```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn import model_selection, naive_bayes
        from sklearn.svm import SVC
        from sklearn.model_selection import GridSearchCV
        from sklearn.metrics import accuracy_score, f1_score
        from tqdm import tqdm
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import MaxAbsScaler
        from sklearn.metrics import roc_auc_score
        import os.path
        import pickle
```

Reading test and train data from already preprocessed pickle file

```
In [2]: X_train = pd.read_pickle('../../../Preprocessing/Data/X_train.pkl')
        X_test = pd.read_pickle('../../../Preprocessing/Data/X_test.pkl')
        y_train = pd.read_pickle('../../../Preprocessing/Data/y_train.pkl')
        y_test = pd.read_pickle('../../../Preprocessing/Data/y_test.pkl')
```

Performing TF-IDF over the dataset for word embedding

```
In [3]: vec = TfidfVectorizer(ngram_range=(1, 2), min_df=3,
                              max_df=0.9, strip_accents='unicode', use_idf=1, sm
        ooth_idf=1, sublinear_tf=1)
        df = pd.read_pickle('../../../Preprocessing/Data/preprocess_pickle.pkl')
        df['comment_text_final'] = [" ".join(text) for text in df['comment_text_
        final'].values]
        vec = vec.fit(df['comment_text_final'])
```

```
In [4]: train_term_doc = vec.transform(X_train)
        test_term_doc = vec.transform(X_test)
```

Scaling the input data using MaxAbsScaler

```
In [5]: scaler = MaxAbsScaler()
        train_term_doc = scaler.fit_transform(train_term_doc)
        test_term_doc = scaler.fit_transform(test_term_doc)
```

Multilabel Classification using Binary Relevance

Following function performs Multinomial Naive Bayes for each label. In short, it uses Binary Relevance (BR) method for multi-label classification.

In [6]:
```
def perform_NB_for_label(label):
    naive_classifier = naive_bayes.MultinomialNB()
    naive_classifier.fit(train_term_doc, y_train[label])
    predictions_NB = naive_classifier.predict(test_term_doc)
    print(label + " Accuracy Score - " + str(accuracy_score(y_test[label
], predictions_NB)))
    print(label + " F1 Score - " + str(f1_score(y_test[label], predictio
ns_NB)))
    print(label + " ROC-AUC Score - " + str(roc_auc_score(y_test[label],
predictions_NB)) + '\n')
    return predictions_NB
```

Accuracy, F1 Score and ROC-AUC Score for each label

In [7]:
```
label_cols = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'i
dentity_hate']
for label in label_cols:
    perform_NB_for_label(label)
```

```
toxic Accuracy Score - 0.8849890635332418
toxic F1 Score - 0.7758056519583539
toxic ROC-AUC Score - 0.8514251977017934

severe_toxic Accuracy Score - 0.9701409023856757
severe_toxic F1 Score - 0.21837549933422104
severe_toxic ROC-AUC Score - 0.5736055287365285

obscene Accuracy Score - 0.915204232158299
obscene F1 Score - 0.6842204963061186
obscene ROC-AUC Score - 0.8029749306670559

threat Accuracy Score - 0.9897756752632382
threat F1 Score - 0.10666666666666666
threat ROC-AUC Score - 0.5415572527244907

insult Accuracy Score - 0.9009105244417315
insult F1 Score - 0.6066235864297254
insult ROC-AUC Score - 0.7621225253583902

identity_hate Accuracy Score - 0.9741594180782339
identity_hate F1 Score - 0.15333333333333332
identity_hate ROC-AUC Score - 0.5485804496307919
```

Multilabel Classification using Classifier Chain

Following function performs Multinomial Naive Bayes for each label and it uses Classifier Chain method for multi-label classification.

```
In [57]: classifier = ClassifierChain(classifier = naive_bayes.MultinomialNB())
         classifier.fit(train_term_doc, y_train)

Out[57]: ClassifierChain(classifier=MultinomialNB(alpha=1.0, class_prior=None,
                                                  fit_prior=True),
                         order=None, require_dense=[True, True])

In [58]: predictions = classifier.predict(test_term_doc)

In [59]: print(" Accuracy Score - " + str(accuracy_score(y_test, predictions)))

          Accuracy Score - 0.7560404903606649
```