Car Price Prediction

Problem Statement: We must predict the prices of a car using the knowledge about its features.

The prediction of car prices from its features is really an interesting machine learning problem. There are numerous factors that impact or influence the prices of a car in the market. In the following problem we will exploring a dataset which is based on the sale/purchases of cars. Our end goal with this dataset is to predict the prices of the car given its features in order to maximize the profit.

About the dataset

For this problem we will be using an open source Auto Mobile Dataset. This dataset is created by Jeffrey C. Schlimmer.

Sources of the dataset are as follows:

- 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook.
- Personal Auto Manuals, Insurance Services Office, 160 Water Street, New York, NY 10038
- Insurance Collision Report, Insurance Institute for Highway Safety, Watergate 600, Washington, DC 2003

The data set of *Auto Mobile* consists of three types of entities:

- The car's specifications based on its features
- Insurance risk rating of a car,
- its normalized losses in use as compared to other cars present in the dataset.

The second rating compares to how much the auto is riskier than its cost shows. Vehicles are initially assigned a risk factor symbol related with their cost. If it's riskier than this symbol is adjusted accordingly by moving it up or down the given scale. Statisticians call this cycle "symboling". An estimation of +3 shows that the auto is hazardous, - 3 that it is presumably protected.

The third factor is the relative normal misfortune instalment per safeguarded vehicle year. This worth is normalized for all automobiles inside a size arrangement (two-entryway little, station wagons, sports, etc...), and speaks to the average loss per vehicle every year.

Summary of Dataset:

Characteristics:	Multivariate	**Instances:**	205
Attribute Characteristics:	Categorical, Integer, Real	**Total Attributes:**	26
Tasks:	Regression	**Any Nulls?**	Yes

Initial Steps:

We want to predict the prices of cars from its features. In order to achieve that we must find out:

- the variables which have significant impact of the prices of car
- How does these characteristics influence the price of car?

Importing Relevant Libraries and Dataset

Calculation Libraries: Important python libraries, such as numpy and pandas, are imported to perform the multi-dimensional matrix calculations.

Visualization Libraries: Imported the metaplotlib and seaborn libraries for creating attractive and informative statistical graphs. Once we are done with importing libraries.

Naming Columns and Importing Dataset:

As the original dataset does not contain the names of the attributes so we have to name each attribute ourselves. Therefore, we appropriately named all the 26 columns.

Once we are done with naming of columns we import the *Auto Mobile Dataset* using pd.read_csv() command.

Data Cleansing:

- **Removing NaN or NULL Values (Missing Values)**

  First, we will be coercing all the missing entries to Nan. We will count the number of total missing values in each attribute. Once we find the missing values, we will be setting those equivalents to the mean of the attribute's column to which it belongs.

- **Checking and Correcting the Spelling Errors**

  Finding if there are any miss spelling of company names of the manufacturers of cars.

Data Exploration:

In this part we will be trying to understand and explore the dataset. For this purpose, we will be visualizing the data attributes and will try to find the correlation among the different available features of the given dataset. However, specifically, we are interested in finding the attributes which have significant effect on the prices of automobiles.

For this purpose, first we will explore the datatypes and basic information (count, mean, standard deviation, min, max, etc.) of different attributes of the dataset. From variables of different features of dataset, we can say that, both the numerical and categorical data is present in the dataset.

Describing Numerical Data:

```
car_data.describe()
```

	symboling	losses	wheel_base	length	width	height	weight	engine_size	bore	stroke	compression_ratio	horsepower	peak_rpm	city_mpg	hig
count	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	205.00	
mean	0.83	122.00	98.76	174.05	65.91	53.72	2555.57	126.91	3.33	3.26	10.14	104.26	5125.37	25.22	
std	1.25	31.68	6.02	12.34	2.15	2.44	520.68	41.64	0.27	0.31	3.97	39.52	476.98	6.54	
min	-2.00	65.00	86.60	141.10	60.30	47.80	1488.00	61.00	2.54	2.07	7.00	48.00	4150.00	13.00	
25%	0.00	101.00	94.50	166.30	64.10	52.00	2145.00	97.00	3.15	3.11	8.60	70.00	4800.00	19.00	
50%	1.00	122.00	97.00	173.20	65.50	54.10	2414.00	120.00	3.31	3.29	9.00	95.00	5200.00	24.00	
75%	2.00	137.00	102.40	183.10	66.90	55.50	2935.00	141.00	3.58	3.41	9.40	116.00	5500.00	30.00	
max	3.00	256.00	120.90	208.10	72.30	59.80	4066.00	326.00	3.94	4.17	23.00	288.00	6600.00	49.00	

