Hate-Speech Classification

Python version :- `3.6.9`

How to Run :-

1. Install tweet preprocessor and wordninja using these two commands

```
pip install tweet-preprocessor
```

```
pip install wordninja
```

2. Install nltk libraries

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

3. Change hardcoded path of the training file and test file.

```
df = pd.read_csv('/content/data/1fe720be-90e4-4e06-9b52-9de93e0ea937_train.csv')
tdf = pd.read_csv('/content/final_test_q1/fcac6286-6db1-4577-ad80-612fb9d36db9_test.csv')
```

Platform :- Google Colab

Pre-processing :- The pre-processings are involved as follows-

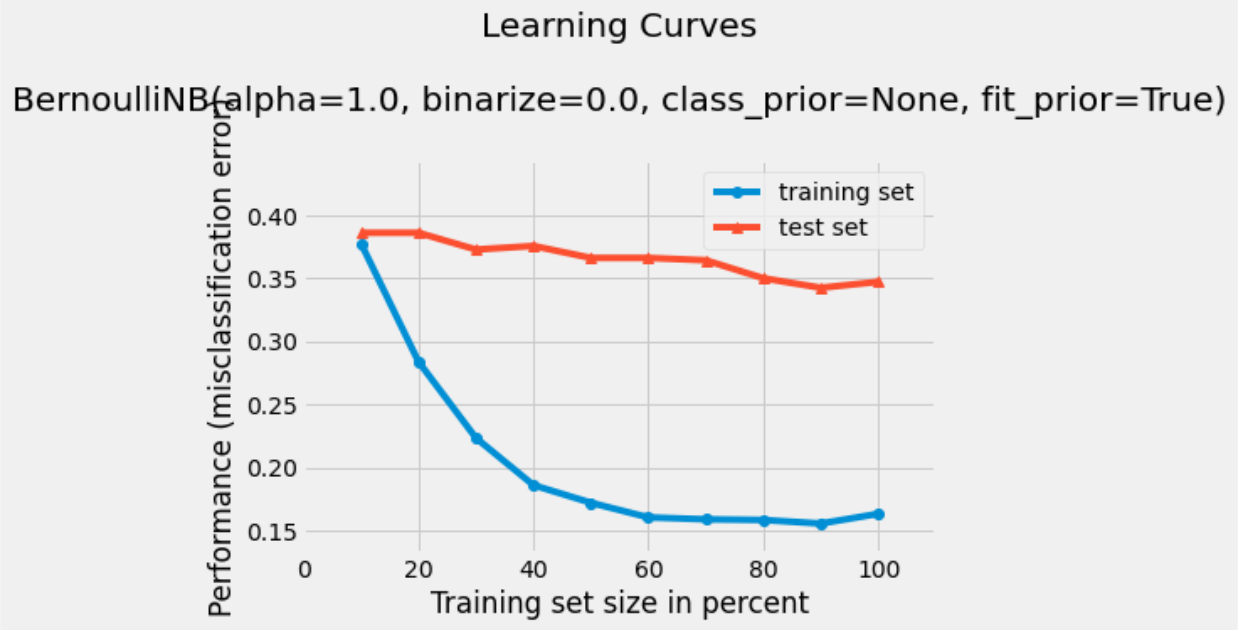
1. Converted to lower case
2. Removed URLs.
3. Removed Emojis.
4. Removed user mentions (e.g.- @username).
5. Removed digits.
6. Removed # from hashtags.
7. Removed (“, \, ’).
8. Replaced multiple spaces with single space.
9. Removed stop word based on nltk library.
10. Removed single character.
11. Removed punctuations.
12. Lemmatized the texts using “WordNetLemmatizer”

