Reddit Classification (Project 3)

Problem Statement:

Create a classification model that will predict which subreddit the post is from.

Classification Process

2. Extract through Reddit API

- Multiple attempts to attain a subreddit with high number of text-only posts
- Saved as CSV file

3. Clean Data

- Remove Null values
- Clean the text
  - Remove symbols & numbers
  - Remove Stop Words
    - Utilized an adapted stop word collection which included possible keywords

```
#Add more words in the stopwords
stop_words = stopwords.words('english')
additional_stop_words = ['r/askpsychology', 'psychology', 'sociology' , 'psych' , 'soci', 'r/sociology' , 'psy', 'soc'
stop_words.extend(additional_stop_words)
```

4

Modelling the Data

How did we model the appropriate data to predict the subreddit?

4 Models were used for the modelling process

1. Tf-idf Vectorizer & Logistic Regression
2. Countvectorizer & Logistic Regression
3. Tf-idf Vectorizer & Multinomial NB
4. Countvectorizer & Multinomial NB

Example of pipeline and GridSearchCV

```
#Baseline Score to check accuracy
y.value_counts(normalize=True)
```

```
askpsychology    0.548902
sociology        0.451098
Name: subreddit, dtype: float64
```

```
#Using pipelines ,using tvec and MNB

pipe = Pipeline([
    #('cvec', CountVectorizer()),
    ('tvec', TfidfVectorizer()),
    ('cls', MultinomialNB()),
    #('lr' , LogisticRegression())

])
```

```
#Using tvec and MNB
pipe_params = {
     'tvec__max_features': [2500, 3000, 3500],
    'tvec__min_df': [0.01, 0.03, 0.05],
    'tvec__max_df': [0.85,.9, .95],
    'tvec__ngram_range': [(1,1), (1,2),(1,3)],
    'tvec__norm': ['l1', 'l2']
}
gs_1 = GridSearchCV(pipe, param_grid=pipe_params, cv=3)
gs_1.fit(X_train, y_train)
print(gs.best_score_)
gs_1.best_params_
```

```
0.8074534161490683
{'tvec__max_df': 0.85,
 'tvec__max_features': 2500,
 'tvec__min_df': 0.01,
 'tvec__ngram_range': (1, 2),
 'tvec__norm': 'l2'}
```

```
#Using tvec and MNB
gs_1.score(X_train, y_train)
```

```
0.8881987577639752
```

```
#Using tvec and MNB
gs_1.score(X_test, y_test)
```

```
0.8218085106382979
```

5

Evaluation of model

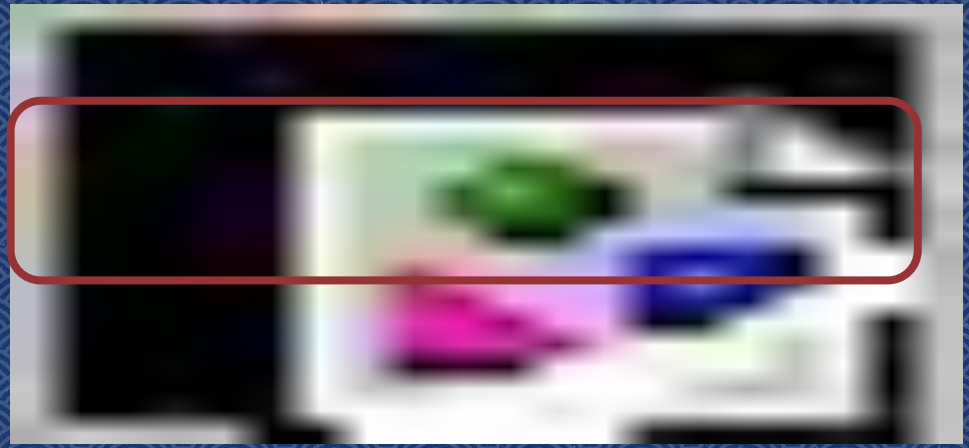
Which model performed well?

```
predicted = gs_1.predict(X_test)
print (classification_report(y_test, predicted))
```

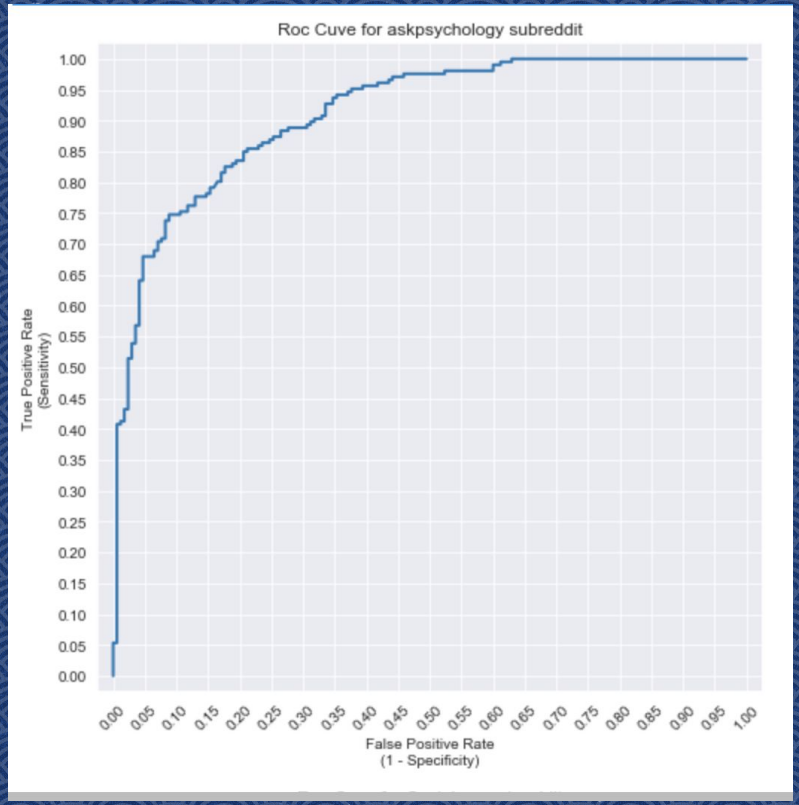
	precision	recall	f1-score	support
askpsychology	0.83	0.85	0.84	206
sociology	0.81	0.79	0.80	170
micro avg	0.82	0.82	0.82	376
macro avg	0.82	0.82	0.82	376
weighted avg	0.82	0.82	0.82	376

