**Waiter Tips Prediction with Machine Learning**

**Code Snippet #1: Loading Dataset**

import pandas as pd

import numpy as np

import plotly.express as px

import plotly.graph\_objects as go

data = pd.read\_csv("./dataset/tips.csv")

data.head()

**Function:**

This code imports several libraries: pandas for data handling, numpy for numerical operations (though not used directly here), and plotly.express/plotly.graph\_objects for creating visualizations. Then, the code reads a CSV file named "tips.csv" located in the ./dataset/ directory. The pd.read\_csv() function loads the CSV file into a DataFrame, which is a table-like structure in pandas used to store data. Lastly, data.head() displays the first five rows of the DataFrame, which helps you get a quick look at the data.

**Figure #1:**

****

**Description:**

The output will be a display of the first five rows of the dataset in a tabular format. This table will show the column names (such as total bill, tip, gender, and others) and their corresponding values for the first five entries in the dataset. It gives you a quick summary of what the data looks like, allowing you to understand its structure and get familiar with the values you will be working with for analysis or visualization.

**Code Snippet#2: Display Scatter Plot Classify by Days of the week**

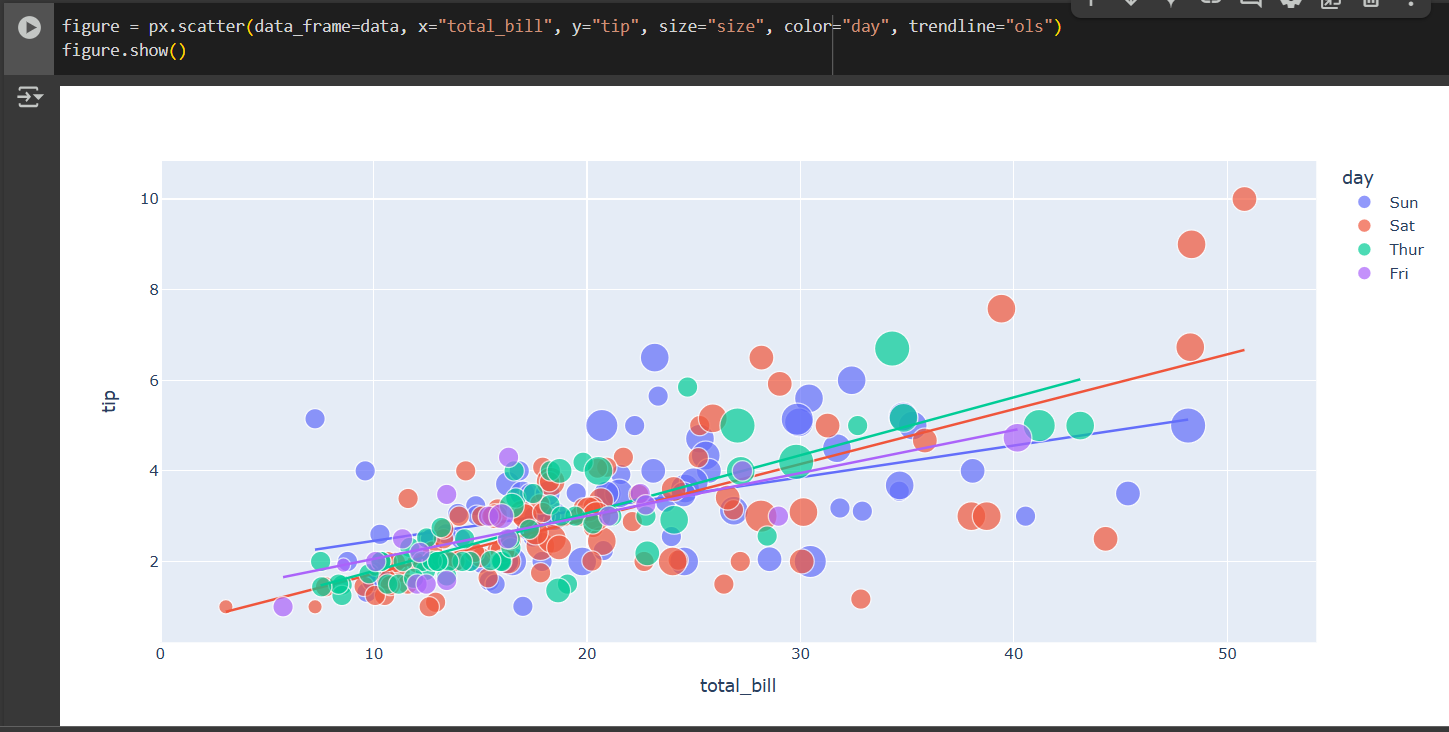
figure = px.scatter(data\_frame=data, x="total\_bill", y="tip", size="size", color="day", trendline="ols")

figure.show()

