LAPORAN TUGAS MACHINE LEARNING :

PROJECT BASE

Kode Dosen : DDR

Disusun Oleh :

Anyelir Belia Azzahra - 1301200048
Fakhri Maulana Falah - 1301202117
Gilang Satya Nugraha - 1301202319
Mirai Tsuchiya - 1301203555

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY BANDUNG
2022

DAFTAR ISI

Bagian 1
Ikhtisar Kumpulan Data

A. Formulasi Masalah

Tujuan dari pembuatan laporan ini adalah untuk memprediksi tingkat kehematan bahan bakar kendaraan/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. Diberikan sebuah dataset yaitu autos_mpg.csv dari drive : drive autos_mpg.csv dataset autos_mpg.csv berisi 398 data entries dan 9 kolom atribut

```
df=pd.read_csv("autos_mpg.csv")
df
```

	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
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

Bagian 2
Eksplorasi dan Pra-Pemrosesan Data

A. Pada Pra-Pemrosesan data dilakukan beberapa tahap yaitu :

1. Data Cleaning, Mengecek Tipe Data dari masing-masing row pada Dataset, Menghapus Data yang kosong (Nan), menghapus fitur atau kolom yang tidak diperlukan. Teknik ini dilakukan karena perlu adanya handling missing value untuk tahap selanjutnya.
2. Correlation plot dan Heatmap, untuk mengecek korelasi antara setiap fitur. Teknik ini dilakukan untuk mencari korelasi terbesar yang akan dipakai di proses berikutnya.
3. Scatter Plot, Menampilkan trend atau penyebaran data, juga outliers pada data tersebut.
4. Outliers, Menghitung Outliers dan Penghilangan Outliers. Mencari atau menghitung outliers dilakukan untuk mencari data yang memiliki pencilan atau data tersebut menyimpang terlalu jauh dari data yang lainnya dalam suatu pencilan dan menghasilkan data yang samar, sehingga perlu dihilangkan. Pencarian itu bisa ditampilkan menggunakan Scatter Plot.
5. Normalisasi data / Scaling, menggunakan metode Standard Scaling untuk menghapus rata-rata dan menskalakan setiap fitur/variabel ke varian satuan. Ini dilakukan berdasarkan fitur dengan cara yang independen. StandardScaler dapat dipengaruhi oleh outlier (jika ada dalam kumpulan data) karena melibatkan estimasi rata-rata empiris dan standar deviasi dari setiap fitur. Normalisasi ini digunakan agar data yang memiliki rentang yang lebih luas tidak dapat memiliki pengaruh yang lebih besar terhadap hasil, dan dapat menyebabkan bias dalam hasil.
6. Splitting data, Kami membagi dataset menjadi set pelatihan dan pengujian untuk mengevaluasi seberapa baik kinerja model kami. Set train digunakan agar sesuai dengan model, dan agar statistik set train dapat diketahui. Himpunan kedua disebut himpunan data uji, himpunan ini digunakan untuk prediksi.

B. Implementasi Code dari Pra-Pemrosesan Data

1. Data Cleaning :

   Melakukan handling missing value dengan mengubah data yang berisi '?' menjadi null atau mengubah data kategorikal menjadi null.

```
# Mengubah data yang berisi '?' menjadi null
df = df.replace('?',np.NaN)
```

Disini kita mengecek duplikat kolom, dan data yang kosong pada autos_mpg

```
# Menampilkan kolom pada dataset autos-mpg
print("Kolom :", df.columns, "\n")

# Menampilkan data yang duplikat pada autos-mpg
print("Duplikat :", df.duplicated().sum(), "\n")

# Menampilkan data yang kosong pada autos-mpg
print(df.isnull().sum())
```

```
Kolom : Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'car_name'],
      dtype='object')

Duplikat : 0

mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

Didapat 6 data kosong pada kolom horsepower dan tidak ada kolom duplikat. Data null pada data tersebut harus dihilangkan.

Langkah pertama untuk handlingnya adalah dengan melakukan pengisian data yang memiliki value "?" dengan metode imputer :

```
[ ]  null = []
     for i in df:
       value = []
       if df[i].isna().sum() > 0:
         null.append(i)
```

Melakukan handling missing value “null” dengan mengisi data yang kosong dengan median, sehingga datanya tidak ada yang null.

