



**VISHWAKARMA**  
**UNIVERSITY**  
*Maximising Human Potential*

**Activity based**  
**Project Report on**  
**Artificial Intelligence**  
**Project Phase - II**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice**  
**(C2P2)**

**By**

**Imaad Imran Hajwane**

**SRN No: 202101132**

**Roll No: 23**

**Div: A**

**Third Year Engineering**

**Faculty In charge: - Prof. Tarapore Sir**

**Date Of Project Phase 2: - 23<sup>rd</sup> February 2024**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024 Term-II**

### AI: Phase II

**Project Name: Product Review Sentiment Analysis for Restaurant Review with Food Images**

---

#### 1. Problem Statement and Objectives:

Clearly define the problem statement including the implementation plan for the given algorithm.

Implement the Sentiment Analysis system in a programming language, incorporating the designed data processing pipeline and sentiment classification model. Develop modules for aspect-based sentiment analysis, feature extraction, and machine learning algorithms for sentiment classification. Utilize a labeled dataset for training and validating the model. Ensure the system accurately analyzes product reviews, considering both overall sentiment and sentiments related to specific aspects of the product.

#### Plan:

##### 1. Understanding Requirements

- Review the provided statement to understand the requirements thoroughly.
- Clarify any ambiguities or uncertainties with stakeholders.

##### 2. Data Acquisition

- Identify and obtain a labeled dataset suitable for sentiment analysis.
- Ensure the dataset covers various aspects of products and includes both positive and negative sentiments.
- Preprocess the dataset to remove noise, handle missing values, and standardize text formats.

##### 3. Designing Data Processing Pipeline

- Develop a pipeline to preprocess raw text data before feeding it into the sentiment analysis model.
- Include steps such as tokenization, lowercasing, stop word removal, stemming or lemmatization, and handling special characters.

##### 4. Aspect-Based Sentiment Analysis Module

- Design a module to extract aspects or features mentioned in the reviews.
- Associate sentiment polarity with each aspect mentioned in the reviews.

### 5. Machine Learning Algorithms for Sentiment Classification

- Choose appropriate machine learning algorithms for sentiment classification, such as:
- Logistic Regression
- Support Vector Machines (SVM)
- Random Forest
- Split the dataset into training, validation, and testing sets.
- Train multiple models using different algorithms and hyperparameters.
- Evaluate models using appropriate metrics like accuracy, precision, recall, and F1-score.

### 6. Model Evaluation and Selection

- Compare the performance of different models using validation data.
- Select the best-performing model based on evaluation metrics.
- Fine-tune the selected model if necessary.

### 7. Testing and Validation

- Conduct thorough testing to ensure the system functions correctly under various scenarios.
- Validate the accuracy and effectiveness of sentiment analysis, both overall and for specific product aspects, using test datasets and real-world reviews.

## 2. Methodology details:

- Identify dataset

The dataset used for this machine learning project is a text-based dataset for restaurant reviews. It contains 7 distinct columns in the dataset.

The columns are:

<i>Columns Names</i>	<i>Descriptions:</i>
<i>Restaurant</i>	Name of the restaurant
<i>Reviewer</i>	Name of the reviewer
<i>Review</i>	Text of the review
<i>Rating</i>	Rating given by the reviewer
<i>Metadata</i>	Additional metadata related to the review (if any)
<i>Time</i>	Time of the review
<i>Pictures</i>	Number of pictures attached to the review (if any)

These are the columns present in the dataset, which are used to create this machine learning project, using these columns and their values we can achieve the processing result for various machine learning algorithms.

