Toxic Comment Classification:

R.Lakshmi Priya(IMT2015037)
G.Dharmik.(IMT2015016)
Mounica Sidde.(IMT2016119)

Problem Statement:

In this competition, you're challenged to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful.

Solution Format:

We need to create a model(logistic regression) which predicts a probability of each type of toxicity for each comment.

Approach:

Importing all the needed.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.metrics import log_loss,confusion_matrix,classification_report,roc_curve,auc

from scipy import sparse
import pickle
```

We first start our analysis by digging into the given datasets.

1.Loading the datasets.

```
training_data = pd.read_csv('train.csv')
testing_data = pd.read_csv('test.csv')
```

2.Checking the Shapes of the datasets.
3.Check and filling NA.

4.Storing the target in a variable.

```
target_col = ['toxic', 'severe_toxic', 'obscene', 'threat','insult', 'identity_hate']
y = training_data[target_col]
del train_vec, test_vec, train_vec_ch, test_vec_ch
```

5.Splitting the train and test datasets.

```
train_vec =V_word.fit_transform(training_data['comment_text'])
test_vec = V_word.transform(testing_data['comment_text'])
train_vec_ch = V_char.fit_transform(training_data['comment_text'])
test_vec_ch = V_char.transform(testing_data['comment_text'])
```

6.Text to feature vectors(TfidfVectorizer).

```
V_word = TfidfVectorizer(max_features=20000, lowercase=True, analyzer='word',
                         stop_words= 'english',ngram_range=(1,3),dtype=np.float32)
V_char = TfidfVectorizer(max_features=40000, lowercase=True, analyzer='char',
                         stop_words= 'english',ngram_range=(3,6),dtype=np.float32)
```

7.Learn the vocabulary from the feature vectors.

```
X = sparse.hstack([train_vec, train_vec_ch])
X_test = sparse.hstack([test_vec, test_vec_ch])
```

8.Transfrom train data to document-term matrix.

9.Import and instantiate logistic regression model.

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.metrics import log_loss,confusion_matrix,classification_report,roc_curve,auc
```

Since the target variable is more than one, we will use loop and apply our model and then we can see the prediction.

```
prd = np.zeros((X_test.shape[0],y.shape[1]))
scores=[]
for i,col in enumerate(target_col):
    lr = LogisticRegression(C=2,random_state = i,class_weight = 'balanced')
    lr.fit(X,y[col])
    prd[:,i] = lr.predict_proba(X_test)[:,1]

    cv_score = np.mean(cross_val_score(
        lr, X, y[col], cv=3, scoring='roc_auc'))
    scores.append(cv_score)
    print('CV score for class {} is {}'.format(col, cv_score))
```

10.Submitting our prediction to Kaggle.

```
submit = pd.concat([testing_data['Id'],prd_1],axis=1)
submit.to_csv('toxic_lr.csv',index=False)
submit.head()
```

Conclusion:

-This model has performed very well with the Accuracy score 0.9889.

- This model performs very well for both "negative" and "positive" classes, even though the dataset is highly imbalanced with the majority of the Negative class.

Further Plan:

-Improve the prediction ability, in which the model can recognise the word context (sentiment analysis) so that it can evaluate the toxic content probability of sentences, not only by specific words.

References:

i.)https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

ii.)https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e