Assignment:

Case Study

Submitted by:

M.Mohsin ur Rehman Khan

Submitted to:

Dr. Muhammad Ammar Tufail
email:mohsinrehman677@gmail.com
Ph # +923048827042
Mianwali,Punjab,Pakistan

Date:(04/08/2022)

Flight Price Prediction Model

Instructions

1. You will have dataset
2. Find cheapest and expensive flight at a specific time
3. You have to go through EDA
4. ML model
5. Find a sweet spot for cheap ticket

- Ahmad is customer of sastaticket.pk.He is planning to fly from karachi to islamabad.
- But he doesnt buy ticket now and he thinks that prices are too much high now and he must wait.
- Can you tell ahmad he is making wrong decision with your analysis and a very confident prediction

1.EDA(Exploratory Data Analysis)

(We will extract info from our data)

In [2]:
```
#Importing libraries
import seaborn as sns
import pandas as pd
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
```

In [ ]:
```
# reading testing data
X_test=pd.read_csv("https://raw.githubusercontent.com/AammarTufail/machinelearning_ka_chill
a/master/Sastaticket_datasets/sastaticket_test.csv")
```

In [ ]:
```
#head
X_test.head()
```

In [ ]:
```
# save it to csv file sample (250)
X_test.to_csv("sastaticket_test.csv")
```

In [ ]:
```
# reading testing data sample(250)
df_test=pd.read_csv("sastaticket_test.csv")
df_test.head()
```

In [ ]:
```
# reading training dataset(sample) from github
training_data=pd.read_csv("https://raw.githubusercontent.com/AammarTufail/machinelearning_ka
_chilla/d4693bff19a9e9d47908b55218daa0b9a77a13c3/Sastaticket_datasets/sastaticket_train.csv"
)
```

In [ ]:
```
#head
training_data.head(3)
```

In [ ]:
```
# save to csv file
training_data.to_csv("sastaticket_trainsample.csv")
```

In [3]:
```
#read
df=pd.read_csv("sastaticket_trainsample.csv")
```

In [ ]:
```
#head
df.head(2)
```

In [ ]:
```
#shape
df.shape
```

In [ ]:
```
# info
df.info()
```

In [ ]:
```
#check null values
#df.isnull().sum()
df.notnull().sum()
```

In [ ]:
```
#summary statistics
df.describe()
```

In [ ]:
```
#head
df.head(3)
```

In [4]:
```
# first we drop unnamed data
df.drop(["Unnamed: 0.3","Unnamed: 0.2","Unnamed: 0","Unnamed: 0.1"], axis=1, inplace=True)
```

In [ ]:
```
#columns
df.columns
```

In [ ]:
```
df.nunique()
```

In [ ]:
```
# separate categorical variables in a list
cat_list=["f2","f3","f6","f7","f8","f9","f10"]
```

In [ ]:
```
# for loop to check unique values in each

for i in cat_list:

    print(i, df[i].unique())
    print("............................")# separatorline
```

In [5]:
```
# As we can see that f2 and f3 have no unique values and they will not effect the target.
# f10 is flight no which also has no impact on target.
# so it's better to drop f2,f3,f10

df.drop(["f2","f3","f10"], axis=1, inplace=True)
```

Assignment#1

If we replace axis from 1 to 0 it will give error of "not found in axis"

In [6]:
```
#head
df.head()
```

Out[6]:

	f1	f4	f5	f6	f7	f8	f9	target
0	2021-01-08 12:43:27.828728+00:00	2021-01-23 05:00:00+00:00	2021-01-23 07:00:00+00:00	gamma	True	0.0	0	7400.0
1	2021-07-01 04:45:11.397541+00:00	2021-07-01 13:00:00+00:00	2021-07-01 15:00:00+00:00	alpha	True	35.0	1	15377.0
2	2021-06-24 11:28:47.565115+00:00	2021-07-29 14:00:00+00:00	2021-07-29 16:00:00+00:00	gamma	True	20.0	1	6900.0
3	2021-06-05 11:09:48.655927+00:00	2021-06-09 16:00:00+00:00	2021-06-09 18:00:00+00:00	alpha	True	15.0	1	9707.0
4	2021-07-29 09:53:51.065306+00:00	2021-08-23 05:00:00+00:00	2021-08-23 06:55:00+00:00	beta	True	20.0	0	6500.0

In [ ]:
```
# casting
df.info()
```

In [7]:
```
# as columns f1,f4,f5 shows date and time with dtype object
# we convert dtype to datetime
#for that import datetime library
from datetime import date,time
df["f1"]=pd.to_datetime(df["f1"])
df["f4"]=pd.to_datetime(df["f4"])
df["f5"]=pd.to_datetime(df["f5"])
```