```
[ ]  # Mengisi data kosong dengan median
     imputer = SimpleImputer(strategy = 'median', missing_values = np.nan)
     imputer.fit(df[null])
     df[null] = imputer.transform(df[null])
```

Hasil data setelah dilakukan pengisian value adalah sebagai berikut. Terlihat tidak ada data yang null dalam data ini.

Hasil Pengisian

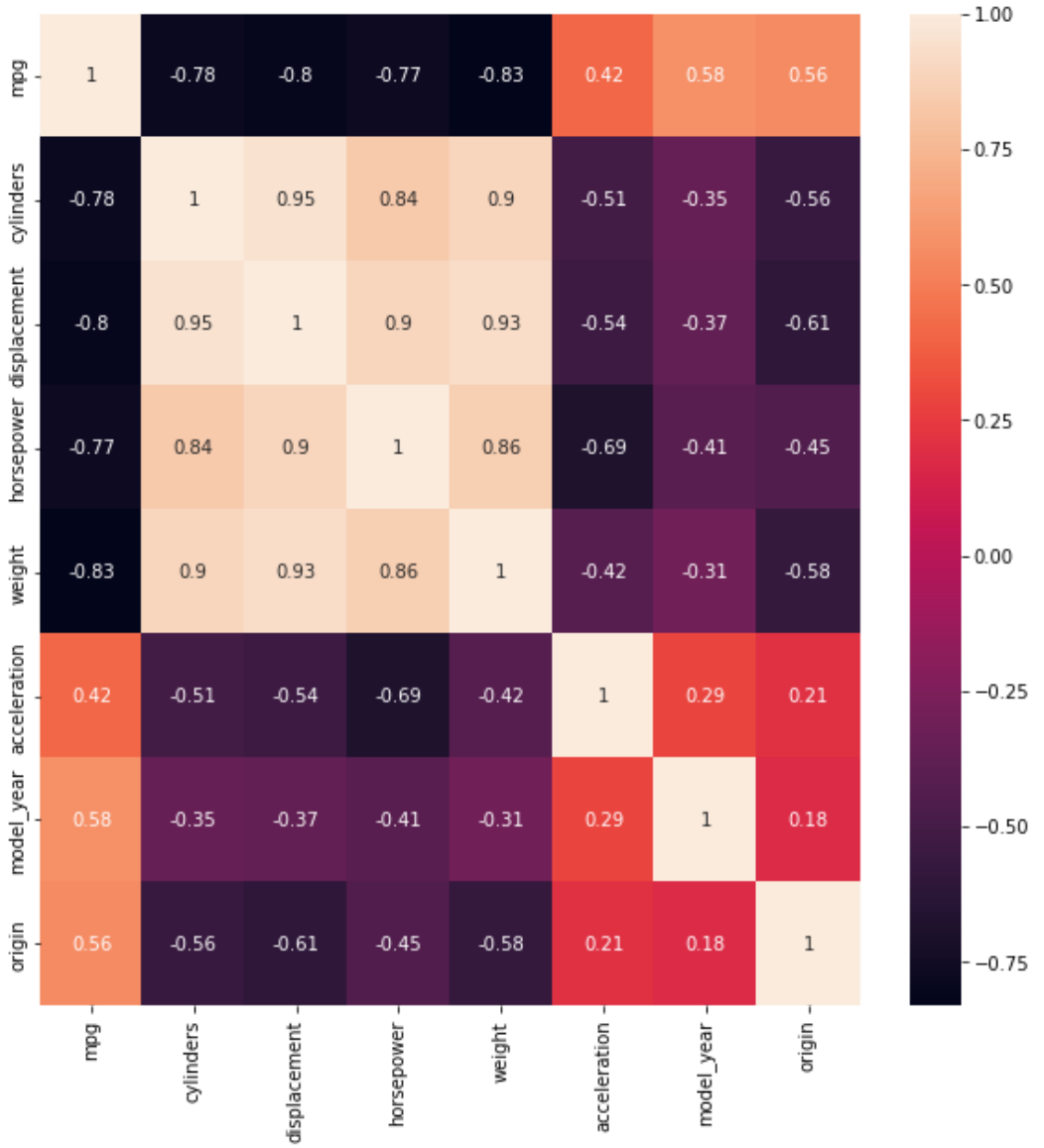
```
[ ]  # Menampilkan data yang kosong setelah diisi
     print(df.isnull().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

2. Correlation :
   Selanjutnya akan dilakukan penampilan korelasi atau heatmap untuk mencari korelasi dengan target yang dibandingkan adalah “mpg”:

```
# print korelasi / heatmap
print(df.corr())
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(df.corr(), annot=True)
plt.show()

corr_mat=df.corr()
print(corr_mat["mpg"].sort_values(ascending=False))
```

Tampilan dari heatmap atau korelasi :




```
mpg             1.000000
model_year      0.579267
origin          0.563450
acceleration    0.420289
horsepower     -0.773453
cylinders      -0.775396
displacement   -0.804203
weight         -0.831741
Name: mpg, dtype: float64
```

