

Predicticting Car Price by Using Multiple Linear Regression



Thank You
Nilufer Metin
Data Science Tools-2
University of Denver

Predicticting Car Price by Using Multiple Linear Regression



Simple vs. Multiple Linear Regression

- Simple Linear Regression – one independent variable.

$$y = b_0 + b_1 x_1$$

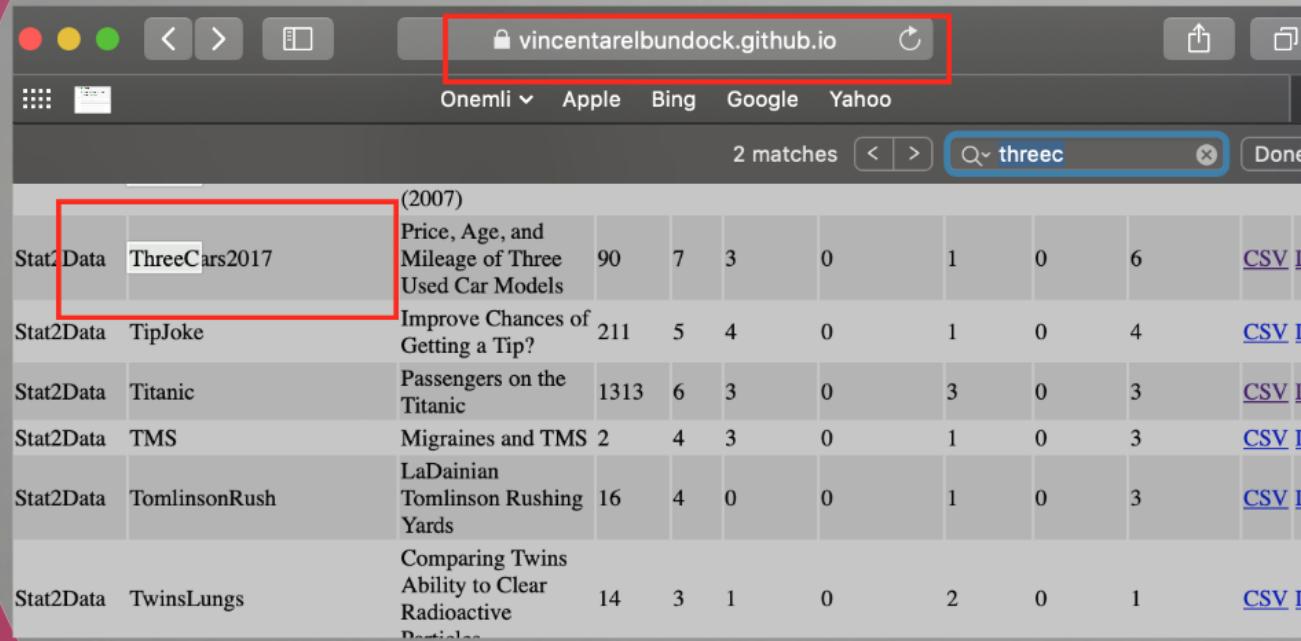
- Multiple Linear Regression – multiple independent variables.

$$y = b_0 + b_1 x_1 + b_2 x_2 \dots + b_n x_n$$

2nd independent
variable and
weight (coefficient)

nth independent
variable and
weight (coefficient)

1-FEATURE ENGINEERING



The screenshot shows a web browser window with a yellow header bar containing the title "1-FEATURE ENGINEERING". Below the header is a search bar with the URL "vincentarelbundock.github.io" and a search term "threec". The main content area displays a search results page with several entries:

Stat2Data	ThreeCars2017	(2007) Price, Age, and Mileage of Three Used Car Models	90	7	3	0	1	0	6	CSV	D
Stat2Data	TipJoke	Improve Chances of Getting a Tip?	211	5	4	0	1	0	4	CSV	D
Stat2Data	Titanic	Passengers on the Titanic	1313	6	3	0	3	0	3	CSV	D
Stat2Data	TMS	Migraines and TMS	2	4	3	0	1	0	3	CSV	D
Stat2Data	TomlinsonRush	LaDainian Tomlinson Rushing Yards	16	4	0	0	1	0	3	CSV	D
Stat2Data	TwinsLungs	Comparing Twins Ability to Clear Radioactive Particles	14	3	1	0	2	0	1	CSV	D



```
In [56]: import pandas as pd
import numpy as np
import missingno as msno
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.formula.api as sm
```

```
In [57]: #!pip install missingno
```

```
In [58]: data = pd.read_csv("2-ThreeCars2017.csv")
```

```
In [59]: data.head()
```

Out[59]:

	Unnamed: 0	CarType	Age	Price	Mileage	Mazda6	Accord	Maxima
0	1	Mazda6	3	15.9	17.8	1	0	0
1	2	Mazda6	2	16.4	19.0	1	0	0
2	3	Mazda6	1	18.9	20.9	1	0	0
3	4	Mazda6	2	16.9	24.0	1	0	0
4	5	Mazda6	2	20.5	24.0	1	0	0

