



AI & ML Internship Project Report

PROBLEM STATEMENT 3:

Collect the data of used cars and bikes and try to predict the price using the user input information of car or bike data.

Project submitted by:

-Balaji Ramesh

-Mohammad Hashim Bhat

BUILDING A LINEAR REGRESSION MODEL TO PREDICT THE PRICE OF USED CARS AND BIKES

Introduction:

Predicting the price of used cars is both an important and interesting problem. According to data obtained from the National Transport Authority, the number of cars registered between 2003 and 2013 has witnessed a spectacular increase of 234%. From 68,524 cars registered in 2003, this number has now reached 160,701. With difficult economic conditions, it is likely that sales of second-hand imported (reconditioned) cars and used cars will increase. It is reported that the sales of new cars has registered a decrease of 8% in 2013. In many developed countries, it is common to lease a car rather than buying it outright. A lease is a binding contract between a buyer and a seller (or a third party – usually a bank, insurance firm or other financial institutions) in which the buyer must pay fixed instalments for a pre-defined number of months/years to the seller/financer. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e. its expected resale value. If the residual value is under-estimated by the seller/financer at the beginning, the instalments will be higher for the clients who will certainly then opt for another seller/financer. If the residual value is over-estimated, the instalments will be lower for the clients but then the seller/financer may have much difficulty at selling these high-priced used cars at this over-estimated residual value. Thus, we can see that estimating the price of used cars is of very high commercial importance as well. Manufacturers' from Germany made a loss of 1 billion Euros in their USA market because of mis-calculating the residual value of leased cars. Most individuals in Mauritius who buy new cars are also very apprehensive about the resale value of their cars after certain number of years when they will possibly sell it in the used cars market. Predicting the resale value of a car is not a simple task. It is trite knowledge that the value of used cars depends on a number of factors. The most important ones are usually the age of the car, its make (and model), the origin of the car (the original country of the manufacturer), its mileage (the number of kilometers it has run) and its horsepower. Due to rising fuel prices, fuel economy is also of prime importance. Unfortunately, in practice, most people do not know exactly how much fuel their car consumes for each km driven.

The dataset that I am using in this project is written by my own, containing for a full description of the like vehicle_type, name, year, odometer, fuel, owner, selling price.

The prices of new cars and other vehicles in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase. There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offer this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used car's actual market value. It is important to know their actual market value while both buying and selling.

This project focuses on the exploratory data analysis phase of the dataset. In particular, I will try to detect associations between variables, especially against price. The end-goal of such a project would be to build a price-prediction model for vehicles sold.

Reading And Understanding Data:

The dataset is well structured but there are some free text fields and many missing values. The “name” column is problematic: It is free text which causes all sorts of issues, and although an engineer could perhaps find interesting information in it, I chose to simply drop it.

First we import all the required packages.

