Data Glacier

Your Deep Learning Partner

Exploratory Data Analysis

Hate Speech Detection Using Transformers

Team – Solitude Ensemble

27-Feb-2024

Agenda

Data Glacier

Your Deep Learning Partner

Executive Summary

- As a part of final internship project, we want to create a machine learning model which can classify the tweets in two parts whether it is hateful tweet or not.

Problem Statement

- Objective : The term hate speech is understood as any type of verbal, written or behavioural communication that attacks or uses derogatory or discriminatory language against a person or group based on what they are, in other words, based on their religion, ethnicity, nationality, race, colour, ancestry, sex or another identity factor. In this problem, We will take you through a hate speech detection model with Machine Learning and Python. But, before that we are going to do analysis of data and based on analysis we will modify for better performance of our model.

Approach

- Data Understanding
- Data Cleaning and manipulation
- Analysing and visualizing data
- Recommendations of machine learning models

Data Exploration

- 2 datasets: Training and Test
- 3 features in total in training data with 2 input features and 1 target
- 2 features in total in test data with 2 input features
- Total Data:

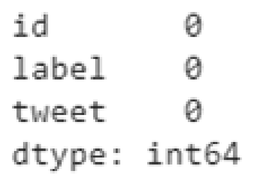
1) training dataset : 31962

2) test dataset : 17197

Data Analysis

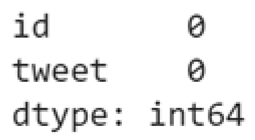
Finding Empty data in dataset

- in Training Data



```
id        0
label     0
tweet     0
dtype: int64
```

- in Test Data



```
id        0
tweet     0
dtype: int64
```

- So, here as you can see there is no empty data in our both datasets. So, we do not need to fill up that space.

Data Modification

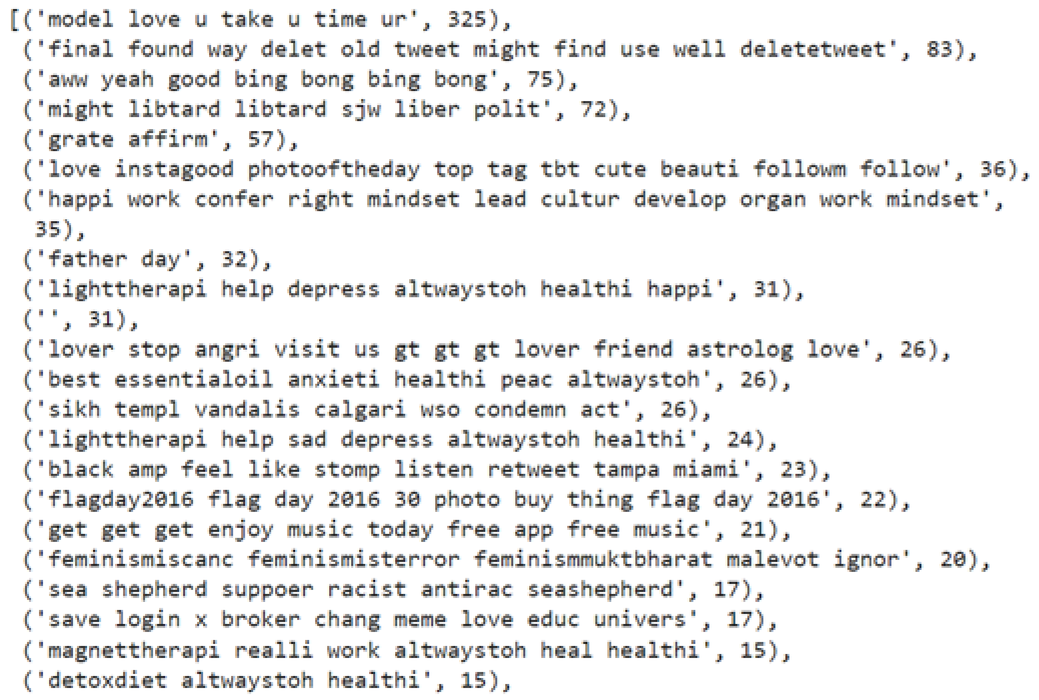
Removing unnecessary words

- In our datasets we have some unnecessary words such as user names , special characters , website links or numbers which do not have any impact in predictions. So, we need to remove that to create better and faster machine learning model.

Data Modification

Removing Duplicate Data (I)

- First we are going to find that whether we have any duplicate data(tweets) or not. And, in below picture we can see tweet and occurrence of that tweet in whole data.