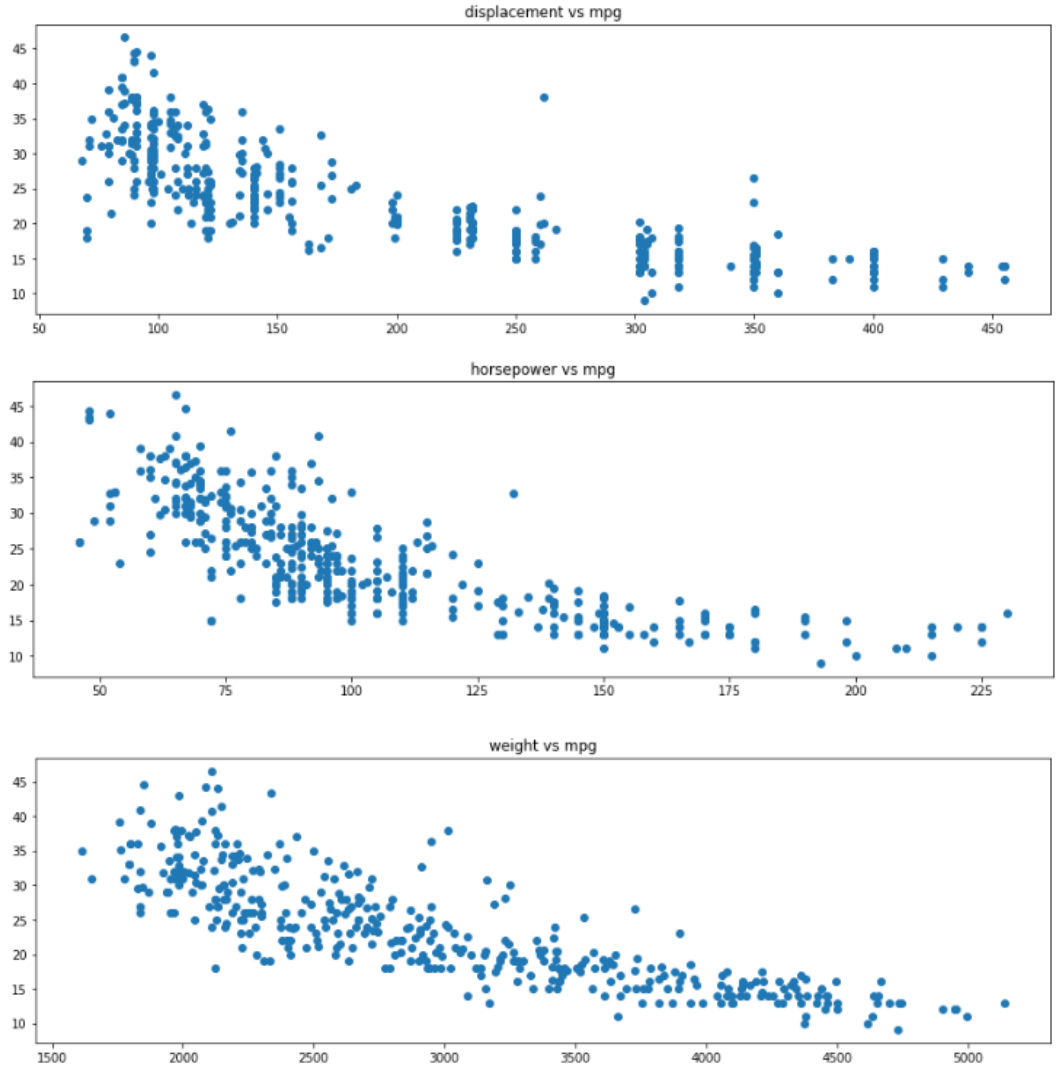
Dapat terlihat dalam hasil heatmap tersebut korelasi terbesar dengan mpg adalah horsepower, cylinders, displacement, dan weight. Sehingga keempat data tersebut yang akan kami pakai pada proses selanjutnya.

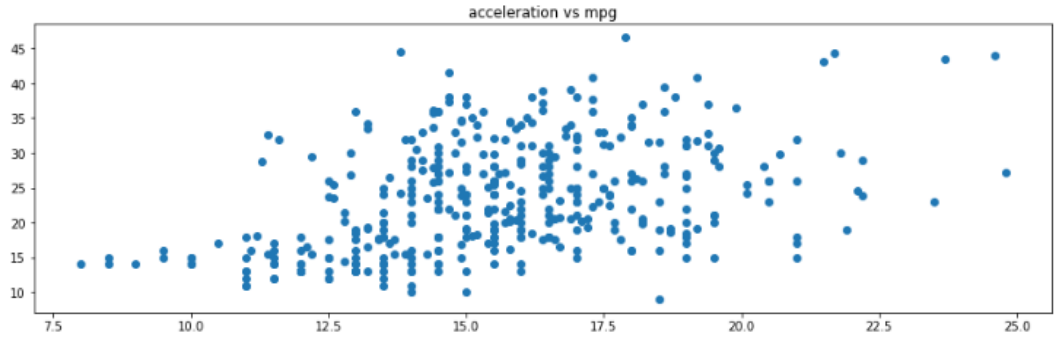
3. Scatter Plot :

   Pada bagian ini,kami melakukan penampilan scatter plot dari keempat kolom yang dipilih kemudian dibandingkan dengan MPG nya.

```
# menampilkan scatter plot dari keempat kolom yang dipilih dibandingkan
# dengan mpgnya
fig, ax = plt.subplots(4, 1,figsize=(15,20))
cont = ['displacement', 'horsepower', 'weight', 'acceleration']
for i,x in enumerate(cont):
    ax[i].scatter(df[x],df['mpg'],label=("{} vs mpg".format(x)))
    ax[i].set_title("{} vs mpg".format(x))
plt.show()
```

   Tampilan atau hasil dari scatter plot dengan y merupakan “mpg” dan y merupakan kolom yang dipilih:



