tertentu.



Terdapat beberapa metode dalam clustering, salah satunya adalah k means clustering. Berikut adalah garis besar tahapan - tahapan clustering menggunakan k means clustering dengan menggunakan dataset online retail:

1. Mengimport beberapa Library Python untuk kebutuhan dataframe dan menginput dataset
2. Menentukan nilai k. Pada langkah ini, akan dikonfigurasi dan ditentukan

```
# Library untuk dataframe dan visualisasi
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# Import library untuk Clustering
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```
# Load dataset OnlineRetail
df = pd.read_csv('OnlineRetail.csv', sep=",", encoding="ISO-8859-1",
header=0)
df.head()
```


Introduction to Unsupervised Learning-Classification

- Menentukan nilai k. Pada langkah ini, akan dikonfigurasi dan ditentukan nilai inisiasi k (n_clusters) sebesar 4 cluster

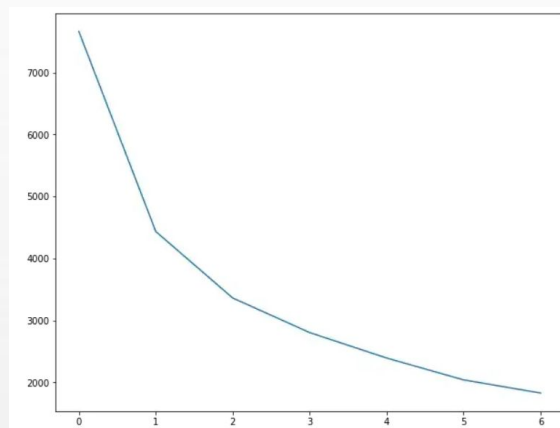
```
kmeans = KMeans(n_clusters=4, max_iter=50)
kmeans.fit(rfm_df_scaled)
```

- Mencari jumlah optimal dari cluster (k) dengan menggunakan metode Elbow Curve. Elbow Curve merupakan salah satu metode yang bisa digunakan untuk menemukan jumlah optimal dari cluster (k), yang langkah-langkah pengerjaan adalah sebagai berikut.

```
# Elbow-curve/SSD
ssd = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)

    ssd.append(kmeans.inertia_)

# plot the SSDs for each n_clusters
plt.plot(ssd)
```



Pada metode elbow curve, lokasi 'tikungan' yang terbentuk di plot, pada umumnya dianggap sebagai indikator jumlah cluster yang tepat. Akan tetapi nilai k optimal yang diperoleh dari metode elbow curve, sering kali bersifat "ambigu" atau belum pasti akan menghasilkan jumlah cluster (k) yang optimal. Oleh karena itu, langkah selanjutnya akan digunakan Silhouette Analysis guna mencari nilai k optimal.

- Silhouette analysis adalah pengukuran seberapa dekat setiap titik pada sebuah cluster dengan titik - titik data lain di cluster-nya. Semakin tinggi nilai rata - rata dari silhouette, menunjukkan suatu peng-cluster-an yang baik. Nilai dari Silhouette ada diantara -1 sampai dengan 1. Jika nilainya mendekati angka 1, maka titik data akan sangat mirip dengan titik data lainnya di cluster yang sama

Introduction to Unsupervised Learning-Classification

```
# Silhouette Analysis
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:

    # Initialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)
    cluster_labels = kmeans.labels_

    # Silhouette Score
    silhouette_avg = silhouette_score(rfm_df_scaled, cluster_labels)
    print("For n_clusters={0}, the silhouette score is
    {1}".format(num_clusters, silhouette_avg))
```

```
For n_clusters=2, the silhouette score is 0.5411246404292333
For n_clusters=3, the silhouette score is 0.5084896296141937
For n_clusters=4, the silhouette score is 0.4816217519322445
For n_clusters=5, the silhouette score is 0.4662700564189704
For n_clusters=6, the silhouette score is 0.41707960376211345
For n_clusters=7, the silhouette score is 0.4177054772702703
For n_clusters=8, the silhouette score is 0.39347414421132215
```

Berdasarkan output tersebut, dapat disimpulkan bahwa untuk $n_clusters = 2$ menghasilkan nilai silhouette yang tinggi

```
# Final model with k=2
kmeans = KMeans(n_clusters=2, max_iter=50)
kmeans.fit(rfm_df_scaled)
```

5. Menetapkan label cluster untuk setiap titik data sebagai berikut

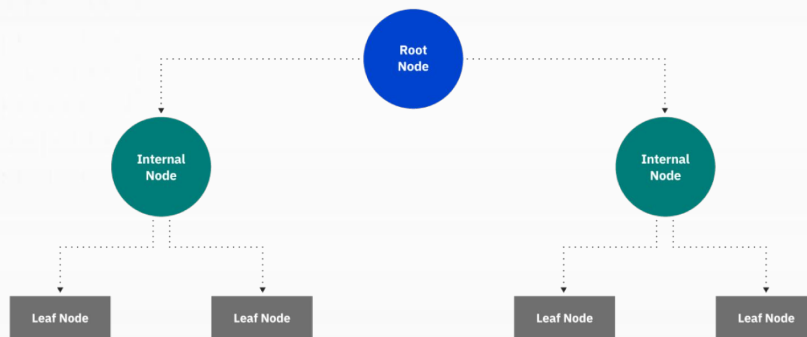
```
# Assign the label
rfm['Cluster_Id'] = kmeans.labels_
rfm.head()
```

	CustomerID	Monetary	Frequency	Recency	Cluster_Id
0	12346.0	0.00	2	325	1
1	12347.0	4310.00	182	1	0
2	12348.0	1797.24	31	74	1
3	12349.0	1757.55	73	18	1
4	12350.0	334.40	17	309	1

