TRIBHUVAN UNIVERSITY

Faculty of Management

National College of Computer Studies

Paknajol, Kathmandu

Python project on the topic of Car Price Regression

Submitted By:

Name: Anish Maharjan

BIM 5th Semester

Sec: A

Roll no: 01

Submitted To:

Mausam Rajbanshi

Letter of Certificate

This is to certify that the project report entitled “Car Price Regression”, is the work who helped us in our project under the guidance and supervision.

To the best of my knowledge and belief, this work embodies the work of candidates themselves, has duly been completed, fulfills the requirement of the ordinance relating to the bachelor degree of the university and is up to the standard in respect of content, presentation and language for begins referred to the examiner.

____

Signature of Invigilator

Acknowledgement

I would like to acknowledge the use of data regression analysis as a fundamental component of this report on car prices. Data regression allowed us to analyze and model the relationship between various independent variables and the dependent variable, namely car prices, using Python programming language. This analysis was crucial in gaining insights into the factors influencing car prices and ultimately enhancing the quality of this report. The regression analysis provided a robust framework for exploring and quantifying the impact of factors such as car attributes, market conditions, and other relevant variables on car prices. It allowed us to develop predictive models that can aid in decision-making processes and offer valuable information for the automotive industry.

Table of Contents

Introduction

Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. It is used for:

- Web development(server-side),
- Software development,
- Mathematics,
- System scripting

Why Python?

- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- It runs on the interpreter system i.e. the code can be executed as soon as it is written. Thus, prototyping can be very quick.
- It can be treated in a procedural way, an object oriented or a functional way.
- It also has simple syntax similar to the English language.

What Python can do?

- Python can be used on a server to create web applications.
- Python can be used for rapid prototyping, or for production-ready software development.
- Python can be used to handle big data and perform complex mathematics.
- Python can be connect to database systems. It can also read and modify files.

Background

In today's data-driven world, businesses and individuals are increasingly relying on data-driven insights to make informed decisions. Predicting car prices is a common task in the automotive industry, as well as for consumers looking to buy or sell cars. One powerful tool for this task is data regression, a statistical technique used to model the relationship between a dependent variable and one or more independent variables.

Steps in Car Price Prediction using Data Regression:

1. **Data Collection**: Gather a dataset containing information about various cars, including their prices and relevant features.
2. **Data Preprocessing:** Clean and preprocess the data by handling missing values, encoding categorical variables, and scaling numerical features if necessary.
3. **Data Exploration**: Explore the dataset through summary statistics, visualizations, and correlations to gain insights into the data.
4. **Feature Selection:** Identify the most relevant features that are likely to influence car prices. Feature selection helps in building more accurate models.
5. **Data Splitting:** Split the dataset into a training set and a test set. The training set is used to train the regression model, while the test set is used to evaluate its performance.
6. **Model Training:** Apply a regression algorithm (e.g., Linear Regression) to the training data to learn the relationship between the independent variables and car prices.
7. **Model Evaluation:** Evaluate the model's performance using appropriate metrics such as Mean Squared Error (MSE), R-squared, or Root Mean Squared Error (RMSE).
8. **Prediction:** Use the trained model to make predictions on new, unseen data.
9. **Interpretation:** Interpret the model's coefficients to understand the impact of each independent variable on car prices.
10. **Conclusion:** Summarize the findings, discuss the model's accuracy, and provide insights into factors affecting car prices.

This report will follow a structured approach to demonstrate how data regression techniques in Python can be utilized to predict car prices effectively. It will include data preprocessing, model selection, evaluation, and interpretation sections, with the goal of providing a reliable car price prediction model.

Objectives

- To analyze and classify car prices using machine learning algorithms that will help consumers to know about the prices of the car or any mechanical devices.
- To identify the correlations between the parameters that are likely to be responsible for changes in prices.
- To provide valuable insights into what drives car pricing trends and fluctuations.

Implementation

The following steps outline the key implementation elements:

1. Data Preparation:
    - Import necessary libraries (Numpy, pandas, scikit-learn).
    - Load the car price dataset (CSV or Excel).
    - Check for and handle missing values, duplicates, and inconsistent data types.
2. Exploratory Data Analysis (EDA):
    - Perform EDA to understand the dataset.
    - Create summary statistics, visualizations, and correlation matrices.
    - Identify influential features that likely affect car prices.
3. Data Splitting:
    - Split the dataset into training and testing subsets (e.g., 80/20 or 70/30).
    - Ensure randomization to avoid bias.
4. Model Selection:
    - Choose an appropriate regression model (e.g., Linear Regression, Ridge, Lasso).
    - Initialize the selected regression model.
5. Model Training:
    - Fit the regression model to the training data.
    - Tune model hyper parameters if needed for better performance.
6. Model Evaluation:
    - Assess the model's performance using metrics like MSE, or RMSE.
    - Evaluate its generalization on the testing dataset.
7. Predictions:
    - Use the trained model to predict car prices on the testing dataset.
    - Compare predicted prices to actual prices to gauge model accuracy.
8. Interpretation:
    - Analyze feature coefficients to understand their impact on car prices.
    - Identify the most influential features affecting price predictions.
9. Model Visualization:
    - Create visualizations (e.g., scatter plots, residual plots) to visualize model performance and assumptions.
10. Reporting and Conclusion:
    - Summarize findings, including model accuracy and feature insights.
    - Discuss practical implications for the automotive industry.
    - Address any limitations and suggest areas for future research.