STEP 2

```
In [60]: data=data.drop(data.columns[0],axis=1)
```

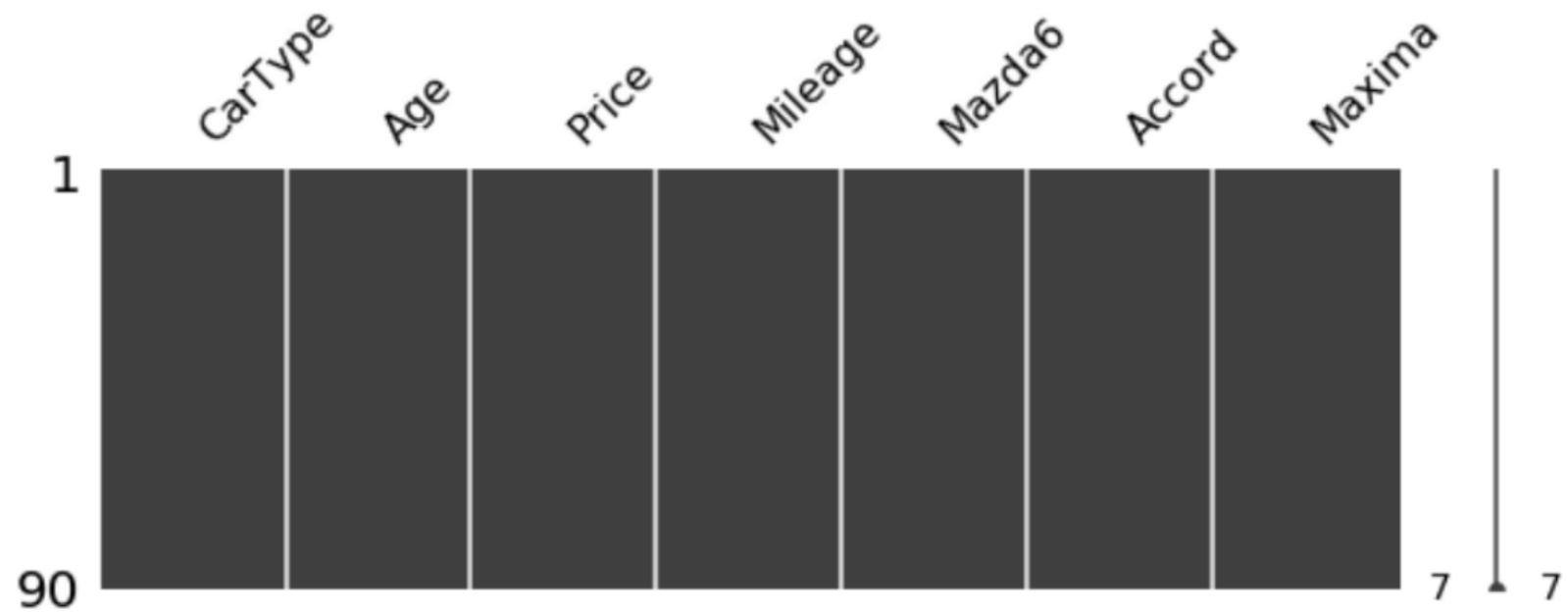
```
In [61]: data.head()
```

Out[61]:

	CarType	Age	Price	Mileage	Mazda6	Accord	Maxima	Camry
0	Mazda6	3	15.9	17.8	1	0	0	0
1	Mazda6	2	16.4	19.0	1	0	0	0
2	Mazda6	1	18.9	20.9	1	0	0	0
3	Mazda6	2	16.9	24.0	1	0	0	0
4	Mazda6	2	20.5	24.0	1	0	0	0

```
In [62]: msno.matrix(data,figsize=(10,3))
```

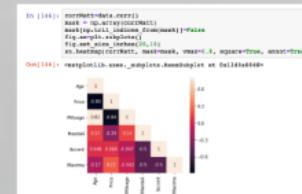
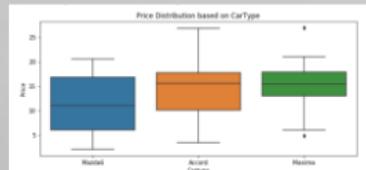
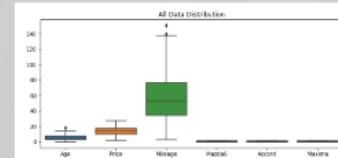
```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x122b3fc18>
```



