Rekomendasi Harga EV

BY SUSU BERUANG

Susu Beruang

Agenda

- Rumusan Masalah
- Hipotesis
- Data Pre-pocessing dan Cleaning
- Exploration Data Analysis
- Modelling And Hyperparameter Turning
- Feature Engineering
- Evaluation Model
- Conclusion and Recomendation

Susu Beruang

Rumusan Masalah dan Hipotesis

Back to Agenda Page

Rumusan Masalah

Apakah prediksi harga berdasarkan kapasitas baterai dan efesiensi dapat dijadikan dasar rekomendasi harga EV?

Hipotesis

- H0 : Prediksi harga berdasarkan kapasitas baterai dan efesiensi tidak dapat dijadikan dasar rekomendasi harga EV.
- H1 : Prediksi harga berdasarkan kapasitas baterai dan efesiensi dapat dijadikan dasar rekomendasi harga EV..

HIPOTESIS

Model Summary

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Harga(Rp)   R-squared:                       0.546
Model:                            OLS   Adj. R-squared:                  0.535
Method:                 Least Squares   F-statistic:                     49.83
Date:                Sat, 26 Aug 2023   Prob (F-statistic):           6.07e-15
Time:                        01:30:05   Log-Likelihood:                -1833.2
No. Observations:                  86   AIC:                             3672.
Df Residuals:                      83   BIC:                             3680.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                        -1.124e+09   3.49e+08     -3.221      0.002   -1.82e+09    -4.3e+08
Kapasitas Baterai (kWh)       2.934e+07   3.54e+06      8.292      0.000    2.23e+07    3.64e+07
Efisiensi (Wh/km)             1.156e+06   2.05e+06      0.564      0.574   -2.92e+06    5.23e+06
==============================================================================
Omnibus:                       11.295   Durbin-Watson:                   0.862
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               11.594
Skew:                           0.797   Prob(JB):                      0.00304
Kurtosis:                       3.834   Cond. No.                     1.51e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.51e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Y	-1.124e+09 + 2.934e+07X1 + 1.156e+06X2

Dari hipotesis yang telah dianalisis, didapati bahwa harga yang diprediksi tidak terlalu naik disebabkan **konstanta a menunjukkan angka yang negatif**

METODOLOGI DAN VARIABEL

METODOLOGI	VARIABEL
• Regresi • Alternative hypothesis • Machine Learning metode Supervised Learning	• Brand • Harga • Efisiensi • Kapasitas Baterai

Drop Data

- Duplicate rows
- Missing data

Manage outlier

- Menemukan IQR dari harga, efisiensi, dan kapasitas baterai mobil elektrik
- Menemukan pencilan atas dan bawah dari setiap items
- Menganalisis outlier

Statistik Dasar

- Agregasi Data
- Membuat model regresi, dengan sumbu-x efisiensi dan kapasitas baterai dan sumbu-y harga
- Menentukan prediksi harga berdasarkan model regresi

Pre-Processing and Cleaning Data

Pre-Processing and Cleaning Data

1 import important libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import statsmodels.api as sm
import math
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import graphviz
from sklearn.metrics import accuracy_score
from sklearn import tree
from numpy.lib.function_base import percentile
from sklearn.metrics import mean_squared_error
```

2 import dataset

```
data = pd.read_csv('ev_sum.csv')
data.head()
```

	Brand	Efisiensi (Wh/km)	Kapasitas Baterai (kWh)	Harga(Rp)
0	Audi e-tron GT quattro	202	85.0	1.712956e+09
1	Audi e-tron GT RS	210	85.0	2.321660e+09
2	Audi Q4 e-tron 40	189	76.6	9.877786e+08
3	Audi Q4 e-tron 50 quattro	199	76.6	1.112251e+09
4	Audi Q8 e-tron 55 quattro	214	106.0	1.517859e+09

3