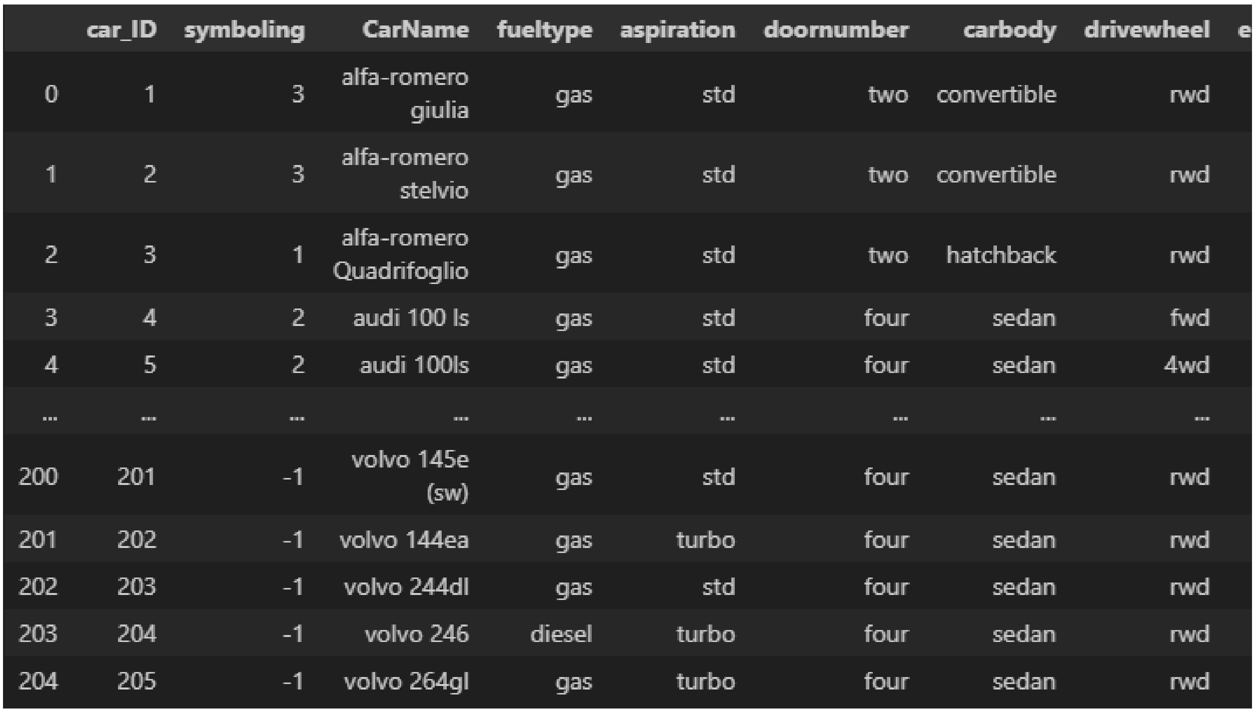
Codes used in this project are:

Importing Modules

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

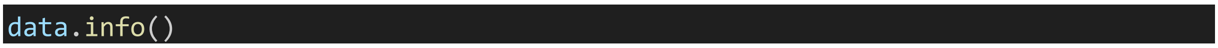
Loading Dataset

```
data=pd.read_csv("./CarPrice_Assignment.csv")
data
```

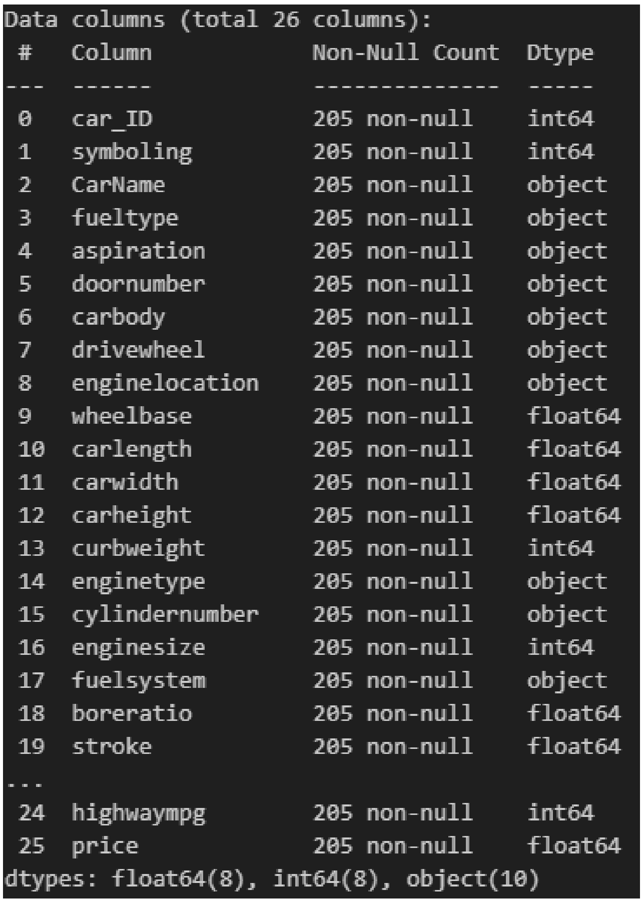


	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	e
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	

