Modified_UsedCarPrice_base.ipynb_-_Colaboratory.pdf

by

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

Submission ID: 2140706133

File name: Modified_UsedCarPrice_base.ipynb_-_Colaboratory.pdf (308.26K)

Word count: 2321

Character count: 12080

```
import numpy as nump # for mathematical caalculation
import pandas as panda # for data manipulation
import seaborn as sea   # for 3D data visualization
import matplotlib.pyplot as mplot   # for 2D data visualization
from sklearn.model_selection import train_test_split  # splitting the data in train and test part
from sklearn.metrics import  mean_squared_error , r2_score, mean_absolute_error #To calculate accuracy  and error
import scipy.stats as stats
from sklearn.model_selection import cross_val_score # importing for cross validation
from sklearn.neighbors import KNeighborsRegressor  # import the regressor
```

```
df=panda.read_csv("car_price (1).csv")  #DATAset is imported
```

```
df.info()
```

Show hidden output

```
df.head()
```

Show hidden output

```
df.shape
```

Show hidden output

Data Preprocessing

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

```
df.drop("Unnamed: 0",axis=1,inplace=True)  # Removing the column not required to be involved
```

```
df.Year = df.Year.astype("object")
```

```
df.head()
```

Show hidden output

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

Show hidden output

```
df.drop("Name",axis=1,inplace=True) #NAME coloumn is Dropped
```

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

Show hidden output

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

```
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

```
sea.boxplot(df.Price)
```

Show hidden output

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

Show hidden output

```
for i in df.select_dtypes(exclude=nump.number).columns[1:-1]:
    print(i,"- % of df")
    print(df[i].value_counts(normalize=True))
    print()
```

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

```
data1 = df.copy()
df.drop("Kilometers_Driven",axis=1,inplace=True)
data1 = df.copy()
df.head()
```

Show hidden output

```
from sklearn.preprocessing import LabelEncoder #HERE IMPORTNG lABel ancoder
lbb_e = LabelEncoder()
df['Fuel_Type'] = lbb_e.fit_transform(df['Fuel_Type']) #converting categorical into numerical data
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 AND ABOVE
        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

```
X = df.drop("Price",axis=1)  # creating a dependent variable containing all the attributes except price
Y = df.Price         # creating independent variable containing only the price column
```

Training & Testing (0.25)

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.25,random_state=18) # dividing the data into 75%  train part and rest test part
```

```
knn = KNeighborsRegressor().fit(X_t,y_t) # fit is used to train
knn_tst_pre = knn.predict(X_tst)   #for The prediction of test data
r2 = r2_score(y_tst,knn_tst_pre)    #calculating R2SCORe

avgae=mean_squared_error(y_tst,knn_tst_pre)  #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_tst,knn_tst_pre)) #calcUlate square root of average of squares of error
avgae = mean_absolute_error(y_tst,knn_tst_pre)  #calcUlate average of absolute error




print("TEST R2 SCORE :",r2);
print("TEST avgae SCORE :",avgae);
print("TEST Ravgae SCORE :",ravgae);
print(" TEST avgae SCORE :",avgae);

     TEST R2 SCORE : 0.8334731886360316
     TEST avgae SCORE : 1.9590073089700994
     TEST Ravgae SCORE : 4.494284540786755
      TEST avgae SCORE : 1.9590073089700994
```

Training & Testing (0.20)

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.20,random_state=16) # dividing the data into 80%  train part and rest test part
```

```
knn = KNeighborsRegressor().fit(X_t,y_t)  #training
knn_t_pre = knn.predict(X_t)   # making predictions on train data
knn_tst_pre = knn.predict(X_tst)     # making predictions on test data
r2 = r2_score(y_tst,knn_tst_pre)
r2_train = r2_score(y_t,knn_t_pre)    #calculating R2SCORe
avgae=mean_squared_error(y_tst,knn_tst_pre)    #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_tst,knn_tst_pre)) #calcUlate square root of avverage of the squares of error
avgae = mean_absolute_error(y_tst,knn_tst_pre)   #calcUlate avverage of the squares of absolute error



print("TEST R2 SCORE :",r2)
print("TEST avgae SCORE :",avgae)
print("TEST Ravgae SCORE :",ravgae)
print("TEST avgae SCORE :",avgae)

     TEST R2 SCORE : 0.8406138998369084
     TEST avgae SCORE : 2.0065930232558142
     TEST Ravgae SCORE : 4.6994221045016875
     TEST avgae SCORE : 2.0065930232558142
```

Training & Testing (0.15)

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.15,random_state=18) # dividing the data into 85%  train part and rest test part
```

