Airline Sentiment Analysis

Natural language processing

شريف أشرف أحمد رشدي

عبدالرحمن محمد خليل

محمد حسني مسعد

يحيى زكريا ابراهيم

صبيح صلاح صبيح

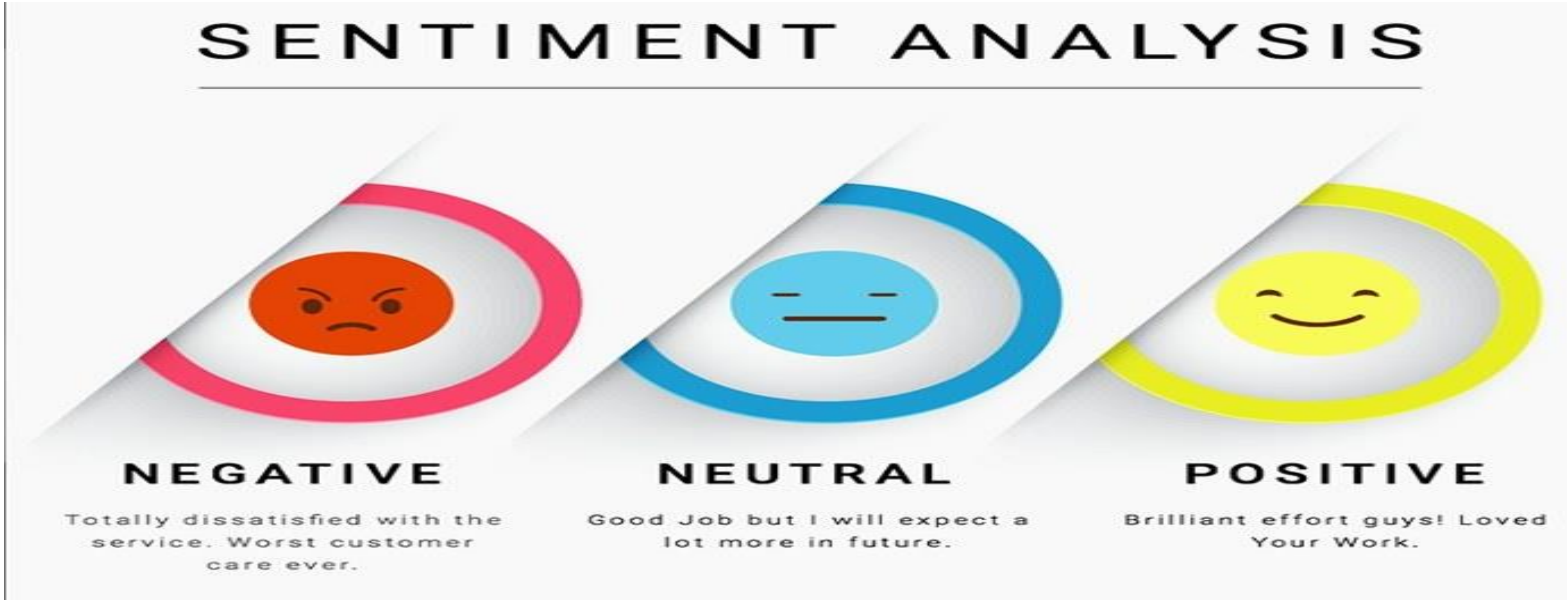
Content

1. **Dataset information.**
2. **Preprocessing the data**

   (Tokenization, Removing Stopwords, Lemmatization/ Stemming ,loer case **).**
3. **Vectorization (tf/idf).**
4. **Using different classifiers**

   (SVM, Random Forest, Kneighbors, BernoulliNB, LogisticRegression , MultinomialNB,Voting).ode
5. **Save Model**
6. **Check the model after load**

project is based on the sentiment analysis of airline data set which consists of reviews given by passengers of the particular airline and our classes consists of 3 sentiments which are negative, positive and neutral.

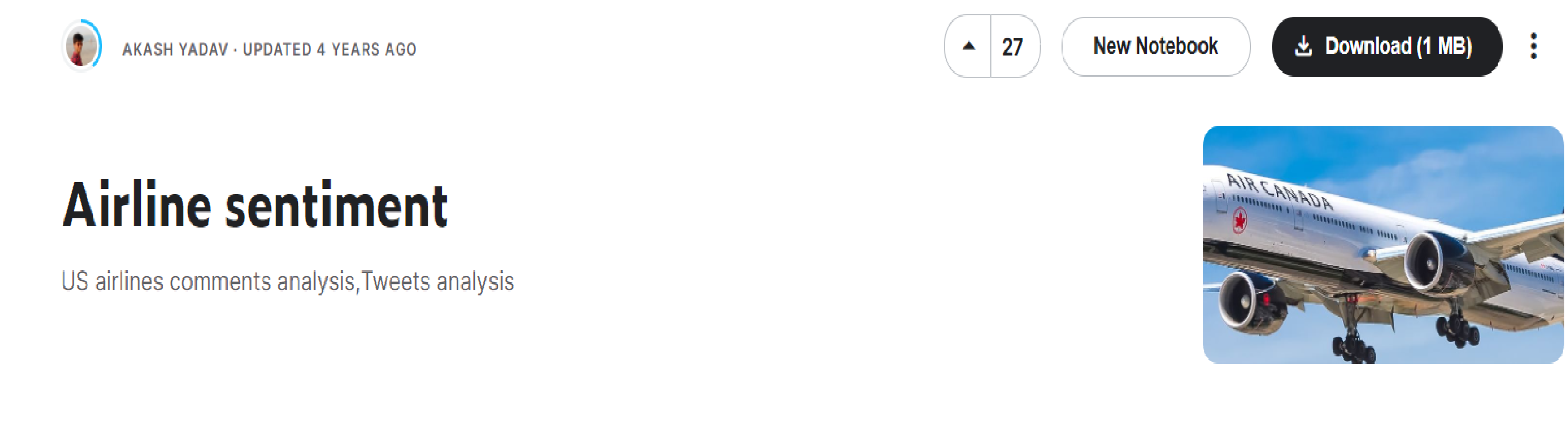


Dataset information.

