

CAR PRICE PREDICTION

Study of car prices post covid-19 impact

Problem statement

- With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. With the change in market due to covid 19 impact, the previous ML models are not performing well.

- Extracting the car mode, and the cars manufacturer form the name.
- Now name column is done its work and it can be dropped form the dataset.

jupyter model building Last Checkpoint: an hour ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [2]:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 %matplotlib inline
6 import warnings
7 warnings.filterwarnings('ignore')
```

In [3]:

```
1 data = pd.read_csv('cars24data.csv')
```

In [4]:

```
1 data.head()
```

Out[4]:

	Unnamed: 0	Brand	Transmission	Fuel	Kms	Price	year
0	0	2013 Maruti Swift	Manual	Diesel	65,561 km	3,32,699	2013
1	1	2016 Maruti Swift	Manual	Petrol	58,818 km	4,07,599	2016
2	2	2014 Maruti Swift Dzire	Manual	Diesel	55,542 km	3,43,799	2014
3	3	2018 Maruti Vitara Brezza	Manual	Diesel	71,192 km	6,74,099	2018
4	4	2014 Maruti Swift Dzire	Manual	Diesel	55,542 km	3,43,799	2014

In [5]:

```
1 data.isnull().sum()
```

Out[5]:

```
Unnamed: 0    0
Brand         0
Transmission  0
Fuel          0
Kms           0
Price         0
year          0
dtype: int64
```

In [8]:

```
1 data['Brand'].value_counts()
```

Out[8]:

```
2013 Maruti Swift    1
2016 Maruti Swift    1
2014 Maruti Swift Dzire    1
2018 Maruti Vitara Brezza    1
2014 Maruti Swift Dzire    1
dtype: object
```

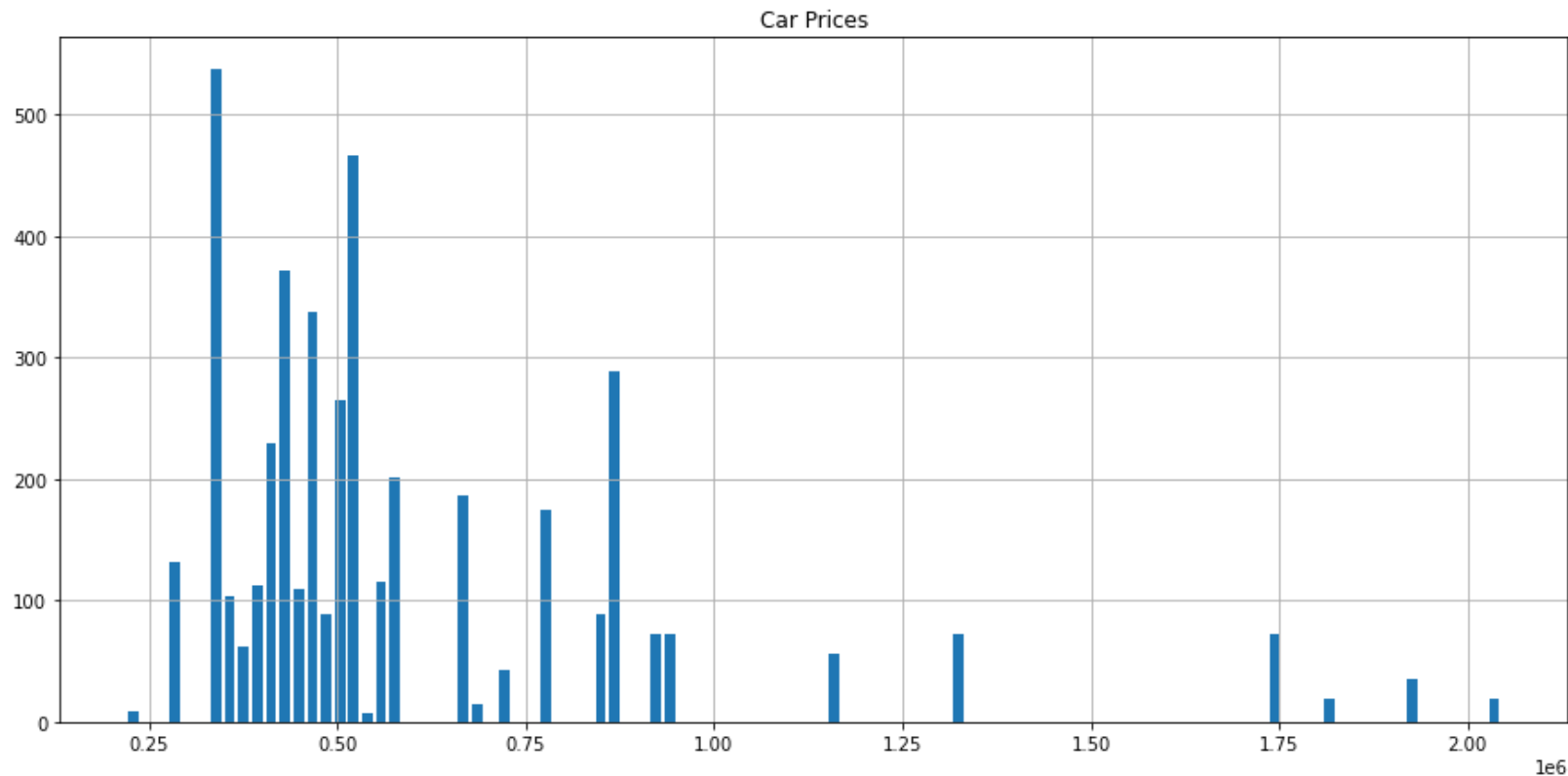
EDA

Minimum Mileage 350 km; Maximum Mileage 1,000,000 km

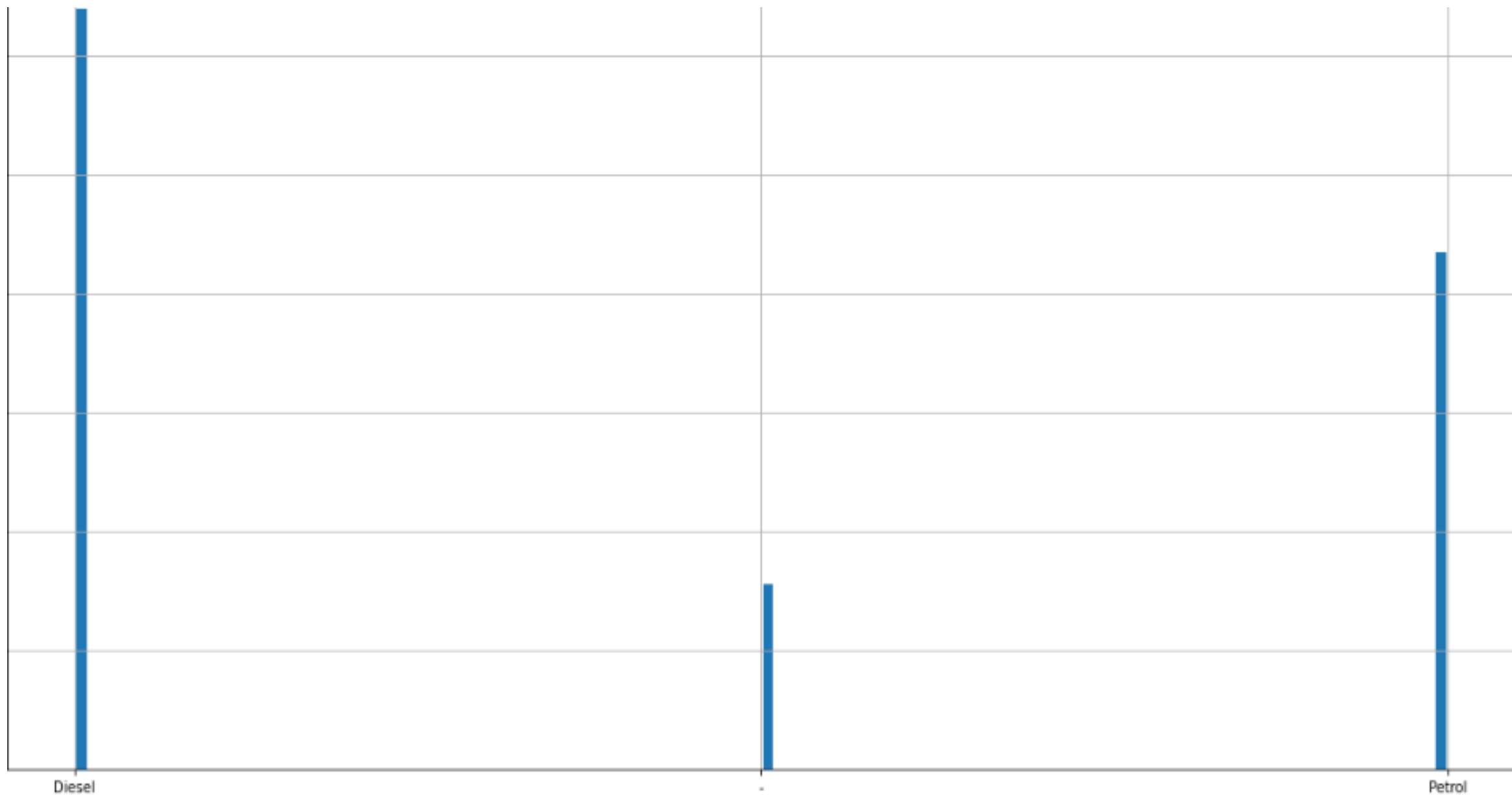
Oldest car: 2008; Newest car: 2021