Dataset Information

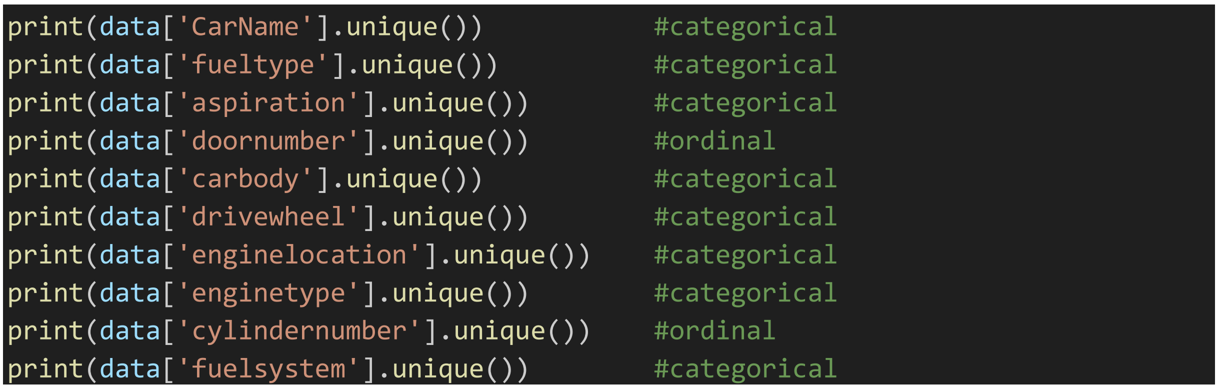


```
data.info()
```



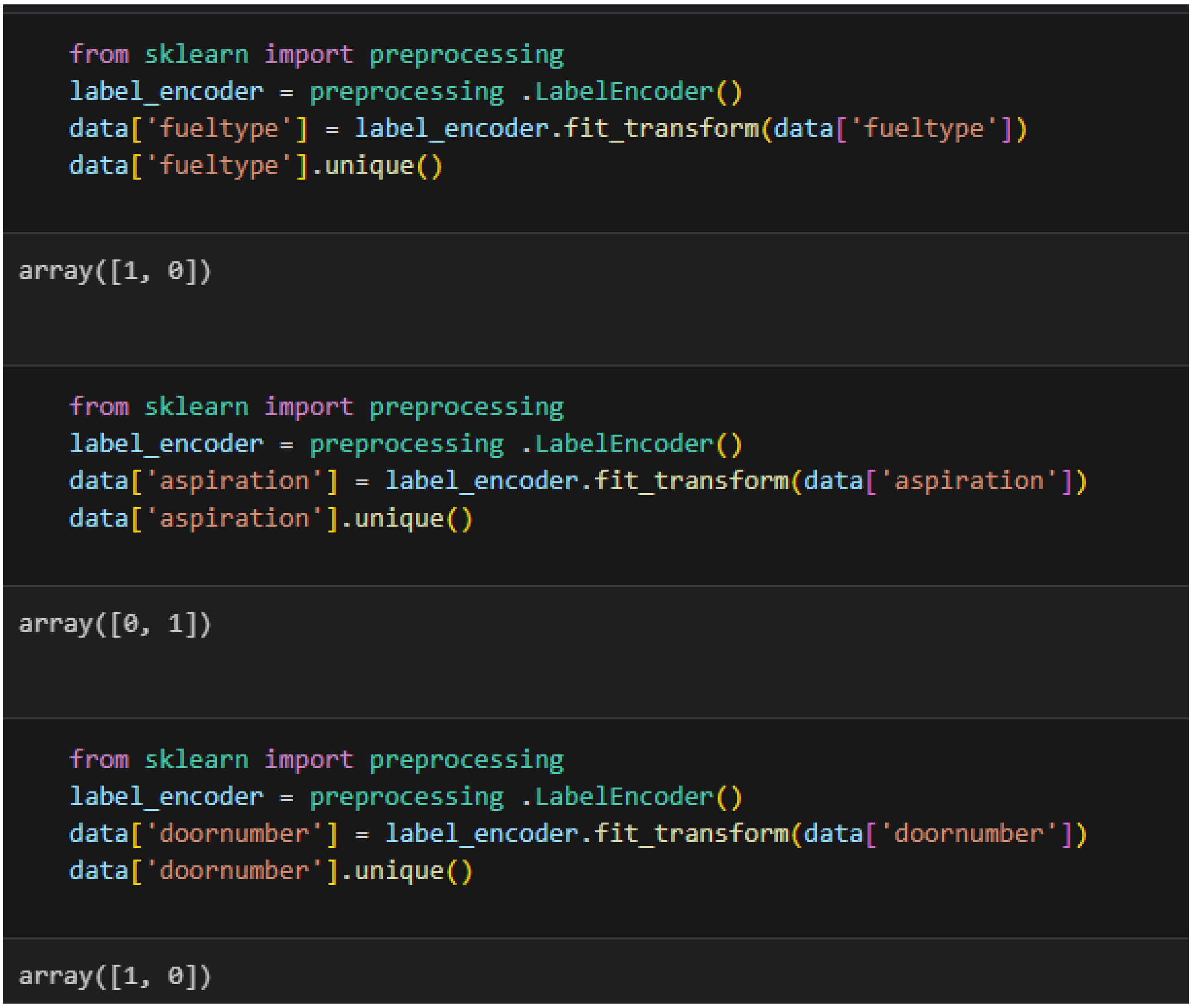
```
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
...
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
```

Identifying columns i.e. categorical and ordinal



```
print(data['CarName'].unique())          #categorical
print(data['fueltype'].unique())         #categorical
print(data['aspiration'].unique())       #categorical
print(data['doornumber'].unique())       #ordinal
print(data['carbody'].unique())          #categorical
print(data['drivewheel'].unique())       #categorical
print(data['enginelocation'].unique())   #categorical
print(data['enginetype'].unique())       #categorical
print(data['cylindernumber'].unique())   #ordinal
print(data['fuelsystem'].unique())       #categorical
```

Data Preprocessing



```
from sklearn import preprocessing
label_encoder = preprocessing .LabelEncoder()
data['fueltype'] = label_encoder.fit_transform(data['fueltype'])
data['fueltype'].unique()
```

```
array([1, 0])
```

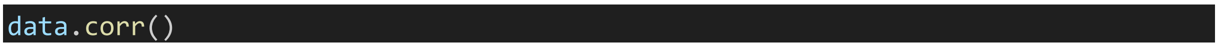
```
from sklearn import preprocessing
label_encoder = preprocessing .LabelEncoder()
data['aspiration'] = label_encoder.fit_transform(data['aspiration'])
data['aspiration'].unique()
```

```
array([0, 1])
```

