USED CAR PRICE PREDICTION PROJECT

Submitted By:

ANSHUL DUBEY

The completion of this thesis study would not have been possible without thesupport and extensive knowledge of several website.

I cannot begin without expressing my thanks to my SME from *Flip Robo Technologies* , *Khushboo Garg*, for the valuable experience and insight in conducting and writing scientific reports. Thank you for always answering all questions and giving me helpful practical suggestions.

Someone whose help cannot be overestimated is my advisor from Career Coach Thank you for always taking your time to support and guide me through the project. Without your extensive knowledge and

insightful suggestions, the final result would not be what it is today.

Finally, I would like to express my sincere gratitude to my family for their constant encouragement and support throughout my time on this thesis.

Introduction

Here, we get data from the website www.OLX.com, filled with information on a wide variety of cars, including their selling price,owner,milage,date of buying etc. We realize that we can use this data to make sure we get a good deal on a new car. In particular, we can figure out exactly how much one should pay for a specific type of car.

In this article, I am going to walk you through how we can train a model that will help us predict car prices. you'll practice the machine learning workflow you've learned so far to predict a car's market price using its attributes. The dataset we will be working with contains information on various cars.

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. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model. This project contains two phase-

- Data Collection Phase
- Model Building Phase

The inspiration behind this is I just want to know about the price of the car as well as my idea of doing some useful things. I think this is a limited motivation to make this project better.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem:-

In this project, we will develop and evaluate the performance and predictability of trained and tested models based on

data collected from OLX. Once we get a good fit, we will use this model to predict the monetary value of a car.

In here we will use various regression algorithm to predict our target. Let's have an overview of the algorithms we will use for our predictions. To read more about these algorithms , just click on the algorithms name.

LinearRegression:- Linear-regression models are relatively simple and provide an easy-to-interpret mathematical formula that can generate predictions. Linear regression can be applied to various areas in business and academic study.You'll find that linear regression is used in everything from biological, behavioral, environmental and social sciences to business. Linear-regression models have become a proven way to scientifically and reliably predict the

- future. Because linear regression is a long-established statistical procedure, the properties of linear-regression models are well understood and can be trained very quickly.

- DecisionTreeRegressor:- Decision trees **help you to evaluate your options**. Decision Trees are excellent tools for helping you to choose between several courses of action. They provide a highly effective structure within which you can lay out options and investigate the possible outcomes of choosing those options.

SVR:- The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points. Unlike other Regression models that try to minimize the error between the real and predicted

- value, the SVR tries to fit the best line within a threshold value.

- KNeighborsRegressor:- KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

RandomForestRegressor:- A random forest regressor. A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting***Data Sources and their formats:-*** I have imported the dataset which was in the csv file using pandas. The dataset contains 1168 rows and 81 columns having both numerical and categorical data.

In this dataset " **Car price**" is our target variable which has float values. So this is a "Regression type" problem.

Data Preprocessing Done:-

For the purpose of the project the dataset has been preprocessed as follows:

- Checking missing value
- we divide all the columns into categorical and numerical types
- Univariate Analysis Of Categorical Columns and numerical columns.
- Checking correlation with target column using Heatmap
- Checking skewness of each columns
- Handling missing value
- Checking Outliers and removing outliers by zscore
- Encoding the categorical column by Label Encoder
- Dividing data into features and vectors
- Checking VIF score
- Removing skewness by power transform method
- Standardizing the data by standard scaler

We'll now open a python 3 Jupyter Notebook and execute the following code snippet to load the dataset and remove the non-essential features. Recieving a

success message if the actions were correclty performed.

Hardware and Software Requirements and Tools Used:-

- ***Hardware:-*** ***Desktop/Laptop***
- ***Software:-*** ***Anaconda***
- ***Libraries:-***

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import StandardScaler
from scipy.stats import zscore
from matplotlib import rcParams
from sklearn.metrics import plot_confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn import linear_model
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split

from sklearn.linear_model import Ridge, Lasso

from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
```

Model/s Development and Evaluation

- ***Identification of possible problem-solving approaches (methods):***
  - ***Missing Value Handling:***

There are 2 primary ways of handling missing values:

- Deleting the Missing values:-Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values.

- Imputing the Missing Values:- There are different ways of replacing the missing values
  - ✓ Replacing With Mean
  - ✓ Replacing With Mode
  - ✓ Replacing With Median,etc.

  For my dataset,I will use mode for categorical features and median for numerical features by the code below.

