GitHub Link:

https://github.com/dheerajreddykomandla/NLP_TextEmotionDetection

CSCE 5290
Natural Language Processing
Text Emotion Detection
Final Project

1. Project Title and Members:

- **Project Title: Text Emotion Detection**
- **Team Number:** Team 10

Team Members:

Dheeraj Reddy Komandla (11526265)
Prem sai Vuppula (11528103)
Rahul Reddy Geereddy (11511846)
Sravan Boinapalli (11553755)

2. Goals and Objectives:

- **Motivation:** We opted to work on the Text categorization problem as a team in this project. In which we have decided to work on the Text Classification challenge. Sentiment analysis has recently been used in a wide range of applications. Instead of working on the binary classification task, we as a team decided to challenge ourselves. The key reason for using Text Emotion Detection is that the class labels are connected to the sentiment analysis problem and more resemble the intent classification problem used in conversation bots. According to our investigation, and for the reasons stated above, we decided on the Text Emotion Detection challenge.
- **Significance:** As previously stated, the primary value of this model is that it has a broad range of real-world applications. Consider a chatbot conversation in which emotions play a key part in communication between people from different cultures or areas. Assume we have divided emotions into four categories. They are: joy, anger, sadness, fear.

  We can see the difference between single word and multiple word assertions in the example below.

  Single Words

  - Great
  - Nope
  - Recommend

  Multiple Words Statements

  - This is great!
  - Nope, won't buy again!
  - Definitely recommend this product!

The emotion represented in both statements is the same, but the amount of words varies. As a result, it is critical for a chatbot model to accurately classify the emotion based on which future communication must take place.

Let’s try to understand the significance of the model with one more example. If you consider the below statements.

- This is not a friendly product.
- I hate this product; not useful at all.
- I want to return this product, maybe order a replacement.

If you observe the above 3 statements, all are having the sentiment negative. But the meaning, emotion, and the customer requirement for the 3 statements are different.

Objectives:

Our team has various goals that must be met in order to properly construct an emotion detection model. The following objectives depict the project's workflow.

1. Data Selection: As we did some research, we have shortlisted some of the datasets available in the Kaggle. We would like to spend some more time to pick the right dataset with good number of emotion samples.

2. Text Cleaning: In any of the NLP tasks, it is important to clean the text dataset. Because the raw dataset may contain lot of missing values, duplicate values etc. We need to handle all the cases. If the text is not cleaned, then the model may give bad results, or it fails.

3. Text Preprocessing: It is also an important phase in all NLP related tasks. In this phase, we generally remove the white spaces, remove unnecessary special characters, stop words removal etc.

4. Text Vectorization: This phase is also known as text featurization. In this phase we generally convert the text into vectors. There are some popular Vectorization techniques available such as Bag of Words, Tf-IDF, Word2Vec etc. By using these techniques, we will be able to convert the text into n dimensional vector.

5. Modeling: Once we converted the text into vector, then we are good to go for modeling. Here we want to try this various machine learning algorithms. Of all the models that are trained, which ever gives the best accuracy, we will choose that model as best model.

6. Visualization & Evaluation of Results: In this phase, we basically compare the results of each model and plot the results to understand the performance of the model. Since it is a classification model, we would like to compare the

performance with accuracy. If the dataset is imbalanced, then we will also
look into f1-score, AUC score.

Features:

As discussed earlier we would like to spend some more time on picking the dataset, but in most of the text classification datasets, it contains 2 features.

1. Text: This feature mainly contains the raw text. This text represents the either review or statement for which we need to predict the emotion of the
statement.

2. Label: This feature represents the class label of the given sentence. In most of the datasets we basically have 4 classes. They are:

1) Fear
2) Joy
3) Anger
4) Sadness

There are some import machine learning libraries are required to solve this problem. They are listed below:

1) Pandas
2) SKlearn
3) Matplotlib
4) XGboost

3. Related Work (Background):

According to our study, the most often utilized strategy for text categorization problems includes the steps listed below.

1) Text Cleaning

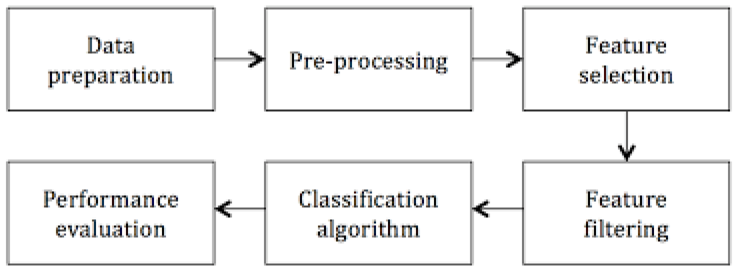
2) Text Preprocessing

3) Text Vectorization

4) Train Test Split

5) Modeling

6) Visualization of Results



Some of the research that we have done are listed below.

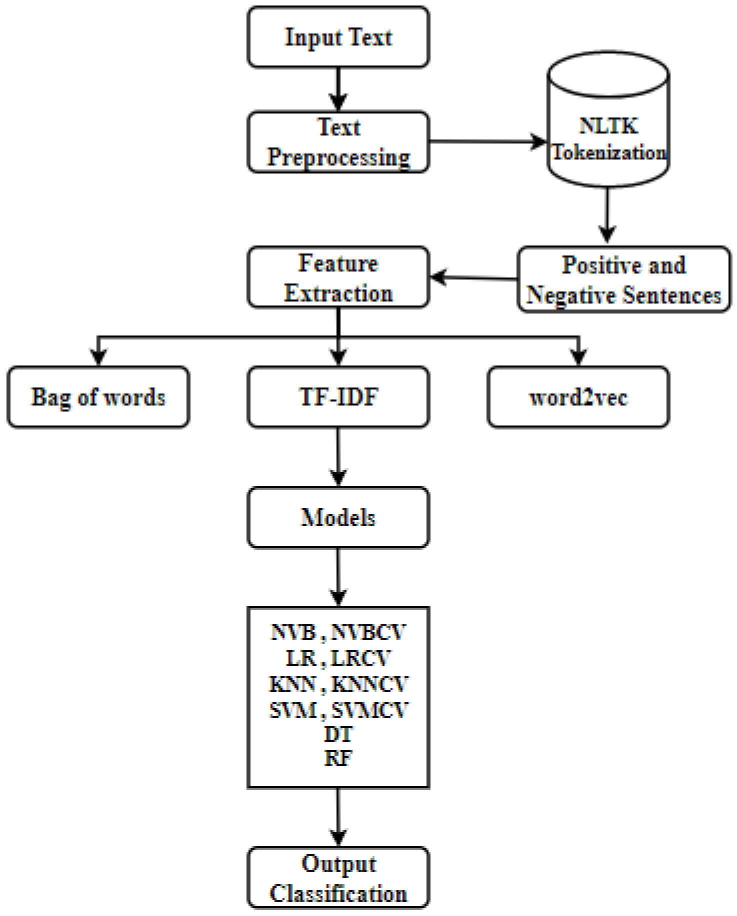
https://stackabuse.com/text-classification-with-python-and-scikit-learn/

https://www.datacamp.com/tutorial/naive-bayes-scikit-learn

https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

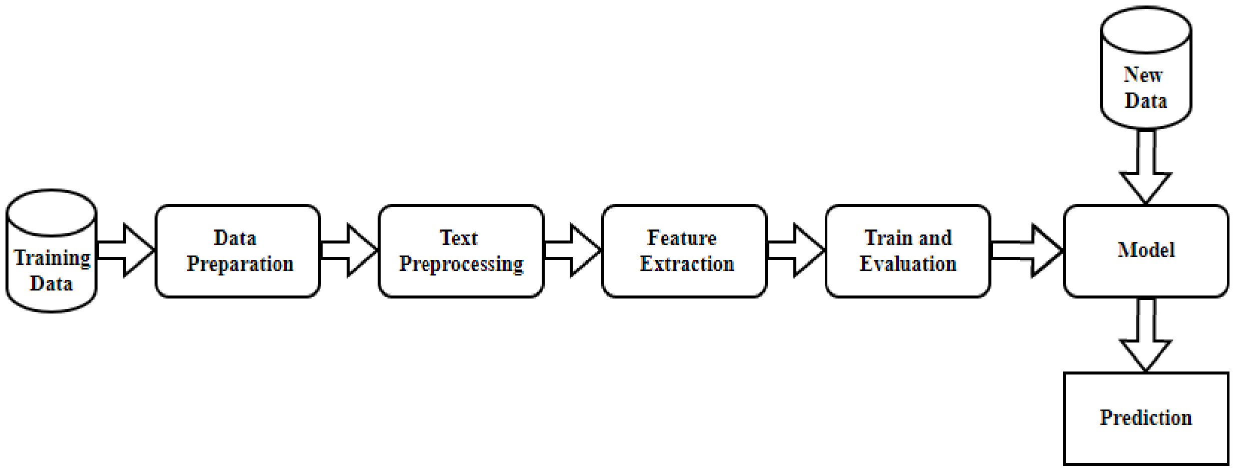
4.Model:

Architecture Diagram :



The architecture diagram is shown above. It contains various blocks. During the pre processing step the accuracy is improved by removing the components from the data which are not necessary. Tokenization is the process of breaking up the input text into smaller units called tokens. Tokenization technique allows the classifiers to quickly grasp the text. Stop words are often used phrases that don't really add anything to the language. Stop words are removed from text. During positive and negative data we have removed the records which are having fear and angry classes. In the final filtered dataset we have joy class labels as positive and sadness class labels as negative. In next step data is processed for feature extraction. We have used methods like bag of words, TF-IDF, word2vec for feature extraction. After that we prepared the train, validation and test data. We use different classifiers on train, validation and test data to calculate the accuracy of the model. The analysis is based on several performance criteria, and a comparison analysis has been performed to determine its efficiency.

Model Workflow:



The initial stage is to collect text emotion recognition data. The dataset used in this experiment was obtained from the kaggle repository. The following step is to prepare the data. This stage entails removing stop words and other unneeded information. Text pre-processing is the next step. We have joy, sadness, fear, and anger classes in the present dataset. We solely evaluated joy and melancholy in order to do sentimental analysis. We know that all of the joy texts are positive and all of the sad texts are negative. As a result, we have erased the records that contain fear and anger classes. Now we have to perform Train Test Split operation. It helps us to train the model on training dataset and test the model on the test dataset. It also helps to understand if the model is over-fitting. In our dataset we have a separate dataset for testing. Hence we are using the split-ted dataset for validation and the separate test file for the testing the final accuracy. The test dataset is used to test the model and provide information about how good is trained classifier. The confusion matrix indicates that this model can categorize joy and melancholy, and the output shows the accuracy of the train, validation, and test data of different models.

5. Data Set:

Dataset Description:

The dataset we are currently using in this project was obtained from the kaggle repository. Kaggle is a repository of a huge number of datasets that are freely available for use in practicing data science challenges. The link to the dataset's kaggle repository is provided below.

https://www.kaggle.com/datasets/anjaneyatripathi/emotion-classification-nlp

We have two columns in the dataset to work with as we solve the Multi-Class text classification problem.

1. Text Column: This column contains the raw text. Our model should be able to take this text as input and predict the mood of the text.

2. Label: This column contains the class label for the relevant input text. The Label column has four different types of feelings. They are Fear, Anger, Joy, and Sadness.

Detail Design of Features: When it comes to machine learning or deep learning, the models cannot directly interpret text data. As a result, we must convert the text into numerical features. There are numerous text enhancement techniques available. Popular featurization techniques include:

1. Bag of words

2. TF-IDF

3. Word2Vec

In general, every featurization algorithm will turn the text into an N-dimensional array. We are now using the TF-IDF featurization technique in this project.

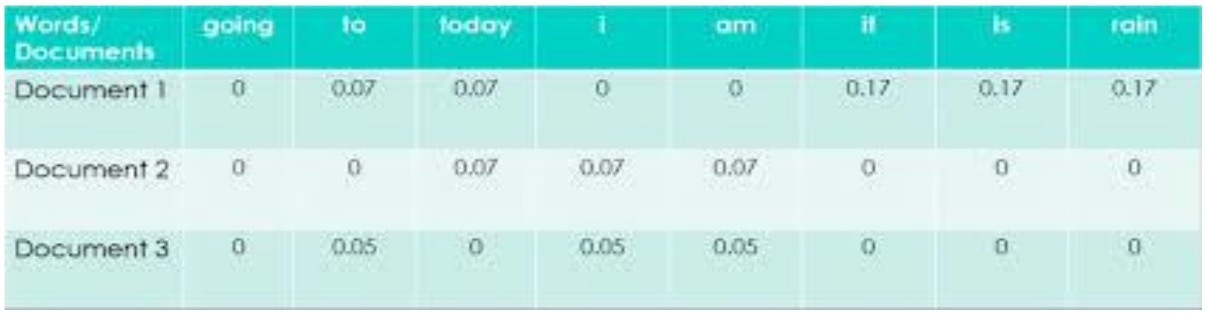
TF-IDF Featurization: TF-IDF stands for term frequency and inverse document frequency. The TF-IDF technique is nothing more than assigning a weight to each unique word in the dataset in relation to the given input text. The formula for calculating the weight of a word in relation to a specific text row is shown in the picture below.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

The example below shows how text can be transformed into an N-dimensional vector.



