Activity based

Project Report on

Artificial Intelligence

Project Phase - III

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: - 23rd April 2024

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2023-2024 Term-II

AI: Phase III

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:

Columns Names	*Descriptions:*
Restaurant	Name of the restaurant
Reviewer	Name of the reviewer
Review	Text of the review
Rating	Rating given by the reviewer
Metadata	Additional metadata related to the review (if any)
Time	Time of the review
Pictures	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.

- Preprocess dataset

Cleaning & Preparing Data

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

[8] Python

```python
y = df["Rating"]
X = df.drop(columns=["Rating"])
X.shape
```

[9] Python

```
(10000, 1)
```

```python
y = y.replace({'Like':3})
```

[10] Python

```python
y.isnull().sum()
```

[11] Python

```
38
```

```python
y = pd.to_numeric(y)
```

[12] Python

```python
y = y.fillna(y.median())
```

[13] Python

```python
for i in range(0,len(y)):
    y.iloc[i] = round(y.iloc[i],0)
```

[14] Python

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

[15] Python

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

[16] Python

```python
y.unique()
```

[17] Python

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

```python
y.shape
```

[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:

Models Name	Accuracy	Recall	Precision	F-Beta-Score
MultinomialNB	0.9084	0.856	0.7684	0.8865
Random Forest	0.8924	0.855	0.8321	0.8064
SVM	0.891	0.80	0.785	0.777
KNN	0.8156	0.70	0.689	0.80
LR	0.77	0.50	0.38	0.43
XGBoost	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

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

```

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

```

```
from sklearn.metrics import classification_report

# Train the model
model.fit(X_train, y_train_encoded)

# Predictions
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Decode labels
y_pred_train_decoded = label_encoder.inverse_transform(y_pred_train)
y_pred_test_decoded = label_encoder.inverse_transform(y_pred_test)

# Evaluate the model
train_accuracy = (y_pred_train_decoded == y_train).mean()
test_accuracy = (y_pred_test_decoded == y_test).mean()

print("Train Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)

# Classification report
print("\nClassification Report on Train Data:")
print(classification_report(y_train, y_pred_train_decoded))

print("\nClassification Report on Test Data:")
print(classification_report(y_test, y_pred_test_decoded))

```

```
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
```

- 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 Consusion 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 the 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

```python
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(X)):
    review = re.sub('[^a-zA-Z]',' ', str(X['Review'][i]))
    review = review.lower() #Lowering the words is very imporatant in avoiding classifying same words as different words
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')] #Eleminating words that do not put much value in se
    review = ' '.join(review) #Reconstructing sentences
    corpus.append(review)

```

[19] Python

For implementing LSTM goto direct LSTM Section Else Continue

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

```

[20] Python

Train-Test Split

```python
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)

```

[21] Python

Deciding Best Model:

Trying Out MultinomialNB

```python
from sklearn.naive_bayes import MultinomialNB
restaurant_review_model = MultinomialNB().fit(X_train, y_train)

```

[22] Python

```python
y_pred = restaurant_review_model.predict(X_test)

```

[23] Python

```python
from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(y_test, y_pred)

```

[24] Python

```python
print(confusion_m)

```

[25] Python

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

[26]
```python
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```
Python

```
0.9084
```

Trying out Random Forest

[27]
```python
from sklearn.ensemble import RandomForestClassifier
randomclassifier=RandomForestClassifier(n_estimators=200,criterion='entropy')
randomclassifier.fit(X_train,y_train)
```
Python

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

[28]
```python
y_pred = randomclassifier.predict(X_test)
```
Python

[29]
```python
from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(y_test, y_pred)
print(confusion_m)
```
Python

```
[[ 387  197]
 [  72 1844]]
```

[30]
```python
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```
Python

```
0.8924
```

SVM

[31]
```python
from sklearn import svm
clf = svm.SVC()
clf.fit(X_train, y_train)
```
Python

[ ]
```python
y_pred = clf.predict(X_test)
```
Python

[ ]
```python
from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(y_test, y_pred)
print(confusion_m)
```
Python

```
[[ 448  269]
 [  58 2225]]
```

```python
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

```

```
0.891
```

Trying KNN

```python
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
confusion_m = confusion_matrix(y_test, y_pred)
print(confusion_m)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

```

```
[[ 425  292]
 [ 261 2022]]
0.8156666666666667
```

LR

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

```

```python
logistic_model = LogisticRegression()

```

```python
logistic_model.fit(X_train, y_train)

```

```
▾ LogisticRegression
LogisticRegression()
```

```python
# Evaluate the model
y_pred = logistic_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))

```

```
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 ai
  _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 ai
  _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 ai
  _warn_prf(average, modifier, msg_start, len(result))
```

XGBoost

```python
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report

```

Python

```python
# Define the XGBoost model
model = xgb.XGBClassifier()

```

Python

```python
from sklearn.metrics import classification_report

# Train the model
model.fit(X_train, y_train_encoded)

# Predictions
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Decode labels
y_pred_train_decoded = label_encoder.inverse_transform(y_pred_train)
y_pred_test_decoded = label_encoder.inverse_transform(y_pred_test)

# Evaluate the model
train_accuracy = (y_pred_train_decoded == y_train).mean()
test_accuracy = (y_pred_test_decoded == y_test).mean()

print("Train Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)

# Classification report
print("\nClassification Report on Train Data:")
print(classification_report(y_train, y_pred_train_decoded))

print("\nClassification Report on Test Data:")
print(classification_report(y_test, y_pred_test_decoded))

```

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      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

