Laporan Project Based
Mata Kuliah Pembelajaran Mesin
Regresi (dataset: autos MPG) - Boosting

Mohammad Andiez Satria Permana - 1301218598
HNW Syahuda Nahatmasuni - 1301218603
Wandi Yusuf Kurniawan – 1301218601
IFX-45-GAB – IKN

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2023

1. Formulasi masalah

Pada tugas kali ini kelompok kami mendapatkan tugas tipe 2 yaitu regresi pada dataset autos MPG menggunakan metode *boosting*. Memprediksi tingkat kehematan bahan bakar kendaraan atau MPG (miles per gallon: rataan jarak tempuh mobil dalam mil untuk setiap galon bahan bakar yang dikonsumsi) berdasarkan profil mobil yang diberikan yang diwakili oleh atribut-atribut seperti silinder, daya (tenaga kuda), tahun keluaran, dll. Berikut merupakan atribut yang terdapat pada dataset autos MPG.

1. mpg: continuous (target attribute)
2. cylinders: multi-valued discrete
3. displacement: continuous
4. horsepower: continuous
5. weight: continuous
6. acceleration: continuous
7. model year: multi-valued discrete
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

2. Eksperimen

Eksperimen yang kami lakukan adalah membandingkan skor, error, dan kepentingan error antara model SVM dengan XGBoost.

3. Eksplorasi dan pra-pemrosesan data (termasuk data splitting)

Pertama-tama, kami melakukan import library yang dibutuhkan untuk mengerjakan tugas ini pada Google Collab menggunakan bahasa Python, diantaranya adalah:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
from sklearn.decomposition import PCA
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score, mean_squared_error, mean_absolute_error, r2_score
```

Kemudian inisialisasi variable `data` untuk membaca file .csv yang diambil dari https://drive.google.com/drive/folders/1HmKavcNCij76k02nCYQdP012cepM2nmz dan memanggil .head() untuk memastikan bahwa variabel tersebut sudah berisi dataset.

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Cek .info() pada data untuk mengetahui informasi kolom, berdasarkan table di bawah terdapat 3 kolom dengan data type float64, 4 kolom dengan data type int64 dan 2 kolom dengan data type object seperti pada table di bawah ini.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car_name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

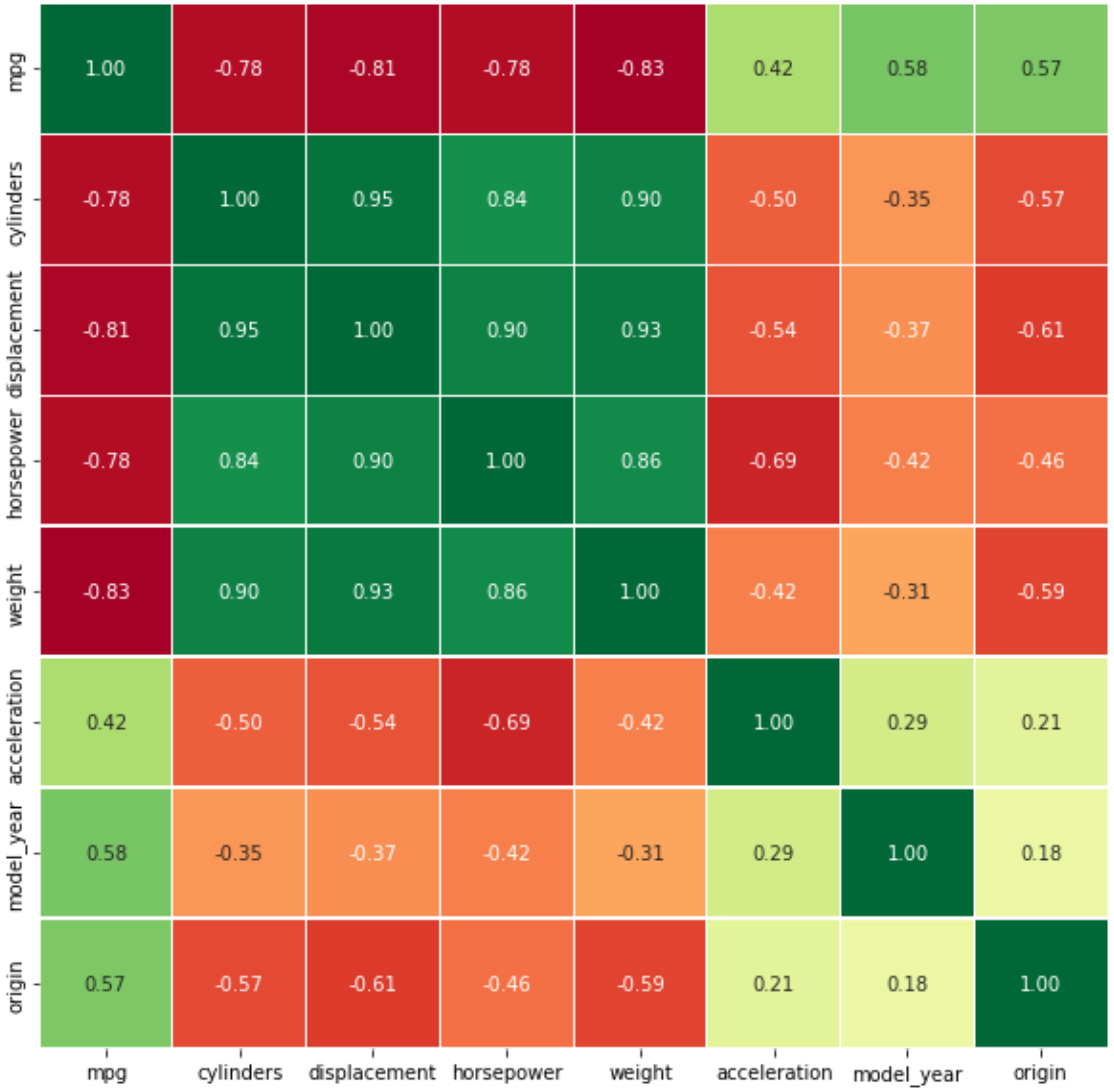
Data type object pada horsepower diubah ke numeric dan selanjutnya barisan yang tidak mempunyai nilai horsepower dihapus. Jumlah row setelah penghapusan nilai null pada horsepower berkurang menjadi 392.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64  
 7   origin        392 non-null    int64  
 8   car_name      392 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 30.6+ KB
```

