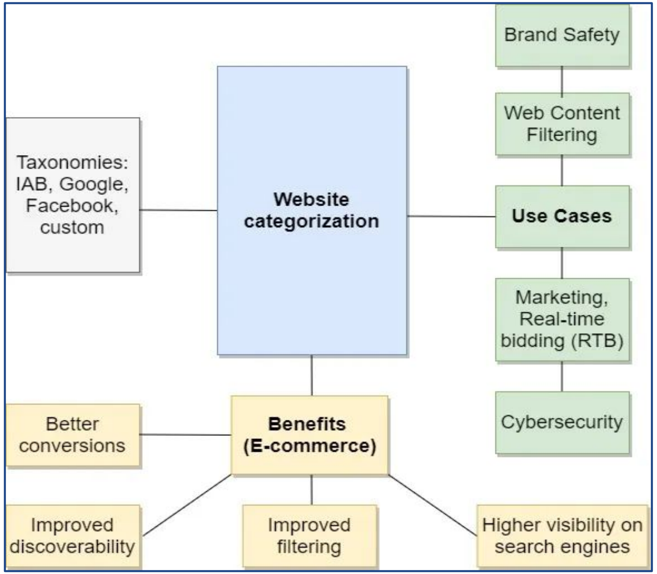
Website Classification

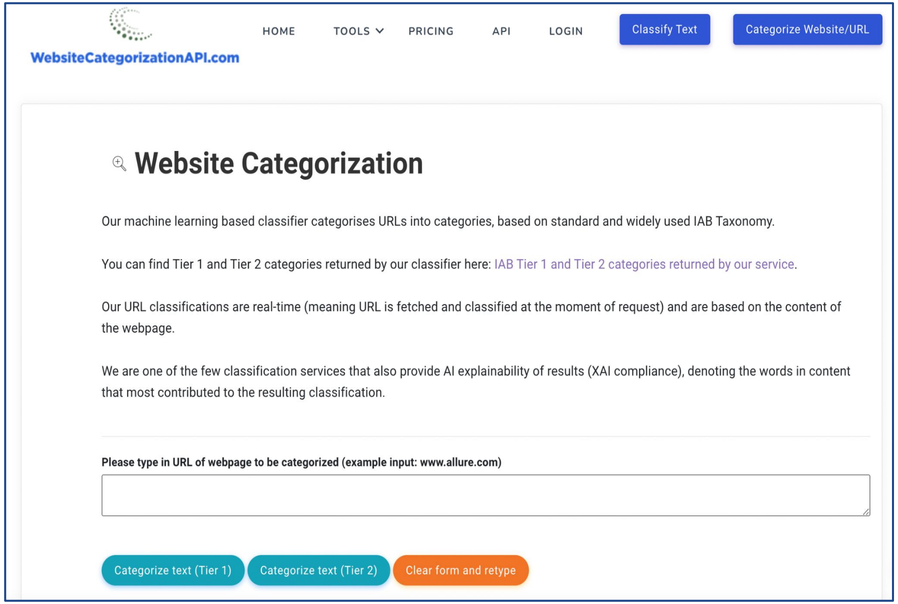
W207 Final Project - Spring 2023
Kusam Brar, Sean Seneviratne & Theresa Azinge

Problem Definition

Use Cases



Sample Categorization Database



Website Categorization

Our machine learning based classifier categorises URLs into categories, based on standard and widely used IAB Taxonomy.

You can find Tier 1 and Tier 2 categories returned by our classifier here: IAB Tier 1 and Tier 2 categories returned by our service.

Our URL classifications are real-time (meaning URL is fetched and classified at the moment of request) and are based on the content of the webpage.

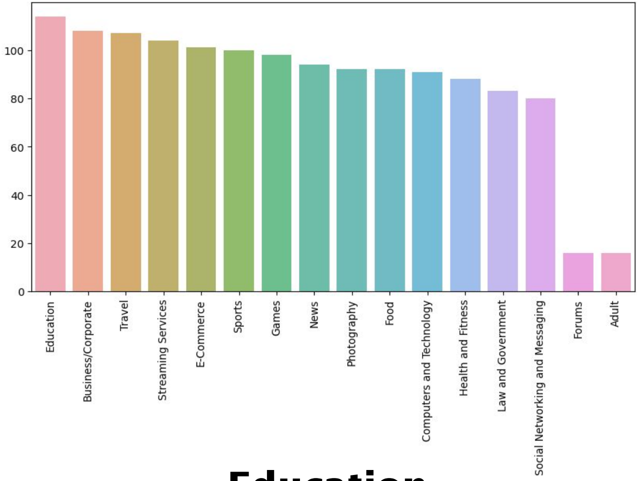
We are one of the few classification services that also provide AI explainability of results (XAI compliance), denoting the words in content that most contributed to the resulting classification.