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

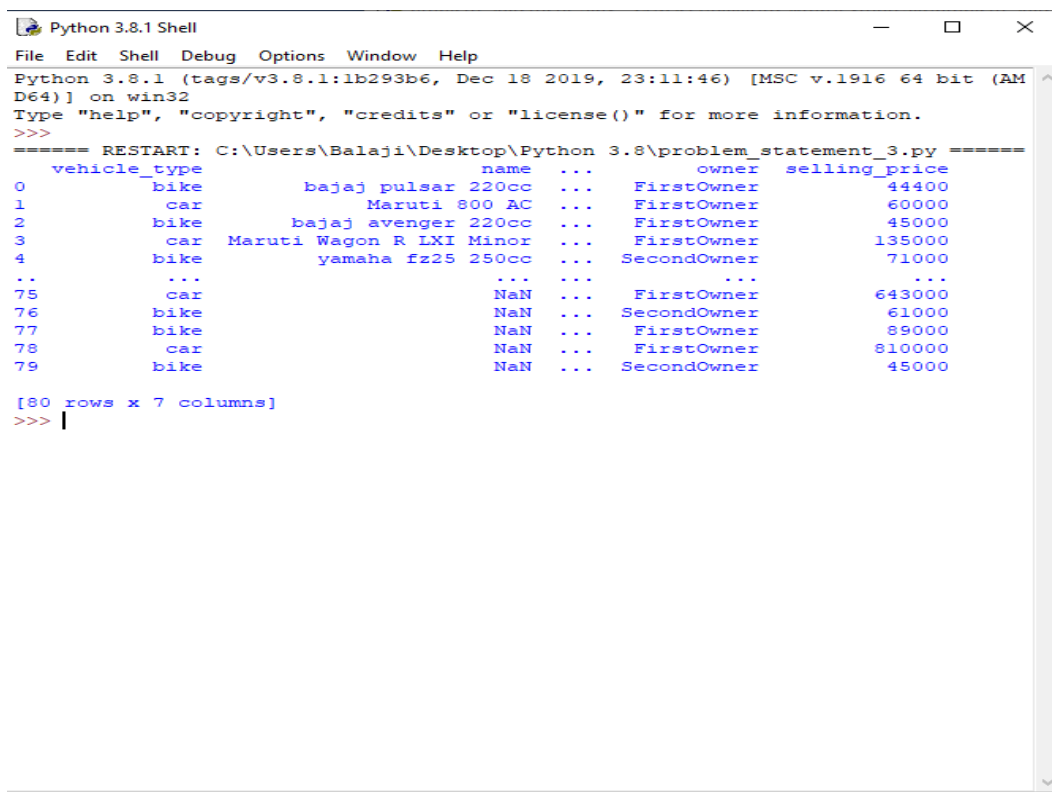
```
import seaborn as sns
```

Read the dataset

```
dataset=pd.read_csv("used cars and bikes.csv")
```

Print the dataset

```
Print(dataset)
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
   vehicle_type  name  ...  owner  selling_price
0      bike      bajaj pulsar 220cc  ...  FirstOwner      44400
1      car      Maruti 800 AC  ...  FirstOwner      60000
2      bike      bajaj avenger 220cc  ...  FirstOwner      45000
3      car      Maruti Wagon R LXI Minor  ...  FirstOwner      135000
4      bike      yamaha fz25 250cc  ...  SecondOwner      71000
..          ...          ...  ...  ...
75     car          NaN  ...  FirstOwner      643000
76     bike          NaN  ...  SecondOwner      61000
77     bike          NaN  ...  FirstOwner      89000
78     car          NaN  ...  FirstOwner      810000
79     bike          NaN  ...  SecondOwner      45000

[80 rows x 7 columns]
>>> |
```

Data Inspection

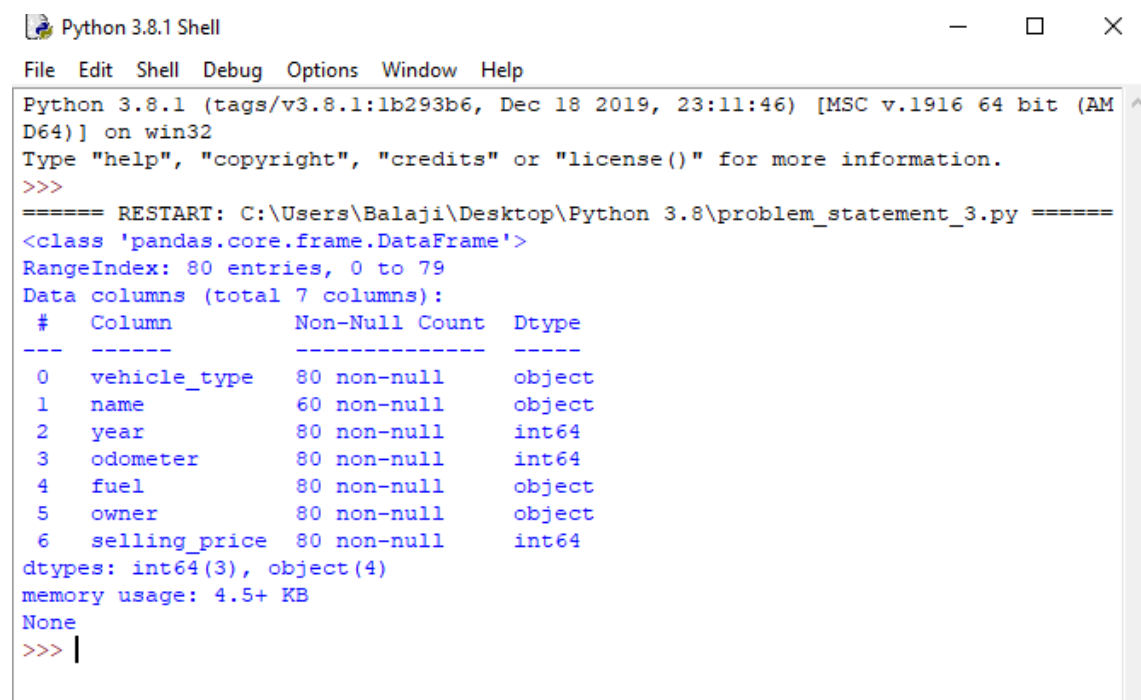
Print(dataset.shape())

```
# Data has 7 columns and 80 rows.  
(80, 7)
```

