```
import pandas as pd
import numpy as np
```

Reading the files

```
train_df = pd.read_csv('emotion-labels-train.csv')
test_df = pd.read_csv('emotion-labels-test.csv')
```

```
train_df.head()
```

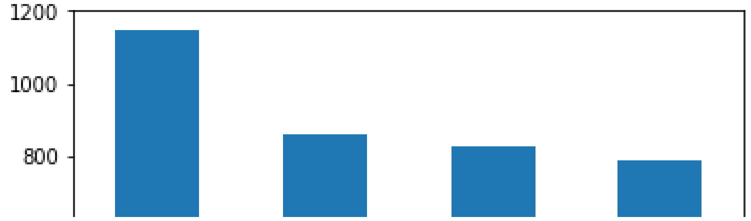
	text	label
0	Just got back from seeing @GaryDelaney in Burs...	joy
1	Oh dear an evening of absolute hilarity I don'...	joy
2	Been waiting all week for this game ❤️❤️❤️ #ch...	joy
3	@gardiner_love : Thank you so much, Gloria! Yo...	joy
4	I feel so blessed to work with the family that...	joy

```
test_df.head()
```

	text	label
0	You must be knowing #blithe means (adj.) Happ...	joy
1	Old saying 'A #smile shared is one gained for ...	joy
2	Bridget Jones' Baby was bloody hilarious 😅 #Br...	joy
3	@Elaminova sparkling water makes your life spa...	joy
4	I'm tired of everybody telling me to chill out...	joy

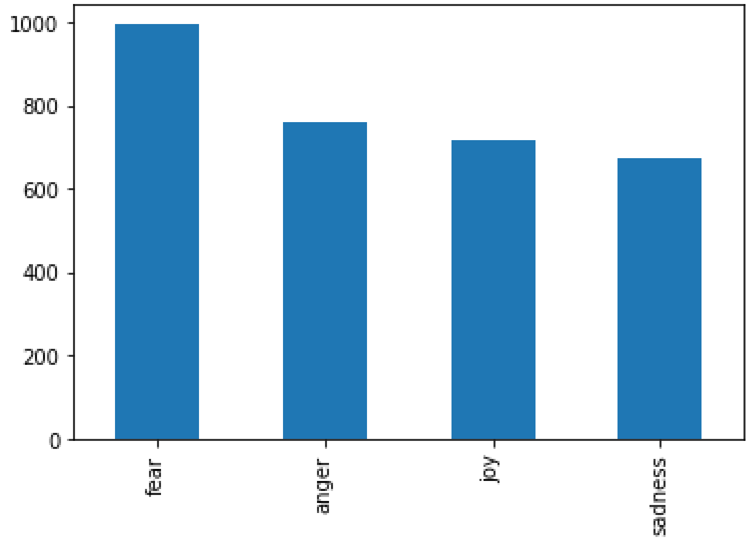
```
train_df.label.value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f881c156f50>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f881c051650>
```



```
train_df = train_df[train_df.label != 'anger']
train_df = train_df[train_df.label != 'fear']
test_df = test_df[test_df.label != 'anger']
test_df = test_df[test_df.label != 'fear']
```

```
train_df.shape,test_df.shape
```

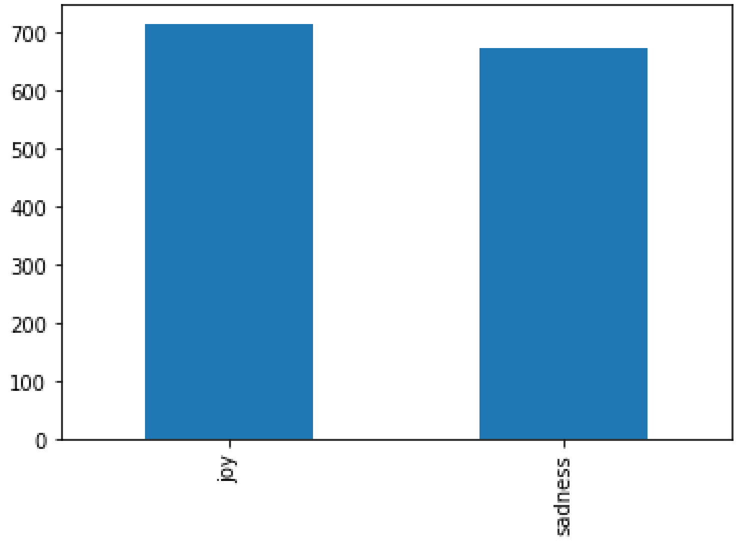
```
((1609, 2), (1387, 2))
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f881bb85ad0>
```

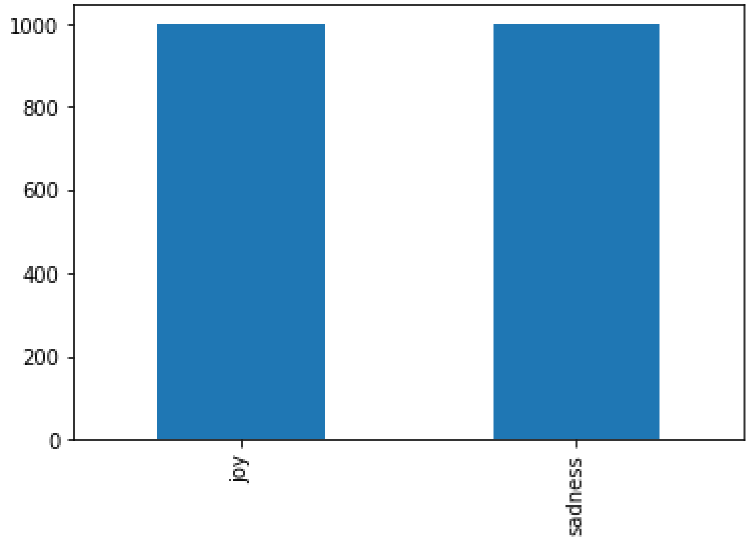
```
test_df.label.value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f881baf54d0>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f881ba7bd10>
```



Created Dataframe with Reviews and Labels

Preprocessing Section

```
def label_encoding(label):
    if label=='joy':
        return '1'
    else :
        return '0'
```

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

```
train_df.head()
```

	text	label
328	@Rbrutti what a #happy looking #couple !	1
652	@flutterpolitely she is from the heyday 80's -...	1
809	Studying the phases of oscillation. How jovial...	1
575	#food #деньги #smile microsoft_.net_framewo...	1
343	Watch this amazing live.ly broadcast by @elise...	1