```
p=[x for x in df.columns if df[x].isnull().sum()!=0]
```

➢ ***Outliers Checking:***

Identification of potential outliers is important for the following reasons.

- ✓ An outlier may indicate bad data. For example, the data may have been coded incorrectly or an experiment may not have been run correctly. If it can be determined that an outlying point is in fact erroneous, then the outlying value should be deleted from the analysis (or corrected if possible).

- ✓ In some cases, it may not be possible to determine if an outlying point is bad data. Outliers may be due to random variation or may indicate something scientifically interesting. In any event, we typically do not want to simply delete the outlying observation. However, if the data contains significant outliers, we may need to consider the use of robust statistical techniques.

Boxplots, histograms, and scatterplots can highlight outliers.

Boxplots display asterisks or other symbols on the graph to indicate explicitly when datasets contain outliers. These graphs use the interquartile method with fences to find outliers.

➢ ***Outliers Removing:***

There are many ways to detect and remove Outliers.Here I have used Z-score function defined in scipy library to detect the outliers.

The formula for calculating a z-score is is $z = (x-\mu)/\sigma$, where x is the raw score, μ is the population mean, and σ is the population standard deviation.

We have created a new dataframe df_new where no outliers present.Let's see the code

```
#import zscore
from scipy.stats import zscore
```

```
z=np.abs(zscore(df[outliers]))
z
```

➢ ***Label Encoding:***

Label Encoding refers to converting the labels into a numeric form so as **to convert them into the**

machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

I have applied label encoding method to our cleaned dataframe df_new and converted the categorical columns into numerical

```
# checking for categorical columns
categorical_columns=[]
for i in df_new.dtypes.index:
    if df_new.dtypes[i]=='object':
        categorical_columns.append(i)
print(categorical_columns)
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in list_c:
    df_new[i]=le.fit_transform(df_new[i]).astype(int)
```

- ***Dividing Data In Features And Vectors:***

We have to divide our dataset columns into X and Y. X variables so that we could have all the attribute columns and our label / target variable with the Y variable.

```
x=df_new.drop("car_price",axis=1) #Independent variable
y=df_new.iloc[:,1] #Dependent variable
```

- ***Multicolinearity Checking:***

Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model. This means that an independent variable can be predicted from another independent variable in a regression model.

The best way to identify the multicollinearity is to calculate the Variance Inflation Factor (VIF) corresponding to every independent Variable in the

Dataset. VIF tells us about how well an independent variable is predictable using the other independent variables.

Let's see the code how to get VIF score:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
#define a function to calculate VIF score
def vif_clac():
    vif=pd.DataFrame()
    vif["VIF Factor"]=[variance_inflation_factor(p.values,i) for i in range(p.shape[1])]
    vif["features"]=p.columns
    print(vif)
```

```
#checking VIF score
vif_clac()
```

- ## Skewness Removing:

  If there are too much skewness in the data, then many statistical model don't work .In skewed data, the tail region may act as an outlier for the statistical model and we know that **outliers adversely affect the model's performance especially regression-based models**.

  I used power transformation(method= yeo-johnson) method to remove skewness of the independent dataset.

```
from sklearn.preprocessing import power_transform
x=power_transform(x,method='yeo-johnson')
x
```

```
array([[ 1.35033987, -0.15606345, -0.33834441, ..., -0.60440412,
         0.39752544,  0.02382018],
       [ 0.48377075, -0.15606345,  1.38612866, ..., -0.60440412,
         0.39752544,  0.02382018],
       [-1.16083124, -0.15606345,  2.05610743, ...,  1.63482104,
        -2.7775254 ,  0.02382018],
       ...,
       [ 1.6741358 , -0.15606345, -2.48927497, ...,  0.88976533,
         0.39752544,  0.02382018],
       [ 0.68749385, -2.8818269 , -0.90638229, ...,  0.14341105,
         0.39752544,  0.02382018],
       [ 0.48377075, -0.15606345, -0.33834441, ..., -1.35359903,
         0.39752544,  0.02382018]])
```

```
#print skewness after power transform
df_new1=pd.DataFrame(x,columns=df_new.drop("SalePrice",axis=1).columns)
df_new1.skew()
```