Dataset.describe()

	year	odometer	selling_price
count	80.000000	80.000000	8.000000e+01
mean	2014.750000	37887.212500	3.050800e+05
std	2.910044	31553.804555	3.875567e+05
min	2007.000000	5000.000000	2.100000e+04
25%	2014.000000	17999.750000	6.000000e+04
50%	2015.000000	25500.000000	1.240500e+05
75%	2017.000000	44125.000000	4.600000e+05
max	2019.000000	141000.000000	1.945999e+06

print(dataset.info())



```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 80 entries, 0 to 79  
Data columns (total 7 columns):  
#   Column          Non-Null Count  Dtype  
---  ---            -  
0   vehicle_type    80 non-null     object  
1   name            60 non-null     object  
2   year            80 non-null     int64  
3   odometer        80 non-null     int64  
4   fuel            80 non-null     object  
5   owner           80 non-null     object  
6   selling_price   80 non-null     int64  
dtypes: int64(3), object(4)  
memory usage: 4.5+ KB  
None  
>>> |
```

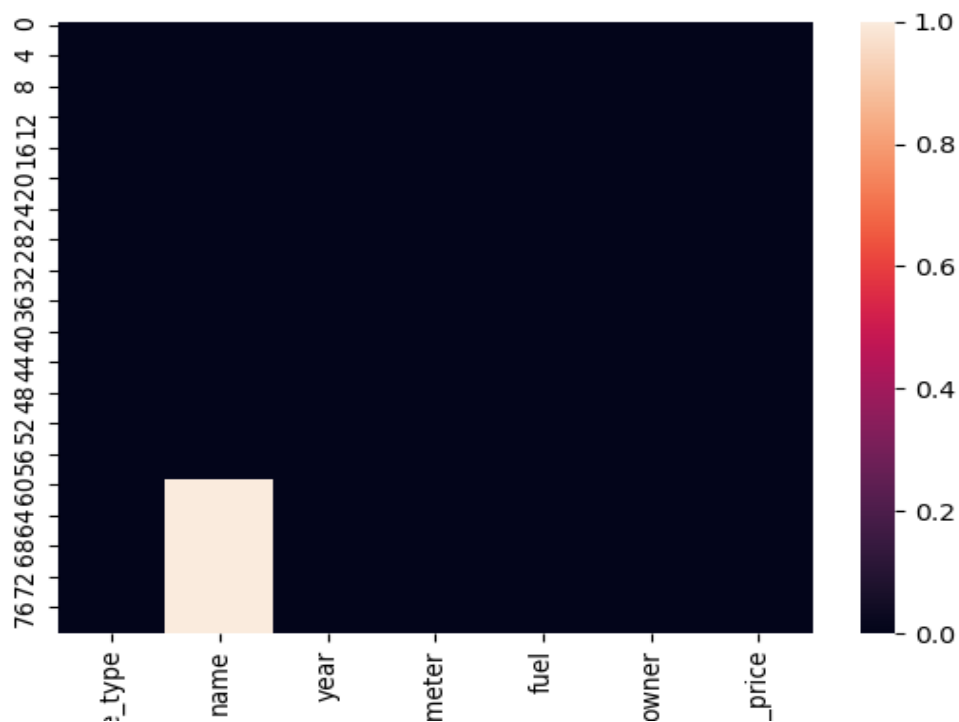
Print(dataset.columns)

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
Index(['vehicle_type', 'name', 'year', 'odometer', 'fuel', 'owner',
      'selling_price'],
      dtype='object')
>>> |
```

We check for any null values