**Function:**

This code generates a scatter plot using the plotly.express library. It uses the px.scatter() function to create the plot based on the data DataFrame. The x-axis represents the total\_bill column, and the y-axis represents the tip column. The size of each point on the plot corresponds to the size column, likely indicating the number of people in the group. The color of each point is determined by the day column, which differentiates data based on the day of the week. Additionally, the code includes a trendline using Ordinary Least Squares (OLS), which fits a linear regression line to show the relationship between total bill and tip.

**Figure #2**



**Description:**

The output is an interactive scatter plot where each point represents a record from the dataset. The x-axis shows the total bill amounts, and the y-axis shows the tip amounts. The points vary in size depending on the group size and are colored based on the day of the week. A linear trendline is included to show the general pattern of how tips increase with total bills. This plot allows for an interactive exploration where you can hover over individual points to see detailed information and visually analyze the relationship between the two variables.

**Code Snippet#3: Display Scatter Plot Classify by Sex**

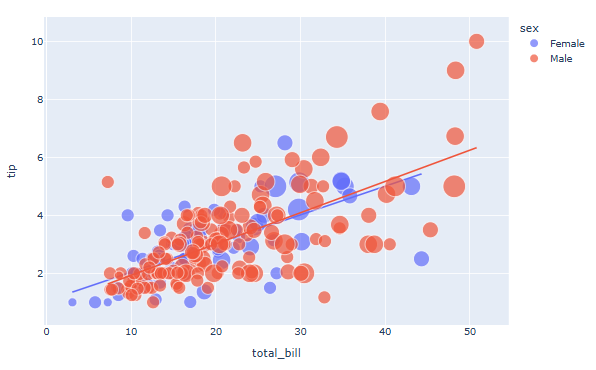
figure = px.scatter(data\_frame=data, x="total\_bill", y="tip", size="size", color="sex", trendline="ols")

figure.show()

**Function:**

This code creates a scatter plot using the plotly.express library. The px.scatter() function generates the plot based on the data DataFrame. In this case, the x-axis represents the total\_bill column, while the y-axis represents the tip column. The size of each point corresponds to the size column, which likely indicates the number of people in the group. The color of the points is determined by the sex column, which differentiates the data based on gender. Additionally, a trendline is added using the Ordinary Least Squares (OLS) method to fit a linear regression line, showing the relationship between the total bill and tip.

**Figure #3**



**Description:**

The output will be an interactive scatter plot where each point represents a record from the dataset. The x-axis will display total bill amounts, and the y-axis will show tip amounts. The points will vary in size based on the group size and will be colored according to gender (male or female). The plot will also include a linear trendline that highlights the general relationship between the total bill and the tip. This visualization allows you to explore how tips correlate with the total bill and how the data is distributed across genders.

**Code Snippet#4: Display Scatter Plot Classify by Time**

figure = px.scatter(data\_frame=data, x="total\_bill", y="tip", size="size", color="time", trendline="ols")

figure.show()

**Function:**

This code generates a scatter plot using the plotly.express library. The plot maps total\_bill to the x-axis and tip to the y-axis. The size of each point is determined by the size column, which likely represents the number of people in the group. The points are colored based on the time column, which likely differentiates between lunch and dinner times. A trendline is also included using Ordinary Least Squares (OLS) to visualize the linear relationship between the total bill and the tip.

**Figure #4**



**Description:**

The output will be an interactive scatter plot where each point represents a record from the dataset. The x-axis shows the total bill amounts, and the y-axis shows the corresponding tip amounts. The points' size is based on the group size, and the color indicates whether the record corresponds to lunch or dinner. A linear trendline will also be shown to highlight the overall pattern between the total bill and tip. This allows you to visually compare the relationship for lunch vs. dinner times, along with the group size influence on the tips.

