Project 3: Reddit NLP Multi-Label Classification

Zachary Katsnelson
General Assembly, DSI-10, Toronto
Jan.28 2020

Reddit Data: Defining the Problem

What is the problem?

Big Tech comprises the largest and most valuable companies in the world. Also known as the S&P 5, they are:

1. **Facebook** (r/facebook)
2. **Apple** (r/apple)
3. **Amazon** (r/amazon)
4. **Google** (r/google)
5. **Microsoft** (r/microsoft)

We want to be able to correctly classify when a comment or post in reddit belongs to either companies subreddit.

How are we going to try to solve it?

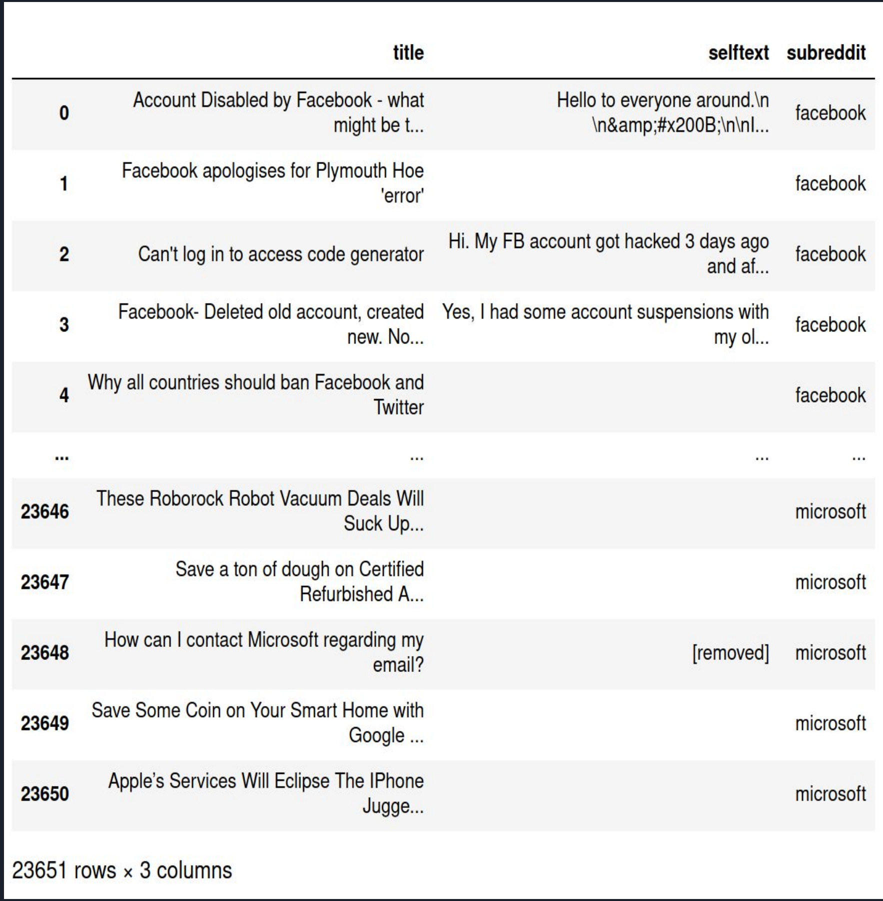
1. Successfully use the Pushshift API to collect and store text data from each subreddit.
2. Clean, transform, and analyze this data for NLP.
3. Pass data into a machine learning model for multi-label classification.
4. Optimize model structure and hyperparameters for performance metrics (accuracy, precision, recall).

Step 1:
Data Collection

Using the pushift.io Reddit API I downloaded:

- 23651 submissions. (~4500 each from each company subreddit).
- Used pd.concat() to join all submissions into one dataframe for cleaning.
- Dropped any duplicates.
- fillna.replace("[removed]").

Each row corresponds to ['title', 'selftext', and 'subreddit'].



	title	selftext	subreddit
0	Account Disabled by Facebook - what might be t...	Hello to everyone around.\n \n&#x200B;\n\nl...	facebook
1	Facebook apologises for Plymouth Hoe 'error'		facebook
2	Can't log in to access code generator	Hi. My FB account got hacked 3 days ago and af...	facebook
3	Facebook- Deleted old account, created new. No...	Yes, I had some account suspensions with my ol...	facebook
4	Why all countries should ban Facebook and Twitter		facebook
...	...	...	...
23646	These Roborock Robot Vacuum Deals Will Suck Up...		microsoft
23647	Save a ton of dough on Certified Refurbished A...		microsoft
23648	How can I contact Microsoft regarding my email?	[removed]	microsoft
23649	Save Some Coin on Your Smart Home with Google ...		microsoft
23650	Apple's Services Will Eclipse The IPhone Jugge...		microsoft