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

```
test_df.head()
```

	text	label
0	You must be knowing #blithe means (adj.) Happ...	1
1	Old saying 'A #smile shared is one gained for ...	1
2	Bridget Jones' Baby was bloody hilarious 😅 #Br...	1
3	@Elaminova sparkling water makes your life spa...	1
4	I'm tired of everybody telling me to chill out...	1

```
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()
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
nltk.download('punkt')

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)
```

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

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

```
{'ain', 'being', 'so', 'has', 'that', 've', "you're", 'in', 'myself', "hasn't", 'shouldn
```

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

```
328                  rbrutti what a happy looking couple
652    flutterpolitely she is from the heyday 80 is c...
809        studying the phases of oscillation how jovial
575    food smile microsoft net framework 4 5 1 full ...
343    watch this amazing live ly broadcast by elise ...
Name: text, dtype: object
```

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

```
0    you must be knowing blithe means adj happy che...
1    old saying a smile shared is one gained for an...
2    bridget jones baby was bloody hilarious bridge...
3    elaminova sparkling water makes your life sparkly
4    i am tired of everybody telling me to chill ou...
Name: text, dtype: object
```

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

```
328                          rbrutti happy looking couple
652    flutterpolitely heyday 80 could go either way ...
809                 studying phases oscillation jovial
575    food smile microsoft net framework 4 5 1 full ...
343    watch amazing live ly broadcast elise awsome 1...
Name: text, dtype: object
```

```
test_df['text'] = test_df['text'].apply(remove_stopwords)
test_df['text'].head()
```

```
0         must knowing blithe means adj happy cheerful
1    old saying smile shared one gained another day...
2    bridget jones baby bloody hilarious bridgetjon...
3         elaminova sparkling water makes life sparkly
4    tired everybody telling chill everythings ok f...
Name: text, dtype: object
```

```
train_df.head()
```

	text	label
328	rbrutti happy looking couple	1
652	flutterpolitely heyday 80 could go either way ...	1
809	studying phases oscillation jovial	1
575	food smile microsoft net framework 4 5 1 full ...	1
343	watch amazing live ly broadcast elise awsome 1...	1

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(ngram_range=(1,2),max_features=5000)
tfidf.fit(train_df['text'])
train_tfidf = tfidf.transform(train_df['text'])
test_tfidf = tfidf.transform(test_df['text'])
```

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

```
((2000, 5000), (1387, 5000))
```

```
print(tfidf.get_feature_names()[100:140])
```

```
['album', 'album digging', 'alicia', 'alicia year', 'alive', 'almost', 'almost mobile',
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```

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
```

```
model_name = []
train_accuracies = []
val_accuracies = []
test_accuracies = []
```

Multinomial Naive Bayes Model

```
from sklearn.naive_bayes import MultinomialNB

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

```
     MultinomialNB()
```

Prediction of Probabilities

```
train_probabilities = clf.predict_proba(X_train)[:,1]
val_probabilities = clf.predict_proba(X_val)[:,1]
```

```
train_probabilities[:10]
```

```
     array([0.71495331, 0.07091431, 0.28295576, 0.8349996 , 0.83948288,
            0.15024221, 0.75775791, 0.70809465, 0.93790308, 0.17172747])
```

```
val_probabilities[:10]
```

```
     array([0.7438836 , 0.8331312 , 0.91697738, 0.1549029 , 0.59563524,
            0.87658851, 0.82075745, 0.26992965, 0.79039297, 0.14983711])
```

Prediction of log probabilities

```
train_log_probabilities = clf.predict_log_proba(X_train)[:,1]
val_log_probabilities = clf.predict_log_proba(X_val)[:,1]
```

```
train_log_probabilities[:10]
```

```
     array([-0.33553804, -2.64628297, -1.2624647 , -0.18032403, -0.1749692 ,
            -1.89550658, -0.27739132, -0.3451775 , -0.06410866, -1.76184653])
```

```
val_log_probabilities[:10]
```

```
     array([-0.2958707 , -0.18256415, -0.08667247, -1.86495681, -0.51812681,
            -0.1317176 , -0.19752765, -1.30959392, -0.23522502, -1.8982065 ])
```

Prediction of labels

```
predicted_train = clf.predict(X_train)
predicted_val = clf.predict(X_val)
```

Accuracy Score

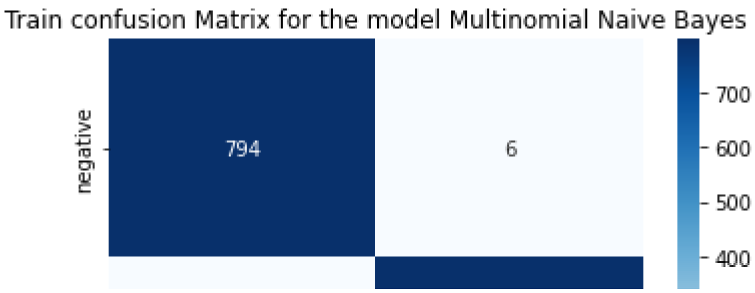
```
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score

print("The Train accuracy for the model Multinomial Naive Bayes is : ",accuracy_score(y_train
print("The Validation accuracy for the model Multinomial Naive Bayes is : ",accuracy_score(y_
```

```
     The Train accuracy for the model Multinomial Naive Bayes is :  0.99375
     The Validation accuracy for the model Multinomial Naive Bayes is :  0.985
```

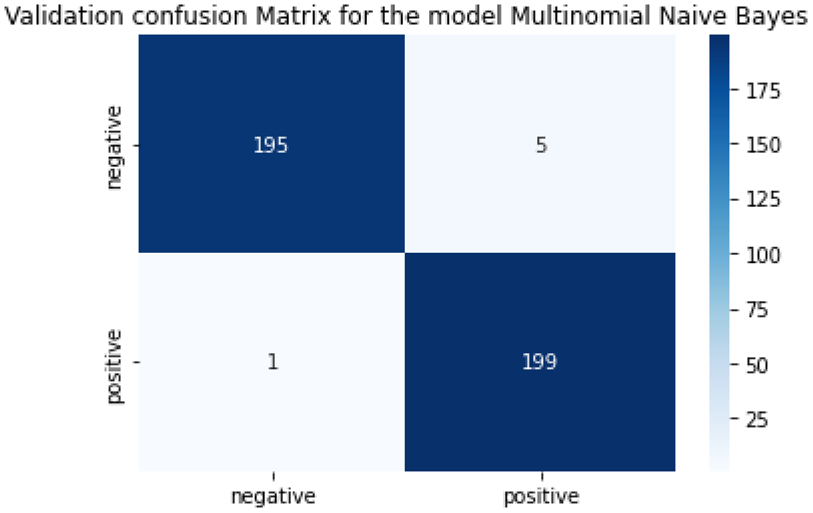
Plotting Confusion Matrix

```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_confusion_matrix(cf_matrix,title):
    sns.heatmap(cf_matrix, annot=True,fmt='g',cmap="Blues",xticklabels=["negative","positive"
    plt.title(title)
    plt.show()
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