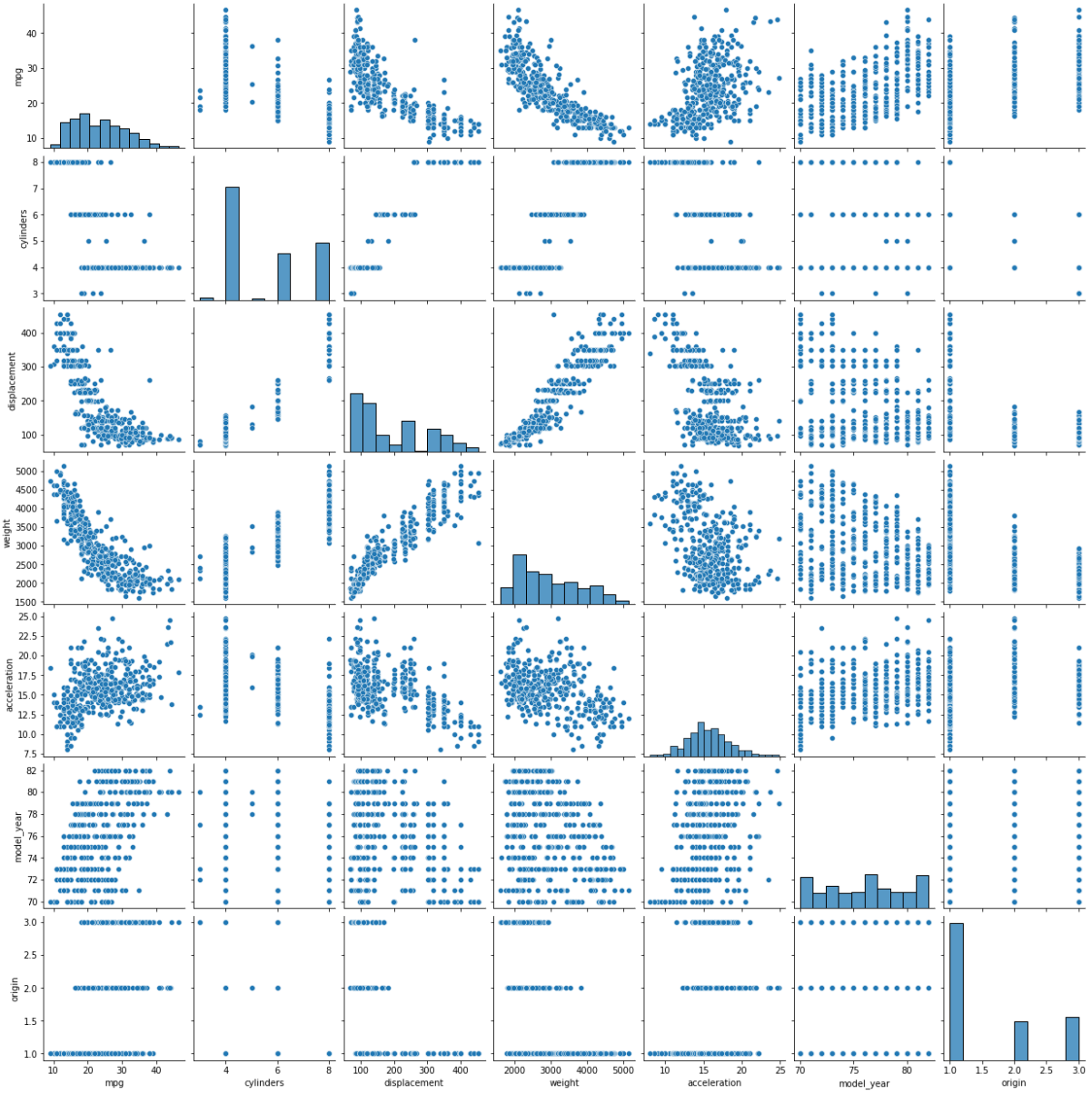
Selanjutnya lakukan .describe() untuk mengetahui count, mean, std, min dan max dari setiap kolom pada data autos MPG. Berikut table yang akan tampil.

