Binary Classification Problem

Classification of Subreddits

→

Outline

-Problem Statement

- Data Collection
- Data Cleaning
- Exploratory Data Analysis
- Model Prep
- Model Fit and Evaluation

-Business Recommendation

Problem Statement

- identify the best classification model
- identify important word features

01

Target 1: /r/keto
Target 0:/r/gainit

Commonalities shared by both:

1. Focus on caloric intake and fitness
2. Macronutrients: high-protein, high-fat diet.

→

Stakeholders

Advertisers- develop marketing strategies specifically targeted at each subreddit

Moderators- flag out misclassified posts to maintain the integrity of the posts

SUMMER

SALE

ENDS 15.09.20

ACHIEVE YOUR HEALTH GOALS!

Data Collection

- Reddit API: Provides 25 posts per request
- 20 iterations

- list of nested json dictionaries is obtained of which are saved in 2 separate csv files.

Cleaning

- Approximately half of the dataset are found to be duplicates and they are subsequently dropped.

- 2 null values filled with arbitrary text.

- balance of classes of ~ 50% for both target class 1 and 0.

→

Cleaning

- Removing text that corresponds to custom regex patterns that includes one that identifies url

- To prevent data leakage, the subreddit topic is also removed.

- After obtaining only letters from the previous steps, the text is converted to lowercase and split into individual words.