```
testset_predicted = clf.predict(test_tfidf)
```

```
actual = test_df.label.values
actual
```

```
     array(['1', '1', '1', ..., '0', '0', '0'], dtype=object)
```

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

```
     The testset accuracy for the model Multinomial Naive Bayes is :  0.8774333093006489
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```

Testset confusion Matrix for the model Multinomial Naive Bayes

```
model_name.append("Multinomial Naive Bayes")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),2))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),2))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),2))
```

Multinomial Naive Bayes with 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, estimator=MultinomialNB(),
             param_grid={'alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
             return_train_score=True, scoring='accuracy')
```

```
predicted_train = cv.predict(X_train)
predicted_val = cv.predict(X_val)
```

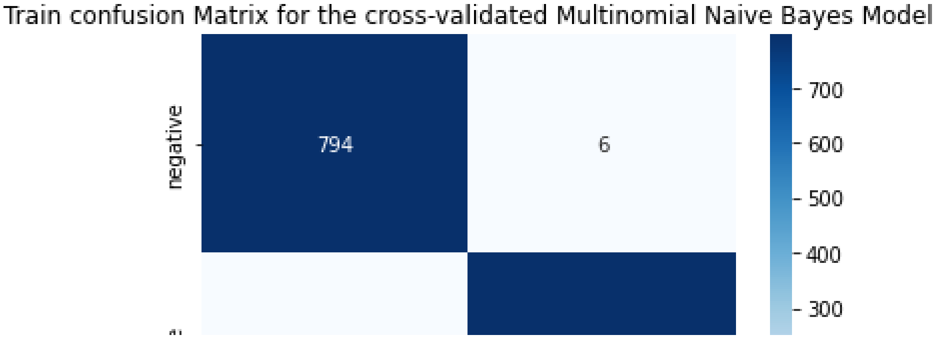
Accuracy Score

```
print("The Train accuracy for the cross-validated Multinomial Naive Bayes is : ",accuracy_sco
print("The Validation accuracy for the cross-validated Multinomial Naive Bayes is : ",accurac
```

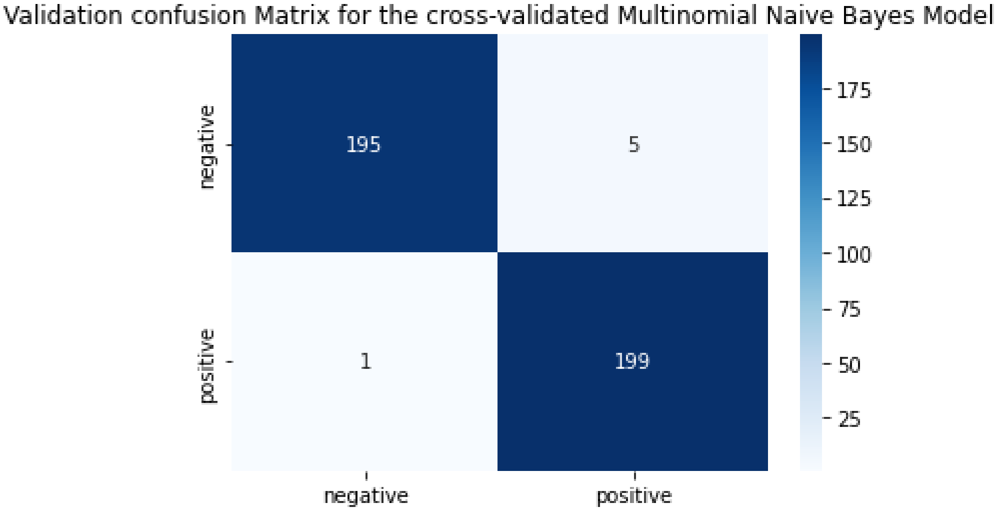
```
The Train accuracy for the cross-validated Multinomial Naive Bayes is :  0.99375
The Validation accuracy for the cross-validated Multinomial Naive Bayes is :  0.985
```

Plotting Confusion Matrix

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



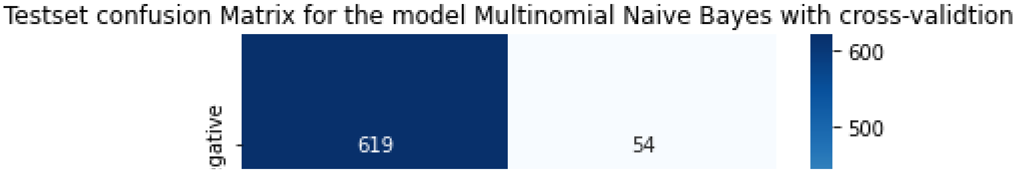
Test Datset Prediction

```
testset_predicted = cv.predict(test_tfidf)
```

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

```
    The testset accuracy for the model Multinomial Naive Bayes with cross-validtion is:  0.8
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("Multinomial Naive Bayes with Cross Validation")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

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

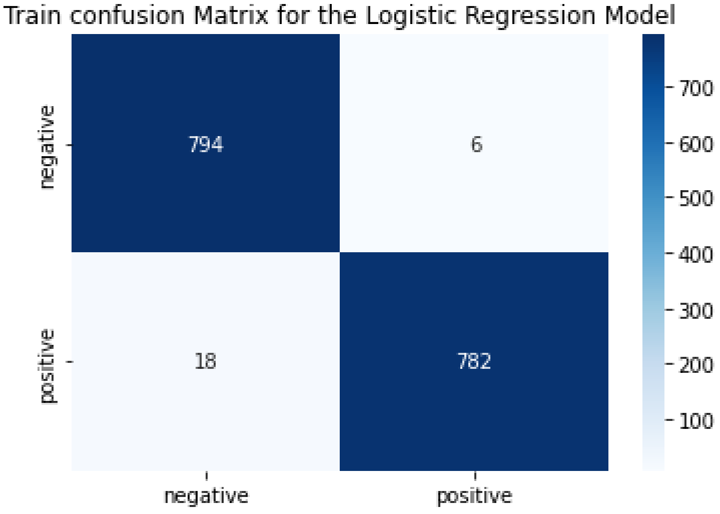
```
     LogisticRegression(C=0.1)
```

```
predicted_train = lr.predict(X_train)
predicted_val = lr.predict(X_val)
```