- Information about dataset
- A dataset for US airlines comments analysis , Tweets analysis on Kaggle.
- Dataset link:

[Airline sentiment | Kaggle]

- This is US airlines data which contain comments of passengers on basis of service provided by airlines**(6 airlines)**



Dataset

Dataset have 15 features

(tweet_id,airline_sentiment,airline_sentiment_confidence,negativereason,negativereason_confidence,airline, airline_sentiment_gold,name,negativereason_gold,retwee_count,text,tweet_created,tweet_location,user_timezone)

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	NaN

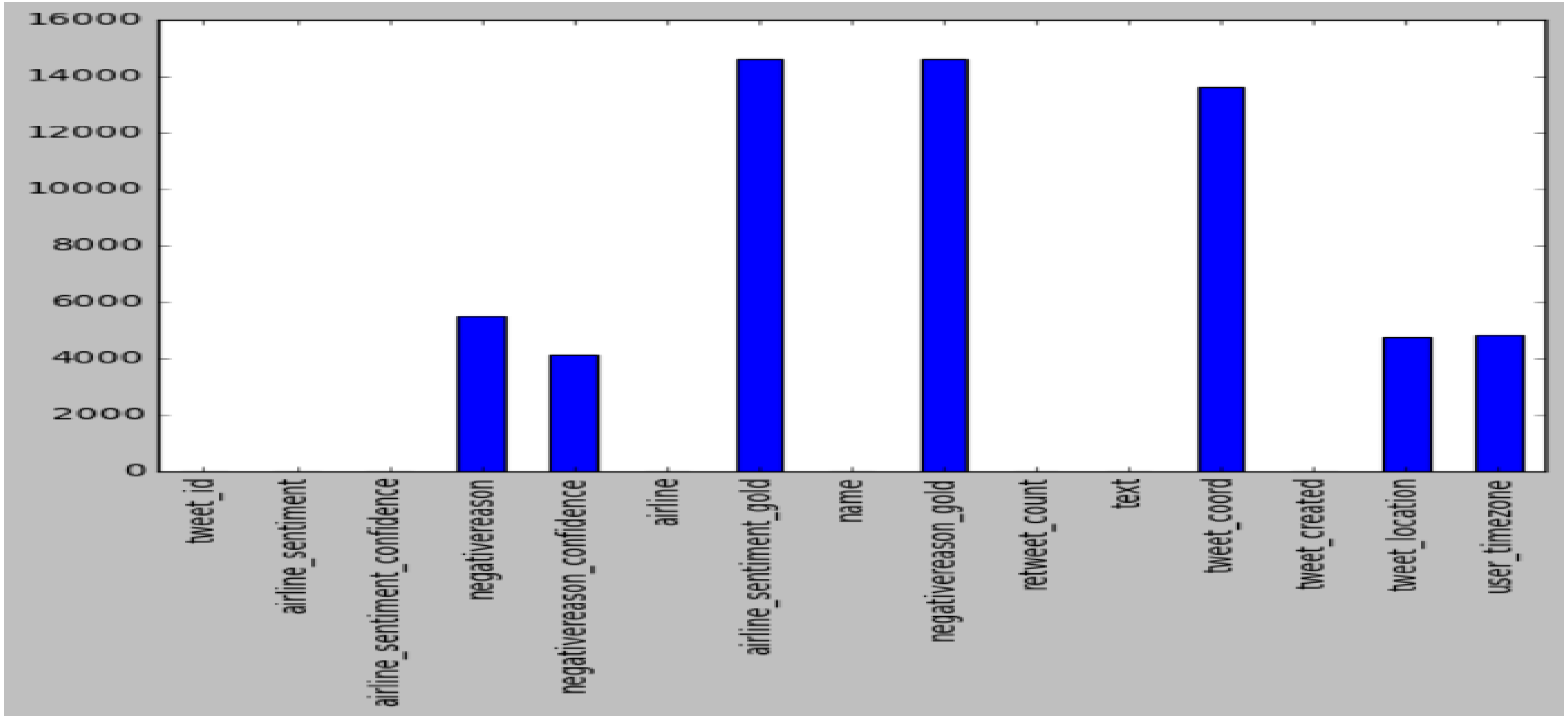
Dataset

- show descriptive statistics include those that summarize the central tendency , dispersion and shape of a dataset distribution.

count	1.464000e+04	14640.000000	10522.000000	14640.000000
mean	5.692184e+17	0.900169	0.638298	0.082650
std	7.791112e+14	0.162830	0.330440	0.745778
min	5.675883e+17	0.335000	0.000000	0.000000
25%	5.685592e+17	0.692300	0.360600	0.000000
50%	5.694779e+17	1.000000	0.670600	0.000000
75%	5.698905e+17	1.000000	1.000000	0.000000
max	5.703106e+17	1.000000	1.000000	44.000000