## Artificial Intelligence – Phase II

- Preprocess dataset

### Cleaning & Preparing Data

```
[8] 1 df = df.drop(columns=["Restaurant","Reviewer","Metadata","Time","Pictures"])
    2

[9] 1 y = df["Rating"]
    2 X = df.drop(columns=["Rating"])
    3 X.shape
    4
... (10000, 1)

[10] 1 y = y.replace({'Like':3})
    2

[11] 1 y.isnull().sum()
    2
... 38
```

```
[12] 1 y = pd.to_numeric(y)
    2

[13] 1 y = y.fillna(y.median())
    2

[14] 1 for i in range(0,len(y)):
    2 |     y.iloc[i] = round(y.iloc[i],0)
    3

[15] 1 for i in range(0,len(y)):
    2 |     if (y[i]>=3):
    3 |         y[i] = "Positive"
    4 |     else:
    5 |         y[i] = "Negative"
    6
```

```
1 # Assuming 'y' is a list of strings representing numbers
2 for i in range(len(y)):
3     try:
4         y[i] = int(y[i]) # Convert string to integer
5         if y[i] >= 3:
6             y[i] = 1
7         else:
8             y[i] = 0
9     except ValueError:
10        # Handle non-numeric values here
11        # For example, you can skip them or assign a default value
12        pass # Skipping for now
13
```

[16] Python

```
1 y.unique()
2
```

[17] Python

```
... array(['Positive', 'Negative'], dtype=object)
```

```
1 y.shape
2
```

[18] Python

```
... (10000,)
```

- Implement algorithm

For this ML Project, we have used:

- LSTM
- MultinomialNB
- Random Forest
- SVM
- KNN
- LR
- XGBoost

After using all these algorithms for the dataset, we have the results as follows:

<i><b>Models Name</b></i>	<b>Accuracy</b>	<b>Recall</b>	<b>Precision</b>	<b>F-Beta-Score</b>
<i>MultinomialNB</i>	0.9084	0.856	0.7684	0.8865
<i>Random Forest</i>	0.8924	0.855	0.8321	0.8064
<i>SVM</i>	0.891	0.80	0.785	0.777
<i>KNN</i>	0.8156	0.70	0.689	0.80
<i>LR</i>	0.77	0.50	0.38	0.43
<i>XGBoost</i>	0.90	0.83	0.87	0.85

From all these algorithms we came to a conclusion that the **XGBoost** algorithm is the best suitable as per the defined dataset.

- Verify output with expected output based on domain knowledge

```
XGBoost

1 import xgboost as xgb
2 from sklearn.metrics import accuracy_score, classification_report
3

[ ] Python

1 # Define the XGBoost model
2 model = xgb.XGBClassifier()
3

[ ] Python

1 from sklearn.metrics import classification_report
2
3 # Train the model
4 model.fit(X_train, y_train_encoded)
5
6 # Predictions
7 y_pred_train = model.predict(X_train)
8 y_pred_test = model.predict(X_test)
9
10 # Decode Labels
11 y_pred_train_decoded = label_encoder.inverse_transform(y_pred_train)
12 y_pred_test_decoded = label_encoder.inverse_transform(y_pred_test)
13
14 # Evaluate the model
15 train_accuracy = (y_pred_train_decoded == y_train).mean()
16 test_accuracy = (y_pred_test_decoded == y_test).mean()
17
18 print("Train Accuracy:", train_accuracy)
19 print("Test Accuracy:", test_accuracy)
20
21 # Classification report
22 print("\nClassification Report on Train Data:")
23 print(classification_report(y_train, y_pred_train_decoded))
24
25 print("\nClassification Report on Test Data:")
26 print(classification_report(y_test, y_pred_test_decoded))
27

[ ] Python

... Train Accuracy: 0.9506666666666667
Test Accuracy: 0.8968

Classification Report on Train Data:
      precision    recall  f1-score   support

   Negative       0.96       0.84       0.89       1863
   Positive       0.95       0.99       0.97       5637

 accuracy          0.95          0.95          0.95       7500
 macro avg          0.95          0.91          0.93       7500
weighted avg          0.95          0.95          0.95       7500

Classification Report on Test Data:
      precision    recall  f1-score   support

   Negative       0.83       0.71       0.76        584
   Positive       0.91       0.95       0.93       1916

 accuracy          0.90          0.90          0.90       2500
 macro avg          0.87          0.83          0.85       2500
weighted avg          0.89          0.90          0.89       2500
```