```
#drop duplicate rows
data = data.drop_duplicates(ignore_index = True)
data
```

	Brand	Efisiensi (Wh/km)	Kapasitas Baterai (kWh)	Harga(Rp)
0	Audi e-tron GT quattro	202	85.0	1.712956e+09
1	Audi e-tron GT RS	210	85.0	2.321660e+09
2	Audi Q4 e-tron 40	189	76.6	9.877786e+08
3	Audi Q4 e-tron 50 quattro	199	76.6	1.112251e+09
4	Audi Q8 e-tron 55 quattro	214	106.0	1.517859e+09
...	...	...	...	...
81	Volkswagen ID.4 Pro	188	77.0	8.702324e+08
82	Volkswagen ID.4 Pro Performance	188	77.0	8.981313e+08
83	Volkswagen ID.4 Pure	182	52.0	7.578562e+08
84	Volkswagen ID.7 Pro	164	77.0	1.034017e+09
85	Volkswagen ID.7 Pro S	165	86.0	1.092546e+09

86 rows × 4 columns

4

```
#data information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86 entries, 0 to 85
Data columns (total 4 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Brand                     86 non-null     object
 1   Efisiensi (Wh/km)         86 non-null     int64
 2   Kapasitas Baterai (kWh)   86 non-null     float64
 3   Harga(Rp)                 86 non-null     float64
dtypes: float64(2), int64(1), object(1)
memory usage: 2.8+ KB
```

Pre-Processing and Cleaning Data

5

```
#missing data
data.isnull()
```

	Brand	Efisiensi (Wh/km)	Kapasitas Baterai (kWh)	Harga(Rp)
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False

6

```
#Outlier atas
#UPPER HARGA
upper_harga= Q3_harga + 1.5*IQR_harga
upper_harga_array=np.array(data['Harga(Rp)']>=upper_harga)
print("Upper harga: ", upper_harga)
print("Outlier above upper bound: ", upper_harga_array.sum())
#UPPER BATERAI
upper_baterai= Q3_baterai + 1.5*IQR_baterai
upper_baterai_array=np.array(data['Kapasitas Baterai (kWh)']>=upper_baterai)
print("Upper baterai: ", upper_baterai)
print("Outlier above upper bound: ", upper_baterai_array.sum())
#UPPER EFISIENSI
upper_efisiensi= Q3_efisiensi + 1.5*IQR_efisiensi
upper_efisiensi_array=np.array(data['Efisiensi (Wh/km)']>=upper_efisiensi)
print("Upper efisiensi: ", upper_efisiensi)
print("Outlier above upper bound: ", upper_efisiensi_array.sum())

#Outlier bawah
#LOWER HARGA
lower_harga= Q1_harga - 1.5*IQR_harga
lower_harga_array=np.array(data['Harga(Rp)']<=lower_harga)
print("Lower harga: ", lower_harga)
print("Outlier below lower bound: ", lower_harga_array.sum())
#LOWER BATERAI
lower_baterai= Q1_baterai - 1.5*IQR_baterai
lower_baterai_array=np.array(data['Kapasitas Baterai (kWh)']<=lower_baterai)
print("Lower baterai: ", lower_baterai)
print("Outlier below lower bound: ", lower_baterai_array.sum())
#LOWER EFISIENSI
lower_efisiensi= Q1_efisiensi - 1.5*IQR_efisiensi
lower_efisiensi_array=np.array(data['Efisiensi (Wh/km)']<=lower_efisiensi)
print("Lower efisiensi: ", lower_efisiensi)
print("Outlier below lower bound: ", lower_efisiensi_array.sum())
```

```
Upper harga:  3309463279.65
Outlier above upper bound:  0
Upper baterai:  123.625
Outlier above upper bound:  0
Upper efisiensi:  246.0
Outlier above upper bound:  3
Lower harga:  -547497236.7500001
Outlier below lower bound:  0
Lower baterai:  32.625
Outlier below lower bound:  0
Lower efisiensi:  126.0
Outlier below lower bound:  0
```

7

