**Price Estimation of Used Cars**

END TERM REPORT

***By***

**Parth Sharma-59**

**Garvit joshi-51**

**Rishabh Sen-55**

**Mridul Pal-52**

K18KY



Department of Intelligent Systems

School of Computer Science Engineering

Lovely Professional University, Jalandhar

April-2020

**Student Declaration**

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Parth Sharma

Roll Number: 59

Rishabh Sen

Roll Number: 55

Garvit joshi

Roll Number: 51

Mridul Pal

Roll Number:52

Jalandhar, Punjab

06/04/2020

**TABLE OF CONTENTS**

1. **Introduction ……………………………………………. 4**
2. **The Code ……………………………………………….. 4**
3. **The Problem ……………………….…………………... 19**
4. **The Data ………………………………….……………. 19**
5. **Bonafide Certificate ………………………………….. 20**

**The Introduction**

The Project was made for price estimation of used cars. The project uses many aspects to find the right price of a car for second hand purposes, it uses fuel type, car body,door numb, car body, wheelbase, engine located, car length, stroke, fuel system, compression, horse power and many more.

**The Code**

#Importing Libraries

import pandas as pd

import numpy as np

import datetime as dt

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import r2\_score         #For Model Evaluation

#reading the dataset

df=pd.read\_csv("CarPrice\_Assignment.csv")

#checking if the data set has any null values

df.isnull().sum(axis=0)

#checking if the data set has duplicates

df.duplicated().sum(axis=0)

#Data Preparation:

#first extracting Company Name from the CarName columns

df['Company']=df.CarName.str.split(' ', expand=True)[0]

#Changing the value for column doornumber and cylindernumber to integers from strings

def doorn(a):

    if a=='two':

        return(2)

    else:

        return(4)

def cylndn(b):

    if b=='two':

        return(2)

    elif b=='three':

        return(3)

    elif b=='four':

        return(4)

    elif b=='five':

        return(5)

    elif b=='six':

        return(6)

    elif b=='eight':

        return(8)

    else:

        return(12)

#changing to the integer values from the strings

df['doorn']=df.doornumber.apply(doorn)

df['cylindern']=df.cylindernumber.apply(cylndn)

#creating a new df and making changes to them

#Dropping the Column CarName

df1=df

#Dropping some of the columns

df1.drop('doornumber',axis=1, inplace=True)

df1.drop('cylindernumber', axis=1, inplace=True)

df1.drop('car\_ID', axis=1, inplace=True)

df1.drop('CarName', axis=1, inplace=True)

#checking the uniqueness of company names

list(df1.Company.unique())

#correcting the company names:

df1.loc[(df1.Company=='maxda'),'Company']='mazda'

df1.loc[(df1.Company=='Nissan'),'Company']='nissan'

df1.loc[(df1.Company=='porcshce'),'Company']='porsche'

df1.loc[(df1.Company=='toyouta'),'Company']='toyota'

df1.loc[(df1.Company=='vokswagen'),'Company']='volkswagen'

df1.loc[(df1.Company=='vw'),'Company']='volkswagen'

# Get the dummy variables for the feature 'Drivewheel' and store it in a new variable - 'gd'

#dropping 4wd

gd=pd.get\_dummies(df1.drivewheel, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('drivewheel', axis=1, inplace=True)

# Get the dummy variables for the feature 'Carbody' and store it in a new variable - 'gd'

gd=pd.get\_dummies(df1.carbody, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('carbody', axis=1, inplace=True)

# Get the dummy variables for the feature 'enginetype' and store it in a new variable - 'gd'

gd=pd.get\_dummies(df1.enginetype, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('enginetype', axis=1, inplace=True)

# Get the dummy variables for the feature 'fuelsystem' and store it in a new variable - 'gd'

#dropping

gd=pd.get\_dummies(df1.fuelsystem, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('fuelsystem', axis=1, inplace=True)

# Get the dummy variables for the feature 'aspiration' and store it in a new variable - 'gd'

#dropping std

gd=pd.get\_dummies(df1.aspiration, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('aspiration', axis=1, inplace=True)

# Get the dummy variables for the feature 'enginelocation' and store it in a new variable - 'gd'

#dropping front

gd=pd.get\_dummies(df1.enginelocation, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('enginelocation', axis=1, inplace=True)