Please type in URL of webpage to be categorized (example input: www.allure.com)

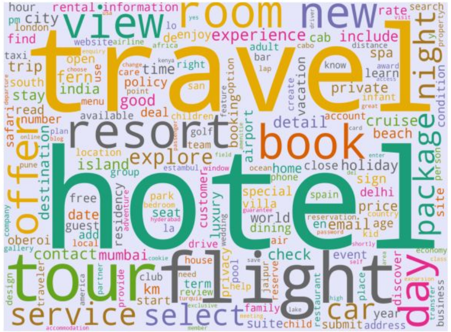
Dataset Description

Dataset consists of **1480** rows of the website url, cleaned website text, and the category of the URL.

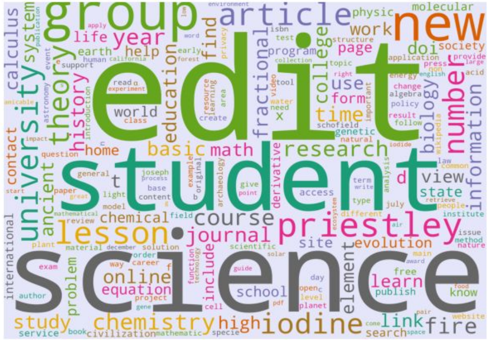
16 categories with Education, Business / Corporate and Travel as the most popular.



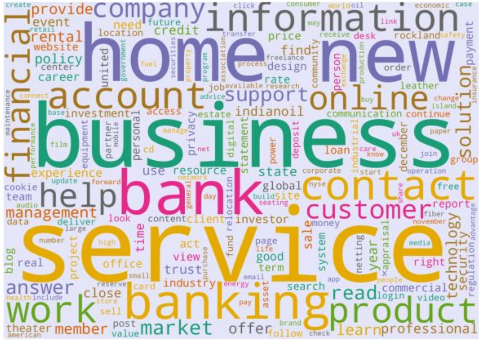
Travel



Education



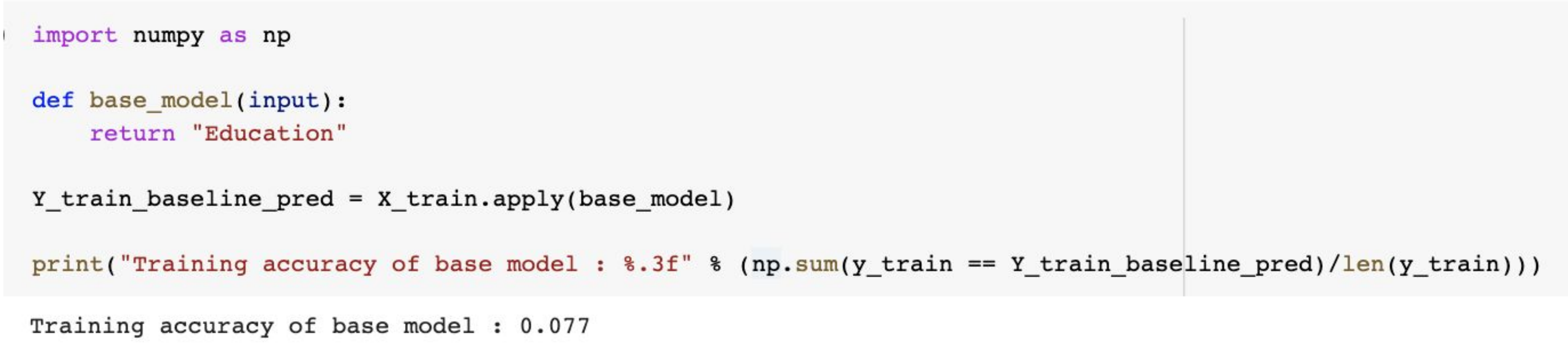
Business / Corporate



Experiments

- Baseline Model
- Bag of Words Model
- Model Analysis
- “Bag of Embeddings” Model
- Model Analysis
- Final Model

