Restaurant Tip Prediction

January 22, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt,seaborn as sns
%matplotlib inline
import warnings
```

[2]:
```
temp=pd.read_excel("Restaurant _tips dataset.xlsx",sheet_name="Sheet1")
temp.head()
```

[2]:

	sex	smoker	day	time	size	total_bill	tip
0	Female	No	Sun	Dinner	2	16.99	1.01
1	Male	No	Sun	Dinner	3	10.34	1.66
2	Male	No	Sun	Dinner	3	21.01	3.50
3	Male	No	Sun	Dinner	2	23.68	3.31
4	Female	No	Sun	Dinner	4	24.59	3.61

[3]:
```
temp.to_csv('tips_prediction.csv',index=False)
```

[4]:
```
inp0=pd.read_csv('tips_prediction.csv')
inp0.head()
```

[4]:

	sex	smoker	day	time	size	total_bill	tip
0	Female	No	Sun	Dinner	2	16.99	1.01
1	Male	No	Sun	Dinner	3	10.34	1.66
2	Male	No	Sun	Dinner	3	21.01	3.50
3	Male	No	Sun	Dinner	2	23.68	3.31
4	Female	No	Sun	Dinner	4	24.59	3.61

[5]:
```
inp0.info()                                          #getting info and shape of of␣
 ↪dataset.
print(inp0.shape)
```

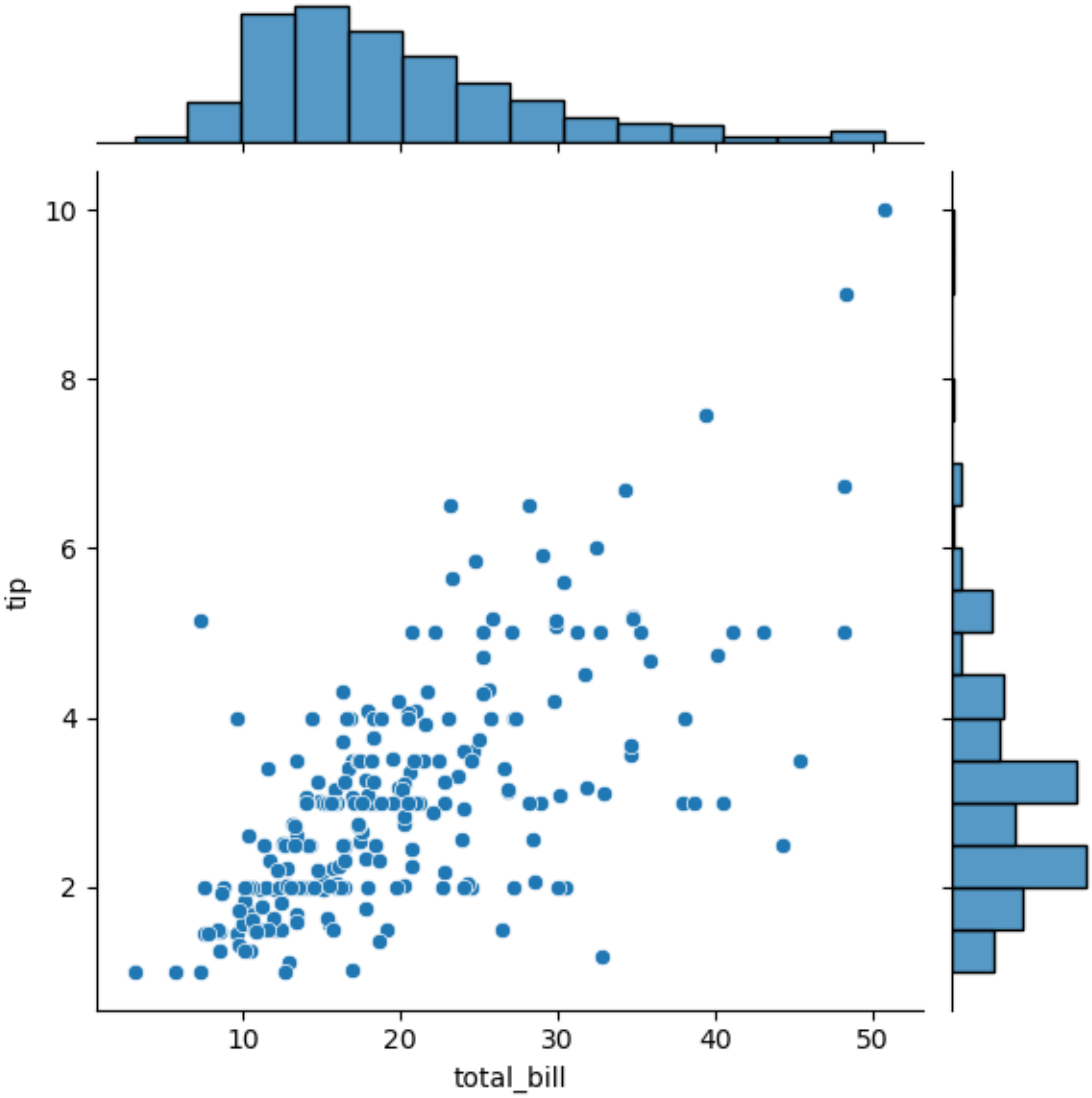
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sex         244 non-null    object
```

```
 1   smoker      244 non-null    object
 2   day         244 non-null    object
 3   time        244 non-null    object
 4   size        244 non-null    int64
 5   total_bill  244 non-null    float64
 6   tip         244 non-null    float64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
(244, 7)
```

