Contents

Comments classification

```
Goal: Develop machine learning model for user's feedback classification to
positive and negative comments. F1 score > 0.75 is required.
```

Data Preparation

Import necessary libraries and packages

In [1]:
```
import pandas as pd
import numpy as np
from nltk.stem import WordNetLemmatizer
import re
from sklearn.feature_extraction.text import CountVectorizer
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.linear_model import SGDClassifier
!pip install lightgbm
nltk.download('punkt')
nltk.download('wordnet')
import warnings
warnings.filterwarnings("ignore")
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Requirement already satisfied: lightgbm in c:\users\hp\anaconda3\lib\site-packa
ges (3.3.1)
Requirement already satisfied: numpy in c:\users\hp\anaconda3\lib\site-packages
(from lightgbm) (1.20.3)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\hp\anaconda3\li
b\site-packages (from lightgbm) (0.24.2)
Requirement already satisfied: wheel in c:\users\hp\anaconda3\lib\site-packages
(from lightgbm) (0.37.0)
Requirement already satisfied: scipy in c:\users\hp\anaconda3\lib\site-packages
(from lightgbm) (1.7.1)
Requirement already satisfied: joblib>=0.11 in c:\users\hp\anaconda3\lib\site-p
ackages (from scikit-learn!=0.22.0->lightgbm) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\li
b\site-packages (from scikit-learn!=0.22.0->lightgbm) (2.2.0)

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Data upload

In [3]:
```
df = pd.read_csv(r'C:\Users\HP\Downloads\toxic_comments.csv')
```

Data review

In [4]:
```
df.head()
```

Out[4]:

	text	toxic
0	Explanation\nWhy the edits made under my usern...	0
1	D'aww! He matches this background colour I'm s...	0
2	Hey man, I'm really not trying to edit war. It...	0
3	"\nMore\nI can't make any real suggestions on ...	0
4	You, sir, are my hero. Any chance you remember...	0

In [5]:
```
df.shape
```

Out[5]:
```
(159571, 2)
```

In [6]:
```
corpus = df['text']
```

In [7]:
```
type(corpus)
```

Out[7]:
```
pandas.core.series.Series
```

Create function for text cleaning.

In [8]:
```
def clear_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z_]', ' ', text)
    text = text.split()
    text = " ".join(text)
    return text
```

In [9]:
```
for x in range(len(corpus)):
    corpus[x] = clear_text(corpus[x])
```

In [10]:
```
corpus
```

Out[10]:
```
0         explanation why the edits made under my userna...
1         d aww he matches this background colour i m se...
2         hey man i m really not trying to edit war it s...
3         more i can t make any real suggestions on impr...
4         you sir are my hero any chance you remember wh...
                              ...
159566    and for the second time of asking when your vi...
159567    you should be ashamed of yourself that is a ho...
159568    spitzer umm theres no actual article for prost...
159569    and it looks like it was actually you who put ...
159570    and i really don t think you understand i came...
Name: text, Length: 159571, dtype: object
```

Tokenization and lemmatization

In [11]:
```
def lemm(text):
    lemmatizer = WordNetLemmatizer()
    word_list = nltk.word_tokenize(text)
    #corpus = [ lemmatize(i) for i in corpus]
    lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in word_list])

    return lemmatized_output
```

In [12]:
```
corpus_lemma = []
for el in corpus:
    corpus_lemma.append(lemm(el))
```

In [13]:
```
print("Original text:", corpus[1])
print("Lemmatized text:", corpus_lemma[1])
```

```
Original text: d aww he matches this background colour i m seemingly stuck with
thanks talk january utc
Lemmatized text: d aww he match this background colour i m seemingly stuck with
thanks talk january utc
```

Calculate tf_idf

Make data ready for training and testing

In [14]:
```
target = df['toxic']
X_train, X_test, y_train, y_test = train_test_split(corpus_lemma, target, test_si
tfidfvectorizer = TfidfVectorizer(analyzer='word' , stop_words='english',)
tfidfvectorizer.fit(X_train)
tfidf_train = tfidfvectorizer.transform(X_train)
tfidf_test = tfidfvectorizer.transform(X_test)
```

Model Training

```
_Logistic Regression model_
```

In [15]:
```
model = LogisticRegression()
```

In [16]:
```
model.fit(tfidf_train, y_train)
```

Out[16]:
```
LogisticRegression()
```

In [17]:
```
predictions = model.predict(tfidf_test)
```

In [18]:
```
predictions
```

Out[18]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [19]:
```
print("{:.2f}".format(f1_score(y_test, predictions)))
```

```
0.74
```

Logistic Regression did not show required f1-score.

*_SGDClassifier model_*

In [20]:
```
model_2 = SGDClassifier(max_iter=1000)
```

In [21]:
```
model_2.fit(tfidf_train, y_train)
```

Out[21]:
```
SGDClassifier()
```

In [22]:
```
predictions_SGD = model_2.predict(tfidf_test)
```

In [23]:
```
f1_score(y_test, predictions_SGD)
```

Out[23]:
```
0.6341263330598851
```

*_RandomForestClassifier_*

In [24]:
```
clf = RandomForestClassifier(random_state=42, n_jobs=-1)
```

In [25]:
```
clf.fit(tfidf_train, y_train)
```

Out[25]:
```
RandomForestClassifier(n_jobs=-1, random_state=42)
```

In [26]:
```
predictions_clf = clf.predict(tfidf_test)
```

In [27]:
```
f1_score(y_test, predictions_clf)
```

Out[27]:
```
0.7188208616780045
```

SGDClassifier and RandomForestClassifier with standard hyperparameters did now show required f1-score. Logistic Regression shows the best performance which can be improved using regularization.

In [28]:
```
model_log_reg_2 = LogisticRegression(penalty='l2', C=10, max_iter=1000, random_st
```

In [29]:
```
model_log_reg_2.fit(tfidf_train, y_train)
```

Out[29]:
```
LogisticRegression(C=10, max_iter=1000, n_jobs=-1, random_state=42)
```

In [30]:
```
predictions_log_reg_2 = model_log_reg_2.predict(tfidf_test)
```

In [31]:
```
f1_score(y_test, predictions_log_reg_2)
```

Out[31]:
```
0.7770177838577291
```

```
Add cross-validation for hyperparameters tuning
```

In [32]:
```
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] }
reg_with_cv = GridSearchCV(LogisticRegression(penalty='l2'), param_grid)
reg_with_cv
```

Out[32]:
```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]})
```

In [33]:
```
reg_with_cv.fit(tfidf_train, y_train)
```

Out[33]:
```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]})
```

In [34]:
```
reg_with_cv_predictions = reg_with_cv.predict(tfidf_test)
```

In [35]:
```
f1_score(y_test, reg_with_cv_predictions)
```

Out[35]:
```
0.7773610637572452
```

In [36]:
```
reg_with_cv.best_params_
```

Out[36]:
```
{'C': 10}
```

Conclusion

```
The text was prepared, the models were trained and evaluated with F1-score
metric. The best result was shown by Logistic Regression with ridge
regularization and regularization coefficient C=10.
```