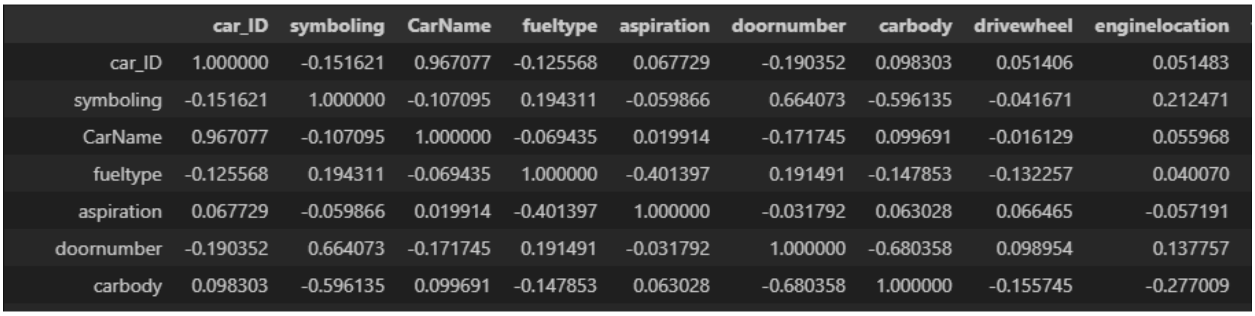
```
from sklearn import preprocessing
label_encoder = preprocessing .LabelEncoder()
data['doornumber'] = label_encoder.fit_transform(data['doornumber'])
data['doornumber'].unique()
```

```
array([1, 0])
```

Correlation



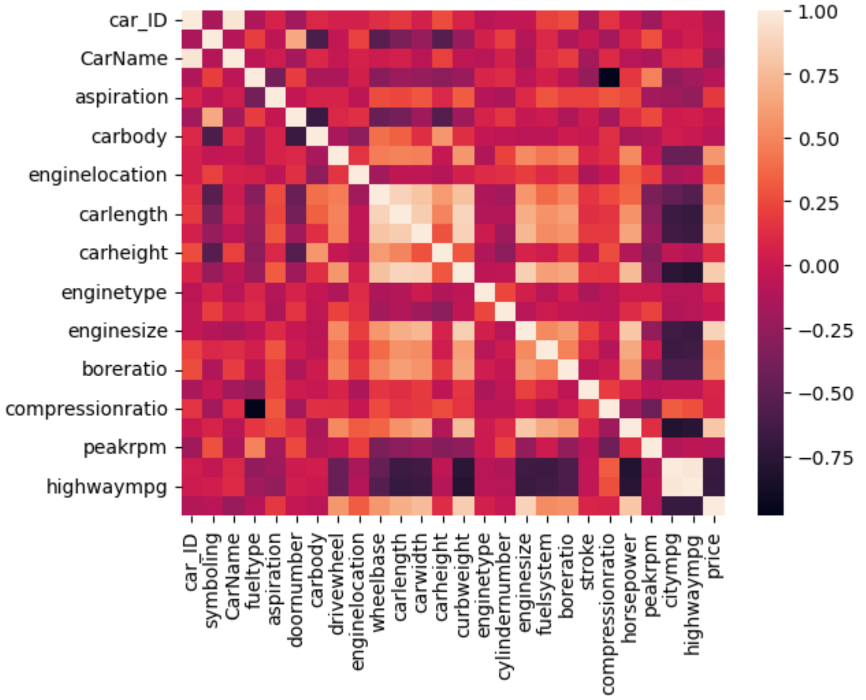
```
data.corr()
```



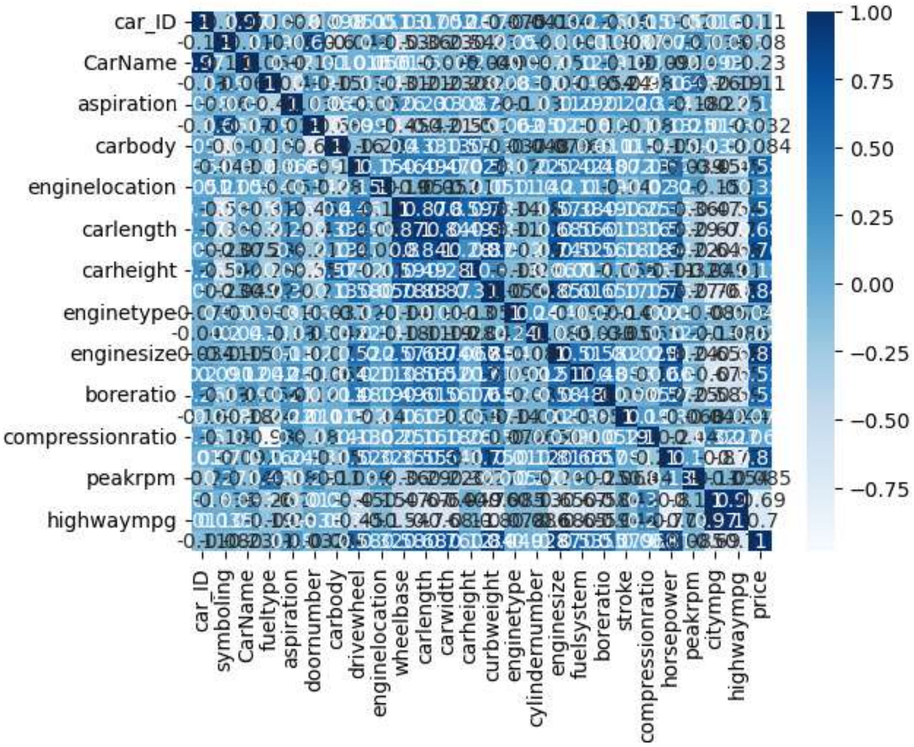
	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation
car_ID	1.000000	-0.151621	0.967077	-0.125568	0.067729	-0.190352	0.098303	0.051406	0.051483
symboling	-0.151621	1.000000	-0.107095	0.194311	-0.059866	0.664073	-0.596135	-0.041671	0.212471
CarName	0.967077	-0.107095	1.000000	-0.069435	0.019914	-0.171745	0.099691	-0.016129	0.055968
fueltype	-0.125568	0.194311	-0.069435	1.000000	-0.401397	0.191491	-0.147853	-0.132257	0.040070
aspiration	0.067729	-0.059866	0.019914	-0.401397	1.000000	-0.031792	0.063028	0.066465	-0.057191
doornumber	-0.190352	0.664073	-0.171745	0.191491	-0.031792	1.000000	-0.680358	0.098954	0.137757
carbody	0.098303	-0.596135	0.099691	-0.147853	0.063028	-0.680358	1.000000	-0.155745	-0.277009