```
#IQR HARGA
Q1_harga = np.percentile(data['Harga(Rp)'], 25, method='midpoint')
Q3_harga = np.percentile(data['Harga(Rp)'], 75, method='midpoint')
IQR_harga = (Q3_harga - Q1_harga)
print('IQR_harga:', IQR_harga)
#IQR KAPASITAS BATERAI
Q1_baterai = np.percentile(data['Kapasitas Baterai (kWh)'], 25, method='midpoint')
Q3_baterai = np.percentile(data['Kapasitas Baterai (kWh)'], 75, method='midpoint')
IQR_baterai = (Q3_baterai - Q1_baterai)
print('IQR_baterai:', IQR_baterai)
#IQR EFISIENSI
Q1_efisiensi = np.percentile(data['Efisiensi (Wh/km)'], 25, method='midpoint')
Q3_efisiensi = np.percentile(data['Efisiensi (Wh/km)'], 75, method='midpoint')
IQR_efisiensi = Q3_efisiensi - Q1_efisiensi
print('IQR_efisiensi:', IQR_efisiensi)
```

```
IQR_harga: 964240129.1
IQR_baterai: 22.75
IQR_efisiensi: 30.0
```

Pre-Processing and Cleaning Data

8

Boolean value indicating the outlier rows

```
#UPPER
upper_harga_array = np.where(data['Harga(Rp)']>=upper_harga)[0]
print(upper_harga_array)
upper_baterai_array = np.where(data['Kapasitas Baterai (kWh)']>=upper_baterai)[0]
print(upper_baterai_array)
upper_efisiensi_array = np.where(data['Efisiensi (Wh/km)']>=upper_efisiensi)[0]
print(upper_efisiensi_array)
#LOWER
lower_harga_array = np.where(data['Harga(Rp)']<=lower_harga)[0]
print(lower_harga_array)
lower_baterai_array = np.where(data['Kapasitas Baterai (kWh)']<=lower_baterai)[0]
print(lower_baterai_array)
lower_efisiensi_array = np.where(data['Efisiensi (Wh/km)']<=lower_efisiensi)[0]
print(lower_efisiensi_array)
```

```
[]
[]
[41 42 43]
[]
[]
[]
```

9

REMOVE OUTLIERS

```
#DICANCEL
##karena  walaupun berada di bawah batas bawah/di atas bawah atas, nilainya masih normal dan faktanya terdapat beberapa
##mobil yang efisiensinya sedikit lebih besar
```

Statistik Dasar

1

```
df = pd.DataFrame(data)
df.head()
```

	Brand	Efisiensi (Wh/km)	Kapasitas Baterai (kWh)	Harga(Rp)
0	Audi e-tron GT quattro	202	85.0	1.712956e+09
1	Audi e-tron GT RS	210	85.0	2.321660e+09
2	Audi Q4 e-tron 40	189	76.6	9.877786e+08
3	Audi Q4 e-tron 50 quattro	199	76.6	1.112251e+09
4	Audi Q8 e-tron 55 quattro	214	106.0	1.517859e+09

2

```
#Data Aggregation
#DROP NON NUMERIC COLUMN
df_without_brand = df.drop(['Brand'], axis=1)

#DATA AGGREGATION
min_values = df.min()
print("NILAI MINIMUM = \n", min_values)
max_values = df.max()
print("NILAI MAKSIMUM = \n", max_values)
mean_values = df_without_brand.mean()
print("RATA-RATA = \n", mean_values)
std_values = df_without_brand.std()
print("STANDARD DEVIATION = \n", std_values)
correlation_matrix = df_without_brand.corr()
print("MATRIKS KORELASI = \n", correlation_matrix)
```