[6]:
```
inp0.isnull().sum()              #checking for blank rows or null value in␣
↪dataset
```

[6]:
```
sex           0
smoker        0
day           0
time          0
size          0
total_bill    0
tip           0
dtype: int64
```

[8]:
```
sns.jointplot(x=inp0.total_bill,y=inp0.tip,kind='scatter');      #ploting to␣
↪observe linear relation between bill and tip
```

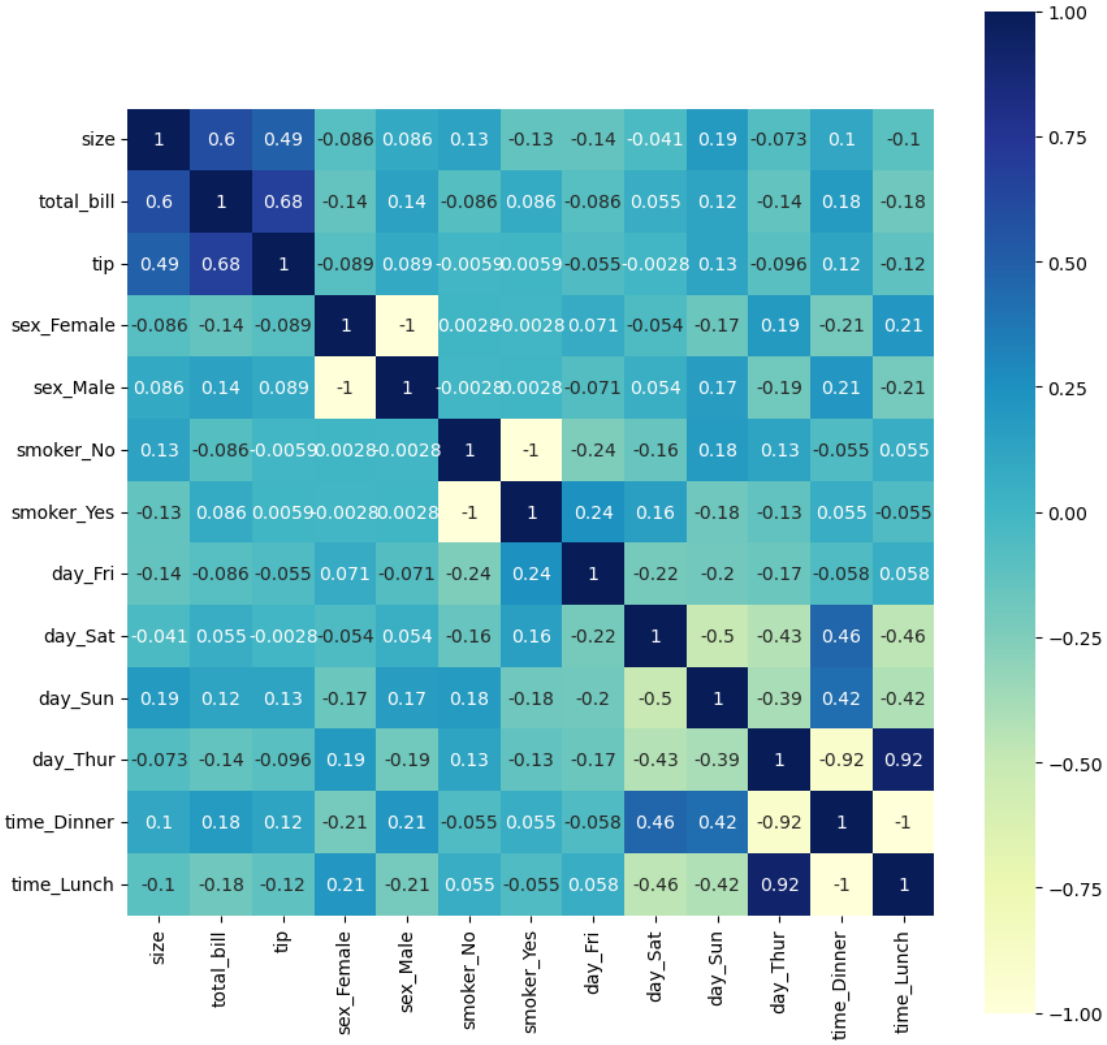


[9]:
```
inp1=pd.get_dummies(inp0)                    #getting dummies for catagorical␣
 ↪variables to evaluate
print(inp1.shape)
inp1.columns
```

```
(244, 13)
```