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

# Initialize the Naive Bayes classifier
naive_bayes_classifier = GaussianNB()

# Train the classifier
naive_bayes_classifier.fit(X_train, y_train_encoded)

# Predictions on training and test data
y_pred_train = naive_bayes_classifier.predict(X_train)
y_pred_test = naive_bayes_classifier.predict(X_test)

# Calculate accuracy
train_accuracy = accuracy_score(y_train_encoded, y_pred_train)
test_accuracy = accuracy_score(y_test_encoded, y_pred_test)

print("Train Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)

# Classification report
print("\nClassification Report on Train Data:")
print(classification_report(y_train_encoded, y_pred_train, target_names=label_encoder.classes_))

print("\nClassification Report on Test Data:")
print(classification_report(y_test_encoded, y_pred_test, target_names=label_encoder.classes_))

```

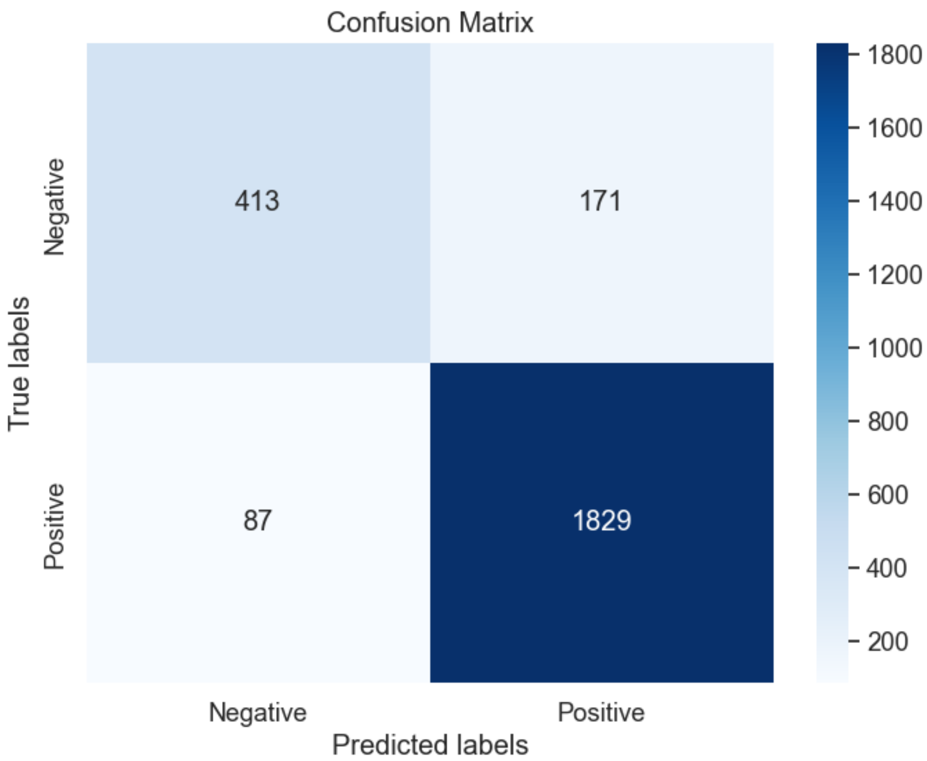
[ ] Python

```
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:

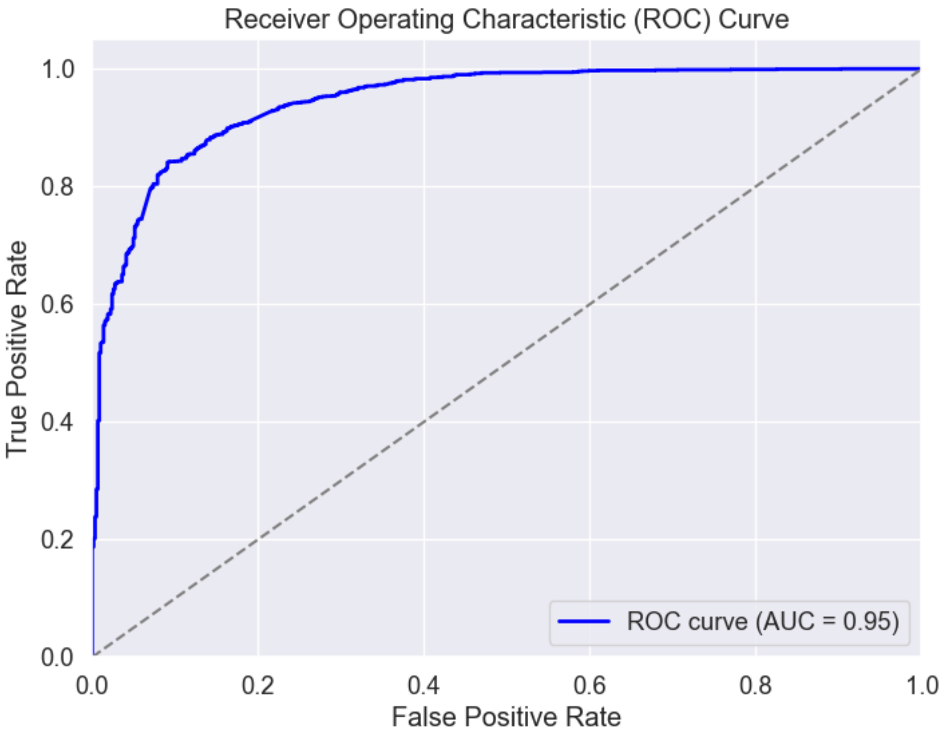
```
from sklearn.preprocessing import LabelEncoder

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