STEP 3

```
# fig, axes = plt.subplots(nrows = 4, ncols= 1)
# fig.set_size_inches(10,20)
sns.boxplot(data=data, y='Price', orient="v", ax=axes[0])
sns.boxplot(data=data, orient="v", ax=axes[1])
sns.boxplot(data=data, y='Price', x="CarType", orient="v", ax=axes[2])
sns.boxplot(data=data, y='Price', x= "Age", orient="v", ax=axes[3])

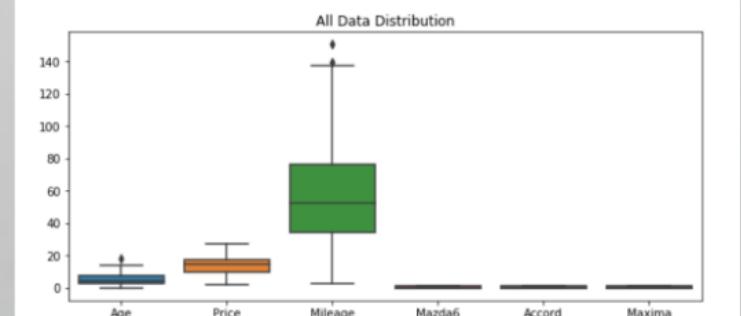
axes[0].set(ylabel='Price', title='Price Distribution')
axes[1].set(title='All Data Distribution')
axes[2].set(ylabel='Price', xlabel="CarType", title='Price Distribution based on CarType')
axes[3].set(ylabel='Price', xlabel = 'Age', title='Price Distribution based on the age of the Car')
```