```
sns.heatmap(dataset.isnull())
```

```
plt.show()
```



Vehicle_type column has null values but we ignore as we are not using it to train our model.

Price:

summary(price)							
##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
##	45000	50000	95000	1100000	6780000	6300000	

From the summary above, we see that prices go up to over 6300000. This is obviously wrong. While looking at all cars over 61000 in more detail, I noticed that many of these prices seemed either entered at random or confused with kilometers: I found patterns such as '111111' or '12345678', or mainstream models over 150,000. To try and filter out most of these issues, I assumed that such high-end cars would most likely be coupes, convertibles or SUVs. I also dropped any observation above 200,000, assuming the majority of them would be input errors. I then dropped any row that did not match these criteria and looked at the brands of cars above 75,0000

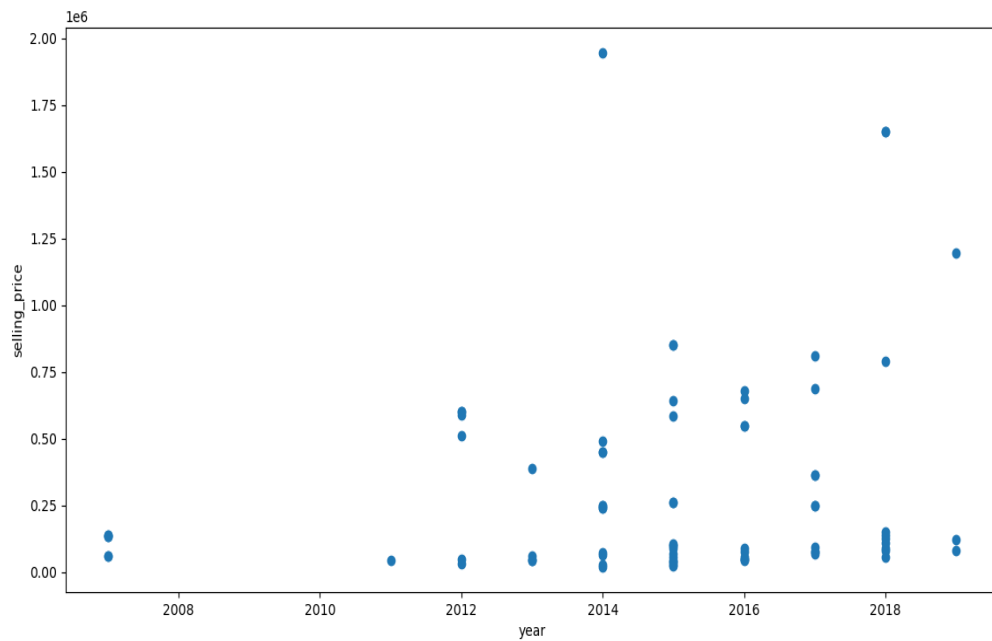