```
NILAI MINIMUM =
 Brand                      Audi Q4 e-tron 40
Efisiensi (Wh/km)                        142
Kapasitas Baterai (kWh)                 39.0
Harga(Rp)                       526665701.3
dtype: object
NILAI MAKSIMUM =
 Brand                      Volkswagen ID.7 Pro S
Efisiensi (Wh/km)                          295
Kapasitas Baterai (kWh)                  108.4
Harga(Rp)                       3157848135.0
dtype: object
RATA-RATA =
 Efisiensi (Wh/km)          1.907326e+02
Kapasitas Baterai (kWh)     7.849186e+01
Harga(Rp)                   1.399750e+09
dtype: float64
STANDARD DEVIATION =
 Efisiensi (Wh/km)          2.738760e+01
Kapasitas Baterai (kWh)     1.586379e+01
Harga(Rp)                   6.530516e+08
dtype: float64
MATRIKS KORELASI =
                          Efisiensi (Wh/km)  Kapasitas Baterai (kWh)  Harga(Rp)
Efisiensi (Wh/km)                  1.000000                0.509128   0.411370
Kapasitas Baterai (kWh)            0.509128                1.000000   0.737475
Harga(Rp)                          0.411370                0.737475   1.000000
```

Statistik Dasar

3

```
#Model Regresi
X = sm.add_constant(df[['Kapasitas Baterai (kWh)', 'Efisiensi (Wh/km)']])
y = df['Harga(Rp)']

