FLIP ROBO

USED CAR PRICE PREDICTION

Submitted by:

TANUJ

SWARNKAR

ACKNOWLEDGMENT

www.cardekho .com

INTRODUCTION

Business Problem Framing

Due to pandemic situation in (2019 to 2021), we have seen abundant changes in the car sales. 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.

Conceptual Background of the Domain Problem

Even before coronavirus had paralyzed the country, personal mobility had become vital to people in India. Urbanization and connectivity had a significant effect on the way we perceived cars, creating a niche in the market for used cars. Today, with a population of 136.64 Cr, India pushes approximately 2.9 million used cars into the market annually.

DATASET INFORMATION

This dataset consists of information about used car listed on cardekho.com especially in Chennai location. It has 16 columns and 828 rows

The features are

Name: It gives information about car name

Year: The year in which car has been purchased.

Driven_kilometers: Number of kilometers car has been driven.

Num_of_owners: The details about cars owners whether the car is owned by single or multiple owners

Fuel: This feature mention the fuel type of car (LPG, petrol, diesel etc.).

Transmission: It gives information about the whether the car is automatic and manual.

Location: Gives the location of the car

Company: the company from which car is built

Price1: price of the car mention by the owner / dealer.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem

- Identifying the datatypes of the dataset .This data set has all object type features
- Name', ' Driven_kilometers ',' Num_of_owners ','Fuel', 'Transmission', 'Location', 'Company', 'Year', 'Price1' are object type features
- Handling null values
- The features Fuel, Driven_kilometers, Num_of_owners, Transmission, Location Having null values
- The features having null values, I have used mode value to replace null value. The features having null values are string type so I have replaced with mode value
- Analyzing each feature and identified feature having unique value
- Univariate analysis using countplot
- The impact of price on object features are analyzed using barplot
- The impact of price on ordinal features are visualized using regplot
- To find the correlation between each feature and target feature I have used correlation matrix
- Outliers in continuous variable price is identified and removed using zscore
- Feature scaling is done using Minmaxscalar
- Converting string value to numeric value using label encoder
- Removed column Unnamed: 0
- It is a regression problem

Data Sources and their formats

- This dataset is in excel format.
- The dataset has 5826 rows and 10 columns

Data Preprocessing Done

- Importing libraries
- Import the dataset
- Handling null values
- The features Fuel, Driven_kilometers, Num_of_owners, Transmission, Location having null values
- For features having null values, I have used mode value to replace null value
  The features having null values are string type so I have replaced with mode value
- Analyzing each feature and identified feature having unique value
- Univariate analysis using countplot
- The feature Unnamed: 0 is of no use so I have dropped that column
- Univariate Analysis using count plot and value count() method
- Multivariate analysis using correlation matrix
- Outliers are identified and removed using zscore
- Encoding the categorical data to numerical data using label encoder
- Feature scaling

❖ If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values so I have used Min_max scalar to covert un scaled data to scaled data.

Data Inputs- Logic- Output Relationships

OUTPUT FEATURE:

The output feature is price1 .It is a continuous variable so it is a regression problem

INPUT FEATURES:

Unnamed: 0, Fuel, Driven_kilometers, Num_of_owners, Transmission, Location, name, Year, Company are input features

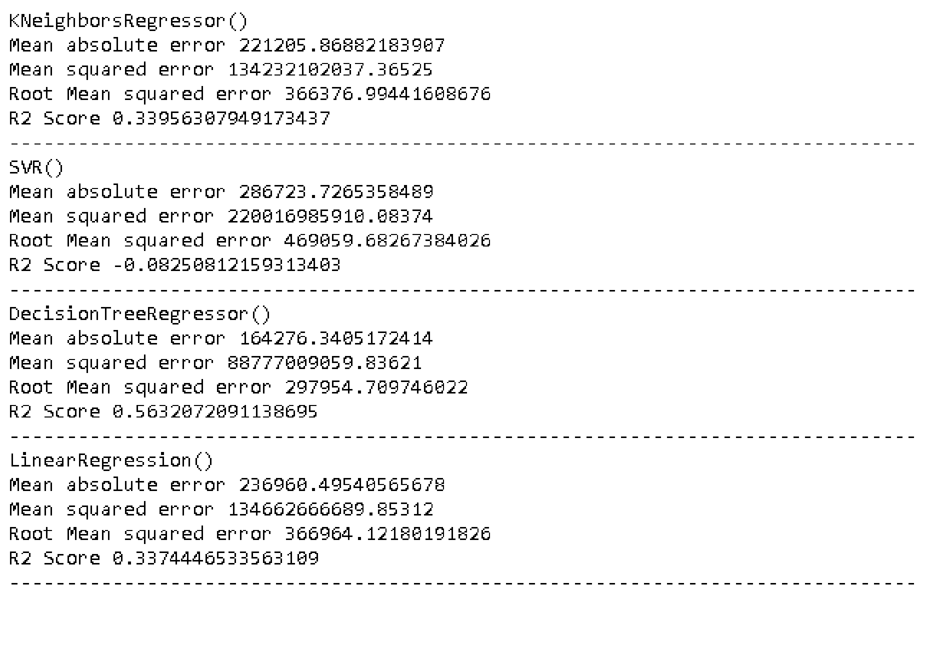
MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches (methods)