➢ ***Data Standardizing:***

Data standardization is a data processing workflow that converts the Data standardization helps improve the quality of your data by transforming and standardizing it. Think of it like a uniform for your databases. By taking this step, you are formatting your records in a way that creates consistency across your systems and makes it easy for businesses to use.

Now let's see how I did standardize the data:

structure of different datasets into one common format of data.

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
x
```

```
array([[ 1.35033987, -0.15606345, -0.33834441, ..., -0.60437069,
         0.39752544,  0.02382018],
       [ 0.48377075, -0.15606345,  1.38612866, ..., -0.60437069,
         0.39752544,  0.02382018],
       [-1.16083124, -0.15606345,  2.05610743, ...,  1.63485447,
        -2.7775254 ,  0.02382018],
       ...,
       [ 1.6741358 , -0.15606345, -2.48927497, ...,  0.88979876,
         0.39752544,  0.02382018],
       [ 0.68749385, -2.8818269 , -0.90638229, ...,  0.14344448,
         0.39752544,  0.02382018],
       [ 0.48377075, -0.15606345, -0.33834441, ..., -1.3535656 ,
         0.39752544,  0.02382018]])
```

- ***Testing of Identified Approaches (Algorithms):-***

➢ *Finding Best Random State:-*

An algorithm might have multiple points that introduce randomness to the process and thus introduce randomness to the result. One method to make sure our results are constant is to set every possible *random_state* available in the functions that we use. We can find the best random state by the code below.

After we have found the value for best random state, we proceeded with the train test split function to create new training and testing datasets and fit them into the models to find our ideal models.

➢ *Train_Test_Split:-*

The Train_Test_Split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

```
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=.30,random_state=233)
```

```
#cheking shape of all variable
print("train_x shape =",train_x.shape)
print("test_x shape =",test_x.shape)
print("train_y shape =",train_y.shape)
print("test_y shape =",test_y.shape)
```

Now I will be using 13 differents regression algorithms to get the ideal one for prediction.

```
LR_model= LinearRegression()
RD_model= Ridge()
LS_model= Lasso()
DT_model= DecisionTreeRegressor()
SV_model= SVR()
KNR_model= KNeighborsRegressor()
RFR_model= RandomForestRegressor()
XGB_model= XGBRegressor()
Elastic_model= ElasticNet()
SGH_model= SGDRegressor()
Bag_model=BaggingRegressor()
ADA_model=AdaBoostRegressor()
GB_model= GradientBoostingRegressor()
```

```
model=[LR_model,RD_model,LS_model,DT_model,SV_model,KNR_model,RFR_model,XGB_model,Elastic_model,SGH_model,Bag_model,ADA_model,GB_
```

- ***<u>Run and Evaluate selected models:-</u>***

```
for i in model:
    p=i
    p.fit(train_x,train_y)
    print("accuracy score of",i,"is =",p.score(train_x,train_y))

    print("\n")
    print("#"*50)
```

```
accuracy score of LinearRegression() is = 0.6452931539854956
```

```
#####################################################
accuracy score of Ridge() is = 0.6452931277823244


#####################################################
accuracy score of Lasso() is = 0.6452931539375557


#####################################################
accuracy score of DecisionTreeRegressor() is = 0.9999916527197423


#####################################################
accuracy score of SVR() is = -0.06459263507374624


#####################################################
accuracy score of KNeighborsRegressor() is = 0.8702798291992135