Heatmap

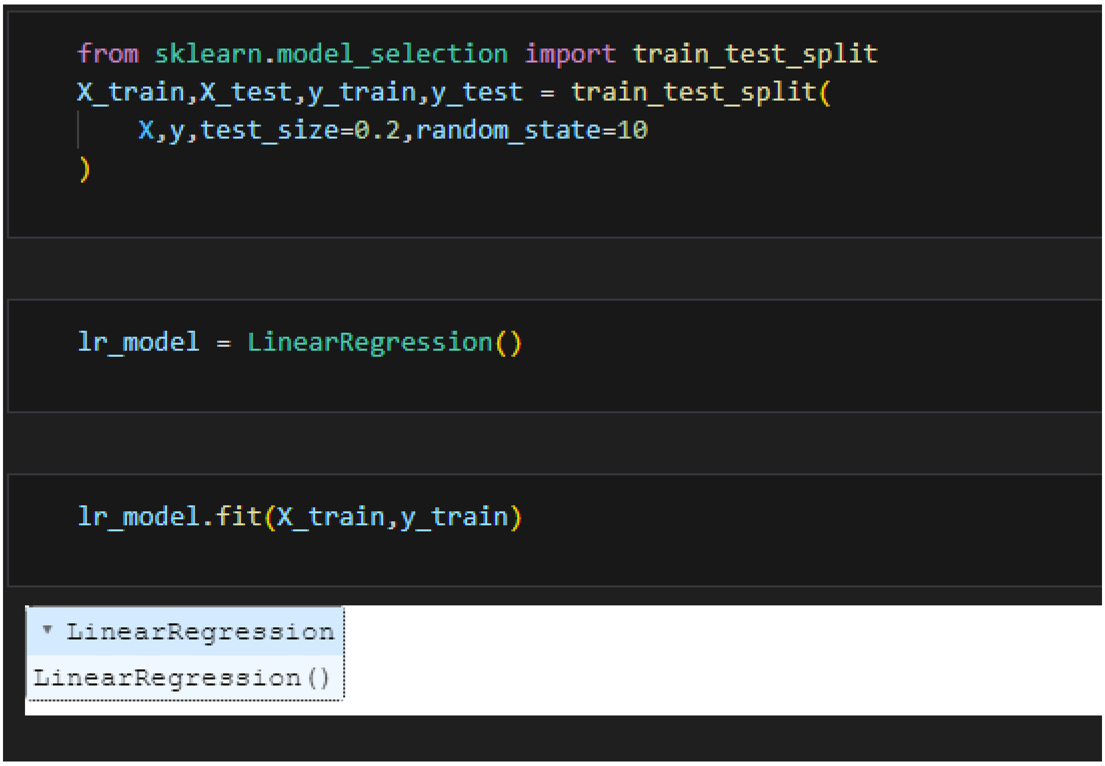
```
sns.heatmap(data.corr())
```



```
sns.heatmap(data.corr(), cmap="Blues", annot=True)
```



Linear Regression

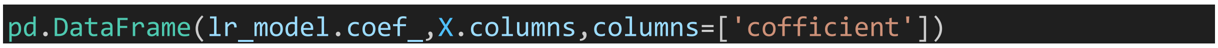


```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(
    X,y,test_size=0.2,random_state=10
)
```

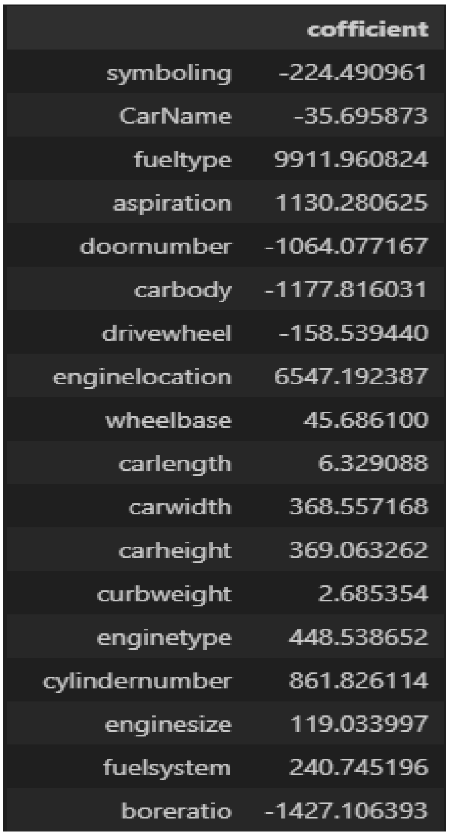
```
lr_model = LinearRegression()
```

```
lr_model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

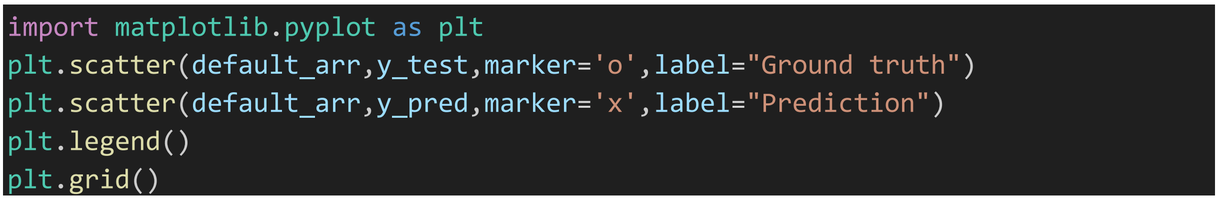