3.4 Decision Trees

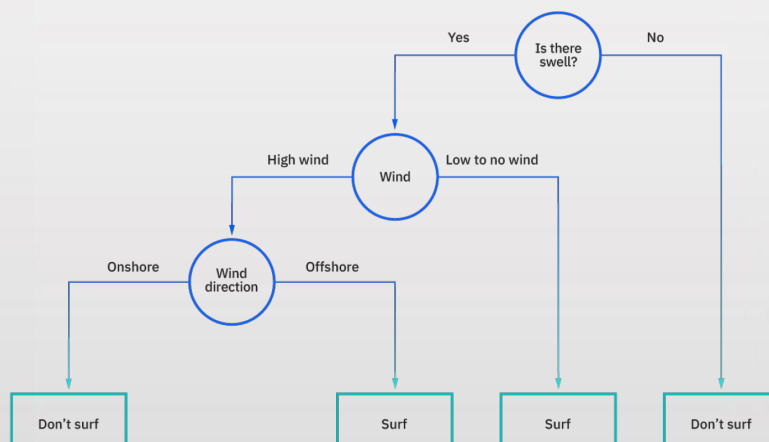
Sebuah pohon memiliki banyak analogi dalam kehidupan nyata, dan ternyata pohon tersebut telah memengaruhi area machine learning yang luas, mencakup klasifikasi dan regresi. Dalam analisis keputusan, decision trees dapat digunakan untuk merepresentasikan keputusan dan pengambilan keputusan secara visual dan eksplisit. Seperti namanya, ini menggunakan model keputusan seperti pohon.

Decision trees adalah algoritma pembelajaran terawasi non-parametrik, yang digunakan untuk tugas klasifikasi dan regresi. Decision trees memiliki hierarki, struktur pohon, yang terdiri dari simpul akar, cabang, simpul internal dan simpul daun.



Seperti yang Anda lihat dari diagram di atas, decision trees dimulai dengan simpul akar (root node), yang tidak memiliki cabang masuk. Cabang keluar dari simpul akar kemudian masuk ke simpul internal (internal node), juga dikenal sebagai simpul keputusan. Berdasarkan fitur yang tersedia, kedua tipe simpul melakukan evaluasi untuk membentuk himpunan bagian yang homogen, yang dilambangkan dengan simpul daun (leaf node), atau simpul terminal. Node daun (leaf node) mewakili semua kemungkinan hasil dalam kumpulan data.

Sebagai contoh, bayangkan jika kita mencoba menilai apakah kita harus berselancar atau tidak, Anda dapat menggunakan aturan keputusan berikut untuk membuat pilihan:



Decision Trees

Jenis struktur flowchart ini juga menciptakan representasi pengambilan keputusan yang mudah dicerna, memungkinkan kelompok yang berbeda di seluruh organisasi untuk lebih memahami mengapa keputusan dibuat.

Pembelajaran decision trees menggunakan strategi membagi dan untuk mengidentifikasi titik perpecahan yang optimal dalam sebuah pohon. Proses pemisahan ini kemudian diulangi secara top-down, rekursif hingga semua, atau sebagian besar telah diklasifikasikan di bawah label kelas tertentu. Apakah semua poin data diklasifikasikan sebagai kumpulan homogen atau tidak sangat bergantung pada kompleksitas decision trees.

Keuntungan

- Mudah diinterpretasikan: Logika Boolean dan representasi visual decision trees membuatnya lebih mudah dipahami dan dikonsumsi. Sifat hierarki decision trees juga memudahkan untuk melihat atribut mana yang paling penting.
- Sedikit atau tidak diperlukan persiapan data: Decision trees memiliki sejumlah karakteristik, yang membuatnya lebih fleksibel daripada pengklasifikasi lainnya. Hal tersebut dapat menangani berbagai tipe data — misalnya nilai diskrit atau kontinu, dan nilai kontinu dapat diubah menjadi nilai kategoris melalui penggunaan ambang batas.
- Selain itu, hal tersebut juga dapat menangani nilai dengan nilai yang hilang, yang dapat menjadi masalah bagi pengklasifikasi lainnya, seperti Naïve Bayes.
- Lebih fleksibel: Decision trees dapat dimanfaatkan untuk tugas klasifikasi dan regresi, membuatnya lebih fleksibel daripada beberapa algoritma lainnya.

Kekurangan

- Rentan terhadap overfitting: Decision trees yang kompleks cenderung overfit dan tidak menggeneralisasi data baru dengan baik. Skenario ini dapat dihindari melalui proses pre-pruning atau post-pruning. Pre-pruning menghentikan pertumbuhan pohon ketika tidak ada cukup data sementara post-pruning menghilangkan sub-pohon dengan data yang tidak memadai setelah konstruksi pohon.
- Estimator varian tinggi: Variasi kecil dalam data dapat menghasilkan pohon keputusan yang sangat berbeda. Bagging, atau rata-rata estimasi, dapat menjadi metode untuk mengurangi varian dari decision trees. Namun, pendekatan ini terbatas karena dapat menyebabkan prediktor yang sangat berkorelasi.
- Lebih mahal: Mengingat bahwa decision trees mengambil pendekatan pencarian serakah selama konstruksi, mereka bisa lebih mahal untuk dilatih dibandingkan dengan algoritma lainnya.
- Tidak didukung sepenuhnya di scikit-learn: Scikit-learn adalah pustaka pembelajaran mesin populer yang berbasis di Python. Meskipun pustaka ini memiliki modul decision trees, implementasi saat ini tidak mendukung variabel kategorikal.

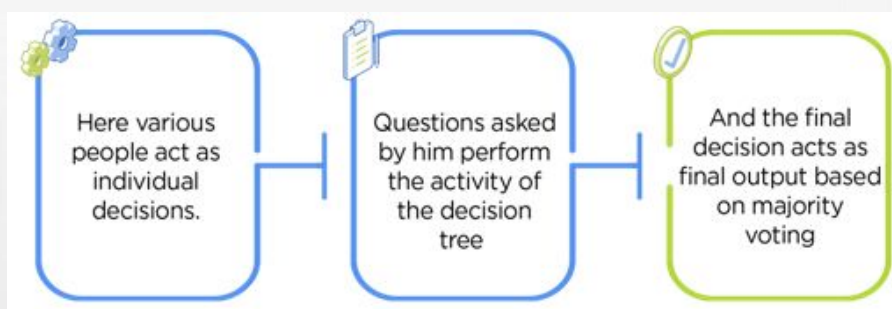
3.5 Random Forest

Random forest adalah Algoritma Supervised Machine Learning yang digunakan secara luas dalam masalah Klasifikasi dan Regresi. Hal tersebut membangun decision trees pada sampel yang berbeda dan mengambil suara mayoritas untuk klasifikasi dan rata-rata dalam kasus regresi.