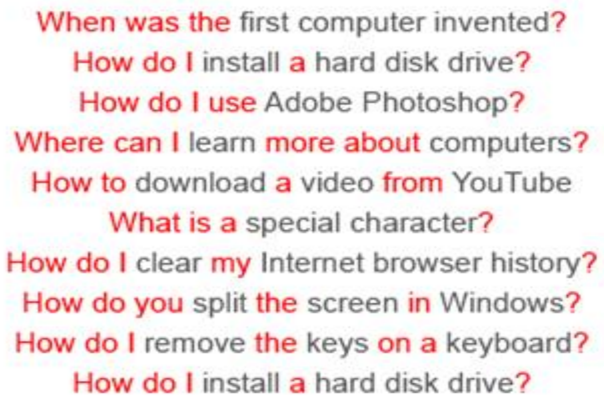
Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

6. Analysis of Data:

Data Preprocessing :

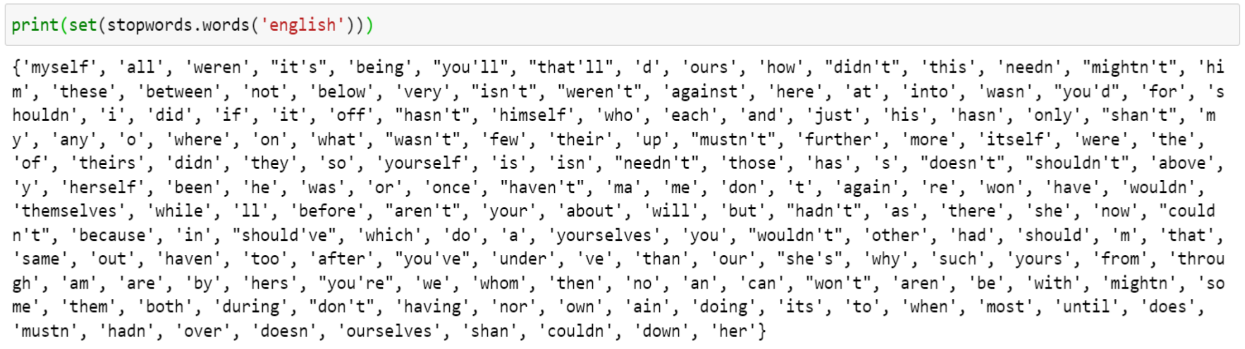
While doing the research we came through an important text preprocessing practice called stopword removal. Stop-words are the set words which don't add much value in a sentence. Examples of stop words in English are "a", "the", "is", "are" and etc. Stop words are commonly used in Text Mining and Natural Language Processing (NLP) to eliminate words that are so commonly used that they carry very little useful information.

In the below image we can find the stopwords which are highlighted words in red color.



When was the first computer invented?
How do I install a hard disk drive?
How do I use Adobe Photoshop?
Where can I learn more about computers?
How to download a video from YouTube
What is a special character?
How do I clear my Internet browser history?
How do you split the screen in Windows?
How do I remove the keys on a keyboard?
How do I install a hard disk drive?

In the below screenshot, we have printed all the stop words that are removed in our each text.



```
print(set(stopwords.words('english')))
```

```
{'myself', 'all', 'weren', "it's", 'being', "you'll", "that'll", 'd', 'ours', 'how', "didn't", 'this', 'needn', "mightn't", 'hi
m', 'these', 'between', 'not', 'below', 'very', "isn't", "weren't", 'against', 'here', 'at', 'into', 'wasn', "you'd", 'for', 's
houldn', 'i', 'did', 'if', 'it', 'off', "hasn't", 'himself', 'who', 'each', 'and', 'just', 'his', 'hasn', 'only', "shan't", 'm
y', 'any', 'o', 'where', 'on', 'what', "wasn't", 'few', 'their', 'up', "mustn't", 'further', 'more', 'itself', 'were', 'the',
'of', 'theirs', 'didn', 'they', 'so', 'yourself', 'is', 'isn', "needn't", 'those', 'has', 's', "doesn't", "shouldn't", 'above',
'y', 'herself', 'been', 'he', 'was', 'or', 'once', "haven't", 'ma', 'me', 'don', 't', 'again', 're', 'won', 'have', 'wouldn',
'themselves', 'while', 'll', 'before', "aren't", 'your', 'about', 'will', 'but', "hadn't", 'as', 'there', 'she', 'now', "could
n't", 'because', 'in', "should've", 'which', 'do', 'a', 'yourselves', 'you', "wouldn't", 'other', 'had', 'should', 'm', 'that',
'same', 'out', 'haven', 'too', 'after', "you've", 'under', 've', 'than', 'our', "she's", 'why', 'such', 'yours', 'from', 'throu
gh', 'am', 'are', 'by', 'hers', "you're", 'we', 'whom', 'then', 'no', 'an', 'can', "won't", 'aren', 'be', 'with', 'mightn', 'so
me', 'them', 'both', 'during', "don't", 'having', 'nor', 'own', 'ain', 'doing', 'its', 'to', 'when', 'most', 'until', 'does',
'mustn', 'hadn', 'over', 'doesn', 'ourselves', 'shan', 'couldn', 'down', 'her'}
```

The below function is used to remove the stopwords.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

def remove_stopwords(sentence):
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(sentence)
    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
    return " ".join(filtered_sentence)
```

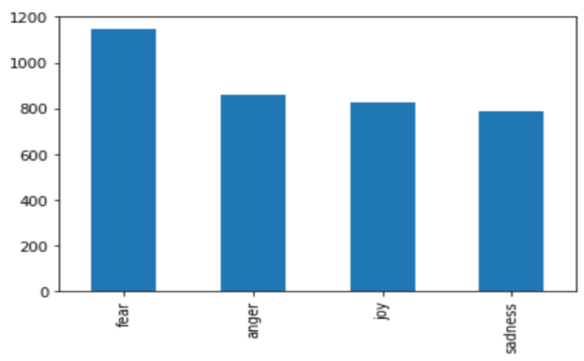
As we are analyzing our dataset, we observed the dataset is actually imbalanced. More often the imbalanced datasets are prone to biased predictions. Hence we have balanced our datasets. In our first we have trained naive model on imbalance dataset but the results were not much promising. Now we have trained the same model using the balanced dataset. The result was still not the best but all the train, validation and test accuracy has improved.

<u>Graph model:</u>

The below screenshot represents the label column before balancing the dataset.

```
train_df.label.value_counts().plot(kind='bar')
```

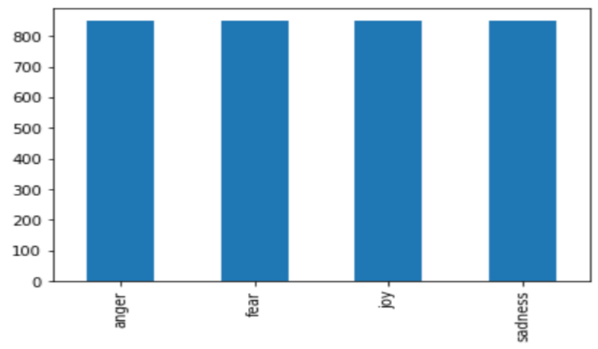
```
<matplotlib.axes._subplots.AxesSubplot at 0x23f4ef
```



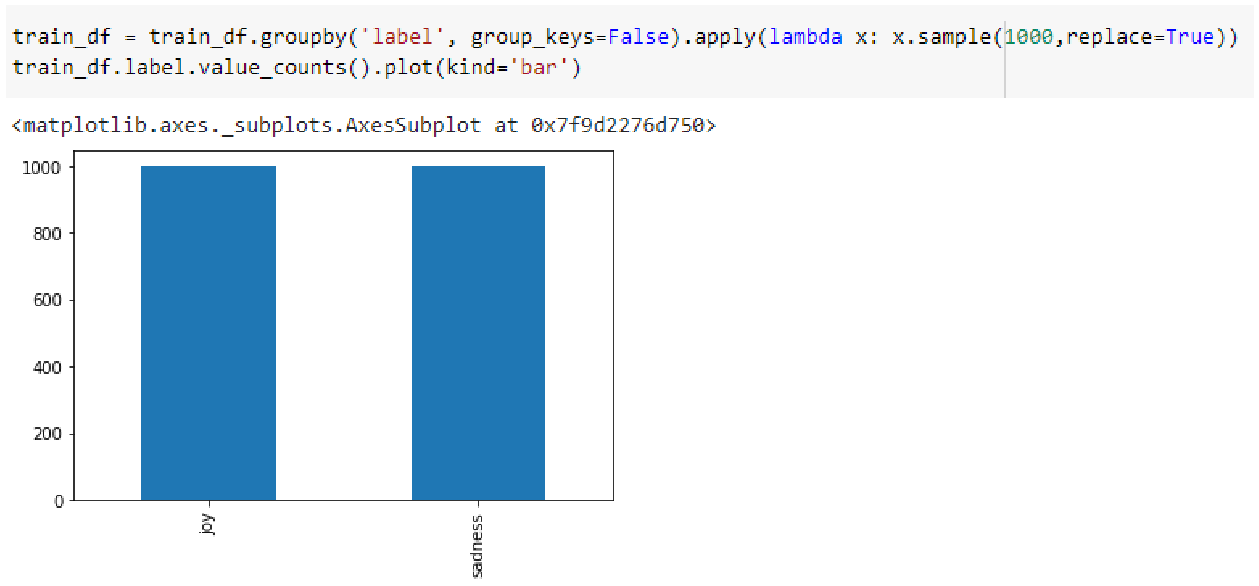
The below screenshot represents the label column after balancing the dataset.

```
train_df = train_df.groupby('label', group_keys=False).apply(lambda x: x.sample(850,replace=True))
train_df.label.value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23f516b7b88>
```



The below screenshot represents the label column to perform sentiment analysis on the dataset.

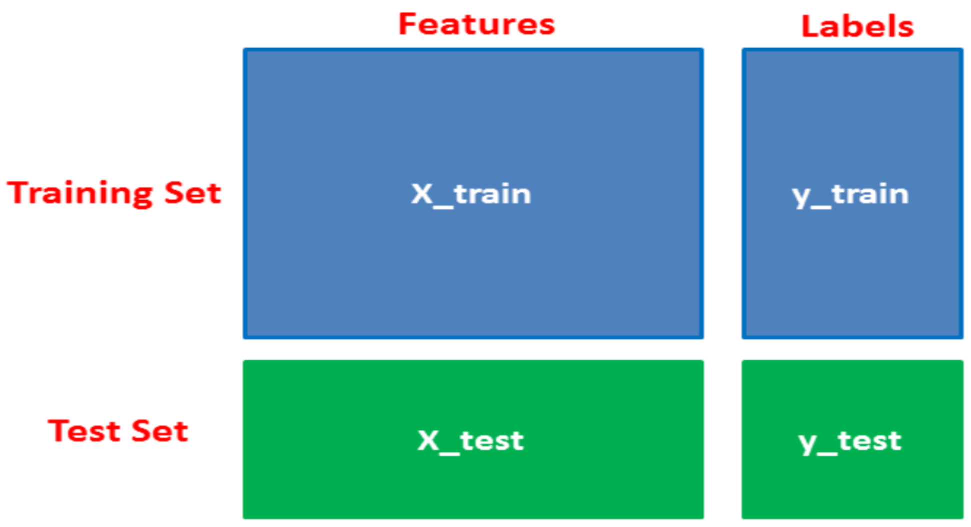


```
train_df = train_df.groupby('label', group_keys=False).apply(lambda x: x.sample(1000,replace=True))
train_df.label.value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9d2276d750>
```

We have sampled each class 850 rows. Now our dataset is balanced and any model which was trained on balanced dataset, will not be biased.

Till now in this section we have the featurized the given input into N-dimensional vector representation. Now we have to perform Train Test Split operation. It is an important practice in the area of machine learning and deep learning. It helps us to train the model on training dataset and test the model on the test dataset. It also helps to understand if the model is over-fitting. The below figure represents the breakup of the train test split. In our dataset we have a separate dataset for testing. Hence we are using the split-ted dataset for validation and the separate test file for the testing the final accuracy.



7.<u>Implementation:</u>

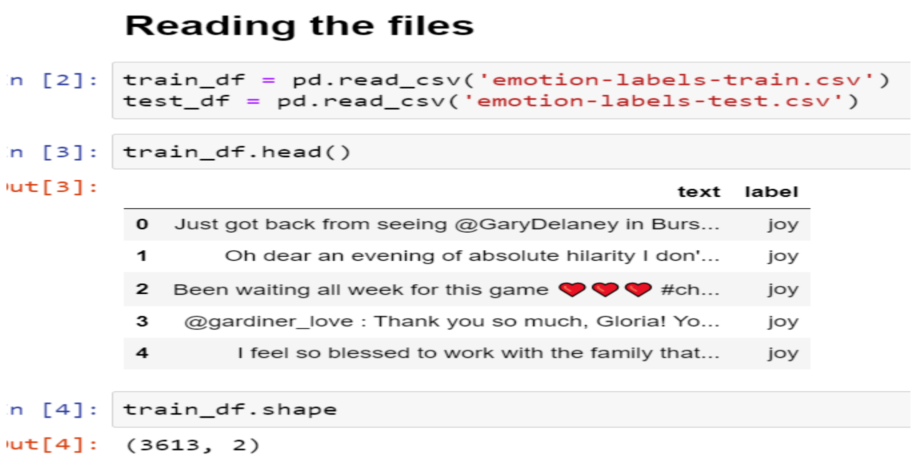
As suggested, we have converted and used the current dataset as sentiment analysis. In the current dataset, we have joy, sadness, fear and angry classes. In general we know that all the joy texts are positive and all the sadness texts are negative. Hence we have removed the records which are having fear and angry classes.

In the final filtered dataset we have joy class labels as positive and sadness class labels as negative. The label id 1 represents the positive class and label id 2 represents the negative class.

we have integrated the concepts of (sentiment analysis + classification) in this submission.

The project flow has already been explained. Each phase's implementation is shown in this section. We have included screenshots for each phase of this project.

1. Data Preparation:



Reading the files