model = sm.OLS(y, X).fit()
print(model.summary())
## y = -1,575x10^-9 + 2.726*10^7 X1 + 4.488*10^6 X2
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Harga(Rp)   R-squared:                       0.546
Model:                            OLS   Adj. R-squared:                  0.535
Method:                 Least Squares   F-statistic:                     49.83
Date:                Sat, 26 Aug 2023   Prob (F-statistic):           6.07e-15
Time:                        07:08:02   Log-Likelihood:                -1833.2
No. Observations:                  86   AIC:                             3672.
Df Residuals:                      83   BIC:                             3680.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                    -1.124e+09   3.49e+08     -3.221      0.002   -1.82e+09    -4.3e+08
Kapasitas Baterai (kWh)   2.934e+07   3.54e+06      8.292      0.000    2.23e+07    3.64e+07
Efisiensi (Wh/km)         1.156e+06   2.05e+06      0.564      0.574   -2.92e+06    5.23e+06
==============================================================================
Omnibus:                       11.295   Durbin-Watson:                   0.862
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               11.594
Skew:                           0.797   Prob(JB):                      0.00304
Kurtosis:                       3.834   Cond. No.                     1.51e+03
==============================================================================
```

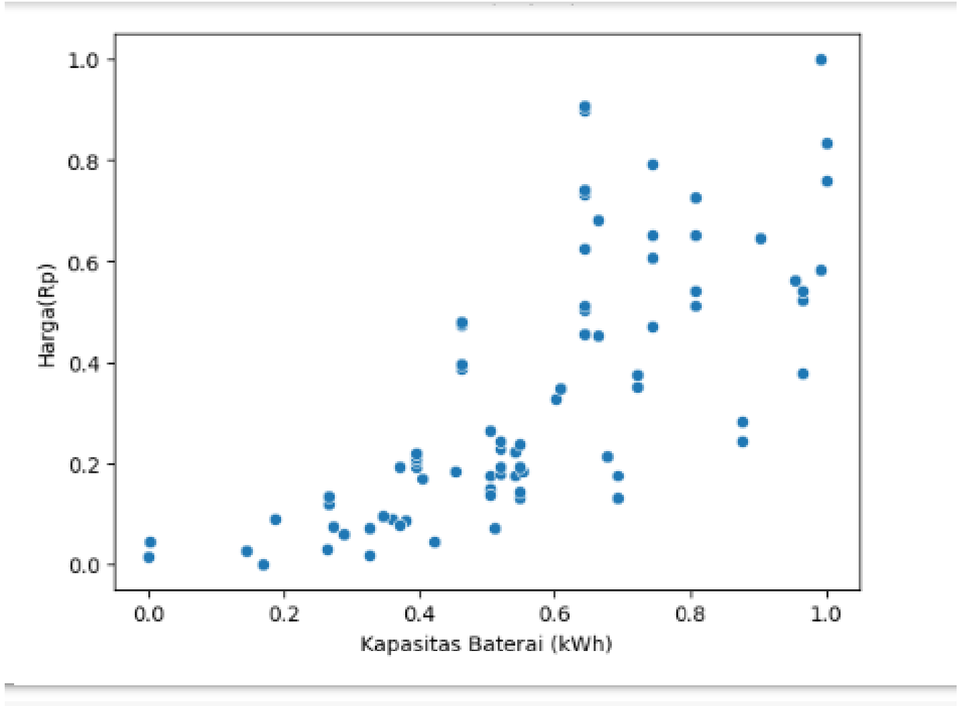
4

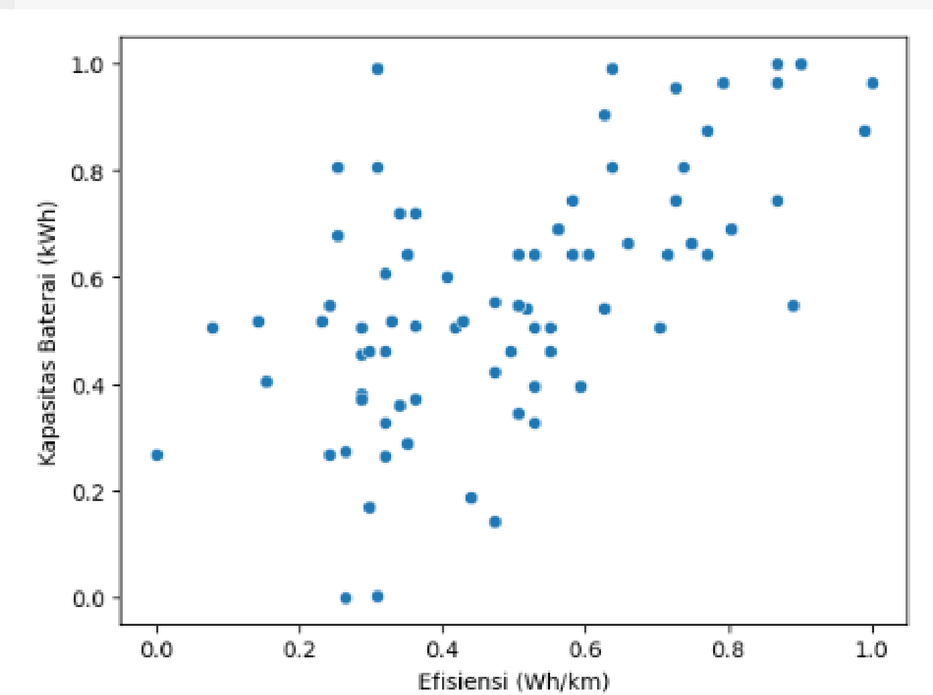
```
#Harga Prediksi
predicted_values = model.predict(X)
df['Predicted_Harga'] = predicted_values
print(df)
```

```
                            Brand  Efisiensi (Wh/km)  \
0              Audi e-tron GT quattro                202
1                   Audi e-tron GT RS                210
2                   Audi Q4 e-tron 40                189
3          Audi Q4 e-tron 50 quattro                199
4          Audi Q8 e-tron 55 quattro                214
..                               ...                ...
81               Volkswagen ID.4 Pro                188
82  Volkswagen ID.4 Pro Performance                 188
83              Volkswagen ID.4 Pure                182
84               Volkswagen ID.7 Pro                164
85             Volkswagen ID.7 Pro S                165

