Project 3 - Web APIs & Classification

Dominic Ong
DSI 16

Agenda

- Problem Statement
- Web Scraping
- Pre-processing of Data
- EDA
- Modelling
- Conclusions & Recommendations

Problem Statement

1. Career paths like **Software Engineering** and **Data Science** have become the most in-demand fields for many people in this rapidly changing world.

2. Reddit has become a popular avenue for people all over the world to ask one another about different career prospects and experiences.

3. Being part of the Data Science Marketing Team of a coding bootcamp, I am interested in finding out what **popular topics** and **keyword jargons** belong to the fields of Data Science and Software Engineering.

4. Conducting analysis on Reddit posts will allow me to craft online content and advertisements to better target people interested in Data Science or Software Engineering.

Developing a Scraping Function

```
# Scraping Software Engineering Subreddit Data

swe_posts=[] # Storing Software Engineering Subreddit Posts
after = None # after will be empty for the first iteration
swe_url = 'https://www.reddit.com/r/softwareengineering.json'

for a in range(40): #range number will indicate how many pages of 25 posts to scrape
    if after == None:
        current_url = swe_url
    else:
        current_url = swe_url + '?after=' + after
    print(current_url)

    res = requests.get(current_url, headers={'User-agent': 'GA Boy'})

    if res.status_code == 200: #to check if request is successful
        current_dict = res.json()
        current_posts = [p['data'] for p in current_dict['data']['children']]
        swe_posts.extend(current_posts) # Adding all posts scraped into this list
        after = current_dict['data']['after']
    else:
        print(results.status_code)
        break

    # generate a random sleep duration to look more 'natural'
    sleep_duration = random.randint(2,6)
    print(sleep_duration)
    time.sleep(sleep_duration)
    time.sleep(1) #seconds to sleep
```

```
# Converting these scrapped posts into a dataframe
swe_df = pd.DataFrame(swe_posts)
swe_df.head()
```

	approved_at_utc	subreddit	selftext	author_fullname	saved	mod_reason_title	gilded	clicked	title	link_flair_richtext
0	None	SoftwareEngineering	Hi /r/SoftwareEngineering - This is just to le...	t2_6w5wq	False	None	0	False	Style changes and automoderator	[]
1	None	SoftwareEngineering	I have been at several large companies now and...	t2_59dwm06	False	None	0	False	How do your companies keep track of multiple s...	[]
2	None	SoftwareEngineering	I am doing a Uni course on Data Engineering, w...	t2_57j4x5fp	False	None	0	False	How to learn programming fundamentals?	[]
3	None	SoftwareEngineering		t2_ayl6m	False	None	0	False	On house building and software development pro...	[]
4	None	SoftwareEngineering	I've been learning to code using Udemy and thi...	t2_60gqusa6	False	None	0	False	Learning to code	[]

After removing duplicate rows: 721 Software Engineering Posts & 562 Data Science Posts

Total # of Posts: 1,283 Posts

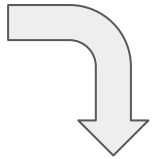
Pre-processing of Data

Feature Engineering

Text Processing

CountVectorizer

```
# Combining selftext and title columns
ds_final['text'] = ds_final['title'] + ' ' + ds_final['selftext']

# Creation of target variable
ds_final['ds'] = 1

# Removing original columns that we have feature engineered
ds_final.drop(['selftext','title','subreddit'], axis=1, inplace=True)
```



Removing Columns and Merging Dataframes

	upvote_ratio	ups	num_comments	subreddit_subscribers	text	ds
216	0.67	3	16	24671	Input on Database Solution for small web app I...	0
1219	1.00	3	7	285444	Senior role in outsourcing company? I may have...	1
712	0.67	5	7	24671	Code Reviews: Honey Trap for Pedants? Or Power...	0
503	0.95	14	23	24671	Have got a decent entry level/graduate offer a...	0
1084	0.67	2	10	285444	On Calculating Certainty I have been making mo...	1