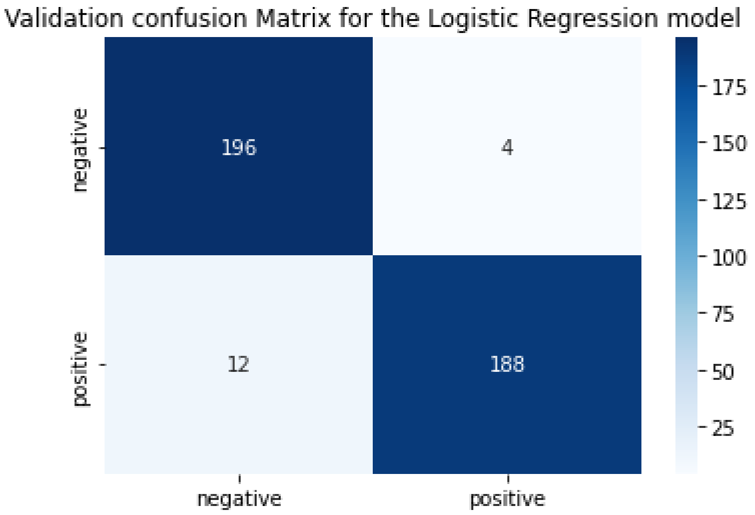
```
print("The Train accuracy for Logistic Regression model is : ",accuracy_score(y_train,predict
print("The Validation accuracy for Logistic Regression model is : ",accuracy_score(y_val,pred
```

```
     The Train accuracy for Logistic Regression model is :  0.985
     The Validation accuracy for Logistic Regression model is :  0.96
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```

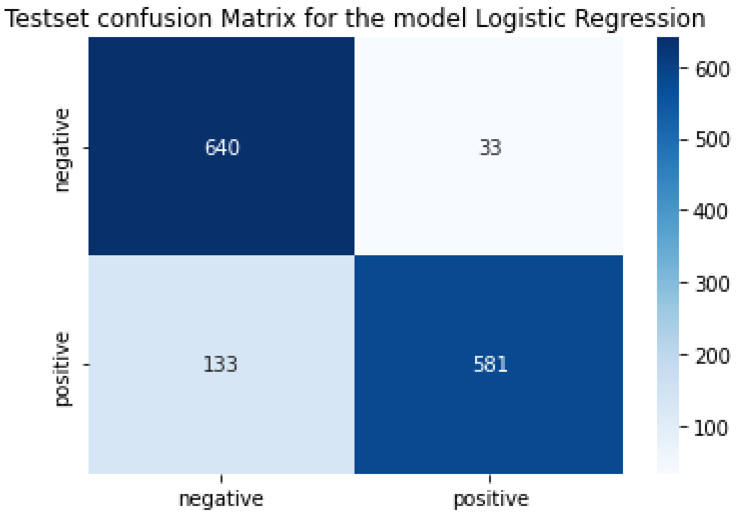


```
testset_predicted = lr.predict(test_tfidf)
```

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

```
     The testset accuracy for the model Logistic Regression is:  0.8803172314347513
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("Logistic Regression")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

Logistic Regression with cross validation using GridSearchCv

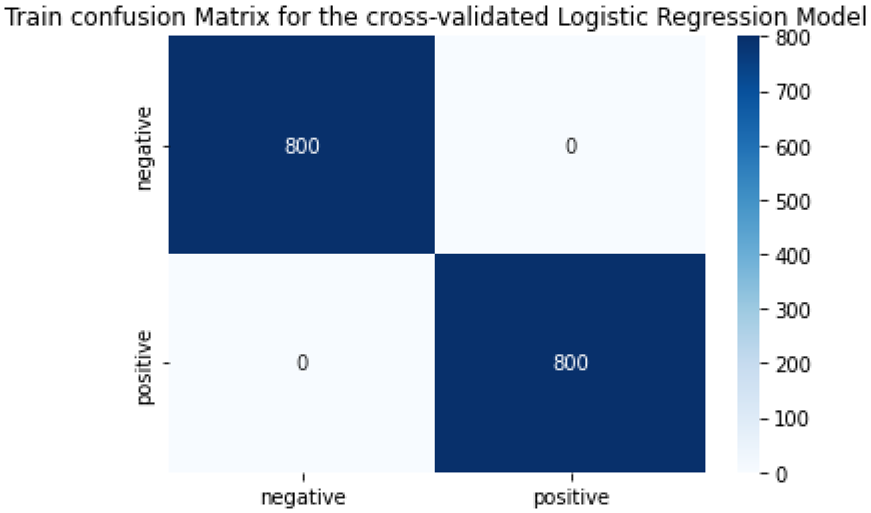
```
grid_values = {'C': [0.001,0.01,0.1,1,10,100]}
cv = GridSearchCV(LogisticRegression(max_iter=250), param_grid=grid_values,cv=5, scoring='acc
cv.fit(X_train, y_train)
```

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

```
predicted_train = cv.predict(X_train)
predicted_val = cv.predict(X_val)


print("The Train accuracy for the cross-validated Logistic Regression model is : ",accuracy_s
print("The Validation accuracy for the cross-validated Logistic Regression model is : ",accur
```

```
    The Train accuracy for the cross-validated Logistic Regression model is :  1.0
    The Validation accuracy for the cross-validated Logistic Regression model is :  0.985
```

