Predicting Price for Used Cars using Linear Regression

Artificial Intelligence CS-617-A

Avalons

Sacred Heart University
School of Computer Science & Engineering
The Jack Welch College of Business & Technology

Submitted To:
Dr. Reza Sadeghi

Fall 2022

Project Report of Predicting Price for used car using Linear Regression

Team Name

Name of the Team **Avalons**

Team Members

1. Sambasiva Rao Chennamsetty chennamsettys@mail.sacredheart.edu(Team Head)
2. Arif Pasha Shaik shaiks11@mail.sacredheart.edu(TM)
3. Jagadishwar Reddy Velma velmaj@mail.sacredheart.edu (TM)
4. Sai Hrithik Peddi peddis4@mail.sacredheart.edu (TM)
5. Vamsi Kiran Kakkera kakkerav@mail.sacredheart.edu (TM)
6. Kaki Rohit Reddy kakir@mail.sacredheart.edu (TM)

Description of Team Members

1. **Sambasiva Rao Chennamsetty**

   I completed my Bachelor's in Information Technology. I had 3+ years of experience as a full-stack developer with Java programming as a backend. I like to work with a team with more commitment to work.

2. **Arif Pasha Shaik**

   I have completed my Bachelor's in Information Technology, I have done a couple of internships on Visual Basic .net, and Business Analytics: Data mining and Data warehousing.And I love working in a team that has its full dedication.

3. **Jagadishwar Reddy Velma**

   I hold 7+ years of experience in SQL Database Administration. I am here to learn and improve better development skills which help me to become an extensive experienced Core Developer.

4. **Sai Hrithik Peddi**

   I am a graduate student at sacred Heart University. I have completed my Undergraduate in Computer Science. After, I worked as an Android Developer at Sensorise Digital services for 6 months. I'm very passionate about my work role.

5. **Vamsi Kiran Kakkera**

   I have done my Bachelor's degree in the stream of computer science. I'm having work Experience of 2.5 years in the AWS cloud as an Associate Developer. I've chosen this team as they are very coordinative and discuss everything with the team members.

6. **Kaki Rohit Reddy**

   I pursued my Bachelor's in Electronics and Communication Engineering then started working as a .net full stack developer in a reputed organization after that to gain more insight and upgrade my skill set and change my career track I came to the United States to pursue a master's in data science in Sacred Heart University.

Table of Contents

List of Figures

1 Introduction

As the world evolving in all directions significantly, the economic gaps between the people are still exist. The livelihood of different people from different financial backgrounds are changing a lot. When it comes to the comfortable travel the cars are playing a vital role. Also, considering the COVID pandemic, most of the lower- and middle-income group of people also attracting to travel in a safe environment and not willing to choose public transport.

- At the same time the car manufacturers also increased the price of the new cars, which is directly affecting the buying capability of low-income group people.
- Hence, most of the people are looking at the used cars now.
- There are few people who cannot afford to buy new luxury car, but they wish to travel in it. For those, this used cars are the sunlight in dark. [1]
- This used cars has become an opportunity for the business. And it's going to generate a decent revenue for business as well.

1.1 Research Questions

- Which variables are significant in predicting the price of a used car?
- How well those variables describe the price of a car?

1.2 GitHub Repository

https://github.com/samba-chennamsetty/used-car-selling-price-linear-regression

2 Dataset Description

2.1 URL of Dataset

Old Car Selling Price with Linear Regression | Kaggle [2]

2.2 Dataset Explanation

- This dataset contains information about used cars listed on www.cardekho.com [3]
- This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning.

2.3 Features of Dataset

The columns are in the given dataset is as follows:

1. **Car_Name:** This column consists of the name of the cars.
2. **Year:** This column has the year in which the car was bought.
3. **Selling_Price:** This column has the price the owner wants to sell the car at.
4. **Present_Price:** This is the current ex-showroom price of the car.
5. **Kms_Driven:** This is the distance completed by the car in km.
6. **Fuel_Type:** Fuel type of the car.
7. **Seller_Type:** Defines whether the seller is a dealer or an individual.
8. **Transmission:** Defines whether the car is manual or automatic.
9. **Owner:** Defines the number of owners the car previously had.

3 Related Work

We are comparing this model with car prediction prices [4] which has multi linear regression with less no of dependencies and models.

3.1 Pro's

The advantages we have over the other related works are

- Using linear regression model allows us to make our analysis simple.
- Providing a variety of visual representations of impact with each feature.
- It is planned to build multiple models based on type of company.
- Considering the best prediction relational fields.

3.2 Con's

- Based on our source project referred there is no multiple regression, which is also based on many features of the referred project.

4 Project Plan

The project plan has the below steps in it.

1. Data-preprocessing
2. Model building
3. Optimizing Model
4. Model Evaluation

5 Data Exploration

5.1.1 Univariate Analysis:

Univariate analyses are used extensively in quality-of-life research. Univariate analysis is defined as analysis carried out on only one (“uni”) variable (“variate”) to summarize or describe the variable. However, another use of the term “univariate analysis” exists and refers to statistical analyses that involve only one dependent variable and which are used to test hypotheses and draw inferences about populations based on samples, also referred to as univariate.