Salah satu fitur terpenting dari Algoritma Random forest adalah dapat menangani kumpulan data yang berisi variabel kontinu seperti kasus regresi dan variabel kategori seperti dalam kasus klasifikasi. Hal tersebut membuahkan hasil yang lebih baik untuk masalah klasifikasi.

Dalam analogi kehidupan nyata untuk memahami konsep ini lebih jauh. Contohnya seorang siswa bernama X ingin memilih mata pelajaran setelah 10+2, dan dia bingung tentang pilihan mata pelajaran berdasarkan keahliannya. Jadi dia memutuskan untuk berkonsultasi dengan berbagai orang seperti sepupunya, guru, orang tua, mahasiswa sarjana, dan pekerja. Dia mengajukan berbagai pertanyaan kepada mereka seperti mengapa dia harus memilih, peluang kerja, biaya kursus, dll. Akhirnya, setelah berkonsultasi dengan berbagai orang tentang kursus, dia memutuskan untuk mengambil kursus yang disarankan oleh kebanyakan orang.

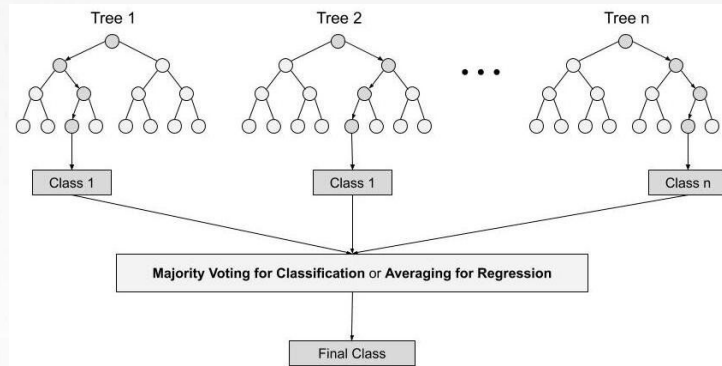
Salah satu fitur terpenting dari Algoritma random forest adalah dapat menangani kumpulan data yang berisi variabel kontinu seperti dalam kasus regresi dan variabel kategori seperti dalam kasus klasifikasi. Hal tersebut menghasilkan hasil yang lebih baik untuk masalah klasifikasi.



Langkah-langkah yang terlibat dalam algoritma random forest:

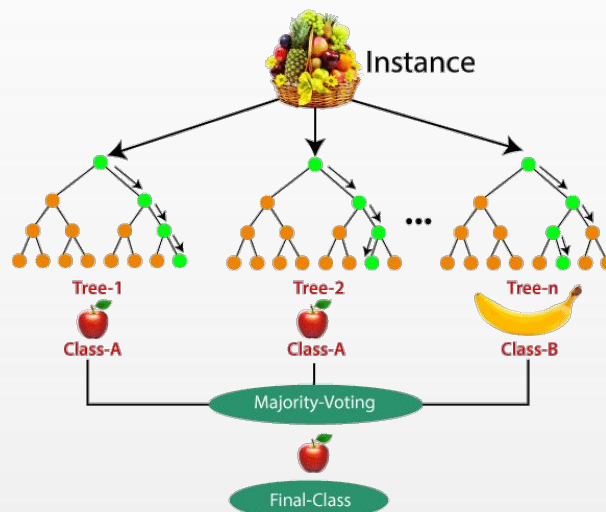
- Langkah 1: Di random forest n sejumlah catatan acak diambil dari kumpulan data yang memiliki sejumlah k catatan.
- Langkah 2: Individual random forest dibangun untuk setiap sampel.
- Langkah 3: Setiap random forest akan menghasilkan output.
- Langkah 4: Hasil akhir dipertimbangkan berdasarkan Pemungutan Suara Mayoritas atau Rata-Rata untuk Klasifikasi dan regresi masing-masing.

Random Forest



Sebagai contoh: pertimbangkan keranjang buah sebagai data seperti yang ditunjukkan pada gambar di bawah ini. Sekarang sejumlah n sampel diambil dari keranjang buah dan *random forest* individual dibuat untuk setiap sampel.

Setiap *random forest* akan menghasilkan *output* seperti yang ditunjukkan pada gambar. Hasil akhir dipertimbangkan berdasarkan suara terbanyak. Pada gambar di bawah ini kita dapat melihat bahwa *random forest* mayoritas menghasilkan apel jika dibandingkan dengan pisang, sehingga hasil akhir diambil sebagai apel.



Fitur penting random forest:

1. Diversity - Tidak semua atribut/variabel/fitur dipertimbangkan saat membuat satu pohon, setiap pohon berbeda.
2. Kebal terhadap dimensi - Karena setiap pohon tidak mempertimbangkan semua fitur.
3. Parallelization - Setiap pohon dibuat secara independen dari data dan atribut yang berbeda. Ini berarti kita dapat menggunakan CPU sepenuhnya untuk membangun hutan acak.
4. Train Test split - Dalam random forest kita tidak perlu memisahkan data untuk latihan dan pengujian karena akan selalu ada 30% data yang tidak terlihat oleh pohon keputusan.
5. Stabilitas - Stabilitas muncul karena hasil didasarkan pada voting mayoritas/rata-rata.

Random Forest

Keuntungan

1. Dapat digunakan dalam masalah klasifikasi dan regresi.
2. Untuk memecahkan masalah overfitting karena output didasarkan pada voting mayoritas atau rata-rata.
3. Performanya baik meskipun data berisi nilai null/hilang.
4. Setiap random trees yang dibuat bersifat independen satu sama lain sehingga menunjukkan sifat paralelisasi.
5. Sangat stabil karena diambil rata-rata jawaban yang diberikan oleh banyak pohon.
6. Mempertahankan keragaman karena semua atribut tidak dipertimbangkan saat membuat setiap pohon keputusan meskipun tidak benar dalam semua kasus.
7. Kebal terhadap dimensi. Karena setiap pohon tidak mempertimbangkan semua atribut, ruang fitur berkurang.
8. Tidak perlu memisahkan data menjadi pelatihan dan pengujian karena akan selalu ada 30% data yang tidak terlihat oleh pohon keputusan yang dibuat dari bootstrap.

