Waiter tips prediction

Submitted By :
A.S.K Viswas(AM.EN.U4CSE20012)
Atla Suraj Reddy(AM.EN.U4CSE20013)
Nandipati Sai Sandeep(AM.EN.U4CSE20049)
Padala Laxmi Praneeth(AM.EN.U4CSE20052)
CGS Pranav Advaith(AM.EN.U4CSE20056)

ABSTRACT

Informal description (ill posed problem)

In this project we will predict tips given to a waiter for his/her service.

Formal description(Make it well posed problem)

- ***Task*** *(T): Analysing and predicting the tips for a waiter.*
- ***Experience*** *(E): A collection of various data sets that include some attribute related to the waiter, services, total bill, smoker, time etc.*
- ***Performance*** *(P): The number of successful predictions made out of the total number of requests received from the client.*

Assumptions:

- Not everyone gets the tip, so we must ensure that who gets the tips for their service.

1. INTRODUCTION

Motivation:

Giving a waiter or waitress a tip does a lot for them. You are signalling to them that you appreciate them and want to give something back. Their hard work makes your experience at the restaurant more enjoyable, So, this machine learning model will help in predicting the tip for the waiter.

Benefits of solution:

The benefits are both professional and personal. The personal benefit of this model is that a waiter can predict the amount of tip he gets from his customers for his service. Professionally, it benefits the manager of restaurant to know who the best waiter and the waiter is can get a hike in salary.

Solution Use

This model predicts the tips for the waiters. So, if we give a proper prediction of tips for waiter and improve their services.

2. Dataset finalization

Data sets:
https://www.kaggle.com/datasets/aminizahra/tips-dataset
https://www.kaggle.com/datasets/rupakroy/waiter-tips-dataset-for-prediction
https://raw.githubusercontent.com/amankharwal/Website-data/master/tips.csv

The above-mentioned data sets predict about the waiters tips according to their services. This can be predicted by total bill, tip, smoker, time, price per person and may contain information about customers.

Importing neccesary libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing datasets

In [2]:

```
path="tips.csv"
df=pd.read_csv(path)
```

Descriptive Statistics

In [3]:

```
# Preview data
df.head(10)
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	35603251
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	44780713
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	60118121
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	46761376
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	48327326
5	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	2131403
6	8.77	2.00	Male	No	Sun	Dinner	2	4.38	Kristopher Johnson	22237275
7	26.88	3.12	Male	No	Sun	Dinner	4	6.72	Robert Buck	35147850
8	15.04	1.96	Male	No	Sun	Dinner	2	7.52	Joseph Mcdonald	35228663
9	14.78	3.23	Male	No	Sun	Dinner	2	7.39	Jerome Abbott	35321245

In [4]:

```
# Dataset dimensions - (rows, columns)
df.shape
```

Out[4]:

```
(244, 11)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    int64  
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

In [6]:

```
df.isnull().sum()
#finding if our data has null values or not
```

Out[6]:

```
total_bill          0
tip                 0
sex                 0
smoker              0
day                 0
time                0
size                0
price_per_person    0
Payer Name          0
CC Number           0
Payment ID          0
dtype: int64
```

In [16]:

```
df = df.drop(columns=['price_per_person','CC Number'])
```

In [17]:

```
df.describe()
```

Out[17]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [18]:

```
correlation = df.corr()
correlation
```

Out[18]:

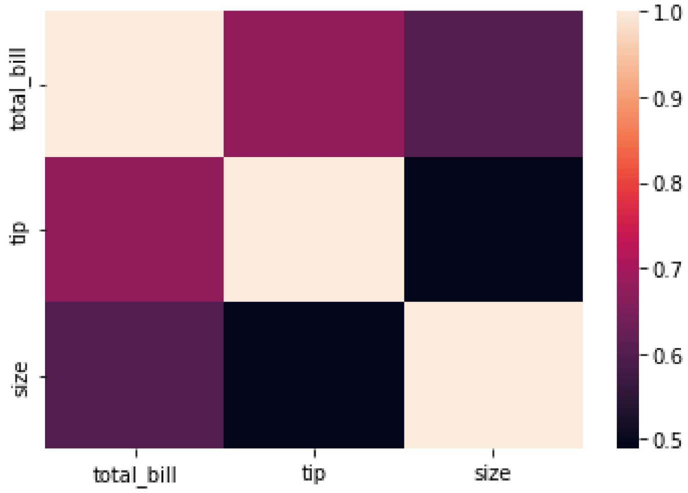
	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

Data Visualization

We will visualize correlation with heatmap, use count plots to see if the women or men come to the restaurant more than one another. Then let's see if the tip left really depends on the gender of the customer with box plot.

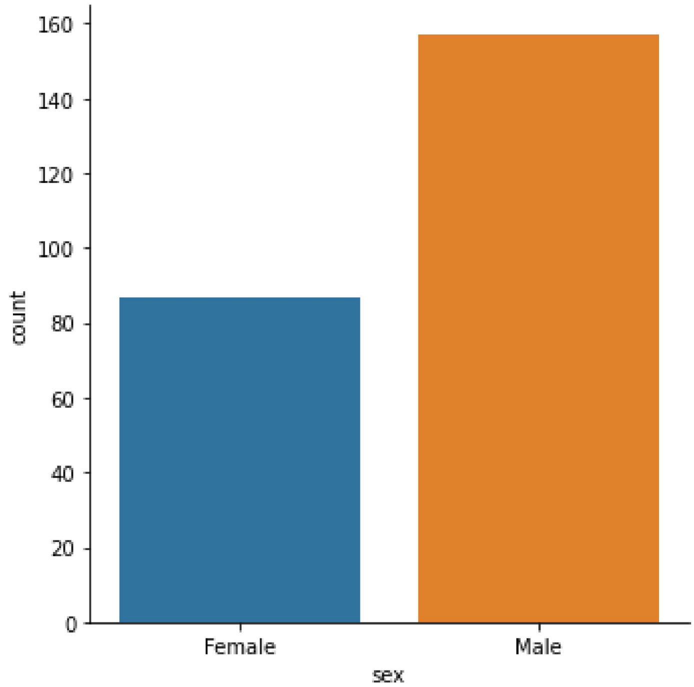
In [19]:

```
sns.heatmap(correlation)
plt.show()
```



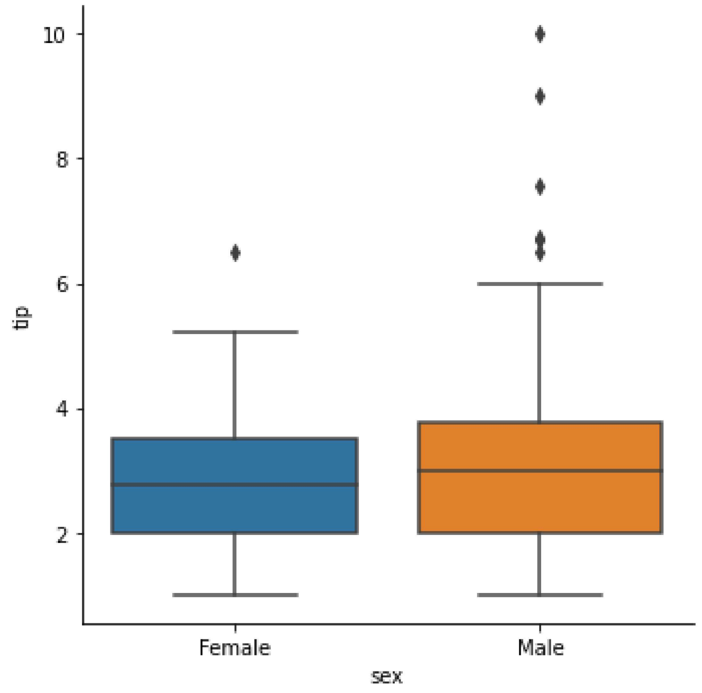
In [20]:

```
sns.catplot(x="sex",data=df,kind="count")
plt.show()
```



In [21]:

```
sns.catplot(x="sex",y="tip",data=df,kind="box")
plt.show()
```



Let's see how many days there are, then look at the relationship between the total bill paid and the tip, with respect to days.

