```
!pip install PyDrive
```

```
Collecting PyDrive
  Downloading https://files.pythonhosted.org/packages/52/e0/0e64788e5dd58ce2d6934549676243dc69d982f198524be9b99e9c
    |████████████████████████████████| 993kB 3.5MB/s
Requirement already satisfied: google-api-python-client>=1.2 in /usr/local/lib/python3.6/dist-packages (from PyDri
Requirement already satisfied: oauth2client>=4.0.0 in /usr/local/lib/python3.6/dist-packages (from PyDrive) (4.1.3
Requirement already satisfied: PyYAML>=3.0 in /usr/local/lib/python3.6/dist-packages (from PyDrive) (3.13)
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from google-api
Requirement already satisfied: httplib2<1dev,>=0.9.2 in /usr/local/lib/python3.6/dist-packages (from google-api-py
Requirement already satisfied: six<2dev,>=1.6.1 in /usr/local/lib/python3.6/dist-packages (from google-api-python-
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.6/dist-packages (from oauth2client>=4.0.0->PyD
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.6/dist-packages (from oauth2client>
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.6/dist-packages (from oauth2client>=4.0.0->
Building wheels for collected packages: PyDrive
  Building wheel for PyDrive (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/fa/d2/9a/d3b6b506c2da98289e5d417215ce34b696db856643bad779f4
Successfully built PyDrive
Installing collected packages: PyDrive
Successfully installed PyDrive-1.3.1
```

```
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

```
%cd "gdrive/My Drive/project"
```

```
[Errno 2] No such file or directory: 'gdrive/My Drive/project'
```

```
!ls
!python
```

```
data             embeddings.h5  n_words_clean.eps  rnn
data_loader.py   mltools        __pycache__        wiki.en.vec
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download('stopwords')
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
>>> nltk.download('wordnet')
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
>>> exit()
```

Creating model and training model

LightGBM

```
import gc
import pandas as pd

from scipy.sparse import csr_matrix, hstack

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel

from sklearn.linear_model import LogisticRegression
import lightgbm as lgb
```

```
class_names = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']

train, valid = data_loader.load_train_data('data/train.csv')
test = data_loader.load_test_data('data/test.csv','data/test_labels.csv').fillna('')
train = train.fillna('')
valid = valid.fillna('')
test = test.fillna('')
print('Loaded')

train_text = train['comment_text']
valid_text = valid['comment_text']
test_text = test['comment_text']
all_text = pd.concat([train_text, valid_text, test_text])

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='word',
    token_pattern=r'\w{1,}',
    ngram_range=(1, 2),
    max_features=50000)

word_vectorizer.fit(all_text)

train_word_features = word_vectorizer.transform(train_text)
valid_word_features = word_vectorizer.transform(valid_text)
test_word_features = word_vectorizer.transform(test_text)

char_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='char',
    stop_words='english',
    ngram_range=(2, 6),
    max_features=50000)

char_vectorizer.fit(all_text)

train_char_features = char_vectorizer.transform(train_text)
valid_char_features = char_vectorizer.transform(valid_text)
test_char_features = char_vectorizer.transform(test_text)


train_features = hstack([train_char_features, train_word_features])
valid_features = hstack([valid_char_features, valid_word_features])
test_features = hstack([test_char_features, test_word_features])


submission = pd.DataFrame.from_dict({'id': test['id']})

train.drop('comment_text', axis=1, inplace=True)
```

```
Loaded
```

```python
import gc

del train_text
del test_text
del all_text
del train_char_features
del test_char_features
del train_word_features
del test_word_features
gc.collect()
```

```
7
```

```python
for class_name in class_names:
    print(class_name)
    train_target = train[class_name]
    valid_target = valid[class_name]

    model = LogisticRegression(solver='sag')
    sfm = SelectFromModel(model, threshold=0.2)

    train_sparse_matrix = sfm.fit_transform(train_features, train_target)
    valid_sparse_matrix = sfm.fit_transform(valid_features, valid_target)
    y_train = train_target
    y_valid = valid_target

    test_sparse_matrix = sfm.transform(test_features)

    d_train = lgb.Dataset(train_sparse_matrix, label=y_train)
    d_valid = lgb.Dataset(valid_sparse_matrix, label=y_valid)

    watchlist = [d_train, d_valid]
    params = {'learning_rate': 0.2,
              'application': 'binary',
              'num_leaves': 31,
              'verbosity': -1,
              'metric': 'auc',
              'data_random_seed': 2,
              'bagging_fraction': 0.8,
              'feature_fraction': 0.6,
              'nthread': 4,
              'lambda_l1': 1,
              'lambda_l2': 1,
             'is_training_metric': True}
    rounds_lookup = {'toxic': 140,
                  'severe_toxic': 50,
                  'obscene': 80,
                  'threat': 80,
                  'insult': 70,
                  'identity_hate': 80}
    model = lgb.train(params,
                      train_set=d_train,
                      num_boost_round=rounds_lookup[class_name],
                      valid_sets=watchlist,
                      early_stopping_rounds=5,
                      verbose_eval=10)
    submission[class_name] = model.predict(test_sparse_matrix)

submission.to_csv('lgb_submission.csv', index=False)
```

```
toxic
Training until validation scores don't improve for 5 rounds.
[10]	training's auc: 0.950639	valid_1's auc: 0.548253
Early stopping, best iteration is:
[10]	training's auc: 0.950639	valid_1's auc: 0.548253
severe_toxic
Training until validation scores don't improve for 5 rounds.
Early stopping, best iteration is:
[3]	training's auc: 0.958062	valid_1's auc: 0.530767
obscene
Training until validation scores don't improve for 5 rounds.
[10]	training's auc: 0.989021	valid_1's auc: 0.544203
Early stopping, best iteration is:
[5]	training's auc: 0.983558	valid_1's auc: 0.552226
threat
Training until validation scores don't improve for 5 rounds.
[10]	training's auc: 0.994965	valid_1's auc: 0.952918
Early stopping, best iteration is:
[11]	training's auc: 0.995318	valid_1's auc: 0.954279
insult
Training until validation scores don't improve for 5 rounds.
Early stopping, best iteration is:
[3]	training's auc: 0.956957	valid_1's auc: 0.569067
identity_hate
Training until validation scores don't improve for 5 rounds.
[10]	training's auc: 0.982558	valid_1's auc: 0.675261
[20]	training's auc: 0.989855	valid_1's auc: 0.782583
Early stopping, best iteration is:
[24]	training's auc: 0.992414	valid_1's auc: 0.803023
```

```
### LGBM score

from sklearn.metrics import roc_auc_score
lgbm_preds = pd.read_csv("lgbm/lgb_submission.csv")
test = data_loader.load_test_data('data/test.csv','data/test_labels.csv').fillna('')
class_names = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
roc_auc_scores_test = 0
for class_name in class_names:
    score = roc_auc_score(test[class_name], lgbm_preds[class_name])
    roc_auc_scores_test += score
    print(score)
print("ROC AUC Test score:", roc_auc_scores_test/6)
```

```
0.5244897910556687
0.5451923876377867
0.5449464102031194
0.8208912527145443
0.5430031400626973
0.755299354482252
ROC AUC Test score: 0.6223037226926781
```