Plotting a graph of independent v/s dependent values:

price column is the dependent value and rest of the column is independent values. So we plot a graph for each independent columns versus price column.

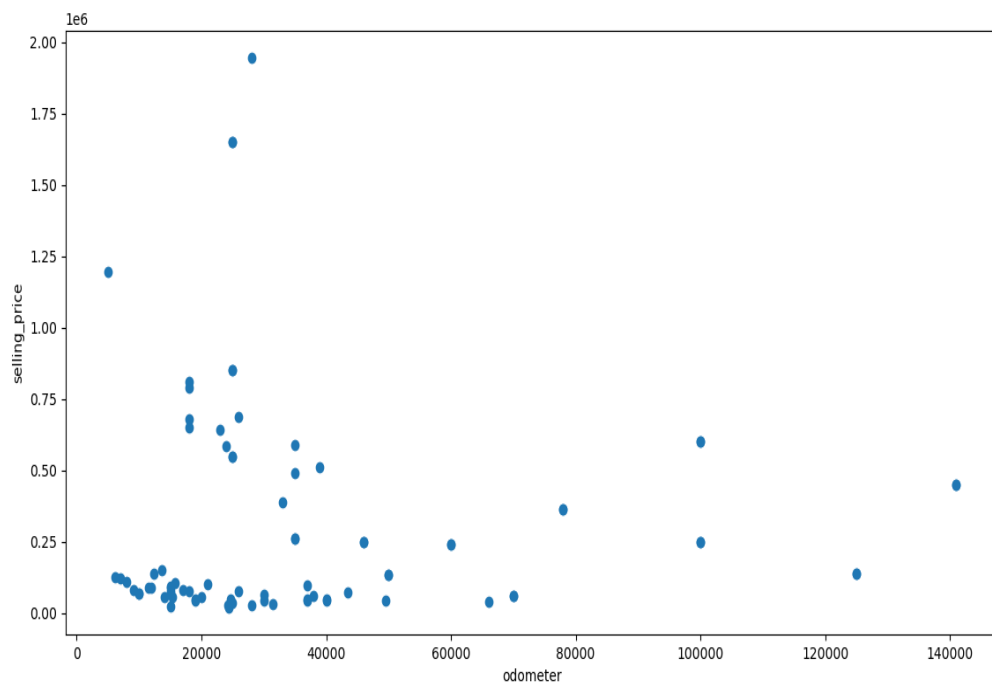
Now we plot a graph against vehicle type versus price:



Now we plot a graph against manufacturing year versus price:

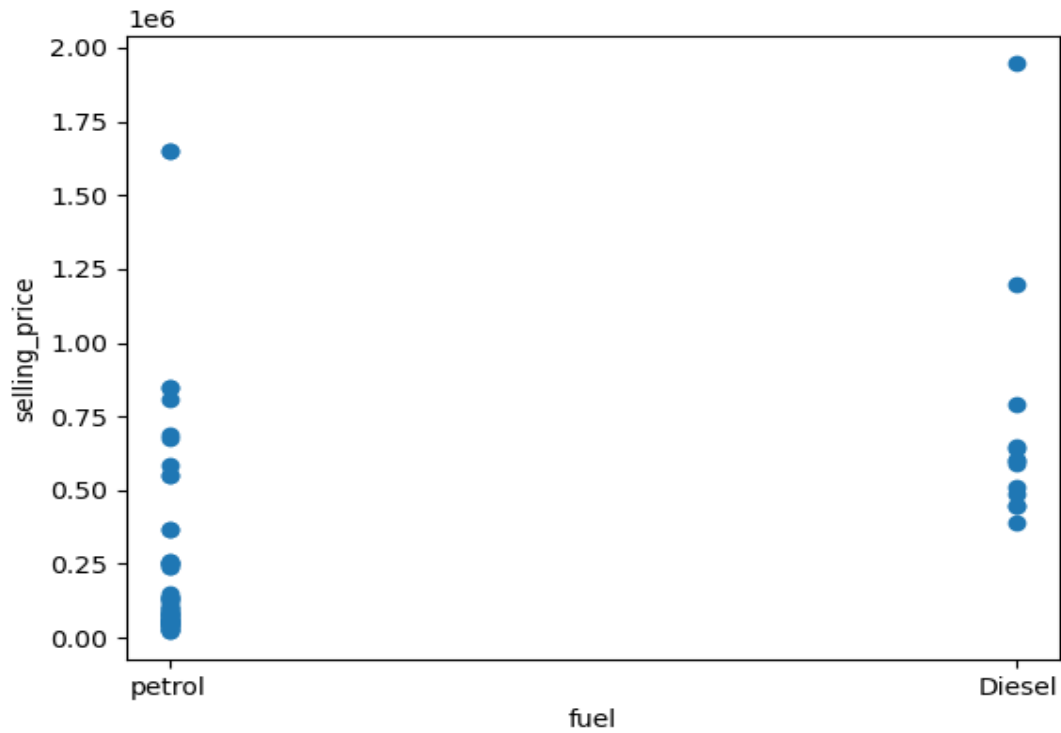


Now we plot a graph against odometer versus price:



Now we plot a graph against fuel type versus price:

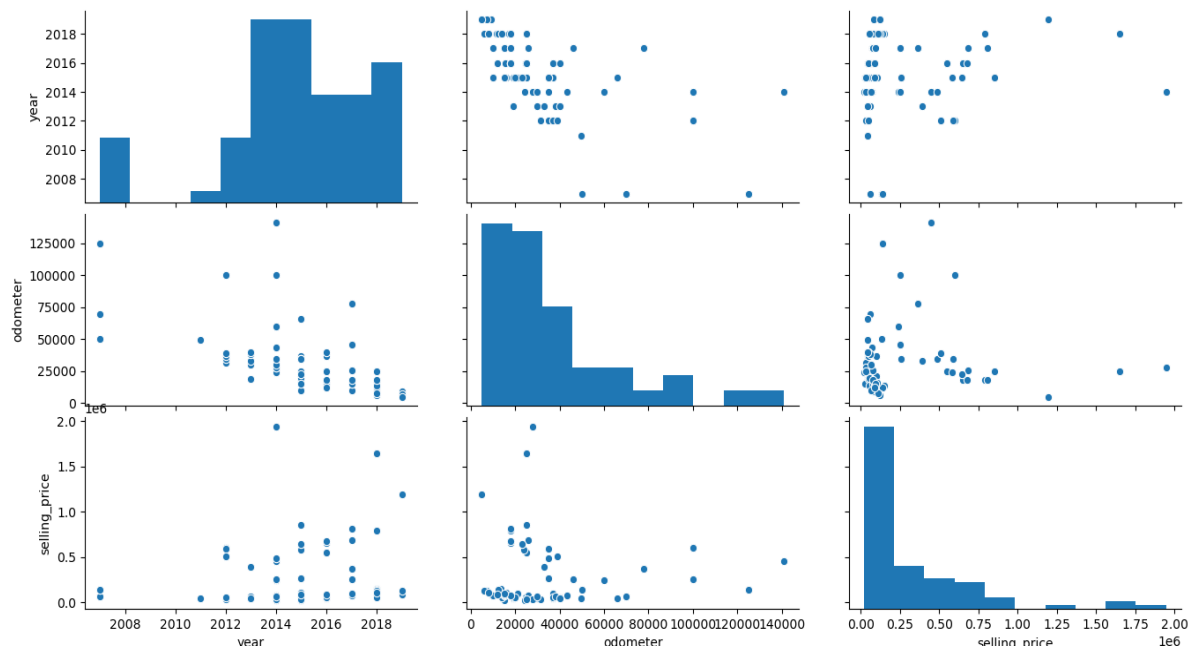
This factor variable also contains the fuel type of vehicle.



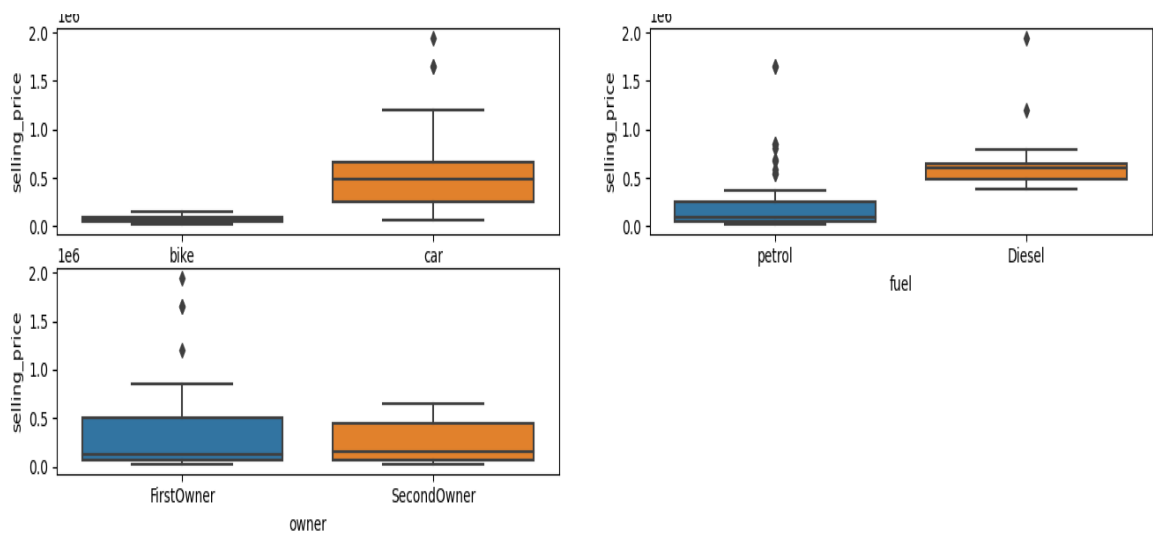
Now we plot a graph against owner versus price:



Visualising numeric variables:



Visualising categorical variables:



Creating Dummies for categorical columns:

Categorical Variables are converted into Numerical Variables with the help of Dummy Variable as machine learning algorithm only understands numeric values.