Minimum price: Rs. 126,000; Maximum price of car: Rs. 29,380,000

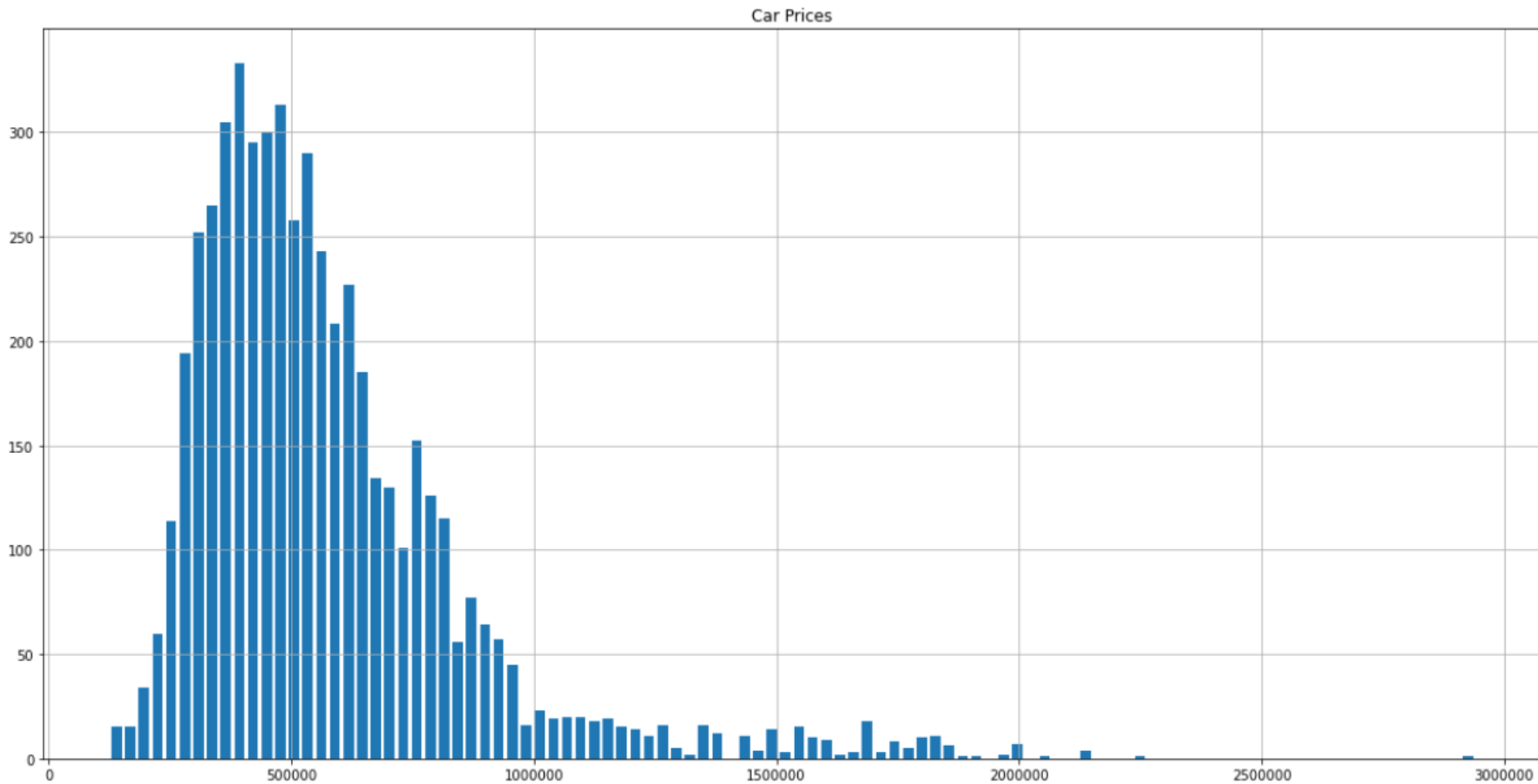
Lets dig deeper into the end points



Max mileage on this i20 car with a mileage of 1002408 and year 2010. the previous owner(s) drove a lot.



Car prices distribution



We can see we have quite some range on the prices of the cars.

Data pre processing

Project M x Home Pa x model bu x Microsoft x Project re x Car_Price x datatrain x car_price x Car_price x Microsoft x car_price x + -

localhost:8888/notebooks/model%20building%20-%20Car%20price%20prediction.ipynb

jupyter model building - Car price prediction Last Checkpoint: 20 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

2012

year

20000

KM

-2

-3

price