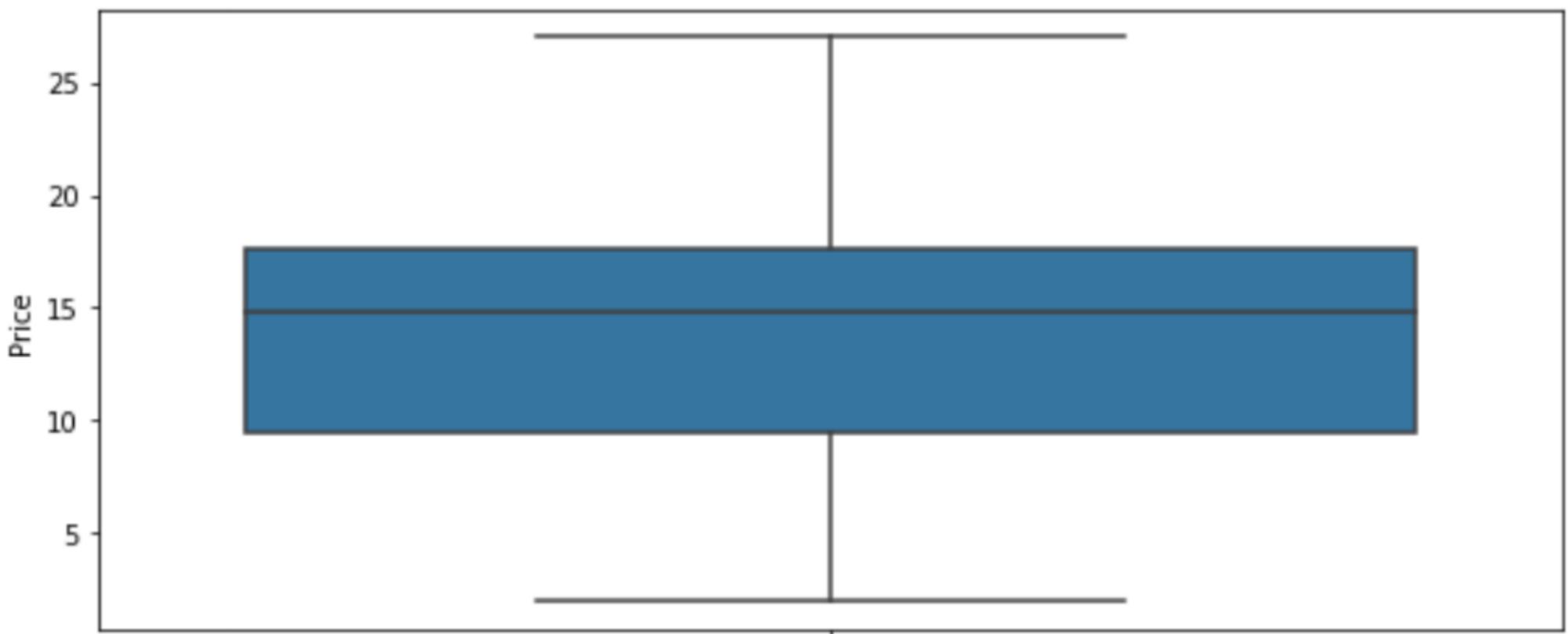
STEP 3

```
: fig, axes = plt.subplots(nrows =4, ncols= 1)
fig.set_size_inches(10,20)
sn.boxplot(data=data, y='Price', orient="v", ax=axes[0])
sn.boxplot(data=data, orient="v", ax=axes[1])
sn.boxplot(data=data, y='Price', x="CarType", orient="v", ax=axes[2])
sn.boxplot(data=data, y='Price', x= "Age", orient="v", ax=axes[3])

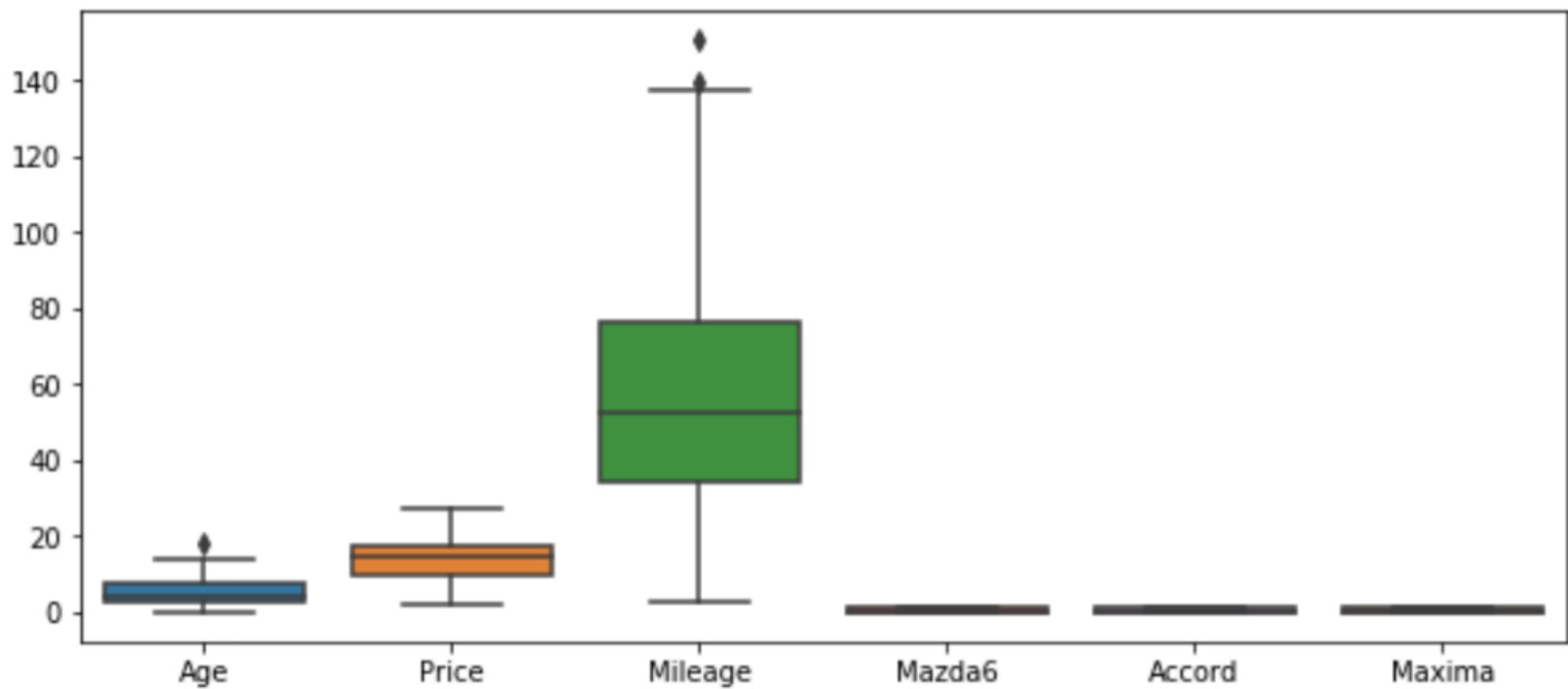
axes[0].set(ylabel='Price', title='Price Distribution')
axes[1].set(title='All Data Distribution')
axes[2].set(ylabel='Price', xlabel='Cartype', title='Price Distribution based on CarType')
axes[3].set(ylabel='Price', xlabel = 'Age', title='Price Distribution based on the age of the Car')
```