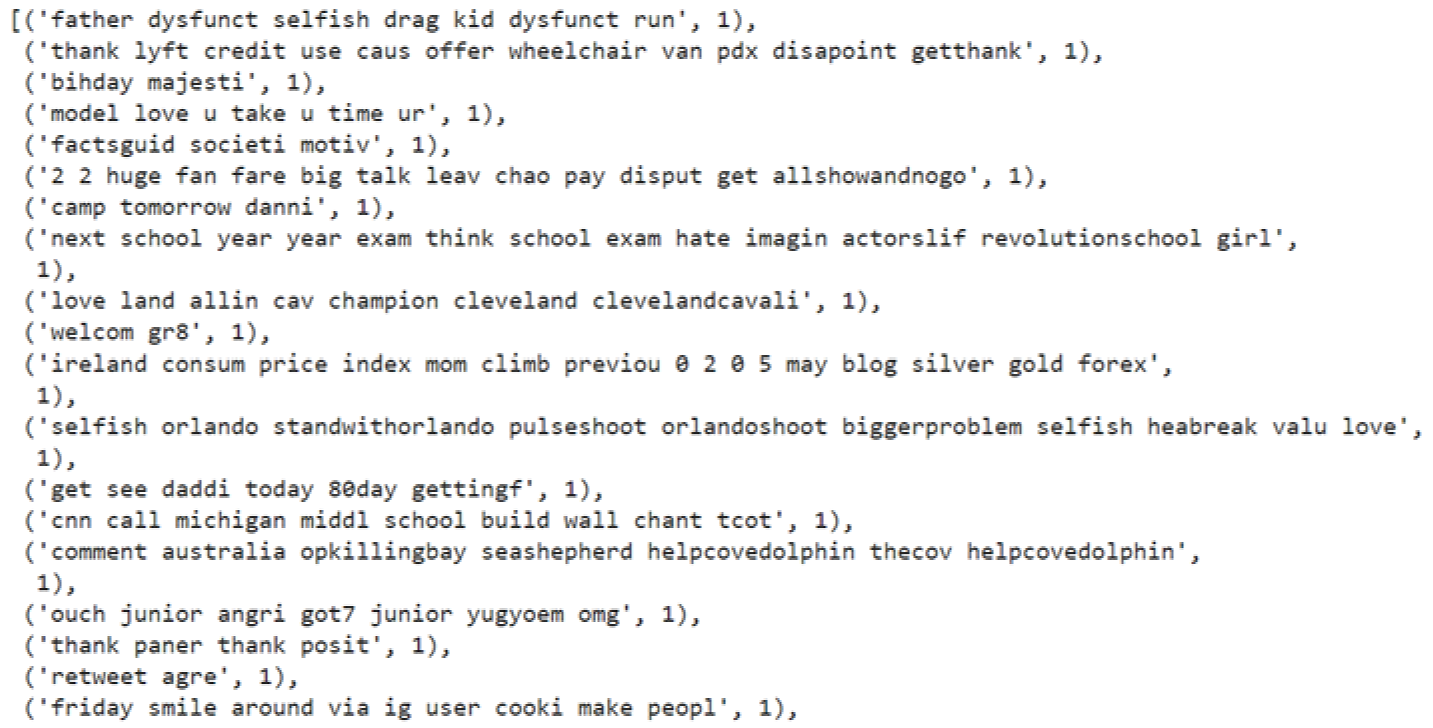


```
[('model love u take u time ur', 325),
 ('final found way delet old tweet might find use well deletetweet', 83),
 ('aww yeah good bing bong bing bong', 75),
 ('might libtard libtard sjw liber polit', 72),
 ('grate affirm', 57),
 ('love instagood photooftheday top tag tbt cute beauti followm follow', 36),
 ('happi work confer right mindset lead cultur develop organ work mindset',
  35),
 ('father day', 32),
 ('lighttherapi help depress altwaystoh healthi happi', 31),
 ('', 31),
 ('lover stop angri visit us gt gt gt lover friend astrolog love', 26),
 ('best essentialoil anxieti healthi peac altwaystoh', 26),
 ('sikh templ vandalis calgari wso condemn act', 26),
 ('lighttherapi help sad depress altwaystoh healthi', 24),
 ('black amp feel like stomp listen retweet tampa miami', 23),
 ('flagday2016 flag day 2016 30 photo buy thing flag day 2016', 22),
 ('get get get enjoy music today free app free music', 21),
 ('feminismiscanc feminismisterror feminismmuktbharat malevot ignor', 20),
 ('sea shepherd suppoer racist antirac seashepherd', 17),
 ('save login x broker chang meme love educ univers', 17),
 ('magnettherapi realli work altwaystoh heal healthi', 15),
 ('detoxdiet altwaystoh healthi', 15),
```

