#insert data set to google colab

import pandas as pd

df=pd.read\_csv('/content/Restaurant\_Reviews.csv')

df

from matplotlib import pyplot as plt

df['Liked'].plot(kind='line', figsize=(8, 4), title='Liked')

plt.gca().spines[['top', 'right']].set\_visible(False)

from matplotlib import pyplot as plt

df['Liked'].plot(kind='hist', bins=20, title='Liked')

plt.gca().spines[['top', 'right',]].set\_visible(False)

df.size

df.shape

df.Liked

#input and output

x=df.iloc[:,0].values

y=df.iloc[:,1].values

print(x)

print(y)

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

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

# Assuming 'X' is your text data and 'y' is the binary target variable

x\_Train, x\_Test, y\_Train, y\_Test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

# Vectorize the text data using TF-IDF (you may need to preprocess the text)

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000) # You can adjust max\_features

x\_Train\_tfidf = tfidf\_vectorizer.fit\_transform(x\_Train)

x\_Test\_tfidf = tfidf\_vectorizer.transform(x\_Test)

# Create and train the logistic regression model

model = LogisticRegression()

model.fit(x\_Train\_tfidf, y\_Train)

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB # Import Multinomial Naive Bayes

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

# Assuming 'X' is your text data and 'y' is the binary target variable

x\_Train, x\_Test, y\_Train, y\_Test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

# Vectorize the text data using TF-IDF (you may need to preprocess the text)

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

x\_Train\_tfidf = tfidf\_vectorizer.fit\_transform(x\_Train)

x\_Test\_tfidf = tfidf\_vectorizer.transform(x\_Test)

# Create and train the Multinomial Naive Bayes model

model = MultinomialNB() # Use Multinomial Naive Bayes

model.fit(x\_Train\_tfidf, y\_Train)

# Now you can evaluate the model's performance, make predictions, etc.

from sklearn.metrics import accuracy\_score

# Assuming you have already trained your model and obtained predictions

y\_pred = model.predict(x\_Test\_tfidf) # X\_test\_tfidf should contain your test data

# Calculate the accuracy

accuracy = accuracy\_score(y\_Test, y\_pred)

print("Accuracy:", accuracy\*100)

# Preprocess your new text data and convert it to TF-IDF vectors

new\_text\_data = ["food is good"] # Replace with your new text data

new\_text\_data\_tfidf = tfidf\_vectorizer.transform(new\_text\_data)

# Make predictions for the new data

predictions = model.predict(new\_text\_data\_tfidf)

print(new\_text\_data\_tfidf) # Check the contents of new\_text\_data\_tfidf

predictions = model.predict(new\_text\_data\_tfidf)

print(predictions)