Modified_Usedcarprice_prp.ipynb_-_Colaboratory.pdf

by

Submission date: 02-Aug-2023 10:05PM (UTC-0700)

Submission ID: 2140706141

File name: Modified_Usedcarprice_prp.ipynb_-_Colaboratory.pdf (542.28K)

Word count: 4490

Character count: 22013

```
import numpy as nump  # to to mathematical calculations
import pandas as panda  # for data manipulation
import seaborn as sb    # for 3D visualization
import lightgbm as ligbm
import matplotlib.pyplot as mplot
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import make_scorer, r2_score,mean_squared_error,mean_absolute_error # for calculating accuracy errors
from sklearn.neighbors import KNeighborsRegressor
```

```
df=panda.read_csv("car_price (1).csv") #IMporting Dataset
```

```
df.head()
```

Show hidden output

```
df.info();
```

Show hidden output

+ Code + Text

Data Preprocessing

```
df.Seats.replace(to_replace=0, value=df.Seats.median(), inplace=True)   # replacing values having 0 with the median of the column
```

```
df.drop("Unnamed: 0",axis=1,inplace=True)  # removing the column
```

```
df.head()
```

Show hidden output

```
df.select_dtypes(exclude=nump.number).nunique()
```

Show hidden output

```
df.drop("Name",axis=1,inplace=True)
```

```
(df.isnull().sum() / len(df)) * 100
```

Show hidden output

```
df.drop("New_Price",axis=1,inplace=True) #DRopping NewPrice coloumn due to NUll VaLue
```

```
for i in ["Mileage","Engine","Power"]:
    df[i].replace({nump.nan:"nan nan"},inplace=True)
```

```
df.head()
```

Show hidden output

```
for i in ["Mileage","Engine","Power"]:
    df[i] = df[i].apply(lambda x:x.split()[0])
```

```
for i in ["Mileage","Engine","Power"]:
    df[i].replace({"nan":"0"},inplace=True)
    df[i].replace({"null":"0"},inplace=True)
```

```
for i in ["Mileage","Engine","Power"]:
    df[i] = df[i].astype("float")
```

```
for i in ["Mileage","Engine","Power"]:
    df[i].replace({0:nump.nan},inplace=True)
```

```
df.isnull().sum() / len(df)*100
```

Show hidden output

```
for i in ["Mileage","Engine","Power","Seats"]:
    df[i].fillna(df[i].median(),inplace=True) # filling the null values with the median of the column
```

```
(df.isnull().sum() / len(df)) * 100
```

Show hidden output

```
df.groupby("Seats")["Price"].mean().sort_values(ascending=False) ##GROUPBY is used and COlumn is sorted in descending order
```

Show hidden output

```
df.groupby("Seats")["Price"].median()
```

Show hidden output

```
df.head()
```

Show hidden output

LabelEncoder

```
from sklearn.preprocessing import LabelEncoder  # for converting object type variables to numerical type variables
lbb_e = LabelEncoder()
df['Fuel_Type'] = lbb_e.fit_transform(df['Fuel_Type'])
df['Transmission'] = lbb_e.fit_transform(df['Transmission'])

print(df)
```

Show hidden output

```
df.Owner_Type.unique()
```

```
     array(['First', 'Second', 'Fourth & Above', 'Third'], dtype=object)
```

```
for i in df.index:
    if df.loc[i,"Owner_Type"] == "First": #providing index to FIRST
        df.loc[i,"Owner_Type"] = 1       #Provided Index is 1
    if df.loc[i,"Owner_Type"] == "Second": ##providing index to SECOND
        df.loc[i,"Owner_Type"] = 2       #Provided Index is 2
    if df.loc[i,"Owner_Type"] == "Third": #providing index to THIRD
        df.loc[i,"Owner_Type"] = 3      #Provided Index is 3
    if df.loc[i,"Owner_Type"] == "Fourth & Above": #providing index to FOURTH
        df.loc[i,"Owner_Type"] = 4       #Provided Index is 4
```

```
df.Owner_Type = df.Owner_Type.astype("int64")
```

```
df.Location = df.Location.map(df.groupby("Location")["Price"].median())
```

```
df.head()
```