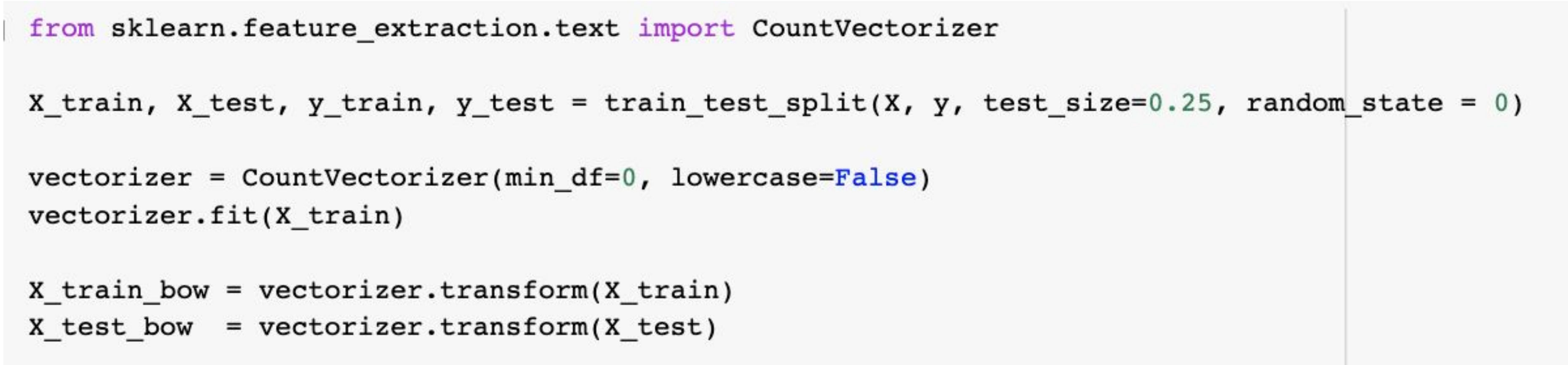
Baseline Model



```
import numpy as np

def base_model(input):
    return "Education"

Y_train_baseline_pred = X_train.apply(base_model)

print("Training accuracy of base model : %.3f" % (np.sum(y_train == Y_train_baseline_pred)/len(y_train)))
```

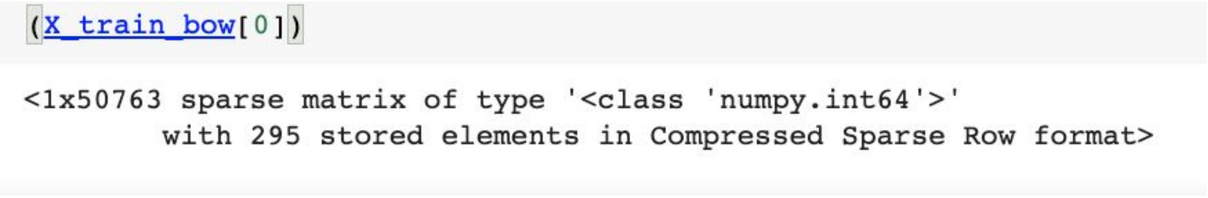
```
Training accuracy of base model : 0.077
```

Bag of Words Model



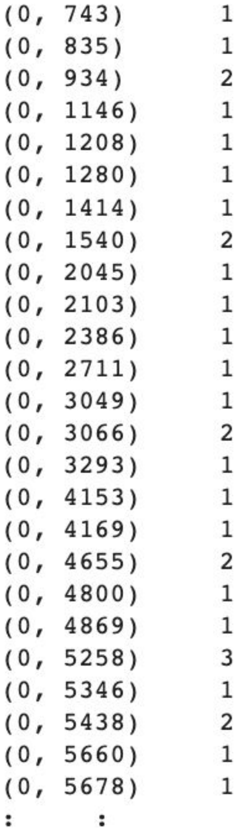
```
from sklearn.feature_extraction.text import CountVectorizer

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 0)

vectorizer = CountVectorizer(min_df=0, lowercase=False)
vectorizer.fit(X_train)

X_train_bow = vectorizer.transform(X_train)
X_test_bow  = vectorizer.transform(X_test)
```