13. Stemmed the texts using “PorterStemmer”

Analysis and Model Selection and Observations :-

1. **BernoulliNB(Default Parameters):-**

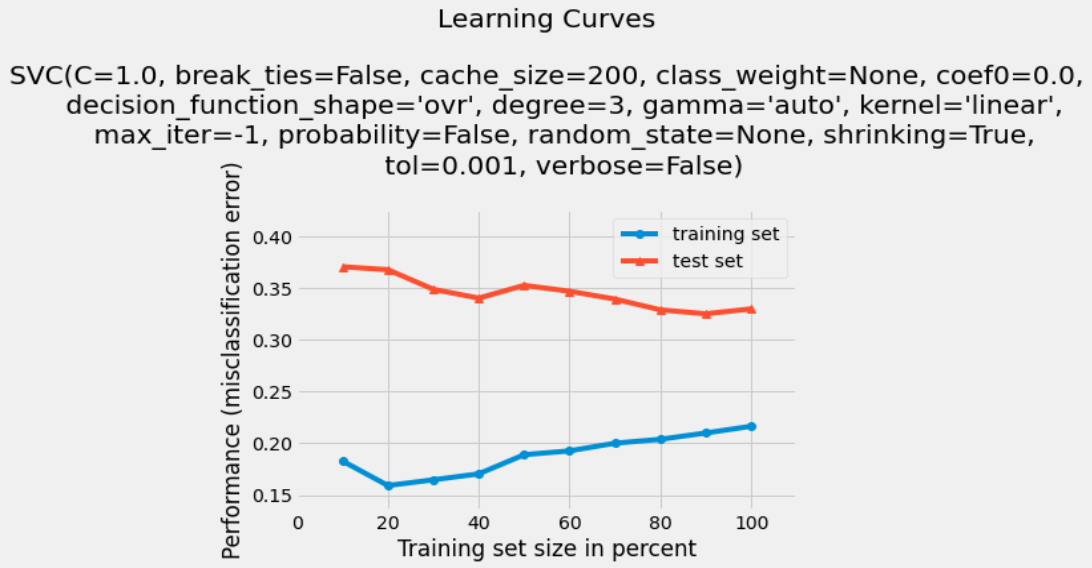
```
nb_cls = Pipeline([('vect', TfidfVectorizer()),('clf', BernoulliNB())])
nb_model = nb_cls.fit(x_train.text,y_train.values.ravel())
```



From the above learning curve it has been noticed that with the increasing of number of training sets the difference between training error and testing error has increased which shows that there is variance problem. This Is not a good model.

2. **Linear SVC(C=1.0, kernel='linear', degree=3, gamma='auto')**

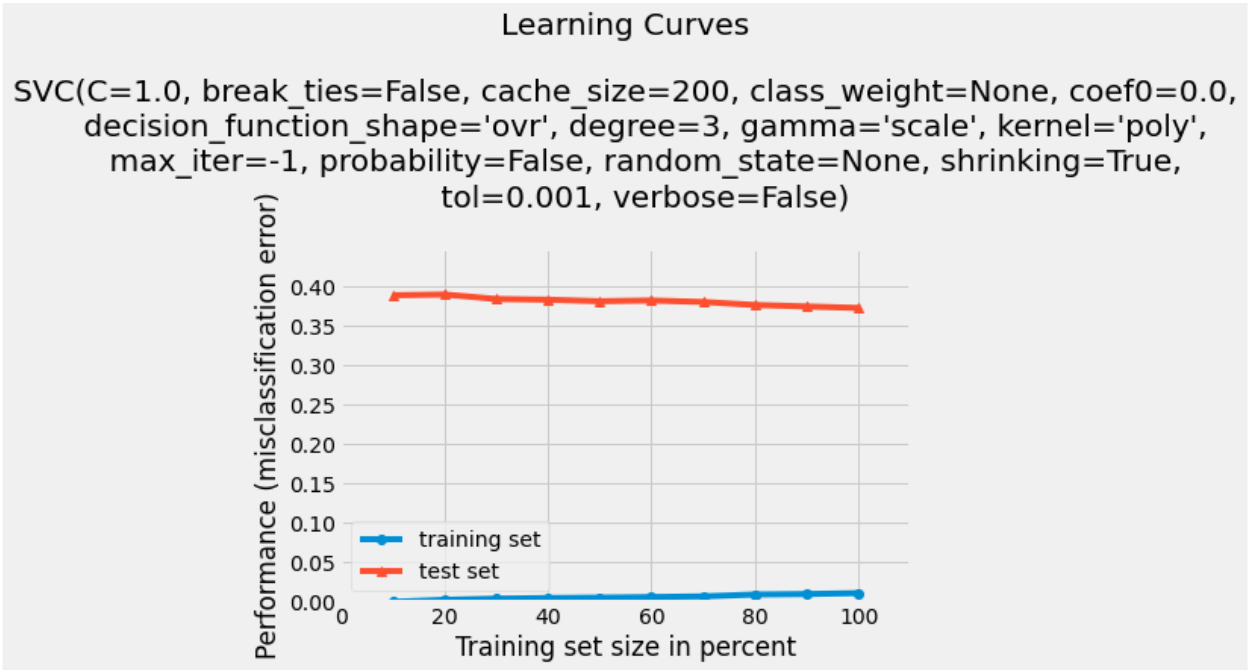
```
svm_linear = Pipeline([('vect', TfidfVectorizer(analyzer= 'char',ngr
am_range=(1,3),max_features=500000)),
                       ('clf', SVC(C=1.0, kernel='linear', degree=3,ga
mma='auto')),
])
linear_model = svm_linear.fit(x_train.text,y_train.values.ravel())
```