- Stopwords are removed

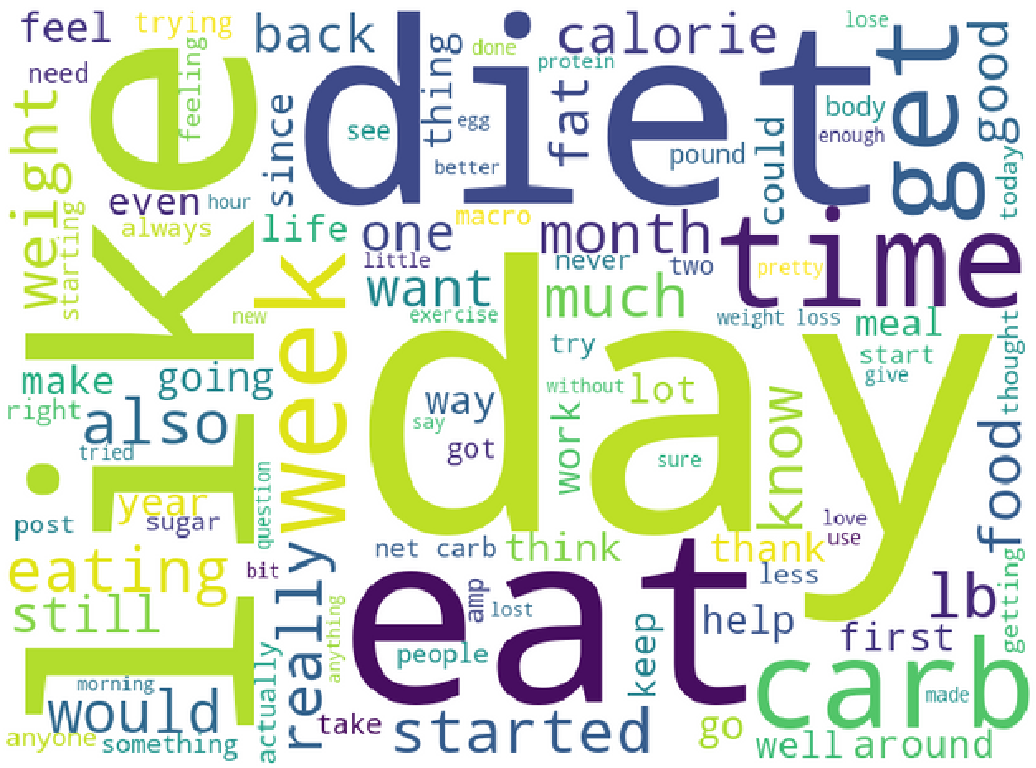
→

03

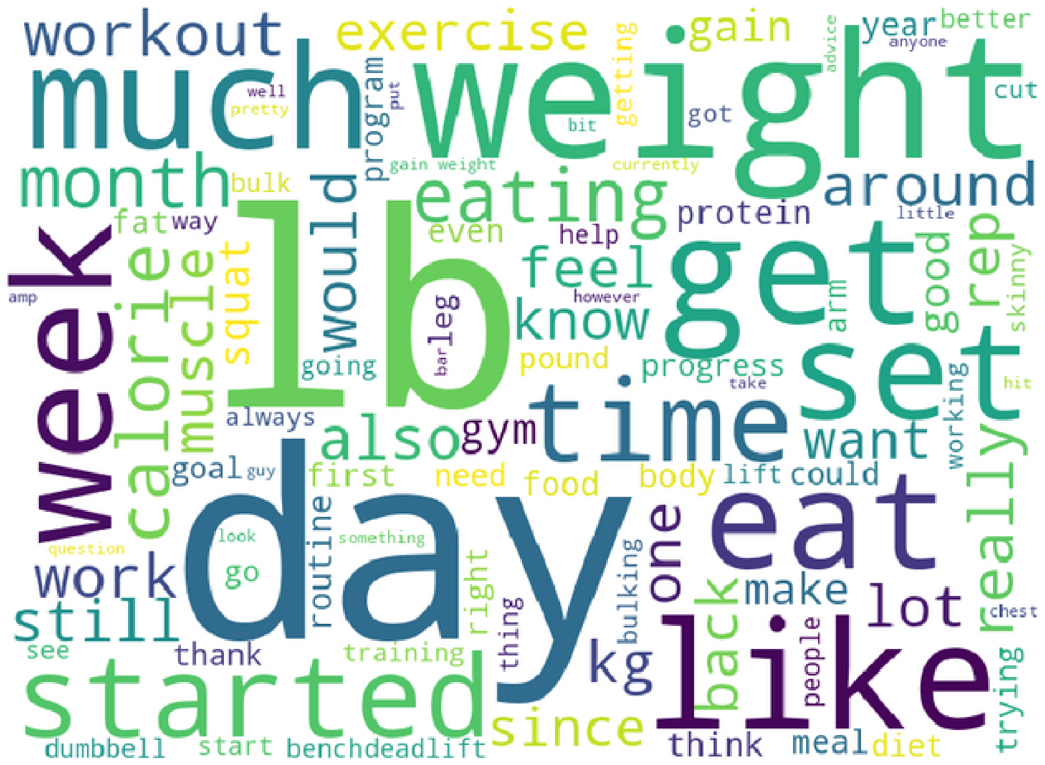