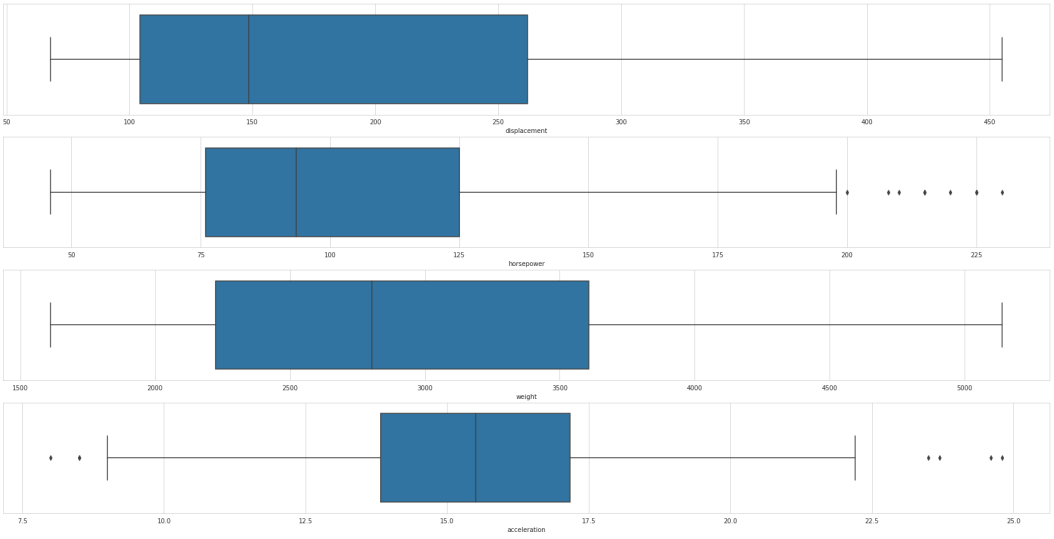


4. Outliers:

   Pada bagian ini kami menampilkan outliers pada keempat kolom yang dipilih menggunakan boxplot

```
# print boxplot dari data diatas
sns.set_style("whitegrid")
fig, axes = plt.subplots(4, 1, figsize=(30, 15))
for i,x in enumerate(cont):
    sns.boxplot(ax=axes[i], x=x, data=df)
plt.show()
```

   Tampilan atau hasil dari boxplot yang masih memiliki outliers



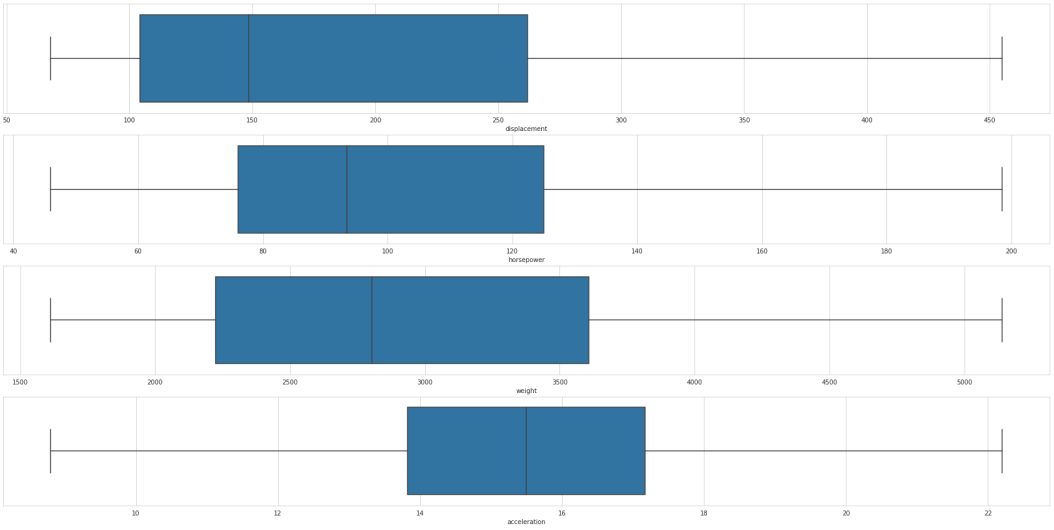
Membuat fungsi untuk menghilangkan outliers, dan menjalankannya

```
# Fungsi menghapus outliers
def remove_outlier(col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1-(1.5 * IQR)
    upper_range= Q3+(1.5 * IQR)
    return lower_range, upper_range
```

```
# Menjalankan penghapusan outliers
for column in df[cont].columns:
    print(column)
    lr,ur=remove_outlier(df[column])
    df[column]=np.where(df[column]>ur,ur,df[column])
    df[column]=np.where(df[column]<lr,lr,df[column])
```

```
displacement
horsepower
weight
acceleration
```

Tampilan atau hasil dari boxplot setelah menjalankan fungsi penghapusan outliers



5. Normalisasi data / Scaling :
   Normalisasi data kami menggunakan StandardScaler seperti yang telah dijelaskan sebelumnya. Berikut adalah program standard scaler kami :

```
scaler = StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.fit_transform(X_test)
```

6. Splitting data :
   Pada proses ini kami melakukan drop pada data data yang tidak dibutuhkan. Setelah dilakukan splitting data, didapat 318 data train dan 80 data tes.

```
X = df.drop(['mpg','car_name',"origin","acceleration","horsepower"],axis='columns')
Y =df['mpg']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)
print("jumlah data train :",len(X_train))
print("jumlah data tes :",len(X_test))
```

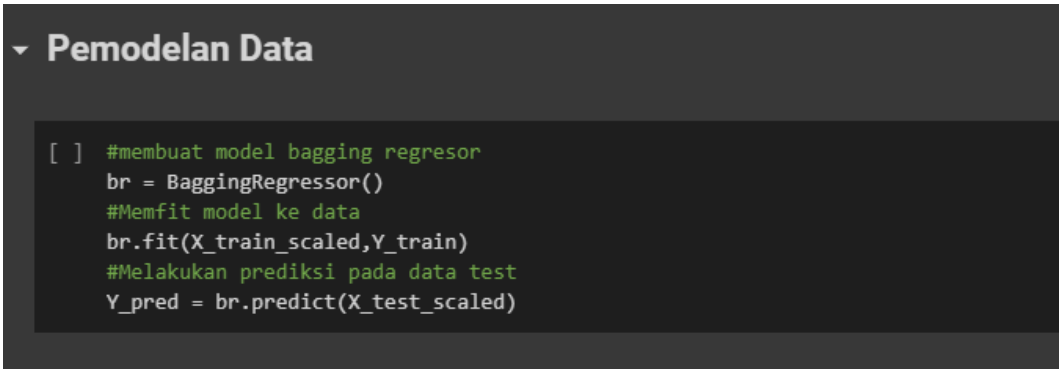
```
jumlah data train : 318
jumlah data tes : 80
```

Bagian 3
Permodelan

Disini kami menggunakan Bagging Regressor yang disediakan oleh library sklearn, Bagging Regressor adalah salah satu teknik pemodelan yang menggunakan bootstrapping dan averaging untuk meningkatkan akurasi dari sebuah model regresi. Bagging regressor akan membuat beberapa model regresi yang terdiri dari sebagian train data yang dipilih secara acak dengan menggunakan teknik bootstrapping, lalu hasil dari masing-masing model regresi tersebut akan diaverage atau dijumlahkan lalu dibagi dengan jumlah model regresi yang dibuat untuk mendapatkan hasil akhir dari bagging regressor.