The Support Vector Classification with linear kernel proves to be a good model as with the increasing of number of training examples the error difference between training set and testing set has been reduced. This model is under consideration.

3. **Support Vector Classification with Polynomial Kernel(kernel = 'poly') :-**

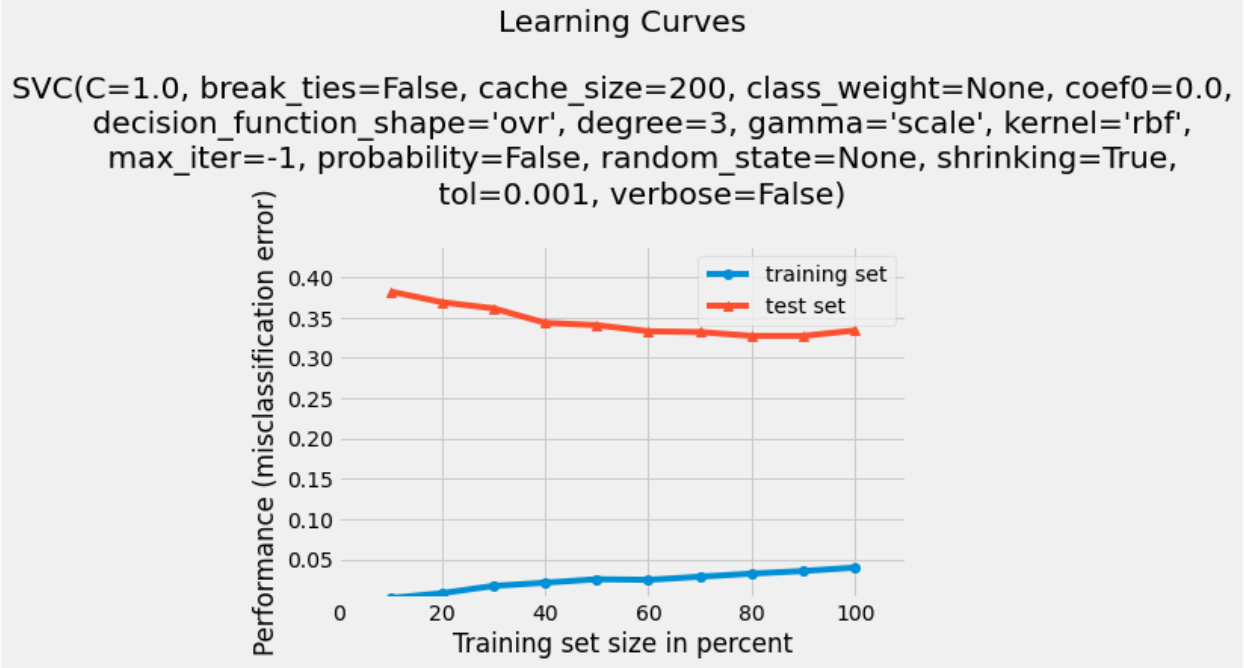
```
svm_poly = Pipeline([('vect', TfidfVectorizer()),
                     ('clf', SVC(kernel='poly')),
])
poly_model = svm_poly.fit(x_train.text,y_train.values.ravel())
```



In Support Vector Classification with Polynomail kernel the error difference between training error and testing error reducing at slower ratew with the increasing of the number of samples.