In [8]:
```
#importing date,timedelta
from datetime import date,timedelta
```

In [9]:
```
# add a new column(time_to_dparture) by subtracting f1 from f4

df.insert(0,"time_to_dep(s)" , ((df['f4']-df['f1']).astype('timedelta64[s]')), True)

# add a new column(travel_time) by subtracting f5 from f4

df.insert(1, "travel_time(s)", ((df["f5"]-df["f4"]).astype("timedelta64[s]")), True)
```

In [ ]:
```
#check head
df.head(3)
```

In [ ]:
```
#uniqueness
df.nunique()
```

In [ ]:
```
#null values
df.notnull().sum()
```

In [ ]:
```
#separating variables
cat_col=["f6","f7","f8","f9"]
num_col=["time_to_dep(s)","travel_time(s)"]
```

In [ ]:
```
# Plotting
# take insights through categorical_columns by countplot
c=1
plt.figure(figsize=(20,45))
for i in cat_col:

    plt.subplot(6,3,c)
    sns.countplot(df[i])
    c=c+1
```

In [ ]:
```
# Plotting
# take insights through numerical_columns by disttplot
c=1
plt.figure(figsize=(20,45))
for i in num_col:

    plt.subplot(6,3,c)
    sns.distplot(df[i])
    c=c+1
```

In [ ]:
```
# Plotting of target
# 1.distplot
sns.displot(df["target"])
```

Assignment#2

1.Remove outliers from dataset

In [10]:
```
# Detection
# IQR
Q1 = np.percentile(df['target'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(df['target'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['target'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['target'] <= (Q1-1.5*IQR))

''' Removing the Outliers '''
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)
```

```
C:\Users\mohsin.DESKTOP-7I5HD4K\AppData\Local\Temp\ipykernel_2396\2468351618.py:3: Deprecatio
nWarning: the `interpolation=` argument to percentile was renamed to `method=`, which has add
itional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to review the m
ethod they. (Deprecated NumPy 1.22)
  Q1 = np.percentile(df['target'], 25,
C:\Users\mohsin.DESKTOP-7I5HD4K\AppData\Local\Temp\ipykernel_2396\2468351618.py:6: Deprecatio
nWarning: the `interpolation=` argument to percentile was renamed to `method=`, which has add
itional options.
Users of the modes 'nearest', 'lower', 'higher', or 'midpoint' are encouraged to review the m
ethod they. (Deprecated NumPy 1.22)
  Q3 = np.percentile(df['target'], 75,
```

```
Old Shape:  (5000, 10)
New Shape:  (4749, 10)
```

In [ ]:
```
#shape
df.shape
```

In [ ]:
```
#Skewness and kurtosis
df.skew()
```

In [ ]:
```
#Kurtosis
df.kurtosis()
```

In [ ]:
```
df.head()
```

In [11]:
```
# encoding

df.drop(["f1","f4","f5"], axis=1, inplace=True)
```

In [12]:
```
#encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [13]:
```
# Apply label encoder
df["f6"]=le.fit_transform(df["f6"])
df["f7"]=le.fit_transform(df["f7"])
df["f8"]=le.fit_transform(df["f8"])
```

In [ ]:
```
#head
df.head()
```

In [14]:
```
df.to_csv("st_file.csv")
```

In [16]:
```
df3=pd.read_csv("st_file.csv")
df3.head()
```

Out[16]:

	Unnamed: 0	time_to_dep(s)	travel_time(s)	f6	f7	f8	f9	target
0	0	1268192.0	7200.0	2	1	0	0	7400.0
1	1	29688.0	7200.0	0	1	4	1	15377.0
2	2	3033072.0	7200.0	2	1	2	1	6900.0
3	3	363011.0	7200.0	0	1	1	1	9707.0
4	4	2142368.0	6900.0	1	1	2	0	6500.0

In [18]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4749 entries, 0 to 4748
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      4749 non-null   int64
 1   time_to_dep(s)  4749 non-null   float64
 2   travel_time(s)  4749 non-null   float64
 3   f6              4749 non-null   int64
 4   f7              4749 non-null   int64
 5   f8              4749 non-null   int64
 6   f9              4749 non-null   int64
 7   target          4749 non-null   float64
dtypes: float64(3), int64(5)
memory usage: 296.9 KB
```

In [19]:
```
df3["time_to_dep(s)"]=df3["time_to_dep(s)"].astype("int64")
df3["travel_time(s)"]=df3["travel_time(s)"].astype("int64")
df3["target"]=df3["target"].astype("int64")
```

In [20]:
```
df3.head()
```

Out[20]:

	Unnamed: 0	time_to_dep(s)	travel_time(s)	f6	f7	f8	f9	target
0	0	1268192	7200	2	1	0	0	7400
1	1	29688	7200	0	1	4	1	15377
2	2	3033072	7200	2	1	2	1	6900
3	3	363011	7200	0	1	1	1	9707
4	4	2142368	6900	1	1	2	0	6500