- Validation and testing

```
# Importing Libraries
import pandas as pd
import numpy as np
from sklearn.metrics import
classification_report, confusion_matrix, accuracy_score
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
import nltk

# Importing Dataset
df=pd.read_csv('Restaurant reviews.csv', encoding = "ISO-8859-1")
df = df.drop(columns=["Restaurant","Reviewer","Metadata","Time","Pictures"])

# Transforming & Cleaning Data
y = df["Rating"]
X = df.drop(columns=["Rating"])
y = y.replace({'Like':3})
y = y.fillna(y.median())
y = pd.to_numeric(y)
for i in range(0,len(y)):
    y.iloc[i] = round(y.iloc[i],0)

for i in range(0,len(y)):
    if (y[i]>=3):
        y[i] = "Positive"
    else:
        y[i] = "Negative"

# Applying Stemming with excluding StopWords
ps = PorterStemmer()
corpus = []
for i in range(0, len(X)):
    review = re.sub('[^a-zA-Z]', ' ', str(X['Review'][i]))
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in
stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)

# Creating Matrix of CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv = CountVectorizer(max_features=9000)
X = cv.fit_transform(corpus).toarray()

# Train-Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)

# Applying MultinomialNB
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(X_train, y_train)

# Making Predictions
y_pred = classifier.predict(X_test)

# Creating Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(y_test, y_pred)
print(confusion_m)

# Getting the Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

# Dumping Models
import pickle
pickle.dump(classifier, open('model.pkl', 'wb'))
pickle.dump(cv, open('cv-model.pkl', 'wb'))
```

The output of this code is the pickle file, which is used to make the model or we can say, it helps us to create an interface model which is used to integrate it in the frontend of the project

```
≡ cv-model.pkl
≡ model.pkl
```



## 3. Source code:

### Machine Learning Algorithms

#### Applying NLP Processes

```
1 import re
2 import nltk
3 from nltk.corpus import stopwords
4 from nltk.stem.porter import PorterStemmer
5 ps = PorterStemmer()
6 corpus = []
7 for i in range(0, len(X)):
8     review = re.sub('[^a-zA-Z]', ' ', str(X['Review'][i]))
9     review = review.lower() #Lowering the words is very important in avoiding classifying same words as different words
10    review = review.split()
11
12    review = [ps.stem(word) for word in review if not word in stopwords.words('english')] #Eliminating words that do not put much value in se
13    review = ' '.join(review) #Reconstructing sentences
14    corpus.append(review)
15
```

[19]

Python

#### For implementing LSTM goto direct LSTM Section Else Continue

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 cv = CountVectorizer(max_features=9000) #After experimenting with 7500, 5000, 2500 ...9000 worked best.
3 X = cv.fit_transform(corpus).toarray()
4
```

[20]

Python

#### Train-Test Split

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
3
```

[21]

Python

#### Deciding Best Model:

##### Trying Out MultinomialNB

```
1 from sklearn.naive_bayes import MultinomialNB
2 restaurant_review_model = MultinomialNB().fit(X_train, y_train)
3
```

[22]

Python

```
1 y_pred = restaurant_review_model.predict(X_test)
2
```

[23]

Python

```
1 from sklearn.metrics import confusion_matrix
2 confusion_m = confusion_matrix(y_test, y_pred)
3
```

[24]

Python

```
1 print(confusion_m)
2
```

[25]

Python

```
... [[ 457  127]
     [ 102 1814]]
```

## Artificial Intelligence – Phase II

```
1 from sklearn.metrics import accuracy_score
2 accuracy = accuracy_score(y_test, y_pred)
3 print(accuracy)
4
```

[26]

Python

... 0.9084

### Trying out Random Forest

```
1 from sklearn.ensemble import RandomForestClassifier
2 randomclassifier=RandomForestClassifier(n_estimators=200,criterion='entropy')
3 randomclassifier.fit(X_train,y_train)
4
```

[27]

Python

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=200)
```

```
1 y_pred = randomclassifier.predict(X_test)
2
```

[28]

Python

```
1 from sklearn.metrics import confusion_matrix
2 confusion_m = confusion_matrix(y_test, y_pred)
3 print(confusion_m)
4
```

[29]

Python

... [[ 387 197]  
[ 72 1844]]