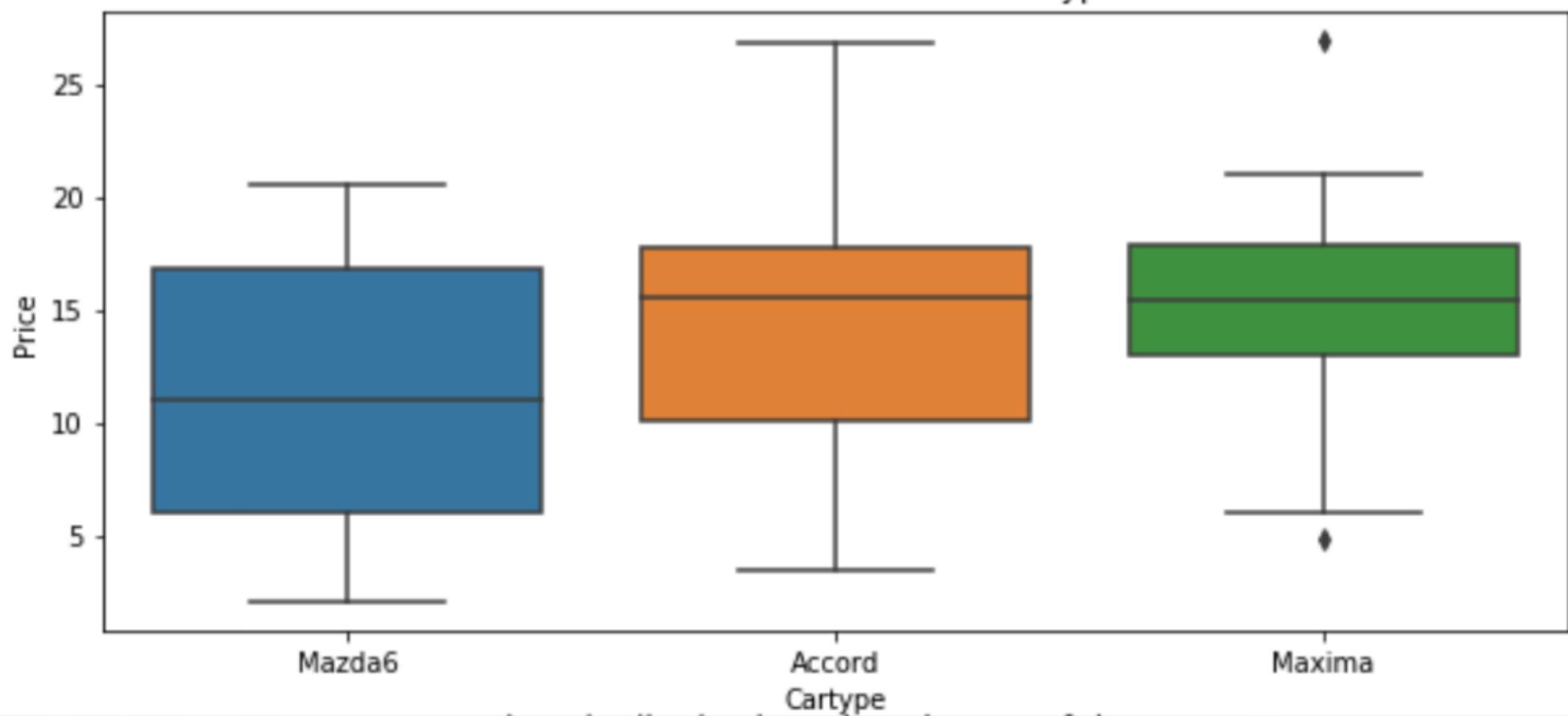
Price Distribution



All Data Distribution

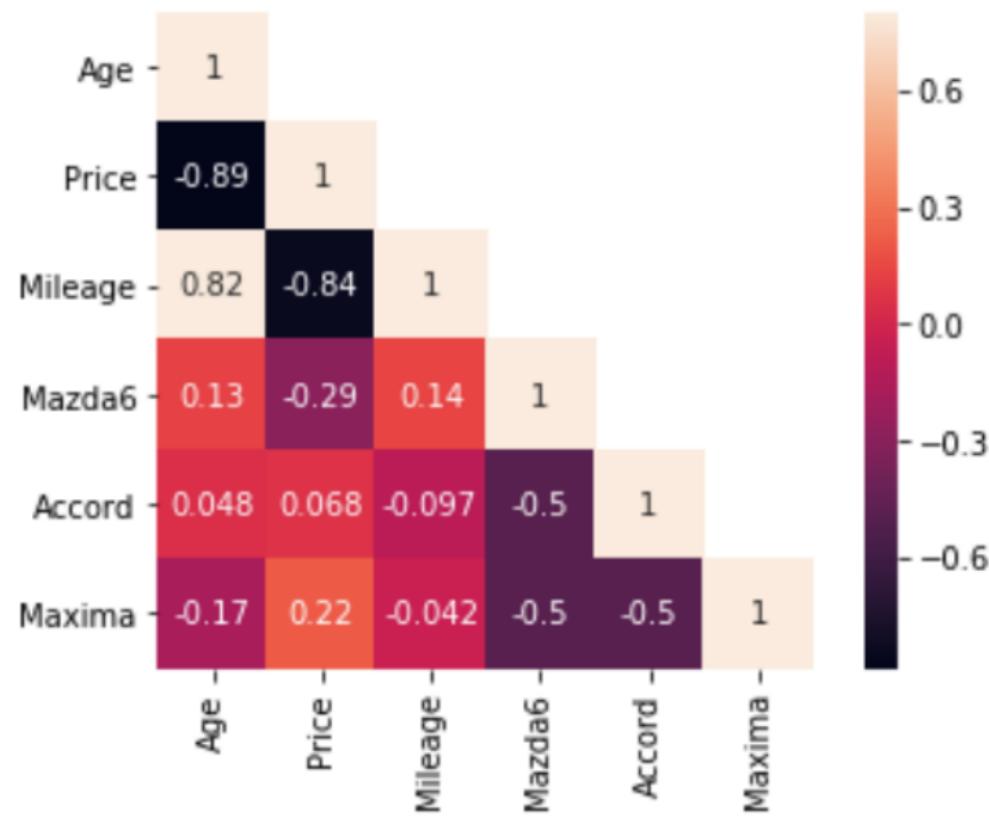


Price Distribution based on CarType



```
In [146]: corrMatt = data.corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig, ax = plt.subplots()
fig.set_size_inches(20, 10)
sn.heatmap(corrMatt, mask=mask, vmax=0.8, square=True, annot=True)
```

Out[146]: <matplotlib.axes._subplots.AxesSubplot at 0x12d0a8048>



APPLYING LINEAR REGRESSION

```
In [147]: data = data[['CarType', 'Age', 'Mileage', 'Model', 'Accord', 'Maxima', 'Price']]  
In [148]: data.head()  
Out[148]:
```

CarType	Age	Mileage	Model	Accord	Maxima	Price
B-Maxda	3	17.0	1	0	0	950
T-Honda	3	16.0	1	0	0	940
A-Mazda	3	19.0	1	0	0	940
B-Mazda	2	24.0	1	0	0	930
A-Mazda	2	24.0	1	0	0	930

```
In [150]: replace_map = {'CarType': { 'Maxima': 1, 'Accord': 2, 'Mazda': 3}}  
In [151]: data.replace(replace_map, inplace=True)  
In [152]: data.head()  
Out[152]:
```

CarType	Age	Mileage	Model	Accord	Maxima	Price
0	3	17.0	1	0	0	152
1	3	16.0	1	0	0	164
2	3	19.0	1	0	0	189
3	3	24.0	1	0	0	163
4	3	24.0	1	0	0	205

```
In [153]: #splitting the data into train and test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)  
In [154]: X_train.size  
Out[154]: 784  
In [155]: X_test.size  
Out[155]: 204
```

```
click to scroll output; double click to hide.  
scaler = StandardScaler()  
In [1]:  
In [166]: X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

```
In [149]: #Applying Multiple regression  
regressor = LinearRegression()  
In [150]: #Training the algorithm  
regressor.fit(X_train, y_train)  
Out[150]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```

```
In [147]: data = data[['CarType', 'Age', 'Mileage', 'Mazda6', 'Accord', 'Maxima', 'Price']]