```
pd.DataFrame(lr_model.coef_,X.columns,columns=['cofficient'])
```

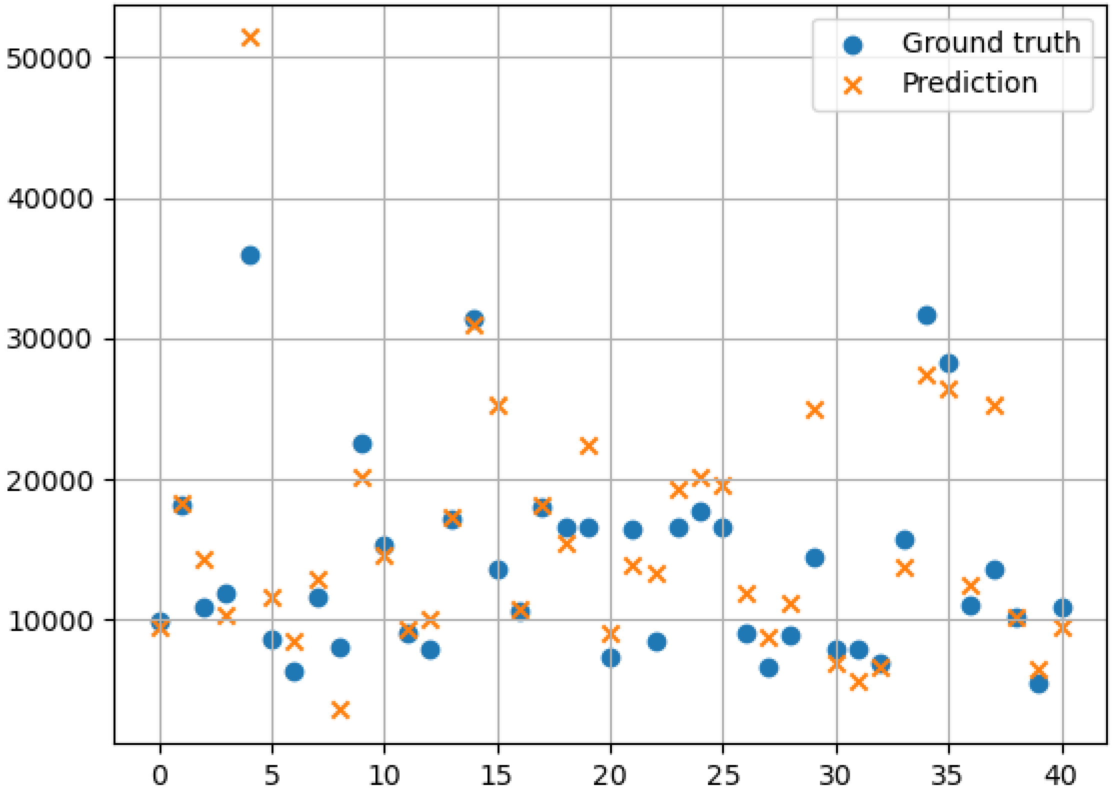


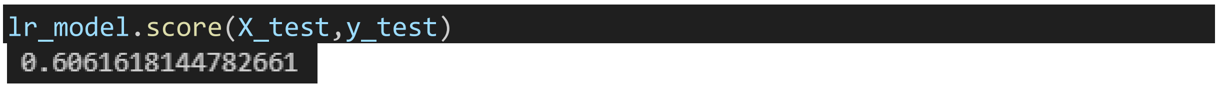
	cofficient
symboling	-224.490961
CarName	-35.695873
fueltype	9911.960824
aspiration	1130.280625
doornumber	-1064.077167
carbody	-1177.816031
drivewheel	-158.539440
enginelocation	6547.192387
wheelbase	45.686100
carlength	6.329088
carwidth	368.557168
carheight	369.063262
curbweight	2.685354
enginetype	448.538652
cylindernumber	861.826114
enginesize	119.033997
fuelsystem	240.745196
boreratio	-1427.106393

Scatterplot



```
import matplotlib.pyplot as plt
plt.scatter(default_arr,y_test,marker='o',label="Ground truth")
plt.scatter(default_arr,y_pred,marker='x',label="Prediction")
plt.legend()
plt.grid()
```

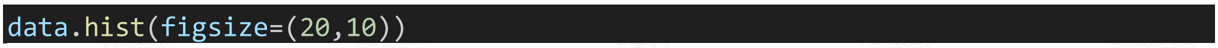




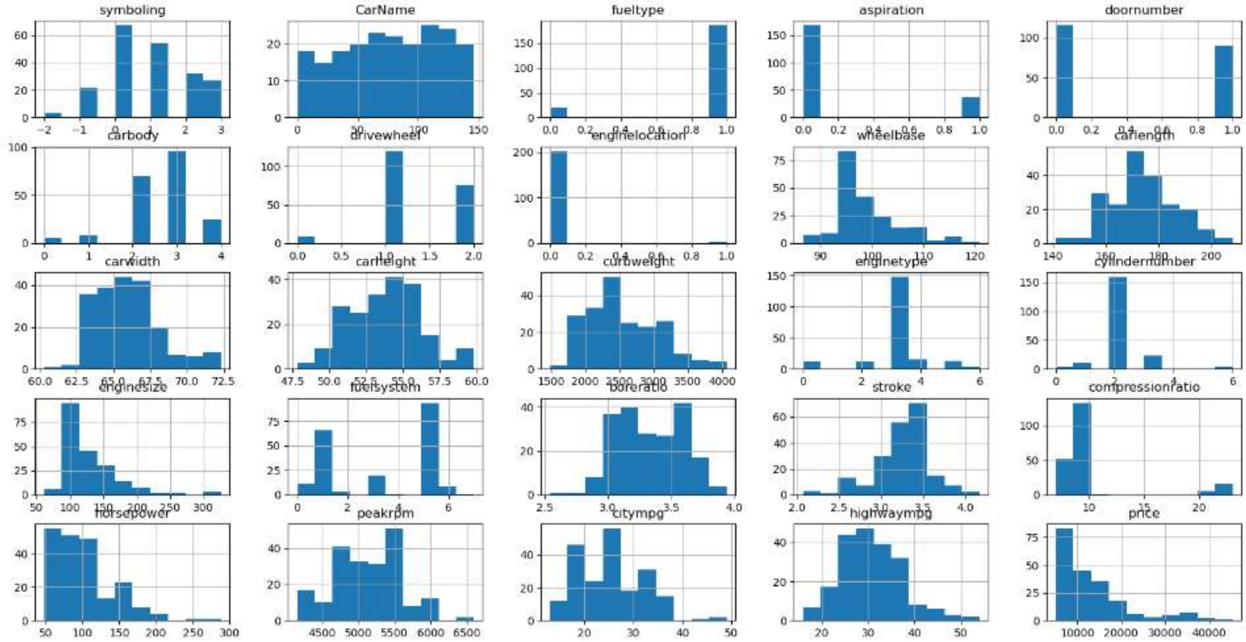
```
lr_model.score(X_test,y_test)
```