Pre-processing of Data

Feature Engineering | **Text Processing** | CountVectorizer

- Tokenizing
- Regex
- Lemmatizing
- Stemming
- Remove (stopwords and https)

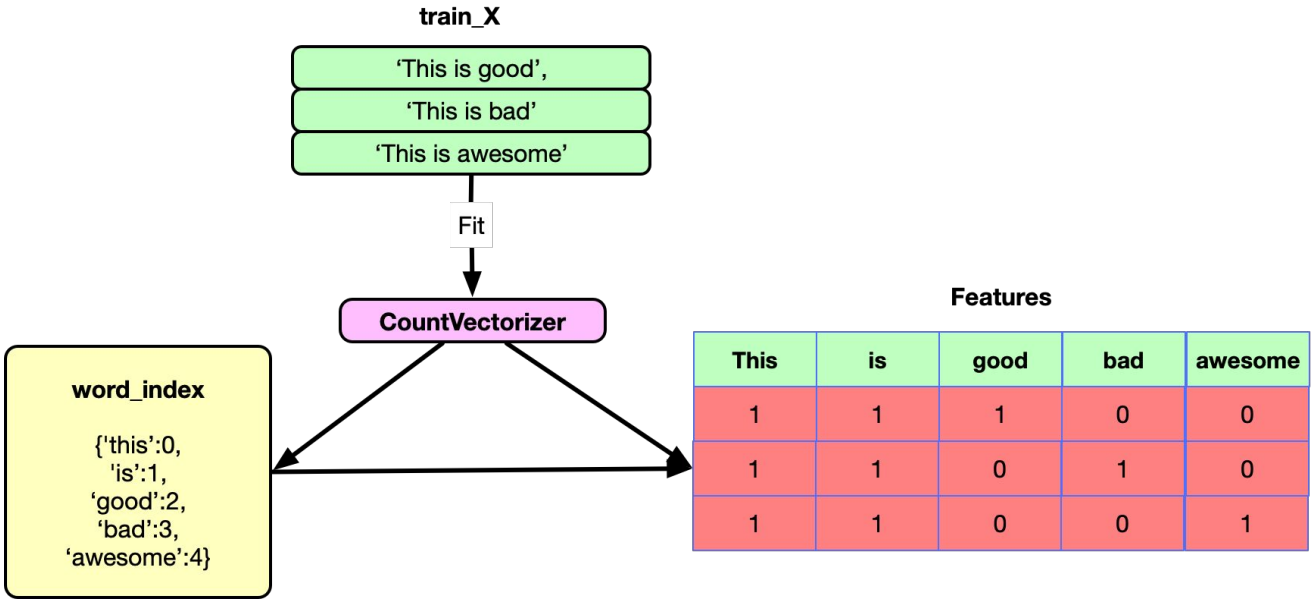
```
# Defining a Function to perform Text Processing

def text_processing(raw_text):

    leakage_words = ['software', 'engineering', 'data', 'science', 'http', 'www']

    # Step 1: Remove HTML
    review_text = BeautifulSoup(raw_text).get_text()

    # Step 2: Tokenizing - Remove punctuation and convert all text to lowercase
    tokenizer = RegexpTokenizer(r'\w+')
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    # Convert all text to lowercase
    words = letters_only.lower().split()

    # Step 3: Remove stopwords and leakage words
    words = [word for word in words if word not in stopwords.words('english')+leakage_words]

    # Step 4: Stem or lemmatize each word of the text,
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    stemmer = PorterStemmer()
    words = [stemmer.stem(word) for word in words]

    # Step 5: Remove any additional stopwords and leakage words that might not have been removed in Step 3
    words = [word for word in words if word not in stopwords.words('english')+leakage_words]

    # Return the final string
    return ' '.join(words)
```