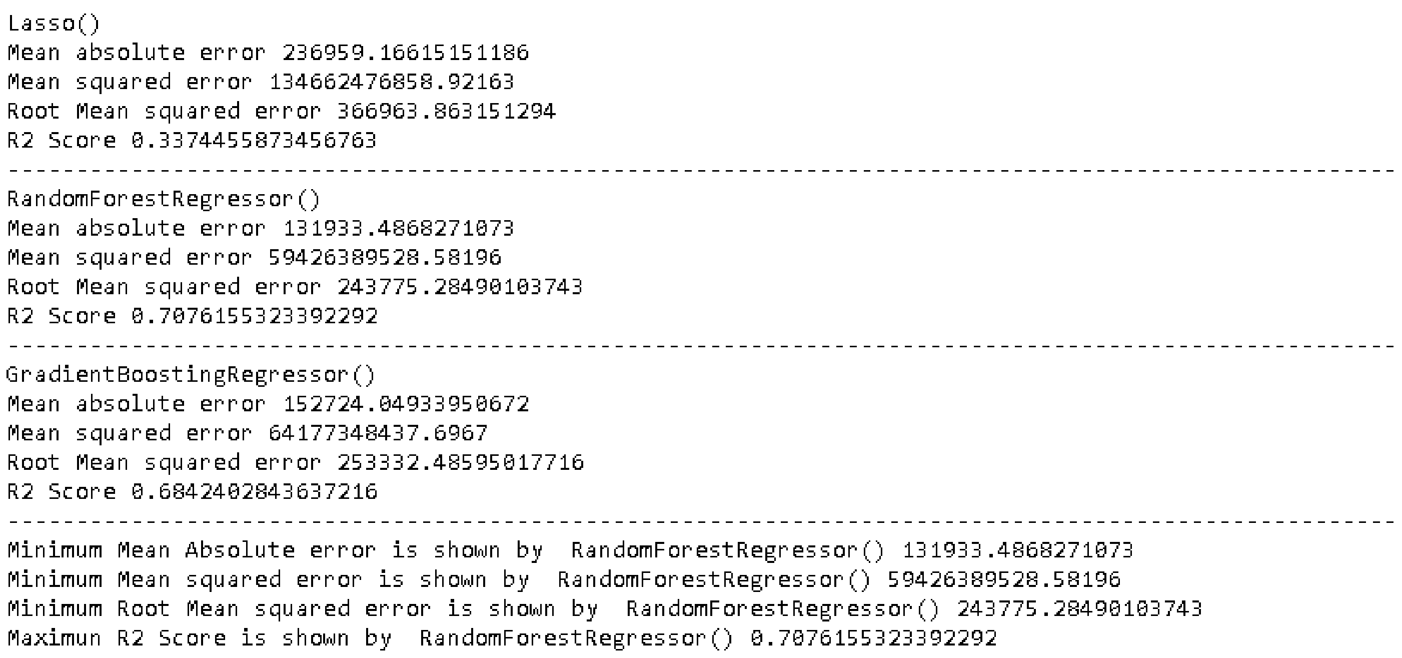
It is a regression problem so I have used 7 modelling algorithm

- KNeighborsRegressor()
- SVR()
- DecisionTreeRegressor()
- LinearRegression()
- Lasso()
- RandomForestRegressor()
- GradientBoostingRegressor()

Run and Evaluate selected models



```
KNeighborsRegressor()
Mean absolute error 221205.86882183907
Mean squared error 134232102037.36525
Root Mean squared error 366376.99441608676
R2 Score 0.33956307949173437
-----------------------------------------------------------------------------
SVR()
Mean absolute error 286723.7265358489
Mean squared error 220016985910.08374
Root Mean squared error 469059.68267384026
R2 Score -0.08250812159313403
-----------------------------------------------------------------------------
DecisionTreeRegressor()
Mean absolute error 164276.3405172414
Mean squared error 88777009059.83621
Root Mean squared error 297954.709746022
R2 Score 0.5632072091138695
-----------------------------------------------------------------------------
LinearRegression()
Mean absolute error 236960.49540565678
Mean squared error 134662666689.85312
Root Mean squared error 366964.12180191826
R2 Score 0.3374446533563109
-----------------------------------------------------------------------------
```



```
Lasso()
Mean absolute error 236959.16615151186
Mean squared error 134662476858.92163
Root Mean squared error 366963.863151294
R2 Score 0.3374455873456763
------------------------------------------------------------------------------------------------------------
RandomForestRegressor()
Mean absolute error 131933.4868271073
Mean squared error 59426389528.58196
Root Mean squared error 243775.28490103743
R2 Score 0.7076155323392292
------------------------------------------------------------------------------------------------------------
GradientBoostingRegressor()
Mean absolute error 152724.04933950672
Mean squared error 64177348437.6967
Root Mean squared error 253332.48595017716
R2 Score 0.6842402843637216
------------------------------------------------------------------------------------------------------------
Minimum Mean Absolute error is shown by  RandomForestRegressor() 131933.4868271073
Minimum Mean squared error is shown by  RandomForestRegressor() 59426389528.58196
Minimum Root Mean squared error is shown by  RandomForestRegressor() 243775.28490103743
Maximun R2 Score is shown by  RandomForestRegressor() 0.7076155323392292
```