Show hidden output

```
# Box Plots
num_cols = df.shape[1]
fig, axs = mplot.subplots(num_cols, 1, dpi=95, figsize=(7, 3*num_cols))
for loop, col in enumerate(df.columns):          # iterating every colmn of data
    axs[loop].boxplot(df[col], vert=False)  #BOXplot
    axs[loop].set_ylabel(col)          #for the ouTliers

mplot.tight_layout()  # To ensure proper spacing between subplots
mplot.show()
```

Show hidden output

```
quo1, quo3 = nump.percentile(df['Year'], [25, 75]) # Identifying the lower and upper quartile
inter_qr = quo3 - quo1
L_boun = quo1 - (1.5 * inter_qr)    # caculating bounds
U_boun = quo3 + (1.5 * inter_qr)     # caculating bounds
df = df[(df['Year'] >= L_boun)      # dropping outliers
                & (df['Year'] <= U_boun)]


quo1, quo3 = nump.percentile(df['Engine'], [25, 75])  # Identifying the lower and upper quartile
inter_qr = quo3 - quo1
L_boun = quo1 - (1.5 * inter_qr)  # caculating bounds
U_boun = quo3 + (1.5 * inter_qr)  # caculating bounds
df = df[(df['Engine'] >= L_boun)   # dropping outliers
                & (df['Engine'] <= U_boun)]


quo1, quo3 = nump.percentile(df['Power'], [25, 75])  # Identifying the lower and upper quartile
inter_qr = quo3 - quo1
L_boun = quo1 - (1.5 * inter_qr)  # caculating bounds
U_boun = quo3 + (1.5 * inter_qr) # caculating bounds
df = df[(df['Power'] >= L_boun)  # dropping outliers
                & (df['Power'] <= U_boun)]


quo1, quo3 = nump.percentile(df['Seats'], [25, 75]) # Identifying the lower and upper quartile
inter_qr = quo3 - quo1
L_boun = quo1 - (1.5 * inter_qr) # caculating bounds
U_boun = quo3 + (1.5 * inter_qr)  # caculating bounds
df = df[(df['Seats'] >= L_boun)   # dropping outliers
                & (df['Seats'] <= U_boun)]



quo1, quo3 = nump.percentile(df['Price'], [25, 75]) # Identifying the lower and upper quartile

inter_qr = quo3 - quo1
L_boun = quo1 - (1.5 * inter_qr)  # caculating bounds
U_boun = quo3 + (1.5 * inter_qr)  # caculating bounds
df = df[(df['Price'] >= L_boun)  # dropping outliers
                & (df['Price'] <= U_boun)]


X = df.drop("Price",axis=1)   # creating dependent variable containing all attribbutes except the price
y = df.Price              # creating dependent variable containing the attribute price
```

Training =75% and Testing=25% using Random Forest

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.25, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 25 pr


r_reg = RandomForestRegressor(n_estimators=100, random_state=42)

r_reg.fit(X_t, y_t)   # using fit to train

# Make predictions On The test Data
y_pre = r_reg.predict(X_tst)



# Evaluate the model
M_S_E=mean_squared_error(y_tst,y_pre)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre)) #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre) #calculating R2SCORe
print("M_S_E  OF TEST DATA USING RANDOM FOREST",M_S_E)
print("RM_S_E  OF TEST DATA USING RANDOM FOREST",rM_S_E)
print("M_A_E OF TEST DATA USING RANDOM FOREST",M_A_E)
print("R2-SCORE OF TEST DATA USING RANDOM FOREST:", r2)
```

```
M_S_E  OF TEST DATA USING RANDOM FOREST 1.169398665132205
RM_S_E  OF TEST DATA USING RANDOM FOREST 1.0813873797729494
M_A_E OF TEST DATA USING RANDOM FOREST 0.6926525077447335
R2-SCORE OF TEST DATA USING RANDOM FOREST: 0.8671108683443405
```

Training =80% and Testing=20% using Random Forest

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.20, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 20 pr

# Create the Random Forest regressor
r_reg = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the training data
r_reg.fit(X_t, y_t)

# Make predictions on the test data
y_pre = r_reg.predict(X_tst)



# Evaluate the model
M_S_E=mean_squared_error(y_tst,y_pre)   #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre) ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre) #calculating R2SCORe
print("M_S_E OF TEST DATA USING RANDOM FOREST",rM_S_E)
print("RM_S_E OF TEST DATA USING RANDOM FOREST",rM_S_E)
print("M_A_E OF TEST DATA USING RANDOM FOREST",M_A_E)
print("R2-SCORE OF TEST DATA USING RANDOM FOREST", r2)
```

```
     M_S_E OF TEST DATA USING RANDOM FOREST 1.0661767161294604
     RM_S_E OF TEST DATA USING RANDOM FOREST 1.0661767161294604
     M_A_E OF TEST DATA USING RANDOM FOREST 0.6859726001511715
     R2-SCORE OF TEST DATA USING RANDOM FOREST 0.872780632254947
```