Pre-processing of Data

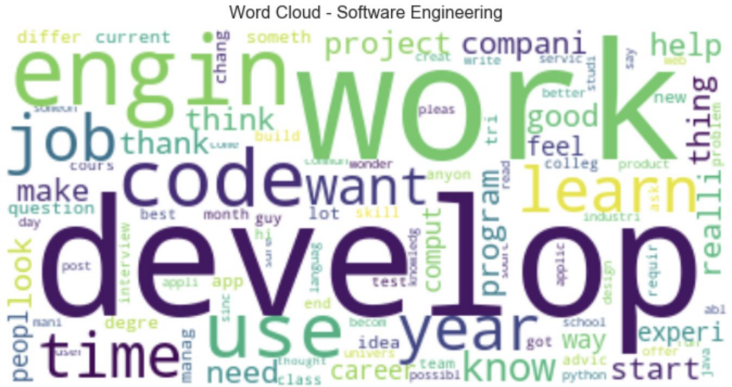
Feature Engineering

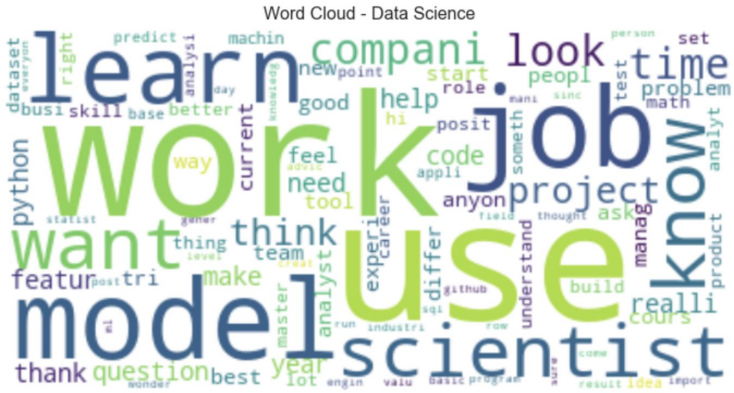
Text Processing

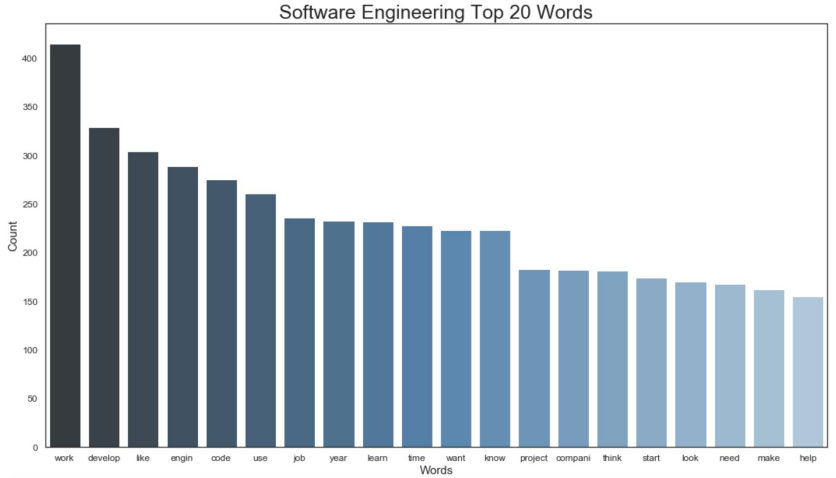
CountVectorizer

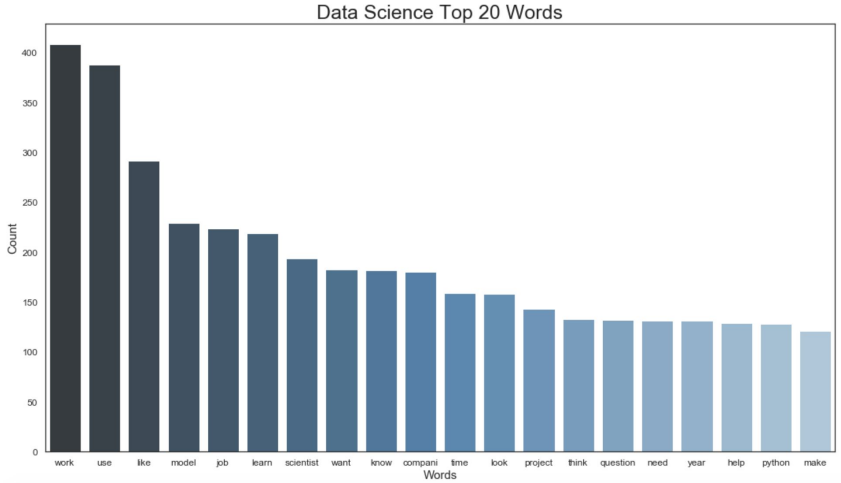


Exploratory Data Analysis

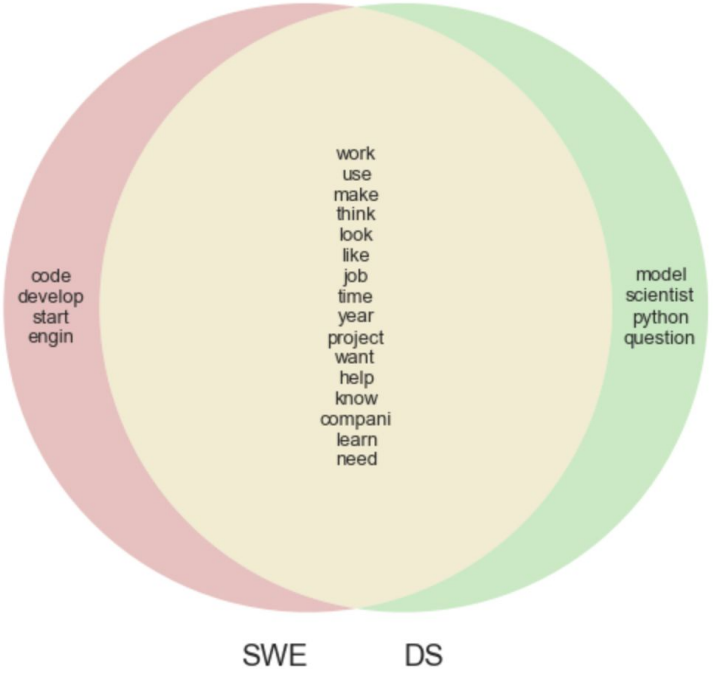


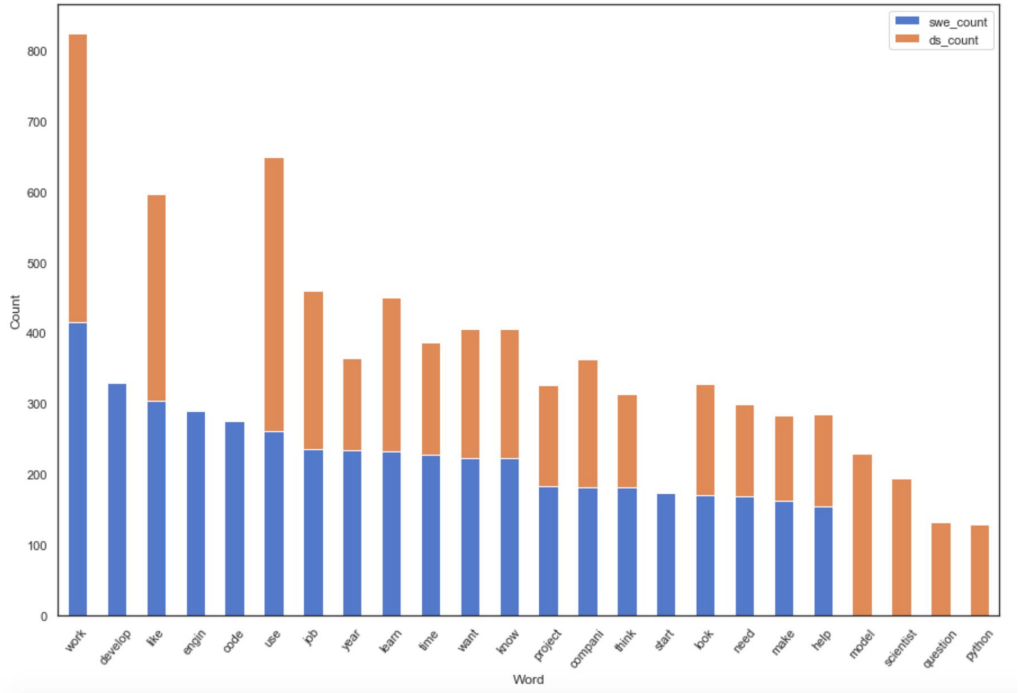






Exploratory Data Analysis





Modelling - Results

	Model	Vectorizer	Training Score	Testing Score	TN	FN	TP	FP	Sensitivity	Specificity