Kekurangan

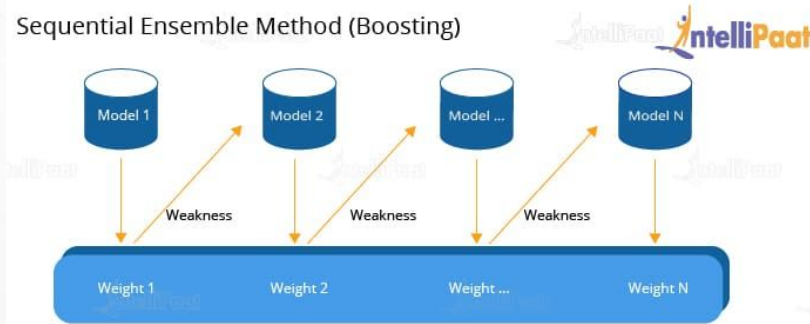
1. Random forest sangat kompleks jika dibandingkan dengan decision trees di mana keputusan dapat dibuat dengan mengikuti jalur pohon tersebut.
2. Waktu pelatihan lebih banyak dibandingkan dengan model lain karena kerumitannya. Setiap kali harus membuat prediksi, setiap decision trees harus menghasilkan keluaran untuk data masukan yang diberikan.

3.6 Gradient Boosting

Boosting adalah teknik untuk menggabungkan pembelajar yang lemah dan mengubahnya menjadi pembelajar yang kuat dengan bantuan algoritma Machine Learning. Teknik ini menggunakan pembelajaran ansambel untuk meningkatkan akurasi model. Ensemble learning adalah teknik untuk meningkatkan akurasi model Machine Learning. Ada dua jenis pembelajaran ansambel:

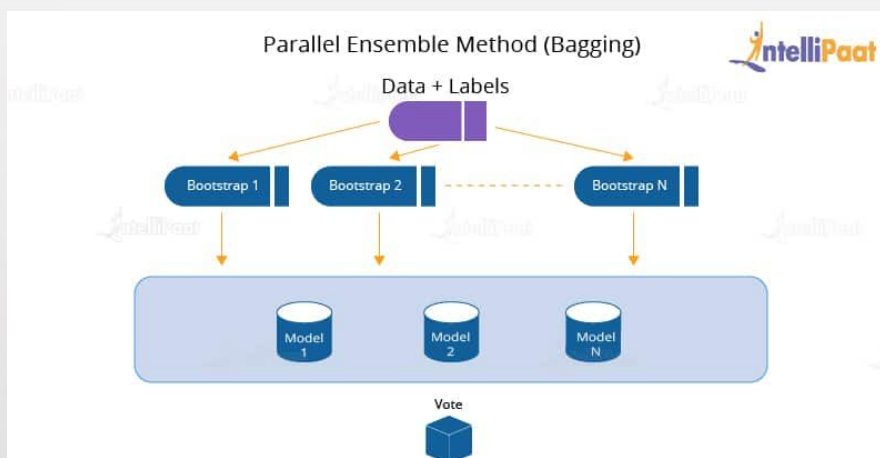
1. *Sequential Ensemble Learning*

Ini adalah teknik peningkatan di mana output dari yang lemah diasosiasikan secara berurutan selama fase pelatihan. Performa model ditingkatkan dengan cara memberikan bobot yang lebih tinggi pada sampel yang salah diklasifikasikan. Algoritma AdaBoost adalah contohnya.



2. *Parallel Ensemble Learning*

Ini adalah teknik yang di mana output dari yang lemah dihasilkan secara paralel. Hal tersebut mengurangi kesalahan dengan merata-ratakan output dari yang lemah. Algoritma random forest adalah contoh pembelajaran ansambel paralel.



Gradient Boosting

Mekanisme Boosting Algorithm

Boosting adalah membuat algoritma generik dengan mempertimbangkan prediksi mayoritas yang lemah. Hal ini membantu dalam meningkatkan kekuatan prediksi model machine learning. Hal tersebut dilakukan dengan melatih serangkaian model yang lemah. Berikut adalah langkah-langkah yang menunjukkan mekanisme algoritma boosting:

1. Membaca data
2. Memberi bobot pada pengamatan
3. Identifikasi salah tafsir (prediksi salah)
4. Menugaskan prediksi yang salah, bersama dengan bobot yang lebih tinggi, kepada pembelajar berikutnya
5. Iterasi Langkah 2 hingga mendapatkan output diklasifikasikan dengan benar

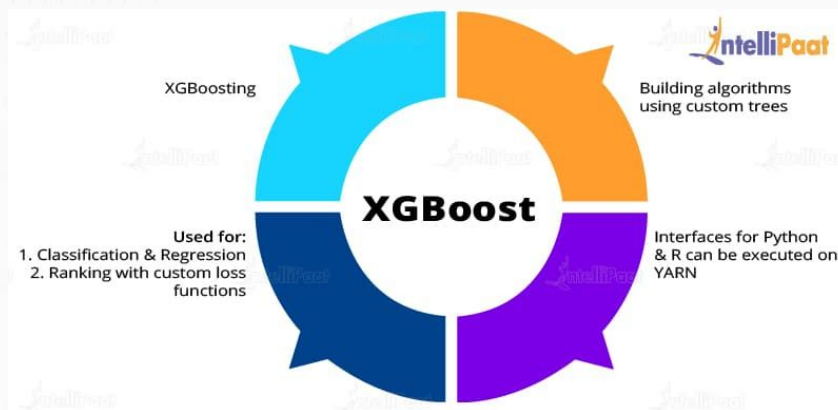
Dalam machine learning, kita dapat menggunakan peningkatan gradien untuk menyelesaikan masalah klasifikasi dan regresi. Teknik pembelajaran ansambel ini berurutan di mana kinerja model meningkat selama iterasi. Metode ini menciptakan model secara bertahap. Hal tersebut menyimpulkan bahwa model tersebut mengaktifkan optimalisasi fungsi kerugian yang dapat dibedakan secara absolut. Saat menambahkan setiap yang lemah, model baru dibuat yang memberikan estimasi variabel respons yang lebih tepat.