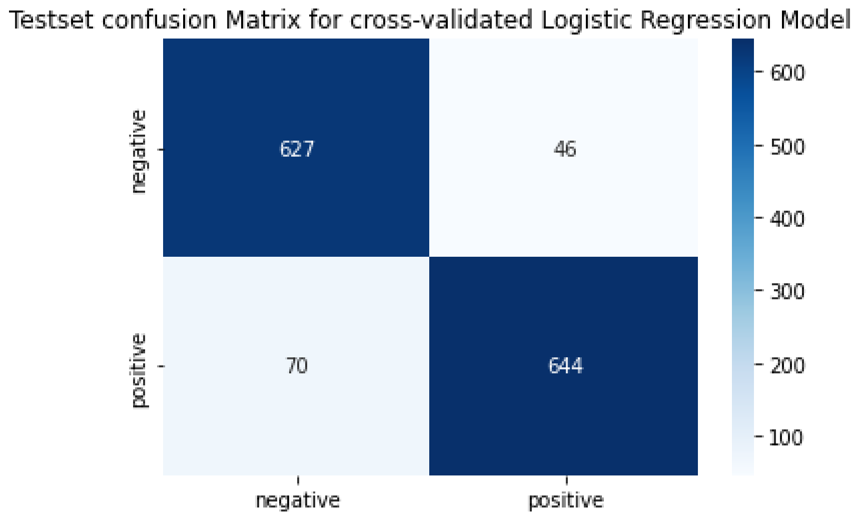
```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
testset_predicted = cv.predict(test_tfidf)
print("The testset accuracy for cross-validated Logistic Regression Model is : ",accuracy_sco
```

```
    The testset accuracy for cross-validated Logistic Regression Model is :  0.916366258111
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("Logistic Regression with Cross Validation")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
```

```
    KNeighborsClassifier(n_neighbors=3)
```

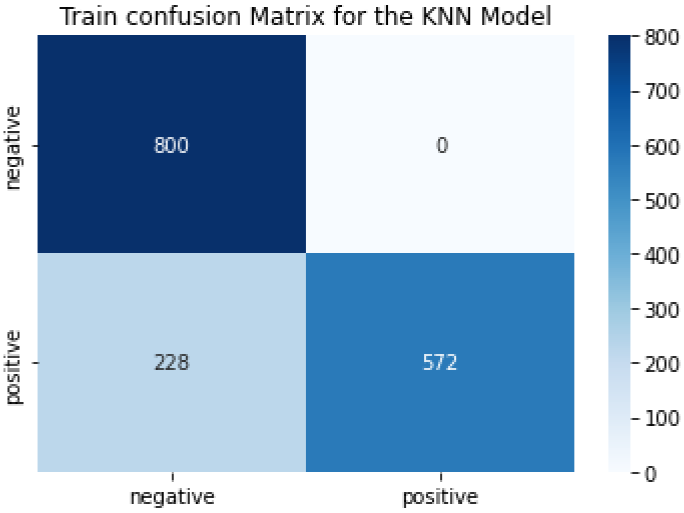
```
predicted_train = knn.predict(X_train)
predicted_val = knn.predict(X_val)
```

```
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))
```

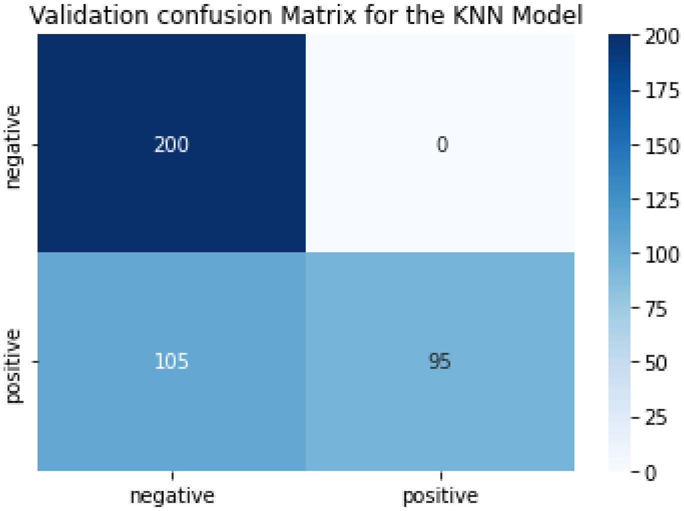
```
    The Train accuracy for the KNN model is :  0.8575
```

```
The Validation accuracy for the KNN model is :  0.7375
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



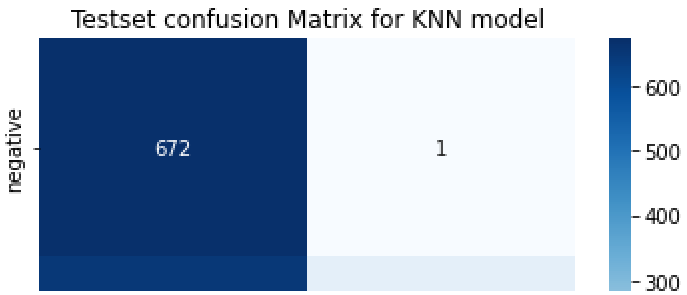
```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



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

```
The testset accuracy KNN model is :  0.5306416726748377
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("KNN-Classifier")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

K-Nearest Neighbours using Cross-Validation

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

```
     GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
                  param_grid={'n_neighbors': [3, 5, 7, 9, 11]},
                  return_train_score=True, scoring='accuracy')
```

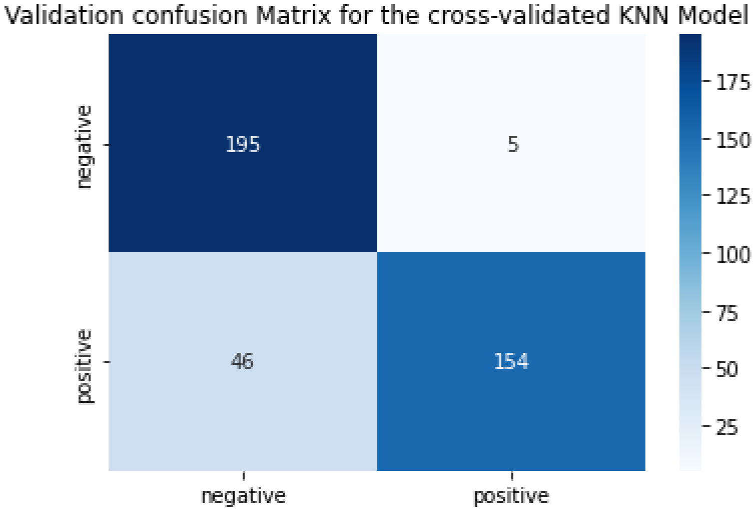
```
predicted_train = cv.predict(X_train)
predicted_val = cv.predict(X_val)
```

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

```
     The Train accuracy for the cross-validated KNN model is :  0.8975
     The Validation accuracy for the cross-validated KNN model is :  0.8725
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```