```
In [46]: 1 x = data.drop(columns = ['price'],axis=1)
        2 y = data['price']

In [47]: 1 from sklearn.preprocessing import StandardScaler

In [48]: 1 scaler=StandardScaler()
        2 X_scaled=scaler.fit_transform(X)
        3 X_scaled[1]

Out[48]: array([0.21727438, 0.41760452, 1.0496594 , 0.02066029, 0.38061861])

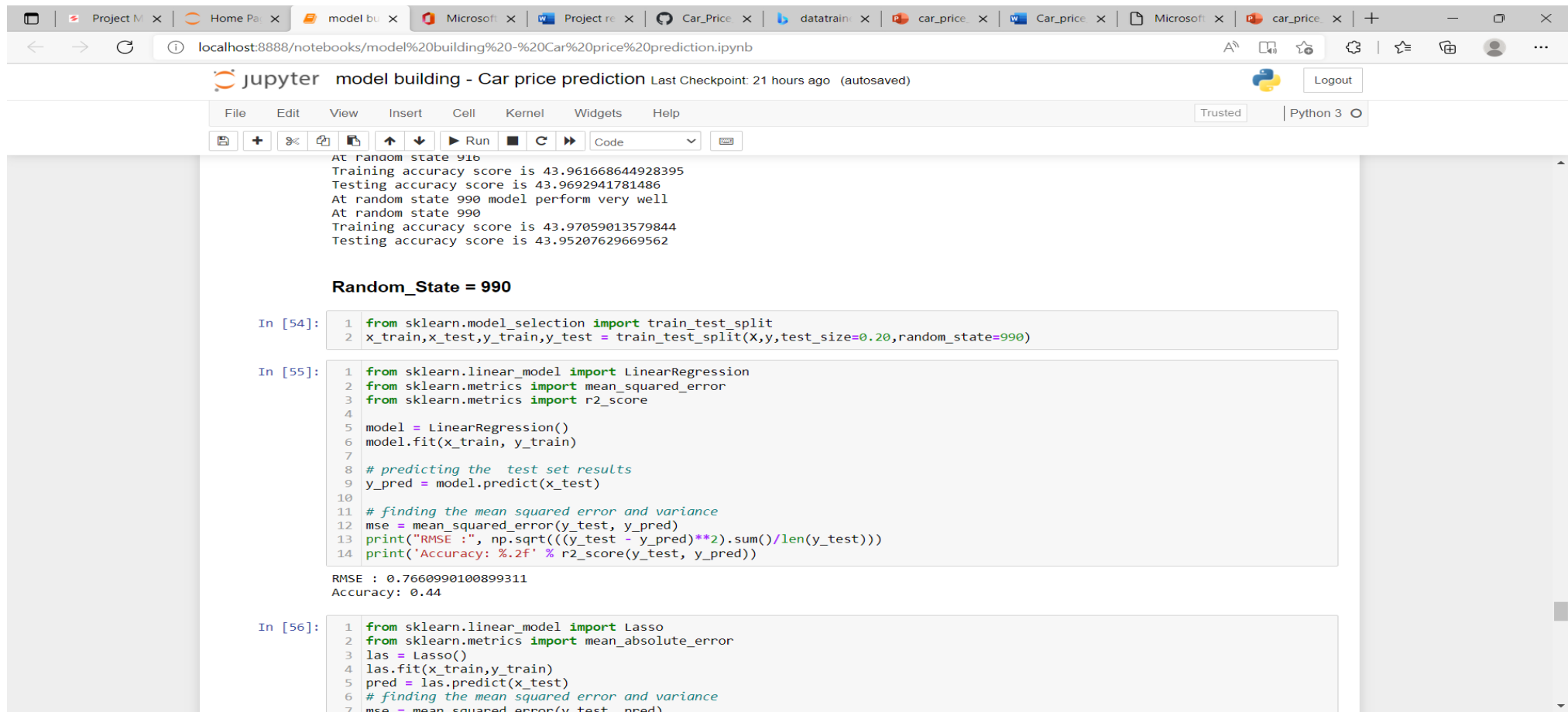
In [49]: 1 from statsmodels.stats.outliers_influence import variance_inflation_factor

In [50]: 1 vif=pd.DataFrame()
        2 vif["vif"]=[variance_inflation_factor(X_scaled,i) for i in range(X_scaled.shape[1])]
        3 vif["Features"]=X.columns
        4
        5 # Lets check the vif
        6 vif

Out[50]:
```

	vif	Features
0	31.830620	Brand
1	1.091228	Transmission
2	1.791599	Fuel
3	33.410995	year
4	2.370920	KM

Model building



The screenshot shows a Jupyter Notebook titled "model building - Car price prediction" running on a local host. The interface includes a top toolbar with file, edit, view, insert, cell, kernel, widgets, and help menus. Below the toolbar is a code editor with several cells. The first cell contains output from a random state 916, showing training and testing accuracy scores. The second cell, labeled "Random_State = 990", contains two code blocks. The first code block imports train_test_split and performs a split. The second code block imports LinearRegression, mean_squared_error, and r2_score, fits a model, and prints the RMSE and Accuracy. The third code block, labeled "In [56]:", imports Lasso and mean_absolute_error, fits a Lasso model, and prints the mean squared error.