Figure 1: Numerical Data Description

Describing Categorical Data:

```
car_data.describe(include=[object])
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_location	engine_type	num_cylinders	fuel_system
count	205	205	205	205	205	205	205	205	205	205
unique	22	2	2	3	5	3	2	7	7	8
top	toyota	gas	std	four	sedan	fwd	front	ohc	four	mpfi
freq	32	185	168	114	96	120	202	148	159	94

Figure 2: Categorical Data Description

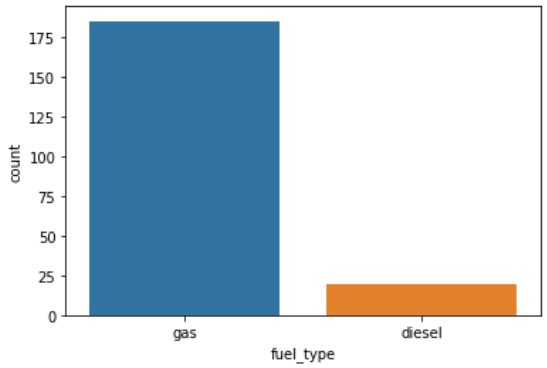

Figure 3: Number of Automobiles based on fuel type

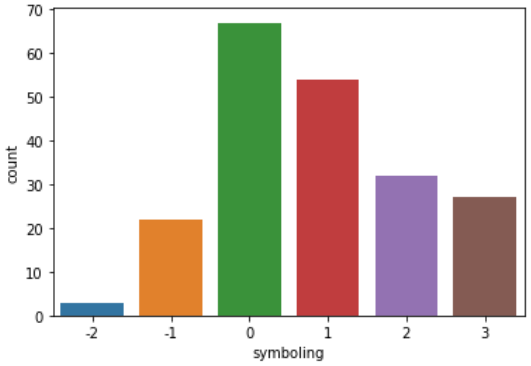

Figure 4: Automobiles Divided on Symbols basis

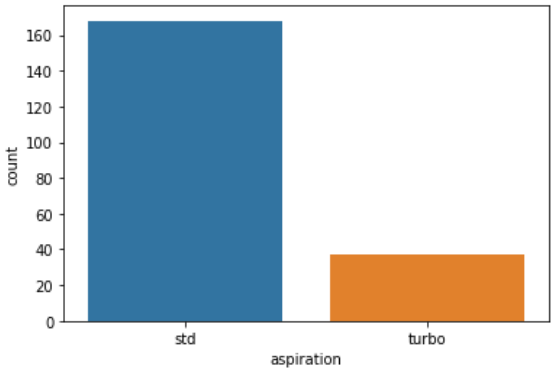

Figure 5: Car Counts Based on Aspiration Type

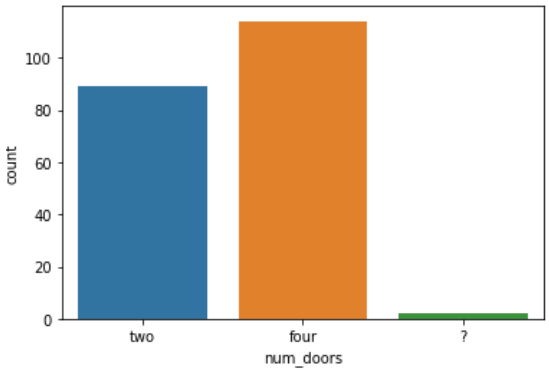

Figure 6: Car Count based on the no. of doors

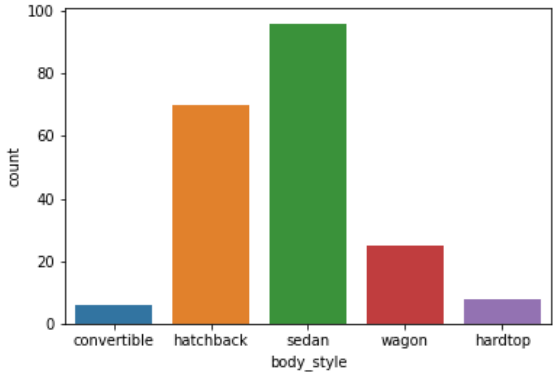

Figure7: Maximum numbers of cars have Sedan body style

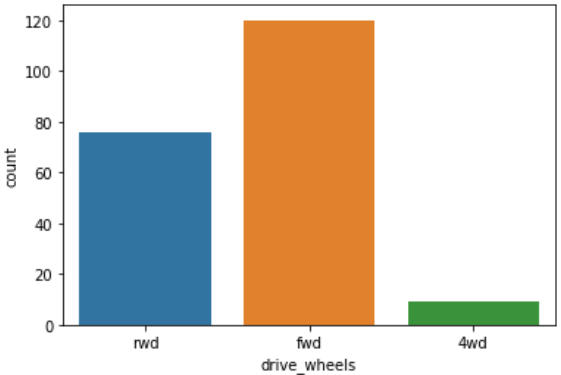

Figure 8: The cars having fwd drive wheels are higher than rwd and 4wd

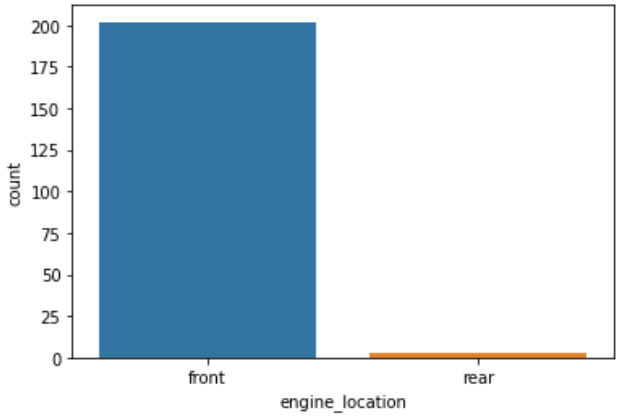

Figure 9: about 200 cars have engine located at front

