Instructions to run our code:

1. Libraries used: Pandas, numpy, nltk and sklearn
2. Installed libraries using “pip install [library name]” command

Step1: Data Pre-processing

In Data Pre-processing We did the following procedures:

- Blank spaces removal: We removed different special characters which are stored in “Separators” variable in the code. Additionally, we removed blank spaces (“__END_PARAGRAPH__, __END_ESSAY__”) as well as stop words.
- Converted some float data into string format for easy evaluation
- Lowercase conversion of all strings in list
- After that lemmatisation (a technique for text normalization) done on the tokens and stored it into variables

Step 2: Token Vectorization of corpus and Feature Extraction using TF-IDF method

Step 3: Parameter tuning for SVM machine:

- For finding best parameters for SVM we defined a method called “finding_best_params” as shown below:

```
# following function was used to tunning the parameters of SVM machine
def finding_best_params(X_train,y_train,X_test,y_test):
    param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid','linear']}
    grid = GridSearchCV(svm.SVC(),param_grid,refit=True,verbose=2,cv=2)
    grid.fit(X_train,y_train)
    print(grid.best_estimator_)
    grid_predictions = grid.predict(X_test)
    print(confusion_matrix(y_test,grid_predictions))
    print(classification_report(y_test,grid_predictions))
```

- Using this step this function derived the best parameters which we pass into SVM model definition.
- We used GridSearchCV function from scikitlearn for parameter tuning

Step 4: Implementation of models

- Based on derived results we finalized that SVM provides the better predictions with TF-IDF as compared top Naïve Byes.
- We used both Naïve byes as well as SVM from which we decided to use SVM model as it gives more accurate predictions.

Step 5: After executions of all functions the predictions will be stored at “prediction.csv” file.