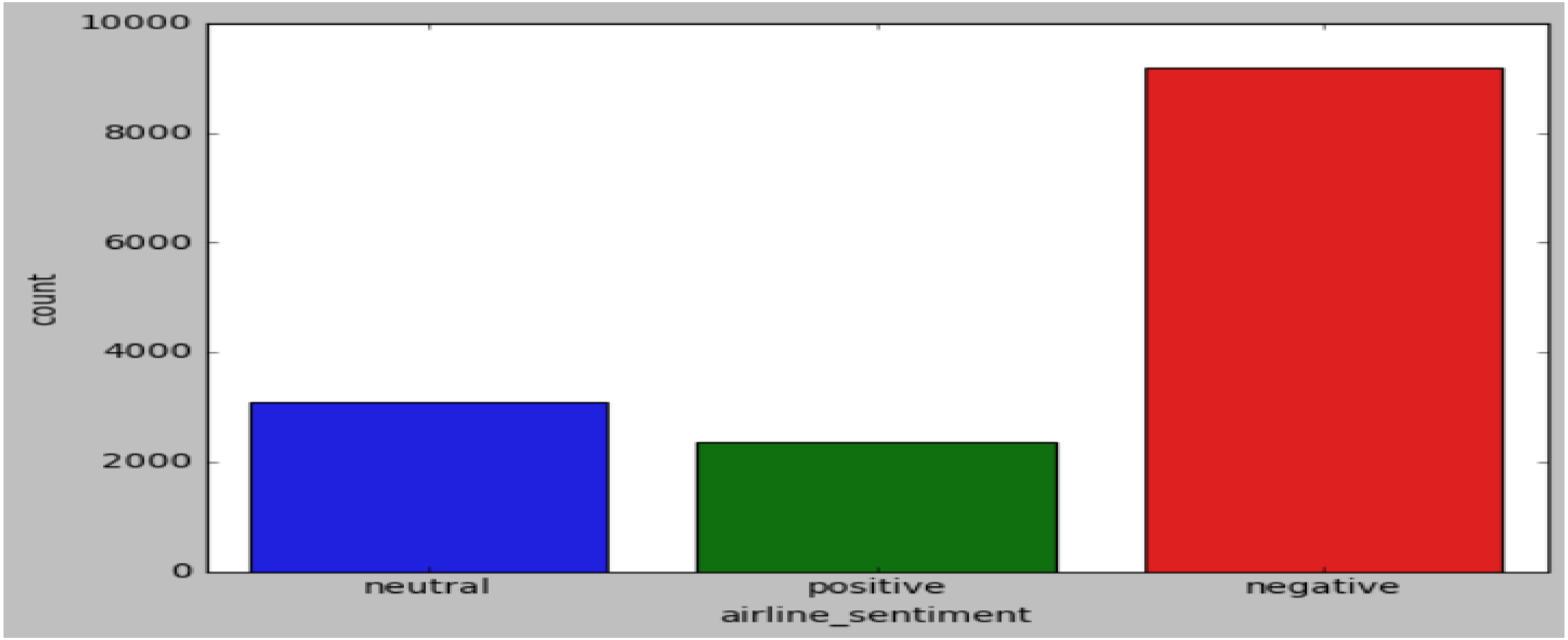
Dataset

- **Dataset check missing value**



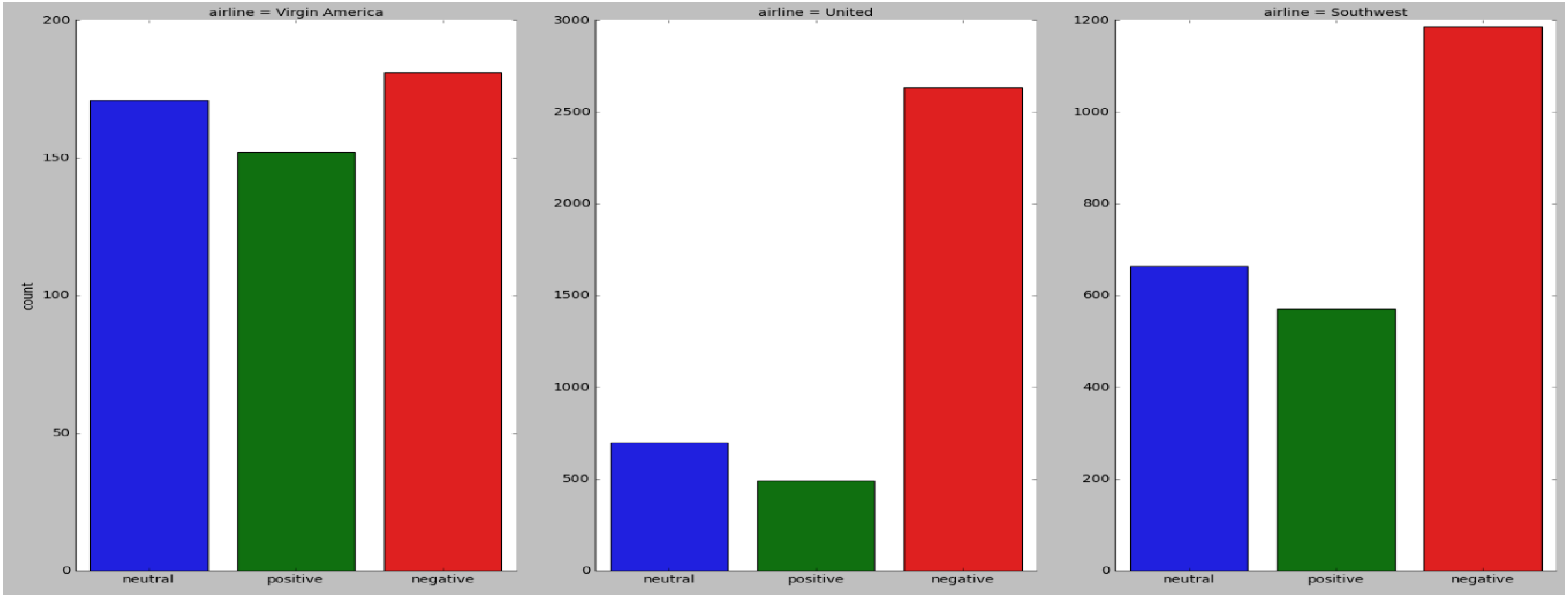
Dataset

- Represent the number of output:(Checking for imbalanced output)