We find the univariate using distplot and boxplot graphs with below code. Here we’re using only uni one feature for the analysis.

Step 2 : Data Visualization

Univaraite Analysis

In [23]:
```
# variables for plotting.
cont_col =[ 'selling_price','km_driven', 'age']
```

distplot

In [24]:
```
for i in cont_col:
    sns.distplot(car[i])
    plt.show()
```

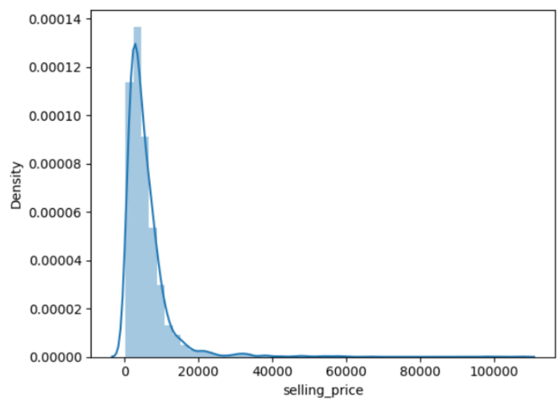


Figure 1: Data Visualization

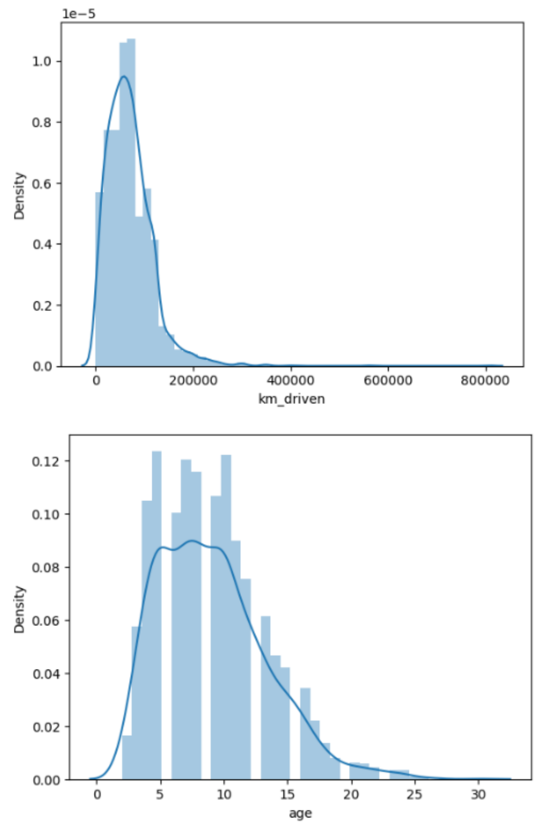


Figure 2: Data Visualization using Univariate Analysis.

Using distplot function Univariate Analysis has been made which gives a similar kind of distribution, some features are showing nearby normal distribution while some are skewed.

Boxplot

- Boxplot - A graphical rendition of statistical data based on the minimum, first quartile, median, third quartile, and maximum.
- In the fig 7 the observation states that the first quartile lies at the lower end of the box and the upper end is the 3rd quartile. The box indicates the range in which the middle 50% of all the data lies.
- The line in between the first quartile and the third quartile lies the median. (Median in the boxplot represents with solid line and the Mean in the boxplot represents with dashed line)

```
In [41]: for i in cont_col:
             plt.title(i)
             sns.boxplot(car[i],showfliers=False)
             plt.show()
```

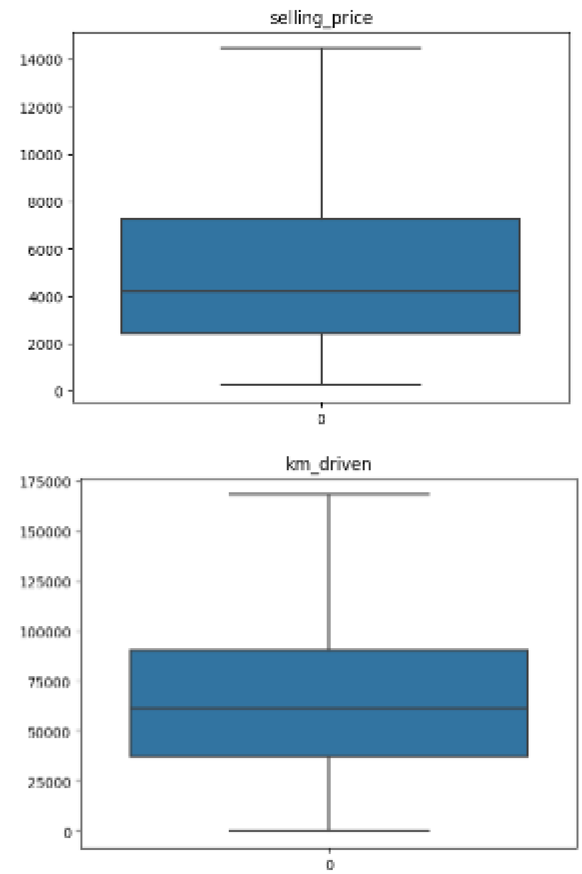


Figure 3: Box Plots

5.2 Bivariate Analysis:

- Bivariate analysis refers to the analysis of two variables to determine relationships between them. Bivariate analyses are often reported in quality-of-life research. For an excellent example of research that utilizes bivariate analyses and demonstrates how the results of bivariate analyses can be used to inform furthermore complex analyses.
- We find the relation between Selling Price and Car age which is bi with scatter plotting.