<u>KEY METRICS</u>

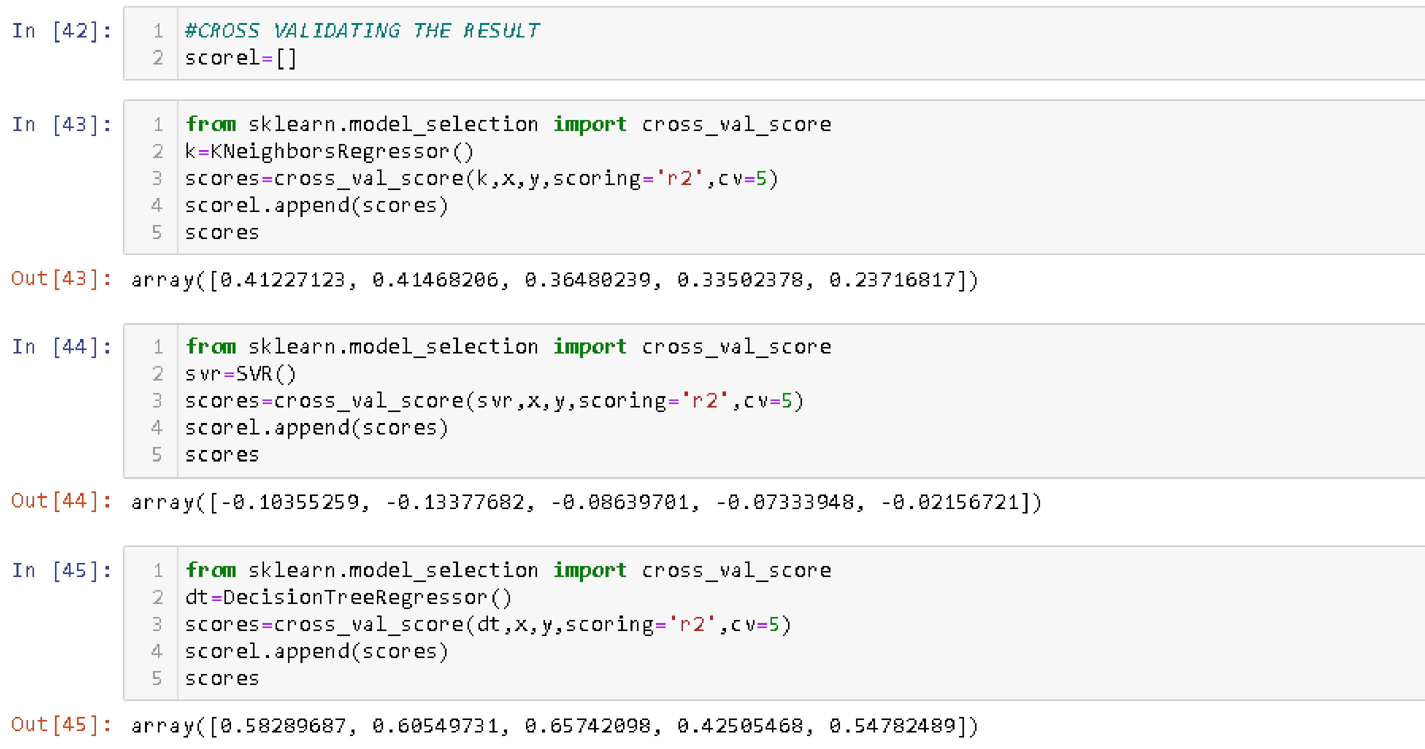
1. R-squared (R2), which is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R-squared, the better the model.

2. Root Mean Squared Error (RMSE), which measures the average error performed by the model in predicting the outcome for an observation. Mathematically, the RMSE is the square root of the *mean squared error (MSE)*, which is the average squared difference between the observed actual outcome values and the values predicted by the model. So, MSE = mean ((observed - predicted) ^2) and RMSE = sqrt (MSE). The lower the RMSE, the better the model.