A. Tahapan Pemodelan

1. Untuk melakukan pemodelan dengan bagging regressor, disini kami memasukkan data yang akan digunakan sebagai data train ke dalam sebuah objek bagging regressor.
2. Kemudian kami memanggil fungsi fit() pada objek tersebut dengan memasukkan data train yang telah di-scaling menggunakan StandardScaler sebagai argumennya.
3. Setelah itu, kami memprediksi nilai target pada data test dengan memanggil fungsi predict() pada objek bagging regressor yang telah dilatih tersebut dengan memasukkan data test yang telah di-scaling sebagai argumennya.



Hasil yang diperoleh dari pemodelan data tersebut adalah model yang dapat digunakan untuk prediksi nilai test, sehingga bisa memanggil fungsi predict. Setelah memanggil fungsi tersebut dapat dievaluasi dengan menggunakan beberapa metrik evaluasi seperti Mean Square Error (MSE), Root Mean Square Error (RMSE), R2 Score (coefficient of determination).

Bagian 4
Evaluasi

A. Evaluasi yang dipilih

Evaluasi yang kita pilih adalah Mean Square Error (MSE), Root Mean Square Error (RMSE), R2 Score (coefficient of determination).

1. Mean Square Error (MSE) adalah ukuran kesalahan yang mengukur seberapa jauh hasil prediksi dari model terhadap nilai yang sebenarnya. MSE dihitung dengan membagi jumlah kuadrat dari selisih antara nilai prediksi dan nilai yang sebenarnya dengan jumlah data yang ada. Semakin kecil MSE, maka model tersebut dianggap semakin baik.
2. Root Mean Square Error (RMSE) adalah MSE yang diakarkan. RMSE dihitung dengan mengakar MSE. RMSE mengukur seberapa jauh hasil prediksi dari model terhadap nilai yang sebenarnya dalam satuan yang sama dengan data yang digunakan untuk membuat model. Semakin kecil RMSE, maka model tersebut dianggap semakin baik.
3. R2 Score (coefficient of determination) adalah ukuran kebaikan model yang mengukur seberapa baik model tersebut dapat menjelaskan variasi dari data yang ada. R2 Score dihitung dengan membagi variasi total dari data dengan variasi yang tidak terjelaskan oleh model. R2 Score bernilai antara 0 dan 1, dengan nilai 1 menunjukkan bahwa model tersebut sangat baik dalam menjelaskan variasi dari data yang ada.

B. Tahapan Evaluasi

1. Menampilkan R2 Score train, score test, dan score test yang menggunakan metrik sklearn, serta MSE dan RMSE

```
[ ]  print("skor train R2 :",br.score(X_train_scaled,Y_train))
     print("skor Test R2 :",br.score(X_test_scaled,Y_test))
     print("skor Test R2 (sklearn metrics) :",r2_score(Y_test,Y_pred))
     print("MSE :",mean_squared_error(Y_test,Y_pred))
     print("RMSE :",mean_squared_error(Y_test,Y_pred,squared=False))
```

```
skor train R2 : 0.9680305254745019
skor Test R2 : 0.8609847795401074
skor Test R2 (sklearn metrics) : 0.8609847795401074
MSE : 9.924115000000002
RMSE : 3.150256338776259
```

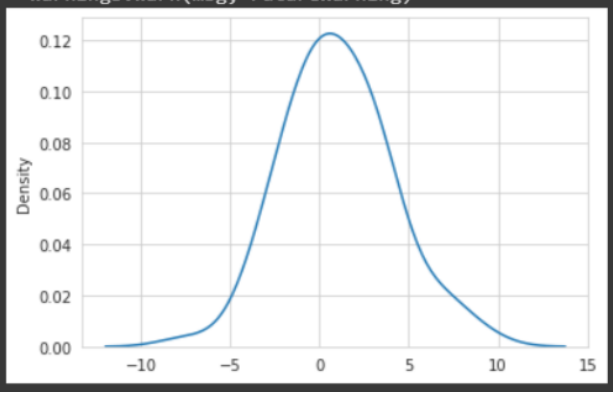
R2 score yang dihasilkan mendekati 1 atau pada range 0.86 - 0.97 , sehingga menunjukkan bahwa model tersebut baik. Hasil dari MSE dan RMSE juga kecil

dibandingkan menggunakan data train yang lain. Sehingga model tersebut dianggap sangat baik.