```
1 from sklearn.metrics import accuracy_score
2 accuracy = accuracy_score(y_test, y_pred)
3 print(accuracy)
4
```

[30]

Python

... 0.8924

### SVM

```
1 from sklearn import svm
2 clf = svm.SVC()
3 clf.fit(X_train, y_train)
4
```

[31]

Python

```
1 y_pred = clf.predict(X_test)
2
```

[ ]

Python

```
1 from sklearn.metrics import confusion_matrix
2 confusion_m = confusion_matrix(y_test, y_pred)
3 print(confusion_m)
4
```

[ ]

Python

... [[ 448 269]  
[ 58 2225]]

## Artificial Intelligence – Phase II

```
1 from sklearn.metrics import accuracy_score
2 accuracy = accuracy_score(y_test, y_pred)
3 print(accuracy)
4
```

0.891

### Trying KNN

```
1 from sklearn.neighbors import KNeighborsClassifier
2 classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
3 classifier.fit(X_train, y_train)
4 y_pred = classifier.predict(X_test)
5
6 from sklearn.metrics import confusion_matrix
7 confusion_m = confusion_matrix(y_test, y_pred)
8 print(confusion_m)
9
10 from sklearn.metrics import accuracy_score
11 accuracy = accuracy_score(y_test, y_pred)
12 print(accuracy)
13
```

[[ 425 292]  
 [ 261 2022]]  
0.8156666666666667

### LR

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.metrics import accuracy_score, classification_report
4
```

```
1 logistic_model = LogisticRegression()
2
```

```
1 logistic_model.fit(X_train, y_train)
2
```

LogisticRegression()  
LogisticRegression()

```
1 # Evaluate the model
2 y_pred = logistic_model.predict(X_test)
3 accuracy = accuracy_score(y_test, y_pred)
4 print("Accuracy:", accuracy)
5 print("Classification Report:")
6 print(classification_report(y_test, y_pred))
7
```

Accuracy: 0.7664  
Classification Report:

	precision	recall	f1-score	support
Negative	0.00	0.00	0.00	584
Positive	0.77	1.00	0.87	1916
accuracy			0.77	2500
macro avg	0.38	0.50	0.43	2500
weighted avg	0.59	0.77	0.67	2500

c:\Users\iamim\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined at \_warn\_prf(average, modifier, msg\_start, len(result))  
c:\Users\iamim\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined at \_warn\_prf(average, modifier, msg\_start, len(result))  
c:\Users\iamim\anaconda3\lib\site-packages\sklearn\metrics\\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined at \_warn\_prf(average, modifier, msg\_start, len(result))

### XGBoost

```
1 import xgboost as xgb
2 from sklearn.metrics import accuracy_score, classification_report
3
```

```
1 # Define the XGBoost model
2 model = xgb.XGBClassifier()
3
```

```
1 from sklearn.metrics import classification_report
2
3 # Train the model
4 model.fit(X_train, y_train_encoded)
5
6 # Predictions
7 y_pred_train = model.predict(X_train)
8 y_pred_test = model.predict(X_test)
9
10 # Decode Labels
11 y_pred_train_decoded = label_encoder.inverse_transform(y_pred_train)
12 y_pred_test_decoded = label_encoder.inverse_transform(y_pred_test)
13
14 # Evaluate the model
15 train_accuracy = (y_pred_train_decoded == y_train).mean()
16 test_accuracy = (y_pred_test_decoded == y_test).mean()
17
```

```
17
18 print("Train Accuracy:", train_accuracy)
19 print("Test Accuracy:", test_accuracy)
20
21 # Classification report
22 print("\nClassification Report on Train Data:")
23 print(classification_report(y_train, y_pred_train_decoded))
24
25 print("\nClassification Report on Test Data:")
26 print(classification_report(y_test, y_pred_test_decoded))
27
```

```
... Train Accuracy: 0.9506666666666667
Test Accuracy: 0.8968
```

```
Classification Report on Train Data:
```

	precision	recall	f1-score	support
Negative	0.96	0.84	0.89	1863
Positive	0.95	0.99	0.97	5637
accuracy			0.95	7500
macro avg	0.95	0.91	0.93	7500
weighted avg	0.95	0.95	0.95	7500

```
Classification Report on Test Data:
      precision    recall  f1-score   support

   Negative      0.83      0.71      0.76        584
   Positive      0.91      0.95      0.93       1916

 accuracy          0.90      2500
 macro avg      0.87      0.83      0.85      2500
weighted avg      0.89      0.90      0.89      2500
```