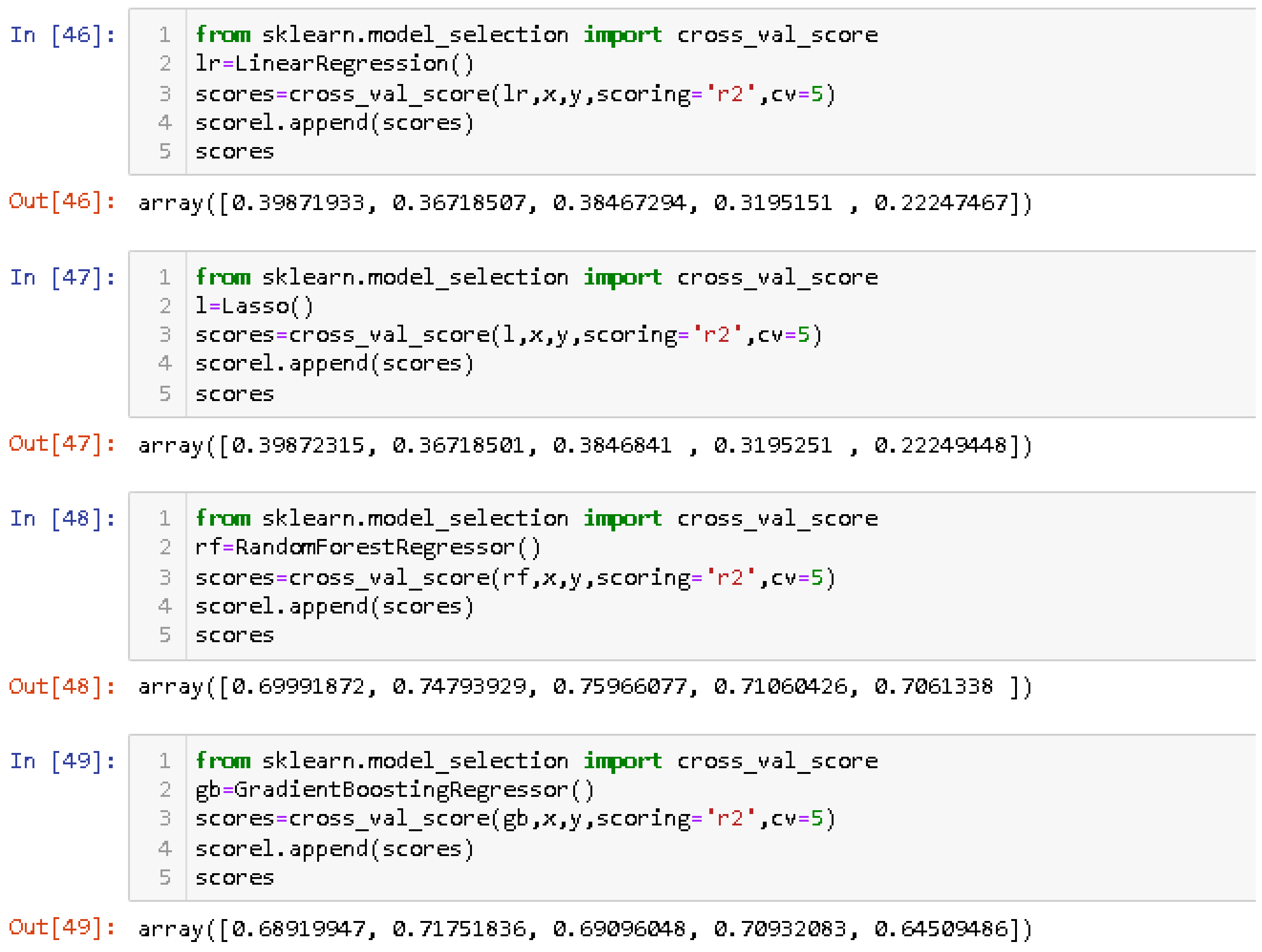
3. Residual Standard Error (RSE), also known as the *model sigma*, is a variant of the RMSE adjusted for the number of predictors in the model. The lower the RSE, the better the model. In practice, the difference between RMSE and RSE is very small, particularly for large multivariate data.

4. Mean Absolute Error (MAE), like the RMSE, the MAE measures the prediction error. Mathematically, it is the average absolute difference between observed and predicted outcomes, MAE = mean (abs (observed - predicted)). MAE is less sensitive to outliers compared to RMSE.

