```
#%%

#%%
import pandas
import seaborn
import sklearn
from sklearn.feature_extraction.text import
  TfidfVectorizer
from sklearn.model_selection import
  train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,
  precision_score, recall_score, f1_score,
  confusion_matrix
from sklearn.linear_model._logistic import
  LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn import metrics as metrics
import numpy as np
from nltk.corpus import stopwords
from sklearn.neural_network import MLPClassifier



# Open and read the files into pandas dataframes
testdf = pandas.read_csv("tweet_emotions.csv")
# print(testdf['sentiment'])

seaborn.catplot(x="sentiment" ,kind="count", data=
  testdf)

# Split the data
X = testdf.content
y = testdf.sentiment

# 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)

# Use default vectorizer
```

```
34 vectorizer = TfidfVectorizer()
35
36 # vectorize
37 X_train = vectorizer.fit_transform(X_train)
38 X_test = vectorizer.transform(X_test)
39
40 # Train the classifier
41 naive_bayes = MultinomialNB()
42 naive_bayes.fit(X_train, y_train)
43
44 # Make predictions and print confusion matrix
45 pred = naive_bayes.predict(X_test)
46 print(confusion_matrix(y_test, pred))
47 print('accuracy score: ', accuracy_score(y_test,
   pred))
48
49 #%% md
50 ## Naive Bayes
51 #%%
52 # Train test/split
53 X_train, X_test, y_train, y_test = train_test_split
   (X, y, test_size=0.2, train_size=0.8, random_state=
   1234)
54
55 # Use default vectorizer
56 vectorizer = TfidfVectorizer()
57
58 # vectorize
59 X_train = vectorizer.fit_transform(X_train)
60 X_test = vectorizer.transform(X_test)
61
62 # Train the classifier
63 naive_bayes = MultinomialNB()
64 naive_bayes.fit(X_train, y_train)
65
66 # Make predictions and print confusion matrix
67 pred = naive_bayes.predict(X_test)
68 print(confusion_matrix(y_test, pred))
69 print('accuracy score: ', accuracy_score(y_test,
   pred))
70 #%% md
```

```
As we can see, the performance of NB on this dataset has much to be desired, likely due to the larger amount of categories it has to work with. With each category getting less data, its predictions become less accurate.

## Linear Regression
#%%
# Create and fit pipeline

pipe1 = Pipeline([
        ('tfidf', TfidfVectorizer()),
        ('logreg', LogisticRegression(multi_class='multinomial', solver='lbfgs',class_weight='balanced')),
])

pipe1.fit(testdf.content, testdf.sentiment)

# Test and evaluate
pred = pipe1.predict(testdf.content)

print("Confusion matrix:\n", metrics.confusion_matrix(testdf.sentiment, pred))

print("\nOverall accuracy: ", np.mean(pred==testdf.sentiment))
#%% md
Linear Regression does notably better, however there is still much to be desired. This overall poor performance is likely due to a combination of multiple categories, biases in the data, and the nature of the task it is asked to do (categorize the sentiments of tweets).

## Neural Networks
#%%
from nltk.corpus import stopwords
from sklearn.neural_network import MLPClassifier

# Text preprocessing
```

```
 99
100 stopwords = set(stopwords.words('english'))
101 vectorizer = TfidfVectorizer(stop_words=stopwords
    , binary=True)
102
103 # Set up pipe, just like the Linear Regression
104
105 pipe1 = Pipeline([
106         ('tfidf', TfidfVectorizer()),
107         ('neuralnet', MLPClassifier(solver='lbfgs'
    , alpha=1e-5,
108                     hidden_layer_sizes=(15, 7),
    random_state=1)),
109          ])
110
111 pipe1.fit(testdf.content, testdf.sentiment)
112
113 # Predict and test
114
115 pred = pipe1.predict(testdf.content)
116
117 print("Confusion matrix:\n", metrics.
    confusion_matrix(testdf.sentiment, pred))
118
119 print("\nOverall accuracy: ", np.mean(pred==testdf
    .sentiment))
120 #%% md
121 Finally, we can see that the Neural Network did
    the best out of all the methods, doing slightly
    better than the Linear Regression and over twice
    as good as the Naive Bayes method. However, as
    said before, the results still leave much to be
    desired to the the biases in the data.
```