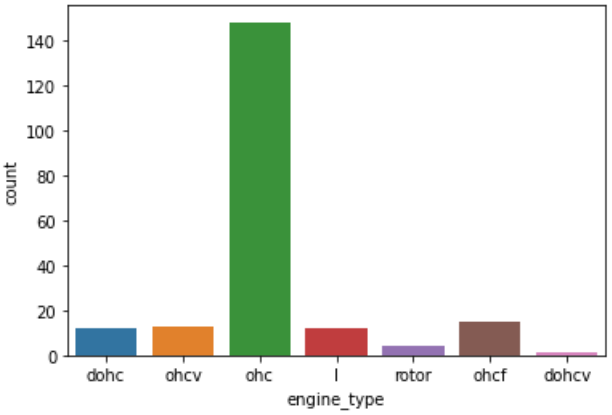

Figure 10: Maximum number of cars have engine type of ohc

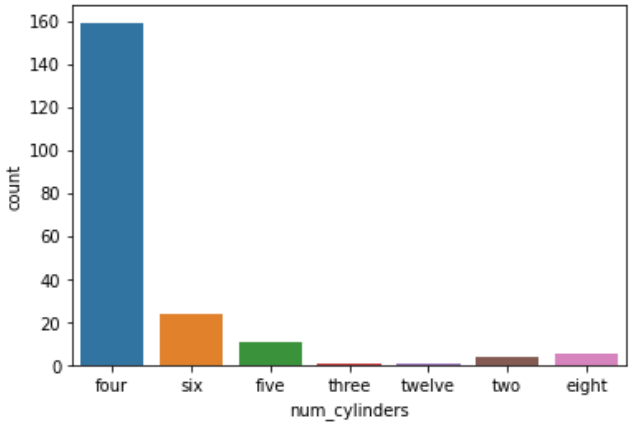

Figure 11: Most of the cars have four no. of cylinders

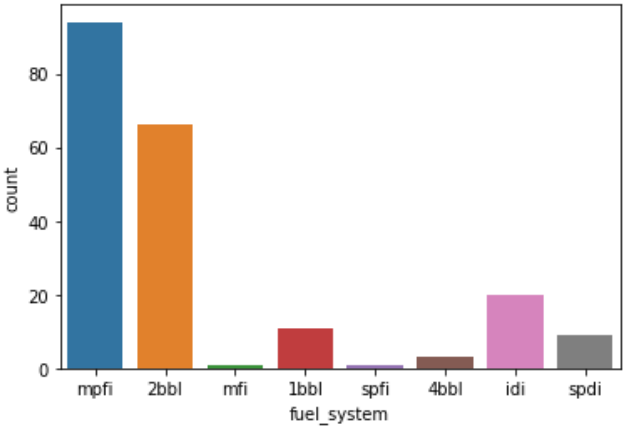

Figure 12: mpfi is the most common fuel system

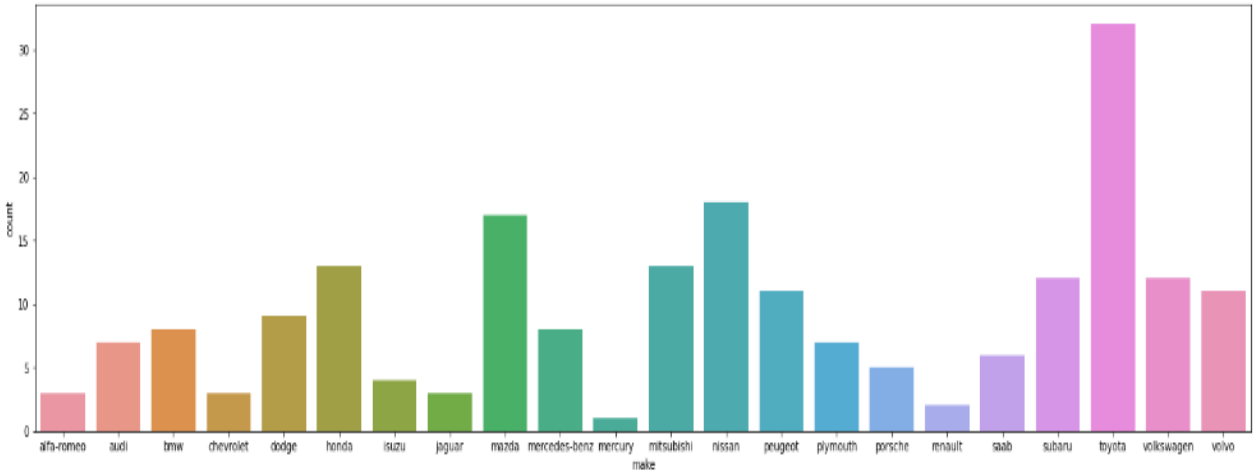

Figure 13: Toyota is the manufacturer of most of automobiles in dataset

Looking into the Spread of car prices:

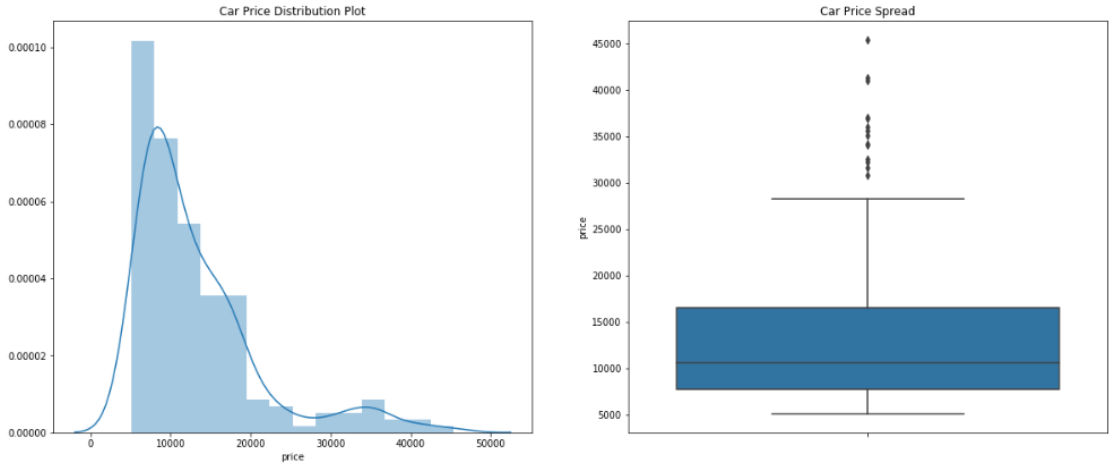

Figure14: Spread of Car Prices

It can bee seen that the maximum cars prices range between **8000** to **17000**. Its starts from around 200 and spreads up to 45000.

Determining the Correlations among the price and other features of the dataset using the Scatter Plot:

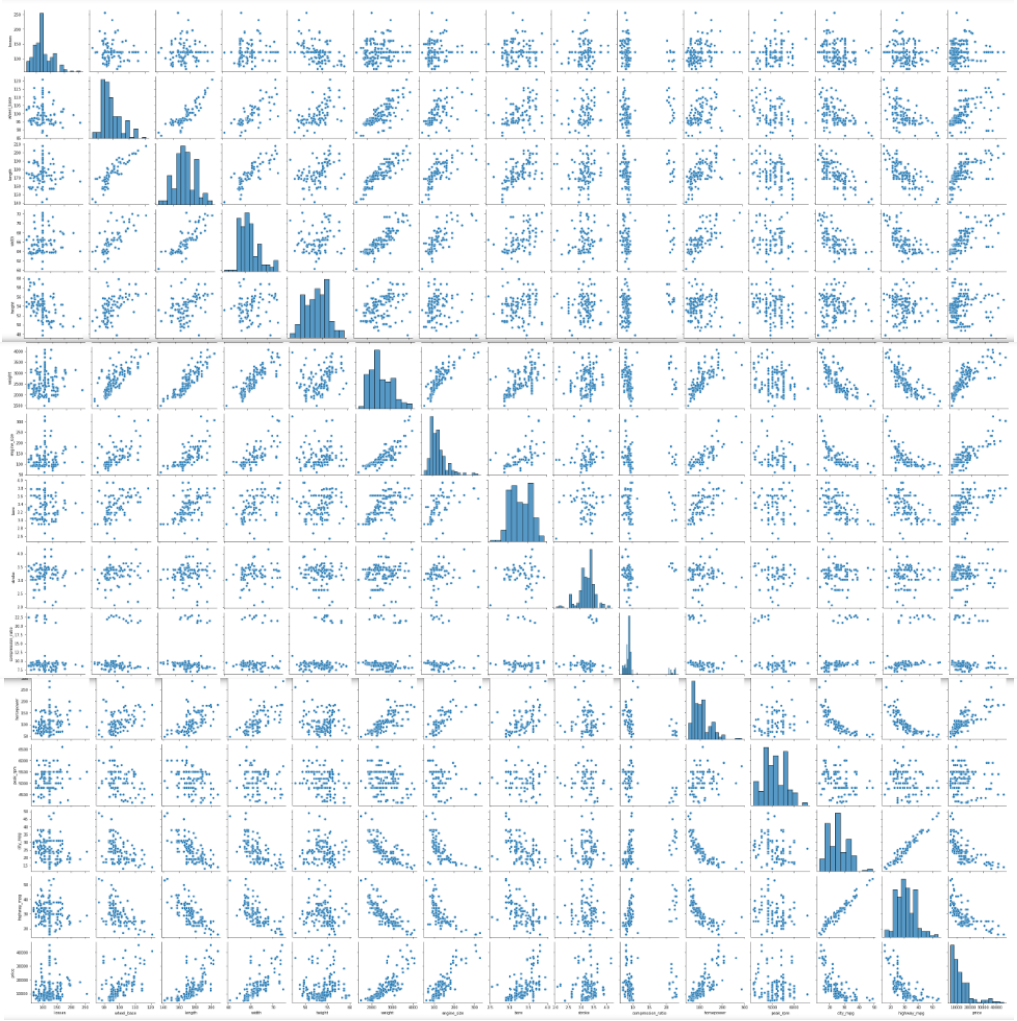

Figure 15: Pairwise Scatter Plot

It is hard to read from the scatter plot so we will plot the correlation matrix or heatmap.

Plotting Heatmap

Plotting heatmaps to study the correlation of features in the dataset.

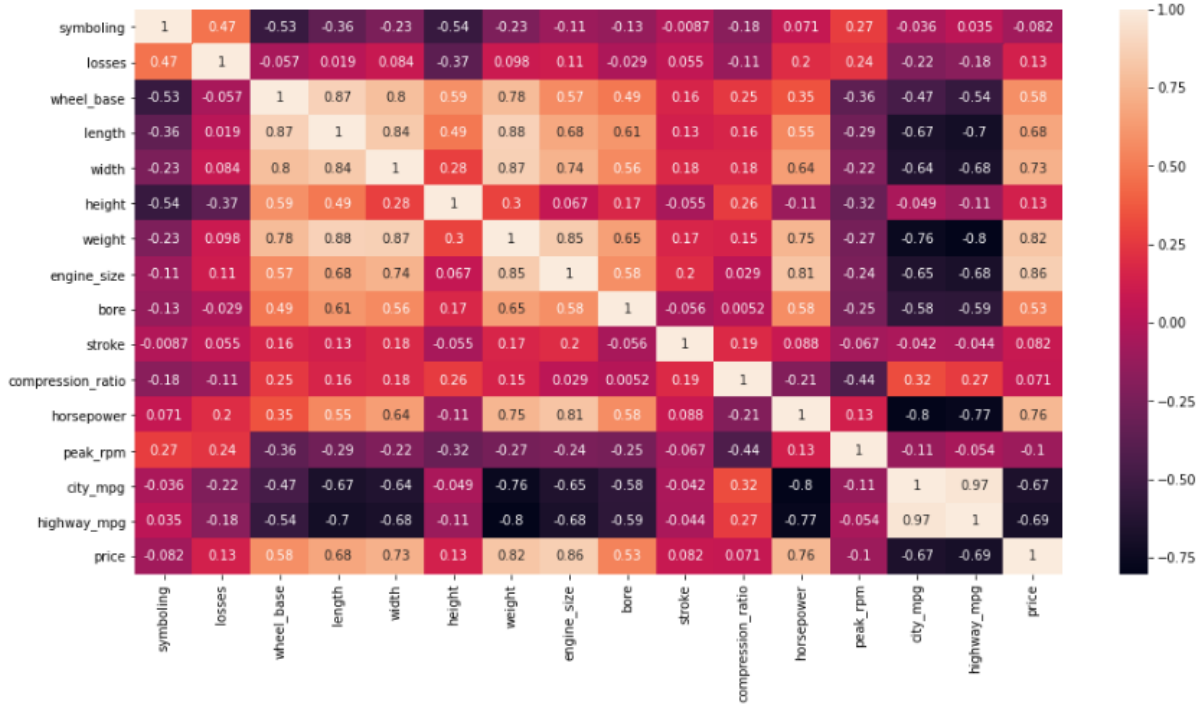

Figure 16: Heat Plot for Examining Correlation