Bivaraite Analysis

```
In [42]: # variables for plotting
         cont_col =[ 'selling_price','km_driven', 'fuel_Petrol']
```

scatter plot

```
In [43]: # plotting graph b/w count and continuous columns taking transmission as hue
         for i in cont_col:
             sns.scatterplot(data = car[i], x = car[i], y = car['age'], hue=car['transmission_Manual'])
             plt.show()
```

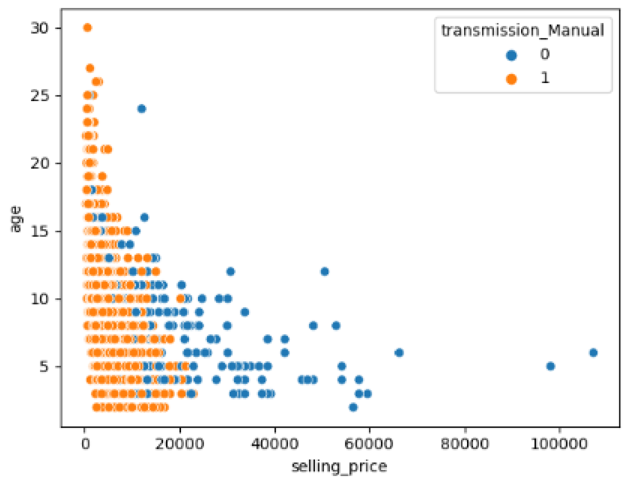


Figure 4: Bivariate Analysis

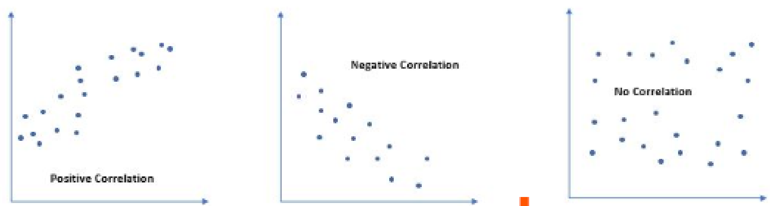
- Here we can notice that as the age goes up the selling prices decreases.
- And most of the manual transmission cars are under the age 15 and price range of 20000.

5.2.1 Pearson Correlation:

Pearson's correlation coefficient is the covariance of the two variables divided by the product of their standard deviations. The form of the definition involves a "product moment", that is, the mean (the first moment about the origin) of the product of the mean-adjusted random variables; hence the modifier product-moment in the name.

Pearson correlation

- Value of 'r' ranges from '-1' to '+1'.
- Value '0' specifies that there is no relation between the two variables.
- A value greater than '0' indicates a positive relationship between two variables where an increase in the value of one variable increases the value of another variable.
- Value less than '0' indicates a negative relationship between two variables where an increase in the value of one decreases the value of another variable.



In [44]:
```
from scipy.stats import pearsonr
corr, _ = pearsonr(car['selling_price'],car['age'])
print("The value of Pearson correlation for Selling Price vs Age is: " + str(corr))

corr2, _ = pearsonr(car['selling_price'],car['km_driven'])
print("The value of Pearson correlation for Selling Price vs Age is: " + str(corr2))
```

```
The value of Pearson correlation for Selling Price vs Age is: -0.42405085293069356
The value of Pearson correlation for Selling Price vs Age is: -0.18735641383299767
```

Inference

- The value of Pearson correlation for Selling Price vs Age is -0.424 and as there is no correlation between Selling Price and Age, hence the Pearson Correlation value between Selling Price & Age is less than 0 which is -0.424.
- The value of Pearson correlation for Selling Price vs Km Driven is -0.187 and as there is no correlation between Selling Price and Km Driven, hence the Pearson Correlation value between Selling Price & Km Driven is less than 0 which is -0.187.

Figure 5: Pearson Correlation

5.2.2 Correlation Matrix:

A correlation heatmap is a graphical representation of a correlation matrix representing the correlation between different variables. The value of correlation can take any value from -1 to 1. Correlation between two random variables or bivariate data does not necessarily imply a causal relationship.