Bag of Words samples



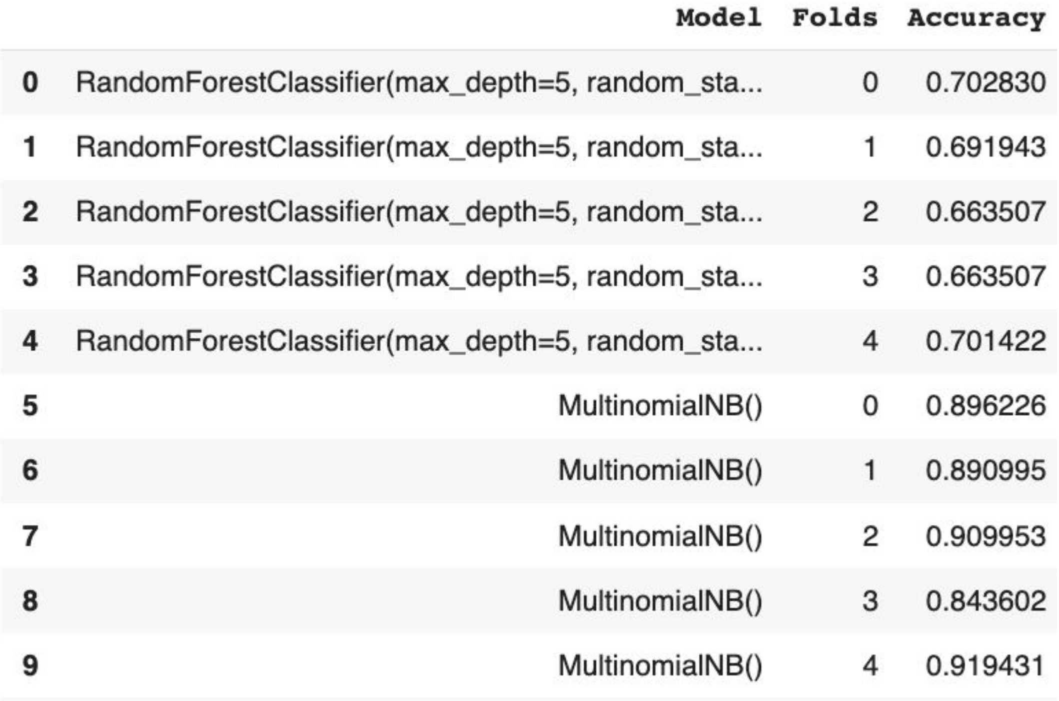
```
(X_train_bow[0])
```

```
<1x50763 sparse matrix of type '<class 'numpy.int64'>'
	with 295 stored elements in Compressed Sparse Row format>
```



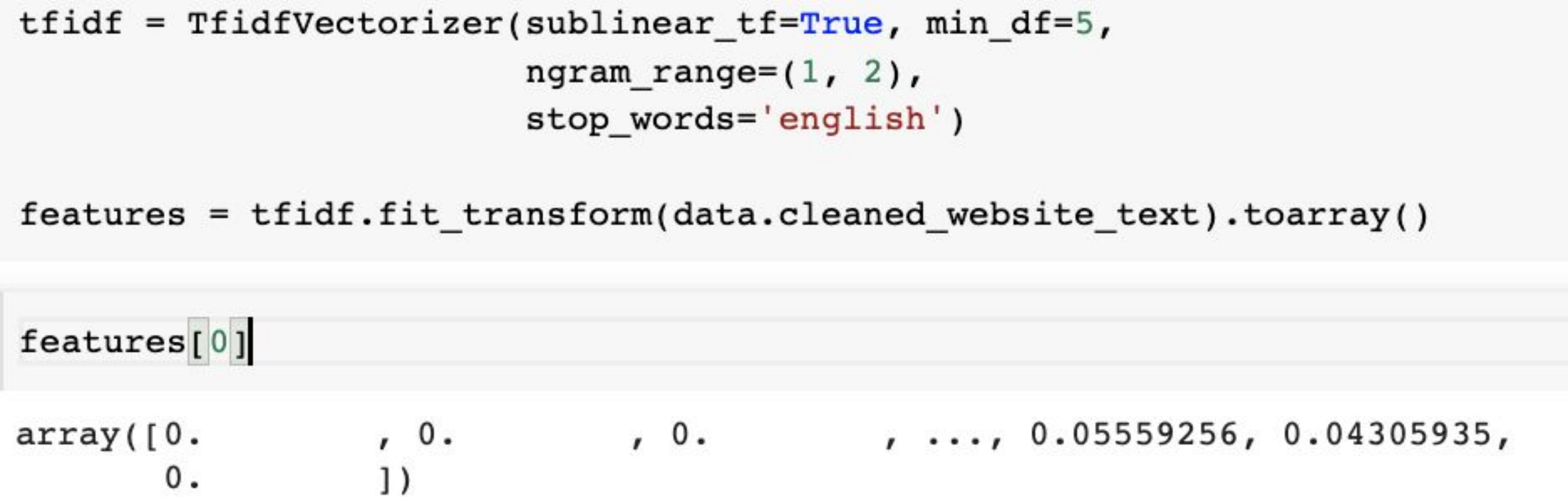
```
(0, 743)	1
(0, 835)	1
(0, 934)	2
(0, 1146)	1
(0, 1208)	1
(0, 1280)	1
(0, 1414)	1
(0, 1540)	2
(0, 2045)	1
(0, 2103)	1
(0, 2386)	1
(0, 2711)	1
(0, 3049)	1
(0, 3066)	2
(0, 3293)	1
(0, 4153)	1
(0, 4169)	1
(0, 4655)	2
(0, 4800)	1
(0, 4869)	1
(0, 5258)	3
(0, 5346)	1
(0, 5438)	2
(0, 5660)	1
(0, 5678)	1
:	:
```