```
n [2]:  train_df = pd.read_csv('emotion-labels-train.csv')
        test_df = pd.read_csv('emotion-labels-test.csv')

n [3]:  train_df.head()
```

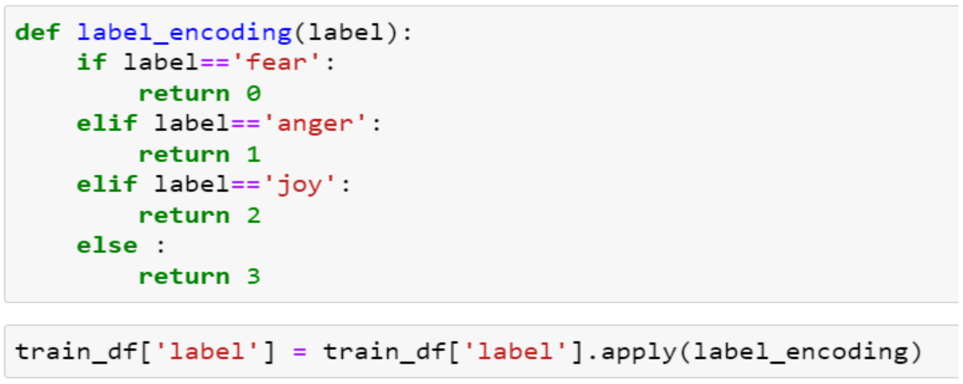
ut[3]:

	text	label
0	Just got back from seeing @GaryDelaney in Burs...	joy
1	Oh dear an evening of absolute hilarity I don'...	joy
2	Been waiting all week for this game ❤❤❤ #ch...	joy
3	@gardiner_love : Thank you so much, Gloria! Yo...	joy
4	I feel so blessed to work with the family that...	joy

```
n [4]:  train_df.shape

ut[4]:  (3613, 2)
```

2. Text Processing:

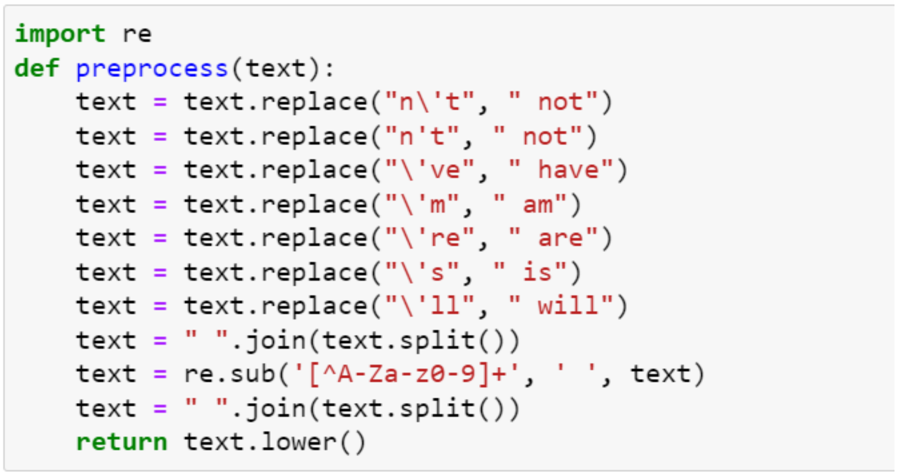
The function listed below converts the class label to the label-id.



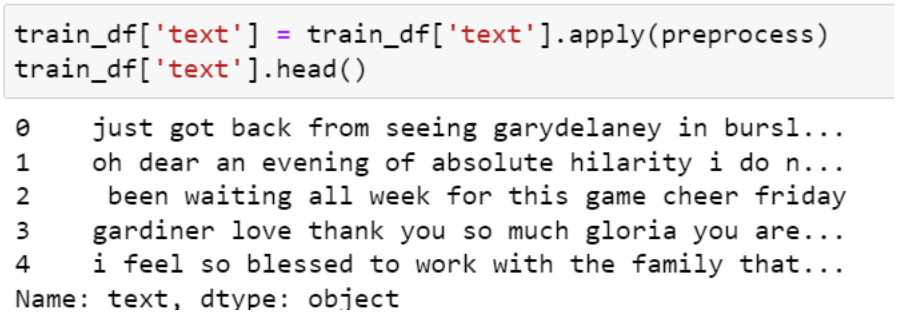
```
def label_encoding(label):
    if label=='fear':
        return 0
    elif label=='anger':
        return 1
    elif label=='joy':
        return 2
    else :
        return 3
```

```
train_df['label'] = train_df['label'].apply(label_encoding)
```

The function below accepts text as input and performs all preparation procedures required to clean the text data.



```python
import re
def preprocess(text):
    text = text.replace("n\'t", " not")
    text = text.replace("n't", " not")
    text = text.replace("\'ve", " have")
    text = text.replace("\'m", " am")
    text = text.replace("\'re", " are")
    text = text.replace("\'s", " is")
    text = text.replace("\'ll", " will")
    text = " ".join(text.split())
    text = re.sub('[^A-Za-z0-9]+', ' ', text)
    text = " ".join(text.split())
    return text.lower()
```

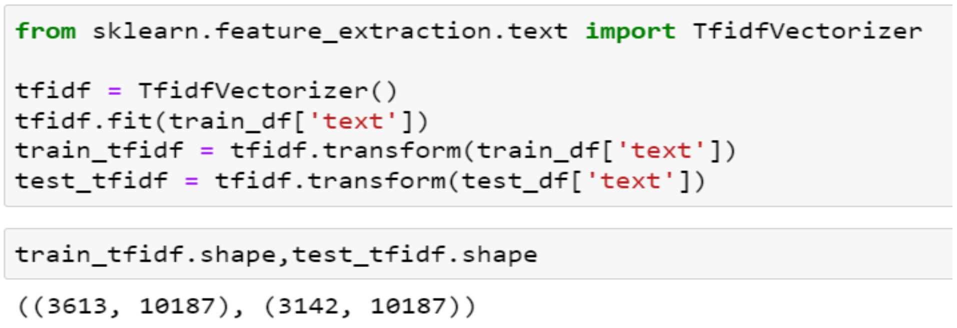


```python
train_df['text'] = train_df['text'].apply(preprocess)
train_df['text'].head()
```

```
0    just got back from seeing garydelaney in bursl...
1    oh dear an evening of absolute hilarity i do n...
2     been waiting all week for this game cheer friday
3    gardiner love thank you so much gloria you are...
4    i feel so blessed to work with the family that...
Name: text, dtype: object
```

3. Feature Extraction:

The code part below discusses feature extraction.



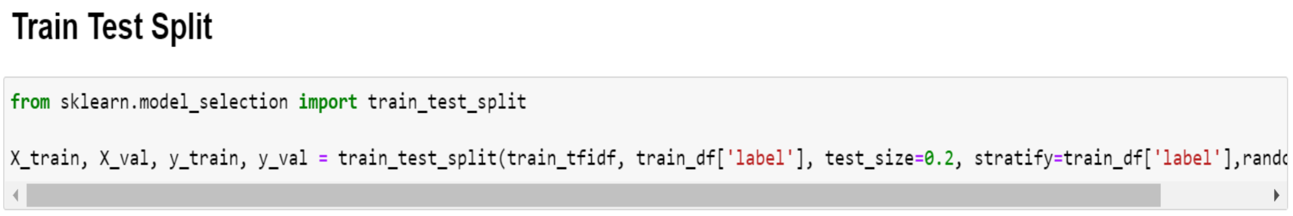
```python
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
tfidf.fit(train_df['text'])
train_tfidf = tfidf.transform(train_df['text'])
test_tfidf = tfidf.transform(test_df['text'])
```

```python
train_tfidf.shape,test_tfidf.shape
```

```
((3613, 10187), (3142, 10187))
```

4. Train Test Split:

The train test split is implemented using the code below.

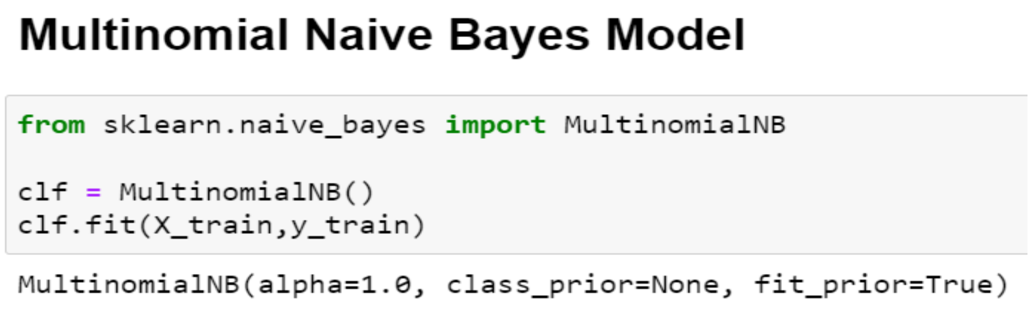


Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(train_tfidf, train_df['label'], test_size=0.2, stratify=train_df['label'],rando
```

5. Modeling: In the first phase of this project we have tried 2 models. 1). Multinomial Naive Bayes 2). Multinomial Naive Bayes with Cross Validation. Now in this increment we have used several models. The implementation for all the models are listed below.

1) **<u>Multinomial Naive Bayes:</u>**

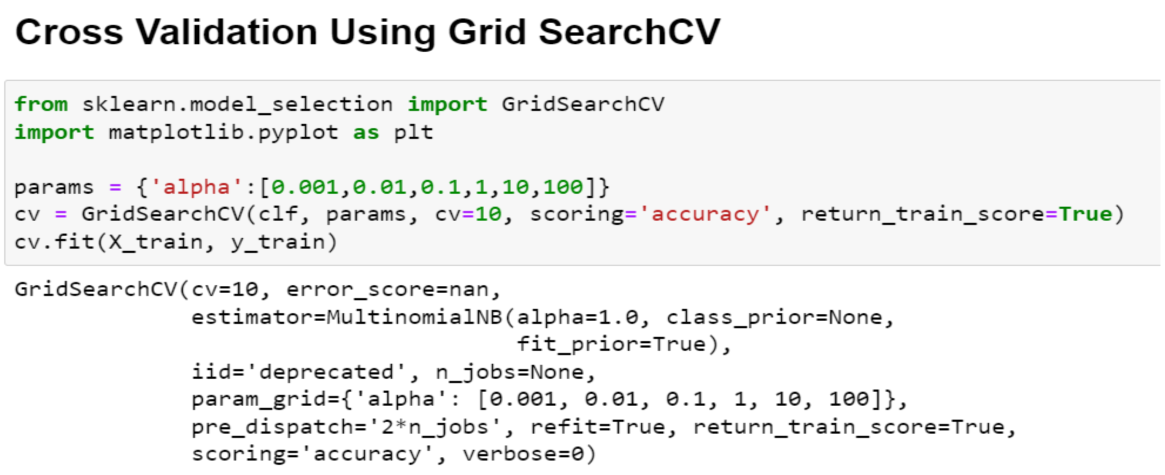


Multinomial Naive Bayes Model

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()
clf.fit(X_train,y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