    Kapasitas Baterai (kWh)     Harga(Rp)  Predicted_Harga
0                      85.0  1.712956e+09     1.603741e+09
1                      85.0  2.321660e+09     1.612985e+09
2                      76.6  9.877786e+08     1.342234e+09
3                      76.6  1.112251e+09     1.353790e+09
4                     106.0  1.517859e+09     2.233817e+09
..                      ...           ...              ...
81                     77.0  8.702324e+08     1.352816e+09
82                     77.0  8.981313e+08     1.352816e+09
83                     52.0  7.578562e+08     6.122994e+08
84                     77.0  1.034017e+09     1.325082e+09
85                     86.0  1.092546e+09     1.590327e+09

[86 rows x 5 columns]
```

Exploration Data Analysis

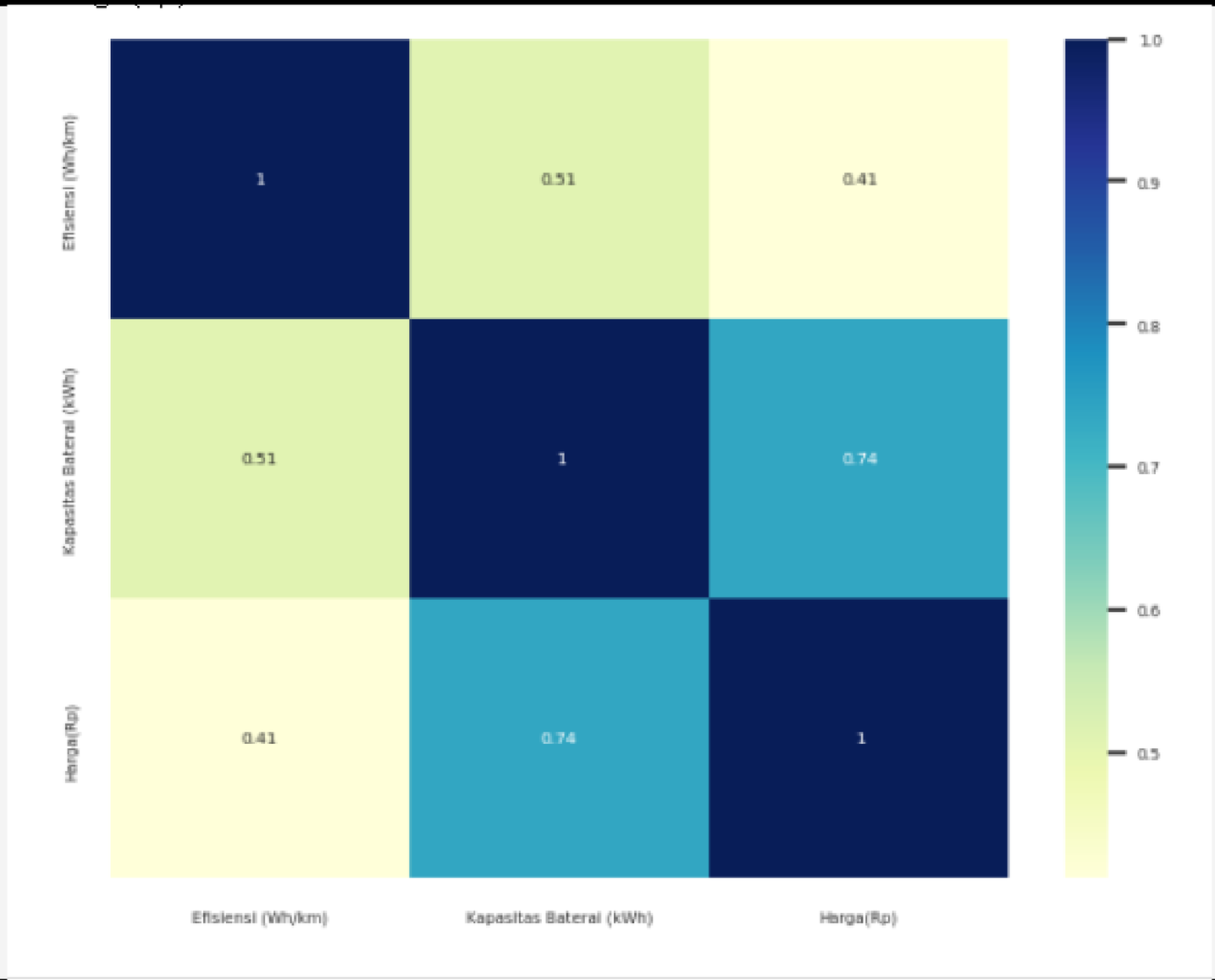
Scatter Plot





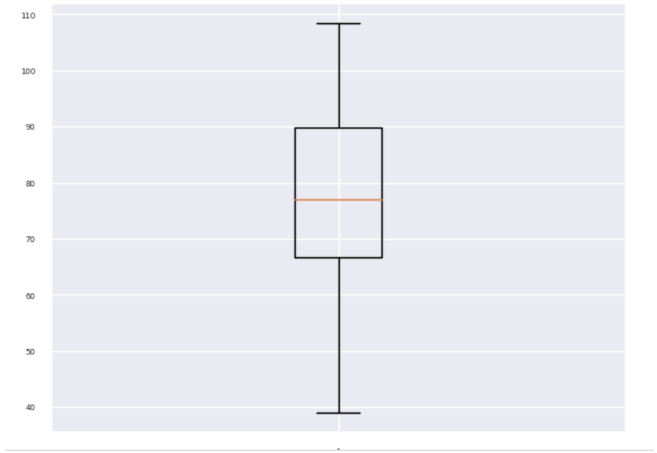
Exploration Data Analysis

Heatmap

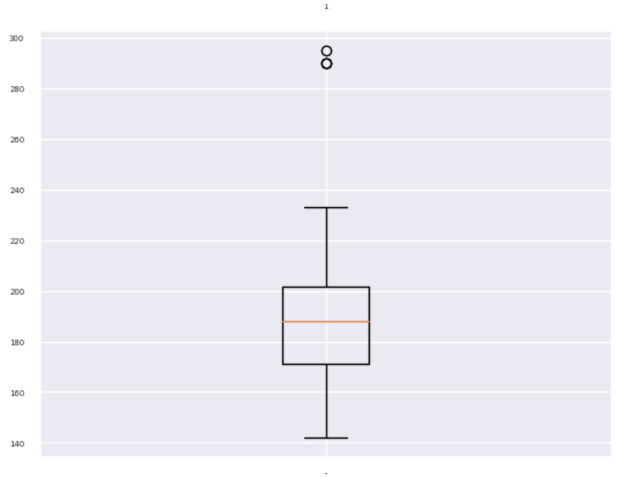


Exploration Data Analysis

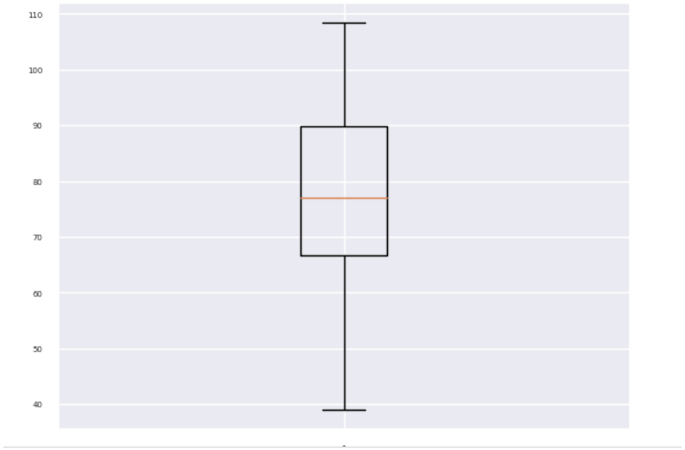
Boxplot



Harga



Efisiensi



Kapasitas Baterai

Susu Beruang

Machine Learning, Modeling, and Hyperparameter Tuning

- **Menambahkan kolom True (Recommend) or No (Not Recommend)**