4. **Support Vector Classification with RBF kernel (kernel = 'rbf', C=1.0) :-**

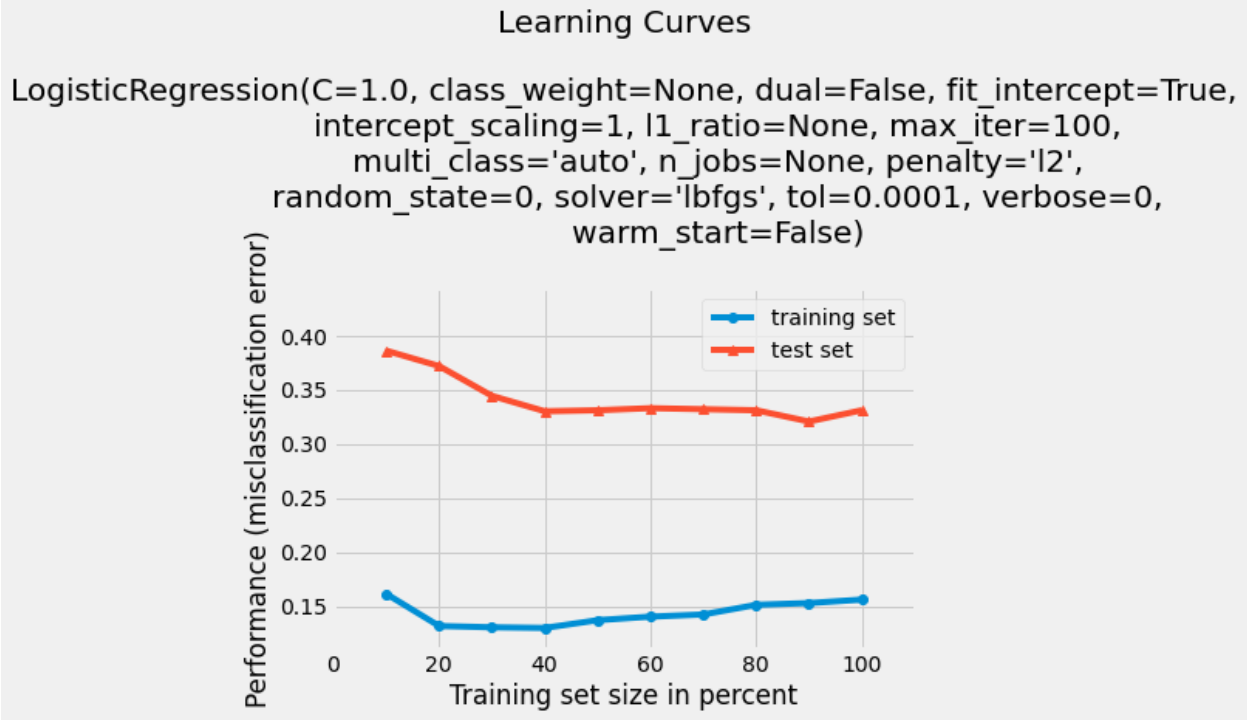
```
svm_rbf = Pipeline([('vect', TfidfVectorizer()),
                    ('clf', SVC(kernel='rbf',C=1.0)),
])
rbf_model = svm_rbf.fit(x_train.text,y_train.values.ravel())
```



With support vector classficiation using rbf kernel it has been noticed that the difference between traning error and testing error is reducing with the increasing of number of samples. Which indicates that the variance problem is reducing. This model is under consideration.

5. **Logistic Regression (random_state = 0) :-**

```
logreg = Pipeline([('vect', TfidfVectorizer(analyzer= 'char',ngram_r
ange=(1,20),max_features=500000, norm='l2')),
                   ('clf', LogisticRegression(random_state=0)),
                  ])
logreg.fit(x_train.text, y_train.values.ravel())
```