CROSS VALIDATION SCORE

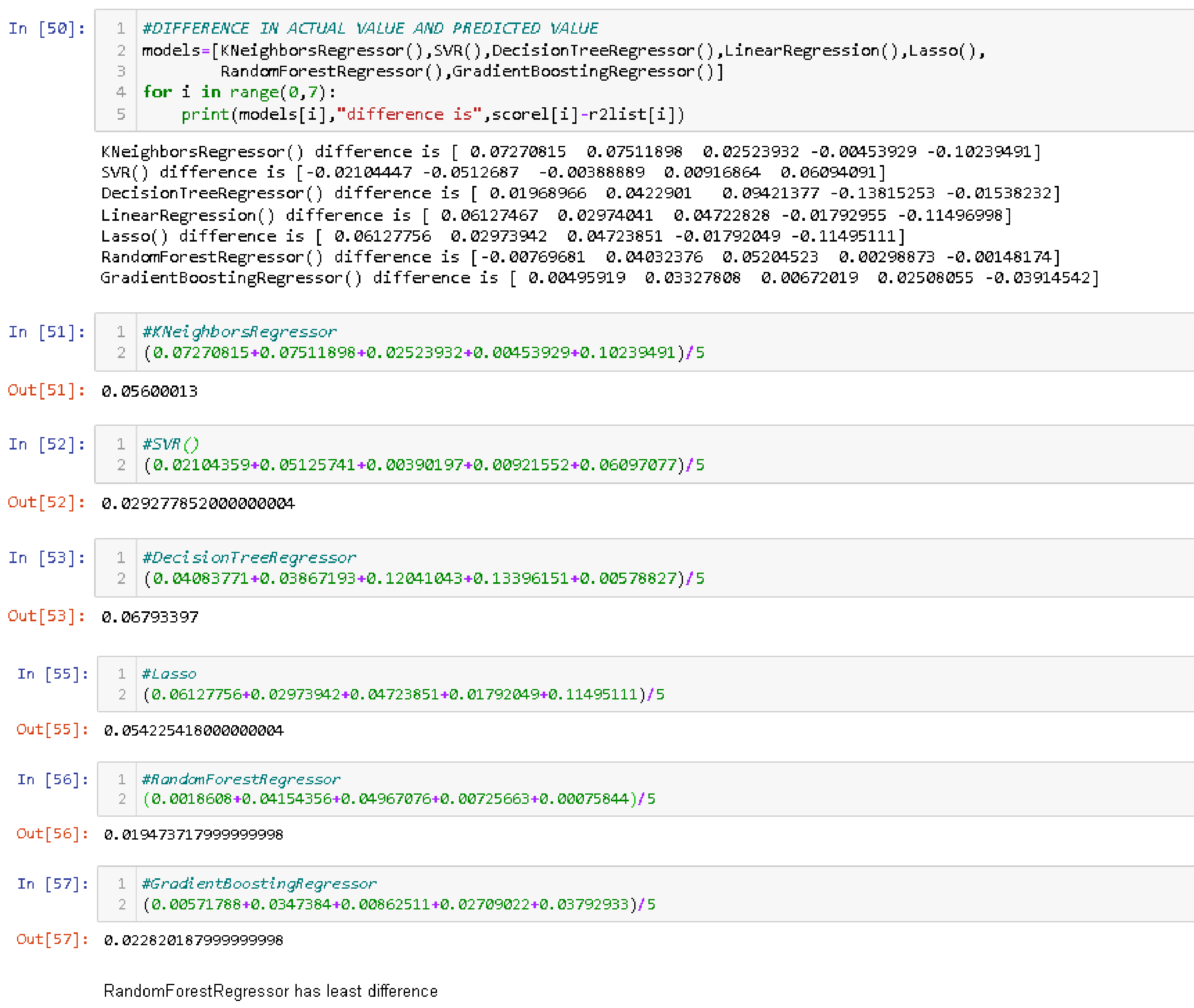


```
In [42]: #CROSS VALIDATING THE RESULT
         scorel=[]

In [43]: from sklearn.model_selection import cross_val_score
         k=KNeighborsRegressor()
         scores=cross_val_score(k,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[43]: array([0.41227123, 0.41468206, 0.36480239, 0.33502378, 0.23716817])

In [44]: from sklearn.model_selection import cross_val_score
         svr=SVR()
         scores=cross_val_score(svr,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[44]: array([-0.10355259, -0.13377682, -0.08639701, -0.07333948, -0.02156721])

In [45]: from sklearn.model_selection import cross_val_score
         dt=DecisionTreeRegressor()
         scores=cross_val_score(dt,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[45]: array([0.58289687, 0.60549731, 0.65742098, 0.42505468, 0.54782489])
```



```
In [46]: from sklearn.model_selection import cross_val_score
         lr=LinearRegression()
         scores=cross_val_score(lr,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[46]: array([0.39871933, 0.36718507, 0.38467294, 0.3195151 , 0.22247467])

In [47]: from sklearn.model_selection import cross_val_score
         l=Lasso()
         scores=cross_val_score(l,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[47]: array([0.39872315, 0.36718501, 0.3846841 , 0.3195251 , 0.22249448])

In [48]: from sklearn.model_selection import cross_val_score
         rf=RandomForestRegressor()
         scores=cross_val_score(rf,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[48]: array([0.69991872, 0.74793929, 0.75966077, 0.71060426, 0.7061338 ])

In [49]: from sklearn.model_selection import cross_val_score
         gb=GradientBoostingRegressor()
         scores=cross_val_score(gb,x,y,scoring='r2',cv=5)
         scorel.append(scores)
         scores

Out[49]: array([0.68919947, 0.71751836, 0.69096048, 0.70932083, 0.64509486])
```

DIFFERENCE OF PREDICTED MODEL AND CROSS VALIDATION SCORE



In [50]:
```python
#DIFFERENCE IN ACTUAL VALUE AND PREDICTED VALUE
models=[KNeighborsRegressor(),SVR(),DecisionTreeRegressor(),LinearRegression(),Lasso(),
        RandomForestRegressor(),GradientBoostingRegressor()]
for i in range(0,7):
    print(models[i],"difference is",score1[i]-r2list[i])
```

```
KNeighborsRegressor() difference is [ 0.07270815  0.07511898  0.02523932 -0.00453929 -0.10239491]
SVR() difference is [-0.02104447 -0.0512687  -0.00388889  0.00916864  0.06094091]
DecisionTreeRegressor() difference is [ 0.01968966  0.0422901   0.09421377 -0.13815253 -0.01538232]
LinearRegression() difference is [ 0.06127467  0.02974041  0.04722828 -0.01792955 -0.11496998]
Lasso() difference is [ 0.06127756  0.02973942  0.04723851 -0.01792049 -0.11495111]
RandomForestRegressor() difference is [-0.00769681  0.04032376  0.05204523  0.00298873 -0.00148174]
GradientBoostingRegressor() difference is [ 0.00495919  0.03327808  0.00672019  0.02508055 -0.03914542]
```