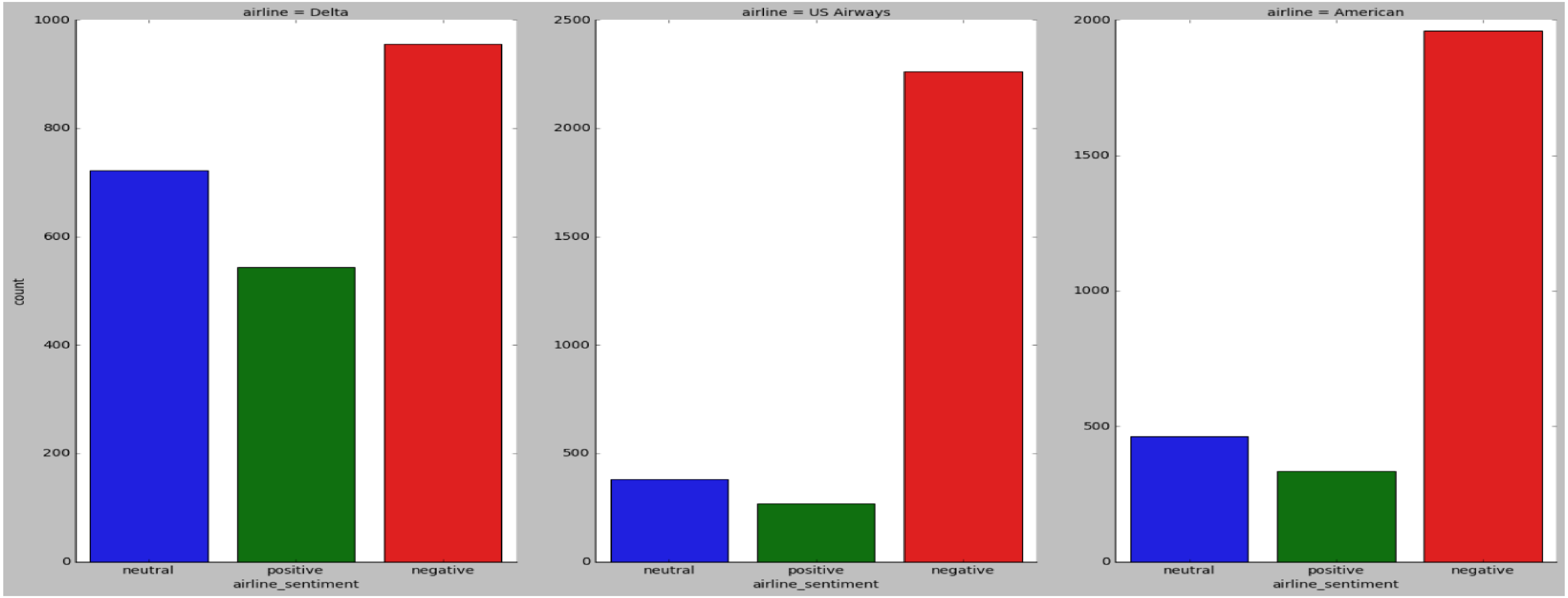
Dataset

- **Graphical representation of airline sentiment with airlines**



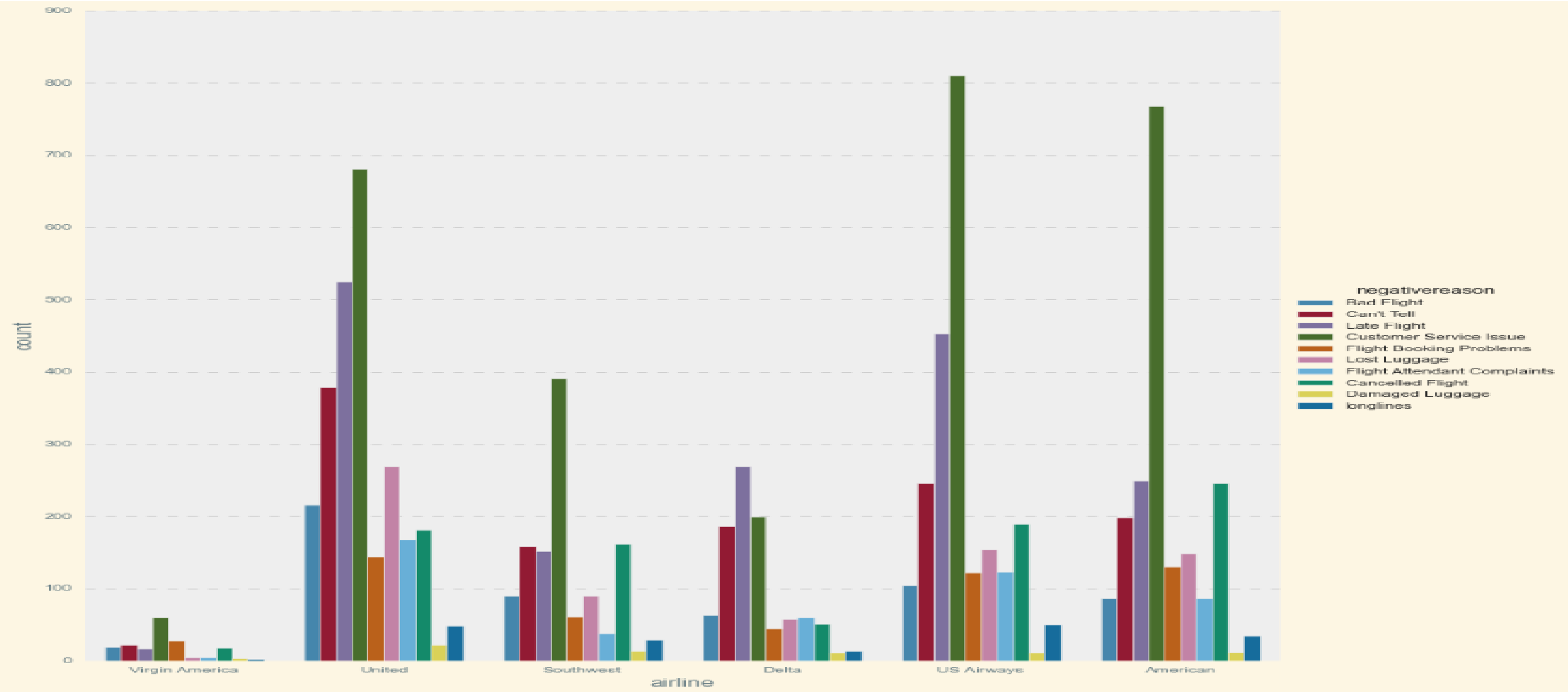
Dataset

- **Graphical representation of airline sentiment with airlines**



Dataset

- **Graphical representation of negative reason towards airlines**



Dataset Preprocessing

- The textual data we receive from the csv file consists of filler words which are not useful for us and has to be removed otherwise they will hinder the process.
- **Tokenization** - It means converting our sentences into words so that they can be easily checked or compared for any update or removal.
- **Removing Stopwords** - Removing stopwords is an important part of data preprocessing as these words are not useful and mainly disturbs our classifiers in choosing the import features as they dont have any meaning in the sentence like “the”,”I”,”has” etc.
- **Lemmatization/ Stemming** - The root form of the word can be generated from both lemmatization and stem but stem is capable of generating a word that isn’t present in the dictionary. We have considered Lemmatization as it is more commonly used than stemming.

Tokenization

- Dataset text.

```
@VirginAmerica What @dhepburn said.
@VirginAmerica plus you've added commercials to the experience... tacky.
@VirginAmerica I didn't today... Must mean I need to take another trip!
@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces &amp; they have little recourse
@VirginAmerica and it's a really big bad thing about it
@VirginAmerica seriously would pay $30 a flight for seats that didn't have this playing.
it's really the only bad thing about flying VA
@VirginAmerica yes, nearly every time I fly VX this “ear worm” won’t go away :)
@VirginAmerica Really missed a prime opportunity for Men Without Hats parody, there. https://t.co/mWpG7grEZP
@virginamerica Well, I didn't…but NOW I DO! :-D
@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.
```

- Dataset text after word tokenize

```
['@', 'VirginAmerica', 'What', '@', 'dhepburn', 'said', '.']
['@', 'VirginAmerica', 'plus', 'you', "'ve", 'added', 'commercials', 'to', 'the', 'experience', '...', 'tacky', '.']
['@', 'VirginAmerica', 'I', 'did', "n't", 'today', '...', 'Must', 'mean', 'I', 'need', 'to', 'take', 'another', 'trip', '!']
['@', 'VirginAmerica', 'it', "'s", 'really', 'aggressive', 'to', 'blast', 'obnoxious', '``', 'entertainment', "''", 'in', 'your', 'guest
s', "'", 'faces', '&', 'amp', ';', 'they', 'have', 'little', 'recourse']
['@', 'VirginAmerica', 'and', 'it', "'s", 'a', 'really', 'big', 'bad', 'thing', 'about', 'it']
['@', 'VirginAmerica', 'seriously', 'would', 'pay', '$', '30', 'a', 'flight', 'for', 'seats', 'that', 'did', "n't", 'have', 'this', 'playi
ng', '.', 'it', "'s", 'really', 'the', 'only', 'bad', 'thing', 'about', 'flying', 'VA']
['@', 'VirginAmerica', 'yes', ',', 'nearly', 'every', 'time', 'I', 'fly', 'VX', 'this', '“', 'ear', 'worm', '”', 'won', '’', 't', 'go', 'a
way', ':', ')']
['@', 'VirginAmerica', 'Really', 'missed', 'a', 'prime', 'opportunity', 'for', 'Men', 'Without', 'Hats', 'parody', ',', 'there', '.', 'htt
ps', ':', '//t.co/mWpG7grEZP']
['@', 'virginamerica', 'Well', ',', 'I', "didn't…but", 'NOW', 'I', 'DO', '!', ':', '-D']
['@', 'VirginAmerica', 'it', 'was', 'amazing', ',', 'and', 'arrived', 'an', 'hour', 'early', '.', 'You', "'re", 'too', 'good', 'to', 'me',
'.']
```