**Code Snippet#5: Code snippet is for creating a donut chart by day**

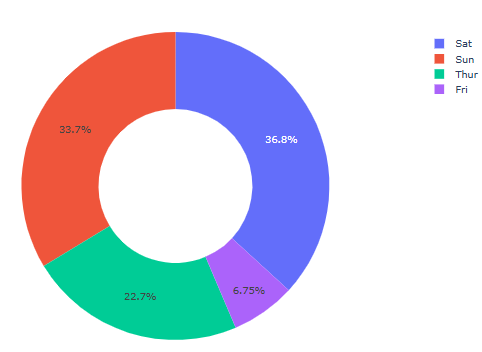
figure = px.pie(data, values="total\_bill", names="day", hole=0.5)

figure.show()

**Function:**

This code creates a pie chart using the plotly.express library. It plots the total\_bill values, with each slice representing a day of the week, based on the day column in the data DataFrame. The values parameter is set to the total\_bill, and the names parameter is set to day, which means each slice will correspond to a day. The hole=0.5 creates a donut chart by making the center of the pie chart empty.

**Figure #5**

****

**Description:**

The output will be an interactive donut chart where each slice represents a day of the week. The size of each slice corresponds to the total bill amount for that specific day. The chart will be a donut-style visualization, with a hole in the center, making it easier to see the proportions between the days. This allows for a clear view of how the total bills are distributed across the different days of the week.

**Code Snippet#6: Code snippet is for creating a donut chart by Sex**

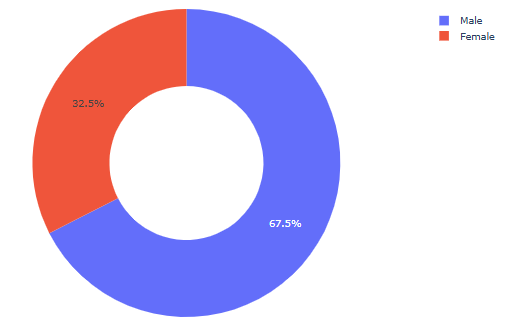
figure = px.pie(data, values="total\_bill", names="sex", hole=0.5)

figure.show()

**Function:**

This code generates a pie chart using the plotly.express library. It plots the total\_bill values with each slice representing gender, based on the sex column in the data DataFrame. The values parameter is set to the total\_bill, and the names parameter is set to sex, so each slice of the pie will represent a gender (male or female). The hole=0.5 creates a donut chart by leaving a hole in the center.

**Figure #6**

****

**Description:**

The output will be an interactive donut chart where each slice represents either male or female. The size of each slice corresponds to the total bill amount for each gender. With the hole in the middle, the chart offers a cleaner visual display of the proportions between males and females. This allows you to quickly compare the total bill distribution between genders.

**Code Snippet#7: Code snippet is for creating a donut chart by smoker category**

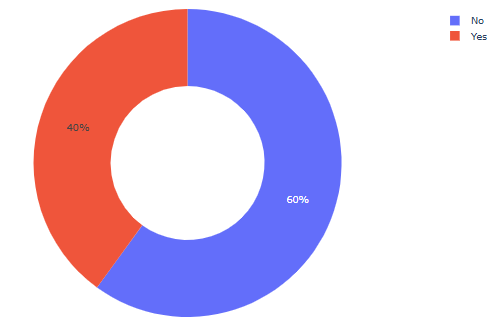
figure = px.pie(data, values="total\_bill", names="smoker", hole=0.5)

figure.show()

**Function:**

This code creates a pie chart using the plotly.express library. It visualizes the total\_bill values, with each slice representing whether a customer is a smoker or not, based on the smoker column in the data DataFrame. The values parameter is set to total\_bill, and the names parameter is set to smoker, so each slice will represent smokers and non-smokers. The hole=0.5 creates a donut chart by leaving a hole in the center.