In [51]:
```python
#KNeighborsRegressor
(0.07270815+0.07511898+0.02523932+0.00453929+0.10239491)/5
```

Out[51]:
```
0.05600013
```

In [52]:
```python
#SVR()
(0.02104359+0.05125741+0.00390197+0.00921552+0.06097077)/5
```

Out[52]:
```
0.029277852000000004
```

In [53]:
```python
#DecisionTreeRegressor
(0.04083771+0.03867193+0.12041043+0.13396151+0.00578827)/5
```

Out[53]:
```
0.06793397
```

In [55]:
```python
#Lasso
(0.06127756+0.02973942+0.04723851+0.01792049+0.11495111)/5
```

Out[55]:
```
0.054225418000000004
```

In [56]:
```python
#RandomForestRegressor
(0.0018608+0.04154356+0.04967076+0.00725663+0.00075844)/5
```

Out[56]:
```
0.019473717999999998
```

In [57]:
```python
#GradientBoostingRegressor
(0.00571788+0.0347384+0.00862511+0.02709022+0.03792933)/5
```

Out[57]:
```
0.022820187999999998
```

RandomForestRegressor has least difference

RandomForest regression have least difference

MODELING USING BEST PARAMETERS

```
In [58]:
#HYPERPARAMETER TO GET BEST PARAMETER
from sklearn.model_selection import GridSearchCV
parameters = { "learning_rate": [0.01, 0.025, 0.05, 0.1, 0.2],
    "min_samples_split": np.linspace(0.1, 0.5, 1),
    "min_samples_leaf": np.linspace(0.1, 0.5, 1),
    "max_depth":[3,5,8],
    "max_features":["log2","sqrt"],
    "subsample":[0.5, 0.6, 0.8, 1.0],
    "n_estimators":[10,100,1000]}
grid = GridSearchCV(GradientBoostingRegressor(), param_grid = parameters, cv = 5, scoring = "r2")
```

```
In [59]:
from sklearn.model_selection import GridSearchCV
parameters = {'max_features': ['auto', 'sqrt','log2'],'n_estimators': [10, 100, 200, 500],'bootstrap':[True],'max_depth': [
grid = GridSearchCV(RandomForestRegressor(), param_grid = parameters, cv = 5, scoring = "r2")
```

```
In [60]:
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=1)
grid.fit(xtrain,ytrain)

print("Best_parameters",grid.best_params_)
```

```
Best_parameters {'bootstrap': True, 'max_depth': 45, 'max_features': 'log2', 'min_samples_split': 5, 'n_estimators': 500}
```

```
In [61]:
x=scaled
y=y1
```

```
In [62]:
#TUNING THE MODEL WITH BEST PARAMETERS
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=7)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,accuracy_score
model=RandomForestRegressor(n_estimators=500,max_features='auto',bootstrap=True,max_depth=23,min_samples_split=5)
model.fit(xtrain,ytrain)
p=model.predict(xtest)
acc=model.score(xtest,ytest)
mae=mean_absolute_error(p,ytest)
mse=mean_squared_error(p,ytest)
rmse=np.sqrt(mean_squared_error(p,ytest))
r2=r2_score(ytest,p)
print('Accuracy',(round(acc,2))*100)
print('Mean absolute error',mae)
print('Mean squared error',mse)
print('Root Mean squared error',rmse)
print('r2 score',(round(r2,2))*100)
```

```
Accuracy 72.0
Mean absolute error 140663.60207107532
Mean squared error 64585326210.1786
Root Mean squared error 254136.43227640266
r2 score 72.0
```

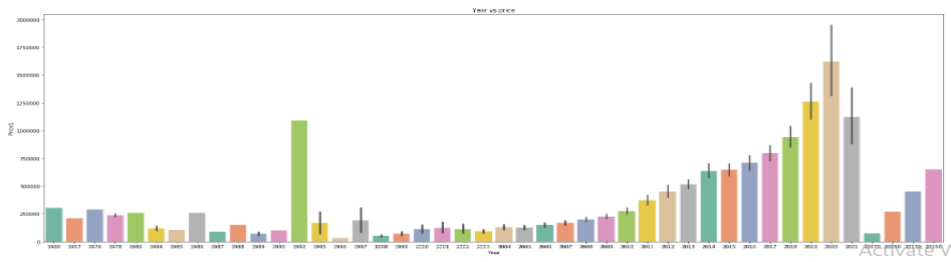
After hyper tuning accuracy is increased to 72%

Activate
Go to Set

VISUALIZATIONS

```
In [25]:
#PLOT TO COMPARE YEAR WITH PRICE OF CARS
plt.figure(figsize=(30,10))
plt.title('Year vs price')
sns.barplot(x=df["Year"],y=df["Price1"],palette="Set2")
```

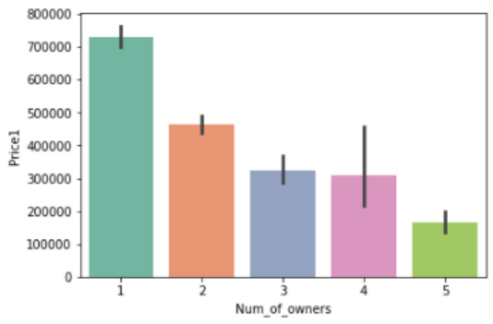
```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x22b28e96d48>
```