```
: #Menambahkan Harga Maksimum
df['rekomendasi'] = df['Harga(Rp)'] <= df['Predicted_Harga']
print(df.to_string())
```

```
                                    Brand  Efisiensi (Wh/km)  Kapasitas Baterai (kWh)     Harga(Rp)  Predicted_Harga  r
ekomendasi
0                   Audi e-tron GT quattro                202                     85.0  1.712956e+09     1.603741e+09
False
1                         Audi e-tron GT RS                210                     85.0  2.321660e+09     1.612985e+09
False
2                         Audi Q4 e-tron 40                189                     76.6  9.877786e+08     1.342234e+09
True
3                  Audi Q4 e-tron 50 quattro                199                     76.6  1.112251e+09     1.353790e+09
True
4                  Audi Q8 e-tron 55 quattro                214                    106.0  1.517859e+09     2.233817e+09
True
5                           Audi SQ8 e-tron                 233                    106.0  1.902201e+09     2.255773e+09
True
6                  Audi SQ8 e-tron Sportback                221                    106.0  1.950975e+09     2.241906e+09
True
7                          BMW i4 eDrive35                  156                     67.0  9.753900e+08     1.022404e+09
True
8                               BMW i4 M50                  179                     80.7  1.386851e+09     1.450986e+09
True
```

- **Memisahkan target variabel**

```
#Separating the target variable
X = df.values[:, 1:5]
Y = df.values[:,5]
Y = Y.astype('int')

#Splitting dataset into test and train
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2, random_state = 100)
```

Susu Beruang

Machine Learning, Modeling, and Hyperparameter Tuning

- **Hyperparameter Tuning**

```
#Hyperparameter Tuning
clf_entropy.tree_.max_depth
```

```
2
```

```
clf_entropy.score(X_train, y_train)
```

```
0.8235294117647058
```

```
#FIRST TRIAL
#Function to perform training with Entropy
clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state=100, max_depth=3, min_samples_leaf=5)
clf_entropy.fit(X_train, y_train)

#Function to make prediction
y_pred_en = clf_entropy.predict(X_test)
print(y_pred_en)

#Checking accuracy
print("Accuracy is ", accuracy_score(y_test,y_pred_en)*100)

#SECOND TRIAL
#Function to perform training with Entropy
clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state=100, max_depth=8, min_samples_leaf=8)
clf_entropy.fit(X_train, y_train)

#Function to make prediction
y_pred_en = clf_entropy.predict(X_test)
print(y_pred_en)

#Checking accuracy
print("Accuracy is ", accuracy_score(y_test,y_pred_en)*100)

#THIRD TRIAL
#Function to perform training with Entropy
clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state=100, max_depth=10, min_samples_leaf=10)
clf_entropy.fit(X_train, y_train)

#Function to make prediction
y_pred_en = clf_entropy.predict(X_test)
print(y_pred_en)

#Checking accuracy
print("Accuracy is ", accuracy_score(y_test,y_pred_en)*100)
```

```
[0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1]
Accuracy is  83.33333333333334
[0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 1 0 1]
Accuracy is  77.77777777777779
[0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1]
Accuracy is  72.22222222222221
```

- **MSE**

```
y_pred = clf_entropy.predict(X_test)
math.sqrt(mean_squared_error(y_test, y_pred))
###semakin kecil nilai MSE maka semakin baik kualitas model
```

```
0.408248290463863
```

Conclusion and Recommendation

Access our TableAU in here :
Click this

Conclusion

Semakin tinggi kapasitas baterai dan efisiensi mobil listrik, maka harga mobil listrik akan semakin tinggi. Dari korelasi ini didapatkan prediksi di mana harga prediksi yang dapat memberikan harga rekomendasi (harga <= harga prediksi

Recommendation

Dari data analisis yang telah kami lakukan, hal tersebut dapat digunakan untuk mengakses lebih dalam mengenai rekomendasi harga mobil sesuai dengan kapasitas baterai dan efisiensi