#####################################################
accuracy score of RandomForestRegressor() is = 0.9852526380106977
#####################################################
accuracy score of XGBRegressor(base_score=0.5, booster='gbtree', cal
lbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytr
ee=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='dept
hwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_oneh
ot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_
weight=1,
             missing=nan, monotone_constraints='()', n_estimators=10
0, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0,
             reg_lambda=1, ...) is = 0.9942845188810525


#####################################################
accuracy score of ElasticNet() is = 0.598853842807132


#####################################################
accuracy score of SGDRegressor() is = 0.6439521543615279


#####################################################
accuracy score of BaggingRegressor() is = 0.9765825460543547
```

```
##################################################
accuracy score of AdaBoostRegressor() is = 0.656591588244267


##################################################
accuracy score of GradientBoostingRegressor() is = 0.908073849551413
```

we get best accuracy score from XGBRegressor, RandomForestRegressor, DecisionTreeRegressor

- ***Cross Validation Score:***

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

Now we check cross val score of the above models by the code below:

```
from sklearn.model_selection import cross_val_score
for i in range(2,10):
    print("For CV =",i)
    for m in model:
        scr=cross_val_score(m,x,y,cv=i)
        print("cross validation score of",m,"is =",scr.mean())
    print('*'*100)
```

- *After running this code we get best cross val score from*

- ***Hyperparameter tuning of GradientBoostingRegressor:-***

GradientBoostingRegressor & RandomForestRegressor for CV=8

```
#creating parameter list to pass in GreadSearchCV
parameters={'max_features':['auto','sqrt','log2'],'max_depth':[3,4,5,6,7,8]}
gcv1=GridSearchCV(RandomForestRegressor(),parameters,cv=8,scoring='accuracy')
gcv1.fit(train_x,train_y)
gcv1.best_params_
```

```
{'max_depth': 3, 'max_features': 'auto'}
```

```
gcv1.best_estimator_
```

```
RandomForestRegressor(max_depth=3)
```

```
gcv1.best_estimator_.fit(test_x,test_y)
print("Test Accuracy=",gcv1.best_estimator_.score(test_x,test_y))
```

```
Test Accuracy= 0.7163954631876809
```

```
accuracy score is not increased after parameter tuning.
```

Parameter Tuning:GradientBoostingRegressor

```
#creating parameter list to pass in GreadSearchCV
parameters={'max_features':['auto','sqrt','log2'],'max_depth':[3,4,5,6,7,8],'learning_rate':[1,0.1,0.001,0.0001]}
gcv1=GridSearchCV(GradientBoostingRegressor(),parameters,cv=3,scoring='accuracy')
gcv1.fit(train_x,train_y)
gcv1.best_params_
```

```
{'learning_rate': 1, 'max_depth': 3, 'max_features': 'auto'}
```

```
gcv1.best_estimator_
```

```
GradientBoostingRegressor(learning_rate=1, max_features='auto')
```

```
gcv1.best_estimator_.fit(test_x,test_y)
print("Test Accuracy=",gcv1.best_estimator_.score(test_x,test_y))
```

```
Test Accuracy= 0.9869274604337335
```

accuracy score has increased after parameter tuning of GradientBoostingRegressor.Now we will create final model with this parameter.

```
final_model=GradientBoostingRegressor(learning_rate=1, max_features='auto')
```