Algoritma peningkatan gradien memerlukan komponen di bawah ini untuk berfungsi:

1. Fungsi kerugian: Untuk mengurangi kesalahan dalam prediksi, kita perlu mengoptimalkan fungsi kerugian. Tidak seperti di AdaBoost, hasil yang salah tidak diberikan bobot yang lebih tinggi dalam peningkatan gradien. Hal tersebut mengurangi fungsi kerugian dengan merata-ratakan output dari pembelajar yang lemah.
2. Weak learner: Dalam peningkatan gradien, kami meminta pembelajar yang lemah untuk membuat prediksi. Untuk mendapatkan nilai riil sebagai keluaran, kami menggunakan pohon regresi. Untuk mendapatkan titik pisah yang paling sesuai, kami membuat pohon dengan cara serakah, karena model ini terlalu sesuai dengan dataset.
3. Model aditif: Dalam peningkatan gradien, kita mencoba mengurangi kerugian dengan menambahkan pohon keputusan. Selain itu, kita dapat meminimalkan tingkat kesalahan dengan mengurangi parameter. Jadi, dalam hal ini, didesain model sedemikian rupa sehingga penambahan pohon tidak mengubah pohon yang ada.

Gradient Boosting

Algoritma XGBoost adalah versi lanjutan dari algoritma gradien booster. Hal tersebut pada dasarnya dirancang untuk meningkatkan kinerja dan kecepatan model Machine Learning. Selain itu, XGBoosting library yang memberi framework machine learning untuk gradien booster untuk berbagai bahasa seperti R, Python, Java, dll.



3.7 Time Series

Time series adalah urutan berbagai titik data yang terjadi secara berurutan untuk jangka waktu tertentu

Tujuan:

1. Untuk memahami cara kerja time series, faktor apa yang mempengaruhi variabel tertentu pada titik waktu yang berbeda.
2. Analisis time series akan memberikan konsekuensi dan wawasan fitur dari kumpulan data yang diberikan yang berubah dari waktu ke waktu.
3. Mendukung untuk menurunkan prediksi nilai masa depan dari variabel time series.
4. Asumsi: Ada satu dan satu-satunya asumsi yang "stasioner", yang berarti asal mula waktu, tidak mempengaruhi sifat proses di bawah faktor statistik.







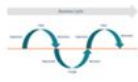
Bagaimana cara menganalisis Time Series?

1. Mengumpulkan data dan membersihkannya
2. Mempersiapkan Visualisasi sehubungan dengan waktu vs fitur utama
3. Mengamati stasioneritas deret
4. Mengembangkan grafik untuk memahami sifatnya.
5. Pembuatan model – AR, MA, ARMA dan ARIMA
6. Mengekstrak wawasan dari prediksi

Komponen Analisis Time Series

1. Tren: Di mana tidak ada interval tetap dan perbedaan apa pun dalam kumpulan data yang diberikan adalah garis waktu yang berkelanjutan. Trennya adalah Tren Negatif atau Positif atau Null
2. Musiman: Di mana interval reguler atau tetap bergeser dalam kumpulan data dalam garis waktu yang berkelanjutan. Apakah kurva lonceng atau gigi gergaji
3. Siklus: Di mana tidak ada interval tetap, ketidakpastian dalam gerakan dan polanya
4. Irregularity: Situasi/peristiwa/skenario tak terduga dan lonjakan dalam rentang waktu singkat.

Time Series

	Trend	Seasonality	Cyclical	Irregularity
Time	Fixed Time Interval	Fixed Time Interval	Not Fixed Time Interval	Not Fixed Time Interval
Duration	Long and Short Term	Short Term	Long and Short Term	Regular/Irregular
Visualization				
Nature - I	Gradual	Swings between Up or Down	Repeating Up and Down	Errored or High Fluctuation
Nature - II	Upward/Down Trend	Pattern repeatable	No fixed period	Short and Not repeatable
Prediction Capability	Predictable	Predictable	Challenging	Challenging
Market Model				Highly random/Unforeseen Events – along with white noise.

Designed by Author (Shanthababu)

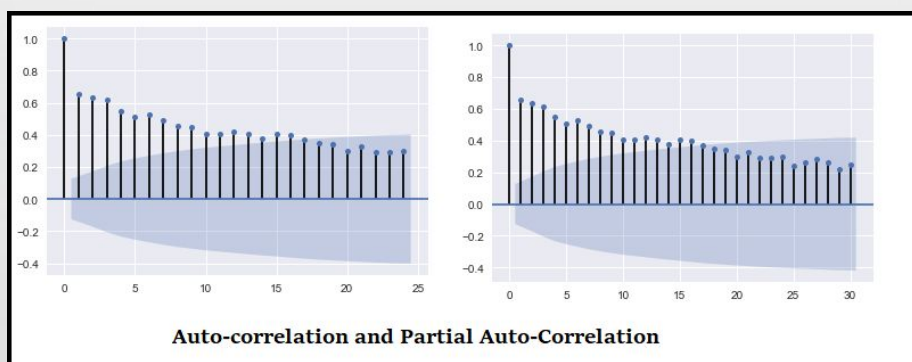
Time Series in Data

Saat berurusan dengan time series dalam Data Science dan Pembelajaran Mesin, ada beberapa opsi model yang tersedia. Di mana model Autoregressive-Moving-Average (ARMA) dengan $[p, d, \text{dan } q]$.

- $P \Rightarrow$ kelambatan autoregresif
- $q \Rightarrow$ kelambatan rata-rata bergerak
- $d \Rightarrow$ perbedaan urutan

Sebelum kita mengenal Arima, sebaiknya pahami terlebih dahulu istilah-istilah di bawah ini.