Exploratory Data Analysis

Wordcloud, unigram, bi-gram

Wordcloud



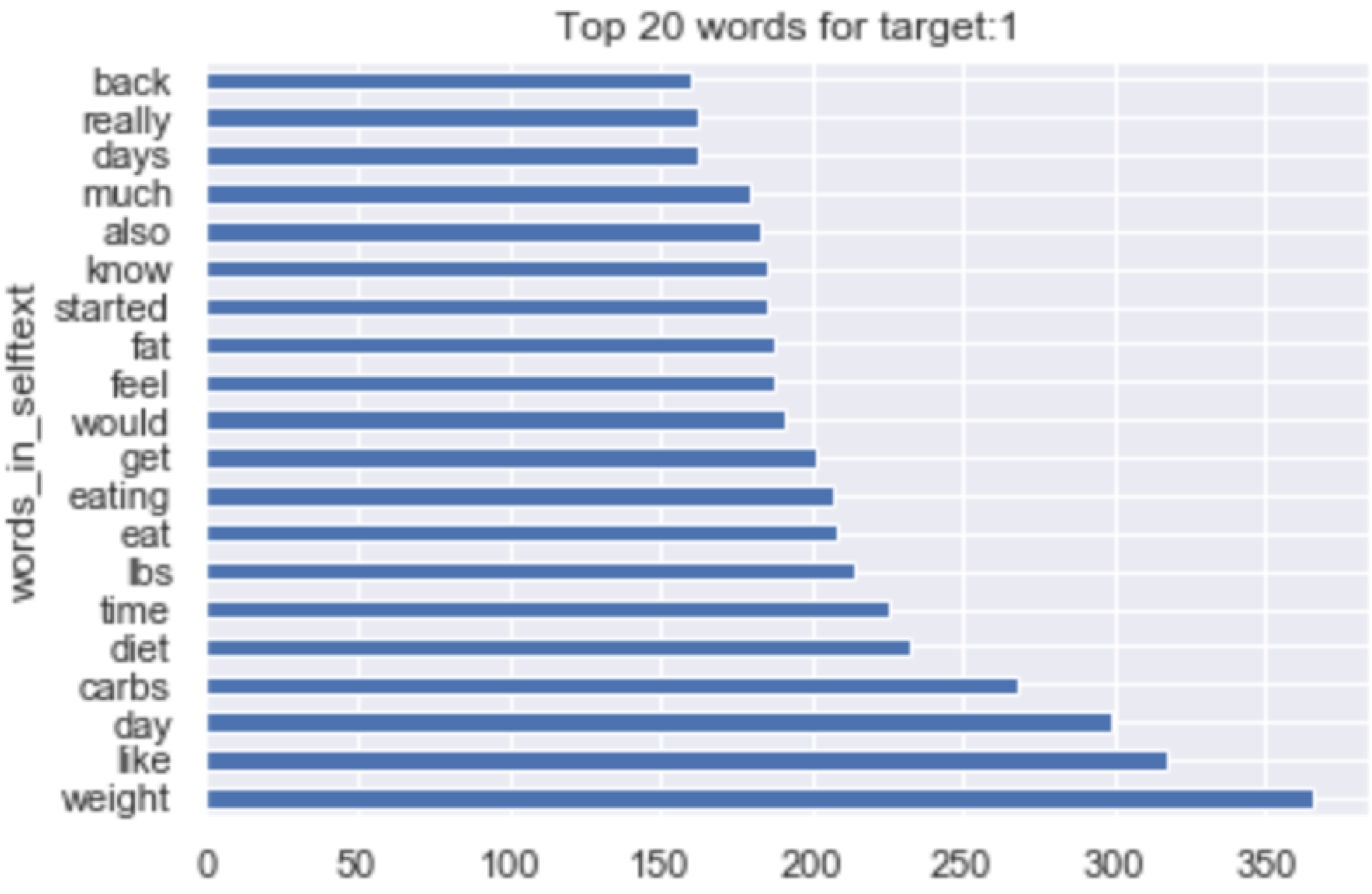