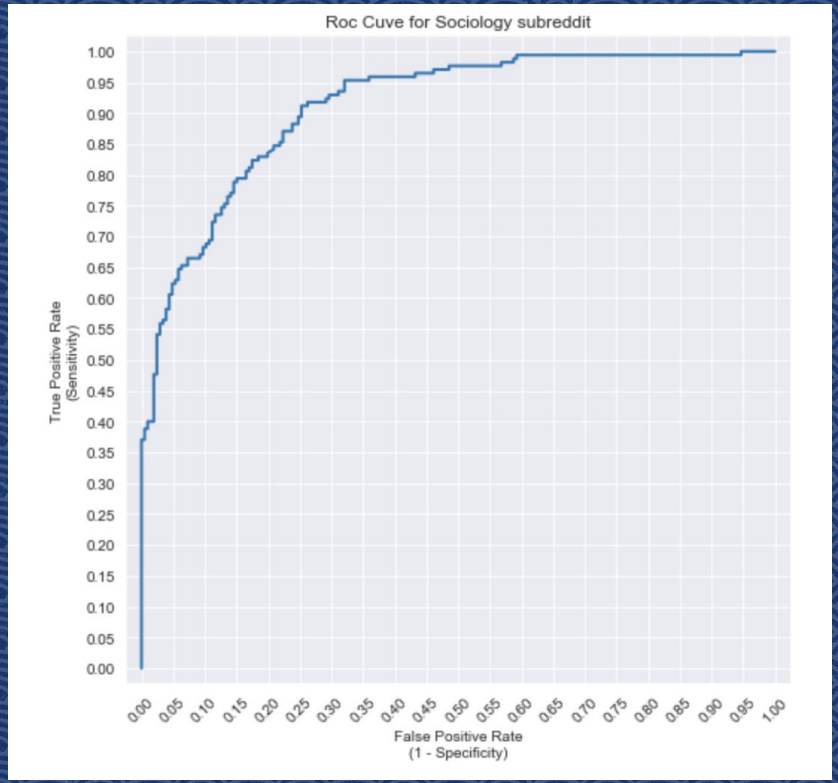
TF-IDF &
Multinomial NB

Countvectorizer &
Multinomial NB



Roc Curve


Roc Cuve for askpsychology subreddit
True Positive Rate
(Sensitivity)
False Positive Rate
(1 - Specificity)


Roc Cuve for Sociology subreddit
True Positive Rate
(Sensitivity)
False Positive Rate
(1 - Specificity)

Findings

```
predicted = gs_1.predict(X_test)

# Incorrectly classified
incorrect_preds = X_test[(predicted != y_test)]


incorrect_preds.shape
```

```
(67,)
```

- Out of 374 test posts, 67 are predicted incorrectly.

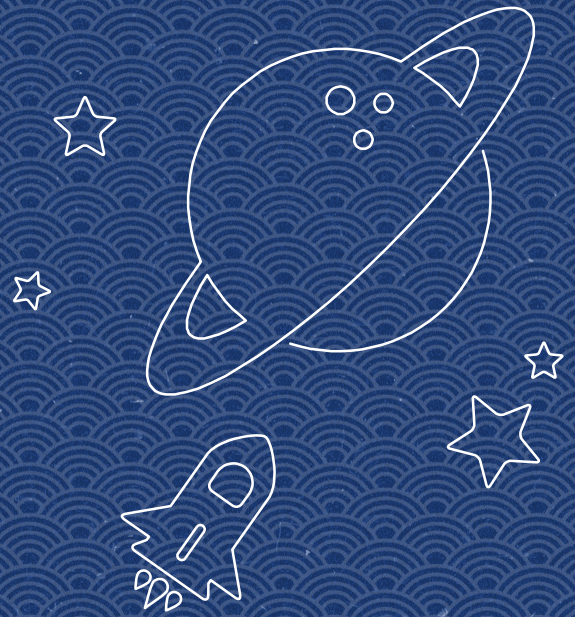
```
# Using Cvec & MNB
gs_4.score(X_train, y_train)
```

```
0.9849157054125999
```

```
# Using Cvec & MNB
gs_4.score(X_test, y_test)
```

```
0.8537234042553191
```

- Countvectorizer combined with logistic regression gave a overfitting test result!



Conclusion

- Tf-idf vectorizer together with Multinomial NB works well as a model
  - Good precision rate for both subreddit categories

Points to note:

- Model might not work well when there are more than 2 subreddit groups