- Autocorrelation Function (ACF): ACF digunakan untuk menunjukkan dan seberapa mirip suatu nilai dalam time series tertentu dan nilai sebelumnya. (ATAU) Ini mengukur tingkat kesamaan antara rangkaian waktu tertentu dan versi tertinggal dari rangkaian waktu tersebut pada interval berbeda yang kami amati. Pustaka Python Statsmodels menghitung autokorelasi. Ini digunakan untuk mengidentifikasi serangkaian tren dalam kumpulan data yang diberikan dan pengaruh nilai yang diamati sebelumnya pada nilai yang diamati saat ini.
- Partial Autocorrelation (PACF): PACF mirip dengan Fungsi Auto-Correlation dan sedikit sukar untuk dipahami. Itu selalu menunjukkan korelasi urutan dengan dirinya sendiri dengan sejumlah unit waktu per urutan urutan di mana hanya efek langsung yang ditunjukkan, dan semua efek perantara lainnya dihapus dari time series yang diberikan.



Time Series

Tipe Data Time Series

- Stasioner: Kumpulan data harus mengikuti aturan praktis di bawah ini, tanpa memiliki komponen Deret waktu Tren, Musiman, Siklus, dan Irregularitas.
 - A. Nilai MEAN dari mereka harus benar-benar konstan dalam data selama analisis
 - B. VARIANSI harus konstan sehubungan dengan kerangka waktu
 - C. COVARIANCE mengukur hubungan antara dua variabel.
- Non- Stasioner: Ini kebalikan dari Stasioner.

Auto-Regressive model

Auto-Regressive adalah model sederhana, yang memprediksi kinerja masa depan berdasarkan kinerja masa lalu. terutama digunakan untuk peramalan, ketika ada beberapa korelasi antara nilai dalam deret waktu tertentu dan nilai yang mendahului dan berhasil (bolak-balik).

Model AR adalah model Regresi Linear, yang menggunakan variabel tertinggal sebagai masukan. Model Regresi Linier dapat dengan mudah dibangun menggunakan library scikit-learn dengan menunjukkan input yang akan digunakan. Pustaka Statsmodels digunakan untuk menyediakan fungsi khusus model autoregresi di mana Anda harus menentukan nilai lag yang sesuai dan melatih model. Itu disediakan di kelas AutoReg untuk mendapatkan hasilnya, menggunakan langkah-langkah sederhana

- Membuat model AutoReg()
- Panggil fit() untuk melatihnya di dataset kami.
- Mengembalikan objek AutoRegResults.

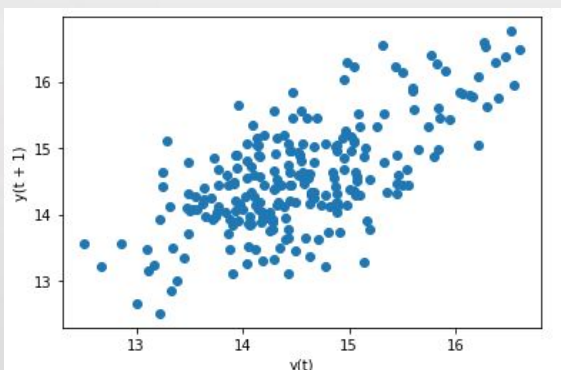
Setelah tepat, buat prediksi dengan memanggil fungsi predict

$$Y_t = C + b_1 Y_{t-1} + b_2 Y_{t-2} + \dots + b_p Y_{t-p} + E_t$$

Key Parameters:

- p =past values
- Y_t =Function of different past values
- E_t =errors in time
- C =intercept

Mari kita periksa, mengingat kumpulan data atau deret waktu acak atau tidak



Observasi: Terlihat acak dan tersebar.

3.8 Forecasting

Forecasting adalah bagian luas dari berbagai industri. Karena itu, kisaran model forecasting juga sangat besar dengan masing-masing model memiliki kelebihan dan kekurangannya sendiri. Pada subab ini, akan membahas beberapa model forecasting dasar dan sederhana. Terlepas dari kesederhanaannya, model ini dapat menawarkan hasil yang baik dalam praktiknya dan memberikan dasar yang baik untuk iterasi.

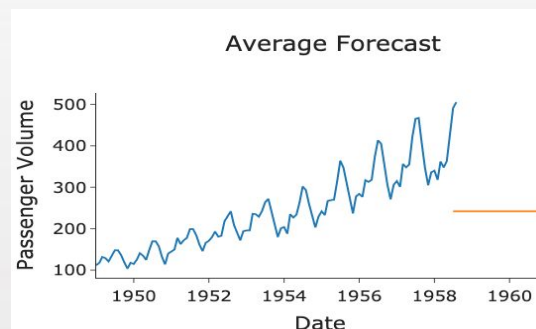
3.8.1 Average Forecast

Model pertama yang akan kami pertimbangkan adalah Average Forecast. Model ini mengasumsikan bahwa semua nilai masa depan sama dengan rata-rata dari semua pengamatan sebelumnya:

$$y_{T+h} = \bar{y} = \frac{\sum_{t=1}^T y_t}{T}$$

Di mana h adalah waktu yang akan datang yang akan dilakukan forecasting, T adalah panjang deret waktu, y_t adalah nilai yang diamati pada waktu t dan \bar{y} adalah rata-rata dari nilai yang diamati.

Untuk model ini kita harus memiliki beberapa data masa lalu yang tersedia untuk menghitung peramalan. Kami dapat mengimplementasikan ini dengan Python menggunakan dataset penumpang pesawat



Dari plot di atas, ramalannya jelas tidak terlalu bagus karena belum menangkap tren atau musiman dalam data dan jelas jauh di bawah perkiraan.

Model kedua, Naive forecasting, menetapkan perkiraan masa depan sama dengan nilai pengamatan terbaru:

$$y_{T+h} = y_T$$

Forecasting

Model ini dianggap sebagai tolok ukur untuk perkiraan apa pun dan sering digunakan untuk memodelkan pasar saham dan data keuangan karena sifatnya yang tidak menentu. Model naive juga bisa disebut model random-walk- without-drift. Itu juga merupakan dasar di balik metrik kesalahan yang berarti metric mean absolute scaled error (MASE).