In [21]:
```
df3.to_csv("st3.file.csv")
```

In [ ]:
```
##Shapiro test to check normality
from scipy.stats import shapiro
stat,p=shapiro(df["f9"])
print("stat=",stat)
print("p=",p)

if  p > 0.05:
    print("data is normal")
else:
    print("data is not normal")
```

Assignment#3

In [ ]:
```
# #As data is not normal go for scaling
# from sklearn.preprocessing import StandardScaler
# sc=StandardScaler()
# df[["time_to_dep(s)"]]=sc.fit_transform(df[["time_to_dep(s)"]])
# df[["travel_time(s)"]]=sc.fit_transform(df[["travel_time(s)"]])
# df[["target"]]=sc.fit_transform(df[["target"]])
```

If we apply scaling it will give values in minus

In [ ]:
```
#head
df.head()
```

In [ ]:
```
#again normality
stat,p=shapiro(df["travel_time(s)"])
print("stat=",stat)
print("p=",p)

if  p > 0.05:
    print("data is normal")
else:
    print("data is not normal")
```

In [ ]:
```
#columns
df.columns
```

In [ ]:
```
# split into X,y
X=df[['time_to_dep(s)','travel_time(s)', 'f6', 'f7', 'f8', 'f9']]
y=df['target']
```

ML Modelling

In [ ]:
```
# Because our target(y) is numerical variable so we choose regression model/pipeline
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error


#rootmeansquarederror
# (rms)=mean_squared_error(y_test,y_pred, squared=False)
```

metrics for regression

r2_score,mean_absolute_error,mean_squared_error

metrics for classification

F1,Recall score,precision score

In [ ]:
```
# shortening the model names
lr=LinearRegression()
dt=DecisionTreeRegressor()
svr=SVR()
knn=KNeighborsRegressor()
```

In [ ]:
```
# model loop
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

for i in [lr,dt,svr,knn]: #read all model
    i.fit(X_train,y_train) #fitting our model
    pred=i.predict(X_test)  #predict
    test_score=r2_score(y_test,pred) #test score
    train_score=r2_score(y_train,i.predict(X_train)) #train score
    if abs(train_score-test_score) <=0.1:
        print(i)
        print("R2 score is:",r2_score(y_test,pred))
        print("Mean absolute error is:",mean_absolute_error(y_test,pred))
        print("Mean squared error is:",mean_squared_error(y_test,pred))
        print("RMSE is:",mean_squared_error(y_test,pred))
        print("---------------------------------------")
#to save prediction
res=pd.DataFrame(pred)
res.index=X_test.index # its important for comparison
res.columns=["prediction"]
res.to_csv("prediction_results_with_traintestsplit.csv")
```

Assignment#4

Now apply all modification steps df_test

In [ ]:
```
# as columns f1,f4,f5 shows date and time with dtype object
# we convert dtype to datetime
#for that import datetime library
from datetime import date,time
df_test["f1"]=pd.to_datetime(df_test["f1"])
df_test["f4"]=pd.to_datetime(df_test["f4"])
df_test["f5"]=pd.to_datetime(df_test["f5"])
```

In [ ]:
```
# add a new column(time_to_dparture) by subtracting f1 from f4

df_test.insert(0,"time_to_dep(s)" , ((df_test['f4']-df_test['f1']).astype('timedelta64[s]'
)), True)

# add a new column(travel_time) by subtracting f5 from f4

df_test.insert(1, "travel_time(s)", ((df_test["f5"]-df_test["f4"]).astype("timedelta64[s]"
)), True)
```

In [ ]:
```
#encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [ ]:
```
# Apply label encoder
df_test["f6"]=le.fit_transform(df_test["f6"])
df_test["f7"]=le.fit_transform(df_test["f7"])
df_test["f8"]=le.fit_transform(df_test["f8"])
```

In [ ]:
```
#dropping unnecessaary columns
df_test.drop(["Unnamed: 0","Unnamed: 0.1","f1","f2","f3","f4","f5","f10"], axis=1,inplace=Tr
ue)
```

In [ ]:
```
#shape
df_test.shape
```

In [ ]:
```
#head
df_test.head(2)
```

In [ ]:
```
df.head(2)
```

In [ ]:
```
#final data prediction
lr=LinearRegression().fit(X,y)
pred=lr.predict(df_test)
res=pd.DataFrame(pred)
res.index=df_test.index # its important for comparison
res.columns=["prediction"]
res.to_csv("prediction_results_with_traintestsplit2.csv")
```

In [ ]:
```
#head
df_test.head(2)
```

In [ ]:
```
#predicted value
pred1=lr.predict([[420706.0,7200.0,3,0,1,1]])
pred1
```