Activate Win
Go to Settings to

Recently bought car has high price compare to others

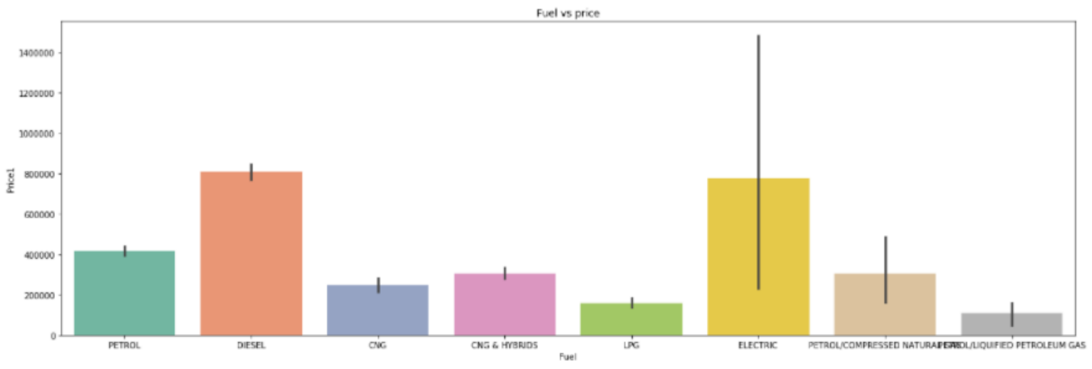
```
In [27]: #PLOT TO CHECK NO. OF OWNERS TO PRICE OF CAR
         sns.barplot(x=df["Num_of_owners"],y=df["Price1"],palette="Set2")

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x22b28e84bc8>
```



Act
Go t

if the car is from first owner it has high value

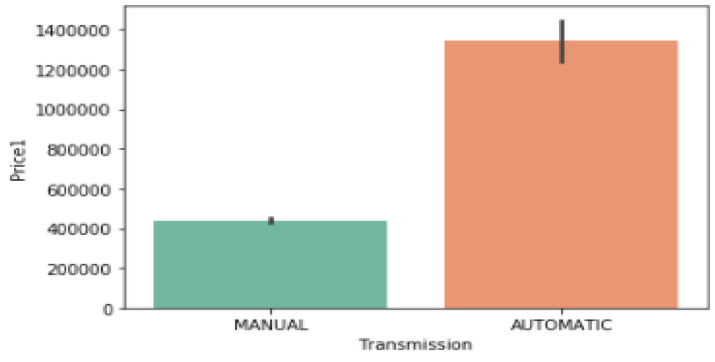
```
In [28]: #PLOT TO COMPARE TYPE OF FUEL TO PRICE OF CAR
         plt.figure(figsize=(22,7))
         plt.title('Fuel vs price')
         sns.barplot(x=df["Fuel"],y=df["Price1"],palette="Set2")

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x22b28e843c8>
```



Diesel cars and electric cars are high pricey than other fuel emissioned cars

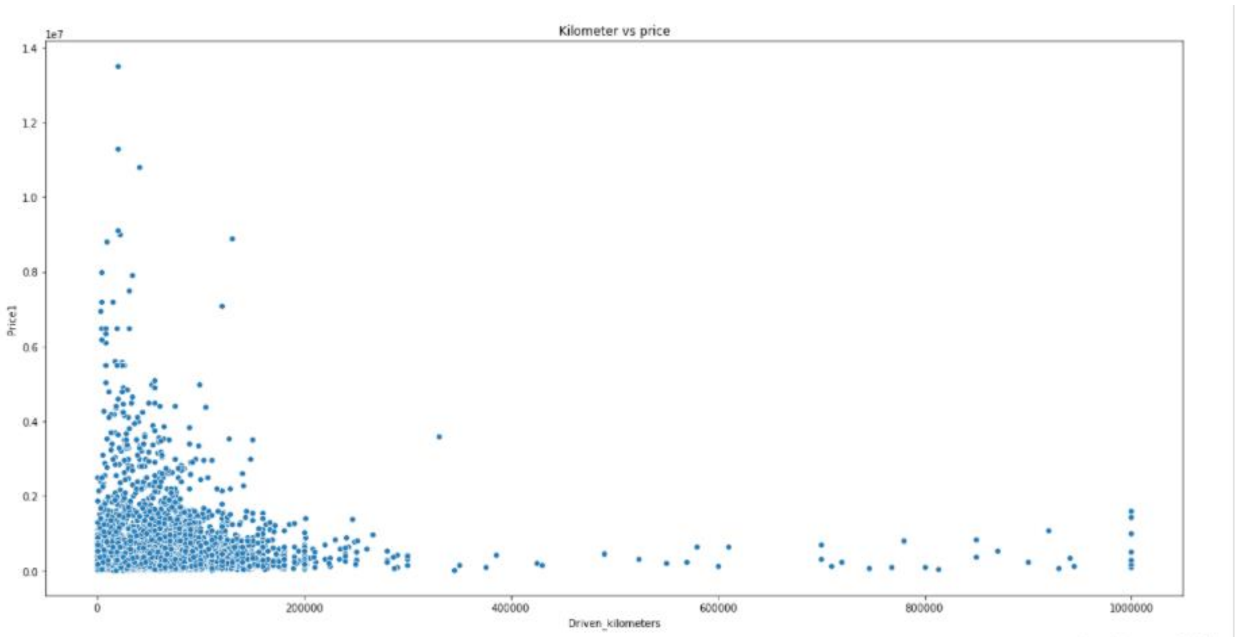
Activate Win

```
In [29]: #PLOT TO COMPARE TRANSMISSION WITH PRICE OF CAR
         sns.barplot(x=df["Transmission"],y=df["Price1"],palette="Set2")

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x22b28f68908>
```