Data Modification

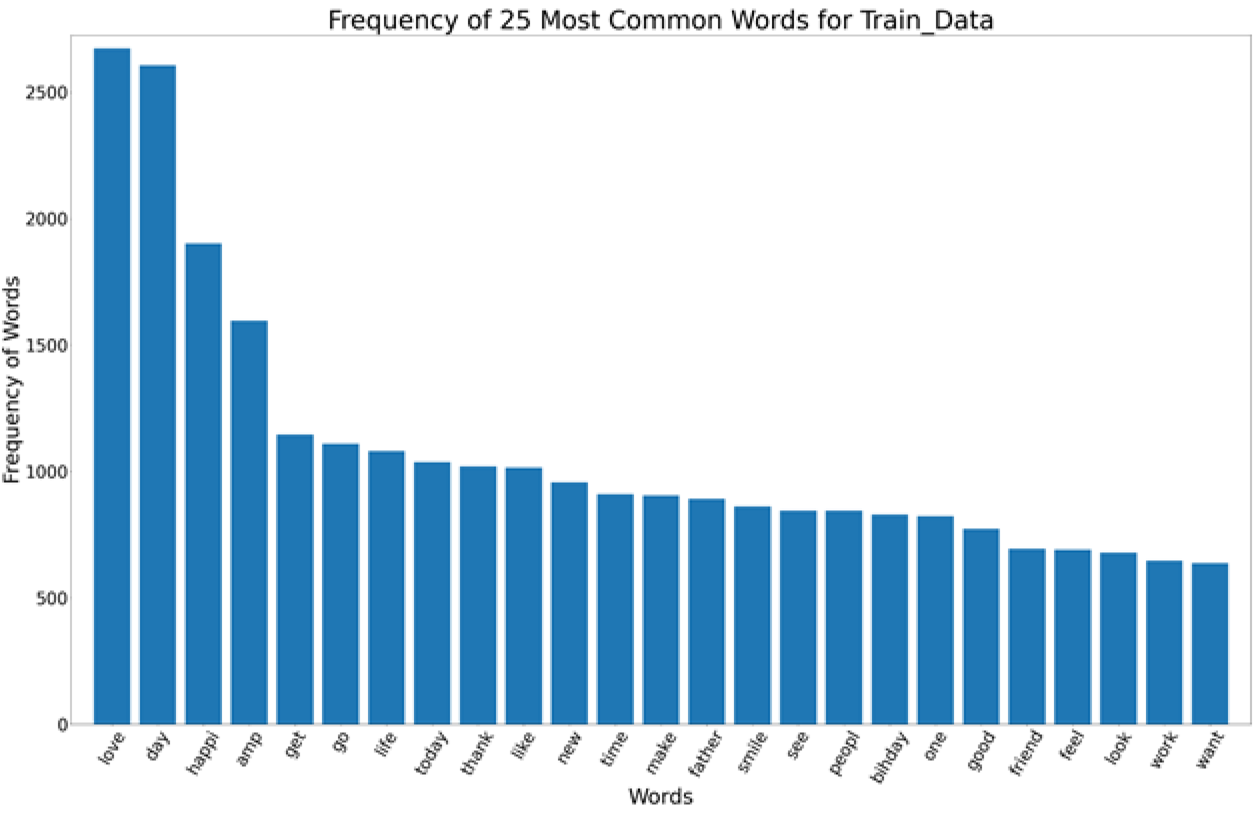
Removing Duplicate Data (II)

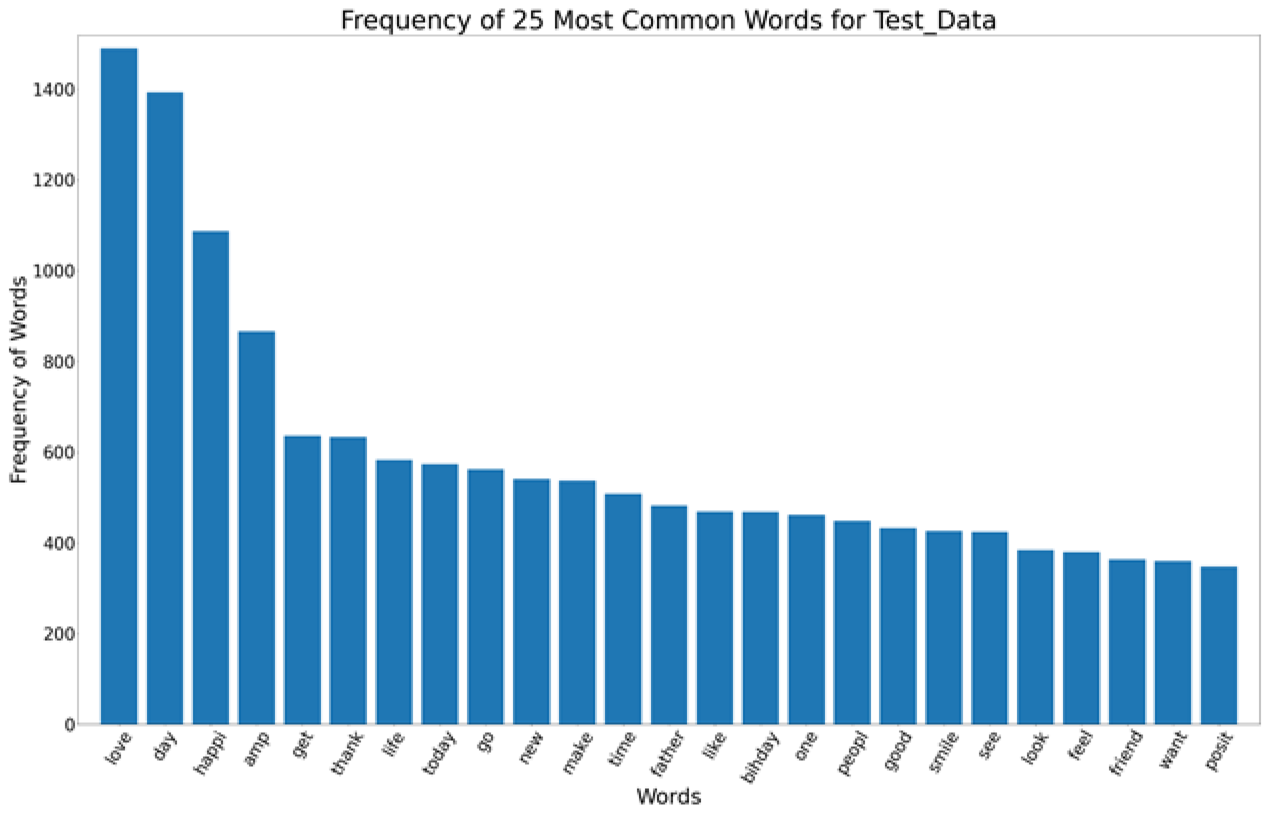
- Now , you can see below the result after removing duplicate data(tweets). Every tweet has only 1 occurrence in dataset.



```
[('father dysfunct selfish drag kid dysfunct run', 1),
 ('thank lyft credit use caus offer wheelchair van pdx disapoint getthank', 1),
 ('bihday majesti', 1),
 ('model love u take u time ur', 1),
 ('factsguid societi motiv', 1),
 ('2 2 huge fan fare big talk leav chao pay disput get allshowandnogo', 1),
 ('camp tomorrow danni', 1),
 ('next school year year exam think school exam hate imagin actorslif revolutionschool girl',
  1),
 ('love land allin cav champion cleveland clevelandcavali', 1),
 ('welcom gr8', 1),
 ('ireland consum price index mom climb previou 0 2 0 5 may blog silver gold forex',
  1),
 ('selfish orlando standwithorlando pulseshoot orlandoshoot biggerproblem selfish heabreak valu love',
  1),
 ('get see daddi today 80day gettingf', 1),
 ('cnn call michigan middl school build wall chant tcot', 1),
 ('comment australia opkillingbay seashepherd helpcovedolphin thecov helpcovedolphin',
  1),
 ('ouch junior angri got7 junior yugyoem omg', 1),
 ('thank paner thank posit', 1),
 ('retweet agre', 1),
 ('friday smile around via ig user cooki make peopl', 1),
```