Correlation Matrix

In [46]:
```
# Let's check the correlation coefficients to see which variables are highly correlated. Note:
# here we are considering only those variables (dataframe: car) that were chosen for analysis

plt.figure(figsize = (25,20))
sns.heatmap(car.corr(), annot = True, cmap="RdBu")
plt.show()
```

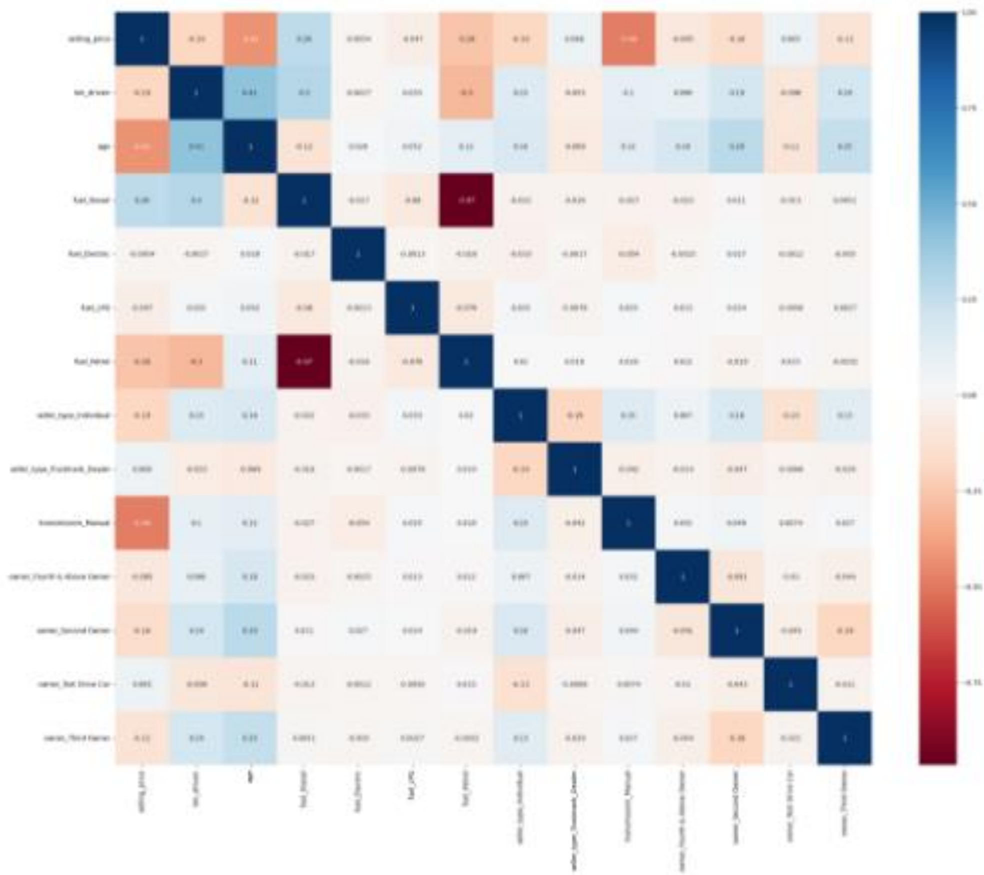


Figure 6: Correlation Matrix - HeatMap

5.2.3 Pair Plot:

Here we took selling price and compare it with km driven and age of car.

pair plot

```
In [28]: def pp(x,y):
             sns.pairplot(car, x_vars=[x,y], y_vars='selling_price',size=4, aspect=1, kind='scatter')
             plt.show()

         pp('km_driven', 'age')
```

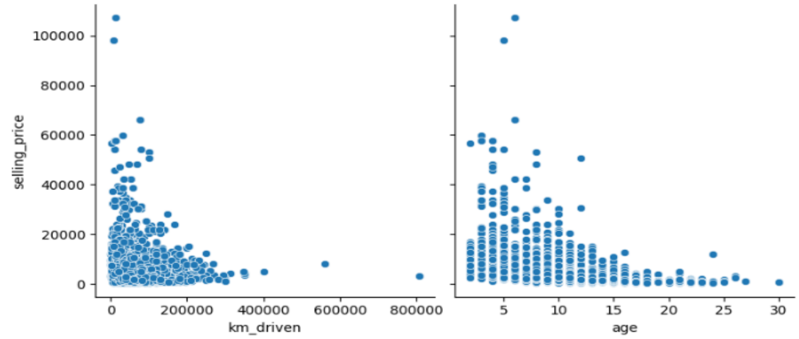


Figure 7: Pair Plot

- Selling price is compared with kms driven and age of the car.
- The pair plot graphs portray the comparison between selling price and kms driven along with age.
- Here, the observation states that the increase in kms driven makes the selling price decrease, vice-versa i.e., data shown in fig9. The car that has driven 800000 kms has a selling price of 0. Whereas highest selling price which is more than 100000 has only driven very less (near to 0 or 10000)

6 Data Modeling

6.1 Pre-Processing

- We import the dataset using the function pd.read_csv("UsedCarDetails.csv") and we are looking for the head rows in the dataset.