Training =85% and Testing=15% using Random Forest

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.15, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 15 pr
```

for n-estimators=90

```
# Create the Random Forest regressor
Rf_Reg = RandomForestRegressor(n_estimators=90, random_state=42)

# Train the model on the training data
Rf_Reg.fit(X_t, y_t)

# Make predictions on the test data
y_pre = Rf_Reg.predict(X_tst)



 #evaluate
M_S_E=mean_squared_error(y_tst,y_pre)   #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre))  #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)    ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)            #calculating R2SCORe
print("M_S_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST",M_S_E)
print("RM_S_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST",rM_S_E)
print("M_A_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST",M_A_E)
print("R2-SCORE OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST", r2)
```

```
     M_S_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST 1.1794091729446972
     RM_S_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST 1.0860060648747305
     M_A_E OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST 0.6890450308778482
     R2-SCORE OF TEST DATA FOR N_ESTIMATORS=90, USING RANDOM FOREST 0.8677358723851435
```

for n-estimators =100

```
# Create the Random Forest regressor
Rf_Reg = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the training data
Rf_Reg.fit(X_t, y_t)

# Make predictions on the test data
y_pre = Rf_Reg.predict(X_tst)


# Evaluate the model for n_estimators=100
M_S_E=mean_squared_error(y_tst,y_pre) #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre)) #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)   #calculating R2SCORe
print("M_S_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST",M_S_E)
print("RM_S_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST",rM_S_E)
print("M_A_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST",M_A_E)
print("R2-SCORE OF TEST FOR N_ESTIMATORS=100, DATA USING RANDOM FOREST", r2)
```

```
    M_S_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST 1.1830042392453741
    RM_S_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST 1.087659983287688
    M_A_E OF TEST DATA FOR N_ESTIMATORS=100, USING RANDOM FOREST 0.689232573713696
    R2-SCORE OF TEST FOR N_ESTIMATORS=100, DATA USING RANDOM FOREST 0.8673327058515228
```

for n-estimators=110

```
# Create the Random Forest regressor
Rf_Reg = RandomForestRegressor(n_estimators=110, random_state=42)

# Train the model on the training data
Rf_Reg.fit(X_t, y_t)

# Make predictions on the test data
y_pre =Rf_Reg.predict(X_tst)


# Evaluate the model for n_estimators=110
M_S_E=mean_squared_error(y_tst,y_pre)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)   #calculating R2SCORe
print("M_S_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST",M_S_E)
print("RM_S_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST",rM_S_E)
print("M_A_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST",M_A_E)
print("R2-SCORE OF TEST FOR N_ESTIMATORS=110, DATA USING RANDOM FOREST", r2)
```

```
    M_S_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST 1.18507413464996
    RM_S_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST 1.0886111034937866
    M_A_E OF TEST DATA FOR N_ESTIMATORS=110, USING RANDOM FOREST 0.689151926943026
    R2-SCORE OF TEST FOR N_ESTIMATORS=110, DATA USING RANDOM FOREST 0.8671005786846143
```

Training =75% and Testing=25% using ligbmM

```
# Step 1: Split data into training and testing sets
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.25, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 25 pr

# Step 2: Create and train the ligbmMRegressor model
Lb_Reg = ligbm.LGBMRegressor(force_row_wise=True)  # Set force_row_wise=True to remove the warning
Lb_Reg.fit(X_t, y_t)

# Step 3: Make predictions on the testing data
y_pre = Lb_Reg.predict(X_tst)
```

Show hidden output

```
# Evaluate the model
M_S_E=mean_squared_error(y_tst,y_pre)   #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre)) #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)  #calculating R2SCORe
print("M_S_E OF TEST DATA USING ligbm",M_S_E)
print("RM_S_E OF TEST DATA USING ligbm",rM_S_E)
print("M_A_E OF TEST DATA USING ligbm",M_A_E)
print("R2-SCORE OF TEST DATA USING ligbm", r2)
```