Insights from Heatmap

The heatmap shows some useful insights.

Correlation of price with independent variables:

- Price is highly (positively) correlated with wheelbase, length, width, weight, engine_size, horsepower (notice how all these variables represent the size/weight/engine power of the car)
- Price is negatively correlated to city_mpg and highway_mpg (-0.70 approximately). This suggest that cars having high mileage may fall in the 'economy' cars category, and are priced lower (think Maruti Alto/Swift type of cars, which are designed to be affordable by the middle class, who value mileage more than horsepower/size of car etc.)

Correlation among independent variables:

- Many independent variables are highly correlated (look at the top-left part of matrix): wheelbase, length, weight, engine_size etc. are all measures of 'size/weight', and are positively correlated
- Thus, while building the model, we'll have to pay attention to multicollinearity (especially linear models, such as linear and logistic regression, suffer more from multicollinearity).

Data Preparation for Modelling

Let's now prepare the data and build the model. We will be converting all the data into numerical data so that we can apply the model.

- **One-Hot Encoding:**

  By the process of *One-Hot Encoding*, we will be converting the categorical data present in the dataset into the numerical data. So that we can provide the data in one format to the machine learning algorithm for better prediction performance.

- **Splitting Dependent and Independent Columns**

  We will be splitting the dependent and independent attributes. The independent attributes are those which helps us in predicting the values of dependent attributes. In this case the *price* is dependent upon the other *25 independent attributes.*

- **Train Test Split**

  There are multiple ways for the validation of the model. One of them is to *"train/test split"*. In this method the dataset is split into the training and testing sets respectively. The train set is used for the purpose of training the model while the test set is used to test the performance of model. This provides much precise evacuation on accuracy of the model as the test dataset have not been used in training the model. This approach of validation is more realistic for the real-world problems.

  In our case we have divided the dataset into test/train sets. The **70%** of data from the dataset will be used for training of the model while the remaining **30%** will be used for testing it.

Modelling

<u>Model 1:</u> Multiple Linear Regression

There are numerous factors that impact or influence the prices of a car in the market and so can be used to predict the prices of auto vehicles. When there is more than one independent variable present in the dataset, then the process we use for modeling is called multiple linear regression. For example, predicting auto vehicle prices using wheel_base, length, width, height, weight, engine_size, bore, stroke ,compression_ratio, horsepower, peak_rpm, city_mpg, highway_mpg, price of cars.

Multiple linear regression is simply the extension of simple linear regression. It is used when we have multiple independent variable linearly related.