Step 1: Reading and Understanding the Data

```
Let's start with the following steps:

 - Importing data using the pandas library
 - Understanding the structure of the data
```

```
In [1]: import warnings
        warnings.filterwarnings('ignore')

In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

In [3]: car = pd.read_csv("UsedCarDetails.csv")

In [4]: car.head()
```

Out[4]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

Figure 8: Data Reading

```
In [8]: car.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 4340 entries, 0 to 4339
        Data columns (total 7 columns):
         #   Column         Non-Null Count  Dtype
        ---  ------         --------------  -----
         0   selling_price  4340 non-null   int64
         1   km_driven      4340 non-null   int64
         2   fuel           4340 non-null   object
         3   seller_type    4340 non-null   object
         4   transmission   4340 non-null   object
         5   owner          4340 non-null   object
         6   age            4340 non-null   int64
        dtypes: int64(3), object(4)
        memory usage: 237.5+ KB

In [9]: car.shape

Out[9]: (4340, 7)

In [10]: car.columns

Out[10]: Index(['selling_price', 'km_driven', 'fuel', 'seller_type', 'transmission',
               'owner', 'age'],
              dtype='object')
```

Figure 9: Info and Columns of Dataset

Here we are,

- Using the .shape function, to find the number of rows and columns in the dataset.
- Using .columns function to view the columns in the function.
- Using .info function to know all the details of the car data set with their datatype.

6.2 Data Splitting

- Adding a new variable for calculating the age of the car.
- As part of this we clean the unwanted data and make the data right and good for the model with error free.

Adding new variable

In [5]:
```
# adding new variable 'current-year' to the car dataframe to calculate car age.
car['current']= 2022
```

Adding new variable age column

In [6]:
```
# calculating current age.
car['age']=car['current']-car['year']

car.head()
```

Out[6]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	current	age
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner	2022	15
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner	2022	15
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner	2022	10
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner	2022	5
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner	2022	8

Step 2 : Data Cleaning and Preparation

Drop all non required or repeative data

In [7]:
```
car.drop(['current','year','name'],axis=1,inplace=True)
car.head()
```

Out[7]:

	selling_price	km_driven	fuel	seller_type	transmission	owner	age
0	60000	70000	Petrol	Individual	Manual	First Owner	15
1	135000	50000	Petrol	Individual	Manual	First Owner	15
2	600000	100000	Diesel	Individual	Manual	First Owner	10
3	250000	46000	Petrol	Individual	Manual	First Owner	5
4	450000	141000	Diesel	Individual	Manual	Second Owner	8

Figure 10: Data Cleaning and Preparation

Duplicate Data Check

Checking if there is any duplicate data and dropping the entire duplicate row if any

Duplicate Check

In [14]:
```
car_dub=car.copy()
# Checking for duplicates and dropping the entire duplicate row if any
car_dub.drop_duplicates(subset=None, inplace=True)
```

In [15]:
```
car_dub.shape
```

Out[15]: (3498, 7)

In [16]:
```
car.shape
```

Out[16]: (4340, 7)

Insights

- The shape after running the drop duplicate command is not same as the original dataframe.

Figure 11: Checking Duplicates and dropping

Identifying junk values:

Checking value_counts() for entire dataframe.

This will help to identify any Unknow/Junk values present in the dataset.

In [20]:
```
for col in car:
    print(car[col].value_counts(ascending=False), '\n\n\n')
```

```
300000     122
250000     107
350000     104
550000      82
150000      81
          ...
2595000      1
368000       1
248000       1
641000       1
865000       1
Name: selling_price, Length: 445, dtype: int64



70000     202
80000     197
120000    192
60000     189
50000     171
         ...
35925       1
40771       1
30500       1
55800       1
112198      1
Name: km_driven, Length: 770, dtype: int64



Diesel      1762
Petrol      1676
CNG           37
LPG           22
Electric       1
Name: fuel, dtype: int64

Individual          2753
Dealer               712
Trustmark Dealer      33
Name: seller_type, dtype: int64



Manual       3187
Automatic     311
Name: transmission, dtype: int64



First Owner             2157
Second Owner             964
Third Owner              285
Fourth & Above Owner      75
Test Drive Car            17
Name: owner, dtype: int64



5     336
10    332
7     327
8     315
9     290
4     285
6     273
11    244
12    205
13    167
3     156
14    127
15    114
16     93
17     60
2      45
18     37
19     22
20     17
21     16
22     12
24      9
23      9
25      3
26      2
27      1
30      1
Name: age, dtype: int64
```

Insights

- There seems to be no Junk/Unknown values in the entire dataset.

Figure 12: Identifying junk values

- **Junk Values** – Data that doesn't serve any real purpose.
- We found that there is no Junk or Unknown values exists in the data set.

- The above figure states that the .value_count function gives the count of each column values. For example, 300000 repeats 122 times in the entire dataset and same for the rest.

6.3 Data Fitting

```
In [71]: X_train_new = X_test.drop(["fuel_Electric"], axis = 1)

In [72]: # Check for the VIF values of the feature variables.
         from statsmodels.stats.outliers_influence import variance_inflation_factor

         # Create a dataframe that will contain the names of all the feature variables and their respective VIFs
         vif = pd.DataFrame()
         vif['Features'] = X_train_new.columns
         vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i in range(X_train_new.shape[1])]
         vif['VIF'] = round(vif['VIF'], 2)
         vif = vif.sort_values(by = "VIF", ascending = False)
         vif

Out[72]:
```

	Features	VIF
1	age	4.42
3	transmission_Manual	4.27
0	km_driven	4.22
2	fuel_Diesel	2.20
4	owner_Test Drive Car	1.01

```
In [73]: # Add a constant
         X_train_lm2 = sm.add_constant(X_train_new)

         # Create a first fitted model
         lr2 = sm.OLS(y_train, X_train_lm2).fit()

In [74]: # Check the parameters obtained

         lr2.params

Out[74]: const                   0.158916
         km_driven              -0.079647
         age                    -0.131720
         fuel_Diesel             0.032285
         transmission_Manual    -0.087426
         owner_Test Drive Car    0.023847
         dtype: float64
```