```

```
In [148]: data.head()
```

Out[148]:

	CarType	Age	Mileage	Mazda6	Accord	Maxima	Price
0	Mazda6	3	17.8	1	0	0	15.9
1	Mazda6	2	19.0	1	0	0	16.4
2	Mazda6	1	20.9	1	0	0	18.9
3	Mazda6	2	24.0	1	0	0	16.9
4	Mazda6	2	24.0	1	0	0	20.5

```
In [150]: replace_map = {'CarType': {'Maxima': 1, 'Accord': 2, 'Mazda6': 3}}
```

```
In [151]: data.replace(replace_map, inplace=True)
```

```
In [152]: data.head()
```

Out[152]:

	CarType	Age	Mileage	Mazda6	Accord	Maxima	Price
0	3	3	17.8	1	0	0	15.9
1	3	2	19.0	1	0	0	16.4
2	3	1	20.9	1	0	0	18.9
3	3	2	24.0	1	0	0	16.9
4	3	2	24.0	1	0	0	20.5

2	3	1	20.9	1	0	0	18.9
3	3	2	24.0	1	0	0	16.9
4	3	2	24.0	1	0	0	20.5

```
In [162]: #splitting the data into train and test  
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size = 0.2, random_state=0)
```

```
In [163]: X_train.size
```

```
Out[163]: 504
```

```
In [164]: X_test.size
```

```
Out[164]: 126
```



#165) #lets see like data

Out[164]: 126

click to scroll output; double click to hide

```
scaler = StandardScaler()
```

In []:

```
In [166]: X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```



Prezi

```
In [169]: #Applying Multiple regression
```

```
regressor = LinearRegression()
```

```
In [170]: #Training the algorithm
```

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

```
Out[170]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,  
normalize=False)
```

RESULTS OF PREDICTION MODEL

```
In [173]: predictions = regressor.predict(X_test)
```

```
predictions
```

```
Out[173]: array([18.09828615, 14.88035618, 19.77602685, 20.14803392, 16.9452516 ,  
12.39109463, 19.25628456, 19.06469556, 16.5956154 , 7.65135739,  
15.09581643, 2.82007337, 13.66944052, 17.07411711, 15.75148706,  
10.89232253, 3.82006976, 17.5079902 ])
```