Bag of Words Results



	Model	Folds	Accuracy
0	RandomForestClassifier(max_depth=5, random_sta...	0	0.702830
1	RandomForestClassifier(max_depth=5, random_sta...	1	0.691943
2	RandomForestClassifier(max_depth=5, random_sta...	2	0.663507
3	RandomForestClassifier(max_depth=5, random_sta...	3	0.663507
4	RandomForestClassifier(max_depth=5, random_sta...	4	0.701422
5	MultinomialNB()	0	0.896226
6	MultinomialNB()	1	0.890995
7	MultinomialNB()	2	0.909953
8	MultinomialNB()	3	0.843602
9	MultinomialNB()	4	0.919431

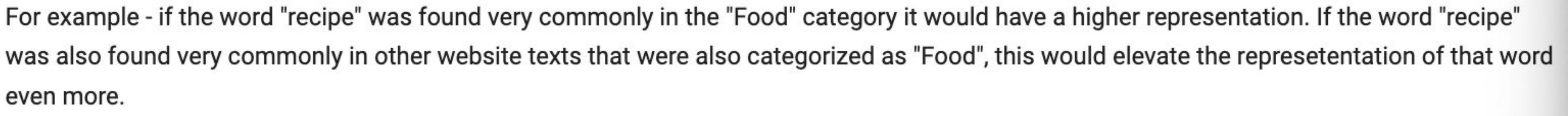
Bag of Embeddings



```
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5,
                        ngram_range=(1, 2),
                        stop_words='english')

features = tfidf.fit_transform(data.cleaned_website_text).toarray()
```

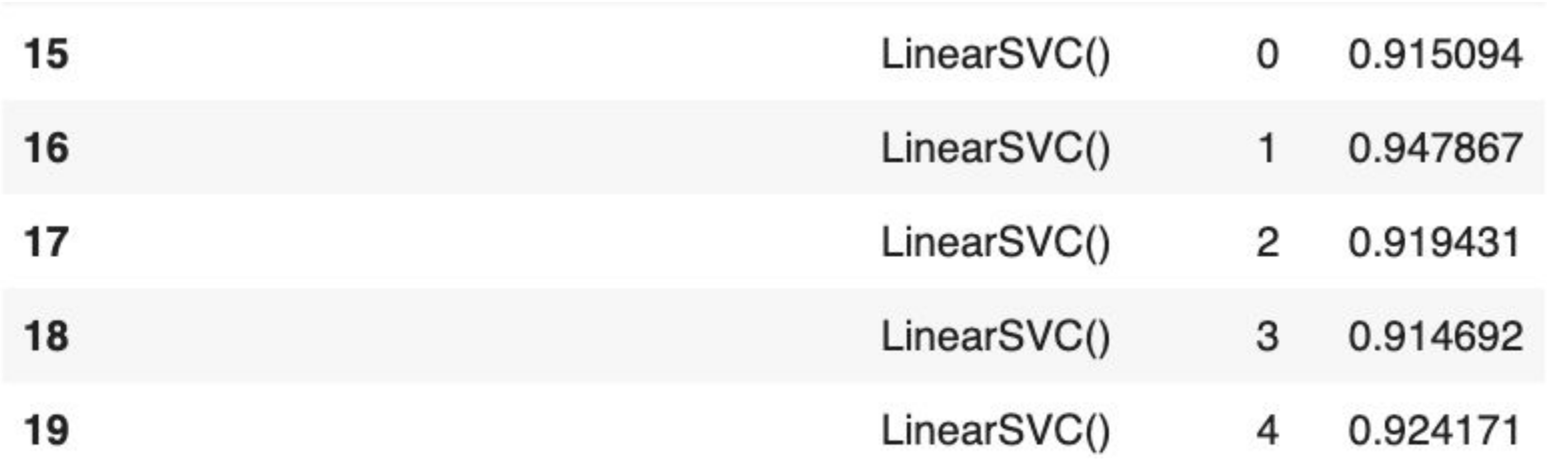
```
features[0]
```

```
array([0.        , 0.        , 0.        , ..., 0.05559256, 0.04305935,
       0.        ])
```