Stop words and punctuations. to remove

```
stops=set(stopwords.words('english'))
punctuations = list(string.punctuation)
stops.update(punctuations)
stops
```

```
{'!',
 '"',
 '#',
 '$',
 '%',
 '&',
 "'",
 '(',
 ')',
 '*',
 '+',
 ',',
 '-',
 '.',
 '/',
 ':',
 ';',
 '<',
 '=',
 '>',
 '?',
 '@',
 '[',
 '\\',
 ']',
 '^',
 '_',
 '`',
 'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
```

Apply Cleaning Dataset

- TOKENIZING,
 LEMMATIZING,
REMOVING STOPWORDS
PUNCTUATIONS,
pos.

```
In [15]: def clean_review(words):
             output_words = []
             for w in words:
                 if w.lower() not in stops:
                     pos = pos_tag([w])
                     clean_word = lemmatizer.lemmatize(w, pos = get_simple_pos(pos[0][1]))
                     output_words.append(clean_word.lower())
             return output_words
```

```
In [16]: document = [(clean_review(doc), category) for doc, category in documents]
```

• compare with data before and after clean

• Before clean.

```
(['@', 'VirginAmerica', 'What', '@', 'dhepburn', 'said', '.'], 'neutral')
(['@', 'VirginAmerica', 'plus', 'you', "'ve", 'added', 'commercials', 'to', 'the', 'experience', '...', 'tacky', '.'], 'positive')
(['@', 'VirginAmerica', 'I', 'did', "n't", 'today', '...', 'Must', 'mean', 'I', 'need', 'to', 'take', 'another', 'trip', '!'], 'neutral')
(['@', 'VirginAmerica', 'it', "'s", 'really', 'aggressive', 'to', 'blast', 'obnoxious', '``', 'entertainment', "''", 'in', 'your', 'guests', "'", 'face
s', '&', 'amp', ';', 'they', 'have', 'little', 'recourse'], 'negative')
(['@', 'VirginAmerica', 'and', 'it', "'s", 'a', 'really', 'big', 'bad', 'thing', 'about', 'it'], 'negative')
(['@', 'VirginAmerica', 'seriously', 'would', 'pay', '$', '30', 'a', 'flight', 'for', 'seats', 'that', 'did', "n't", 'have', 'this', 'playing', '.', 'i
t', "'s", 'really', 'the', 'only', 'bad', 'thing', 'about', 'flying', 'VA'], 'negative')
(['@', 'VirginAmerica', 'yes', ',', 'nearly', 'every', 'time', 'I', 'fly', 'VX', 'this', '“', 'ear', 'worm', '”', 'won', '’', 't', 'go', 'away', ':',
')'], 'positive')
(['@', 'VirginAmerica', 'Really', 'missed', 'a', 'prime', 'opportunity', 'for', 'Men', 'Without', 'Hats', 'parody', ',', 'there', '.', 'https', ':',
'//t.co/mWpG7grEZP'], 'neutral')
(['@', 'virginamerica', 'Well', ',', 'I', "didn't…but", 'NOW', 'I', 'DO', '!', ':', '-D'], 'positive')
(['@', 'VirginAmerica', 'it', 'was', 'amazing', ',', 'and', 'arrived', 'an', 'hour', 'early', '.', 'You', "'re", 'too', 'good', 'to', 'me', '.'], 'posi
tive')
```

• After clean.

```
(['virginamerica', 'dhepburn', 'say'], 'neutral')
(['virginamerica', 'plus', "'ve", 'add', 'commercial', 'experience', '...', 'tacky'], 'positive')
(['virginamerica', "n't", 'today', '...', 'must', 'mean', 'need', 'take', 'another', 'trip'], 'neutral')
(['virginamerica', "'s", 'really', 'aggressive', 'blast', 'obnoxious', '``', 'entertainment', "''", 'guest', 'face', 'amp', 'little', 'recourse'], 'neg
ative')
(['virginamerica', "'s", 'really', 'big', 'bad', 'thing'], 'negative')
(['virginamerica', 'seriously', 'would', 'pay', '30', 'flight', 'seat', "n't", 'play', "'s", 'really', 'bad', 'thing', 'fly', 'va'], 'negative')
(['virginamerica', 'yes', 'nearly', 'every', 'time', 'fly', 'vx', '“', 'ear', 'worm', '”', '’', 'go', 'away'], 'positive')
(['virginamerica', 'really', 'miss', 'prime', 'opportunity', 'men', 'without', 'hats', 'parody', 'http', '//t.co/mwpg7grezp'], 'neutral')
(['virginamerica', 'well', "didn't…but", '-d'], 'positive')
(['virginamerica', 'amaze', 'arrive', 'hour', 'early', "'re", 'good'], 'positive')
```

Vectorization

- After than we have a list of lemmatized words, now starts the main problem how can we **find the most frequency words** the prerequisite for this is that the list
- we chose the ***TF-IDF VECTORIZER***.

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
count_vect=TfidfVectorizer(max_features=5000, max_df=0.8, min_df=0.001)
X_train_features=count_vect.fit_transform(X_train)
X_test_features=count_vect.transform(X_test)
```

```
import pickle

with open('tfidf.pickle', 'wb') as f:
    pickle.dump(count_vect, f)
```

classification models

- **Using different classifiers**

1.SVM(Support Vector Machine) - Gives accuracy of 77%

2.Random Forest Classifier - Gives accuracy of 75%

3.KNeighbors- Gives accuracy of 74%

4.BernoulliNB- Gives accuracy of 76%

5.LogisticRegression- Gives accuracy of 78%

6.MultinomialNB- Gives accuracy of 75%

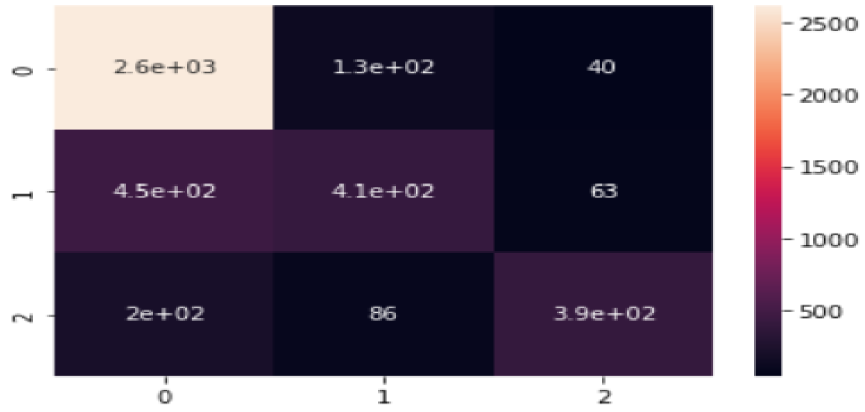
- **Making Voting to six Models**- Gives accuracy of 75%

SVM(Support Vector Machine)