```
    M_S_E OF TEST DATA USING ligbm 0.9616469024571181
    RM_S_E OF TEST DATA USING ligbm 0.9806359683680372
    M_A_E OF TEST DATA USING ligbm 0.6420736066428658
    R2-SCORE OF TEST DATA USING ligbm 0.8907195419002519
```

Training =80% and Testing=20% using ligbmM

```
# Step 1: Split data into training and testing sets
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.20, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 20 pr

# Step 2: Create and train the ligbmMRegressor model
# Set force_row_wise=True to remove the warning
ligbm_regressor = ligbm.LGBMRegressor(force_row_wise=True)
ligbm_regressor.fit(X_t, y_t)

# Step 3: Make predictions on the testing data
y_pre = ligbm_regressor.predict(X_tst)
```

Show hidden output

```
# Evaluate the model
M_S_E=mean_squared_error(y_tst,y_pre)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre)) #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)  #calculating R2SCORe
print("M_S_E OF TEST DATA USING ligbm",M_S_E)
print("RM_S_E OF TEST DATA USING ligbm",rM_S_E)
print("M_A_E OF TEST DATA USING RANDOM ligbm",M_A_E)
print("R2-SCORE OF TEST DATA USING ligbm", r2)
```

```
    M_S_E OF TEST DATA USING ligbm 0.9717493389222678
    RM_S_E OF TEST DATA USING ligbm 0.9857734724176076
    M_A_E OF TEST DATA USING RANDOM ligbm 0.6466856570452988
    R2-SCORE OF TEST DATA USING ligbm 0.8912450334941435
```

Training =85% and Testing=15% using ligbmM

```
# Step 1: Split data into training and testing sets
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.15, random_state=42)  ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 15 p

# Step 2: Create and train the ligbmMRegressor model
# Set force_row_wise=True to remove the warning
ligbm_regressor = ligbm.LGBMRegressor()
ligbm_regressor.fit(X_t, y_t)

# Step 3: Make predictions on the testing data
y_pre = ligbm_regressor.predict(X_tst)
```

Show hidden output

```
# Evaluate the model
M_S_E=mean_squared_error(y_tst,y_pre)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_tst,y_pre)) #calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_tst,y_pre)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_tst, y_pre)   #calculating R2SCORe
print("M_S_E OF TEST DATA USING ligbm",M_S_E)
print("RM_S_E OF TEST DATA USING ligbm",rM_S_E)
print("M_A_E OF TEST DATA USING ligbm",M_A_E)
print("R2-SCORE OF TEST DATA USING ligbm", r2)
```

```
    M_S_E OF TEST DATA USING ligbm 1.0027819382655123
    RM_S_E OF TEST DATA USING ligbm 1.001390003078477
    M_A_E OF TEST DATA USING ligbm 0.6527965020658273
    R2-SCORE OF TEST DATA USING ligbm 0.8875436266775228
```

TRAINING DATA

1) USING RANDOM FOREST

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.25, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 25 pr
r_reg = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the training data
r_reg.fit(X_t, y_t)

# Make predictions on the test data
y_pre_t = r_reg.predict(X_t)


# Evaluate the model
M_S_E=mean_squared_error(y_t,y_pre_t)
rM_S_E= nump.sqrt(mean_squared_error(y_t,y_pre_t))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_t,y_pre_t)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_t, y_pre_t)     #calculating R2SCORe
print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)
```

```
     M_S_E OF TRAIN DATA USING ligbm 0.16206718211419943
     RM_S_E OF TRAIN DATA USING ligbm 0.40257568495153734
     M_A_E OF TRAIN DATA USING ligbm 0.2595597089291247
     R2-SCORE OF TRAIN DATA USING ligbm 0.9804288233385972
```

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.20, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 20 pr

# Make predictions on the test data
y_pre_t = r_reg.predict(X_t)


M_S_E=mean_squared_error(y_t,y_pre_t)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_t,y_pre_t))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_t,y_pre_t) ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_t, y_pre_t)   #calculating R2SCORe
print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)
```