Target 1



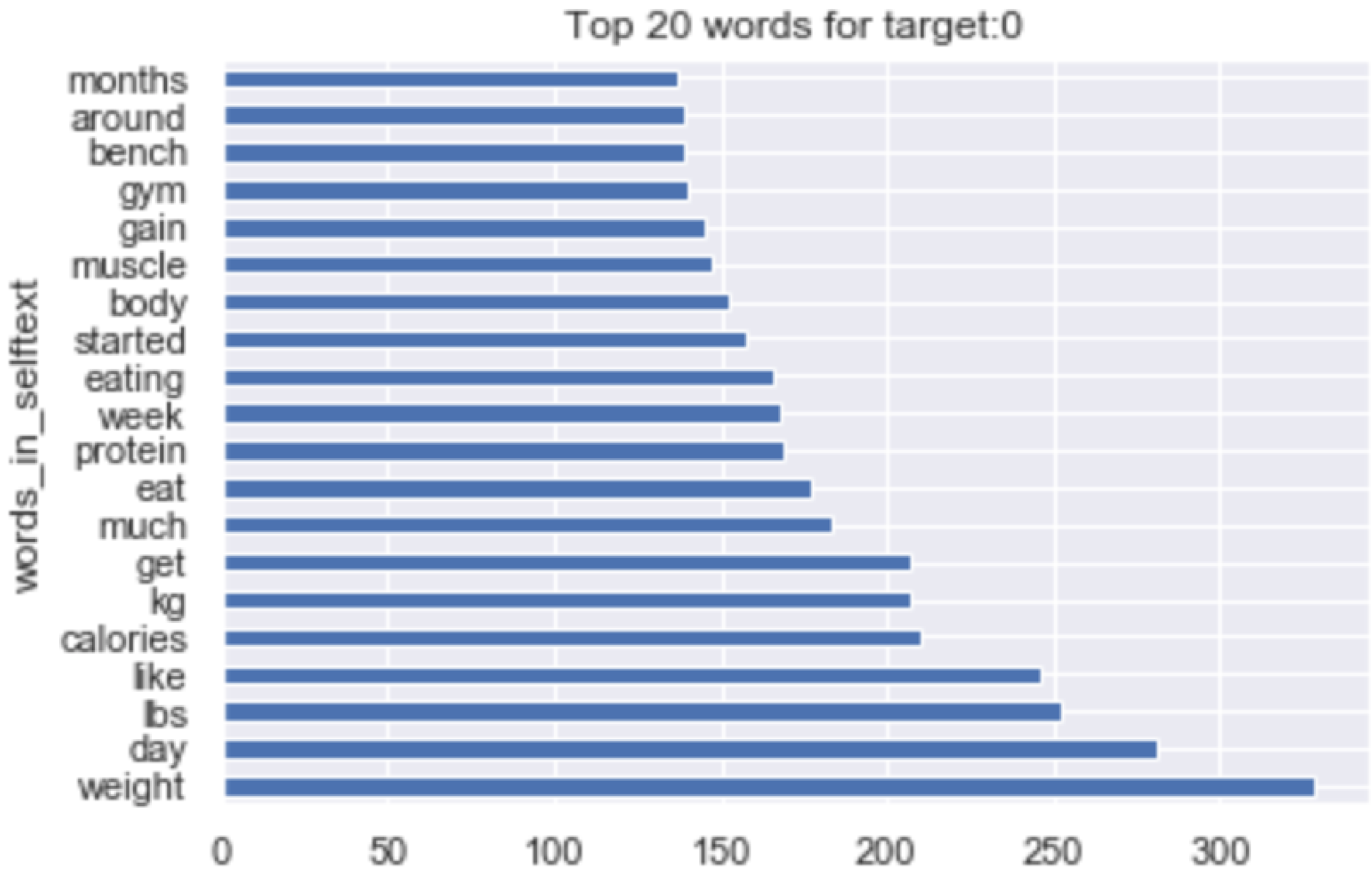
Target 0

→

Unigram



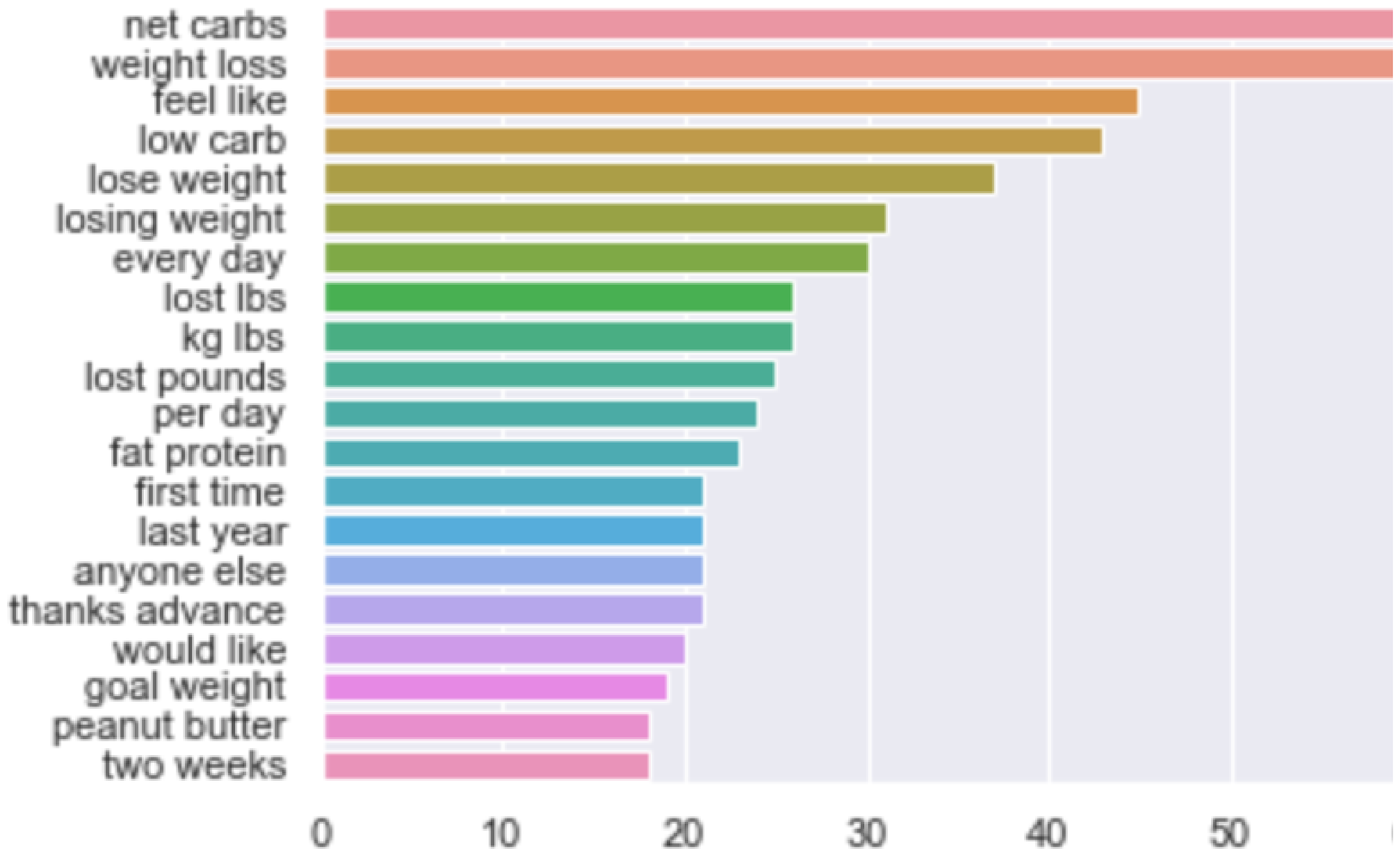