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.979592	1.576531
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.683737	0.805518
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.000000	1.000000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

Selanjutnya data kita tampilkan plot dengan heatmap korelasi menggunakan metode pearson beserta pairplot yang berisi scatter plot untuk berbeda row dan histogram untuk satu row. Berikut plot yang akan tampil.



Selanjutnya dilakukan visualisasi data menggunakan pairplot. Pairplot berguna untuk melihat dominasi dari suatu katergori atau kelas terhadap setiap parameter numerik yang ada.



Sebelum membentuk model, kami membagi dataset menjadi data X (fitur) dan y (nilai prediksi, bukan klasifikasi label). Masing-masing dipisah menjadi data train dan test dengan perbandingan 80:20, kemudian nilai X distandarisasikan untuk menghindari jangkauan yang lebar.

```
X = data.drop(['car_name','mpg'], axis=1, inplace=False)
y = data['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
```

4. Pemodelan

Pemodelan yang dibangun ada dua, yaitu SVM dan XGBoost.

```
# SVM
svm = SVR(kernel = 'rbf')
svm.fit(X_train_std, y_train)

# Memilih param terbaik untuk XGBoost
params = {'objective':['reg:squarederror'],
              'booster':['gbtree','gblinear'],
              'learning_rate': [0.0001, 0.001, 0.01, 0.1, 1.0],
              'max_depth': [3, 5, 7, 9, 12, 16, 20],
              'n_estimators': [100, 150, 200, 250, 300, 400, 500, 600, 750, 1000],
              'subsample': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],}

xgb_model = XGBRegressor(random_state=42)
grid_obj_xgb = RandomizedSearchCV(xgb_model, params, cv=5, n_iter=10,
scoring='neg_mean_absolute_error', verbose=3, n_jobs=12)
grid_obj_xgb.fit(X_train_std, y_train, verbose=1)

# XGBoost
xgb = XGBRegressor(subsample = 0.1,
                   objective = 'reg:squarederror',
                   n_estimators = 1000,
                   max_depth = 12,
                   learning_rate = 0.01,
                   booster = 'gbtree')
xgb.fit(X_train_std, y_train)
```

Untuk SVM hanya memasukan satu parameter, yaitu kernel=’rbf’(radial basis function). Sedangkan XGBoost memiliki enam parameter dengan masing-masing serangkaian nilai dengan RandomizedSearchCV dengan 5 kali cross-validation dan 10 kali iterasi untuk menentukan parameter terbaik yang nantinya digunakan untuk menghasilkan error seminimal mungkin. Gambar dibawah adalah contoh hasil parameter terbaik (karena bisa berbeda-beda antara hasil pengacakan pada pemisahan data X dan y):