23651 rows × 3 columns

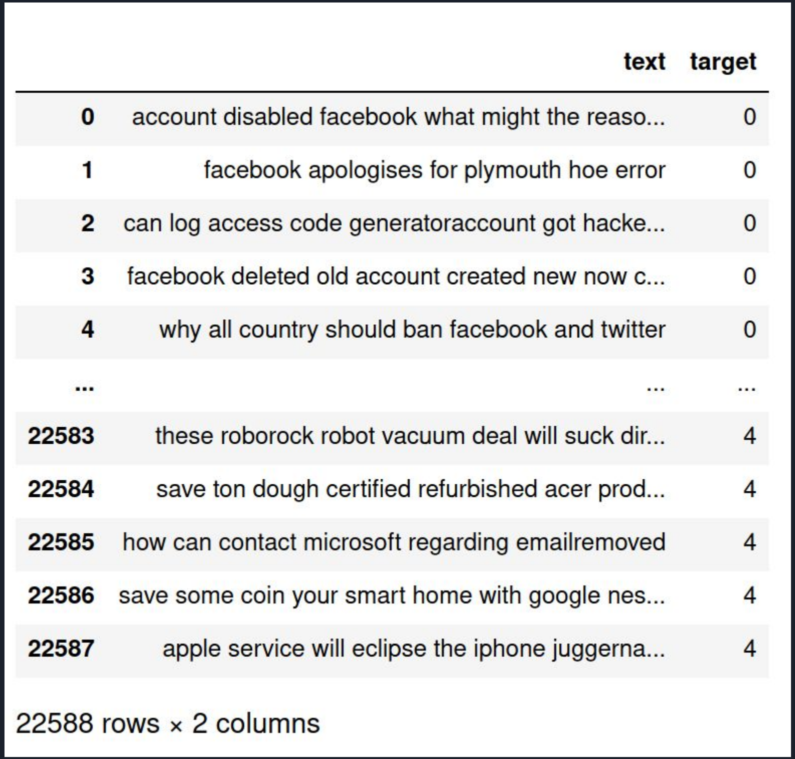
Step 2: Cleaning Data

Various processes were done in order to ensure a clean and passable dataset for our model:

- Cleaning:
  - Use redditcleaner() module to clean data. (This automatically removes superscripts, tables, newlines, code, and any other textual data that will not be helpful for our model).
  - Remove any text under 2 letters.
  - Remove any numbers.

Step 2: (cont'd) Tokenization and Lemmatization

- Use WordNetLemmatizer() to lemmatize all words.
  [googling -> google]
- Machine Learning Preparation:
  - Map all subreddits to binary values. (i.e. 'facebook' : 0, 'apple' : 1, …)
  - Combine title and selftext into one column called 'textdata'.



	text	target
0	account disabled facebook what might the reaso...	0
1	facebook apologises for plymouth hoe error	0
2	can log access code generatoraccount got hacke...	0
3	facebook deleted old account created new now c...	0
4	why all country should ban facebook and twitter	0
...	...	...
22583	these roborock robot vacuum deal will suck dir...	4
22584	save ton dough certified refurbished acer prod...	4
22585	how can contact microsoft regarding emailremoved	4
22586	save some coin your smart home with google nes...	4
22587	apple service will eclipse the iphone juggerna...	4

22588 rows × 2 columns

Step 3: Model Selection

Baseline Model = 0.213% accuracy.

Plan:

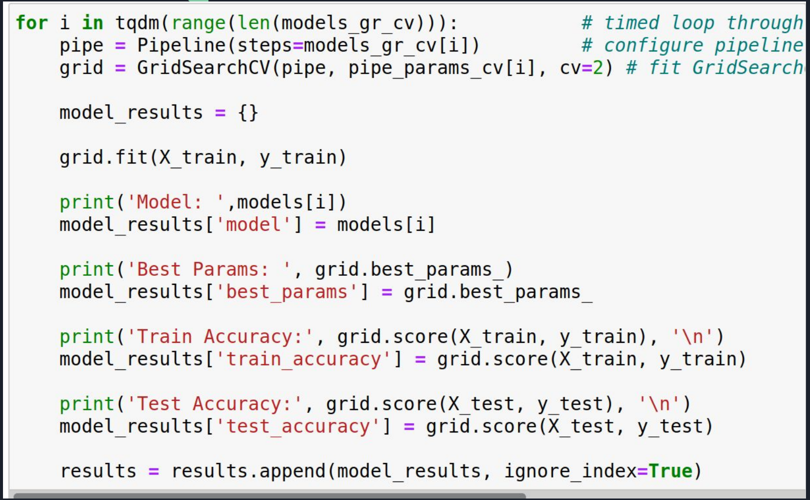
Train_test_split our X and y variables. Add [‘removed’] to **stop_words**.