### Naive Bayes

```
1 from sklearn.naive_bayes import GaussianNB
2 from sklearn.metrics import accuracy_score, classification_report
3
4 # Initialize the Naive Bayes classifier
5 naive_bayes_classifier = GaussianNB()
6
7 # Train the classifier
8 naive_bayes_classifier.fit(X_train, y_train_encoded)
9
10 # Predictions on training and test data
11 y_pred_train = naive_bayes_classifier.predict(X_train)
12 y_pred_test = naive_bayes_classifier.predict(X_test)
13
14 # Calculate accuracy
15 train_accuracy = accuracy_score(y_train_encoded, y_pred_train)
16 test_accuracy = accuracy_score(y_test_encoded, y_pred_test)
17
18 print("Train Accuracy:", train_accuracy)
19 print("Test Accuracy:", test_accuracy)
20
21 # Classification report
22 print("\nClassification Report on Train Data:")
23 print(classification_report(y_train_encoded, y_pred_train, target_names=label_encoder.classes_))
24
25 print("\nClassification Report on Test Data:")
26 print(classification_report(y_test_encoded, y_pred_test, target_names=label_encoder.classes_))
27
```

Python

```
... Train Accuracy: 0.6945333333333333
Test Accuracy: 0.562
```

```
Classification Report on Train Data:
      precision    recall  f1-score   support

   Negative      0.45      1.00      0.62       1863
   Positive      1.00      0.59      0.74       5637

 accuracy          0.69      7500
 macro avg      0.72      0.80      0.68      7500
weighted avg      0.86      0.69      0.71      7500
```

```
Classification Report on Test Data:
      precision    recall  f1-score   support

   Negative      0.31      0.74      0.44        584
   Positive      0.86      0.51      0.64       1916
```

### 4. Model Evaluation and Model Testing

Confusion matrix:

```
from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit LabelEncoder to your target variable and transform it
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

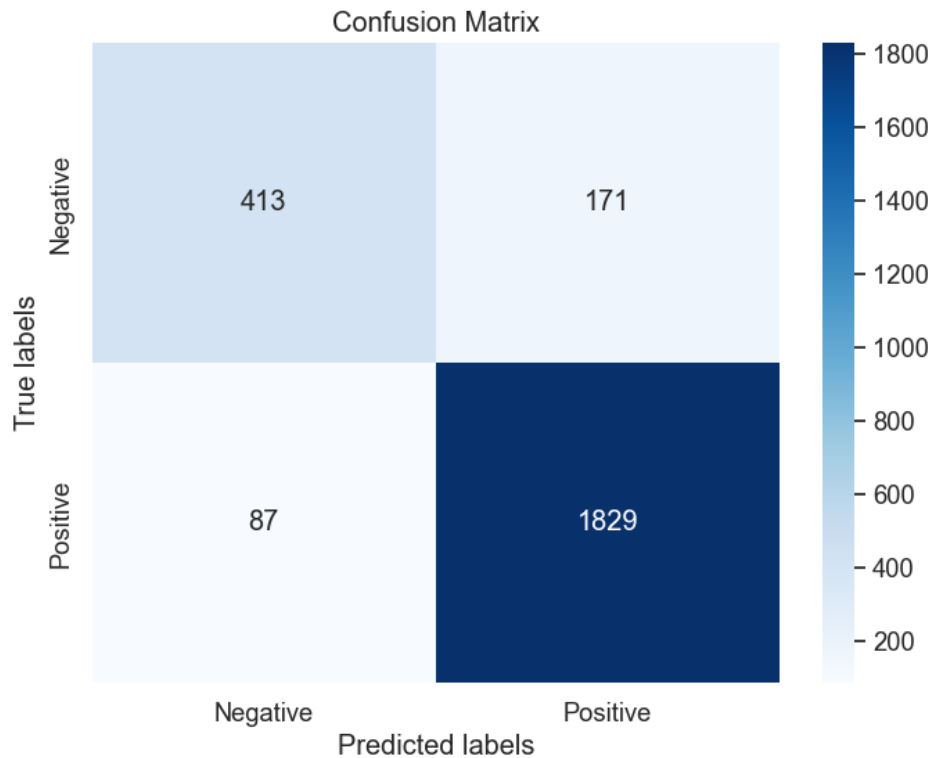
# Now, initialize and train the XGBoost model
model = XGBClassifier()
model.fit(X_train, y_train_encoded)

# Now make predictions on the test set
predicted_labels = model.predict(X_test)

# Generate confusion matrix
cm = confusion_matrix(y_test_encoded, predicted_labels)

# Define classes (assuming binary classification)
classes = label_encoder.classes_

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2) # for label size
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=classes, yticklabels=classes)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



### Roc Curve:

```
from sklearn.metrics import roc_curve, roc_auc_score

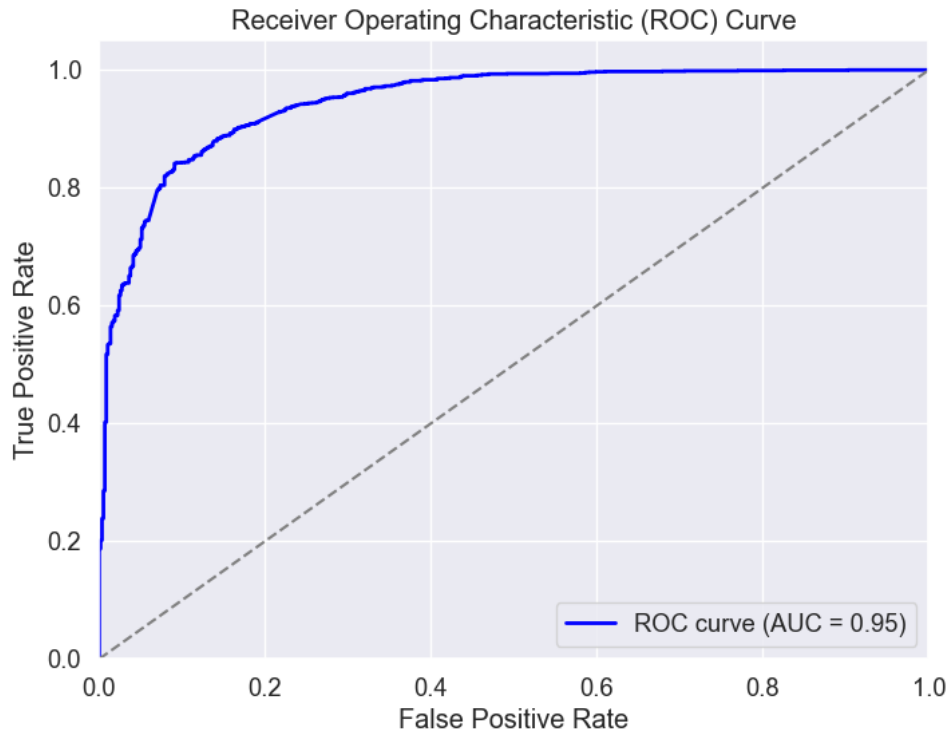
# Get predicted probabilities for positive class
y_prob = model.predict_proba(X_test)[: , 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test_encoded, y_prob)

# Calculate AUC
auc_value = roc_auc_score(y_test_encoded, y_prob)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2,
         label='ROC curve (AUC = %0.2f)' % auc_value)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



An ROC (Receiver Operating Characteristic) curve graphically illustrates the performance of a binary classifier. A 0.95 AUC (Area Under the Curve) indicates strong discriminatory ability, where the model distinguishes between classes with high accuracy. The curve plots true positive rate against false positive rate, with higher AUC values indicating better classifier performance. This metric is crucial in evaluating and comparing the effectiveness of various classification models.

### 5. Conclusion

In this second phase of project to develop sentiment analysis model for restaurant reviews a machine learning model was trained and evaluated based on pre processed data.