```
     M_S_E OF TRAIN DATA USING ligbm 0.22668702777660252
     RM_S_E OF TRAIN DATA USING ligbm 0.47611661153188356
     M_A_E OF TRAIN DATA USING ligbm 0.2863795770039566
     R2-SCORE OF TRAIN DATA USING ligbm 0.9726204204270692
```

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.15, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 15 pr

# Make predictions on the test data
y_pre_t = r_reg.predict(X_t)


print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)
```

```
     M_S_E OF TRAIN DATA USING ligbm 0.22668702777660252
     RM_S_E OF TRAIN DATA USING ligbm 0.47611661153188356
     M_A_E OF TRAIN DATA USING ligbm 0.2863795770039566
     R2-SCORE OF TRAIN DATA USING ligbm 0.9726204204270692
```

2)ligbmM

```
# Step 1: Split data into training and testing sets
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.25, random_state=42)  ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 25 p
```

```
# Step 2: Create and train the ligbmMRegressor model
ligbm_regressor = ligbm.LGBMRegressor(force_row_wise=True)  # Set force_row_wise=True to remove the warning
ligbm_regressor.fit(X_t, y_t)

# Step 3: Make predictions on the testing data
y_pre_t = ligbm_regressor.predict(X_t)
```

Show hidden output

```
M_S_E=mean_squared_error(y_t,y_pre_t)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_t,y_pre_t))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_t,y_pre_t) ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_t, y_pre_t)    #calculating R2SCORe
print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)

     M_S_E OF TRAIN DATA USING ligbm 0.44306920878056544
     RM_S_E OF TRAIN DATA USING ligbm 0.6656344408010793
     M_A_E OF TRAIN DATA USING ligbm 0.46441004696390087
     R2-SCORE OF TRAIN DATA USING ligbm 0.9464951161292965
```

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.20, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 20 pr

# Step 3: Make predictions on the testing data
y_pre_t = ligbm_regressor.predict(X_t)
```

```
M_S_E=mean_squared_error(y_t,y_pre_t)   #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_t,y_pre_t))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_t,y_pre_t) ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_t, y_pre_t)   #calculating R2SCORe
print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)

     M_S_E OF TRAIN DATA USING ligbm 0.4736417452559269
     RM_S_E OF TRAIN DATA USING ligbm 0.6882163506165244
     M_A_E OF TRAIN DATA USING ligbm 0.47503478563636997
     R2-SCORE OF TRAIN DATA USING ligbm 0.9427928806491901
```

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.15, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 15 pr

# Step 3: Make predictions on the testing data
y_pre_t = ligbm_regressor.predict(X_t)
```

```
M_S_E=mean_squared_error(y_t,y_pre_t)  #calcUlating MEAN SUARRED ERROR
rM_S_E= nump.sqrt(mean_squared_error(y_t,y_pre_t))#calcUlating ROOT MEAN SUARRED ERROR
M_A_E=mean_absolute_error(y_t,y_pre_t)  ##calcUlating MEAN ABSOLUTE ERROR
r2 = r2_score(y_t, y_pre_t)   #calculating R2SCORe
print("M_S_E OF TRAIN DATA USING ligbm",M_S_E)
print("RM_S_E OF TRAIN DATA USING ligbm",rM_S_E)
print("M_A_E OF TRAIN DATA USING ligbm",M_A_E)
print("R2-SCORE OF TRAIN DATA USING ligbm", r2)

     M_S_E OF TRAIN DATA USING ligbm 0.49560055417437069
     RM_S_E OF TRAIN DATA USING ligbm 0.7039925722219709
     M_A_E OF TRAIN DATA USING ligbm 0.48304244115541484
     R2-SCORE OF TRAIN DATA USING ligbm 0.9404406729785681
```

For Cross-Validation=5 using ligbmM

```
X_t, X_tst, y_t, y_tst = train_test_split(X, y, test_size=0.15, random_state=42) ##Importing TrainTestSplit From SKL.MODELSELECTION FOR 15 pr


# Step 1: Create the ligbmMRegressor model
ligbm_regressor = ligbm.LGBMRegressor(force_row_wise=True)

# Step 2: Define the scoring function (R-squared in this case)
sc = make_scorer(r2_score)
```

```
# Step Perform cross-validation and get the scores
cv5 = cross_val_score(ligbm_regressor, X, y, cv=5, scoring=sc)

# Step 4: Calculate the average R-squared score
average_score = nump.mean(cv5)
```