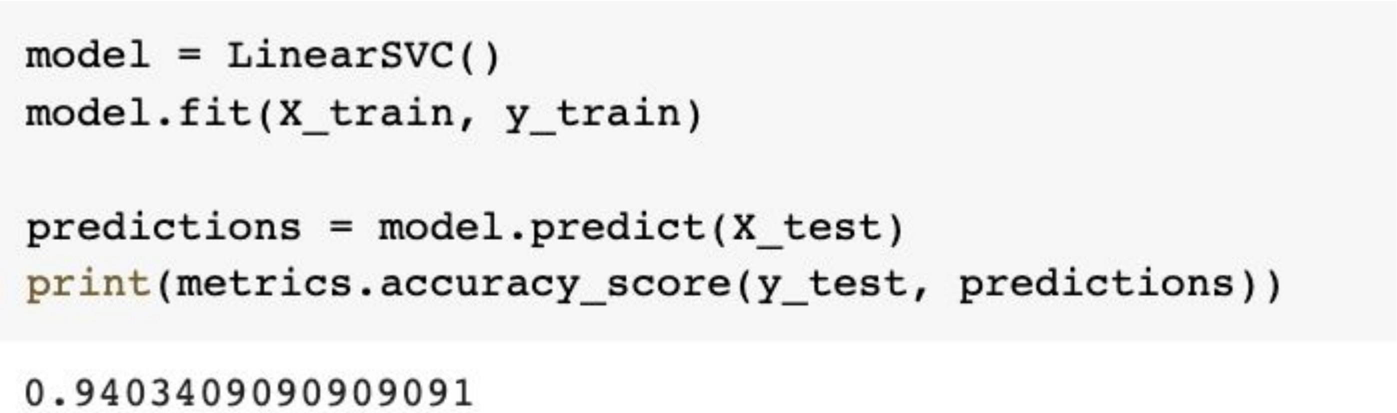
For example - if the word "recipe" was found very commonly in the "Food" category it would have a higher representation. If the word "recipe" was also found very commonly in other website texts that were also categorized as "Food", this would elevate the represetentation of that word even more.

Bag of Embeddings Results



15	LinearSVC()	0	0.915094
16	LinearSVC()	1	0.947867
17	LinearSVC()	2	0.919431
18	LinearSVC()	3	0.914692
19	LinearSVC()	4	0.924171