Figure 13: Splitting Data – OLS

6.4 Measuring Performance

6.4.1 Plot ROC curve

- ROC curves in logistic regression are used for determining the best cutoff value for predicting whether a new observation is a "failure" (0) or a "success" (1).

```
In [59]: from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, plot_roc_curve
         # print accuracy
         scoreLR = accuracy_score(y_test, predictionsLR)
         scoreKNN = accuracy_score(y_test, predictionsKNN)
         print(f' the accuracy of LR and KNN are {scoreLR} and {scoreKNN} , respectively')
         # print confusion matrix including precision, recall, f1-score
         print("LR model details")
         print(classification_report(y_test, predictionsLR))
         tn, fp, fn, tp = confusion_matrix(y_test, predictionsLR).ravel()
         print(f'TN = {tn}, FP = {fp}, FN = {fn}, TP= {tp}')
         plot_roc_curve(LR, X_test, y_test)

         print("KNN model details")
         print(classification_report(y_test, predictionsKNN))
         tn, fp, fn, tp = confusion_matrix(y_test, predictionsKNN).ravel()
         print(f'TN = {tn}, FP = {fp}, FN = {fn}, TP= {tp}')
         plot_roc_curve(KNN, X_test, y_test)

          the accuracy of LR and KNN are 1.0 and 1.0 , respectively
         LR model details
                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00       254
                    1       1.00      1.00      1.00       236

             accuracy                           1.00       490
            macro avg       1.00      1.00      1.00       490
         weighted avg       1.00      1.00      1.00       490

         TN = 254, FP = 0, FN = 0, TP= 236
         KNN model details
                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00       254
                    1       1.00      1.00      1.00       236

             accuracy                           1.00       490
            macro avg       1.00      1.00      1.00       490
         weighted avg       1.00      1.00      1.00       490

         TN = 254, FP = 0, FN = 0, TP= 236
Out[59]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x21080ccb7c0>
```