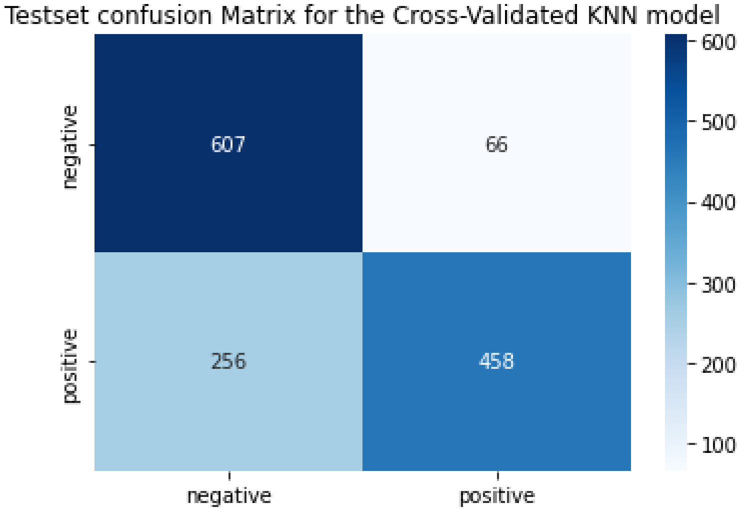
```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



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

```
The testset accuracy for the Cross-Validated KNN model is :  0.7678442682047585
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("KNN-Classifier with Cross-Validation")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

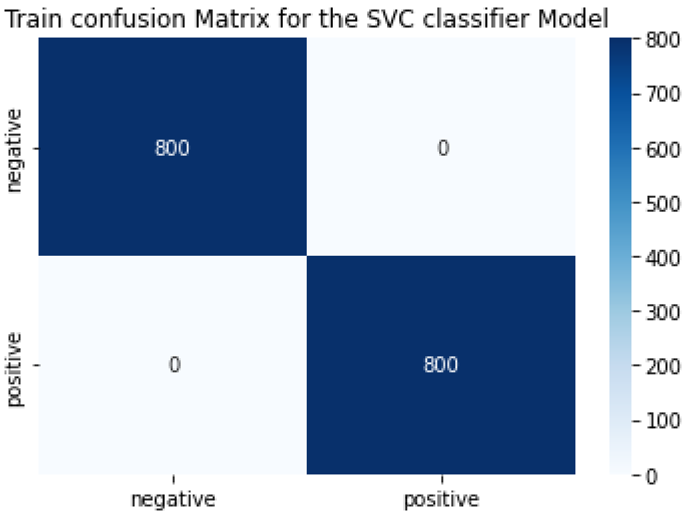
Support Vector Machine Classifier (SVM)

```
from sklearn.svm import SVC

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

     SVC()
```

```
predicted_train = svc.predict(X_train)
predicted_val = svc.predict(X_val)
```

```
print("The Train accuracy for the SVC classifier is : ",accuracy_score(y_train,predicted_trai
print("The Validation accuracy for the SVC classifier is : ",accuracy_score(y_val,predicted_v

     The Train accuracy for the SVC classifier is :  1.0
     The Validation accuracy for the SVC classifier is :  0.9875
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```

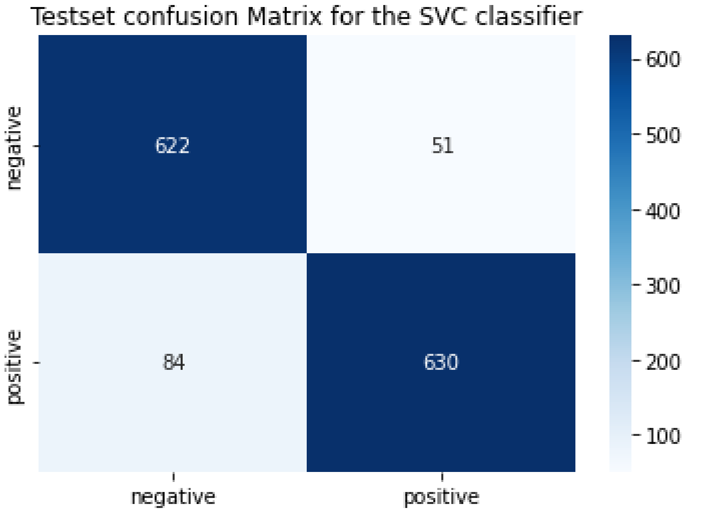


```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```

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

```
The testset accuracy for the SVC classifier is is :  0.9026676279740447
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("SVM-Classifier")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

SVM 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)
```

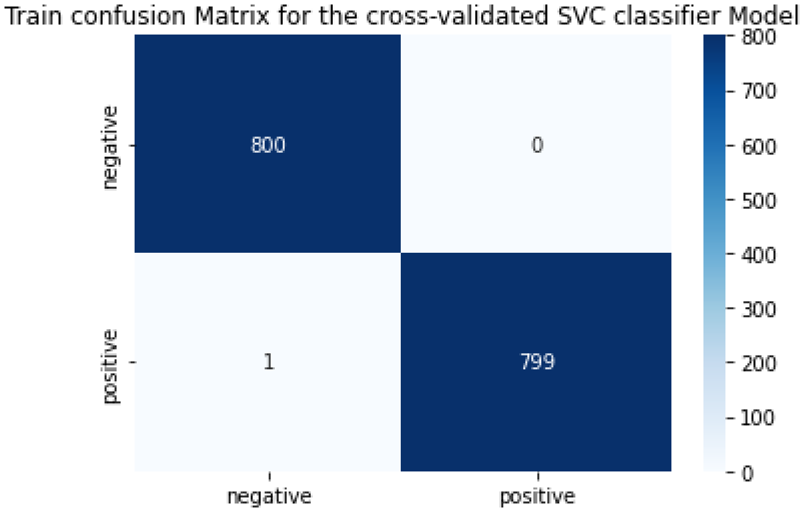
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid=[{'C': [0.01, 0.1, 1, 10, 100],
                          'gamma': [0.01, 0.1, 1, 10]}],
             return_train_score=True, scoring='accuracy')
```

