Lab 8: Project Work(Semantic Analyzer)

```
import pickle
```

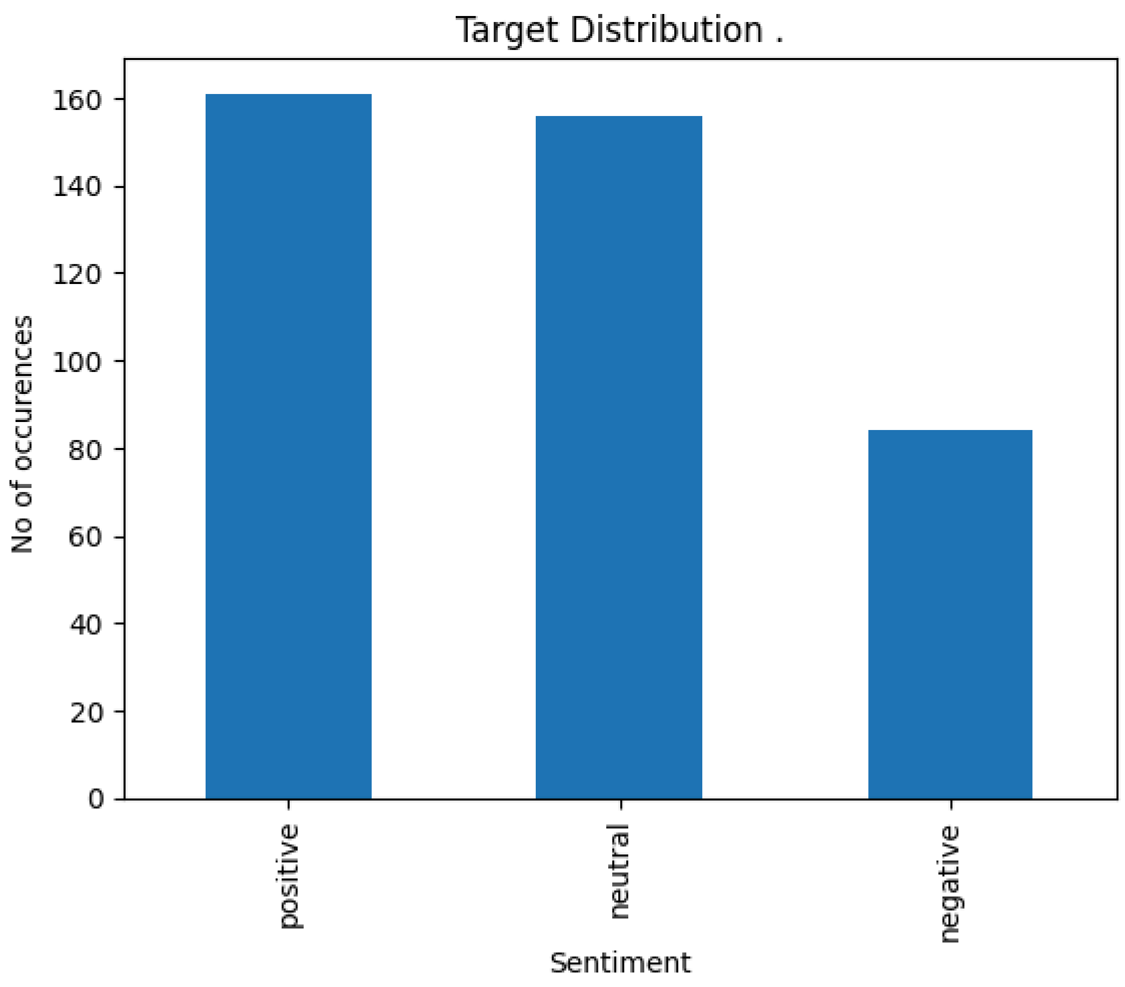
```
# Load the dataset
import numpy as np
import pandas as pd

df=pd.read_csv('samples.csv')
df.head()
```

	Index	Sentence	Sentiment
0	0	Weather feels pleasant today.	positive
1	1	Nabin travels a lot.	neutral
2	2	He missed the deadline.	negative
3	3	He is reading a book.	neutral
4	4	Team failed to meet expectations.	negative