Target 1



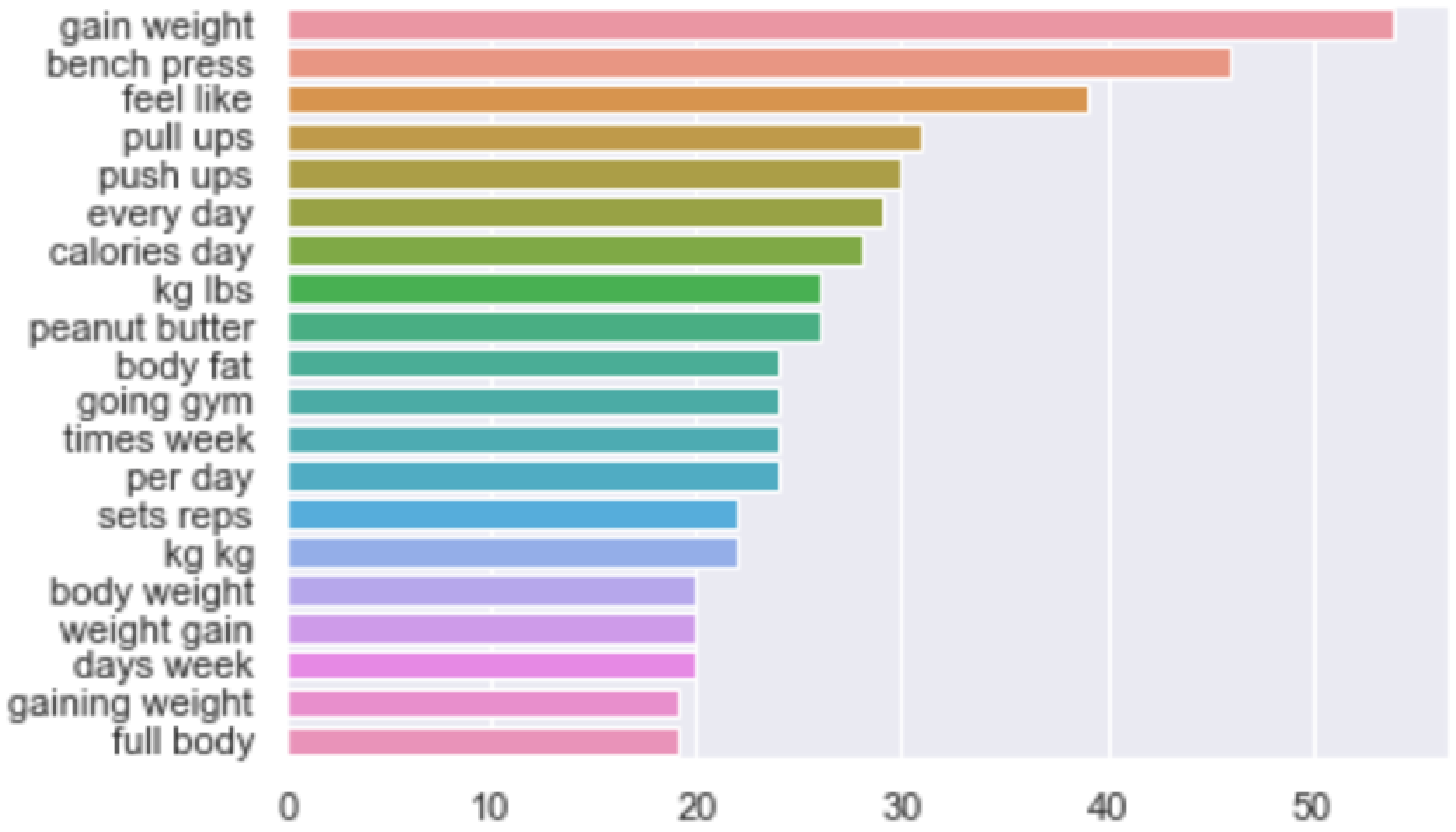
Target 0

→

Bi-gram



Target 1



Target 0

→

