Description of Dataset

```
import pandas as pd
df = pd.read_csv("/content/apple-twitter-sentiment-texts.csv")
print('rows and columns:', df.shape)
print(df.head())
```

```
rows and columns: (1630, 2)
                                                 text  sentiment
0  Wow. Yall needa step it up @Apple RT @heynyla:...         -1
1  What Happened To Apple Inc?   http://t.co/FJEX...          0
2  Thank u @apple I can now compile all of the pi...          1
3  The oddly uplifting story of the Apple co-foun...          0
4  @apple can i exchange my iphone for a differen...          0
```

What the Dataset is and What Our model Should Predict

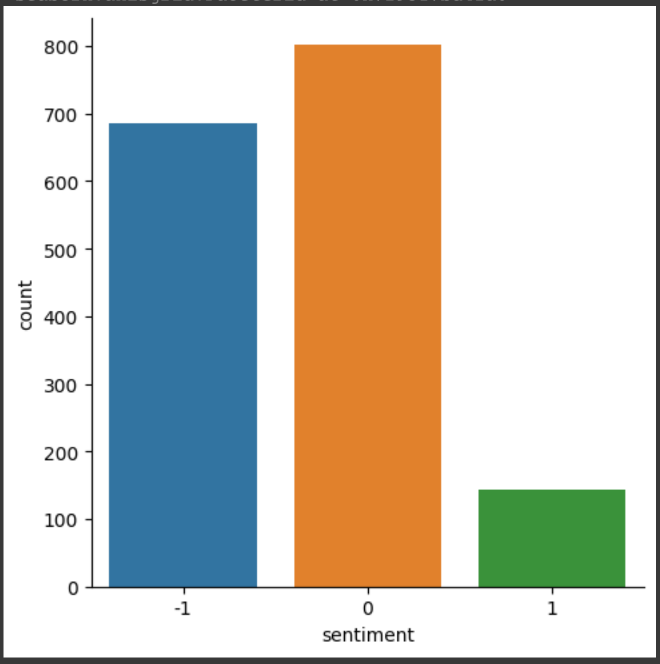
This dataset is compilation of twitter posts mentioning the company Apple. Each reply has a sentiment assigned to it and our models aim to predict the sentiment of the twitter posts using a subset of the dataset. -1 represents a negative sentiment, 0 is neutral, 1 is postive.

Graphs

```
import seaborn as sb

# plot distribution of classes
sb.catplot(x="sentiment", kind="count", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f9317bd4fa0>
```



There are very few postive tweets compared to ones with neutral and negative sentiments.

Naive Bayes

Text Pre-Processing

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

stopwords = list(set(stopwords.words('english')))
#vectorizer = TfidfVectorizer(stop_words=stopwords)
vectorizer = TfidfVectorizer(stop_words=None)
#vectorizer = TfidfVectorizer(ngram_range=(1, 2), max_features=50000, min_df=2, stop_words=None)

#import re

#df['text'].replace('[\d][\d]+', ' num ', regex=True, inplace=True)
#df['text'].replace('[!@#*][!@#*]+', ' punct ', regex=True, inplace=True)
#df['text'].replace('[A-Z][A-Z]+', ' caps ', regex=True, inplace=True)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
# set up X and y
X = df.text
y = df.sentiment
```

train test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)
```

```
X_train.shape
```

```
(1304,)
```

```
# apply tfidf vectorizer
X_train = vectorizer.fit_transform(X_train)  # fit and transform the train data
X_test = vectorizer.transform(X_test)        # transform only the test data
```

train the naive bayes classifier

```
from sklearn.naive_bayes import MultinomialNB

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

evaluate on test data

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# make predictions on the test data
pred = naive_bayes.predict(X_test)

# print confusion matrix
print(confusion_matrix(y_test, pred))
```

```
[[126  10   0]
 [ 35 124   0]
 [ 14  17   0]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

          -1       0.72      0.93      0.81       136
           0       0.82      0.78      0.80       159
           1       0.00      0.00      0.00        31

    accuracy                           0.77       326
   macro avg       0.51      0.57      0.54       326
weighted avg       0.70      0.77      0.73       326

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
```

Processing the text to recognize punctation and caps actually reduced accuracy for this dataset. Notice that positive sentiment was never predicted correctly shown by the confusion matrix above so while using Naive Bayes, we should attempt to predict negative or neutral sentiments. Inclusion/Exclusion of stopwords did not have a significant effect on accuracy. Changes to the vectorizer also made no significant change.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(C=2.5, n_jobs=4, solver='lbfgs', random_state=17, verbose=1)
clf.fit(X_train, y_train)
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   1 out of   1 | elapsed:    1.2s finished
▾              LogisticRegression
LogisticRegression(C=2.5, n_jobs=4, random_state=17, verbose=1)
```

```
pred2 = clf.predict(X_test)
```

```
confusion_matrix(y_test, pred2)
```

```
array([[118,  16,   2],
       [ 20, 137,   2],
       [ 10,   9,  12]])
```

```
print(classification_report(y_test, pred2))
```

```
              precision    recall  f1-score   support

          -1       0.80      0.87      0.83       136
           0       0.85      0.86      0.85       159
           1       0.75      0.39      0.51        31

    accuracy                           0.82       326
   macro avg       0.80      0.71      0.73       326
weighted avg       0.82      0.82      0.81       326
```

Including stopwords improved accuracy. Processing the text to recognize punctation and caps also reduced accuracy for this dataset. Changing the vectorizer also reduced accuracy.

Neural Networks

```
from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(14, 2), random_state=1)
classifier.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(14, 2), random_state=1,
              solver='lbfgs')
```

```
pred3 = classifier.predict(X_test)
```

```
confusion_matrix(y_test, pred3)
```

```
array([[109,  16,  11],
       [ 12, 139,   8],
       [ 14,  12,   5]])
```

```
print(classification_report(y_test, pred3))
```

```
              precision    recall  f1-score   support

          -1       0.81      0.80      0.80       136
           0       0.83      0.87      0.85       159
           1       0.21      0.16      0.18        31

    accuracy                           0.78       326
   macro avg       0.62      0.61      0.61       326
weighted avg       0.76      0.78      0.77       326
```

Changing the vectorizer resulted in a miniscule increase in accuracy but it correctly identified postive sentiment more often. The current vectorizer implementation doesn't recognize it at all. lbfgs solver had the best perfomance and I couldn't reasonably find better hidden layer values than (15, 2).

Analysis of Performance

Refer to the end of each section above for more information.

Naive Bayes

I was able to achieve .77 accuracy with Naive Bayes. Interestingly, it never correctly identified any tweet with positive sentiment. With so few positive tweets this isn't totally unexpected. However, it distinguised between neutral and negative relatively well.

Logistic Regression

Logistic regression performed the best of the three models. It achieved .82 accuracy. It had the least trouble with identifying positive sentiment tweets but still misidentified many.

Neural Networks

I expected the neural network to perform the best but it performed about the same as Naive Bayes (.78). I think that since this dataset is small a neural network couldn't do much. I also think since there were so few positive tweets, all the models had too little data to identify positve sentiment tweets consistently.