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
RandomizedSearchCV(cv=5, estimator=XGBRegressor(random_state=42), n_jobs=12,
                   param_distributions={'booster': ['gbtree', 'gblinear'],
                                        'learning_rate': [0.0001, 0.001, 0.01,
                                                          0.1, 1.0],
                                        'max_depth': [3, 5, 7, 9, 12, 16, 20],
                                        'n_estimators': [100, 150, 200, 250,
                                                         300, 400, 500, 600,
                                                         750, 1000],
                                        'objective': ['reg:squarederror'],
                                        'subsample': [0.1, 0.2, 0.3, 0.4, 0.5,
                                                      0.6, 0.7, 0.8, 0.9, 1]},
                   scoring='neg_mean_absolute_error', verbose=3)
```

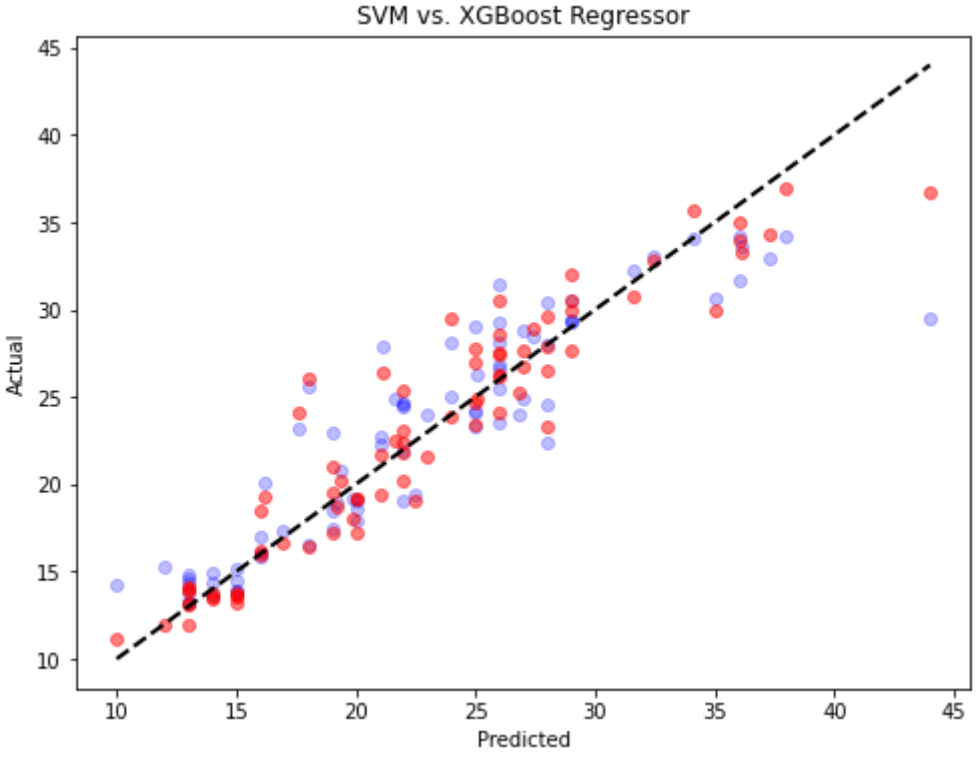
```
Best Parameters:
 {'subsample': 0.4, 'objective': 'reg:squarederror', 'n_estimators': 250, 'max_depth': 20, 'learning_rate': 0.1, 'booster': 'gbtree'}