Frequency of word in data

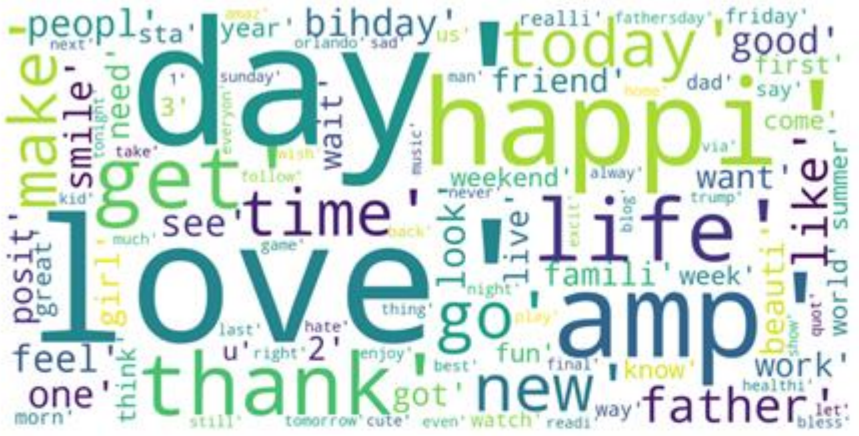




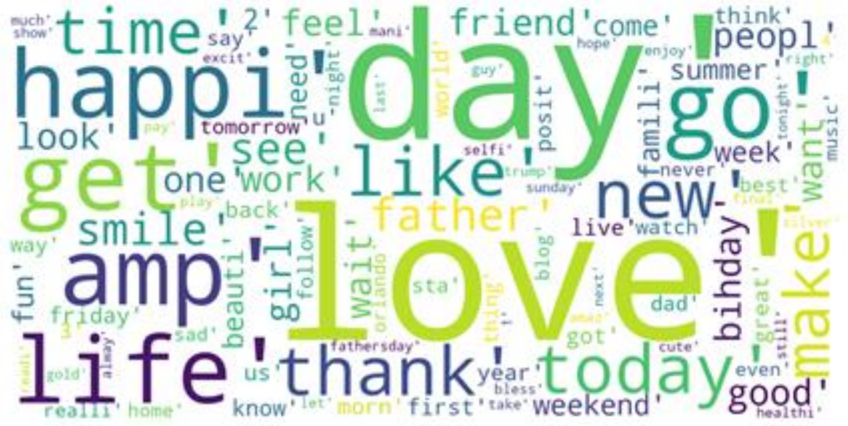
- In above graph we can see the 25 most common words in Training dataset and Test Dataset.
- We can see that both the datasets have almost same most common words. For this reason we might get accurately trained model through which we can get high accuracy while testing and predicting results.