2) **<u>Multinomial Naive Bayes with Cross Validation:</u>**

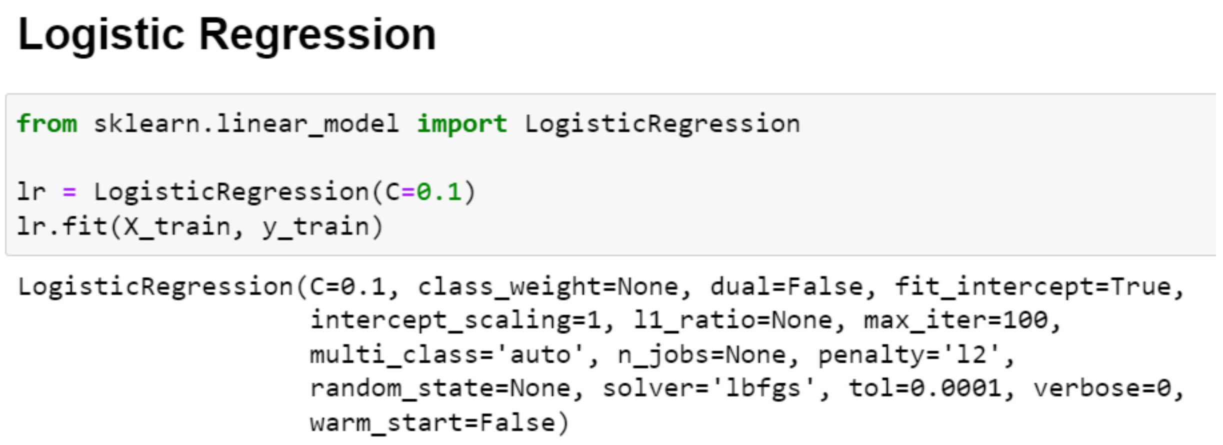


Cross Validation Using Grid SearchCV

```
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

params = {'alpha':[0.001,0.01,0.1,1,10,100]}
cv = GridSearchCV(clf, params, cv=10, scoring='accuracy', return_train_score=True)
cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=10, error_score=nan,
             estimator=MultinomialNB(alpha=1.0, class_prior=None,
                                     fit_prior=True),
             iid='deprecated', n_jobs=None,
             param_grid={'alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=0)
```

3) <u>Logistic Regression:</u>

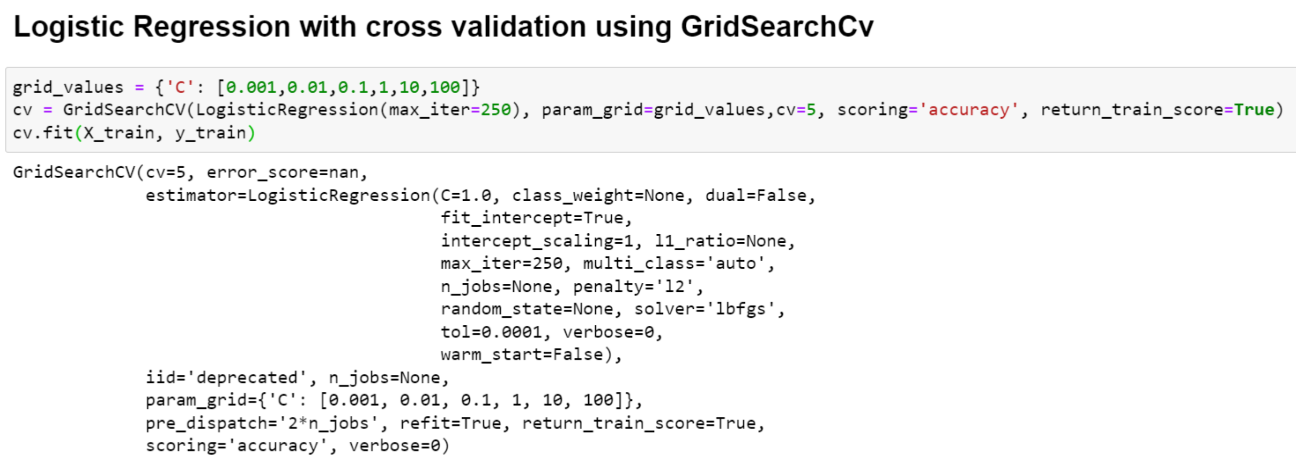


Logistic Regression

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(C=0.1)
lr.fit(X_train, y_train)
```

```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

4) <u>Logistic Regression with Cross Validation:</u>

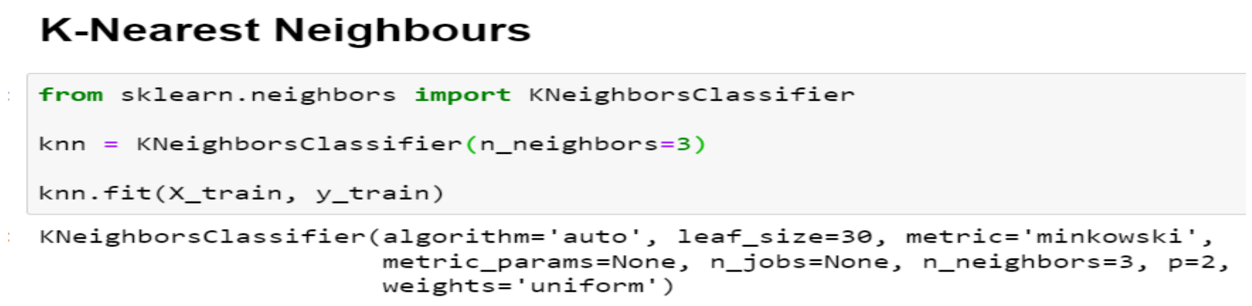


Logistic Regression with cross validation using GridSearchCv

```
grid_values = {'C': [0.001,0.01,0.1,1,10,100]}
cv = GridSearchCV(LogisticRegression(max_iter=250), param_grid=grid_values,cv=5, scoring='accuracy', return_train_score=True)
cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=250, multi_class='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=0)
```

5) <u>K-Nearest Neighbours:</u>

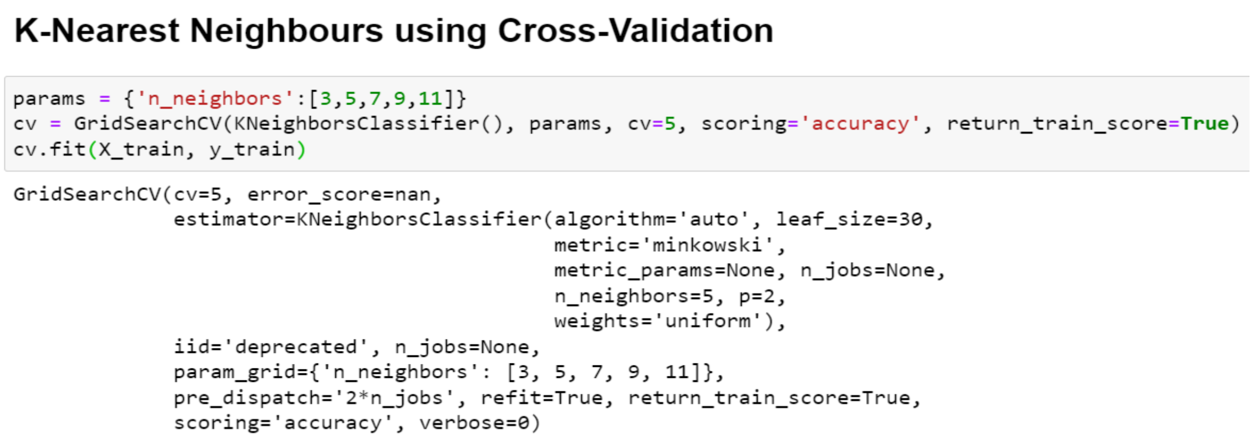


K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

6) <u>K-Nearest Neighbours with Cross validation:</u>

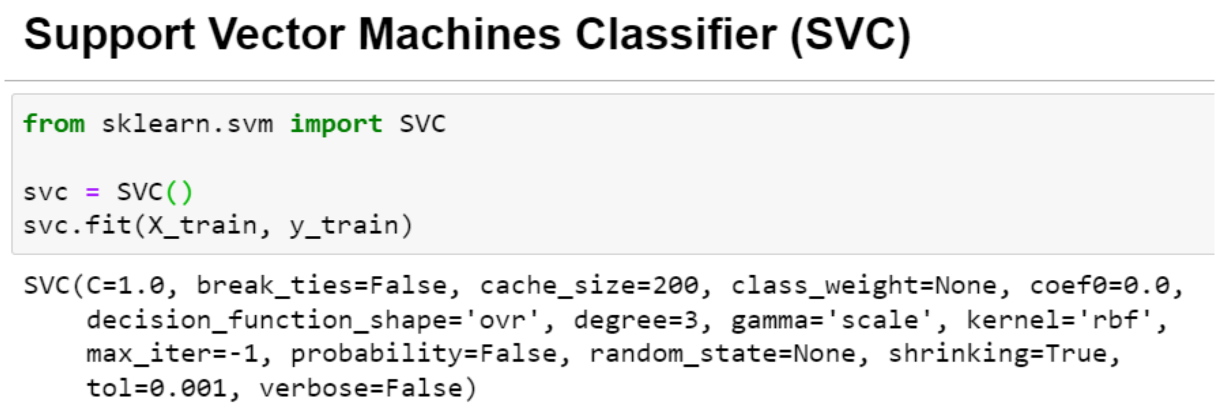


K-Nearest Neighbours using Cross-Validation

```
params = {'n_neighbors':[3,5,7,9,11]}
cv = GridSearchCV(KNeighborsClassifier(), params, cv=5, scoring='accuracy', return_train_score=True)
cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=None,
             param_grid={'n_neighbors': [3, 5, 7, 9, 11]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=0)
```

7) <u>Support Vector Machine Classifier:</u>

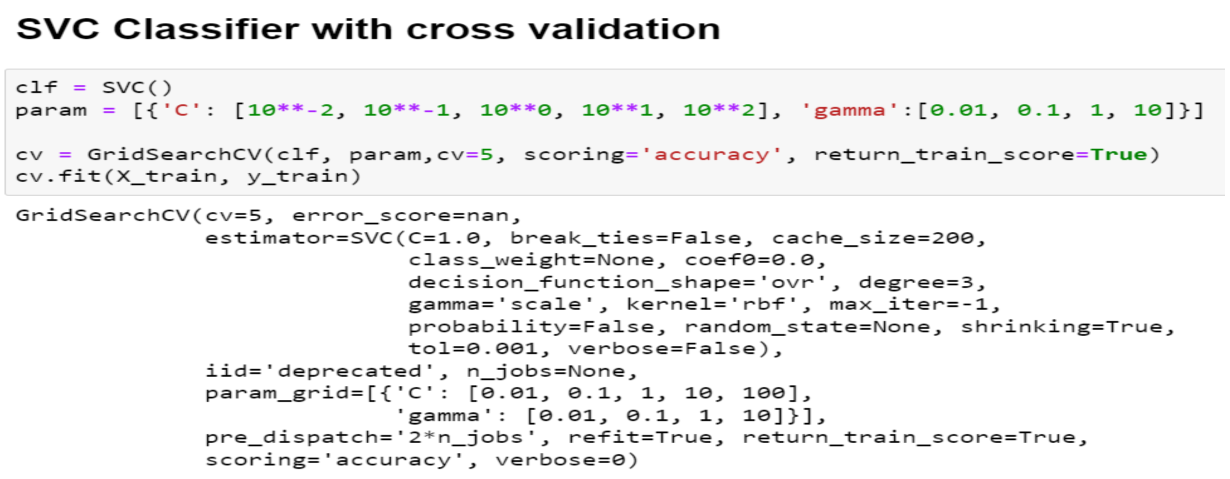


Support Vector Machines Classifier (SVC)

```
from sklearn.svm import SVC

svc = SVC()
svc.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

8) <u>Support Vector Machine Classifier with Cross Validation:</u>

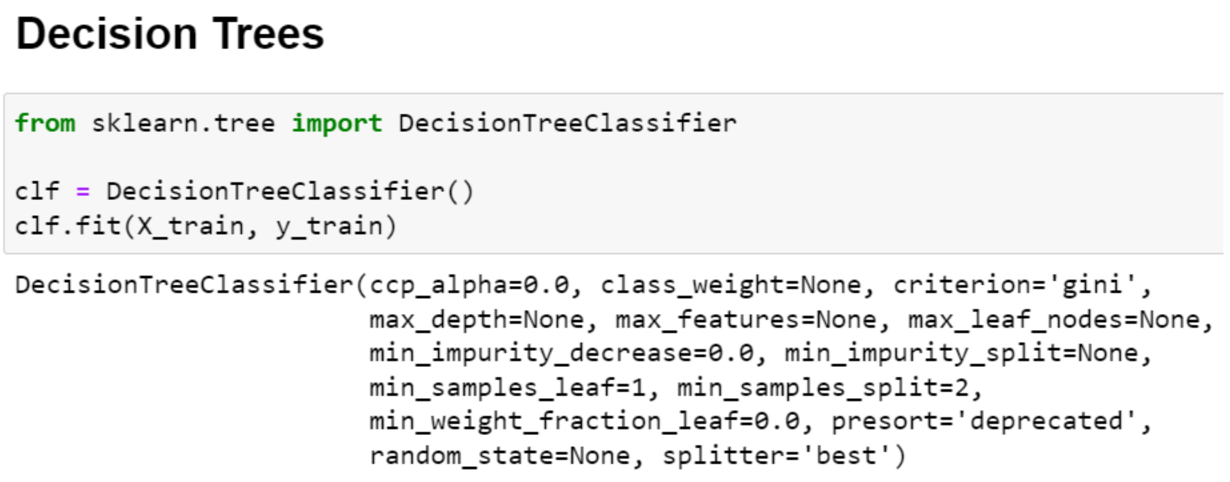


SVC Classifier with cross validation