3.8.2 Seasonal Naive Forecasting

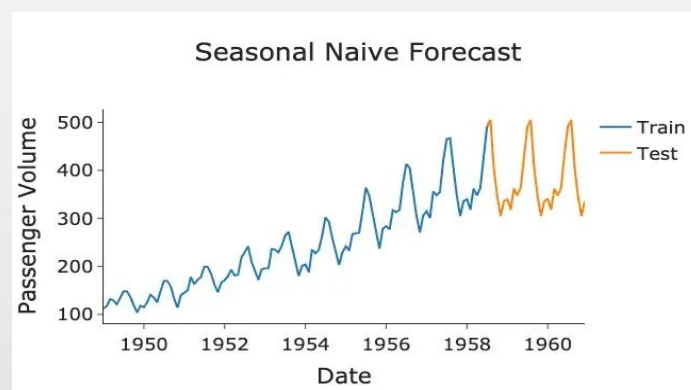
Metode ketiga adalah perluasan dari metode naif, tetapi kali ini ramalannya sama dengan nilai pengamatan terbaru di musim yang sama. Oleh karena itu, ini dikenal sebagai Seasonal Naive Forecasting. Misalnya, ramalan untuk kuartal satu berikutnya sama dengan nilai kuartal satu tahun sebelumnya. Model ini berguna ketika kita memiliki variasi musiman yang jelas dan besar dalam deret waktu kita. Secara matematis model ditulis sebagai:

$$y_{T+h} = y_{T+h-m}$$

Di mana m adalah musiman dari data. Jadi, untuk data bulanan dengan musiman tahunan $m=12$, data triwulanan akan memiliki $m=4$ dan data mingguan akan memiliki $m=52$.

Di bawah ini adalah implementasi Seasonal Naive Forecasting dengan Python untuk kumpulan data penumpang maskapai AS:

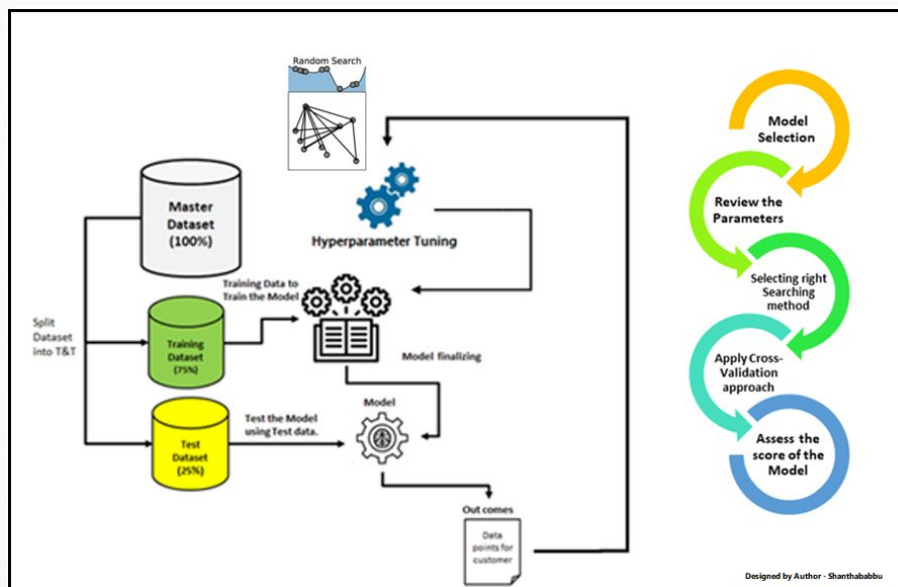
```
# Seasonal naive forecast
train['month_number'] = pd.DatetimeIndex(train['Month']).month
test['month_number'] = pd.DatetimeIndex(test['Month']).month
snaive_fc = []
for row_idx, row in test.iterrows():
    month = row['month_number']
    forecast = train['#Passengers'].loc[train['month_number'] == month].iloc[-1]
    snaive_fc.append(forecast)
plot_func(snaive_fc, 'Seasonal Naive Forecast')
```



Karena model memiliki komponen musiman yang cukup jelas dan besar, Seasonal Naive Forecasting berperforma cukup baik. Namun, itu belum sepenuhnya menangkap tren data karena kami memperkirakan volume penumpang akan meningkat dari waktu ke waktu.

3.9 Hyperparameter Tuning

Setelah membuat model dalam proses machine learning, kita akan melakukan hyperparameter tuning untuk menemukan model yang tepat dan meningkatkan performa dari model tersebut. Hyperparameter tuning adalah sebuah proses dalam menentukan kombinasi yang tepat bagi setiap hyperparameter yang ada di setiap model, yang dapat memaksimalkan hasil prediksi model tersebut.



<https://editor.analyticsvidhya.com/uploads/64801HPTT.png>

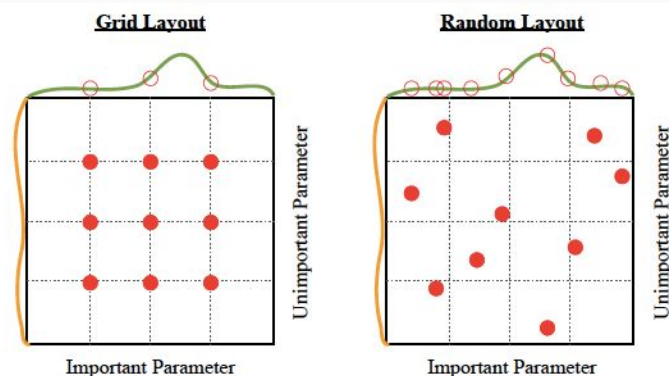
Setiap model memiliki hyperparameter yang berbeda dan proses penentuan hyperparameter optimal tentunya akan berbeda. Hyperparameter apa yang optimal juga akan berbeda di setiap kasusnya. Sebagai seorang data scientist, beberapa pertanyaan akan muncul saat kita melakukan hyperparameter tuning:

- Saat membuat model linear, degree apa yang kita gunakan?
- Dalam membuat model decision tree, berapa maximum depthnya? Berapa sampel minimal yang dibutuhkan dalam leaf-nodenya?
- Berapa banyak trees perlu disertakan dalam model random forest?
- Berapa banyak neurons yang diperlukan dalam model neural network?
- dan pertanyaan-pertanyaan lainnya.