```
predicted_train = cv.predict(X_train)
predicted_val = cv.predict(X_val)
```

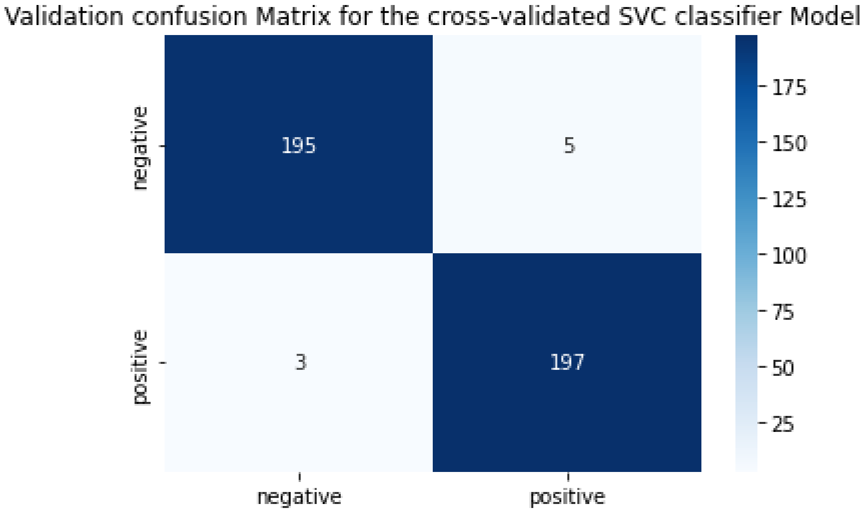
```
print("The Train accuracy for the cross-validated SVC classifier is : ",accuracy_score(y_trai
print("The Validation accuracy for the cross-validated SVC classifier is : ",accuracy_score(y
```

```
The Train accuracy for the cross-validated SVC classifier is :  0.999375
The Validation accuracy for the cross-validated SVC classifier is :  0.98
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```


Train confusion Matrix for the cross-validated SVC classifier Model

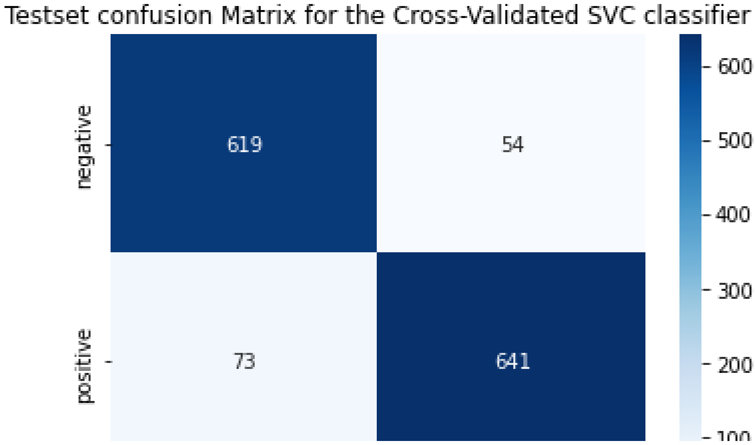
```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```


Validation confusion Matrix for the cross-validated SVC classifier Model

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

```
The testset accuracy score for the Cross-Validated SVC classifier model is :  0.9084354
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("SVM-Classifier with Cross Validation")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

Decision Trees

```
from sklearn.tree import DecisionTreeClassifier

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

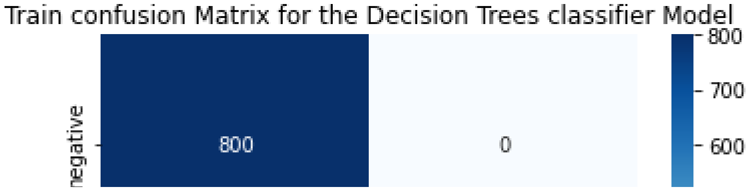
```
     DecisionTreeClassifier()
```

```
predicted_train = clf.predict(X_train)
predicted_val = clf.predict(X_val)
```

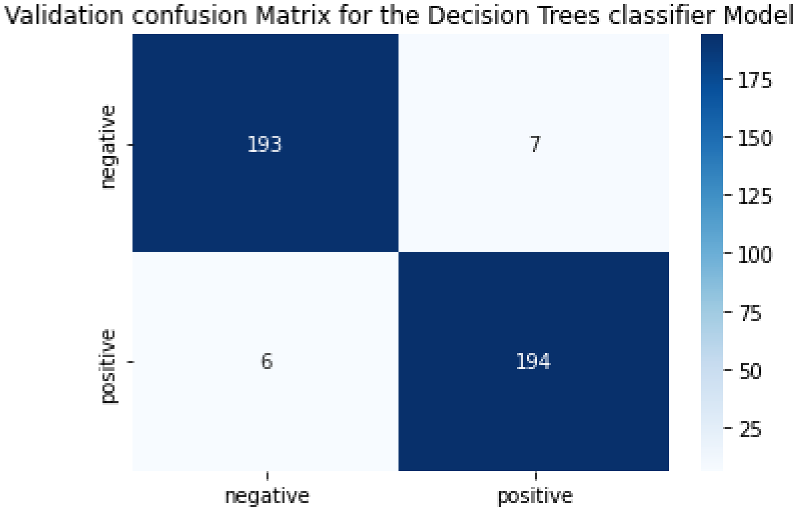
```
print("The Train accuracy for the Decision Trees classifier Model is : ",accuracy_score(y_tra
print("The Validation accuracy for the Decision Trees classifier Model is : ",accuracy_score(
```

```
     The Train accuracy for the Decision Trees classifier Model is :  1.0
     The Validation accuracy for the Decision Trees classifier Model is :  0.9675
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



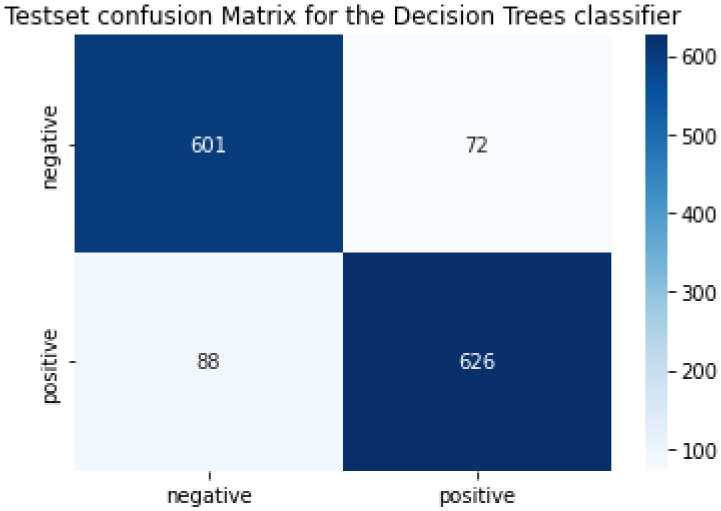
```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



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