# Get the dummy variables for the feature 'Company' and store it in a new variable - 'gd'

#dropping front

gd=pd.get\_dummies(df1.Company, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('Company', axis=1, inplace=True)

# Get the dummy variables for the feature 'Fueltype' and store it in a new variable - 'gd'

#dropping front

gd=pd.get\_dummies(df1.fueltype, drop\_first=True)

df1=pd.concat([gd,df1],axis=1)

df1.drop('fueltype', axis=1, inplace=True)

'''

sns.pairplot(df1)

plt.show()

'''

plt.figure(figsize=(22, 14))

plt.subplot(2,2,1)

sns.scatterplot(x = 'enginesize', y = 'price', data = df1)

plt.subplot(2,2,2)

sns.scatterplot(x = 'horsepower', y = 'price', data = df1)

plt.subplot(2,2,3)

sns.scatterplot(x = 'wheelbase', y = 'price', data = df1)

plt.subplot(2,2,4)

sns.scatterplot(x = 'carlength', y = 'price', data = df1)

plt.show()

plt.figure(figsize = (20, 10))

sns.boxplot(x = 'carbody', y = 'price', hue = 'cylindern', data = df)

plt.show()

plt.figure(figsize = (20, 10))

sns.boxplot(x = 'carbody', y = 'price', hue = 'fueltype', data = df)

plt.show()

#importing Scikit learn library for modelling

from sklearn.model\_selection import train\_test\_split

np.random.seed(0)

df\_train, df\_test = train\_test\_split(df1, train\_size = 0.7, test\_size = 0.3, random\_state = 100)

#importing MinMaxScarler

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Applying scaler() to all the columns except the 'yes-no' and 'dummy' variables

num\_vars = ['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',

            'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price','symboling']

df\_train[num\_vars] = scaler.fit\_transform(df\_train[num\_vars])

print(df\_train.head())

print(df\_train.describe())

plt.figure(figsize = (32, 20))

sns.heatmap(df\_train.corr(), annot = True, cmap="YlGnBu")

plt.show()

plt.figure(figsize=[6,6])

plt.scatter(df\_train.enginesize, df\_train.price)

plt.show()

y\_train = df\_train.pop('price')

X\_train = df\_train

#importing statsmodel

import statsmodels.api as sm

# Adding a constant

X\_train\_lm = sm.add\_constant(X\_train[['horsepower']])

lr = sm.OLS(y\_train, X\_train\_lm).fit()

lr.params

plt.scatter(X\_train\_lm.iloc[:, 1], y\_train)

plt.plot(X\_train\_lm.iloc[:, 1], -0.0194 + 1.050\*X\_train\_lm.iloc[:, 1], 'r')

plt.show()

print(lr.summary())

#Addding most of the variables/features

X\_train\_lm = X\_train[['horsepower','enginesize', 'rear', 'turbo', '2bbl', '4bbl','wheelbase', 'carlength', 'carwidth', 'idi'

                      ,'mfi', 'mpfi','spdi','spfi','dohcv','l','ohcf','ohcv','hardtop','hatchback','sedan','wagon','fwd','rwd','boreratio','curbweight','stroke','compressionratio','peakrpm','citympg',

                      'highwaympg','doorn','cylindern','gas','audi','bmw','buick','chevrolet','dodge','honda','isuzu','jaguar','mazda','mercury',

                      'mitsubishi','nissan','peugeot','plymouth','porsche','renault','saab','subaru','toyota','volkswagen','volvo','symboling']]

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_lm)

lr = sm.OLS(y\_train, X\_train\_lm).fit()

lr.params

# Check the summary

print(lr.summary())

# Checking for the VIF values of the feature variables.

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

#Turbo happens to be out of p-value range, so removing as its insignificant

X=X\_train\_lm.drop('rear',1)

X\_train\_lm = sm.add\_constant(X)

lr\_2 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_2.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

#2bbl happens to be out of p-value range, so removing as its insignificant

X=X\_train\_lm.drop('idi',1)

X\_train\_lm = sm.add\_constant(X)

lr\_2 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_2.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

#4bbl happens to be out of p-value range, so removing as its insignificant

X=X\_train\_lm.drop('enginesize',1)

X\_train\_lm = sm.add\_constant(X)

lr\_3 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_3.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('gas',1)

X\_train\_lm = sm.add\_constant(X)

lr\_4 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_4.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('subaru',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_36 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_36.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('mpfi',1)