Frequency of word in data

Top 100 Most Common Words for Test_Data



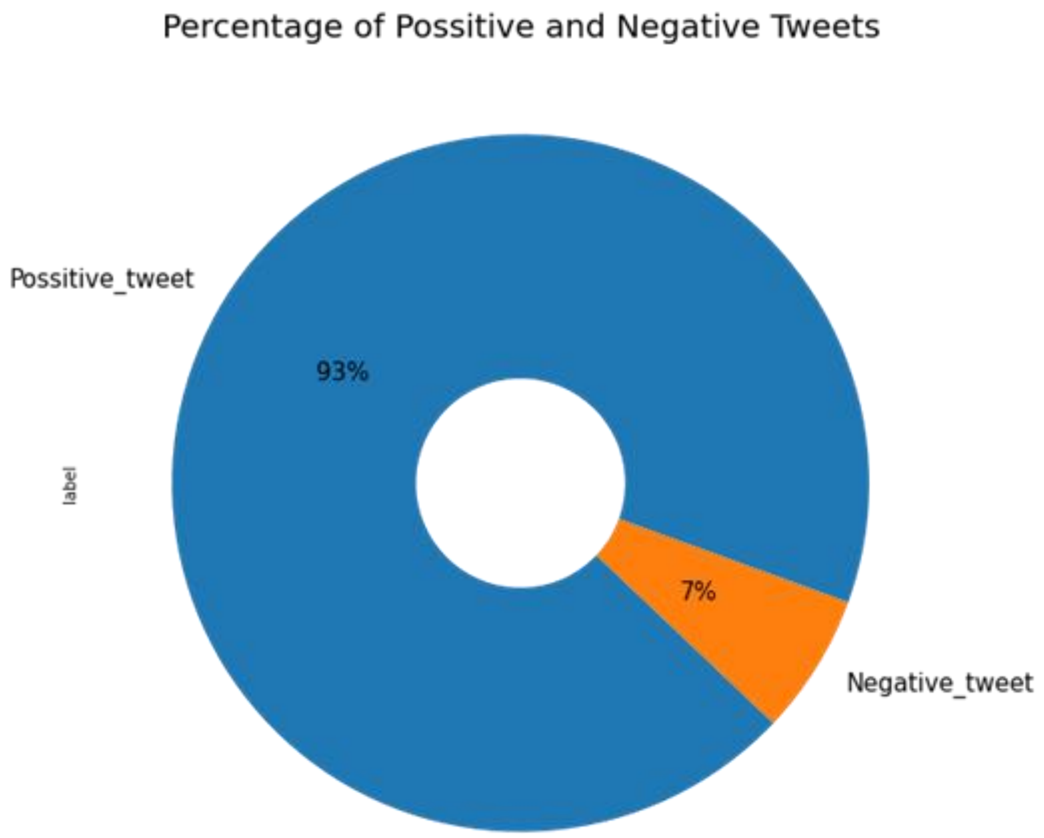
Top 100 Most Common Words for Train_Data



- Here we can 100 most common words in picture. Bigger the word most common the word.

Data Exploration and visualization

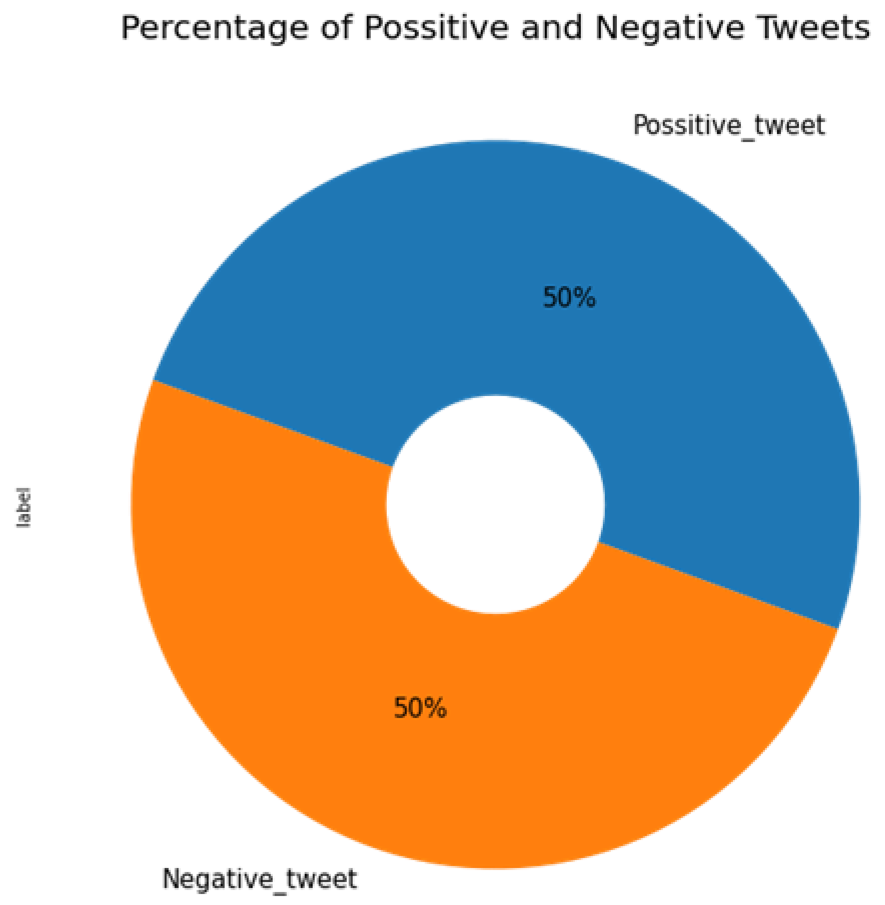
- **Distribution ratio of positive and negative tweets in training data**



- From above chart we can see that we have extremely high number of positive(Non hateful) tweets. This can cause problem in prediction of our test data. Model trained with this dataset will be heavily biased to positive tweets means in most of the cases our model will predict that is is positive tweet even if it is not. So, we have make it 1:1 ratio with sampling method.
- In sampling there is upsampling method in which it will generate data from our old dataset for e.g Here we have less negative tweets so it will create negative tweets to match number of positive tweets.

Data Exploration and visualization

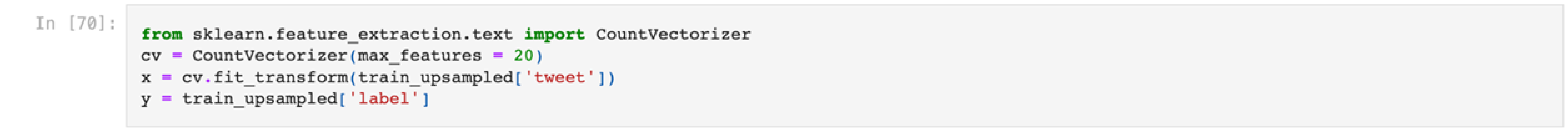
- **Distribution ratio of positive and negative tweets in training data**



- After upsampling this is the result where we have equal number of positive and negative tweets.