Show hidden output

```
print(" R-squared Scores for cv=5 using ligbm:", cv5)
print("Average R-squared Score for cv=5 using ligbm:", average_score)
```

```
      R-squared Scores for cv=5 using ligbm: [0.88580808 0.86347417 0.87785349 0.86619123 0.88643535]
     Average R-squared Score for cv=5 using ligbm: 0.8759524626350881
```

For Cross-Validation=10 using ligbmM

```
# STESP Perform cross-validation and get the scores
cv10 = cross_val_score(ligbm_regressor, X, y, cv=10, scoring=sc)

# STESP 4: Calculate the average R-squared score
average_score = nump.mean(cv10)
```

Show hidden output

```
print(" R-squared Scores for cv=10 using ligbm:", cv10);
print("Average R-squared Score for cv=10 using ligbm:", average_score);
```

```
      R-squared Scores for cv=10 using ligbm: [0.88033199 0.88429176 0.890888   0.89356862 0.89973949 0.86820388
      0.88151438 0.85407091 0.88190856 0.89671098]
     Average R-squared Score for cv=10 using ligbm: 0.8831228569010579
```

For Cross-Validation=5 using Random Forest

```
r_reg = RandomForestRegressor()

# Step 2: definying The Scoring Function (R-squared in this case)
scoring_function = make_scorer(r2_score)

# Step 3: Perform cross-validation and get the scores
cv_scores = cross_val_score(r_reg, X, y, cv=5, scoring=scoring_function)

# Step 4: Calculate the average R-squared score
average_score = nump.mean(cv_scores)

print(" R-squared Scores using for cv=5 using RANDOM FOREST :", cv_scores)
print("Average R-squared Score FOR cv=5 USING RANDOM FOREST:", average_score)
```

```
      R-squared Scores using for cv=5 using RANDOM FOREST : [0.87680168 0.85296071 0.86704244 0.85561365 0.88355367]
     Average R-squared Score FOR cv=5 USING RANDOM FOREST: 0.8671944304686956
```

For Cross-Validation=10 using Random Forest

```
# Step 3: form cross-validation and get the scores
cv_s_10 = cross_val_score(r_reg, X, y, cv=10, scoring=scoring_function)

# Step 4: Calculate the average R-squared score
average_score = nump.mean(cv_s_10)

print(" R-squared Scores using for cv=10 using RANDOM FOREST :", cv_s_10)
print("Average R-squared Score FOR cv=10 USING RANDOM FOREST:", average_score)
```

```
      R-squared Scores using for cv=10 using RANDOM FOREST : [0.8700854  0.87328463 0.86405324 0.86940645 0.90031105 0.84996925
      0.86427017 0.85487379 0.87704339 0.89141713]
     Average R-squared Score FOR cv=10 USING RANDOM FOREST: 0.8714714494067934
```

Modified_Usedcarprice_prp.ipynb_-_Colaboratory.pdf

ORIGINALITY REPORT

21%	14%	14%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Vaibhav Verdhan. "Supervised Learning with Python", Springer Science and Business Media LLC, 2020 Publication	4%
2	Submitted to Keio University Student Paper	2%
3	gifadn.medium.com Internet Source	2%
4	gr.xjtu.edu.cn Internet Source	2%
5	github.com Internet Source	2%
6	Submitted to Aston University Student Paper	2%
7	Submitted to University of Hertfordshire Student Paper	1%
8	Ekaba Bisong. "Building Machine Learning and Deep Learning Models on Google Cloud	1%

	Platform", Springer Science and Business Media LLC, 2019 Publication	
9	www.cfdyna.com Internet Source	1 %
10	www.scss.tcd.ie Internet Source	1 %
11	Submitted to University of Huddersfield Student Paper	<1 %
12	www.kaggle.com Internet Source	<1 %
13	www.programcreek.com Internet Source	<1 %
14	discuss.streamlit.io Internet Source	<1 %
15	Submitted to University College London Student Paper	<1 %
16	riyan-aditya.github.io Internet Source	<1 %
17	Submitted to Coventry University Student Paper	<1 %
18	blog.csdn.net Internet Source	<1 %
19	iq.opengenus.org Internet Source	<1 %

20	blog.lucbertin.com Internet Source	<1 %
21	machinelearningmastery.com Internet Source	<1 %
22	Manohar Swamynathan. "Chapter 4 Step 4 – Model Diagnosis and Tuning", Springer Science and Business Media LLC, 2017 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		