0	Naive-Bayes	Count Vectorizer	0.951	0.888	167	23	118	13	0.84	0.93
1	Naive-Bayes	TFIDF Vectorizer	0.953	0.835	170	43	98	10	0.70	0.94
2	Logistics Regression	Count Vectorizer	0.971	0.869	162	24	117	18	0.83	0.90
3	Logistic Regression	TFIDF Vectorizer	0.960	0.872	173	34	107	7	0.76	0.96
4	Random Forest	Count Vectorizer	1.000	0.857	167	33	108	13	0.77	0.93
5	Random Forest	TFIDF Vectorizer	1.000	0.866	170	33	108	10	0.77	0.94
6	KNeighbors	Count Vectorizer	1.000	0.617	146	89	52	34	0.37	0.81
7	KNeighbors	TFIDF Vectorizer	0.859	0.791	164	51	90	16	0.64	0.91
8	SVM	Count Vectorizer	0.956	0.826	168	44	97	12	0.69	0.93
9	SVM	TFIDF Vectorizer	0.982	0.875	168	28	113	12	0.80	0.93

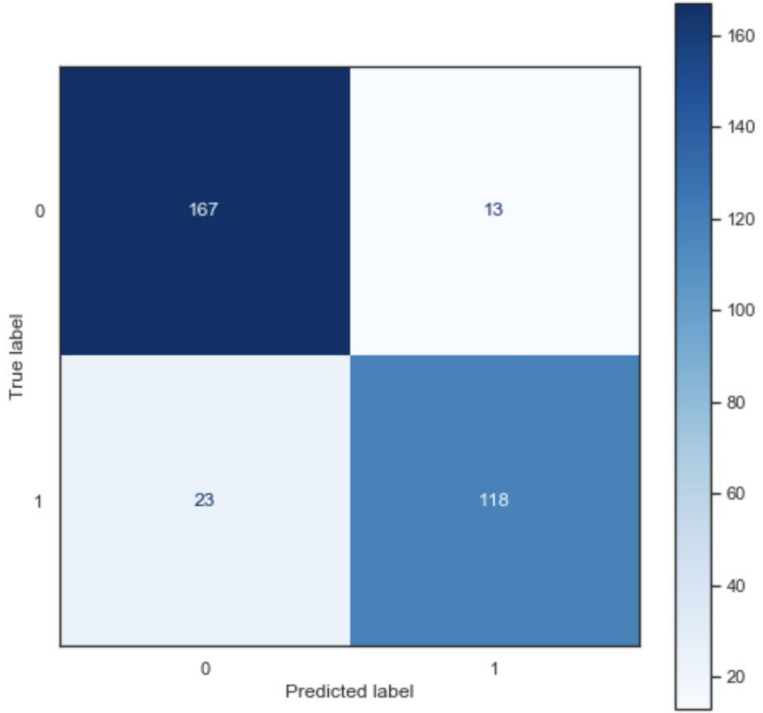
Modelling - Evaluation

After running our data through 10 possible model combinations, I managed to achieve the **highest accuracy (testing) score** using the `MultinomialNB` - `CvecVectorizer` model combination. The model scores are as follows:

- **Testing Score:** 88.8%
- **Specificity**: 92.8%
- **Precision**: 83.7%
- **Misclassification Rate**: 11.2%
- **ROC/AUC Score**: 93.8%
- **Type I Error (FP):** 13
- **Type II Error (FN):** 23

Using GridSearchCV, the **best hyperparameters** for this particular model is:

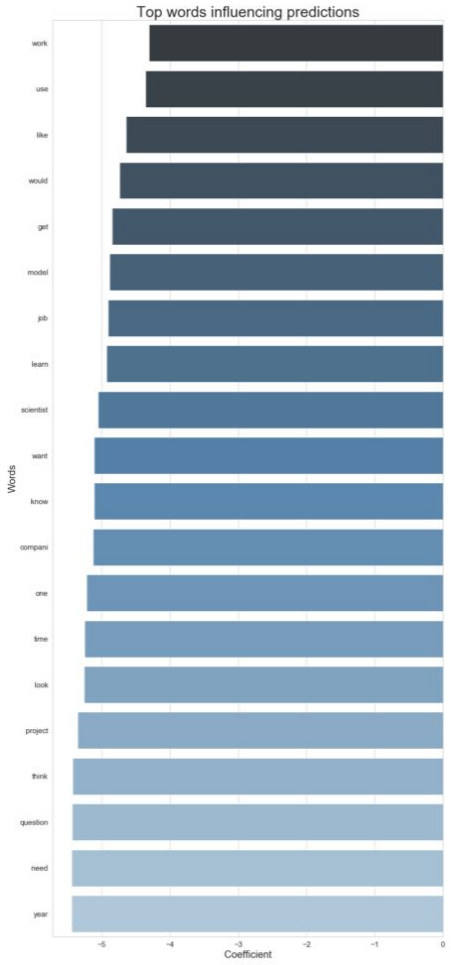
- 'cvec__max_df': 0.6,
- 'cvec__max_features': 2000,
- 'cvec__min_df': 2,
- 'cvec__ngram_range': (1, 1),
- 'nb__alpha': 0.25



Coefficient Analysis

- The **coef_ attribute** of MultinomialNB is a re-parameterization of the Naive Bayes model as a linear classifier model.
- For a binary classification problems, this is basically the log of the estimated probability of a feature given the positive class.
- It means that higher values mean more important features for the positive class.
- Based on our observations, we can infer that words like work, use, and like can be represented as important features to the positive class (Data Science subreddit).

	feature	coef
1974	work	-4.304841
1891	use	-4.357518
973	like	-4.641557
1983	would	-4.738494
724	get	-4.845846
1089	model	-4.884354
902	job	-4.906405
951	learn	-4.928954
1526	scientist	-5.050021
1932	want	-5.108316
923	know	-5.113788
323	compani	-5.124822
1178	one	-5.217777
1795	time	-5.248692
999	look	-5.254991
1345	project	-5.354576
1781	think	-5.426940
1381	question	-5.434473
1124	need	-5.442063
1991	year	-5.442063



Misclassified Posts

Type I Error (13 Posts)

- SWE Posts predicted as DS Posts
- Common words used : **use**, **job**, **work**, **learn**
- DS specific words : **model**

Type II Error (23 Posts)

- DS Posts predicted as SWE Posts
- Common words used : **use**, **job**, **work**, **learn**
- SWE specific words used: **code, develop, learn**

Without conducting further granular analysis into these posts, these are the general assumptions we can make to explain why these posts were misclassified.

Recommendations

- When it comes to modelling, having more rows of data will improve our modelling accuracy scores.

- To increase our data size, one other area where we can scrape more text data is scraping the comments section of Subreddit threads. This can be done using other extensions. (E.g. Pushshift.io)

- To reduce our misclassification rate for our models, we can identify common words that appear within both subreddits that contribute to the misclassification of posts and remove such words during our pre-processing stage and re-run our models again.

Thank You

End of Week 7! :))