**Figure #7**

****

**Description:**

The output will be an interactive donut chart where each slice represents either smokers or non-smokers. The size of each slice corresponds to the total bill amount for each group. With the hole in the center, the chart provides a clearer visual of the proportions between the two groups. This allows for an easy comparison of total bill distribution between smokers and non-smokers.

**Waiter Tip Prediction Model**

**Code Snippet#8: Code Snippet for Converting Categorical Data to Numerical Representation**

# converting to nemerical representation

data["sex"]  =data["sex"].map({"Male":1, "Female":0})

data["smoker"]  =data["smoker"].map({"No":0, "Yes":1})

data["day"]  =data["day"].map({"Thur":0, "Fri":1, "Sat":2, "Sun":3})

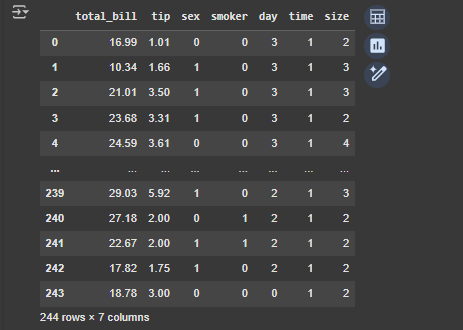
data["time"]  =data["time"].map({"Lunch":0, "Dinner":1})

data

**Function:**

This code converts categorical data into numerical values using the map() function. The sex column, which contains "Male" and "Female", is mapped to 1 and 0, respectively. The smoker column, which has "No" and "Yes" values, is mapped to 0 and 1. The day column, representing the days of the week, is mapped to numbers: "Thur" becomes 0, "Fri" becomes 1, "Sat" becomes 2, and "Sun" becomes 3. Finally, the time column, which indicates either "Lunch" or "Dinner", is mapped to 0 and 1.

**Figure #8**

****

**Description:**

The output will be the data DataFrame where categorical variables, such as sex, smoker, day, and time, have been converted to numerical values. For instance, "Male" will become 1 and "Female" will become 0 in the sex column. Similarly, "Yes" for smoker becomes 1, and "No" becomes 0. The days of the week (day) and times of day (time) will also be represented as numbers, simplifying the data for further analysis or machine learning. This transformation makes the dataset more suitable for algorithms that require numerical inputs.

**Code Snippet#9: Code Snippet for Training a Linear Regression Model**

x = np.array(data[["total\_bill", "sex", "smoker", "day", "time", "size"]])

y = np.array(data["tip"])

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(xtrain, ytrain)

**Function:**

**This code performs data preparation and trains a linear regression model. First, it converts the selected columns (total\_bill, sex, smoker, day, time, and size) into a NumPy array x, which will be used as the input features for the model. The target variable tip is also converted into a NumPy array y. Then, the code splits the dataset into training and testing sets using train\_test\_split() from sklearn.model\_selection, with 80% of the data used for training and 20% for testing. Finally, it initializes a linear regression model and fits it to the training data (xtrain, ytrain).**

**Figure #9**

****

**Description:**

**The output of this code will be a trained linear regression model that has learned the relationship between the features (such as total\_bill, sex, smoker, etc.) and the target variable (tip). The dataset is first split into two parts: one for training the model and the other for testing its performance. The model is then trained using the training data, allowing it to make predictions based on the input features. The trained model can now be evaluated on the test data to assess its accuracy and performance in predicting tips based on the provided features.**

**Code Snippet#10: Code Snippet for Making Predictions with the Trained Model**

features = np.array([[24.50, 1, 0, 0, 1, 4]])

model.predict(features)

**Function:**

This code uses the trained linear regression model to make a prediction. The features array is created with a new data point: total\_bill = 24.50, sex = 1 (Male), smoker = 0 (Non-smoker), day = 0 (Thursday), time = 1 (Dinner), and size = 4. The model.predict(features) function then takes this input and predicts the corresponding tip value based on the learned relationship between the input features and the target variable (tip) from the training data.

**Figure #10**

****

**Description:**

The output value, 3.73842609, is the model's best guess for the tip amount, based on the features provided. The model has considered factors like the total bill, customer gender, smoking status, day of the week, time, and group size to generate this prediction. The prediction indicates that, for these particular characteristics, the expected tip is approximately $3.74. This value is based on the linear regression equation learned during training.