```
At random state 916
Training accuracy score is 43.961668644928395
Testing accuracy score is 43.9692941781486
At random state 990 model perform very well
At random state 990
Training accuracy score is 43.97059013579844
Testing accuracy score is 43.95207629669562
```

Random_State = 990

```
In [54]: 1 from sklearn.model_selection import train_test_split
        2 x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=990)
```

```
In [55]: 1 from sklearn.linear_model import LinearRegression
        2 from sklearn.metrics import mean_squared_error
        3 from sklearn.metrics import r2_score
        4
        5 model = LinearRegression()
        6 model.fit(x_train, y_train)
        7
        8 # predicting the test set results
        9 y_pred = model.predict(x_test)
        10
        11 # finding the mean squared error and variance
        12 mse = mean_squared_error(y_test, y_pred)
        13 print("RMSE :", np.sqrt(((y_test - y_pred)**2).sum()/len(y_test)))
        14 print('Accuracy: %.2f' % r2_score(y_test, y_pred))
```

RMSE : 0.7660990100899311
Accuracy: 0.44

```
In [56]: 1 from sklearn.linear_model import Lasso
        2 from sklearn.metrics import mean_absolute_error
        3 las = Lasso()
        4 las.fit(x_train,y_train)
        5 pred = las.predict(x_test)
        6 # finding the mean squared error and variance
        7 mse = mean_squared_error(y_test, pred)
```


Hyper parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = { 'n_estimators' : [100,150],  
                'min_samples_leaf' : [1,2],  
                'min_samples_split' : [2,3],  
                'criterion': ['mse','mae']  
}
```

```
GCV = GridSearchCV(RandomForestRegressor(),parameters,cv=5)
```

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

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),  
             param_grid={'criterion': ['mse', 'mae'],  
                         'min_samples_leaf': [1, 2],  
                         'min_samples_split': [2, 3],  
                         'n_estimators': [100, 150]})
```