[9]:
```
Index(['size', 'total_bill', 'tip', 'sex_Female', 'sex_Male', 'smoker_No',
       'smoker_Yes', 'day_Fri', 'day_Sat', 'day_Sun', 'day_Thur',
       'time_Dinner', 'time_Lunch'],
      dtype='object')
```

[30]:
```
corr = inp1.corr()                                  #checking for correaltion between␣
 ↪columns
plt.figure(figsize=[10,10])
sns.heatmap(corr, square = True,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values,
            cmap="YlGnBu", annot=True);
```



[10]:
```
df_train=inp1.drop(['tip','sex_Female', 'sex_Male', 'smoker_No',                    ␣
 ↪#droping these rows as they have negative corr with tip
       'smoker_Yes', 'day_Fri', 'day_Sat', 'day_Sun', 'day_Thur',
       'time_Dinner', 'time_Lunch'],axis=1)
display(df_train.columns)
df_test=inp1['tip']
```

```
df_test
```

```
Index(['size', 'total_bill'], dtype='object')
```

[10]:
```
0      1.01
1      1.66
2      3.50
3      3.31
4      3.61
       …
239    5.92
240    2.00
241    2.00
242    1.75
243    3.00
Name: tip, Length: 244, dtype: float64
```

[11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_train,df_test, test_size␣
↪= 0.20, random_state = 1)
display ("X Train Head",X_train.head())
display ("YTrain Head",y_train.head())
display ("X Test Head",X_test.head())
display ("Y Test Head",y_test.head())
```

```
'X Train Head'
```

	size	total_bill
0	2	16.99
154	4	19.77
167	4	31.71
110	2	14.00
225	2	16.27

```
'YTrain Head'
```

```
0      1.01
154    2.00
167    4.50
110    3.00
225    2.50
Name: tip, dtype: float64
```

```
'X Test Head'
```

```
     size  total_bill
67      1        3.07
243     2       18.78
206     3       26.59
122     2       14.26
89      2       21.16
```

```
'Y Test Head'
```

```
67     1.00
243    3.00
206    3.41
122    2.50
89     3.00
Name: tip, dtype: float64
```

[12]:
```
X_train.to_csv("X_train.csv")            #savinbg different test dataset
X_test.to_csv("X_test.csv")
y_train.to_csv("Y_train.csv")
y_test.to_csv("Y_test.csv")
```

[22]:
```
from scipy import stats
xp=stats.pearsonr(inp1['size'],inp1['tip'])           #crosschecking relation␣
 ↪with pearson method
xp
```

[22]:
```
PearsonRResult(statistic=0.489298775230357, pvalue=4.3005433272249695e-16)
```

[23]:
```
from scipy import stats
xp=stats.pearsonr(inp1['total_bill'],inp1['tip'])
xp
```

[23]:
```
PearsonRResult(statistic=0.6757341092113642, pvalue=6.6924706468640476e-34)
```

[13]:
```
from sklearn.linear_model import LinearRegression
linear_reg=LinearRegression()
```

[14]:
```
linear_reg.fit(X_train,y_train)                                    #using Linear␣
 ↪regression as a technique
print(round(linear_reg.intercept_,3))
print(np.round(linear_reg.coef_,3))
```

```
0.767
[0.254 0.078]
```

[15]:
```
y_pred=linear_reg.predict(X_train)                    #Predicting on train␣
↪data set
y_pred[:10]
```

[15]:
```
array([2.59438677, 3.31766875, 4.24576464, 2.36197415, 2.53842119,
       1.86139311, 4.42570497, 3.64277665, 2.96049495, 2.33030147])
```

[16]:
```
y_pred=linear_reg.predict(X_test)                     #Predicting on test␣
↪data set
y_pred[:10]
```

[16]:
```
array([1.25878921, 2.73352343, 3.59419085, 2.38218394, 2.91852077,
       1.87538451, 2.14743942, 3.15190729, 2.39928454, 3.69232722])
```

[20]:
```
from sklearn.metrics import r2_score   # R2 score is not good so we can say its␣
↪not a good model to evaluate tips and we would need more data
print (r2_score(y_test, y_pred))
```

```
0.5077599375375339
```

[ ]:
```

```