Figure 14: Measuring performance using ROC Curve

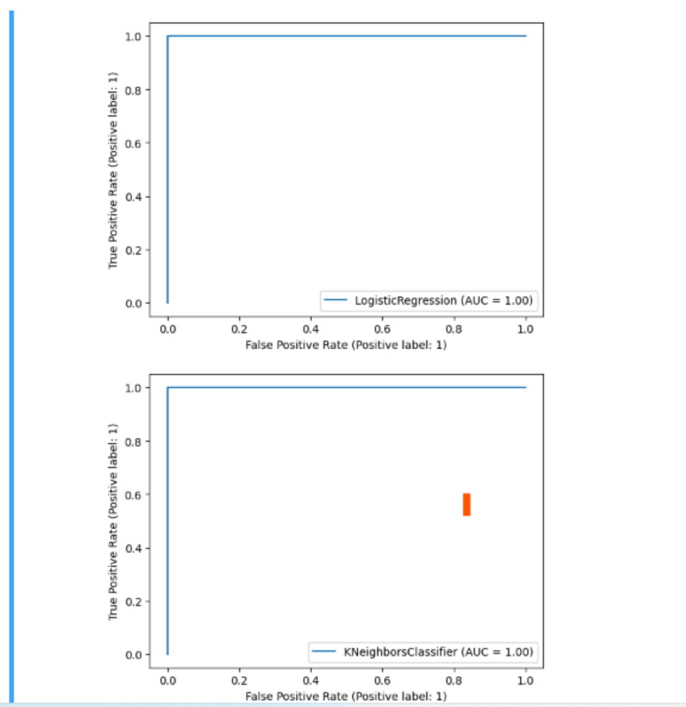


Figure 15: ROC Graph

6.4.2 **Model 1**

```
In [70]: # Print a summary of the linear regression model obtained
         print(lr1.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.434
Model:                            OLS   Adj. R-squared:                  0.433
Method:                 Least Squares   F-statistic:                     312.2
Date:                Sat, 19 Nov 2022   Prob (F-statistic):         1.98e-297
Time:                        03:01:01   Log-Likelihood:                 4122.4
No. Observations:                2448   AIC:                            -8231.
Df Residuals:                    2441   BIC:                            -8190.
Df Model:                           6
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                     0.1591      0.003     45.595      0.000       0.152       0.166
km_driven                -0.0798      0.018     -4.558      0.000      -0.114      -0.045
age                      -0.1314      0.007    -19.135      0.000      -0.145      -0.118
fuel_Diesel               0.0323      0.002     16.291      0.000       0.028       0.036
fuel_Electric            -0.0516      0.045     -1.144      0.253      -0.140       0.037
transmission_Manual      -0.0877      0.003    -27.029      0.000      -0.094      -0.081
owner_Test Drive Car      0.0239      0.014      1.745      0.081      -0.003       0.051
==============================================================================
Omnibus:                     2669.806   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           446489.043
Skew:                           5.147   Prob(JB):                         0.00
Kurtosis:                      68.356   Cond. No.                         73.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 16: Model 1

6.4.3 **Model 2**

- Removing the variable 'fuel_Electric' based on its High p-value

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.434
Model:                            OLS   Adj. R-squared:                  0.433
Method:                 Least Squares   F-statistic:                     374.3
Date:                Sat, 19 Nov 2022   Prob (F-statistic):         1.87e-298
Time:                        03:01:02   Log-Likelihood:                 4121.8
No. Observations:                2448   AIC:                            -8232.
Df Residuals:                    2442   BIC:                            -8197.
Df Model:                           5
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                     0.1589      0.003     45.594      0.000       0.152       0.166
km_driven                -0.0796      0.018     -4.548      0.000      -0.114      -0.045
age                      -0.1317      0.007    -19.190      0.000      -0.145      -0.118
fuel_Diesel               0.0323      0.002     16.305      0.000       0.028       0.036
transmission_Manual      -0.0874      0.003    -27.013      0.000      -0.094      -0.081
owner_Test Drive Car      0.0238      0.014      1.743      0.081      -0.003       0.051
==============================================================================
Omnibus:                     2671.212   Durbin-Watson:                   2.023
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           446732.312
Skew:                           5.152   Prob(JB):                         0.00
Kurtosis:                      68.372   Cond. No.                         29.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 17: Model 2

6.4.4 Model 3

- Removing the variable 'owner_Test Drive Car' based on its High p-value

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.433
Model:                            OLS   Adj. R-squared:                  0.432
Method:                 Least Squares   F-statistic:                     466.7
Date:                Sat, 19 Nov 2022   Prob (F-statistic):          3.70e-299
Time:                        03:01:02   Log-Likelihood:                 4120.3
No. Observations:                2448   AIC:                            -8231.
Df Residuals:                    2443   BIC:                            -8202.
Df Model:                           4
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                   0.1593      0.003     45.771      0.000       0.152       0.166
km_driven              -0.0811      0.018     -4.634      0.000      -0.115      -0.047
age                    -0.1326      0.007    -19.352      0.000      -0.146      -0.119
fuel_Diesel             0.0323      0.002     16.300      0.000       0.028       0.036
transmission_Manual    -0.0874      0.003    -26.981      0.000      -0.094      -0.081
==============================================================================
Omnibus:                     2665.316   Durbin-Watson:                   2.026
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           442938.191
Skew:                           5.133   Prob(JB):                         0.00
Kurtosis:                      68.093   Cond. No.                         29.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 18: Model 3

6.4.5 Model 4:

- Removing the variable 'transmission_Manual' based on its High p-value

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.264
Model:                            OLS   Adj. R-squared:                  0.263
Method:                 Least Squares   F-statistic:                     292.6
Date:                Sat, 19 Nov 2022   Prob (F-statistic):          2.78e-162
Time:                        03:01:02   Log-Likelihood:                 3801.0
No. Observations:                2448   AIC:                            -7594.
Df Residuals:                    2444   BIC:                            -7571.
Df Model:                           3
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.0845      0.002     35.252      0.000       0.080       0.089
km_driven      -0.1121      0.020     -5.636      0.000      -0.151      -0.073
age            -0.1436      0.008    -18.443      0.000      -0.159      -0.128
fuel_Diesel     0.0335      0.002     14.843      0.000       0.029       0.038
==============================================================================
Omnibus:                     2902.711   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           441682.873
Skew:                           6.067   Prob(JB):                         0.00
Kurtosis:                      67.676   Cond. No.                         23.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 19: Model 4

7 Residual Analysis

We plot the graph to find the error terms of the model w.r.t prediction value of price. The error graph shows the increase in the density and drops down when error is increased. So, at o the density is high and it is distributed normally.

```
In [86]: y_train_pred = lr4.predict(X_train_lm4)

In [87]: res = y_train-y_train_pred
         # Plot the histogram of the error terms
         fig = plt.figure()
         sns.distplot((res), bins = 20)
         fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
         plt.xlabel('Errors', fontsize = 18)

Out[87]: Text(0.5, 0, 'Errors')
```

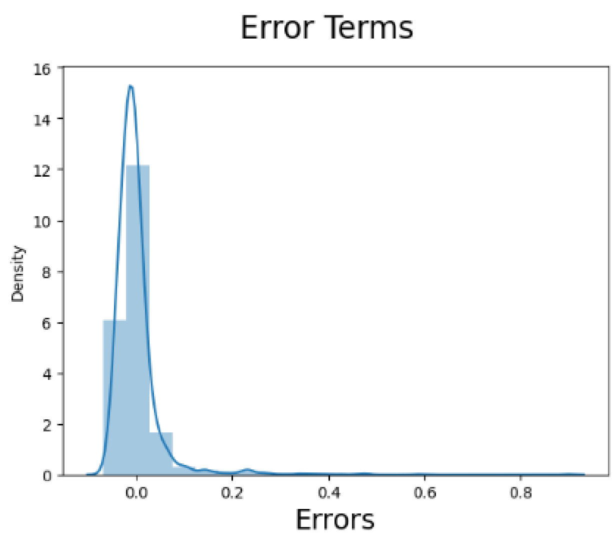


Figure 20: Residual Error

- From the above histogram, we could see that the Residuals are normally distributed. Hence our assumption for Linear Regression is valid.

8 Evolution

- Evaluate the actual price and predicted price with the results obtained by plotting the graph with graphical representation.
- We can observe how the actual and predicted prices has variance we can see a few outliers on the top right with high variance.

MODEL EVALUATION

In [96]:
```
# Plotting y_test and y_pred to understand the spread