```
knn = KNeighborsRegressor().fit(X_t,y_t)  #training
knn_t_pre = knn.predict(X_t)    # making prediction of train data
knn_tst_pre = knn.predict(X_tst)      #making prediction of test data
r2 = r2_score(y_tst,knn_tst_pre)        #calculating R2SCORe
r2_train = r2_score(y_t,knn_t_pre)
avgae=mean_squared_error(y_tst,knn_tst_pre)   #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_tst,knn_tst_pre))   #calcUlate square root of avverage of the squares of error
avgae = mean_absolute_error(y_tst,knn_tst_pre)      #calcUlate avverage of the squares of absolute error


#TEST DATA
print("TEST R2 SCORE USING KNN",r2)
print("TEST avgae SCORE USING KNN:",avgae)
print("TEST Ravgae SCORE USING KNN:",ravgae)
print("TEST avgae SCORE USING KNN:",avgae)
```

```
    TEST R2 SCORE USING KNN 0.8552603575218423
    TEST avgae SCORE USING KNN: 1.9093045404208193
    TEST Ravgae SCORE USING KNN: 4.247825918292223
    TEST avgae SCORE USING KNN: 1.9093045404208193
```

TRAINING

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.25,random_state=18) # dividing the data into 75%  train part and rest test part


knn_t_pre = knn.predict(X_t)    #for The prediction of train the data
r2_train = r2_score(y_t,knn_t_pre)    #calculating R2SCORe
avgae=mean_squared_error(y_t,knn_t_pre)    #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_t,knn_t_pre)) #calcUlate square root of avverage of the squares of error
avgae = mean_absolute_error(y_t,knn_t_pre)   #calcUlate avverage of the squares of absolute error


print("TRAIN R2 SCORE USING KNN:",r2_train);
print("TRAIN avgae SCORE USING KNN:",avgae);
print("TRAIN Ravgae SCORE USING KNN:",ravgae);
print("TRAIN avgae SCORE USING KNN:",avgae);
```

```
    TRAIN R2 SCORE USING KNN: 0.9020450830362835
    TRAIN avgae SCORE USING KNN: 1.4049065130704477
    TRAIN Ravgae SCORE USING KNN: 3.5186109667218304
    TRAIN avgae SCORE USING KNN: 1.4049065130704477
```

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.20,random_state=16) # dividing the data into 80%  train part and rest test part


knn_t_pre = knn.predict(X_t)
r2_train = r2_score(y_t,knn_t_pre)    #calculating R2SCORe
avgae=mean_squared_error(y_t,knn_t_pre)    #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_t,knn_t_pre))  #calcUlate square root of avverage of the squares of error
avgae = mean_absolute_error(y_t,knn_t_pre)  #calcUlate avverage of the squares of absolute error


print("TRAIN R2 SCORE USING KNN:",r2_train);
print("TRAIN avgae SCORE USING KNN:",avgae);
print("TRAIN Ravgae SCORE USING KNN:",ravgae);
print("TRAIN avgae SCORE USING KNN:",avgae);
```

```
    TRAIN R2 SCORE USING KNN: 0.8909241271947834
    TRAIN avgae SCORE USING KNN: 1.4639102803738318
    TRAIN Ravgae SCORE USING KNN: 3.6447382737464356
    TRAIN avgae SCORE USING KNN: 1.4639102803738318
```

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.15,random_state=18)# dividing the data into 85%  train part and rest test part


knn_t_pre = knn.predict(X_t)
r2_train = r2_score(y_t,knn_t_pre)    #calculating R2SCORe
avgae=mean_squared_error(y_t,knn_t_pre)  #calcUlate avverage of the squares of error
ravgae = nump.sqrt(mean_squared_error(y_t,knn_t_pre)) #calcUlate square root of avverage of the squares of error
avgae = mean_absolute_error(y_t,knn_t_pre)    #calcUlate avverage of the squares of absolute error


print("TRAIN R2 SCORE USING KNN:",r2_train);
print("TRAIN avgae SCORE USING KNN:",avgae);
print("TRAIN Ravgae SCORE USING KNN:",ravgae);
print("TRAIN avgae SCORE USING KNN:",avgae);
```

```
    TRAIN R2 SCORE USING KNN: 0.8966261712991226
    TRAIN avgae SCORE USING KNN: 1.4160054730258014
    TRAIN Ravgae SCORE USING KNN: 3.5972240972212726
    TRAIN avgae SCORE USING KNN: 1.4160054730258014
```