```
vehicle=pd.get_dummies(dataset['vehicle_type'],drop_first=True)
```

```
print(vehicle)
```

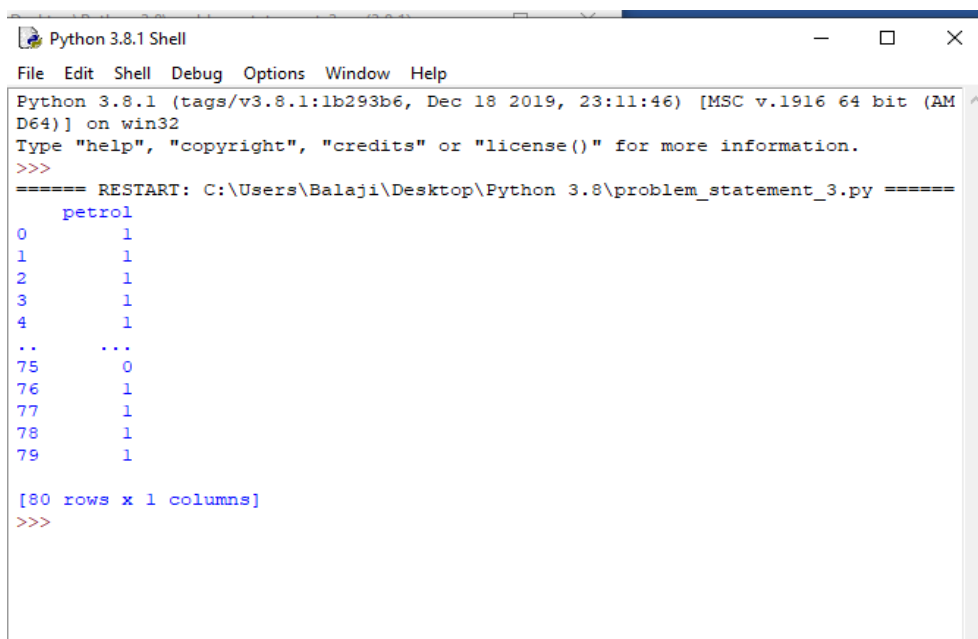


```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
      car
0      0
1      1
2      0
3      1
4      0
..    ...
75     1
76     0
77     0
78     1
79     0