2. Menampilkan Selisih data Y_test dan Y_pred
   Pada proses ini kami mengidentifikasi selisih menggunakan variable delta dan menampilkan selisih tersebut menggunakan plot.

```
[ ]  # Menampilkan selisih data Y_test dan Y_pred
     delta = Y_test - Y_pred
     sns.set_style('whitegrid')
     sns.kdeplot(np.array(delta), bw=0.5)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:1699: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



Hasil dari plot tersebut terlihat nilai density dari 0 hingga 0.12 serta selisih dari Y_test dan Y_pred dari -10 hingga 15 dan yang paling banyak berada di range -5 hingga 5. Sehingga selisih tersebut tergolong kecil dan model data tersebut sangat baik.

Bagian 5
Eksperimen

Eksperimen ini dilakukan dengan melibatkan tahapan eksplorasi dan pra pemrosesan data, pemodelan, dan evaluasi untuk mendapatkan hasil terbaik.

A. Tahapan Eksperimen 1

1. Mengubah Test Size

   Test size diubah menjadi 99% dari yang awalnya 20% untuk melihat data train dan data tesnya

Pengubahan test size dari 20% ke 99%

```
X = df.drop(['mpg','car_name',"origin","acceleration","horsepower"],axis='columns')
Y = df['mpg']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.99)
print("jumlah data train :",len(X_train))
print("jumlah data tes :",len(X_test))
```

```
jumlah data train : 3
jumlah data tes : 395
```

   Dapat dilihat pada hasil data train adalah 3 yang berarti lebih kecil dari data train pada test size 20%, dan data test adalah 395 yang berarti lebih besar dari data test pada test size 20%.

   Dengan menggunakan data train sebanyak 3 data, disini model akan memprediksi data autos-mpg dengan hasil yang underfitting, dimana model tidak cocok dengan data tes dan tidak dapat menangkap pola atau trend dari data tes dengan baik.

2. Melibatkan tahapan eksplorasi dan pra pemrosesan data, pemodelan, dan evaluasi

Scaling (Eksperimen)

```
[ ]  scaler = StandardScaler()
     X_train_scaled=scaler.fit_transform(X_train)
     X_test_scaled=scaler.fit_transform(X_test)
```

Pemodelan Data (Eksperimen)

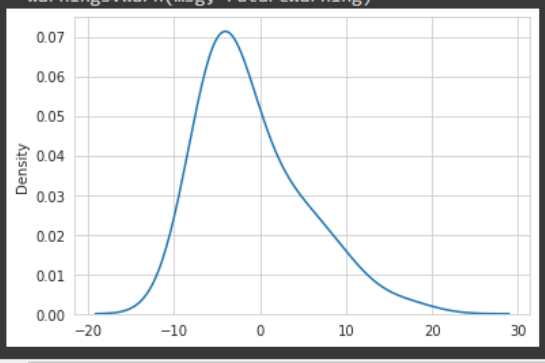
```
[ ]  br = BaggingRegressor()
     br.fit(X_train_scaled,Y_train)
     Y_pred = br.predict(X_test_scaled)
```

Evaluasi (Eksperimen)

```
[ ]  print("skor train R2 :",br.score(X_train_scaled,Y_train))
     print("skor Test R2 :",br.score(X_test_scaled,Y_test))
     print("skor Test R2 (sklearn metrics) :",r2_score(Y_test,Y_pred))
     print("MSE :",mean_squared_error(Y_test,Y_pred))
     print("RMSE :",mean_squared_error(Y_test,Y_pred,squared=False))

     skor train R2 : 0.8907330415754924
     skor Test R2 : 0.4118150674190092
     skor Test R2 (sklearn metrics) : 0.4118150674190092
     MSE : 35.99748734177215
     RMSE : 5.999790608160601
```

MSE yang didapatkan sebesar 35.99 dimana lebih besar dari hasil split 20%

```
[ ]  delta = Y_test - Y_pred
     sns.set_style('whitegrid')
     sns.kdeplot(np.array(delta), bw=0.5)
     plt.show()

     /usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:1699: FutureWarning: The `bw`
       warnings.warn(msg, FutureWarning)
```

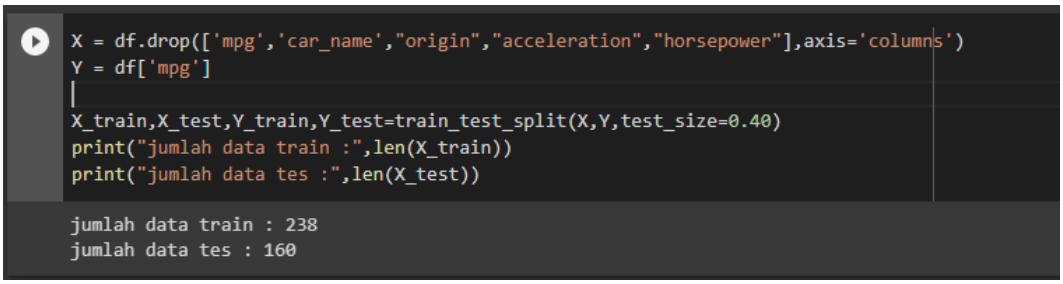


Pada hasil tersebut terlihat density dari 0 hingga 0.07 serta selisih / delta nya pada range -20 hingga 30 dan yang paling banyak berada pada -10 hingga 10. Hal ini menunjukkan bahwa selisih Y_test dan Y_pred pada test size 99% lebih besar dari test size 20%, yang artinya eksperimen tersebut tidak lebih baik dari evaluasi sebelumnya.

B. Tahapan Eksperimen 2

1. Mengubah Test Size

   Test size diubah menjadi 40% dari yang awalnya 20% untuk melihat data train dan data tesnya



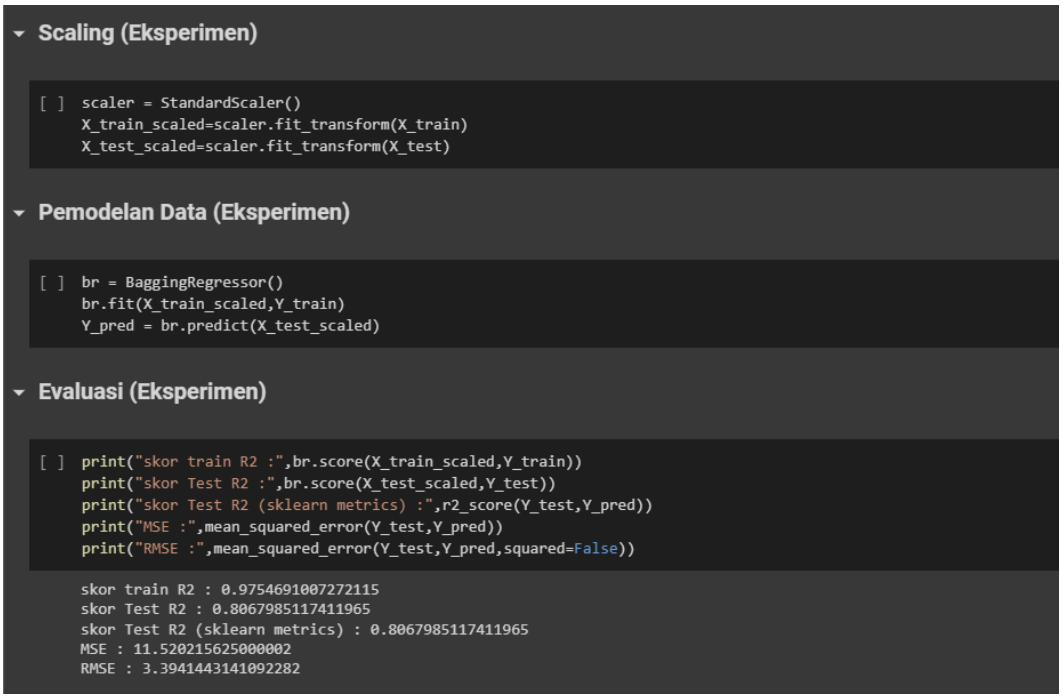
```
X = df.drop(['mpg','car_name',"origin","acceleration","horsepower"],axis='columns')
Y = df['mpg']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.40)
print("jumlah data train :",len(X_train))
print("jumlah data tes :",len(X_test))
```