For Cross Validation=10 and k=3

```
# At fold=10 and testing data =85% we obtain max average accuracy at different k Value
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.15,random_state=18) # dividing the data into 85%  train part and rest test part
classifier =  KNeighborsRegressor(n_neighbors=3)

scr_10 = cross_val_score(classifier, X, Y, cv=10)

print(" Accuracy scr of cross-validation=10 and k=3:", scr_10)
```

```
     Accuracy scr of cross-validation=10 and k=3: [0.85596841 0.76893946 0.8448288  0.82195534 0.91614376 0.78125569
     0.79488476 0.8610244  0.8512696  0.81446828]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=3:",scr_10.mean())
```

```
    AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=3: 0.8310738485109382
```

For Cross Validation=10 and k=4

```
classifier =  KNeighborsRegressor(n_neighbors=4)

scr_10 = cross_val_score(classifier, X, Y, cv=10)

print(" Accuracy scr of cross-validation=10 and k=4:", scr_10)
```

```
     Accuracy scr of cross-validation=10 and k=4: [0.85574747 0.78365493 0.86209139 0.83967908 0.91674976 0.76996416
     0.77557822 0.8601543  0.84333821 0.81046981]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=4:",scr_10.mean())
```

```
    AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=4: 0.8317427333059009
```

For Cross Validation=10 and k=5

```
classifier =  KNeighborsRegressor(n_neighbors=5)

scr_10 = cross_val_score(classifier, X, Y, cv=10)

print(" Accuracy scr of cross-validation=10 and k=5:", scr_10)
```

```
     Accuracy scr of cross-validation=10 and k=5: [0.85792239 0.7743356  0.85289911 0.8248018  0.91204373 0.76521722
     0.77294838 0.84870949 0.82213078 0.80921901]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=5:",scr_10.mean())
```

```
    AVERAGE SCORE OF CROSS-VALIDATION=10 AND K=5: 0.8240227510567971
```

For Cross Validation=5 and k=3

```
X_t,X_tst,y_t,y_tst = train_test_split(X,Y,test_size=0.15,random_state=18)
classifier =  KNeighborsRegressor(n_neighbors=3)

scr_5 = cross_val_score(classifier, X, Y, cv=5)
```

```
print(" Accuracy scr of cross-validation=5 and k=3:", scr_5)
```

```
 Accuracy scr of cross-validation=5 and k=3: [0.79613224 0.82261953 0.84648584 0.8255664  0.82254703]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=3:",scr_5.mean())
```

```
AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=3: 0.8226702097899729
```

For Cross Validation=5 and k=4

```
classifier =  KNeighborsRegressor(n_neighbors=4)

scr_5 = cross_val_score(classifier, X, Y, cv=5)

print(" Accuracy scr of cross-validation=5 and k=4:", scr_5)
```

```
 Accuracy scr of cross-validation=5 and k=4: [0.80303358 0.83773746 0.83709607 0.81209452 0.81462139]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=4:",scr_5.mean())
```

```
AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=4: 0.8209166068454273
```

For Cross Validation=5 and k=5

```
classifier =  KNeighborsRegressor(n_neighbors=5)

scr_5 = cross_val_score(classifier, X, Y, cv=5)

print(" Accuracy scr of cross-validation=5 and k=5:", scr_5)
```

```
 Accuracy scr of cross-validation=5 and k=5: [0.80199785 0.81765146 0.82948365 0.81422963 0.80744354]
```

```
print("AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=5:",scr_5.mean())
```

```
AVERAGE SCORE OF CROSS-VALIDATION=5 AND K=5: 0.8141612268645118
```

Modified_UsedCarPrice_base.ipynb_-_Colaboratory.pdf

ORIGINALITY REPORT

17%	13%	9%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.cfdyna.com Internet Source	4%
2	Submitted to Queen Mary and Westfield College Student Paper	3%
3	Manohar Swamynathan. "Mastering Machine Learning with Python in Six Steps", Springer Science and Business Media LLC, 2017 Publication	2%
4	"Next Generation Information Processing System", Springer Science and Business Media LLC, 2021 Publication	1%
5	Submitted to Universitas Pertamina Student Paper	1%
6	www.cs.colostate.edu Internet Source	1%
7	educationalresearchtechniques.com Internet Source	1%
	Submitted to University of Essex	

8		Student Paper	1%
9	gr.xjtu.edu.cn	Internet Source	1%
10	Submitted to University of North Texas	Student Paper	1%
11	Submitted to Bilkent University	Student Paper	1%
12	Submitted to University of Technology, Sydney	Student Paper	1%
13	compphysics.github.io	Internet Source	<1%
14	machinelearninghd.com	Internet Source	<1%
15	dev.to	Internet Source	<1%
16	www.kaggle.com	Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		