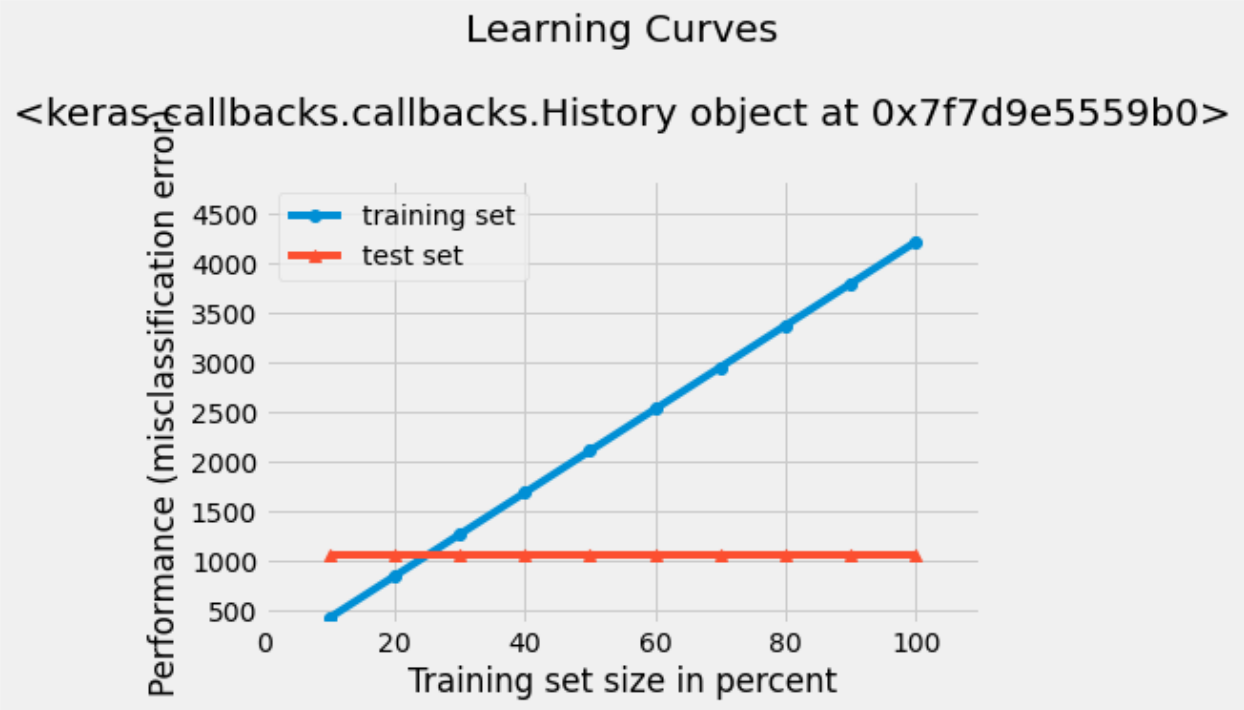
In logisitic regression with th eincreasing of th number of samples the difference between the traning error and tesing erro ris reducing. The variance problem is decreasing. Here has been noticed that `TfidfVectorizer()` parameters play a significant role in achieving more accuracy. The analyser is used as character and ngram_range best suited is (1,20) with max features 500000. This model is under consideration.

6. **Multilayer perceptron :-**

```
nn1 = Sequential()
nn1.add(Dense(10, activation='relu', input_dim = x_train_nn.shape[1]
))
nn1.add(Dense(1,activation='sigmoid'))
nn1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['
accuracy'])
nn1.fit(x_train_nn, y_train, epochs=10, verbose=1)
```

The two layers MLP is used with. The first layer has 10 nodes and the second layers has single node which indicates if it is hate speech or not. Binary Cross entropy loss function is used. And metric is “accuracy”.

```
Epoch 1/10
4212/4212 [==============================] - 1s 222us/step - loss: 0.6766 - accuracy: 0.6097
Epoch 2/10
4212/4212 [==============================] - 1s 156us/step - loss: 0.6339 - accuracy: 0.6239
Epoch 3/10
4212/4212 [==============================] - 1s 153us/step - loss: 0.5686 - accuracy: 0.7305
Epoch 4/10
4212/4212 [==============================] - 1s 157us/step - loss: 0.4955 - accuracy: 0.8274
Epoch 5/10
4212/4212 [==============================] - 1s 154us/step - loss: 0.4277 - accuracy: 0.8621
Epoch 6/10
4212/4212 [==============================] - 1s 155us/step - loss: 0.3709 - accuracy: 0.8896
Epoch 7/10
4212/4212 [==============================] - 1s 156us/step - loss: 0.3221 - accuracy: 0.9091
Epoch 8/10
4212/4212 [==============================] - 1s 153us/step - loss: 0.2820 - accuracy: 0.9224
Epoch 9/10
4212/4212 [==============================] - 1s 162us/step - loss: 0.2486 - accuracy: 0.9330
Epoch 10/10
4212/4212 [==============================] - 1s 152us/step - loss: 0.2204 - accuracy: 0.9461
<keras.callbacks.callbacks.History at 0x7f7d9d71f160>
```