```
clf = SVC(degree=11)
acc_tarin , acc_test = eval_model(clf,X_train_features,y_train,X_test_features,y_test)
```

```
SVC
acc train: 0.927400468384075
acc test: 0.7786885245901639
----------------------------
```

```
import seaborn as sns
import matplotlib.pyplot as plt
y_pred=clf.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
plt.show()
```

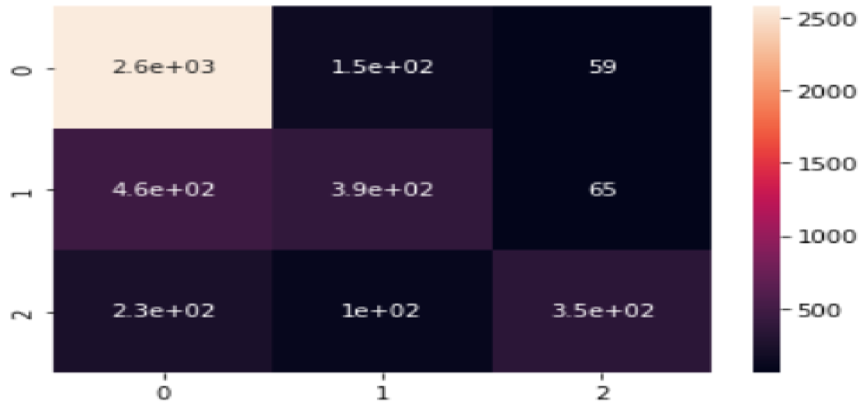


Random Forest Classifier

```
clf1=RandomForestClassifier(n_estimators=100, n_jobs=-1)
acc_tarin1,acc_test1=eval_model(clf1,X_train_features,y_train,X_test_features,y_test)
```

```
RandomForestClassifier
acc train: 0.9917056986729118
acc test: 0.7550091074681239
----------------------------
```

```
import seaborn as sns
import matplotlib.pyplot as plt
y_pred1=clf1.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred1)
sns.heatmap(cm,annot=True)
plt.show()
```

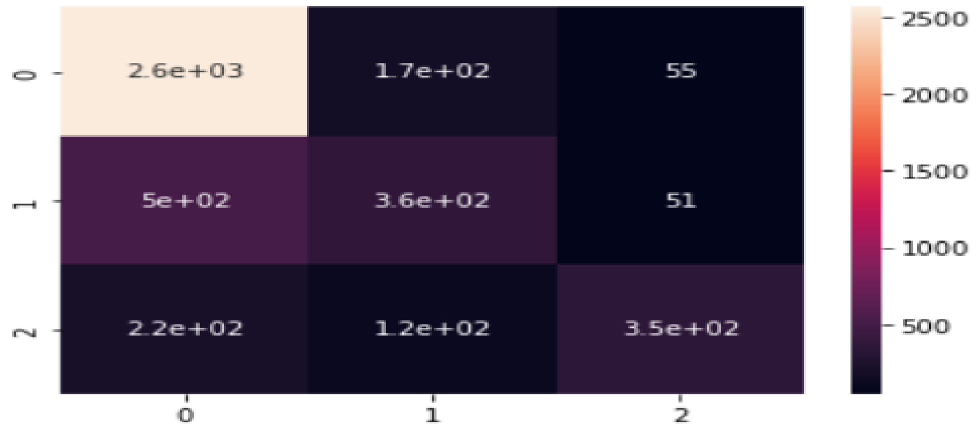


KNeighbors

```
clf2=KNeighborsClassifier(n_neighbors=57)
acc_tarin2,acc_test2=eval_model(clf2,X_train_features,y_train,X_test_features,y_test)
```

```
KNeighborsClassifier
acc train:  0.750975800156128
acc test:  0.7461293260473588
----------------------------
```

```
y_pred2=clf2.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred2)
sns.heatmap(cm,annot=True)
plt.show()
```

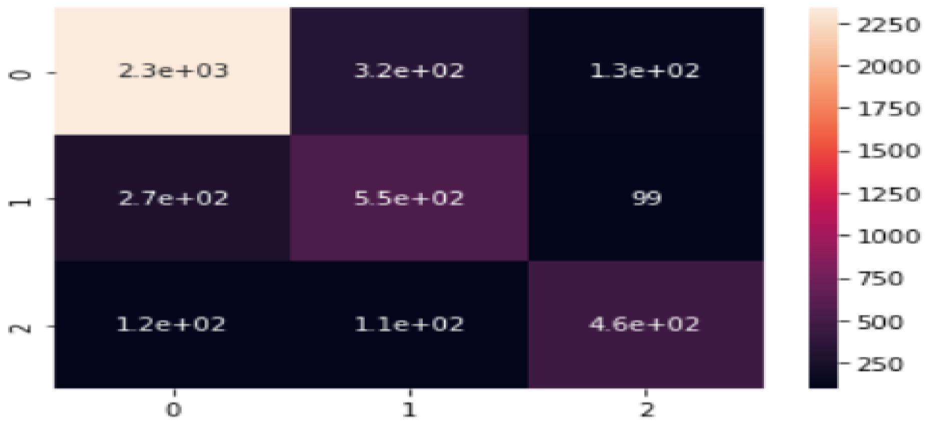


BernoulliNB

```
clf3=BernoulliNB()
acc_tarin3,acc_test3=eval_model(clf3,X_train_features,y_train,X_test_features,y_test)
```

```
BernoulliNB
acc train: 0.8033762685402029
acc test: 0.7607012750455373
-----------------------------
```

```
y_pred3=clf3.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred3)
sns.heatmap(cm,annot=True)
plt.show()
```

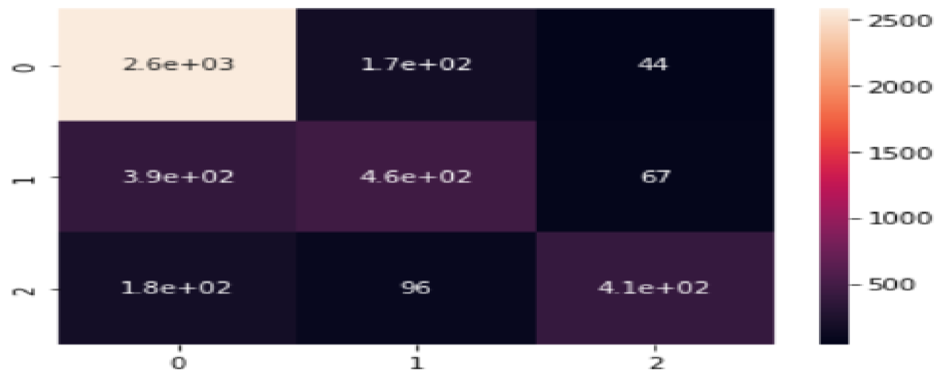


LogisticRegression