```
clf = SVC()
param = [{'C': [10**-2, 10**-1, 10**0, 10**1, 10**2], 'gamma':[0.01, 0.1, 1, 10]}]

cv  = GridSearchCV(clf, param,cv=5, scoring='accuracy', return_train_score=True)
cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid=[{'C': [0.01, 0.1, 1, 10, 100],
                          'gamma': [0.01, 0.1, 1, 10]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=0)
```

9) **<u>Decision Tree Clasifier:</u>**

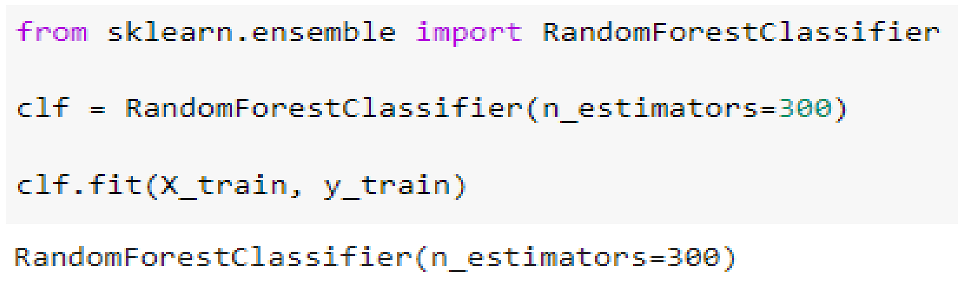


Decision Trees

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

10) **<u>Random Forest Classifier:</u>**

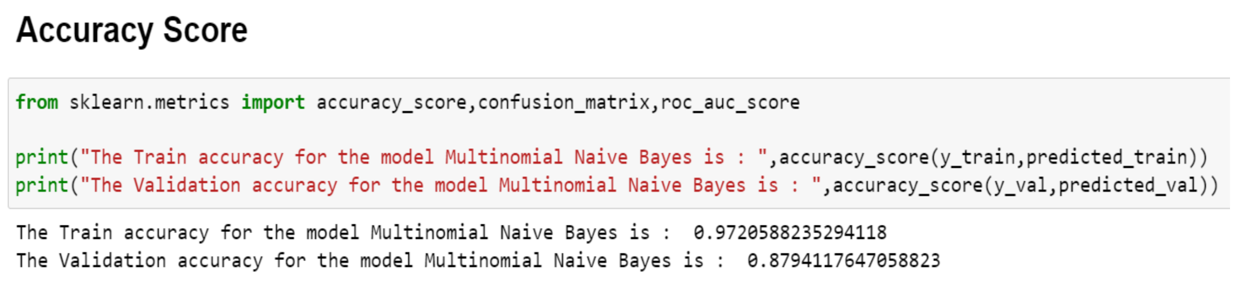


```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=300)

clf.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=300)
```

8. <u>Preliminary Results:</u> As previously stated, we have train and validation datasets in addition to the test dataset. We also stated in the preceding section that we have tried two models so far. The accuracy and confusion matrix can be seen in the screenshots below.

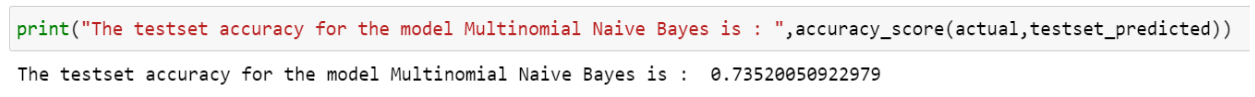
1) **<u>Multinomial Naive Bayes:</u>**



Accuracy Score

```
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score

print("The Train accuracy for the model Multinomial Naive Bayes is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the model Multinomial Naive Bayes is : ",accuracy_score(y_val,predicted_val))
```

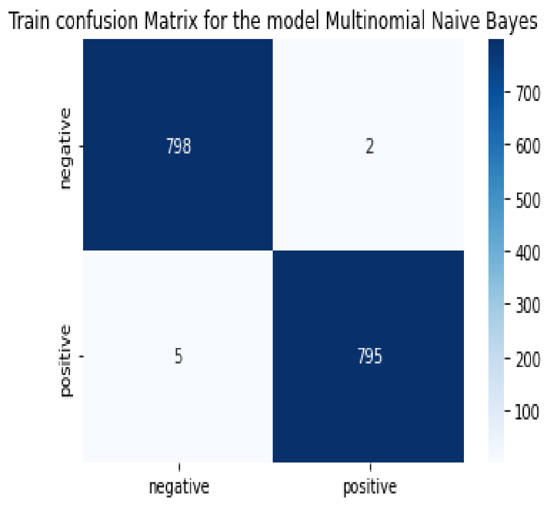
```
The Train accuracy for the model Multinomial Naive Bayes is :  0.9720588235294118
The Validation accuracy for the model Multinomial Naive Bayes is :  0.8794117647058823
```

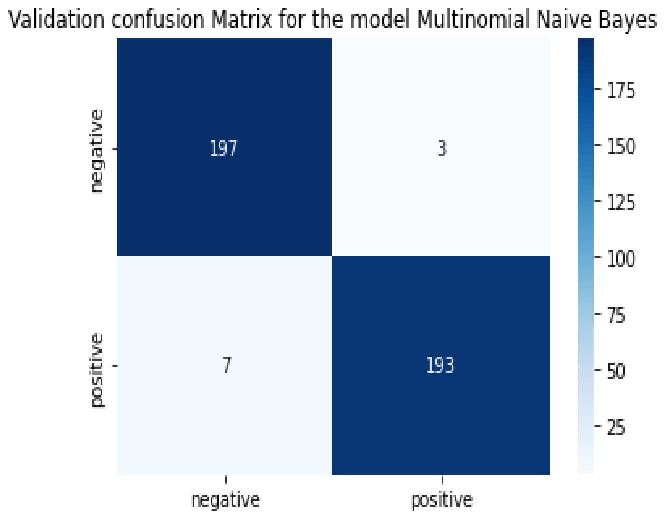


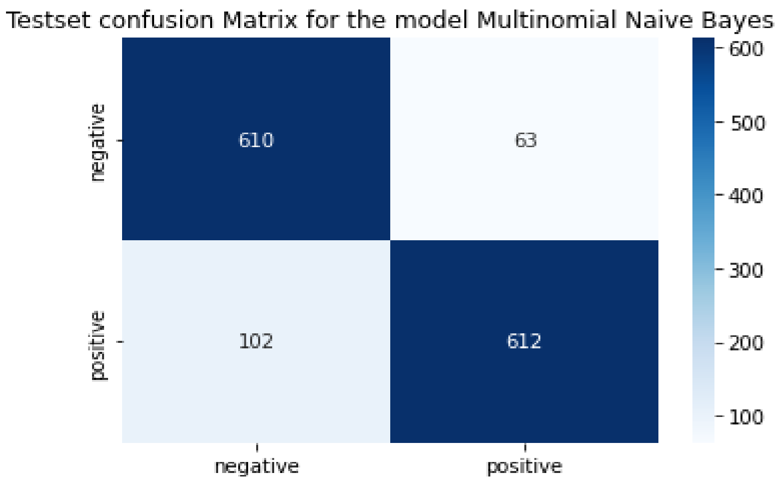
```
print("The testset accuracy for the model Multinomial Naive Bayes is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for the model Multinomial Naive Bayes is :  0.73520050922979
```

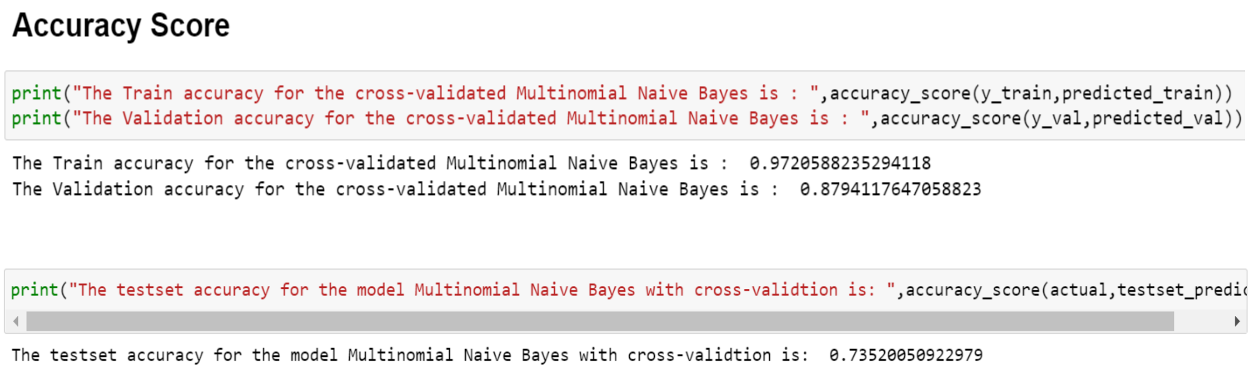
Train ,Validation and Test Confusion Matrix:







2) <u>Multinomial Naive Bayes Cross Validation:</u>



Accuracy Score

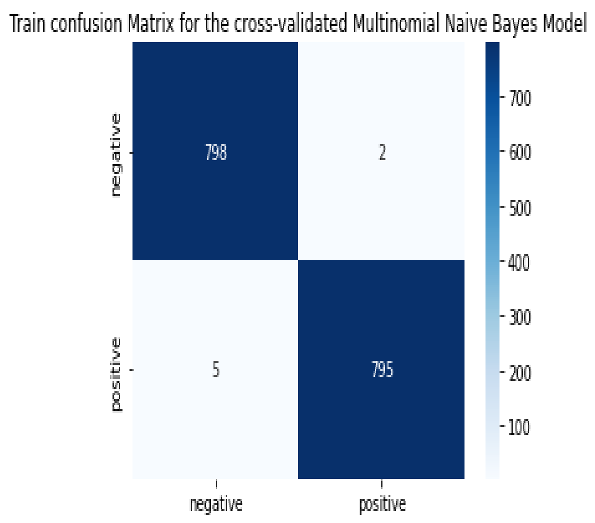
```
print("The Train accuracy for the cross-validated Multinomial Naive Bayes is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the cross-validated Multinomial Naive Bayes is : ",accuracy_score(y_val,predicted_val))
```

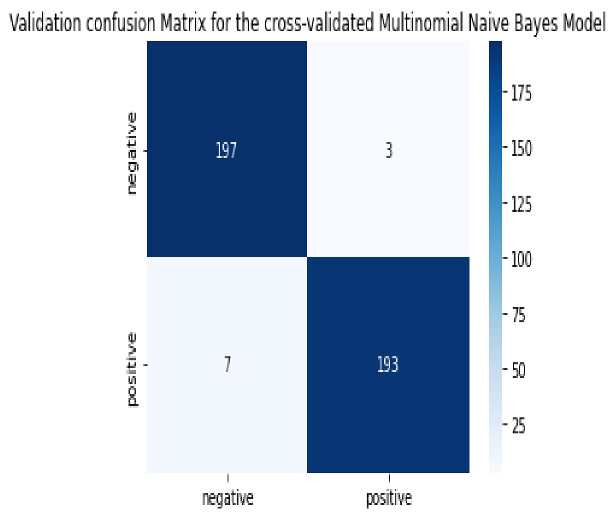
```
The Train accuracy for the cross-validated Multinomial Naive Bayes is :  0.9720588235294118
The Validation accuracy for the cross-validated Multinomial Naive Bayes is :  0.8794117647058823
```

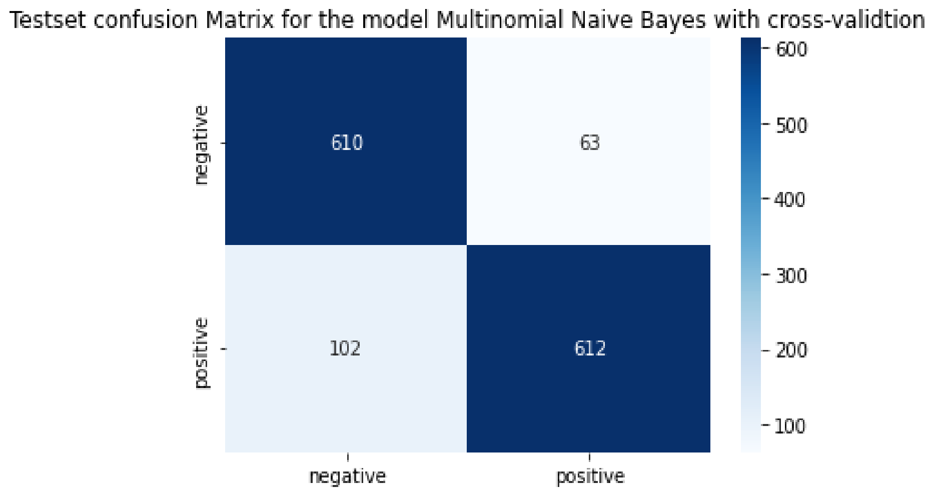
```
print("The testset accuracy for the model Multinomial Naive Bayes with cross-validtion is: ",accuracy_score(actual,testset_predi
```