[80 rows x 1 columns]
>>> |
```

```
Fuel=pd.get_dummies(dataset['fuel'],drop_first=True)
```

```
print(Fuel)
```

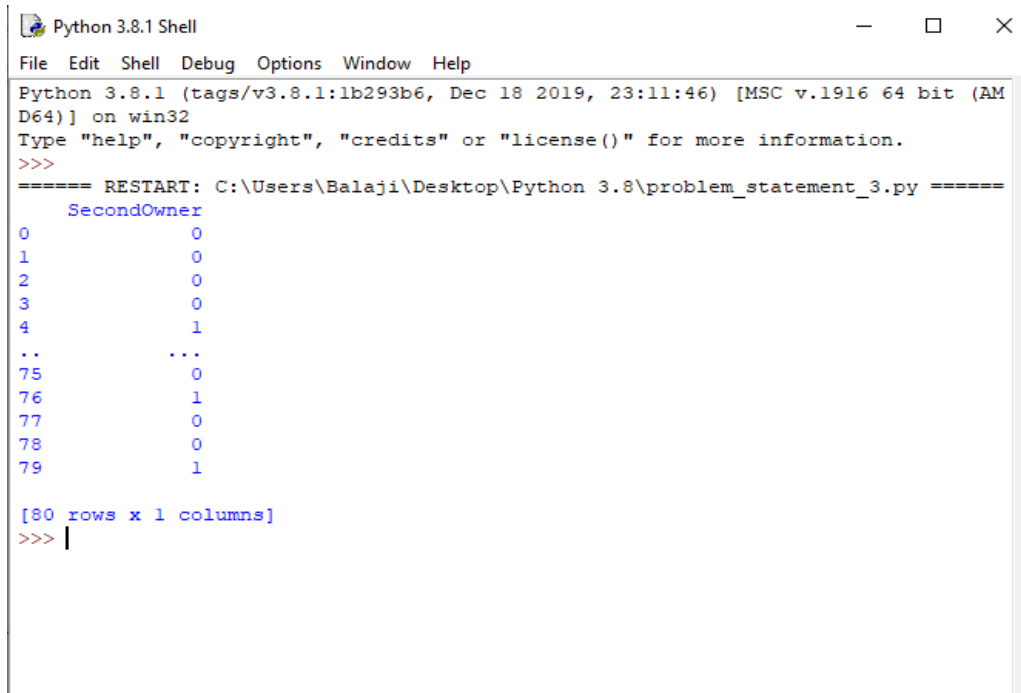


```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
      petrol
0         1
1         1
2         1
3         1
4         1
..    ...
75        0
76         1
77         1
78         1
79         1

[80 rows x 1 columns]
>>>
```

```
Owner=pd.get_dummies(dataset['owner'],drop_first=True)
```

```
print(Owner)
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
      SecondOwner
0              0
1              0
2              0
3              0
4              1
..            ...
75             0
76             1
77             0
78             0
79             1

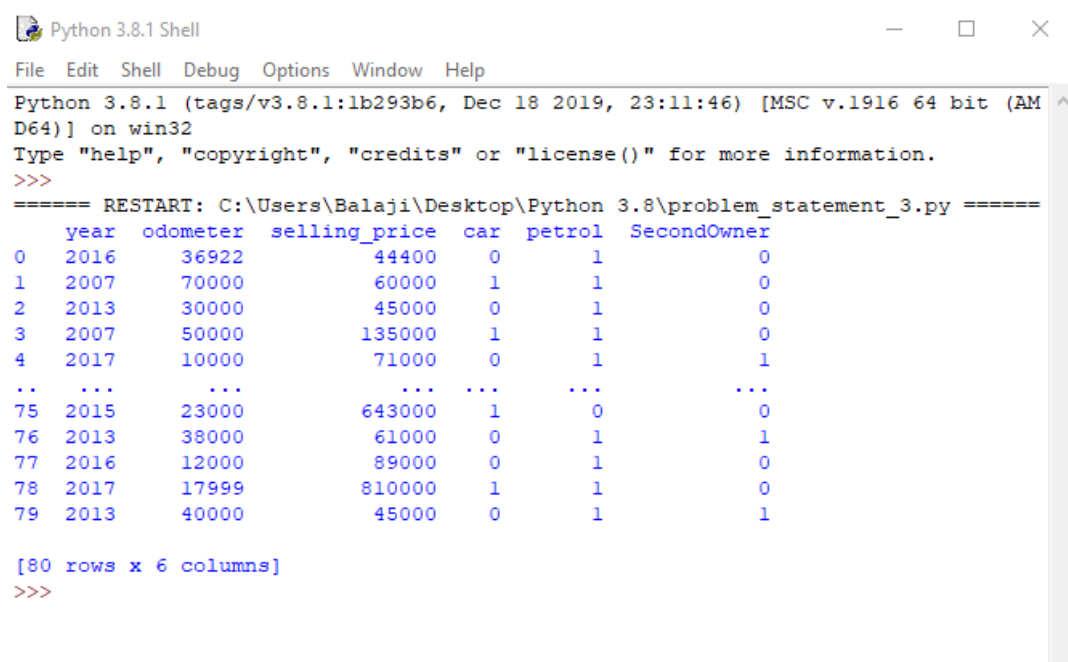
[80 rows x 1 columns]
>>> |
```

Now that we have created the dummy variables we drop the categorical variables and add the dummy variables to the dataset and view the new dataset with only numeric values.