X\_train\_lm = sm.add\_constant(X)

lr\_5 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_5.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('citympg',1)

X\_train\_lm = sm.add\_constant(X)

lr\_6 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_6.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('curbweight',1)

X\_train\_lm = sm.add\_constant(X)

lr\_7 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_7.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('horsepower',1)

X\_train\_lm = sm.add\_constant(X)

lr\_8 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_8.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('sedan',1)

X\_train\_lm = sm.add\_constant(X)

lr\_9 = sm.OLS(y\_train, X\_train\_lm).fit()

print(lr\_9.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('carlength',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_10 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_10.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('peugeot',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_11 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_11.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('carwidth',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_12 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_12.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('rwd',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_13 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_13.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('highwaympg',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_14 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_14.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('toyota',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_15 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_15.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('wheelbase',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_16 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_16.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

X=X\_train\_lm.drop('cylindern',1)

# Build a third fitted model

X\_train\_lm = sm.add\_constant(X)

lr\_17 = sm.OLS(y\_train, X\_train\_lm).fit()

# Print the summary of the model

print(lr\_17.summary())

vif = pd.DataFrame()

vif['Features'] = X\_train\_lm.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train\_lm.values, i) for i in range(X\_train\_lm.shape[1])]

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

vif

y\_train\_price = lr\_17.predict(X\_train\_lm)

# Plotting the histogram of the error terms

fig = plt.figure()

sns.distplot((y\_train - y\_train\_price), bins = 20)

fig.suptitle('Error Terms', fontsize = 20)

plt.xlabel('Errors', fontsize = 18)

#Fitting the test data for predictions

df\_test[num\_vars] = scaler.fit\_transform(df\_test[num\_vars])

df\_test.describe()

y\_test = df\_test.pop('price')

X\_test = df\_test

#Adding Constant

X\_test\_m4 = sm.add\_constant(X\_test)

X\_train\_lm.columns

#Choosing the columns we need for testing

X\_test\_m5=X\_test\_m4[['const', 'turbo', '2bbl', '4bbl', 'mfi', 'spdi', 'spfi', 'dohcv', 'l',

       'ohcf', 'ohcv', 'hardtop', 'hatchback', 'wagon', 'fwd', 'boreratio',

       'stroke', 'compressionratio', 'peakrpm', 'doorn', 'audi', 'bmw',

       'buick', 'chevrolet', 'dodge', 'honda', 'isuzu', 'jaguar', 'mazda',

       'mercury', 'mitsubishi', 'nissan', 'plymouth', 'porsche', 'renault',

       'saab', 'volkswagen', 'volvo', 'symboling']].copy()

y\_pred\_m5 = lr\_17.predict(X\_test\_m5)

## Plotting y\_test and y\_pred to understand the spread

fig = plt.figure()

plt.scatter(y\_test, y\_pred\_m5)

fig.suptitle('y\_test vs y\_pred', fontsize = 20)              # Plot heading

plt.xlabel('y\_test', fontsize = 18)                          # X-label

plt.ylabel('y\_pred', fontsize = 16)

print(lr\_17.summary())

print(r2\_score(y\_test, y\_pred\_m5))

**The Problem**

The prices of new cars 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 offers 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.

**The Data**

The Data Used in This Project was taken from github.com, due to lack of .csv files. I altered the data so that I can test my program with different set of data.

There Were Many Factors that include for price estimation of used cars. We kept the factors in mind a developed the source code, which takes in mind all aspects of a car, such as dents, paint, volume, occupancy and many more.

The Data Is Present in “CarPrice\_Assignement.csv”.

**Technologies and frame to be used**

1. For writing code for this project (price estimation of used car) we have used anaconda Jupiter notebook platform.

2.For our project (price estimation of used car) the code is written in python programming language.

3.We have used different types of libraries:

3.1 pandas.

3.2 Numpy.

3.3 Datetime.

3.4 Seaborn.

3.5 Matplotlib.pyplot.

3.6 Syklearn.metrics and r2\_score.

**BONAFIDE CERTIFICATE**

Certified that this project report Price Estimation of used Cars is the bonafide work of Garvit Joshi, Rishabh Sen, Parth Sharma, Mridul Pal who carried out the project work under my supervision.

Miss Shabnam Sajwan

Assistant Professor

25660

Department of Intelligent System

School of computer Science and Engineering