```
0.6061618144782661
```

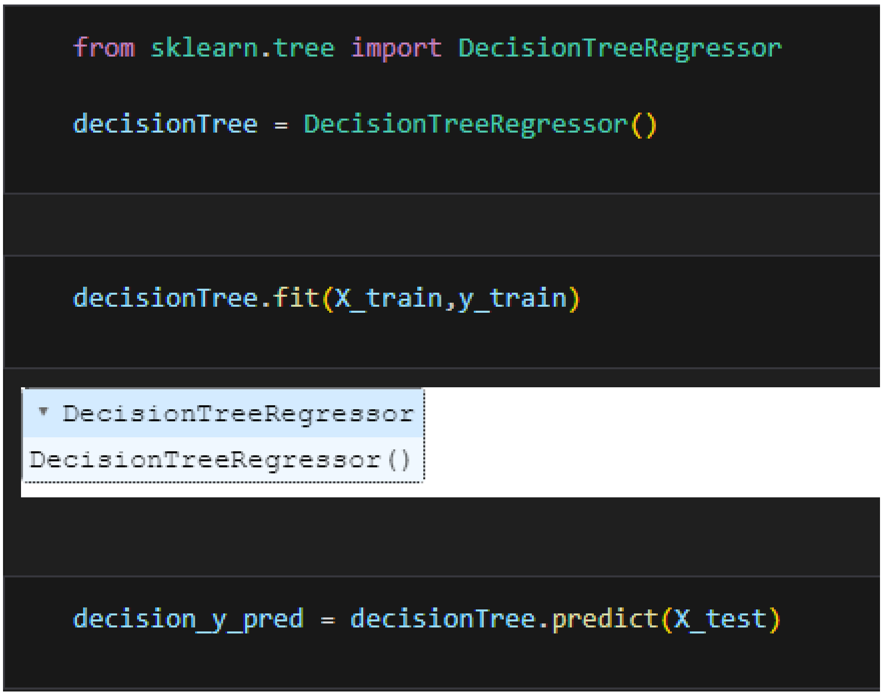
Histogram



```
data.hist(figsize=(20,10))
```



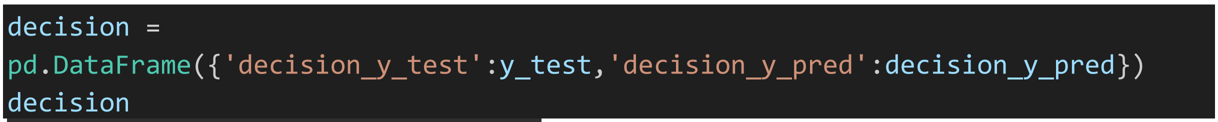
Decision Tree



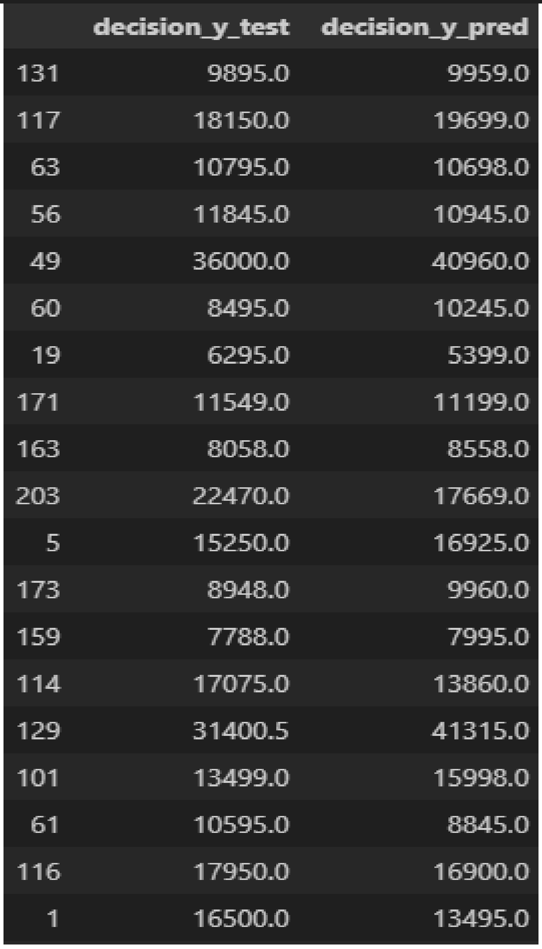
```
from sklearn.tree import DecisionTreeRegressor

decisionTree = DecisionTreeRegressor()