```
final_model.fit(test_x,test_y)
pred=final_model.predict(test_x)
print("Test Accuracy=",final_model.score(test_x,test_y))
```

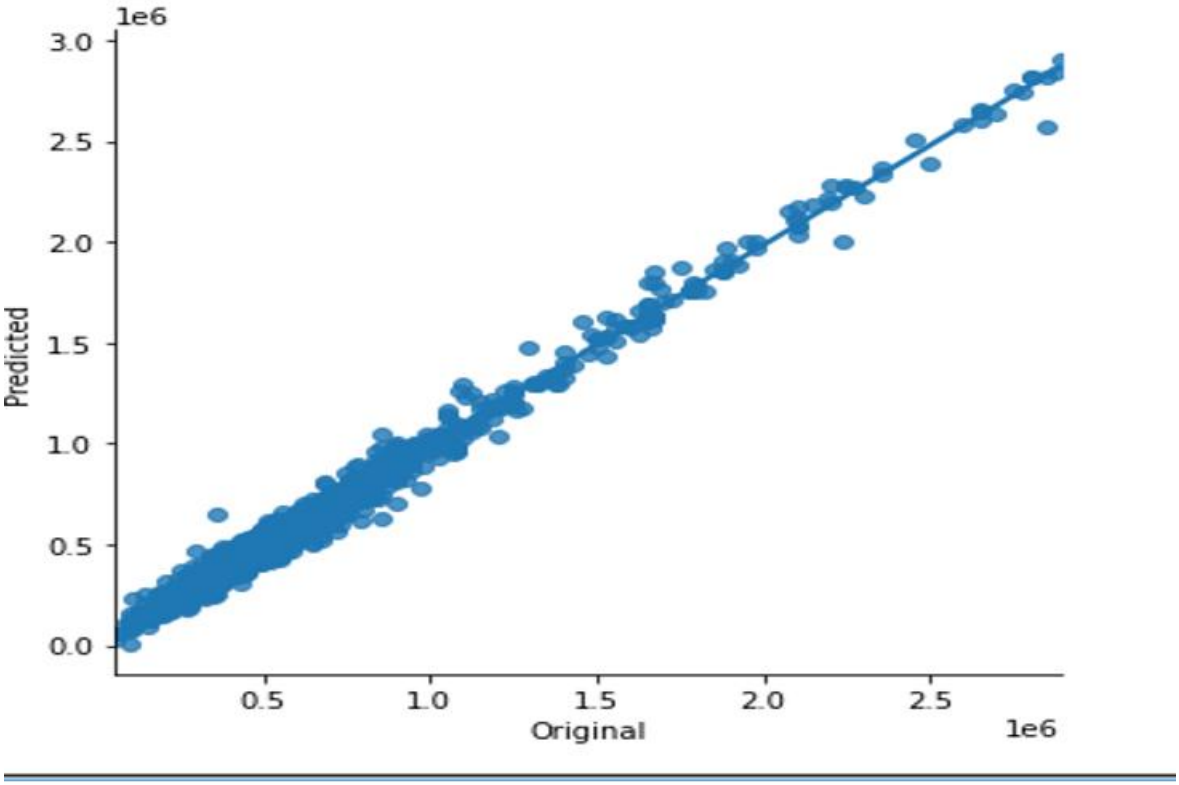
```
Test Accuracy= 0.9999991071285338
```

```
final_model.fit(train_x,train_y)
pred1=final_model.predict(train_x)
print("Train Accuracy=",final_model.score(train_x,train_y))
```

```
Train Accuracy= 0.9995740666077201
```

since difference between train accuracy and test accuracy is very less ,our model is not overfit or underfit.

- ***Visualizations:-***



We can see that the original old car price and final model's predicted price are almost same which means model performance is good

- ***Interpretation of the Results:-***

  This particular problem needs a good vision on data, and in this problem Feature Engineering is the most crucial thing.

  You can see how we have handled numerical and categorical data and how we build different machine learning models on the same dataset.

It is always advised to all of us that atleast we need to use 5 Algorithm in order to figure out which one is performing best among them and we choose that one and we send that for hyper parameter tuning to know that best parameter .

Using hyper parameter tunning we can improve our model accuracy, for instance in this model the accuracy increased.

CONCLUSION

- When predicting old car prices, many complications were involved. There are many variables / attributes to consider in determining the price of a old car and if all or most of them are used, we need a lot of calculating power to get the price of the old car. And it is very difficult to accurately predict old car prices in real life.

- For any of machine learning project my suggestion is first you have to understand the problem on ground level .if you don't allow yourself to work with diligence .if you don' t work harder anything that you are doing or will do , not only in case of machine learning but also in life cycle would be futile. Maybe, my endeavour assist you when ever you will get stuck.

- For future improvements, following step we thought to took-

✓ Replacing model with a latest/different model

✓ Using other robust datasets

✓ Predicting result on differents attributes

THANKYOU