Automatic cars are pricey when compare to manual

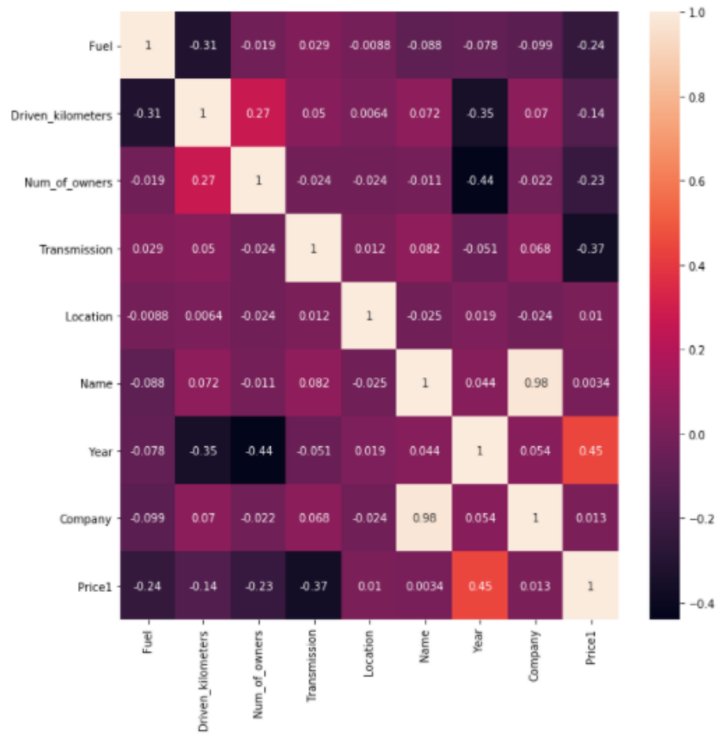
In [26]:
```python
#PLOT TO COMPARE KILOMETERS CAR DRIVEN TO PRICE OF CAR
plt.figure(figsize=(20,10))
plt.title('Kilometer vs price')
sns.scatterplot(x=df["Driven_kilometers"],y=df["Price1"],palette="Set2")
```
Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22b28f5bac8>
```



Activate Windows
Go to Settings to

The cars which has less kilometers driven has high value in the market

In [36]:
```python
#CHECKING CORRELATION
correlation=dff.corr(method='pearson')
plt.figure(figsize=(10,10))
sns.heatmap(correlation,annot=True)
```
Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22b28f62d88>
```



	Fuel	Driven_kilometers	Num_of_owners	Transmission	Location	Name	Year	Company	Price1
Fuel	1	-0.31	-0.019	0.029	-0.0088	-0.088	-0.078	-0.099	-0.24
Driven_kilometers	-0.31	1	0.27	0.05	0.0064	0.072	-0.35	0.07	-0.14
Num_of_owners	-0.019	0.27	1	-0.024	-0.024	-0.011	-0.44	-0.022	-0.23
Transmission	0.029	0.05	-0.024	1	0.012	0.082	-0.051	0.068	-0.37
Location	-0.0088	0.0064	-0.024	0.012	1	-0.025	0.019	-0.024	0.01
Name	-0.088	0.072	-0.011	0.082	-0.025	1	0.044	0.98	0.0034
Year	-0.078	-0.35	-0.44	-0.051	0.019	0.044	1	0.054	0.45
Company	-0.099	0.07	-0.022	0.068	-0.024	0.98	0.054	1	0.013
Price1	-0.24	-0.14	-0.23	-0.37	0.01	0.0034	0.45	0.013	1

INTERPRETATION OF THE RESULTS

- Most of the owners are first owner
- Most of the cars are manual car
- Almost equal number of petrol and diesel cars are available
- Recently bought car has high price compare to others
- The cars which has less kilometers driven has high value in the market
- If the car has single past owner than it has high value
- Diesel cars are high pricey than petrol cars
- New model car is pricier than old model in this reg plot it is apparent that price is positively correlated with year data. In the above representation more points are near to best fit line, it means that the newspaper variable has linear relationship with sales variable
- Year are highly and positively correlated with price
- Transmission is negatively correlated with price it means less number automatic cars available in market but its costlier than manual car

CONCLUSION

- According to the interpretation and modelling **RandomForest regressor** is best model
- Before hyper tuning the model predicted with accuracy **70.7%**
- Best parameters are {'bootstrap': True, 'max_depth': 45, 'max_features': 'log2', 'min_samples_leaf': 5, 'n_estimators': 500,}
- After hyper tuning the model predicted with accuracy is 73%.