```
clf4=LogisticRegression(max_iter=100,random_state=0)
acc_tarin4,acc_test4=eval_model(clf4,X_train_features,y_train,X_test_features,y_test)
```

```
LogisticRegression
acc train: 0.877927400468384
acc test: 0.7809653916211293
----------------------------
```

```
y_pred4=clf4.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred4)
sns.heatmap(cm,annot=True)
plt.show()
```

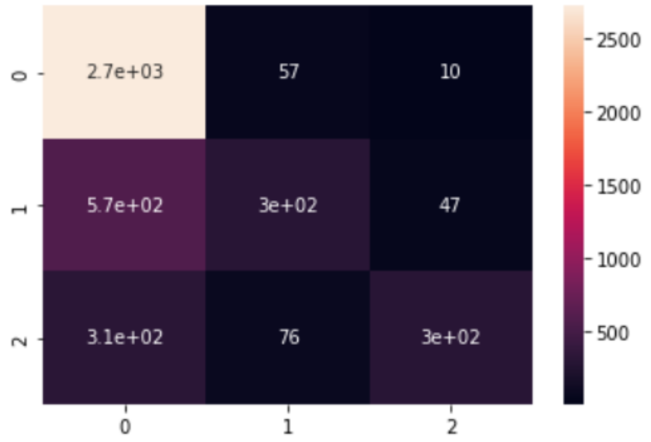


MultinomialNB

```
clf5=MultinomialNB()
acc_tarin5,acc_test5=eval_model(clf5,X_train_features,y_train,X_test_features,y_test)
```

```
MultinomialNB
acc train: 0.773224043715847
acc test: 0.75591985428051
----------------------------
```

```
y_pred5=clf5.predict(X_test_features)
cm=confusion_matrix(y_test,y_pred5)
sns.heatmap(cm,annot=True)
plt.show()
```



Voting to six Models

```
estimators = [
    ('Svm',clf),
    ('RANDOM FOREST', clf1),
    ('KNN', clf2),
    ('BernoulliNB', clf3),
    ('LogisticRegression',clf4),
    ('MultinomialNB',clf5)
]

voting_clf = VotingClassifier(estimators)
```

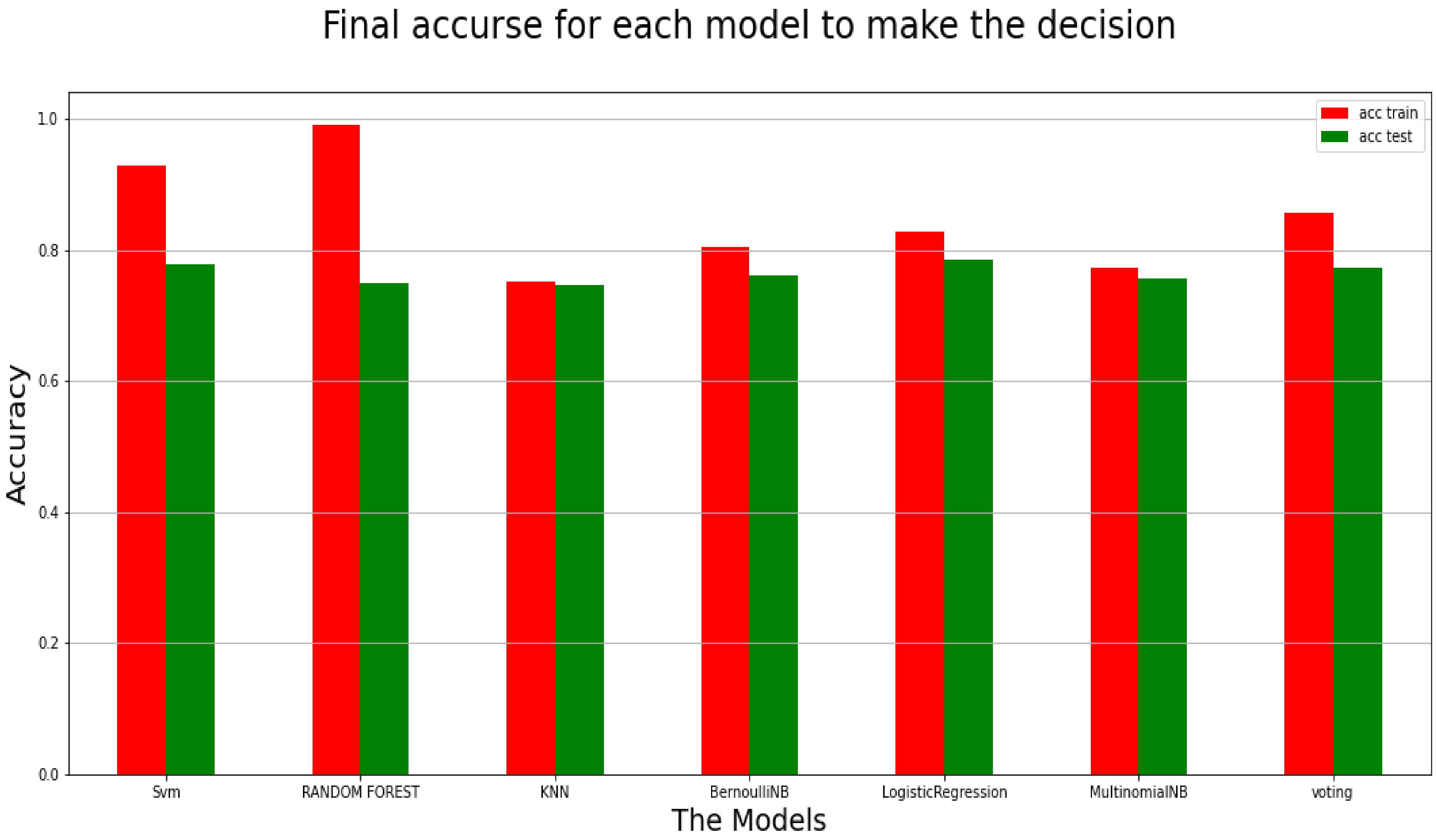
```
all_estimators = estimators + [('voting', voting_clf)]