```
The testset accuracy for the model Multinomial Naive Bayes with cross-validtion is:  0.73520050922979
```

Train , Validation and Test Confusion Matrix:







3) <u>**Logistic Regression:**</u>

Accuracy Score:

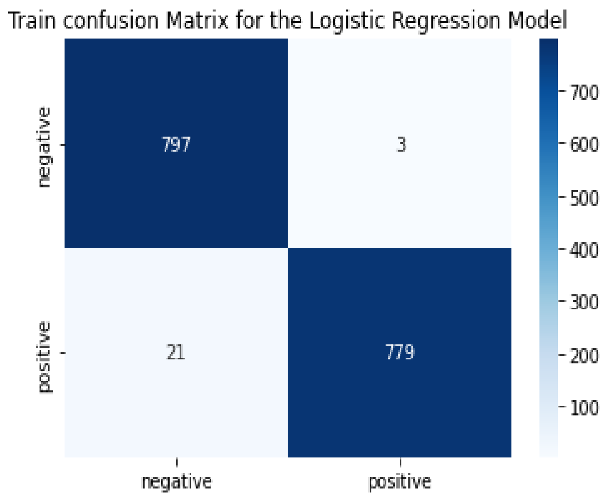
```
print("The Train accuracy for Logistic Regression model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for Logistic Regression model is : ",accuracy_score(y_val,predicted_val))
```

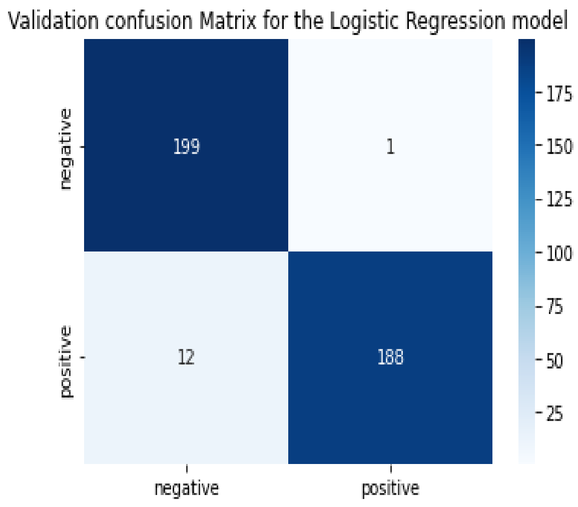
```
The Train accuracy for Logistic Regression model is :  0.9569852941176471
The Validation accuracy for Logistic Regression model is :  0.888235294117647
```

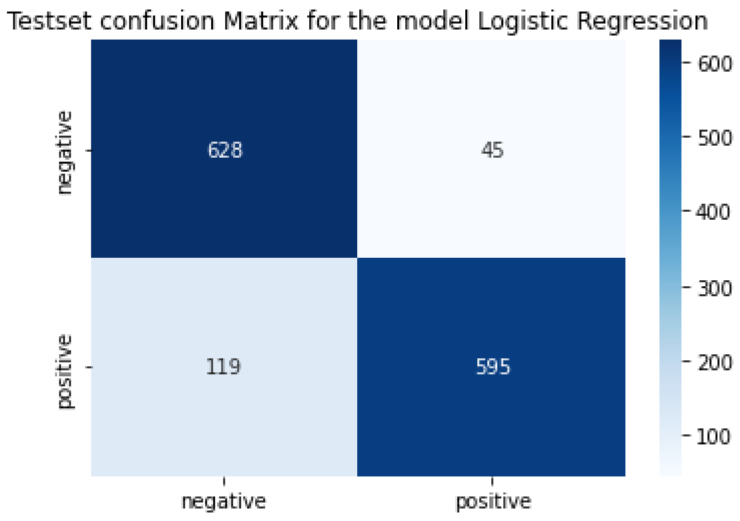
```
print("The testset accuracy for the model Logistic Regression is: ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for the model Logistic Regression is:  0.7708465945257797
```

Train and Validation Confusion Matrix:

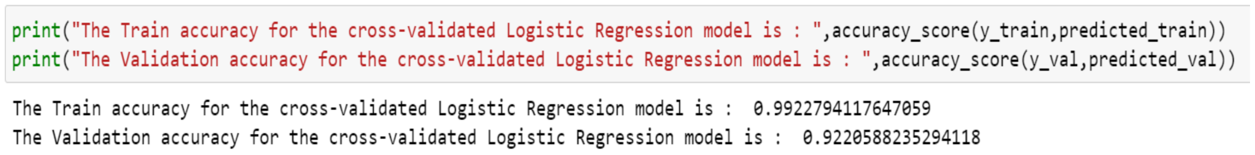






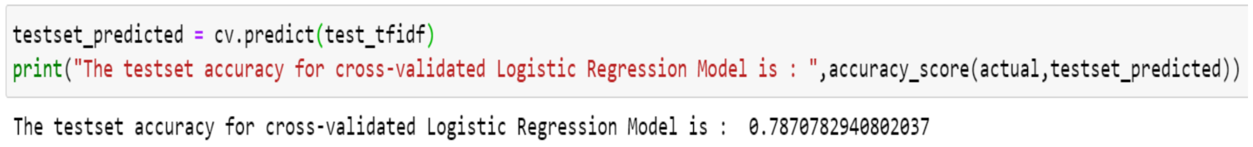
4) <u>Logistic Regression Cross Validation:</u>

Accuracy Score:



```
print("The Train accuracy for the cross-validated Logistic Regression model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the cross-validated Logistic Regression model is : ",accuracy_score(y_val,predicted_val))
```

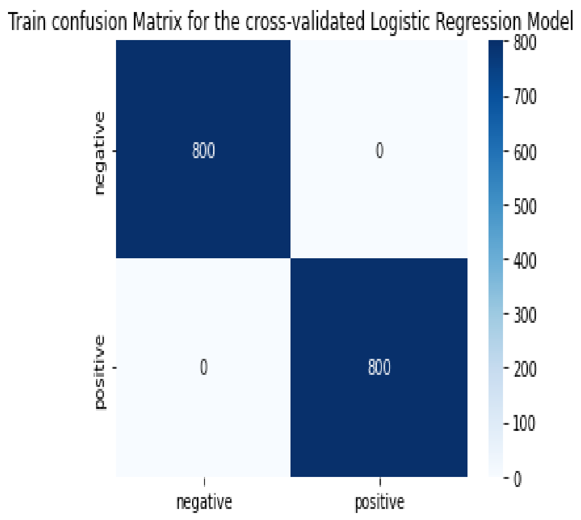
```
The Train accuracy for the cross-validated Logistic Regression model is :  0.9922794117647059
The Validation accuracy for the cross-validated Logistic Regression model is :  0.9220588235294118
```

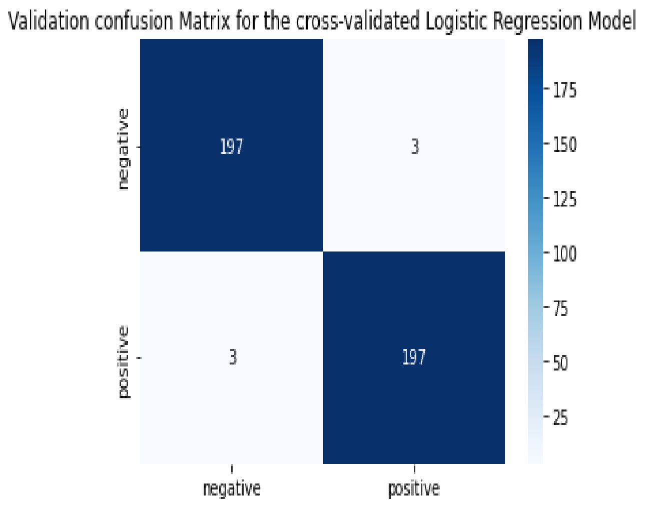


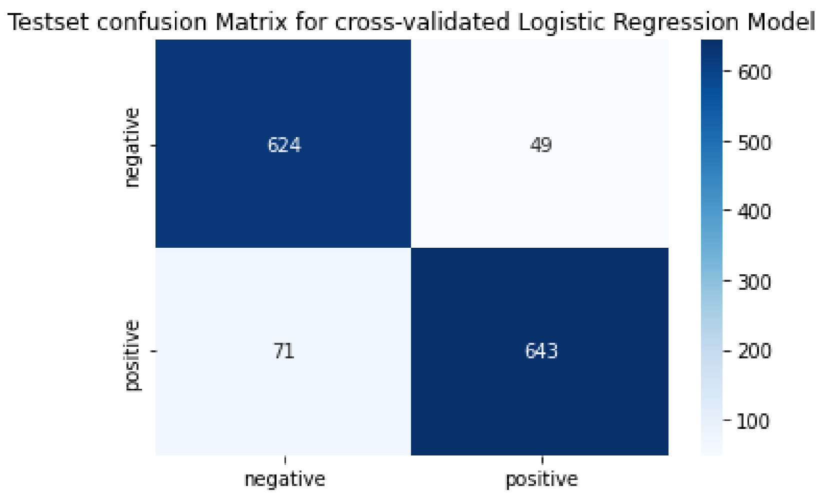
```
testset_predicted = cv.predict(test_tfidf)
print("The testset accuracy for cross-validated Logistic Regression Model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for cross-validated Logistic Regression Model is :  0.7870782940802037
```

Train , Validation and Test Confusion Matrix:

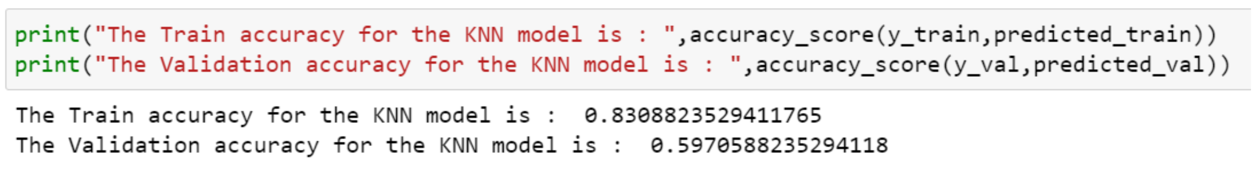






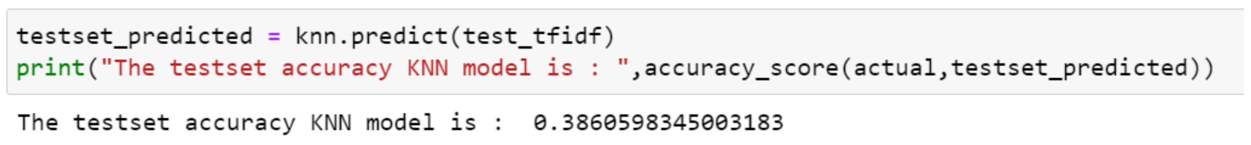
5) <u>K-Nearest Neighbours:</u>

Accuracy Score:



```
print("The Train accuracy for the KNN model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the KNN model is : ",accuracy_score(y_val,predicted_val))
```

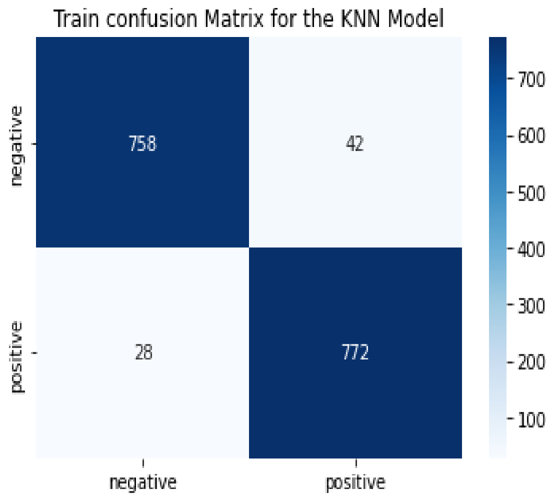
```
The Train accuracy for the KNN model is :  0.8308823529411765
The Validation accuracy for the KNN model is :  0.5970588235294118
```

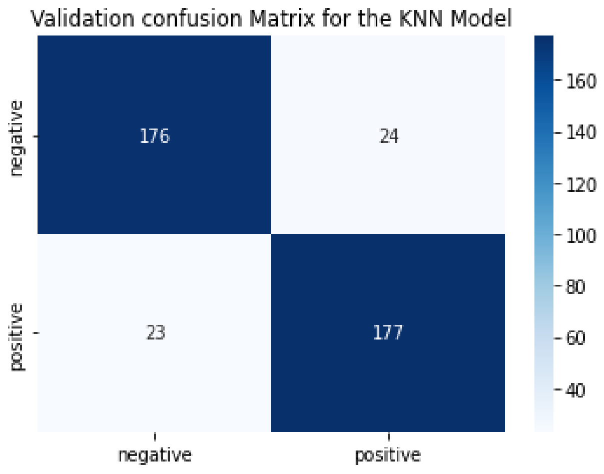


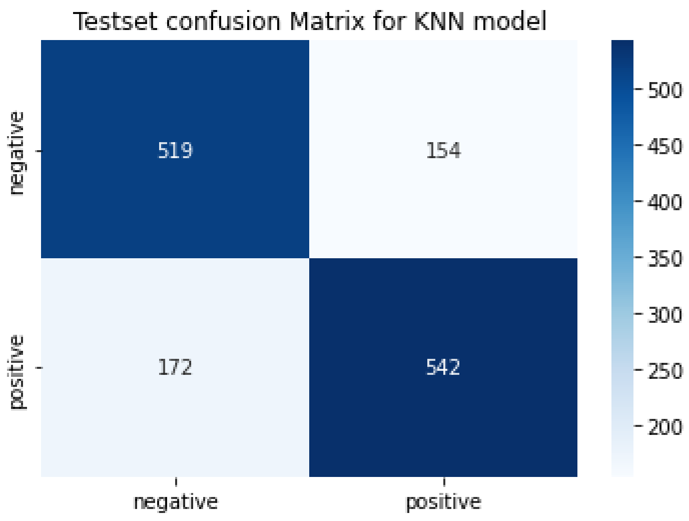
```
testset_predicted = knn.predict(test_tfidf)
print("The testset accuracy KNN model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy KNN model is :  0.3860598345003183
```

Train , Validation and Test Confusion Matrix:

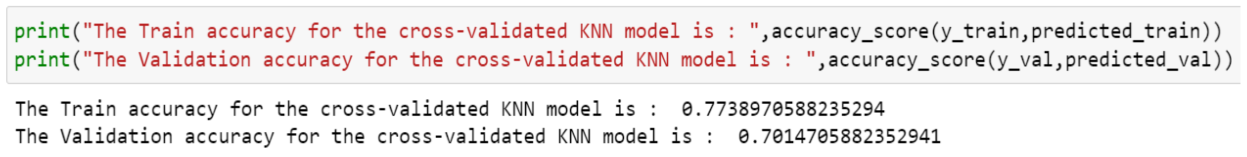






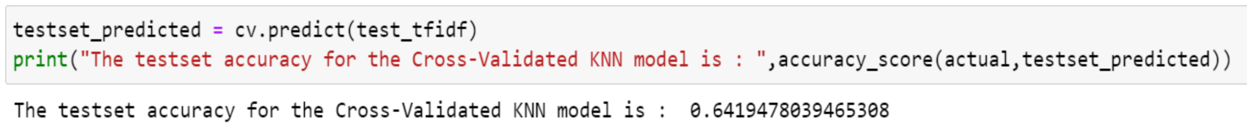
6) <u>K-Nearest Neighbours with Cross Validation:</u>

Accuracy Score:



```
print("The Train accuracy for the cross-validated KNN model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the cross-validated KNN model is : ",accuracy_score(y_val,predicted_val))
```

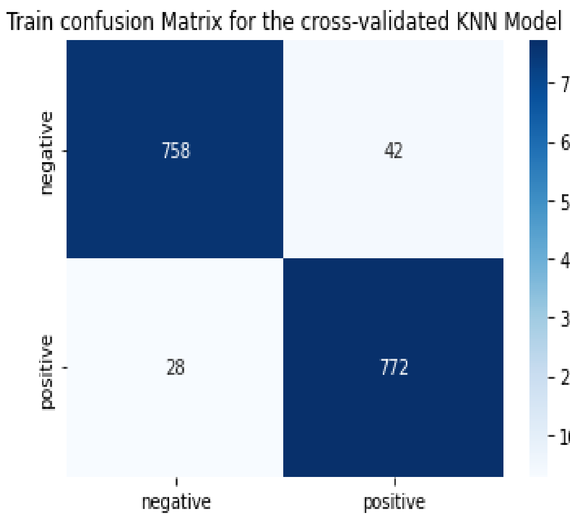
```
The Train accuracy for the cross-validated KNN model is :  0.7738970588235294
The Validation accuracy for the cross-validated KNN model is :  0.7014705882352941
```

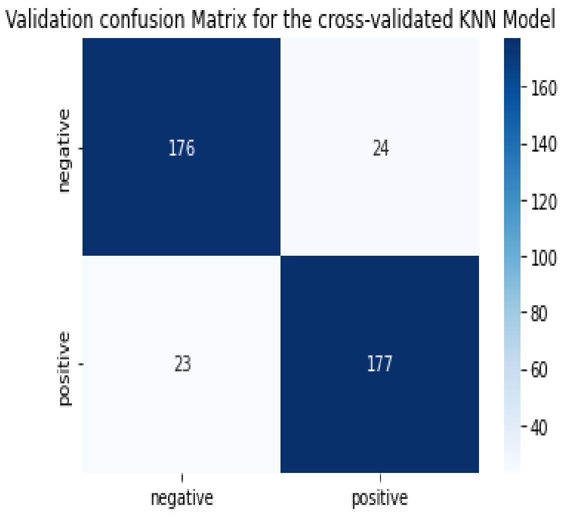


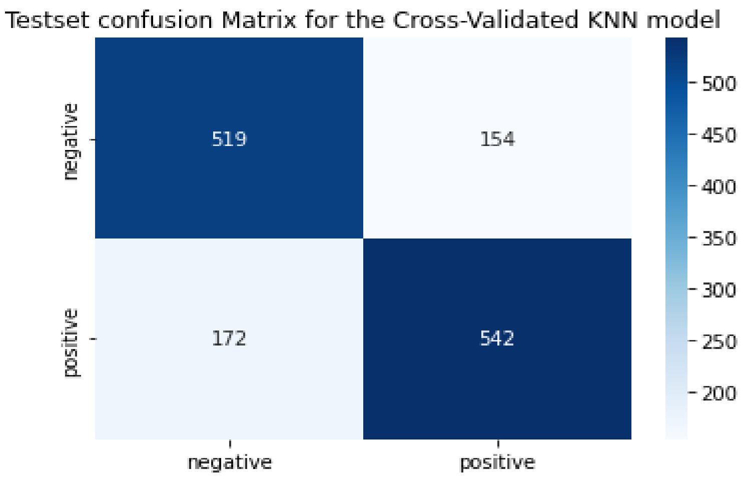
```
testset_predicted = cv.predict(test_tfidf)
print("The testset accuracy for the Cross-Validated KNN model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for the Cross-Validated KNN model is :  0.6419478039465308
```

Train , Validation and Test Confusion Matrix:







7) <u>Support Vector Machine Classifier (SVM):</u>

Accuracy Score:

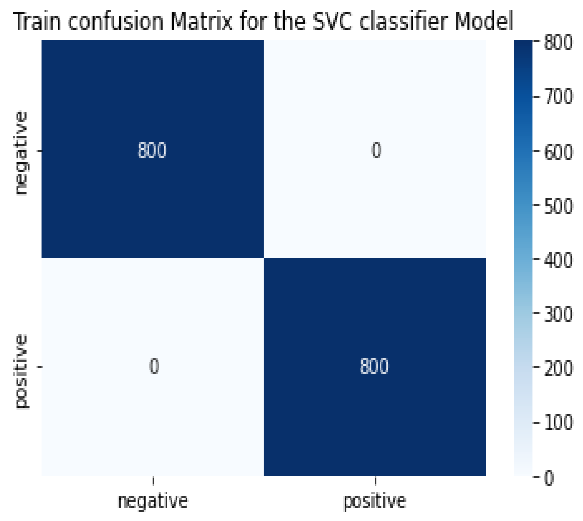
```
print("The Train accuracy for the SVC classifier is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the SVC classifier is : ",accuracy_score(y_val,predicted_val))
```

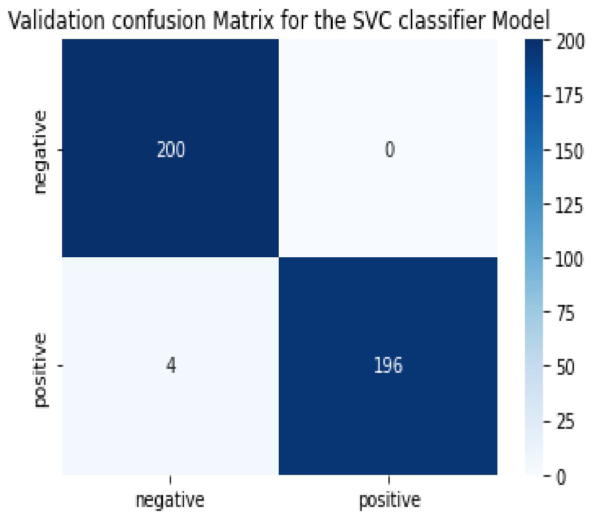
```
The Train accuracy for the SVC classifier is :  0.9915441176470589
The Validation accuracy for the SVC classifier is :  0.9073529411764706
```

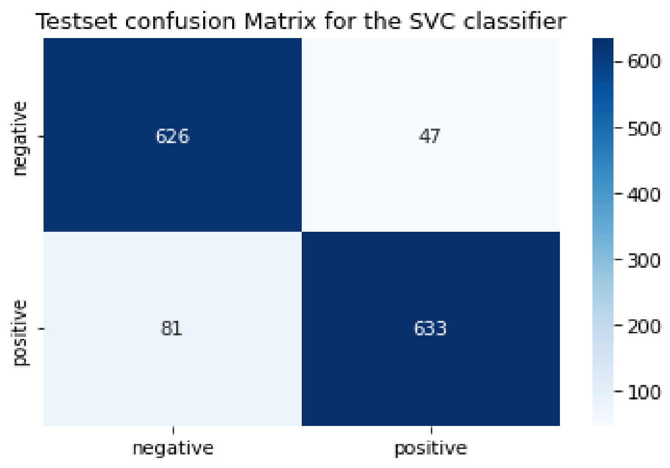
```
testset_predicted = svc.predict(test_tfidf)
print("The testset accuracy for the SVC classifier is is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for the SVC classifier is is :  0.7797581158497772
```

Train , Validation and Test Confusion Matrix:

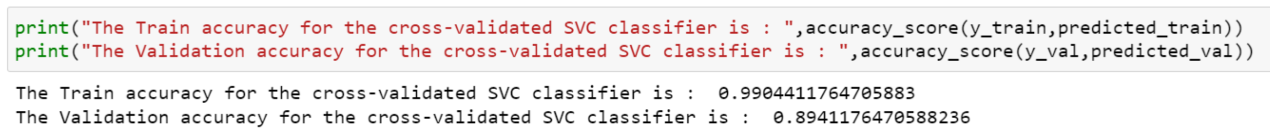






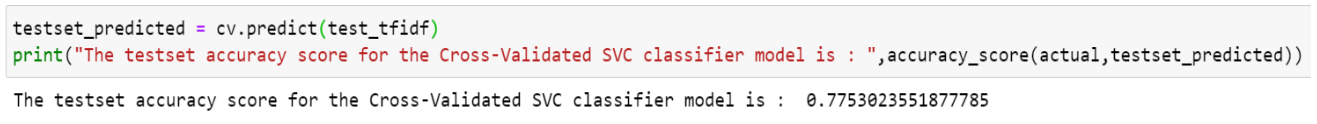
8) <u>SVM Classifier with Cross Validation:</u>

Accuracy Score:



```
print("The Train accuracy for the cross-validated SVC classifier is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the cross-validated SVC classifier is : ",accuracy_score(y_val,predicted_val))
```

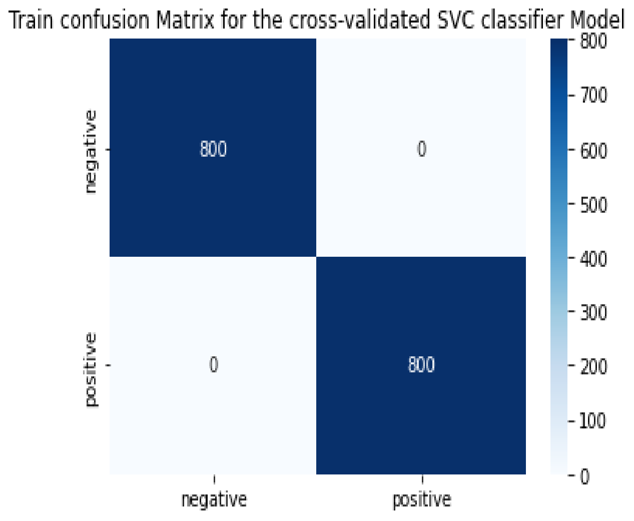
```
The Train accuracy for the cross-validated SVC classifier is :  0.9904411764705883
The Validation accuracy for the cross-validated SVC classifier is :  0.8941176470588236
```

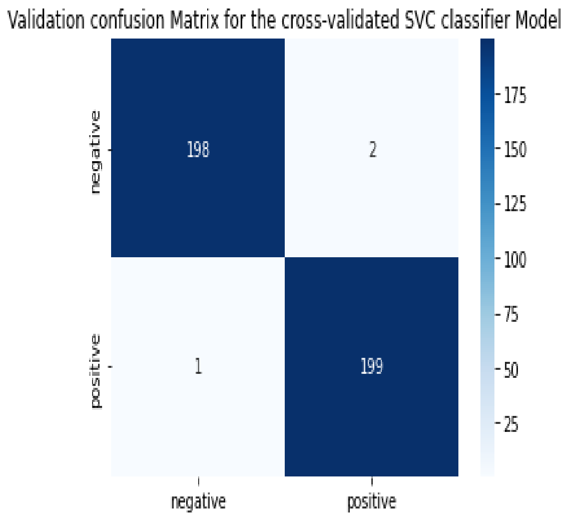


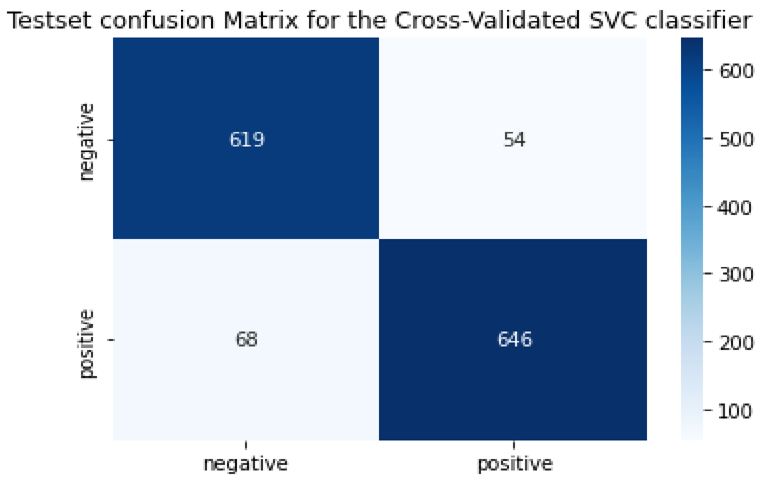
```
testset_predicted = cv.predict(test_tfidf)
print("The testset accuracy score for the Cross-Validated SVC classifier model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy score for the Cross-Validated SVC classifier model is :  0.7753023551877785
```

Train , Validation and Test Confusion Matrix:







9) **Decision Trees:**

Accuracy Score:

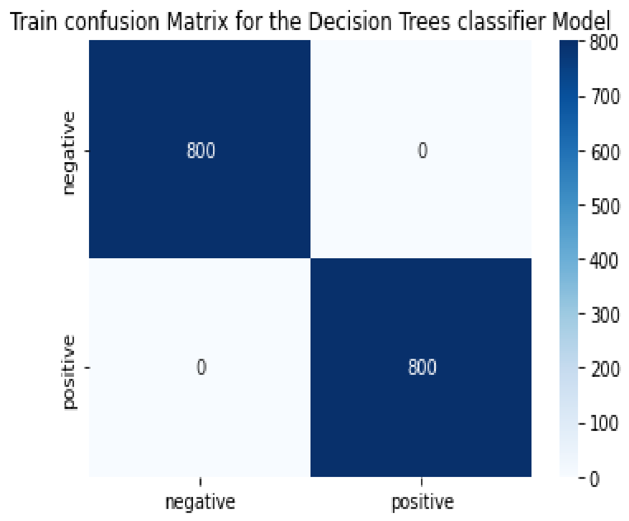
```
print("The Train accuracy for the Decision Trees classifier Model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the Decision Trees classifier Model is : ",accuracy_score(y_val,predicted_val))
```

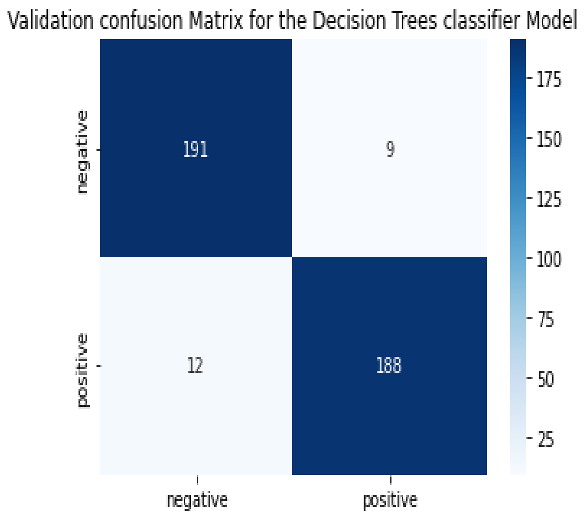
```
The Train accuracy for the Decision Trees classifier Model is :  0.993014705882353
The Validation accuracy for the Decision Trees classifier Model is :  0.9147058823529411
```

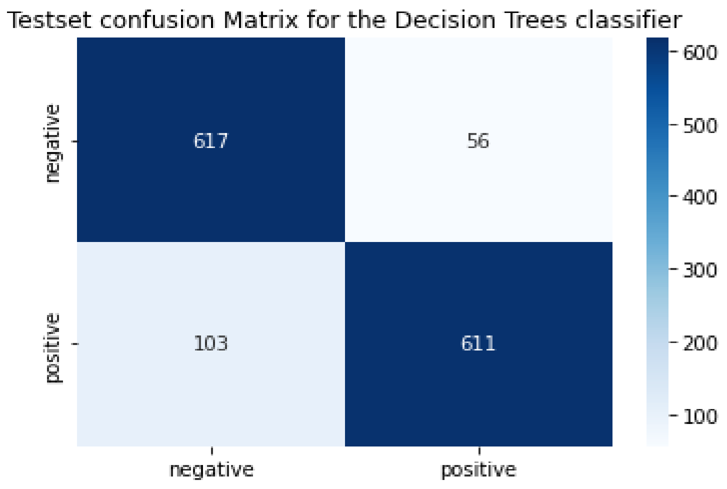
```
testset_predicted = clf.predict(test_tfidf)
print("The testset accuracy score for the Decision Trees classifier Model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy score for the Decision Trees classifier Model is :  0.7947167409293444
```

Train , Validation and Test Confusion Matrix:







10) **Random Forest:**

Accuracy Score:

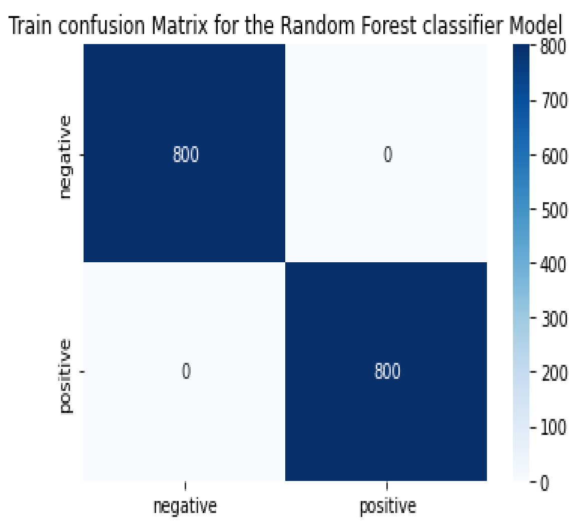
```
print("The Train accuracy for the Random Forest classifier Model is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy for the Random Forest classifier Model is : ",accuracy_score(y_val,predicted_val))
```

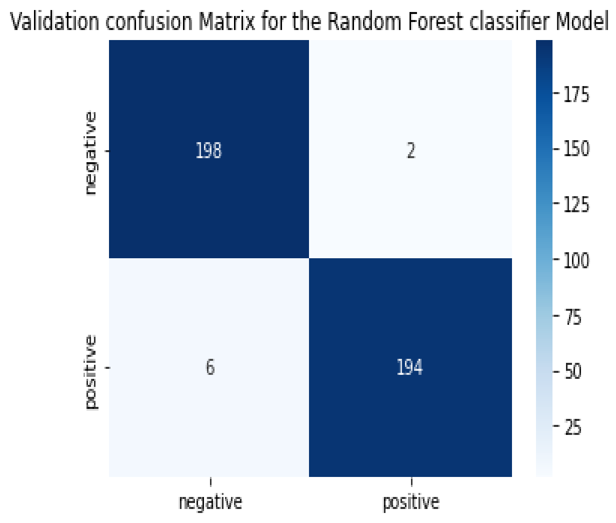
```
The Train accuracy for the Random Forest classifier Model is :  1.0
The Validation accuracy for the Random Forest classifier Model is :  0.98
```

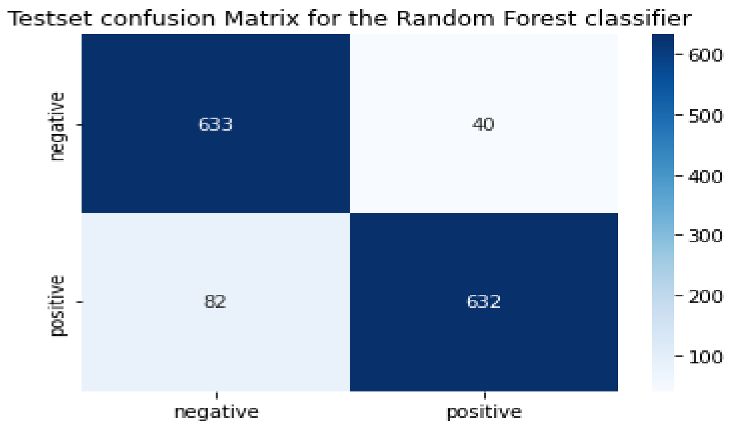
```
testset_predicted = clf.predict(test_tfidf)
print("The testset accuracy for the Random Forest classifier model is : ",accuracy_score(actual,testset_predicted))
```

```
The testset accuracy for the Random Forest classifier model is :  0.9120403749098774
```

Train , Validation and Test Confusion Matrix:





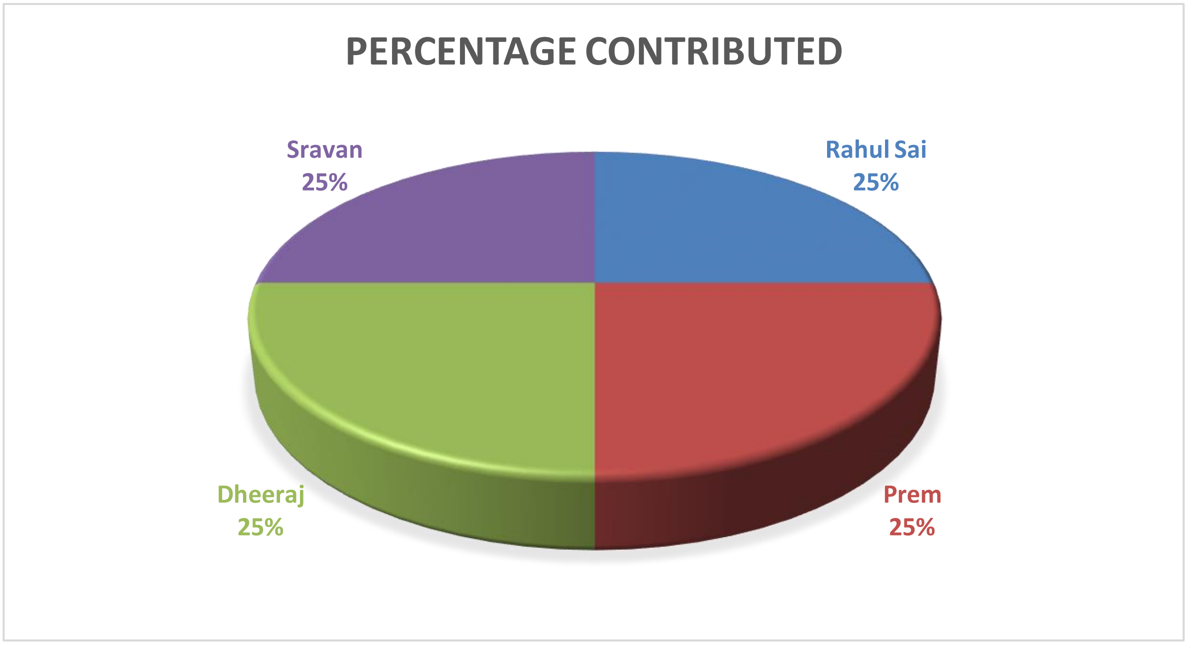


9. Project Management:

Work Completed :

• **Description:** The Text Emotion Detection was chosen primarily because the class labels are connected to the sentiment analysis problem and more closely resemble the intent classification problem utilized in chatbots. Our first step was to get hold of a dataset which we found it on Kaggle which is an ocean of data. Preparing the data to perform analysis is the next step. Thereafter we processed the extracted text data. Before Training the test split data we performed Feature Extraction. Modelling involved Multinomial Naïve Bayes Cross Validation Model, Multinomial Naïve Bayes Model, Logistic Regression Model, Logistic Regression Cross Validation Model, KNN Classifier Model, KNN Classifier Cross Validation Model, SVM Classifier Model, SVM Classifier Cross Validation Model, Decision Tree Model, Random Forest Model.

- **Responsibility (Task, Person):**
  - Gathering dataset and processing the dataset – Prem Sai, Rahul
  - Converting the text data into numerical Features – Rahul, Dheeraj
  - Analyzing the numerical data and separating the data – Sravan, Prem Sai
  - Text Processing and Feature Extraction – Sravan, Dheeraj
  - Train Test Split & Modeling – Rahul, Sravan, Dheeraj, Prem
  - Calculation accuracy score – Dheeraj, Prem, Rahul, Sravan
- **Contributions (members/percentage):**



- **Issues/Concerns:** N/A

8.References:

1. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python and-nltk-c52b92a7c73a
2. https://stackabuse.com/text-classification-with-python-and-scikit-learn/
3. https://www.datacamp.com/tutorial/naive-bayes-scikit-learn
4. https://www.kaggle.com/datasets/anjaneyatripathi/emotion-classification-nlp
5. https://www.javatpoint.com/machine-learning-naive-bayes-classifier