decisionTree.fit(X_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

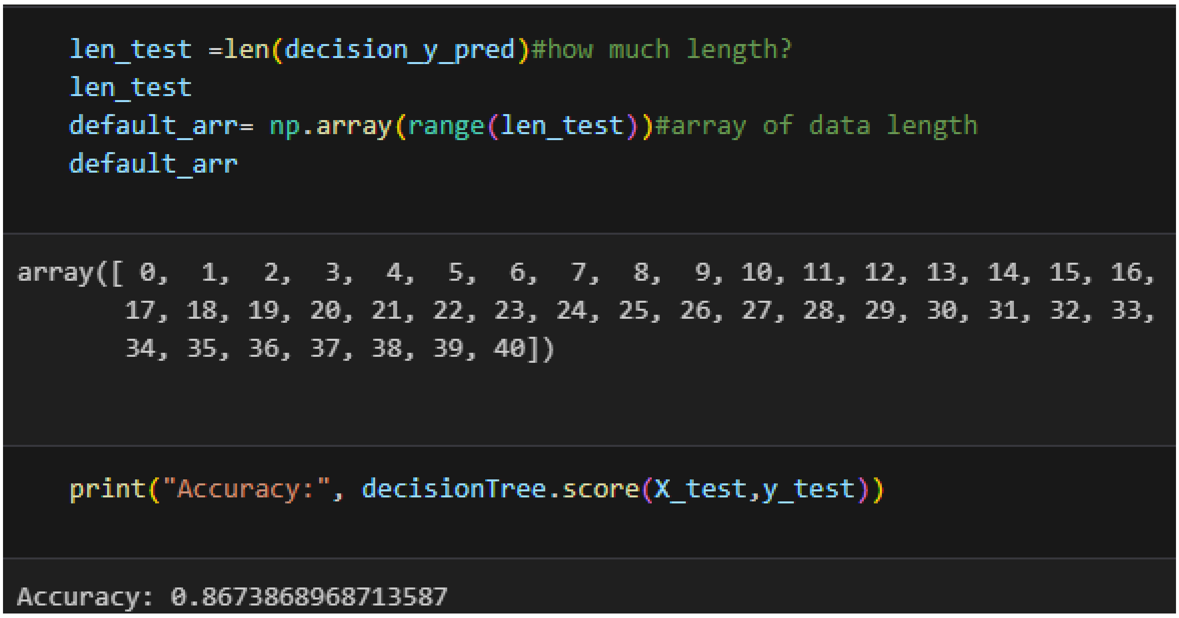
```
decision_y_pred = decisionTree.predict(X_test)
```



```
decision =
pd.DataFrame({'decision_y_test':y_test,'decision_y_pred':decision_y_pred})
decision
```



	decision_y_test	decision_y_pred
131	9895.0	9959.0
117	18150.0	19699.0
63	10795.0	10698.0
56	11845.0	10945.0
49	36000.0	40960.0
60	8495.0	10245.0
19	6295.0	5399.0
171	11549.0	11199.0
163	8058.0	8558.0
203	22470.0	17669.0
5	15250.0	16925.0
173	8948.0	9960.0
159	7788.0	7995.0
114	17075.0	13860.0
129	31400.5	41315.0
101	13499.0	15998.0
61	10595.0	8845.0
116	17950.0	16900.0
1	16500.0	13495.0

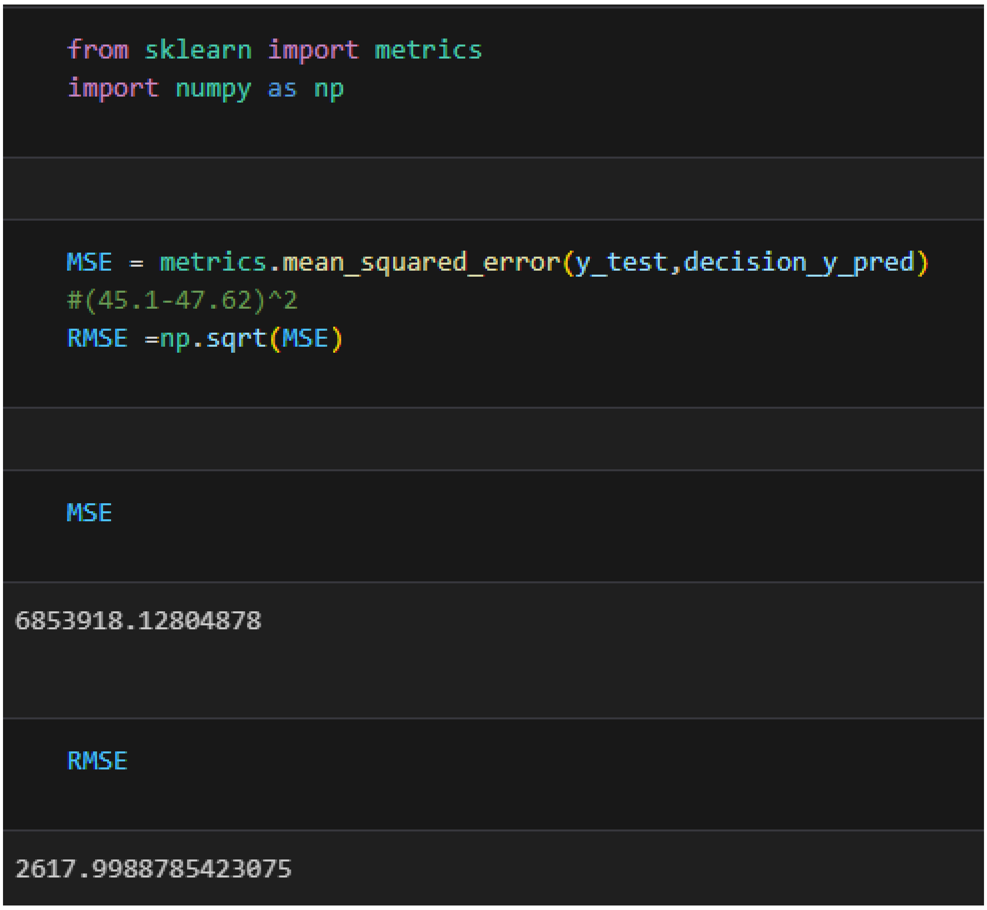


```
len_test =len(decision_y_pred)#how much length?
len_test
default_arr= np.array(range(len_test))#array of data length
default_arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40])
```

```
print("Accuracy:", decisionTree.score(X_test,y_test))
```

```
Accuracy: 0.8673868968713587
```



```
from sklearn import metrics
import numpy as np
```

```
MSE = metrics.mean_squared_error(y_test,decision_y_pred)
#(45.1-47.62)^2
RMSE =np.sqrt(MSE)
```

```
MSE
```

```
6853918.12804878
```

```
RMSE
```

```
2617.9988785423075
```

Conclusion

In conclusion, this Python project on car price regression has equipped us with valuable skills and insights into the world of machine learning and data analysis. It showcases the power of Python and its libraries in solving real-world problems and provides a foundation for future endeavors in predictive modeling and data-driven decision-making within the automotive industry.