```
jumlah data train : 238
jumlah data tes : 160
```

   Dapat dilihat pada hasil data train 238 yang berarti lebih besar dari data train pada test size 20%, dan data test adalah 160 yang berarti lebih kecil dari data test pada test size 20%.

   Dengan menggunakan data train sebanyak 238 data, disini model akan memprediksi data autos-mpg dengan hasil yang bisa jadi overfitting, Overfitting terjadi ketika model yang digunakan terlalu cocok dengan data latih namun kurang baik dalam memprediksi data baru, tetapi karena kami menggunakan bagging regressor, ini dapat mengurangi risiko overfitting dengan menggunakan beberapa model yang terdiri dari sebagian data latih yang dipilih secara acak, sehingga model yang dihasilkan tidak terlalu kompleks.

2. Melibatkan tahapan eksplorasi dan pra pemrosesan data, pemodelan, dan evaluasi



Scaling (Eksperimen)

```
[ ] scaler = StandardScaler()
    X_train_scaled=scaler.fit_transform(X_train)
    X_test_scaled=scaler.fit_transform(X_test)
```

Pemodelan Data (Eksperimen)

```
[ ] br = BaggingRegressor()
    br.fit(X_train_scaled,Y_train)
    Y_pred = br.predict(X_test_scaled)
```

Evaluasi (Eksperimen)

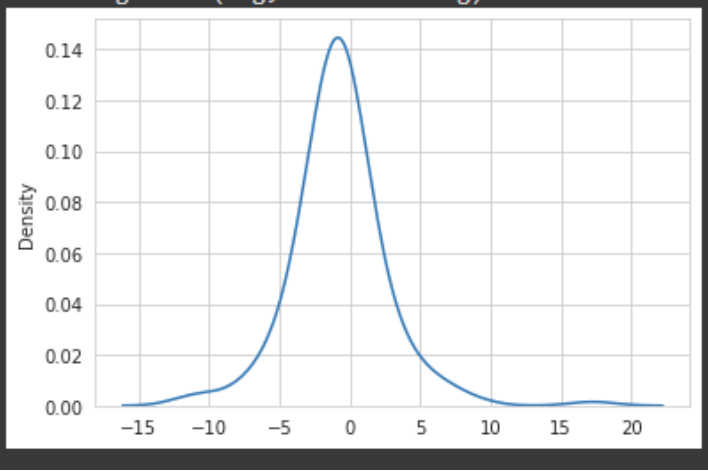
```
[ ] print("skor train R2 :",br.score(X_train_scaled,Y_train))
    print("skor Test R2 :",br.score(X_test_scaled,Y_test))
    print("skor Test R2 (sklearn metrics) :",r2_score(Y_test,Y_pred))
    print("MSE :",mean_squared_error(Y_test,Y_pred))
    print("RMSE :",mean_squared_error(Y_test,Y_pred,squared=False))
```

```
skor train R2 : 0.9754691007272115
skor Test R2 : 0.8067985117411965
skor Test R2 (sklearn metrics) : 0.8067985117411965
MSE : 11.520215625000002
RMSE : 3.3941443141092282
```

MSE yang didapatkan sebesar 11.53 dimana lebih besar dari hasil split 20%