```
The testset accuracy score for the Decision Trees classifier Model is :  0.884643114635
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("Decision Trees Classifier")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

Ensemble Models

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

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

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

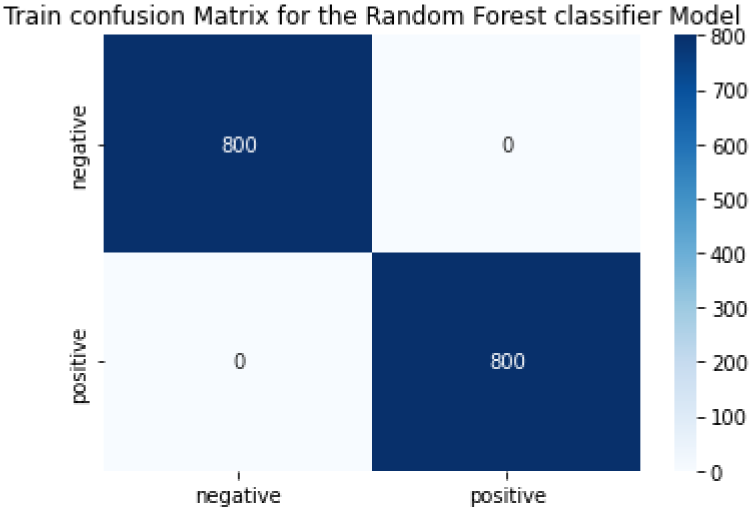
```
     RandomForestClassifier(n_estimators=300)
```

```
predicted_train = clf.predict(X_train)
predicted_val = clf.predict(X_val)
```

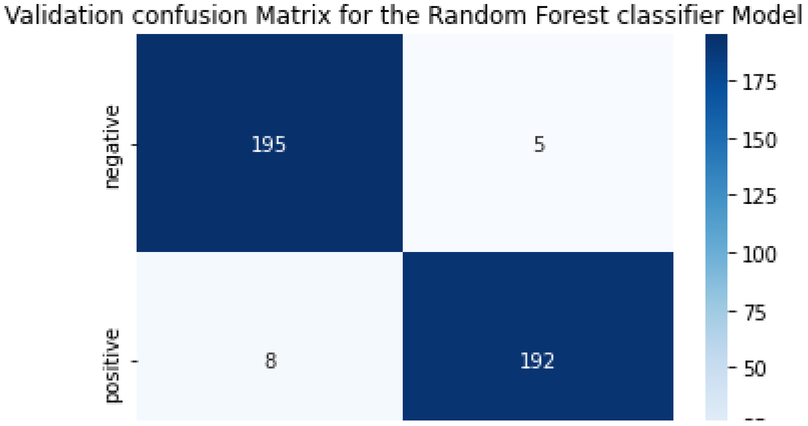
```
print("The Train accuracy for the Random Forest classifier Model is : ",accuracy_score(y_trai
print("The Validation accuracy for the Random Forest classifier Model is : ",accuracy_score(y_
```

```
     The Train accuracy for the Random Forest classifier Model is :  1.0
     The Validation accuracy for the Random Forest classifier Model is :  0.9675
```

```
plot_confusion_matrix(confusion_matrix(y_train,predicted_train),"Train confusion Matrix for t
```



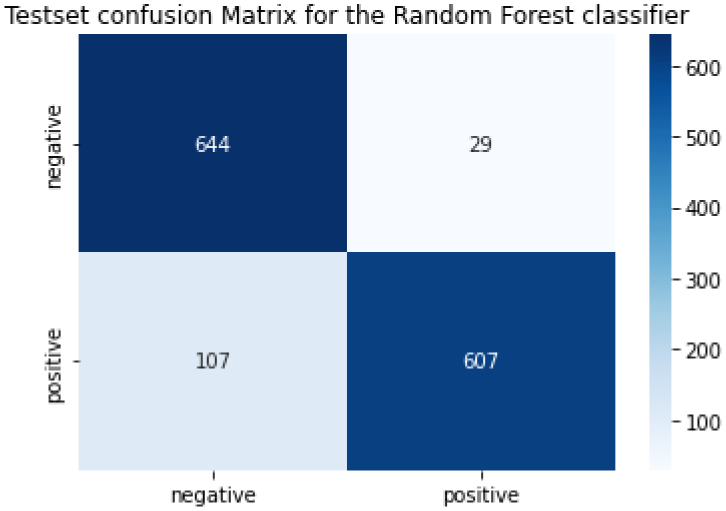
```
plot_confusion_matrix(confusion_matrix(y_val,predicted_val),"Validation confusion Matrix for
```



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

```
     The testset accuracy for the Random Forest classifier model is :  0.9019466474405191
```

```
plot_confusion_matrix(confusion_matrix(actual,testset_predicted),"Testset confusion Matrix fo
```



```
model_name.append("Random Forest Classifier")
train_accuracies.append(np.round(accuracy_score(y_train,predicted_train),4))
val_accuracies.append(np.round(accuracy_score(y_val,predicted_val),4))
test_accuracies.append(np.round(accuracy_score(actual,testset_predicted),4))
```

Accuracy

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Model", "Train Accuracy Score", "Validation Accuracy Score", "Test Accuracy

for i in range(len(model_name)):
```

```
for i in range(len(model_name)):
    x.add_row([model_name[i],train_accuracies[i],val_accuracies[i],test_accuracies[i]])


print(x)
```

```
+-----------------------------------------------+----------------------+----------------
|                     Model                     | Train Accuracy Score | Validation Accu
+-----------------------------------------------+----------------------+----------------
|             Multinomial Naive Bayes           |         0.99         |            0.98
| Multinomial Naive Bayes with Cross Validation |        0.9938        |           0.985
|               Logistic Regression             |        0.985         |            0.96
|   Logistic Regression with Cross Validation   |         1.0          |           0.985
|                  KNN-Classifier               |        0.8575        |           0.737
|      KNN-Classifier with Cross-Validation     |        0.8975        |           0.872
|                  SVM-Classifier               |         1.0          |           0.987
|      SVM-Classifier with Cross Validation     |        0.9994        |            0.98
|            Decision Trees Classifier          |         1.0          |           0.967
|             Random Forest Classifier          |         1.0          |           0.967
+-----------------------------------------------+----------------------+----------------
```

From above we can observe that Logistic Regression with Cross Validation model has the highest accuracy to detect the emotion of the text. We have we have integrated the concepts of sentiment analysis and text classification in this project.