Using **GridSearchCV** we will then optimize over a wide range of Pipelines containing CountVectorizer and base hyperparameters. We will then select the 3 or 4 most promising models for further optimization.

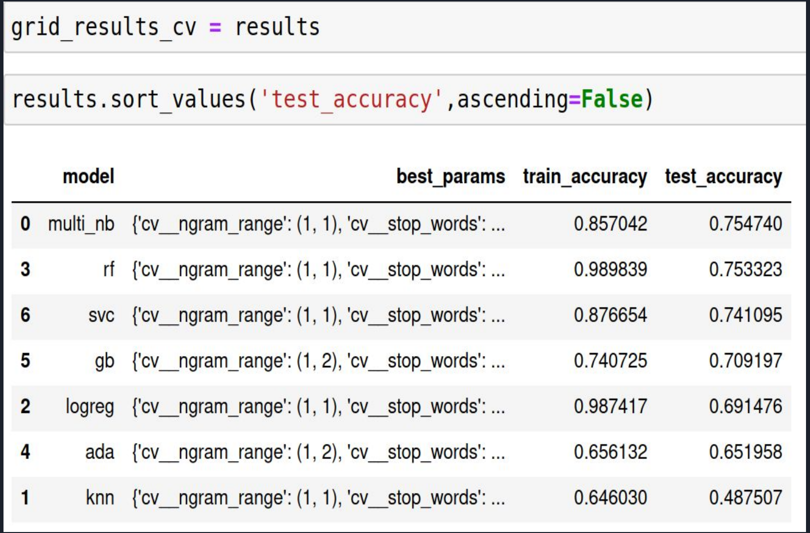
<u>Pipelines we will be training(each with CountVectorizer(stop_words) = ‘english’):</u>

1. MultiNomial Naive Bayes
2. KNN_Classifier with StandardScaler
3. Logistic Regression with StandardScaler
4. RandomForestClassifier
5. AdaBoostClassifier
6. GradientBoostingClassifier
7. Support Vector Machine

Step 3: GridSearchCV Results



```
for i in tqdm(range(len(models_gr_cv))):              # timed loop through
    pipe = Pipeline(steps=models_gr_cv[i])            # configure pipeline
    grid = GridSearchCV(pipe, pipe_params_cv[i], cv=2) # fit GridSearch

    model_results = {}

    grid.fit(X_train, y_train)

    print('Model: ',models[i])
    model_results['model'] = models[i]

    print('Best Params: ', grid.best_params_)
    model_results['best_params'] = grid.best_params_

    print('Train Accuracy:', grid.score(X_train, y_train), '\n')
    model_results['train_accuracy'] = grid.score(X_train, y_train)

    print('Test Accuracy:', grid.score(X_test, y_test), '\n')
    model_results['test_accuracy'] = grid.score(X_test, y_test)

    results = results.append(model_results, ignore_index=True)
```



```
grid_results_cv = results
```

```
results.sort_values('test_accuracy',ascending=False)
```

	model	best_params	train_accuracy	test_accuracy
0	multi_nb	{'cv__ngram_range': (1, 1), 'cv__stop_words': ...	0.857042	0.754740
3	rf	{'cv__ngram_range': (1, 1), 'cv__stop_words': ...	0.989839	0.753323
6	svc	{'cv__ngram_range': (1, 1), 'cv__stop_words': ...	0.876654	0.741095
5	gb	{'cv__ngram_range': (1, 2), 'cv__stop_words': ...	0.740725	0.709197
2	logreg	{'cv__ngram_range': (1, 1), 'cv__stop_words': ...	0.987417	0.691476
4	ada	{'cv__ngram_range': (1, 2), 'cv__stop_words': ...	0.656132	0.651958
1	knn	{'cv__ngram_range': (1, 1), 'cv__stop_words': ...	0.646030	0.487507

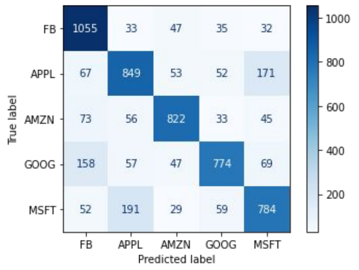
Observations: Most promising models seem to be Multi Naive Bayes, Random Forest, and SVC. We can now optimize these further using GridSearchCV over their respective hyperparameters.