Code Walkthrough

Step_1: Feature_Extraction:



```
In [70]:  from sklearn.feature_extraction.text import CountVectorizer
          cv = CountVectorizer(max_features = 20)
          x = cv.fit_transform(train_upsampled['tweet'])
          y = train_upsampled['label']
```

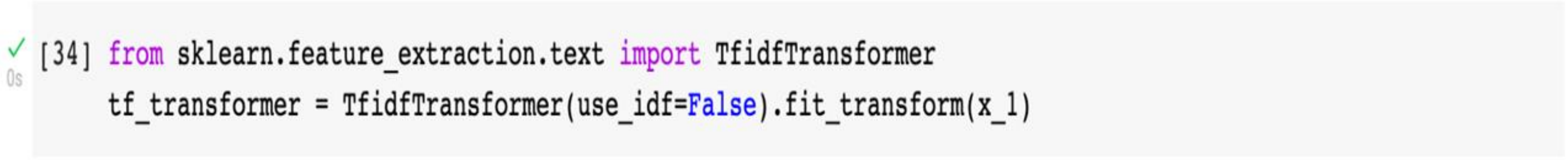
First of all we need to convert text into 0s and 1s. For that we need to do feature extraction and for that we can use Bag of Word model by CountVectorizer, TFIDF, etc…

Here, we used CountVectorizer to count the frequency of word and convert text into 0s and 1s.

Code Walkthrough

Step_2:Transformer:

Transformer use to Transform a count matrix to a normalized tf or tf-idf representation so that we can use TfidfTransformer from sklearn.feature_extraction as shown in belove code.



```
✓  [34] from sklearn.feature_extraction.text import TfidfTransformer
0s      tf_transformer = TfidfTransformer(use_idf=False).fit_transform(x_1)
```

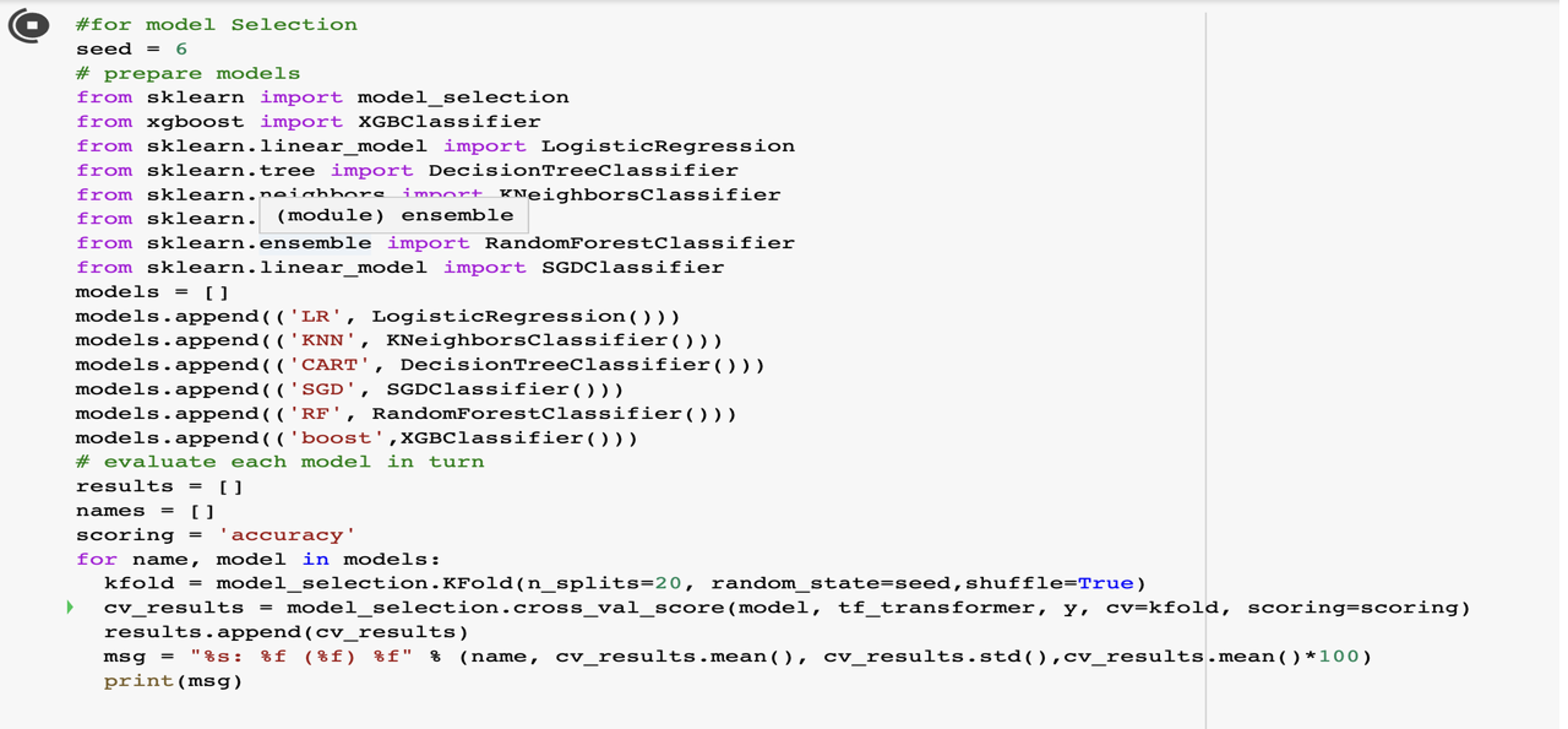
Code Walkthrough

Step 3: Model Selection:

From analysing datasets we can clearly see that we have to **classify** that whether tweet is positive or negative. So, we have to use modern and advance classification machine learning models such as XGboost, Stochastic Gradient descent , Decision Tree , Random Forest Model etc.