```
#applying model on data
lreg = LinearRegression()
lreg.fit(X_train, y_train)
```

```
LinearRegression()
```

```
#evaluating model
train_pred = lreg.predict(X_train)
test_pred= lreg.predict(X_test)

r2_train_lr=r2_score(y_train,train_pred)
r2_test_lr=r2_score(y_test,test_pred)
mse_train_lr=sqrt(mean_squared_error(y_train,train_pred))
mse_test_lr=sqrt(mean_squared_error(y_test,test_pred))

print("R2 Training Score: ", r2_train_lr)
print("R2 Testing Score: ", r2_test_lr)
print("RMSE Training Score: ", mse_train_lr)
print("RMSE Testing Score: ", mse_test_lr)
```

Figure17: Code for Multi-Regression

Score of Model multiple Linear Regression

```
R2 Training Score:  0.9596129471136563
R2 Testing Score:  0.8865439575809779
MSE Training Score:  1532.544654852509
MSE Testing Score:  2810.034060656575
```

Figure 18: regr.fit()

Model 2: Polynomial Regression

Polynomial regression is a type of non-linear regression. There are certain independent attributes in dataset which aren't related linearly to the dependent attributes of auto-mobiles dataset. In order to model the non-linear relationship between the dependent and independent attributes we use the polynomial regression.

For this purpose, we are training model to predict the price of an automobile using the feature attributes which are not linearly related.

Degree = 2

```
poly_reg = PolynomialFeatures(degree=2)

X_tr_poly = poly_reg.fit_transform(X_train)
X_tst_poly = poly_reg.fit_transform(X_test)
poly = LinearRegression()
poly.fit(X_tr_poly, y_train)

train_pred = poly.predict(X_tr_poly)
test_pred = poly.predict(X_tst_poly)

#evaluating model
r2_train_pl2=r2_score(y_train,train_pred)
r2_test_pl2=r2_score(y_test,test_pred)
mse_train_pl2=sqrt(mean_squared_error(y_train,train_pred))
mse_test_pl2=sqrt(mean_squared_error(y_test,test_pred))

print("R2 Training Score: ", r2_train_pl2)
print("R2 Testing Score: ", r2_test_pl2)
print("RMSE Training Score: ", mse_train_pl2)
print("RMSE Testing Score: ", mse_test_pl2)
```

Figure19: Code for Polynomial Regression

Score of Polynomial Regression:

```
R2 Training Score:  0.9992242450350964
R2 Testing Score:  -975.7059512890166
MSE Training Score:  212.39991473558234
MSE Testing Score:  260723.20481374155
```

Figure20: Polynomial Regression Score

We further want to tune this model for better prediction testing scores. IN order to tune this model, we have to step towards the polynomial model of degree 3.

Polynomial Linear Regression Degree = 3

```
poly_reg = PolynomialFeatures(degree=3)

X_tr_poly = poly_reg.fit_transform(X_train)
X_tst_poly = poly_reg.fit_transform(X_test)
poly = LinearRegression()
poly.fit(X_tr_poly, y_train)

train_pred = poly.predict(X_tr_poly)
test_pred = poly.predict(X_tst_poly)

#evaluating models
r2_train_pl3=r2_score(y_train,train_pred)
r2_test_pl3=r2_score(y_test,test_pred)
mse_train_pl3=sqrt(mean_squared_error(y_train,train_pred))
mse_test_pl3=sqrt(mean_squared_error(y_test,test_pred))

print("R2 Training Score: ", r2_train_pl3)
print("R2 Testing Score: ", r2_test_pl3)
print("RMSE Training Score: ", mse_train_pl3)
print("RMSE Testing Score: ", mse_test_pl3)
```

Figure 21: Code for Polynomial Regression Degree = 3

Score of Model:

```
R2 Training Score:  0.999224245035145
R2 Testing Score:  -393.3159006047544
MSE Training Score:  212.39991472893462
MSE Testing Score:  165660.86089039655
```

Figure 22: Score of Polynomial Regression Degree 3

Model 3: Decision Tree Regressor

The decision trees machine learning algorithm is used when we need to fit a sine curve along with the addition noisy observations. As a result, it leads to learn local linear regressions approximating the sine curve.

By setting the maximum depth of the tree (controlled by the parameter max_depth) too high, the decision tree learns all the fine details of the given training dataset, also it learns from the noise and results into overfitting. However, we will set the depth of decision tree that won't let our model to overfit.

```
#applying model on data
dt_regressor = DecisionTreeRegressor(max_depth=5, random_state=0)
dt_regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(max_depth=5, random_state=0)
```

```
#evaluating model
train_pred = dt_regressor.predict(X_train)
test_pred= dt_regressor.predict(X_test)

r2_train_dt=r2_score(y_train,train_pred)
r2_test_dt=r2_score(y_test,test_pred)
mse_train_dt=sqrt(mean_squared_error(y_train,train_pred))
mse_test_dt=sqrt(mean_squared_error(y_test,test_pred))

print("R2 Training Score: ", r2_train_dt)
print("R2 Testing Score: ", r2_test_dt)
print("RMSE Training Score: ", mse_train_dt)
print("RMSE Testing Score: ", mse_test_dt)
```

Figure23: Code for Decision Tree Regressor

Score for Decision Tree Regressor

```
R2 Training Score:  0.9675638336408569
R2 Testing Score:  0.8899247988142187
MSE Training Score:  1373.4309052456986
MSE Testing Score:  2767.8497603686997
```

Figure 24: Decision Tree Regressor Score

Model 4: Random Forest

It is a meta estimator that is used to fit a number of classifying **decision** trees on various sub-samples present in the given dataset and then it uses the averaging to improve the predictive accuracy of model and also control the over-fitting of the overall model.

```
#applying model on data
rf_regressor = RandomForestRegressor(max_depth=5, random_state=0,n_estimator
rf_regressor.fit(X_train, y_train)
```

```
RandomForestRegressor(max_depth=5, random_state=0)
```

```
#evaluating model
train_pred = rf_regressor.predict(X_train)
test_pred= rf_regressor.predict(X_test)

r2_train_rf=r2_score(y_train,train_pred)
r2_test_rf=r2_score(y_test,test_pred)
mse_train_rf=sqrt(mean_squared_error(y_train,train_pred))
mse_test_rf=sqrt(mean_squared_error(y_test,test_pred))

print("R2 Training Score: ", r2_train_rf)
print("R2 Testing Score: ", r2_test_rf)
print("RMSE Training Score: ", mse_train_rf)
print("RMSE Testing Score: ", mse_test_rf)
```

Figure 25: Code for Random Forest Regressor

Model Score

```
R2 Training Score:  0.9575321498309461
R2 Testing Score:  0.9093571142953474
MSE Training Score:  1571.5282573868335
MSE Testing Score:  2511.6818685037447
```

Figure 26: Random Forest Regressor Score

Comparing and Choosing the Best Model

We have implemented five different machine learning models i.e., Linear Regression, Polynomial Regression (Degree = 2), Polynomial Regression (Degree = 3), Decision Tress Regressor, and Random Forest Regressor. Now, we must compare the results score of all the five machine learning models and must choose the best model out of all, the one with higher rate of accuracy.

	model-name	R2 Score (Training)	R2 Score (Testing)	RMSE (Training)	RMSE (Testing)
4	Random Forrest Regressor	0.96	0.91	1571.53	2511.68
3	Decision Tree Regressor	0.97	0.89	1373.43	2767.85
0	Linear Regression	0.96	0.89	1532.54	2810.03
2	Polynomial Regression (Degree=3)	1.00	-393.32	212.40	165660.86
1	Polynomial Regression (Degree=2)	1.00	-975.71	212.40	260723.20

Figure 26: Comparison Scores of all five models

The **best** model is *Random Forest Regressor* because it has the best testing R2-Score of 0.91.