```
In [174]: y_test
```

```
Out[174]: 2    18.9  
13   16.0  
40   21.0  
41   19.5  
66   21.0  
30   12.0  
45   17.5  
43   17.4  
78   16.9  
89   4.8  
7    18.0  
26   8.2  
33   12.9  
63   15.7  
8    13.6  
52   8.0  
24   2.0  
56   16.0  
Name: Price, dtype: float64
```

```
In [176]: df = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
```

```
df
```

	Actual	Predicted
2	18.9	18.09828615
13	16.0	14.88035618
40	21.0	19.77602685
41	19.5	20.14803392
66	21.0	16.9452516
30	12.0	12.39109463
45	17.5	19.25628456
43	17.4	19.06469556
78	16.9	16.5956154
89	4.8	7.65135739
7	18.0	15.09581643
26	8.2	2.82007337
33	12.9	13.66944052
63	15.7	17.07411711
8	13.6	15.75148706
52	8.0	10.89232253
24	2.0	3.82006976
56	16.0	17.5079902

```
In [177]: r2_score(y_test, predictions)
```

```
Out[177]: 0.8749853622053401
```

```
X_opt = X[:, (0,1,2,3,4,5,6)]
```

```
regressor = LinearRegression()
```

```
regressor.fit(X_opt,y)
```

```
regressor.score(X_opt,y)
```

```
OLS Regression Results
```

Dep. Variable	Price	Residuals	D.F.
Model	OLS	Adj. R-squared:	0.87
Method	Least Squares	F-statistic:	122.0
Date	Mon 15 Jul 2019	P-value (F-statistic):	2.2e-31
Time	09:08:06	Log-Likelihood:	-181.86
No. Observations	92	AIC:	415.1
DF Residuals	83	BIC:	419.0
DF Model	8		
Converged:	True		

```
coef intercept 1 18.9 16.09828615  
x2 -0.019 0.019 0.020 0.002 4.75 0.007  
x3 0.072 0.072 0.072 0.003 13.6 0.007  
x4 -0.038 0.038 0.038 0.003 -13.5 -0.008  
x5 0.016 0.016 0.016 0.002 3.82 0.007  
x6 0.014 0.014 0.014 0.002 17.5 0.006  
x7 0.013 0.013 0.013 0.002 10.89 0.005
```

```
Optimal: OLS Df=8 Model: 92 D.F.
```

```
Prob(Optimal): 0.00 Adj-R-squared: 0.871
```

```
Score: 415.1 Pseudo R-squared: 0.875
```

```
In [173]: predictions = regressor.predict(X_test)  
predictions
```

```
Out[173]: array([18.08828616, 14.88035618, 18.77602685, 20.14803392, 16.9452516 ,  
12.39109463, 19.25628456, 19.0646856 , 16.5956154 , 7.65135739,  
15.89581643, 2.82007337, 13.66944052, 17.07411711, 15.75148706,  
10.89232251, 3.82006976, 17.5079902 ])
```

```
In [174]: y_test
```

```
Out[174]: 2      18.9  
13     16.0  
53     21.0  
41     19.5  
66     21.0  
30     12.0  
45     17.5  
43     17.4  
78     16.9  
89      4.8  
7      18.0  
26      5.2  
33     12.5  
63     15.7  
8      13.6  
16      8.0  
24      2.0  
56     16.0  
Name: Price, dtype: float64
```

```
In [176]: df = pd.DataFrame({'Actual': y_test , 'Predicted': predictions})  
df
```

Out[176]:

	Actual	Predicted
2	18.9	18.088286
13	16.0	14.880356
53	21.0	18.776027
41	19.5	20.148034
66	21.0	16.945252
30	12.0	12.391095
45	17.5	19.256285
43	17.4	19.064686
78	16.9	16.595615
89	4.8	7.651357
7	18.0	15.895816
26	5.2	2.820073
33	12.5	13.669441
63	15.7	17.074117
8	13.6	15.751487
16	8.0	10.892323
24	2.0	3.820070
56	16.0	17.507990