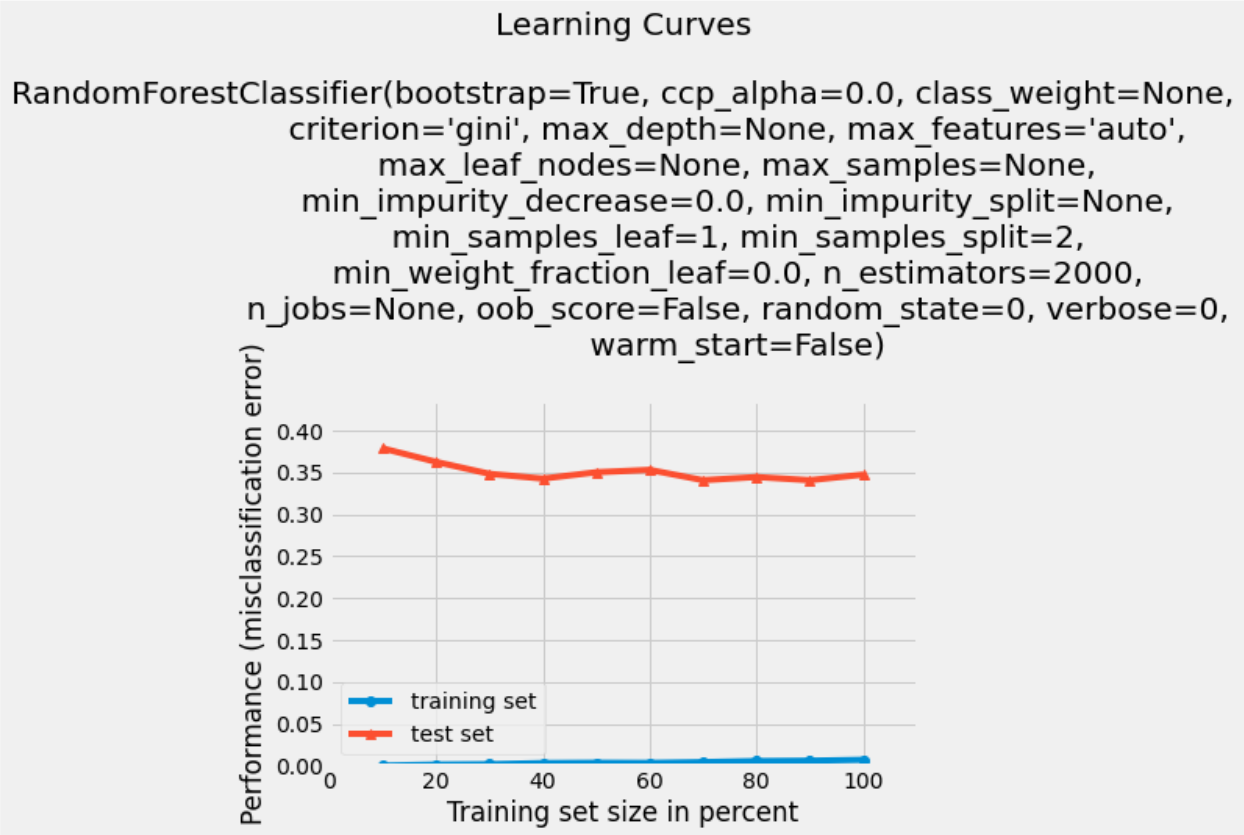


It has been noticed that the MLP works very poorly with initially bias problem and later with the variance problem. This model is not good.

7. **Random Forest (n_estimators=2000, random_state=0) :-**

```
vect = TfidfVectorizer(analyzer= 'char',ngram_range=(1,20),max_features=500000)
x_train_rf = vect.fit_transform(x_train.text.ravel())
x_test_rf = vect.transform(x_test.text.ravel())
classifier = RandomForestClassifier(n_estimators=2000, random_state=0)
classifier.fit(x_train_rf, y_train.values.ravel())
```

The number trees used is 2000. Here has been noticed that `TfidfVectorizer()` parameters play a significant role in achieving more accuracy. The analyser is used as character and ngram_range best suited is (1,20) with max features 500000. This model is under consideration.



From the graph it has been noticed that there is high variance problem in Random forest. It is not a good model to be considered.

Final Model :- After doing the experiments with these models now the considered models are tested with the test data provided for AiCrowd portal. And the best result is produced by the Support vector classification with RBF kernel.

```
svm_rbf = Pipeline([('vect', TfidfVectorizer()),
                    ('clf', SVC(kernel='rbf', C=1.0)),
])
rbf_model = svm_rbf.fit(x.text,y.values.ravel())
```