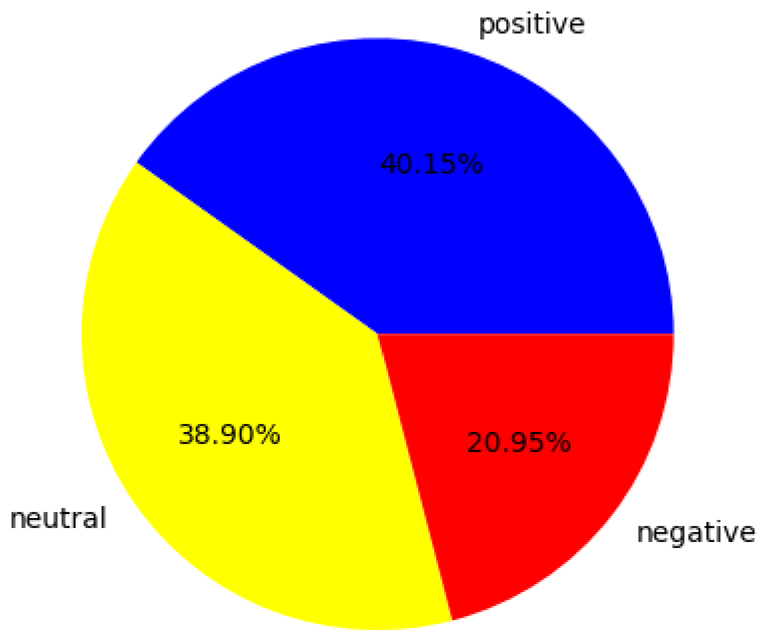
Next steps: Generate code with df   New interactive sheet

```
#Check if any null values
df.isna().sum()

# if there is any nul value, drop it.
df = df.dropna()
```

```
import matplotlib.pyplot as plt
value_counts_target = df['Sentiment'].value_counts()
value_counts_target.plot(kind='bar')
plt.xlabel('Sentiment')
plt.ylabel('No of occurences ')
plt.title('Target Distribution .')
plt.show()
```



```
plt.pie(value_counts_target,labels=value_counts_target.index,autopct="%
plt.show()
```



```
import pandas as pd
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
```

```
# Download NLTK data
nltk.download('stopwords')
from nltk.corpus import stopwords

# Preprocess the text data
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#Create a preprocessing function
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = ''.join([char for char in text if char.isalnum() or char.iss
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_
    return text
```

```
#Preprocess the sentences of dataset by using above function
df['Sentence'] = df['Sentence'].apply(preprocess_text)
```

```
# Vectorize the text data
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['Sentence'])
```

```
# Split the dataset into training and testing sets at 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X, df['Sentiment'],
```

```
X_train
```

```
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 832 stored elements and shape (320, 68)>
```

```
# Train a Naive Bayes classifier
from sklearn import svm
classifier =svm.SVC()
classifier.fit(X_train, y_train)
```

```
▾ SVC ⓘ ?
SVC()
```

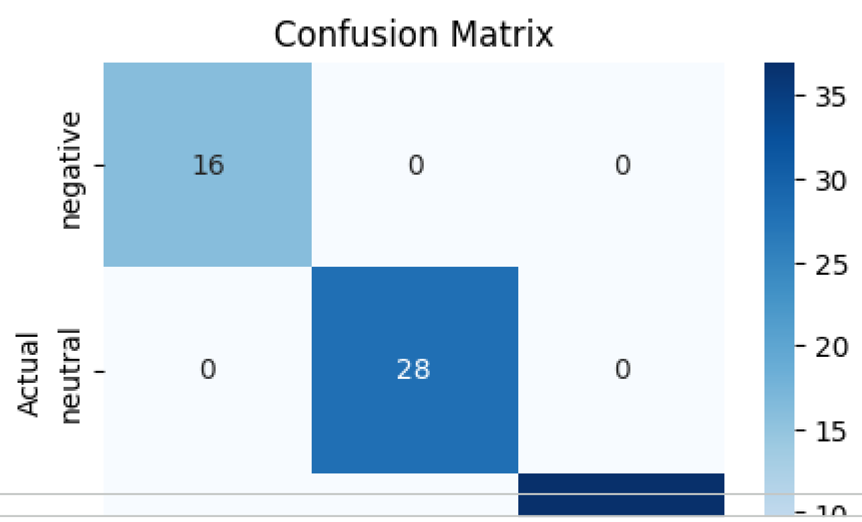
```
# Make predictions
y_pred = classifier.predict(X_test)
```

```
# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00        16
     neutral       1.00      1.00      1.00        28
    positive       1.00      1.00      1.00        37

    accuracy                           1.00        81
   macro avg       1.00      1.00      1.00        81
weighted avg       1.00      1.00      1.00        81
```

```
# Generate the confusion matrix

from sklearn.metrics import accuracy_score, classification_report, conf
import matplotlib.pyplot as plt
import seaborn as sns

conf_matrix = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



```
def predict_sentiment(text):
    # Preprocess the text using the same function you defined earlier
    processed_text = preprocess_text(text)

    # Convert to vector using the same fitted vectorizer
    text_vector = vectorizer.transform([processed_text])

    # Predict sentiment using the trained classifier
    prediction = classifier.predict(text_vector)

    return prediction[0]
```

```
# Test the function with a single text
sample_text = "Ram is happy."
predicted_sentiment = predict_sentiment(sample_text)
print(f'The predicted sentiment for the sample text is: {predicted_senti
```

```
The predicted sentiment for the sample text is: positive
```

Name: Nischal Bhandari

Exam Roll no: 22070136