```
dataset.drop(['vehicle_type','name','fuel','owner'],axis=1,inplace=True)
```

```
dataset=pd.concat([dataset,vehicle,Fuel,Owner],axis=1)
```

```
print(dataset)
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
   year  odometer  selling_price  car  petrol  SecondOwner
0  2016    36922         44400    0     1           0
1  2007    70000         60000    1     1           0
2  2013    30000         45000    0     1           0
3  2007    50000        135000    1     1           0
4  2017    10000         71000    0     1           1
..    ...      ...          ...    ...     ...         ...
75 2015    23000        643000    1     0           0
76 2013    38000         61000    0     1           1
77 2016    12000         89000    0     1           0
78 2017    17999        810000    1     1           0
79 2013    40000         45000    0     1           1

[80 rows x 6 columns]
>>>
```

Model Building:

Splitting the Data into Training and Testing sets:

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(dataset.drop('selling_price',axis=1),dataset['selling_price'],test_size=0.20,random_state=50)
```

Training the model using linear regression:

```
from sklearn.linear_model import LinearRegression
```

```
linreg=LinearRegression()
```

```
linreg.fit(x_train,y_train)
```

```
y_pred=linreg.predict(x_test)
```

```
print(y_pred)
```

```
y_pred1=linreg.predict([[2014,28000,1,0,0]])
```

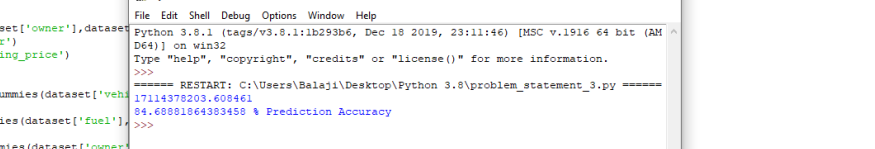
```
#print(y pred1)
```

Calculating the mean squared error and model accuracy:

```
from sklearn.metrics import mean_squared_error
```

```
print(mean_squared_error(y_test,y_pred))
```

```
print(linreg.score(x_test, y_test)*100,'% Prediction Accuracy')
```



The screenshot displays a Jupyter Notebook interface with a code cell containing the following Python code:

```
#plt.show()

plt.scatter(dataset['owner'], dataset['vehicle_type'])
plt.xlabel('owner')
plt.ylabel('selling_price')
plt.show()

vehicle=pd.get_dummies(dataset['vehicle_type'])
#print(vehicle)
Fuel=pd.get_dummies(dataset['fuel'])
#print(Fuel)
Owner=pd.get_dummies(dataset['owner'])
#print(Owner)
dataset.drop(['vehicle_type', 'name', 'year'], axis=1)
dataset=pd.concat([dataset, vehicle, fuel, owner])
#print(dataset)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(dataset[['owner', 'fuel', 'vehicle_type', 'year', 'selling_price']], dataset['selling_price'], test_size=0.2, random_state=50)

from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
linreg.fit(x_train, y_train)

y_pred=linreg.predict(x_test)
#print(y_pred)
y_predi=linreg.predict([['2014', 28000, 'petrol', 'honda']])
#print(y_predi)

from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))

print(linreg.score(x_test, y_test)*100)
```

The Python 3.8.1 Shell window shows the execution output:

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Balaji\Desktop\Python 3.8\problem_statement_3.py =====
17114378203.608461
84.68881864383458  Prediction Accuracy
>>>
```

Conclusions:

Well we've achieved what we set out to achieve. We have a fairly accurate linear regression model that will take in features of a car/bike and predict price with a fair amount of accuracy. What's more interesting, in my opinion is the interesting insights we derived from our algorithm. I mean, I'm definitely glad my partner and I were able to create a model that will help people price their cars and bikes correctly to a certain accuracy, helping people is one of the most important things in the world to me. But figuring out for example, that people who drive luxurious vehicles are largely leasing their cars, tells me a lot about the kind of people who drive them. Its these little things that affect the way we see the world around us that really makes me happy I decided to pursue a career in data science.