fig = plt.figure()
plt.scatter(y_test, y_pred, alpha=.5)
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)
plt.show()
```

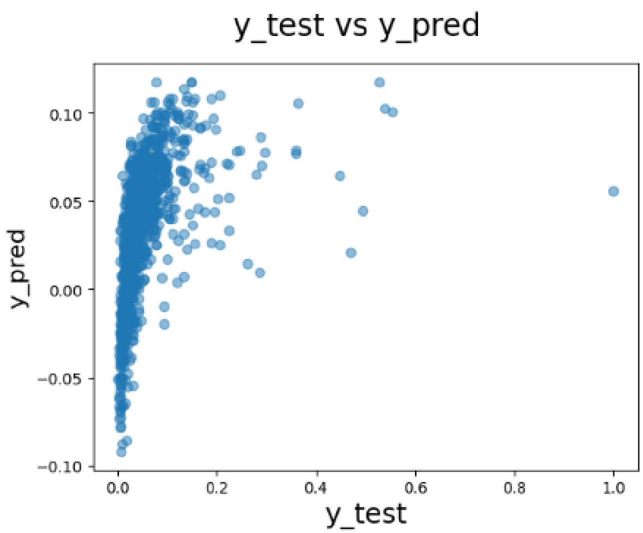


Figure 21: Model Evalution

R^2 Value for TEST

In [97]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[97]: 0.10120230640676553

Adjusted R^2 Value for TEST

In [98]:
```
# We already have the value of R^2 (calculated in above step)

r2=0.3618371256083056
```

In [99]:
```
# Get the shape of X_test
X_test.shape
```

Out[99]: (1050, 3)

In [100]:
```
# n is number of rows in X

n = X_test.shape[0]


# Number of features (predictors, p) is the shape along axis 1
p = X_test.shape[1]

# We find the Adjusted R-squared using the formula

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[100]: 0.36000683055746896

Final Result Comparison

- Train R^2 :0.433
- Train Adjusted R^2 :0.432
- Test R^2 :0.362
- Test Adjusted R^2 :0.360

This seems to be a really good model that can moderate 'Generalize' various datasets.

Figure 22: Adjusted R^2

Final Result Comparison:

- Train R^2 :0.433
- Train Adjusted R^2 :0.432
- Test R^2 :0.362
- Test Adjusted R^2 :0.360

As per our final Model, the top predictor variables that influences the selling_price are:

- km_driven: A coefficient value of '0.081104' indicated that a unit increase in km_driven variable, decreases the selling_price numbers by 0.081104 units.

- age: A coefficient value of '-0.132559' indicated that, a unit increase in age variable, decreases the selling_price numbers by 0.132559 units.

- fuel_Diesel: A coefficient value of '0.032289' indicated that w.r.t Petrol, a unit increase in fuel_Diesel variable increases the selling_price numbers by 0.032289 units.

- transmission_Manual: A coefficient value of '-0.087353' indicated that w.r.t Automatic, a unit increase in transmission_Manual variable decreases the selling_price numbers by 0.087353 units.

```
In [80]: # Print a summary of the linear regression model obtained
         print(lr3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          selling_price   R-squared:                       0.433
Model:                            OLS   Adj. R-squared:                  0.432
Method:                 Least Squares   F-statistic:                     466.7
Date:                Sat, 19 Nov 2022   Prob (F-statistic):          3.70e-299
Time:                        03:01:02   Log-Likelihood:                 4120.3
No. Observations:                2448   AIC:                            -8231.
Df Residuals:                    2443   BIC:                            -8202.
Df Model:                           4
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                   0.1593      0.003     45.771      0.000       0.152       0.166
km_driven              -0.0811      0.018     -4.634      0.000      -0.115      -0.047
age                    -0.1326      0.007    -19.352      0.000      -0.146      -0.119
fuel_Diesel             0.0323      0.002     16.300      0.000       0.028       0.036
transmission_Manual    -0.0874      0.003    -26.981      0.000      -0.094      -0.081
==============================================================================
Omnibus:                     2665.316   Durbin-Watson:                   2.026
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           442938.191
Skew:                           5.133   Prob(JB):                         0.00
Kurtosis:                      68.093   Cond. No.                         29.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 23: Final Summary

9 GitHub Repository

https://github.com/samba-chennamsetty/used-car-selling-price-linear-regression

10 References

[1]https://www.kaggle.com/code/gauravduttakiit/old-car-selling-price-with-linear-regression

[2]https://www.kaggle.com/code/gauravduttakiit/old-car-selling-price-with-linear-regression/data?select=car+data.csv

[3] www.cardekho.com