```
In [177]: r2_score(y_test, predictions)
```

```
Out[177]: 0.8749853622053401
```

```
X_opt = X[:, [0,1,2,3,4,5,6]]
```

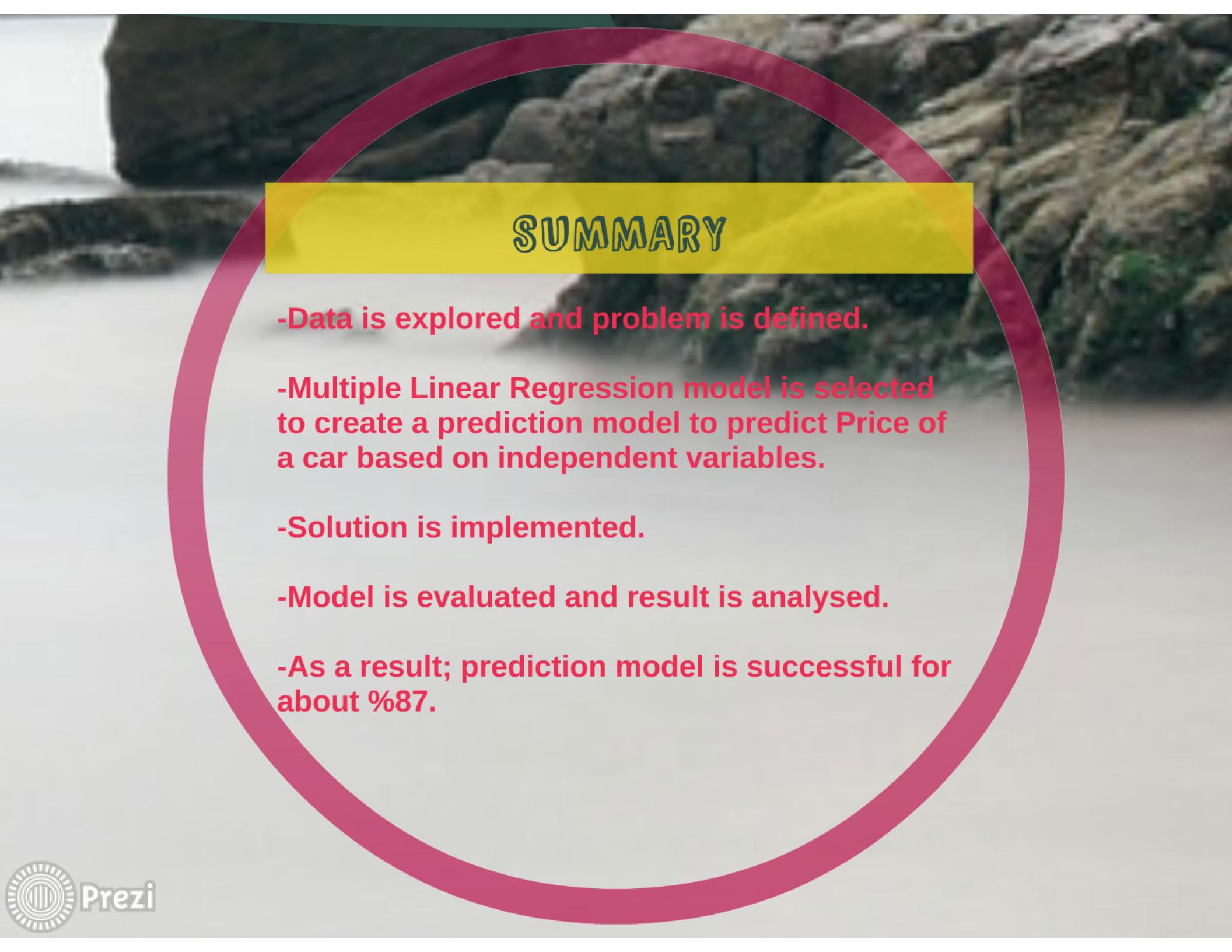
```
regressor_OLS=sm.OLS(endog =y, exog=X_opt).fit()  
regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.852
Model:	OLS	Adj. R-squared:	0.845
Method:	Least Squares	F-statistic:	122.0
Date:	Wed, 05 Jun 2019	Prob (F-statistic):	2.22e-34
Time:	01:35:03	Log-Likelihood:	-195.56
No. Observations:	90	AIC:	401.1
Df Residuals:	85	BIC:	413.6
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	8.5297	0.210	40.624	0.000	8.112	8.947
x1	4.9812	0.154	32.363	0.000	4.675	5.287
x2	-0.8172	0.108	-7.567	0.000	-1.032	-0.602
x3	-0.0520	0.012	-4.490	0.000	-0.075	-0.029
x4	-3.4399	0.219	-15.708	0.000	-3.875	-3.005
x5	3.3314	0.339	9.840	0.000	2.658	4.004
x6	8.6382	0.335	25.796	0.000	7.972	9.304

Omnibus:	6.631	Durbin-Watson:	1.472
Prob(Omnibus):	0.036	Jarque-Bera (JB):	6.011
Skew:	0.591	Prob(JB):	0.0495



SUMMARY

- Data is explored and problem is defined.
- Multiple Linear Regression model is selected to create a prediction model to predict Price of a car based on independent variables.
- Solution is implemented.
- Model is evaluated and result is analysed.
- As a result; prediction model is successful for about %87.

Predicticting Car Price by Using Multiple Linear Regression



Thank You
Nilufer Metin
Data Science Tools-2
University of Denver