final_results = {
    'model': [],
    'acc train': [],
    'acc test': []
}

for (name, clf) in all_estimators:
    acc_train, acc_test = eval_model(
        clf, X_train_features,y_train,X_test_features,y_test
    )
    final_results['model'].append(name)
    final_results['acc train'].append(acc_train)
    final_results['acc test'].append(acc_test)
```

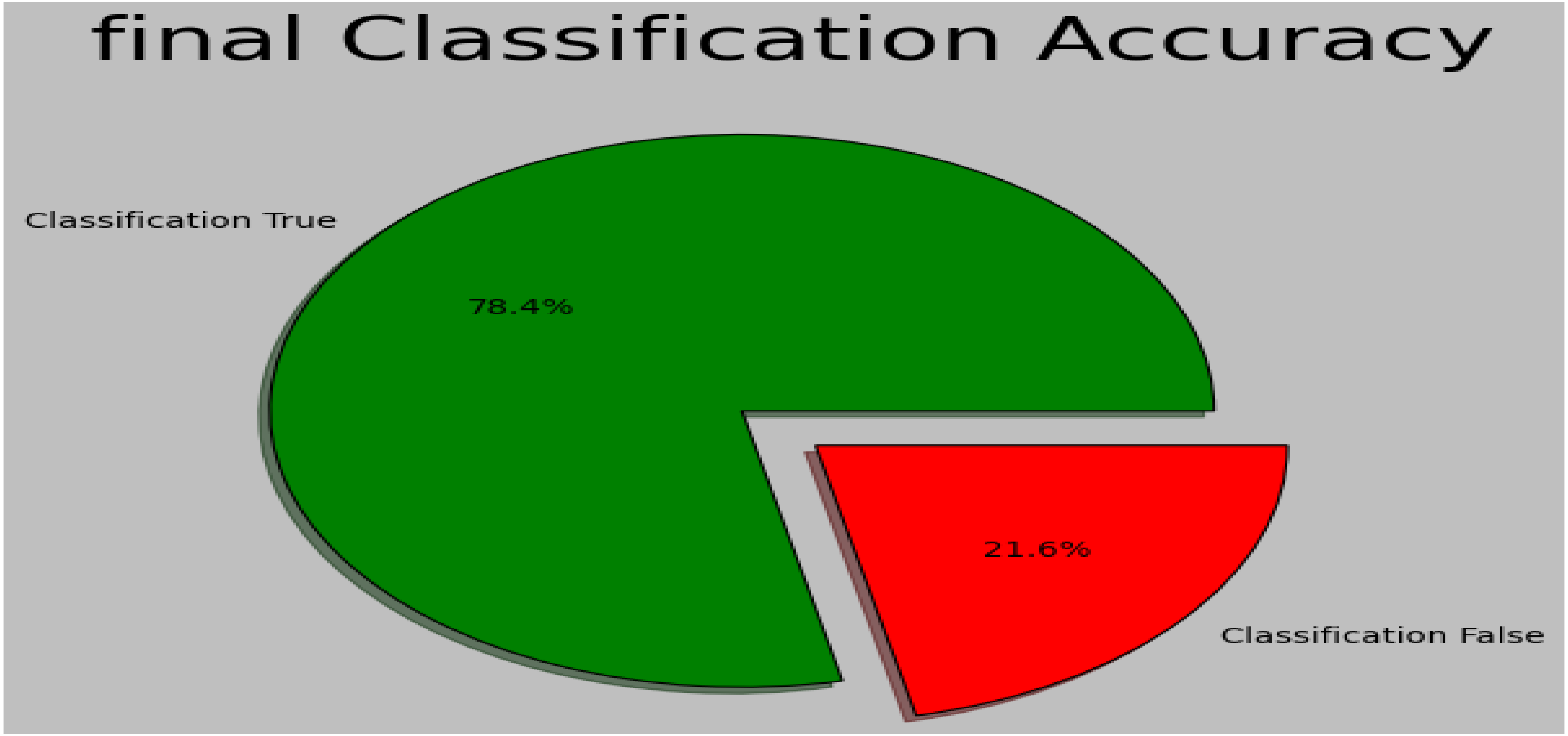
```
acc train: 0.8564597970335676
acc test: 0.7734517304189436
```

Final accurse for each model to make decision



	model	acc train	acc test
0	LogisticRegression	0.877927	0.780965
1	Svm	0.958528	0.778916
2	RANDOM FOREST	0.996097	0.761157
3	voting	0.857533	0.757286
4	KNN	0.737900	0.745446
5	BernoulliNB	0.816257	0.735656
6	MultinomialNB	0.731460	0.682832

Final test using **LogisticRegression**.



Test accuracy model

In [47]:
```python
randnum=np.random.randint(1,1000)
sentence=X_test[randnum]
print(sentence)
sentence_features=X_test_features[randnum]
print("----------------------------------------------------------------------")
print("the actual Sentiment Analysis\n\t\t\t",y_test[randnum])
print("----------------------------------------------------------------------")
print('my model predict:\n\t\t\t',final_y_pred[randnum])
```

```
virginamerica great deal already think 2nd trip australia amp n't even go 1st trip yet p
----------------------------------------------------------------------
the actual Sentiment Analysis
                         positive
----------------------------------------------------------------------
my model predict:
                         negative
```

In [48]:
```python
randnum=np.random.randint(1,1000)
sentence=X_test[randnum]
print(sentence)
sentence_features=X_test_features[randnum]
print("----------------------------------------------------------------------")
print("the actual Sentiment Analysis\n\t\t\t",y_test[randnum])
print("----------------------------------------------------------------------")
print('my model predict:\n\t\t\t',final_y_pred[randnum])
```

```
americanair pls tell get person phone asap
----------------------------------------------------------------------
the actual Sentiment Analysis
                         negative
----------------------------------------------------------------------
my model predict:
                         negative
```

In [49]:
```python
randnum=np.random.randint(1,1000)
sentence=X_test[randnum]
print(sentence)
sentence_features=X_test_features[randnum]
print("----------------------------------------------------------------------")
print("the actual Sentiment Analysis\n\t\t\t",y_test[randnum])
print("----------------------------------------------------------------------")
print('my model predict:\n\t\t\t',final_y_pred[randnum])
```

```
jetblue celebrates 15-year anniversary new livery cnnmoney http //t.co/p7skxzve1a
----------------------------------------------------------------------
the actual Sentiment Analysis
                         neutral
----------------------------------------------------------------------
my model predict:
                         neutral
```

In [53]:
```python
randnum=np.random.randint(1,1000)
sentence=X_test[randnum]
print(sentence)
sentence_features=X_test_features[randnum]
print("----------------------------------------------------------------------")
print("the actual Sentiment Analysis\n\t\t\t",y_test[randnum])
print("----------------------------------------------------------------------")
print('my model predict:\n\t\t\t',final_y_pred[randnum])
```

```
united terrific many thanks looking forward back ua tomorrow great flight vancouver
----------------------------------------------------------------------
the actual Sentiment Analysis
                         positive
----------------------------------------------------------------------
my model predict:
                         positive
```

Save mode

Save Model

```python
import pickle
with open('saved-model.pickle', 'wb') as f:
    pickle.dump(final_model, f)
```

Python

Check The Model After Load

```python
import pickle

with open('saved-model.pickle', 'rb') as f:
    my_model = pickle.load(f)

with open('tfidf.pickle', 'rb') as f:
    my_feature = pickle.load(f)
```

Python

♥♥ Thank you.♥♥