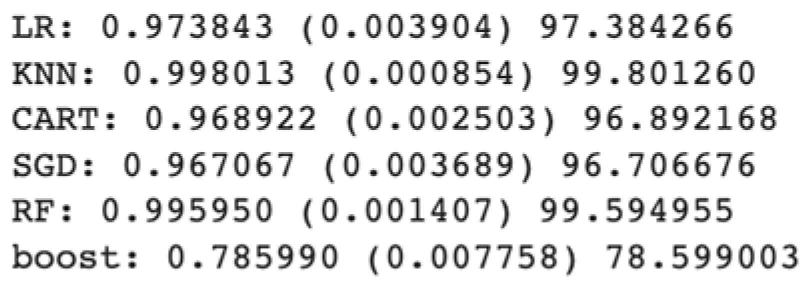
But the Question is which classification model is more accurate in this dataset?
So, for that we developed model to determine the accuracy of all classification model.By that, we can say which model is more accurate in this data set.

Code Walkthrough



```
#for model Selection
seed = 6
# prepare models
from sklearn import model_selection
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn. (module) ensemble
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
models = []
models.append(('LR', LogisticRegression()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('SGD', SGDClassifier()))
models.append(('RF', RandomForestClassifier()))
models.append(('boost',XGBClassifier()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
  kfold = model_selection.KFold(n_splits=20, random_state=seed,shuffle=True)
  cv_results = model_selection.cross_val_score(model, tf_transformer, y, cv=kfold, scoring=scoring)
  results.append(cv_results)
  msg = "%s: %f (%f) %f" % (name, cv_results.mean(), cv_results.std(),cv_results.mean()*100)
  print(msg)
```

Code Walkthrough



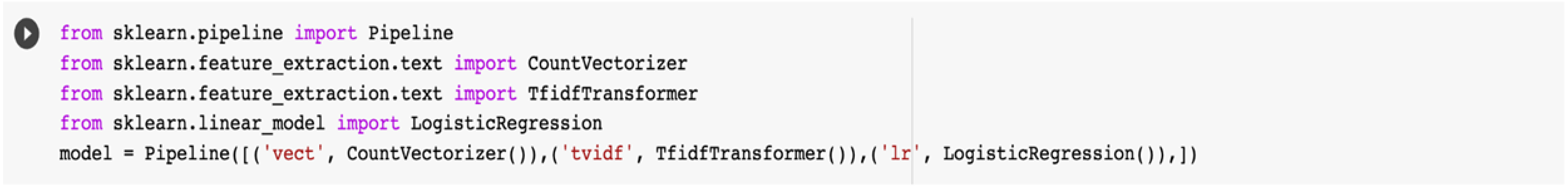
```
LR: 0.973843 (0.003904) 97.384266
KNN: 0.998013 (0.000854) 99.801260
CART: 0.968922 (0.002503) 96.892168
SGD: 0.967067 (0.003689) 96.706676
RF: 0.995950 (0.001407) 99.594955
boost: 0.785990 (0.007758) 78.599003
```

From the output of above model we can clearly say which model is more accurate in this data set.
In XGboost model there is a very less accuracy as compare to other so we can’t select that model.while,other models have almost same accuracy but according to time taken by each model we can select Logistic Regression and SGD.But, SGD is optimized model and that model is more accurate on Large data set.in contradiction we don’t have larger data set so according to Data_size, time and accuracy Logistic Regression model is more suitable.

Code Walkthrough

<u>Step 4: Model Building and Applying on Test data:</u>

Using Sklearn Pipeline is a convenient way to enforce the steps with preprocessing steps and ensures the code's reproducibility. Unstable and inconsistent results for production models can significantly impact businesses if they are relying on machine learning models to make their decisions every day.Hence,we used pipeline in our code.

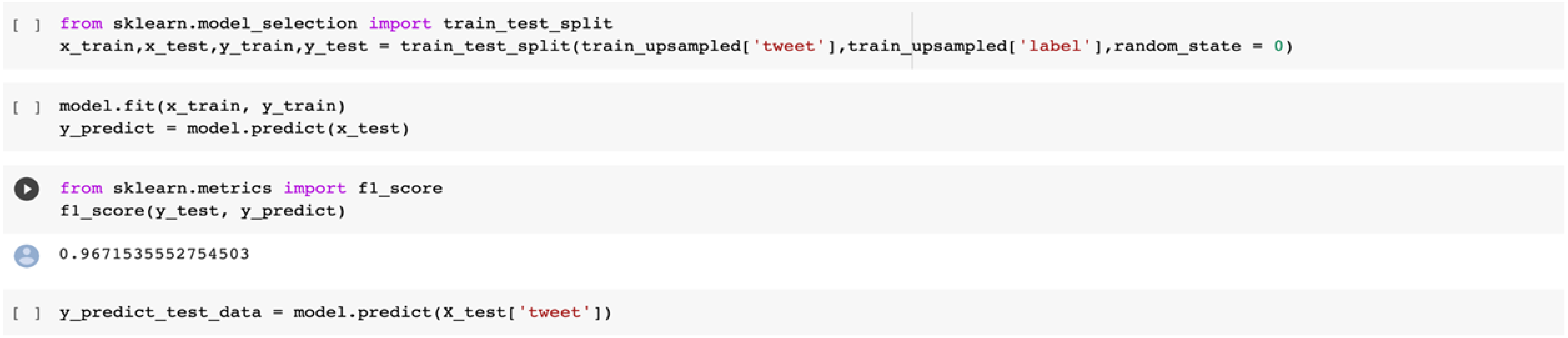


```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import LogisticRegression
model = Pipeline([('vect', CountVectorizer()),('tvidf', TfidfTransformer()),('lr', LogisticRegression()),])
```