Final Model Results

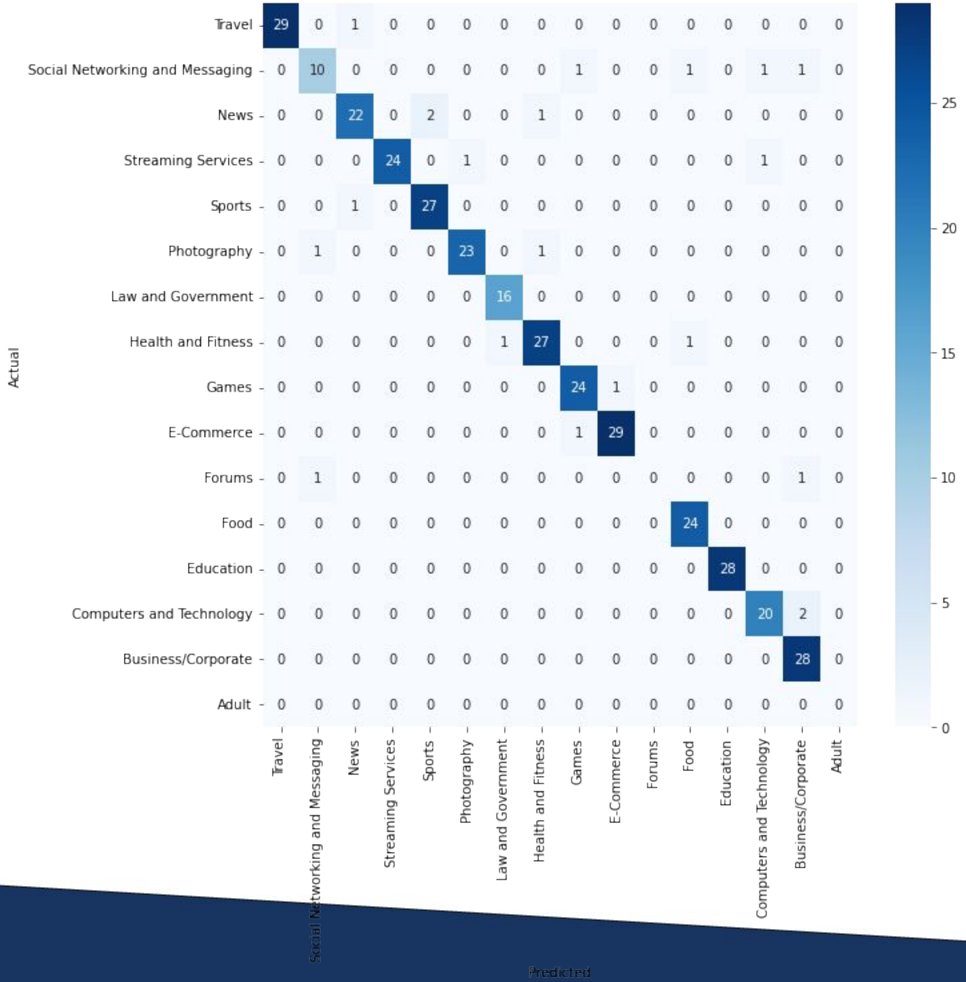


```
model = LinearSVC()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
print(metrics.accuracy_score(y_test, predictions))
```

```
0.9403409090909091
```

Results

Model	**Validation Accuracy**
Baseline Model	**8.3%**
CountVectorizer / MultinomialNB	91.9%
TF-IDF / Linear SVC	94.7%



	Travel	Social Networking and Messaging	News	Streaming Services	Sports	Photography	Law and Government	Health and Fitness	Games	E-Commerce	Forums	Food	Education	Computers and Technology	Business/Corporate	Adult
Travel	29	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Social Networking and Messaging	0	10	0	0	0	0	0	0	1	0	0	1	0	1	1	0
News	0	0	22	0	2	0	0	1	0	0	0	0	0	0	0	0
Streaming Services	0	0	0	24	0	1	0	0	0	0	0	0	0	1	0	0
Sports	0	0	1	0	27	0	0	0	0	0	0	0	0	0	0	0
Photography	0	1	0	0	0	23	0	1	0	0	0	0	0	0	0	0
Law and Government	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
Health and Fitness	0	0	0	0	0	0	1	27	0	0	0	1	0	0	0	0
Games	0	0	0	0	0	0	0	0	24	1	0	0	0	0	0	0
E-Commerce	0	0	0	0	0	0	0	0	1	29	0	0	0	0	0	0
Forums	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Food	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0
Education	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0
Computers and Technology	0	0	0	0	0	0	0	0	0	0	0	0	0	20	2	0
Business/Corporate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0
Adult	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NeurIPS paper checklist

- (a) Do the **main claims** made in the abstract and introduction accurately reflect the paper's contributions and scope?
  - Yes
- (b) Have you read the **ethics review guidelines** and ensured that your paper conforms to them?
  - Yes
- (c) Did you discuss any potential **negative societal impacts** of your work?
  - Limitations of our research
    1. Data may not be reproducible
    2. Non-english datasets

Conclusion

- Limitations with this type of data collection
- Bag of words vs embeddings
- SVC model accuracy