```
GCV.best_params_
```

```
{'criterion': 'mse',  
 'min_samples_leaf': 1,  
 'min_samples_split': 2,  
 'n_estimators': 150}
```

Hyper parameter tuning was performed on the best performing algorithm, which was the random forest regression.

These variables were selected, the tuning took many hours to complete.

The best parameters were used to train another accurate model.

Training best model

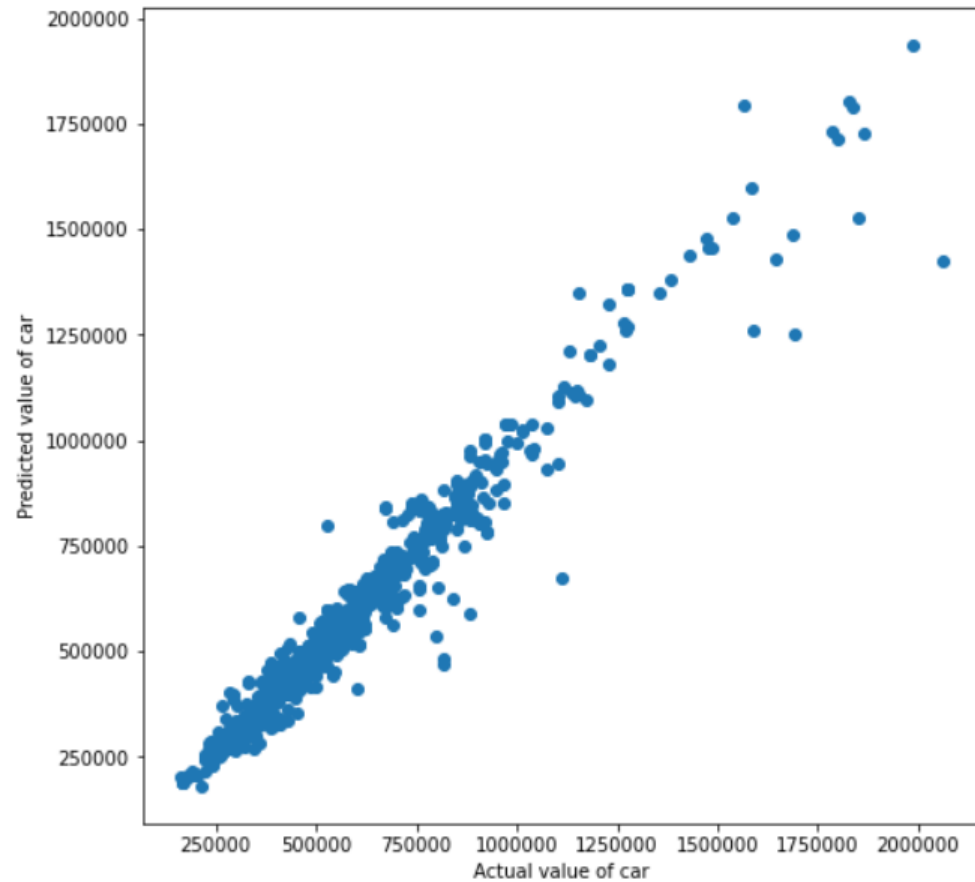
```
mod = RandomForestRegressor(min_samples_leaf= 1, min_samples_split =2, n_estimators = 150, criterion='mse')  
mod.fit(X_train,y_train)  
pred = mod.predict(X_test)  
mod.score(X_test,y_test)
```

0.9630274955526736

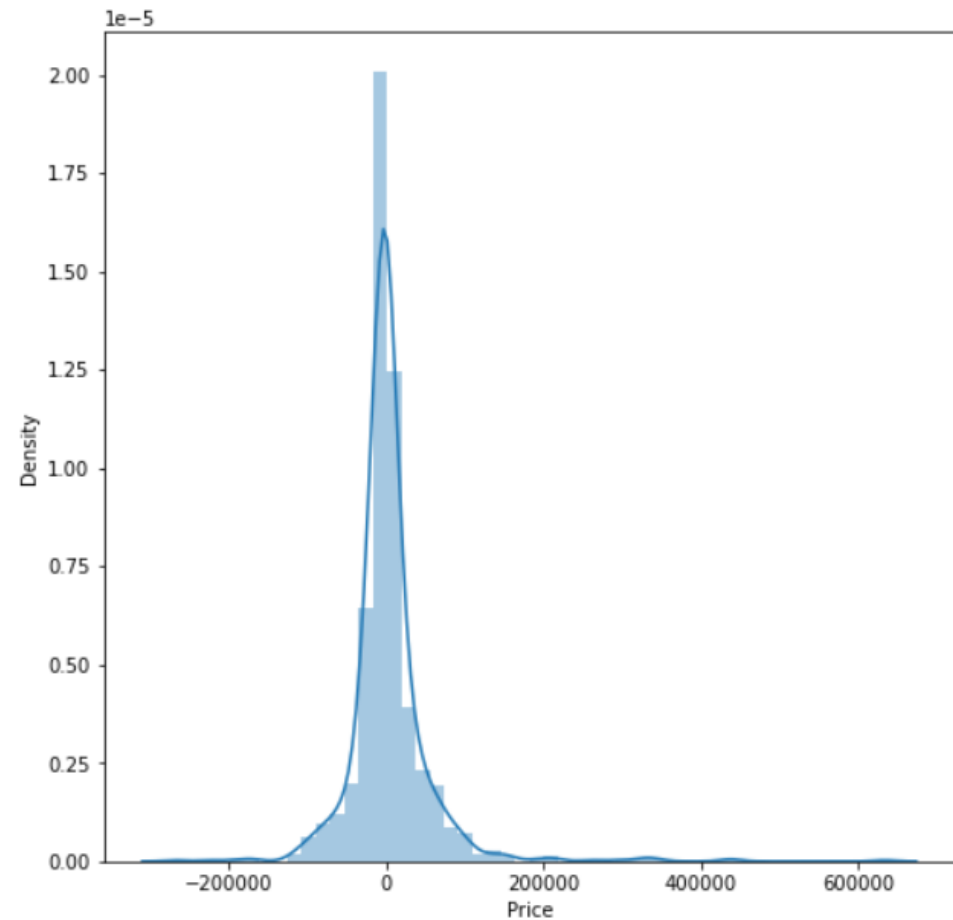
```
scr = cross_val_score(mod, X,y, cv=4)  
print(scr.mean())
```

0.94254259734771

This is the best model with a cross validation, R-squared result of 0.943



Scatter plot of predicted vs actual price



Distribution plot of (predicted value – actual value)

After observing these graphs we can conclude that the model is performing very well and it is ready to be sent to the vendors.