Code Walkthrough

Step 4: Model Building and Applying on Test data:

After that, We trained our model using train_data and applied it into Test_data and got almost 97% accuracy.



```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test = train_test_split(train_upsampled['tweet'],train_upsampled['label'],random_state = 0)
```

```
[ ] model.fit(x_train, y_train)
    y_predict = model.predict(x_test)
```

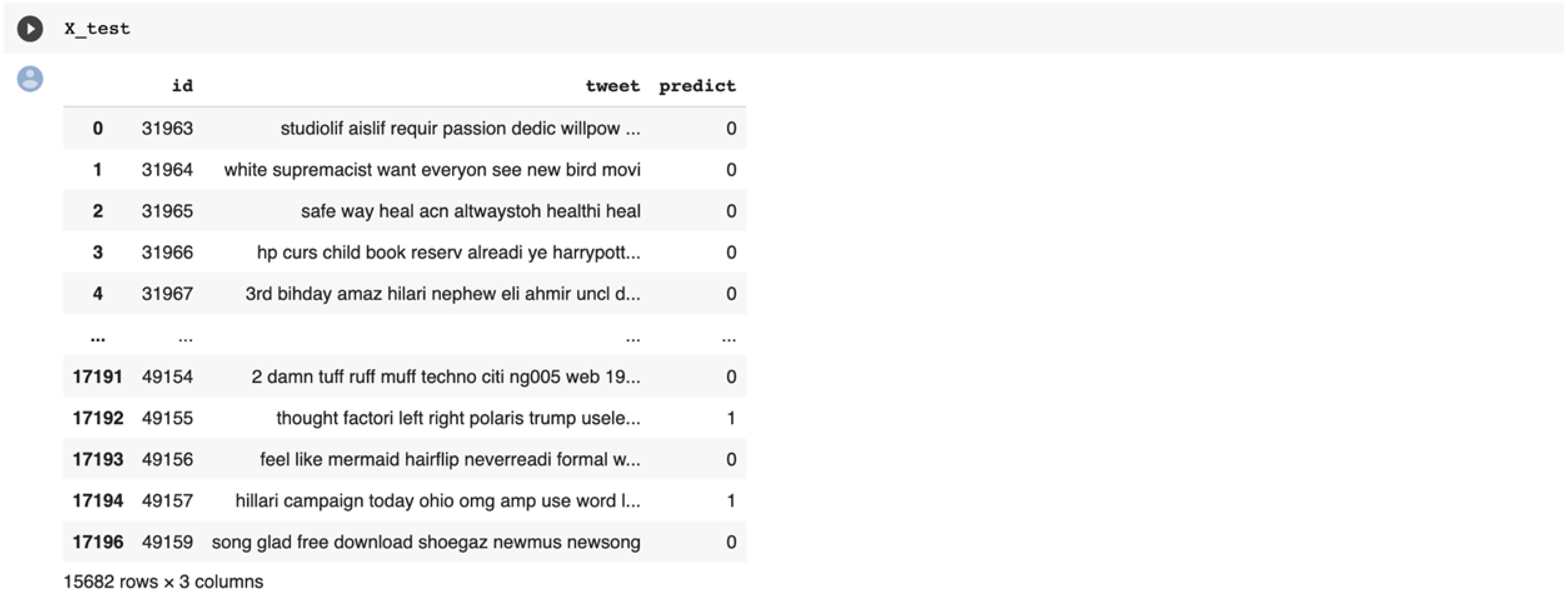
```
    from sklearn.metrics import f1_score
    f1_score(y_test, y_predict)

    0.9671535552754503
```

```
[ ] y_predict_test_data = model.predict(X_test['tweet'])
```

Code Walkthrough

Step 5: Prediction



```
X_test
```

```
              id                                           tweet  predict
0          31963     studiolif aislif requir passion dedic willpow ...        0
1          31964  white supremacist want everyon see new bird movi        0
2          31965          safe way heal acn altwaystoh healthi heal        0
3          31966     hp curs child book reserv alreadi ye harrypott...        0
4          31967     3rd bihday amaz hilari nephew eli ahmir uncl d...        0
...          ...                                               ...      ...
17191      49154     2 damn tuff ruff muff techno citi ng005 web 19...        0
17192      49155      thought factori left right polaris trump usele...        1
17193      49156     feel like mermaid hairflip neverreadi formal w...        0
17194      49157     hillari campaign today ohio omg amp use word l...        1
17196      49159  song glad free download shoegaz newmus newsong        0

15682 rows × 3 columns
```

Thank You

Data Glacier

Your Deep Learning Partner