```
[ ]  delta = Y_test - Y_pred
     sns.set_style('whitegrid')
     sns.kdeplot(np.array(delta), bw=0.5)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:1699: Fu
  warnings.warn(msg, FutureWarning)
```



Pada hasil tersebut terlihat density dari 0 hingga 0.14 serta selisih / delta nya pada range -15 hingga 10 dan yang paling banyak berada pada -5 hingga 5. Hal ini menunjukkan bahwa selisih Y_test dan Y_pred pada test size 40% lebih besar dari test size 20%, yang artinya eksperimen tersebut tidak lebih baik dari evaluasi sebelumnya.

Bagian 6
Kesimpulan

Disini kami diberikan dataset autos-mpg yang berisikan data-data kendaraan beserta tingkat kehematan bensinnya yang dilambangkan dengan mpg, atau *miles-per-gallon* disini kami ditugaskan untuk membuat model dengan metode regresi bagging terhadap kolom mpg tersebut yang kemudian akan dievaluasi dan dilakukan eksperimen.

Bagging Regressor adalah sebuah model pemodelan yang terdiri dari beberapa model regresi yang dibuat dengan menggunakan teknik bootstrapping pada data latih, lalu hasil dari masing-masing model tersebut diaverage atau dijumlahkan lalu dibagi dengan jumlah model yang dibuat untuk mendapatkan hasil akhir dari Bagging Regressor.

Sebelum kami memasuki tahap pemodelan, kami melakukan eksplorasi dan *pre-processing* data, dengan mencari kolom-kolom dengan korelasi terbesar dengan kolom mpg, dimana kami mendapatkan empat kolom, yaitu *displacement, weight, horsepower* dan *acceleration*.

Keempat kolom tersebut kemudian di scaling dan digunakan sebagai model untuk dilakukan pemodelan dengan *Bagging Regressor*, yang kemudian kami tes dengan data train sejumlah 318 data, dan data tes sebanyak 80 data.

```
X = df.drop(['mpg','car_name',"origin","acceleration","horsepower"],axis='columns')
Y =df['mpg']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)
print("jumlah data train :",len(X_train))
print("jumlah data tes :",len(X_test))
```

```
jumlah data train : 318
jumlah data tes : 80
```

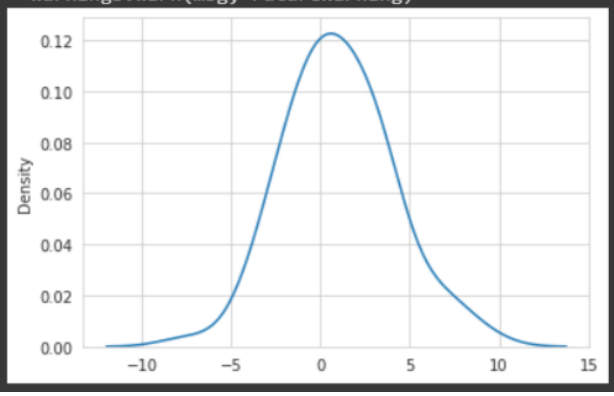
Lalu data tersebut kami evaluasi menggunakan MSE, R2 Score dan RMSE. Mean Squared Error (MSE) sendiri adalah metrik evaluasi yang menghitung rata-rata kuadrat dari selisih antara nilai target yang sebenarnya dengan nilai target yang diprediksi oleh model. Semakin kecil nilai MSE, semakin baik model yang digunakan.

Kami mendapatkan nilai metrik evaluasi sebagai berikut

```
[ ]  print("skor train R2 :",br.score(X_train_scaled,Y_train))
     print("skor Test R2 :",br.score(X_test_scaled,Y_test))
     print("skor Test R2 (sklearn metrics) :",r2_score(Y_test,Y_pred))
     print("MSE :",mean_squared_error(Y_test,Y_pred))
     print("RMSE :",mean_squared_error(Y_test,Y_pred,squared=False))

     skor train R2 : 0.9680305254745019
     skor Test R2 : 0.8609847795401074
     skor Test R2 (sklearn metrics) : 0.8609847795401074
     MSE : 9.924115000000002
     RMSE : 3.150256338776259
```

Dengan nilai MSE sebesar 9.93 yang lebih baik dibandingkan dengan eksperimen yang kami lakukan dengan hanya 3 data train yang menghasilkan model underfitting. Kami pun mendapatkan hasil delta atau selisih yang baik, dimana selisih tersebar rata di kisaran -5 hingga 5, dengan densitas tertinggi di ~0.13

```
[ ]  # Menampilkan selisih data Y_test dan Y_pred
     delta = Y_test - Y_pred
     sns.set_style('whitegrid')
     sns.kdeplot(np.array(delta), bw=0.5)
     plt.show()

     /usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:1699: FutureWarning:
       warnings.warn(msg, FutureWarning)
```



Bagging Regressor cocok digunakan untuk dataset seperti autos-mpg karena Bagging Regressor merupakan model pemodelan yang dapat digunakan untuk memprediksi nilai target pada data dengan tipe data apapun, termasuk data berupa angka float seperti yang terdapat pada dataset autos-mpg.

Selain itu, Bagging Regressor juga merupakan model pemodelan yang dapat menangani masalah overfitting pada data.Bagging Regressor dapat mengurangi risiko overfitting dengan menggunakan beberapa model yang terdiri dari sebagian data latih yang dipilih secara acak, sehingga model yang dihasilkan tidak terlalu kompleks dan lebih baik dalam memprediksi data baru.

Bagian 7
Link Source Code dan Video Presentasi

Source Code (.ipynb) :
https://colab.research.google.com/drive/15dD5FSsgYs_5mpmDmY8lkjq2EyS_rydq?usp=sharing

Video Presentasi : https://youtu.be/1J5vZzYK2qk