Best Score: -2.1683143977366894
```

5. Evaluasi:

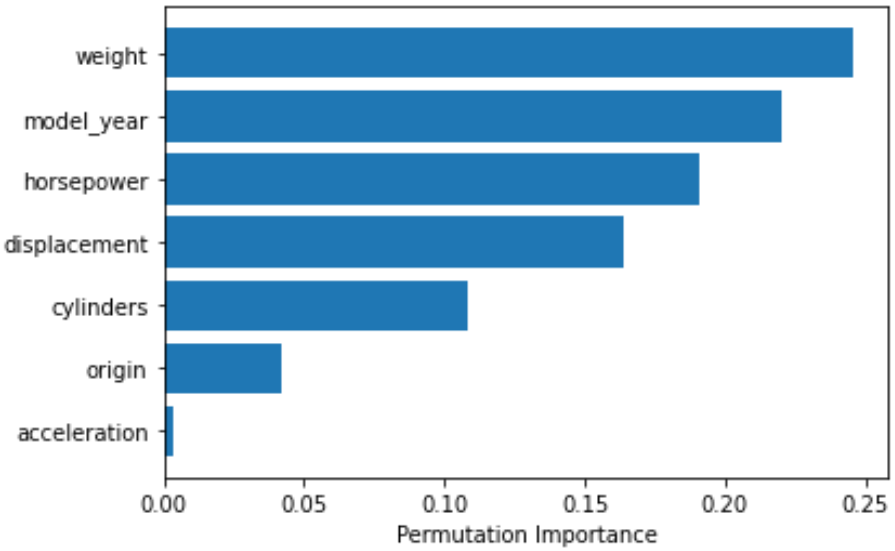
Setelah melakukan eksekusi, kami menemukan hasil berupa perbandingan skor (train dan test) dan error (RMSE, MSE, MAE dan R2) pada kedua model:

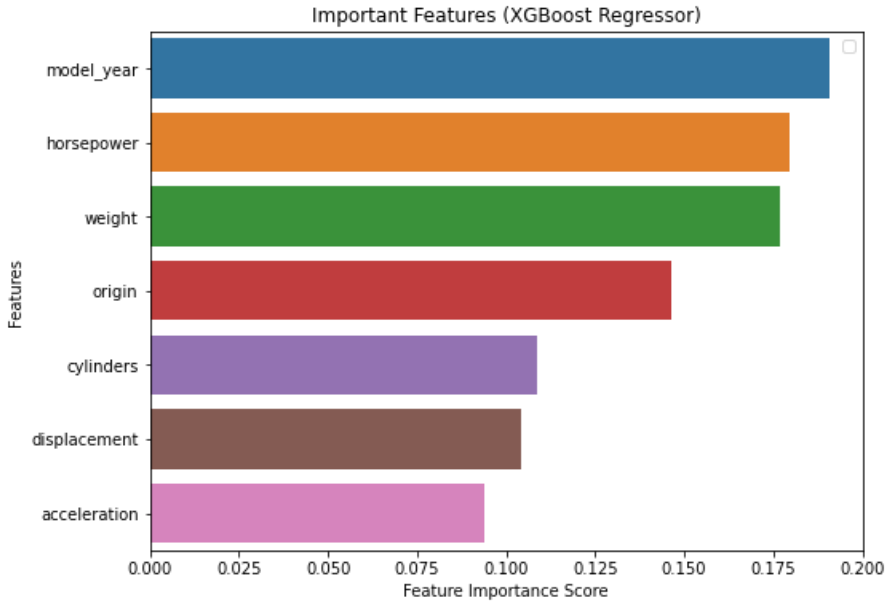
Model	SVM	XGBoost
Train score	0.86	**0.94**
Test score	0.82	**0.89**
RMSE	3.05	**2.40**
MSE	9.27	**5.75**
MAE	2.12	**1.72**
R2	0.82	**0.89**

Selanjutnya untuk perbandingan scatter plot, nilai prediksi yang dihasilkan dari model SVM (biru) dan XGBoost (merah) dengan nilai aktual dari dataset yang ada. Terlihat jelas bahwa titik-titik XGBoost lebih mendekati garis putus-putus (kesesuaian antara nilai prediksi dan aktual) daripada titik-titik SVM, sangat berkaitan dengan score yang lebih tinggi dan error yang lebih rendah.



Selanjutnya adalah menampilkan fitur yang penting setelah dilakukan training dan testing pada setiap model. Dapat dilihat pada plot bar, fitur yang paling penting adalah "weight" (untuk model SVM) dan "model_year" (untuk model XGBoost). Semakin besar nilai kepentingannya, semakin besar kontribusi yang diberikan pada nilai error, terlihat pada kesenjangan fitur di model SVM dengan yang lebih merata di model XGBoost.





6. Kesimpulan

Secara keseluruhan, model XGBoost lebih baik daripada SVM karena menghasilkan skor yang lebih tinggi dan error yang lebih rendah, terlihat dari perbandingan hasil scatter plot dengan titik-titik yang mendekati garis putus-putus (kesesuaian antara nilai prediksi dan aktual) untuk XGBoost. Kepentingan fitur juga berkontribusi dalam nilai error, semakin besar kesenjangan antar fitur, semakin beresiko untuk mendapatkan nilai error yang besar.

Link Google Colab:

https://colab.research.google.com/drive/1s7ZXsrgU1PmOBfv1u0xIZObXEgUFpIZn#scrollTo=yfBp7_0xNC0E

Link YouTube/OneDrive: Presentasi_Project_Based_98.mp4