In [22]:

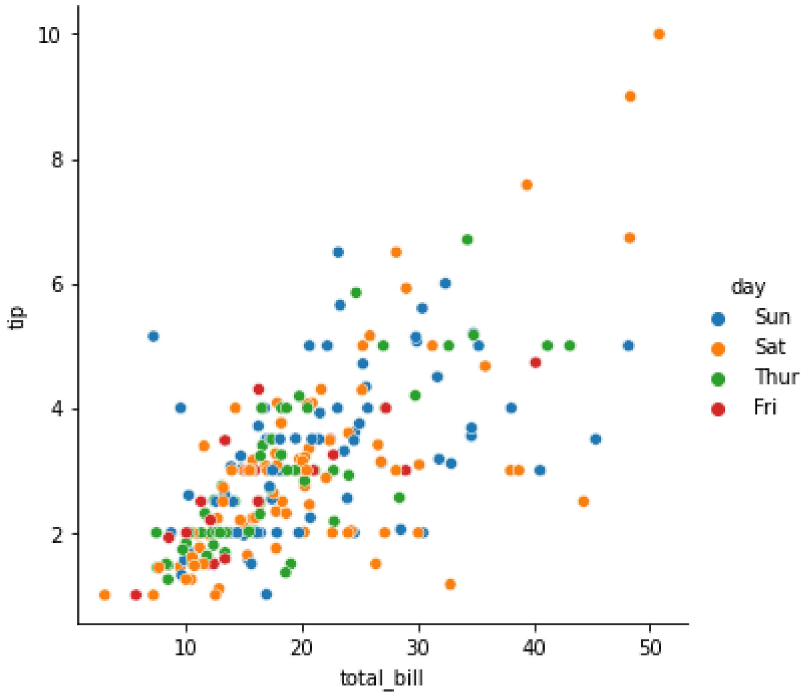
```
df.day.unique()
```

Out[22]:

```
array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```

In [23]:

```
sns.relplot(x="total_bill",y="tip",data=df,kind="scatter",hue="day")
plt.show()
```



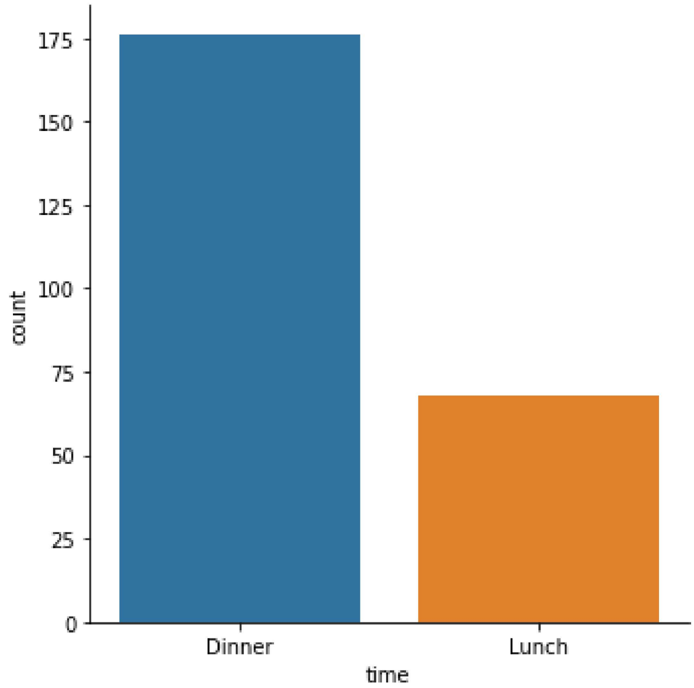
Let's see if people come over more in day time or in the evening.

In [24]:

```
sns.catplot(x="time",data=df,kind="count")
plt.show
```

Out[24]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Let's see the correlation between the tip column with others.

In [25]:

```
df.corrwith(df["tip"])
```

Out[25]:

```
total_bill    0.675734
tip           1.000000
size          0.489299
dtype: float64
```

Preprocessing

We want to predict the tip from other columns, therefore, We have to scale the numerical columns and encode categorical columns. For binary ones, you have to either use label encoding or one hot encode them, then drop duplicate ones.

In [26]:

```
columns_to_encode = ["sex","smoker","time"]
columns_to_scale = ["total_bill","tip","size"]
```

There are two unique values in sex, time and smoker columns and four values in day column, so we have to encode sex, time and smoker with label encoder and encode day with one hot encoder.

Now we are going to encode the day column with pandas' get_dummies method then directly append it to the main dataframe with pd.concat.

In [27]:

```
df = pd.concat([df, pd.get_dummies(df["day"],prefix="day")], axis=1)
df.head()
```

Out[27]:

	total_bill	tip	sex	smoker	day	time	size	Payer Name	Payment ID	day_Fri	day_Sat
0	16.99	1.01	Female	No	Sun	Dinner	2	Christy Cunningham	Sun2959	0	0
1	10.34	1.66	Male	No	Sun	Dinner	3	Douglas Tucker	Sun4608	0	0
2	21.01	3.50	Male	No	Sun	Dinner	3	Travis Walters	Sun4458	0	0
3	23.68	3.31	Male	No	Sun	Dinner	2	Nathaniel Harris	Sun5260	0	0
4	24.59	3.61	Female	No	Sun	Dinner	4	Tonya Carter	Sun2251	0	0

We will import Label Encoder from sklearn, instantiate it, and fit and transform the columns that We want to label, then convert the output array into dataframe and insert it to the original one.

In [29]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["sex"]=pd.DataFrame(le.fit_transform(df["sex"]))
```

In [30]:

```
df["time"]=pd.DataFrame(le.fit_transform(df["time"]))
df["smoker"]=pd.DataFrame(le.fit_transform(df["smoker"]))
```

Now we will import MinMaxScaler to normalize the numerical columns, and put them in another dataframe called scaled_columns, then drop the original columns and append them to the original dataframe.

In [31]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_columns = pd.DataFrame(scaler.fit_transform(df[columns_to_scale]),columns=columns_to
scaled_columns.describe()
```

Out[31]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	0.350145	0.222031	0.313934
std	0.186477	0.153738	0.190220
min	0.000000	0.000000	0.000000
25%	0.215281	0.111111	0.200000
50%	0.308442	0.211111	0.200000
75%	0.441087	0.284722	0.400000
max	1.000000	1.000000	1.000000

In [32]:

```
df.drop(["total_bill","tip","size", "day"],axis=1,inplace=True)
df = pd.concat([df, scaled_columns], axis=1)
```

Let's see the last version of the dataframe. Everything should be numeric.

In [33]:

```
df.head()
```

Out[33]:

	sex	smoker	time	Payer Name	Payment ID	day_Fri	day_Sat	day_Sun	day_Thur	total_bill
0	0	0	0	Christy Cunningham	Sun2959	0	0	1	0	0.291579
1	1	0	0	Douglas Tucker	Sun4608	0	0	1	0	0.152283
2	1	0	0	Travis Walters	Sun4458	0	0	1	0	0.375786
3	1	0	0	Nathaniel Harris	Sun5260	0	0	1	0	0.431713
4	0	0	0	Tonya Carter	Sun2251	0	0	1	0	0.450775

In [34]:

```
correlation = df.corr().abs()
plt.figure(figsize=(8,8))
sns.heatmap(correlation, annot=True)
plt.show()
```