Diatas merupakan contoh beberapa hyperparameter yang dimiliki beberapa model. Dokumentasi hyperparameter bagi setiap model bisa dilihat di web [scikit learn](https://scikit-learn.org/).

Hyperparameter tuning

Tentunya, kita tidak mungkin melakukan percobaan hyperparameter tuning secara manual dalam menemukan kombinasi hyperparameter terbaik. Oleh karena itu, kita dapat meminta python untuk menemukan hyperparameter tuning yang optimal. Secara umum, terdapat dua cara untuk melakukan automated hyperparameter tuning yakni grid search dan random search.



Grid Search dan Random Search akan mencoba kombinasi dari hyperparameter yang sudah kita tentukan. Namun perbedaannya, grid search akan mencari setiap kombinasi, sementara random search tidak. Keduanya akan mencoba kombinasi dari hyperparameter, dan akan melihat evaluasi dari model tersebut. Kemudian, keduanya akan memberikan hyperparameter apa yang terbaik untuk sebuah model.

Grid search akan menemukan kombinasi “terbaik” karena akan mencoba semua kombinasi yang memungkinkan. Namun kekurangannya adalah grid search membutuhkan sumberdaya yang tinggi, baik secara waktu maupun monetary. Di sisi lain, random search biasanya membutuhkan waktu yang lebih sedikit, namun kekurangannya adalah hasil yang diberikan mungkin saja bukan merupakan yang “terbaik”.

Contoh code Grid Search

```
from sklearn.model_selection import GridSearchCV
model = LogisticRegression()
grid_vals = {'penalty': ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1]}
grid_lr = GridSearchCV(estimator=model, param_grid=grid_vals, scoring='accuracy',
                       cv=6, refit=True, return_train_score=True)
#Training and Prediction
grid_lr.fit(X_train, y_train)
preds = grid_lr.best_estimator_.predict(X_test)
```

Contoh code Random Search

```
from sklearn.model_selection import RandomizedSearchCV
model = RandomForestClassifier()
param_vals = {'max_depth': [200, 500, 800, 1100], 'n_estimators': [100, 200, 300, 400],
              'learning_rate': [0.001, 0.01, 0.1, 1, 10]}
random_rf = RandomizedSearchCV(estimator=model, param_distributions=param_vals,
                               n_iter=10, scoring='accuracy', cv=5,
                               refit=True, n_jobs=-1)
#Training and prediction
random_rf.fit(X_train, y_train)
preds = random_rf.best_estimator_.predict(X_test)
```

3.9 Feature Importance

Feature Importance merupakan sebuah teknik yang dapat digunakan untuk melihat “seberapa penting” sebuah feature dalam memprediksi target variabel. Feature importance dapat digunakan untuk melihat ranking dari setiap variabel yang ada.

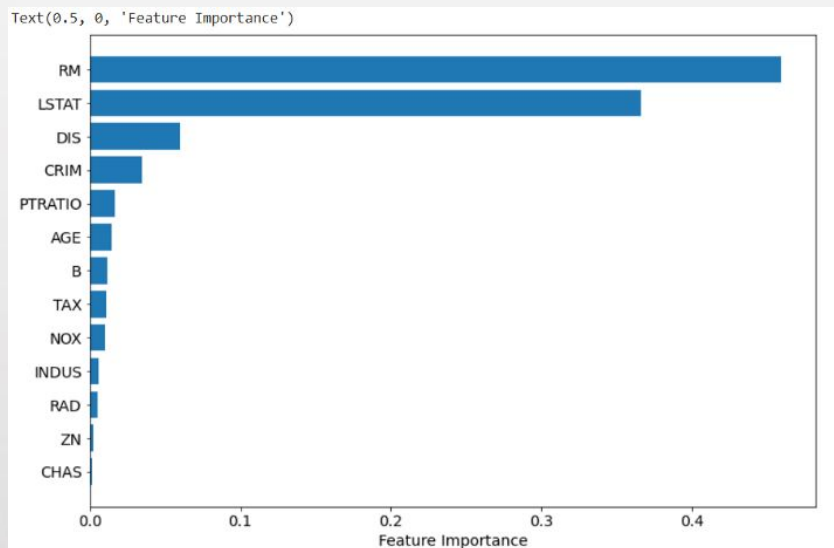
Misalnya saat kita ingin membeli rumah pastinya ada beberapa hal yang akan kita pertimbangkan. Apabila kita mementingkan kedekatan lokasi rumah dengan tempat kerja dibandingkan faktor lainnya seperti lingkungan, keamanan, dan lainnya maka “importance” dari jarak akan menjadi sangat penting dalam keputusan. Sama seperti analogi sebelumnya, feature yang memiliki importance paling tinggi akan memiliki pengaruh paling besar terhadap prediksi kita di sebuah model.

Feature importance sendiri memiliki beberapa keuntungan yakni:

- Kita dapat memahami data lebih baik dan juga memahami hubungan antara fitur dan variabel target
- Meningkatkan performa model dengan menghilangkan feature yang memiliki importance rendah untuk menurunkan dimensionalitas
- Memudahkan interpretasi model.

Contoh feature importance

```
sort = rf.feature_importances_.argsort()  
plt.barh(boston.feature_names[sort],  
rf.feature_importances_[sort])  
plt.xlabel("Feature Importance")
```



https://miro.medium.com/max/720/0*vKRFyD57QMHC1WAp

EXERCISE

Berikut adalah langkah - langkah untuk mengerjakan *exercise*:

1. Buka *link* Google Colab menggunakan Google Chrome
2. Klik 'Copy to Drive'
3. Kerjakan *exercise* sesuai instruksi yang tertera

Google Colab dapat diakses pada:

[Exercise Modul 3 DS](#)



MODULE

DATA SCIENCE TRACK