Step 4: Model Optimization

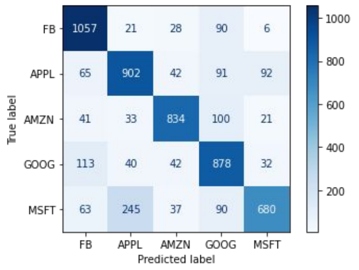
Model	Train Accuracy	Test Accuracy	Best Parameters
MultiNomial_ NB	0.88	0.7591	alpha = 0.1
Random Forest	0.98	0.77	min_samples_split =5 max_depth = None n_estimators = 200
Support Vector Machine	0.93	0.757	C = 0.1 Degree = 2 Kernel = linear

Confusion Matrices:

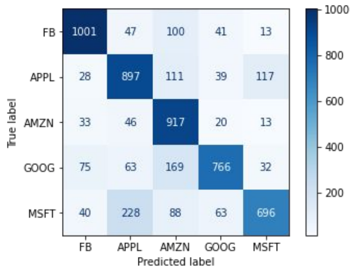
MultiNomial NB



Random Forest



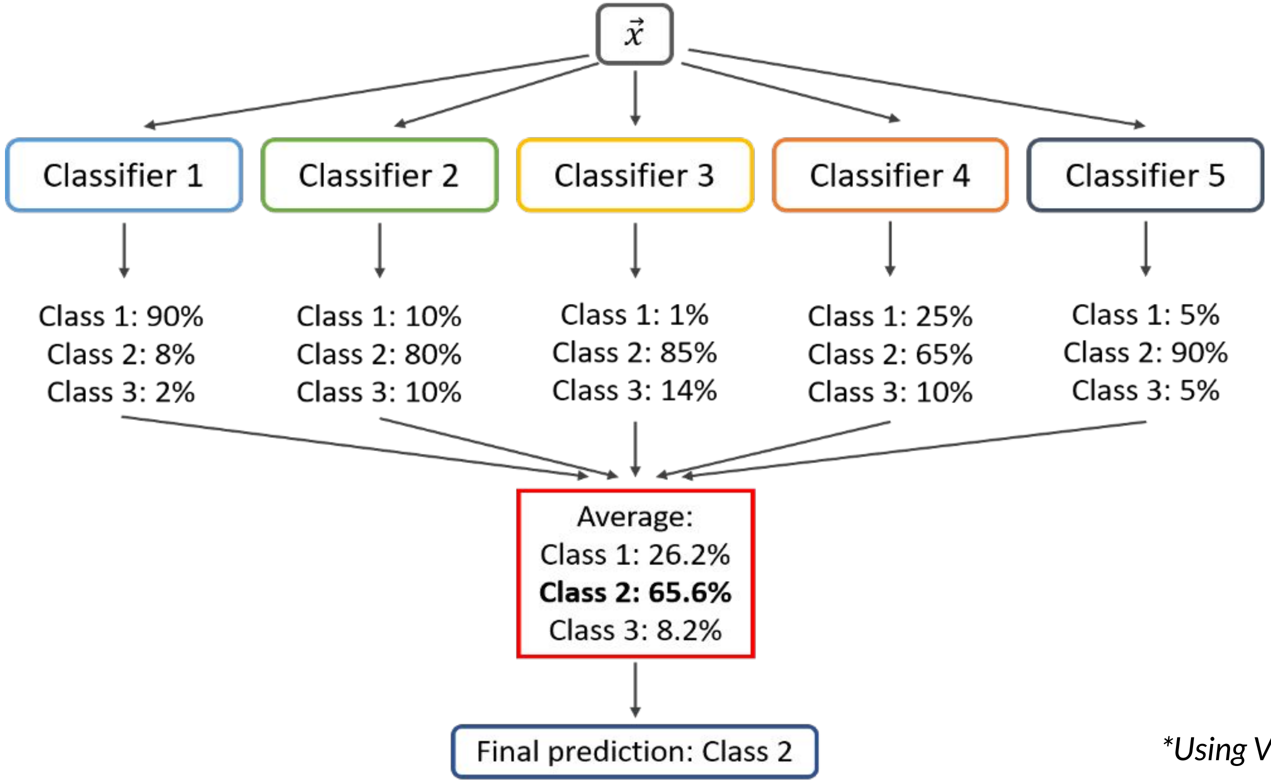
Support Vector Machine



Because different models have different confusion matrices (some models have higher precision, others specificity, etc.) it makes sense to combine all of our models!

Final Model: Stacked VotingClassifier

Train Accuracy: 0.98% Test Accuracy: **0.795%**



**Using Voting = ‘soft’*

Conclusions

- Using StackedVotingClassifier() we were able to increase test accuracy by nearly **4X** relative to the baseline accuracy.
- Using an ensemble of models usually yields better results than just one.
- Potential Improvements:
  - Collect more training data.
  - Add more stop_words in preprocessing.
  - Feature engineering (especially with Apple/Microsoft, where overlapping creates noise).