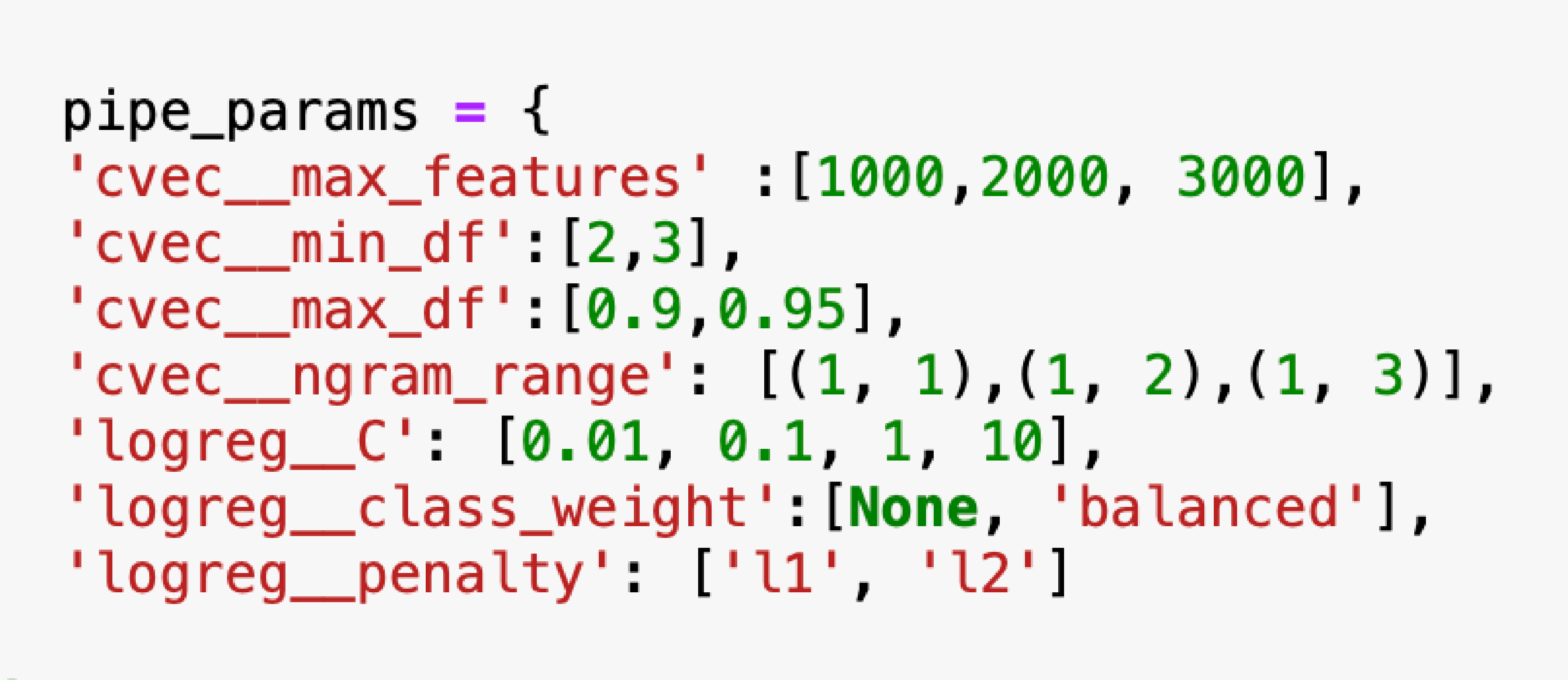
04

Model Prep

A pipeline is created for each model, varying the vectorizer for each

→

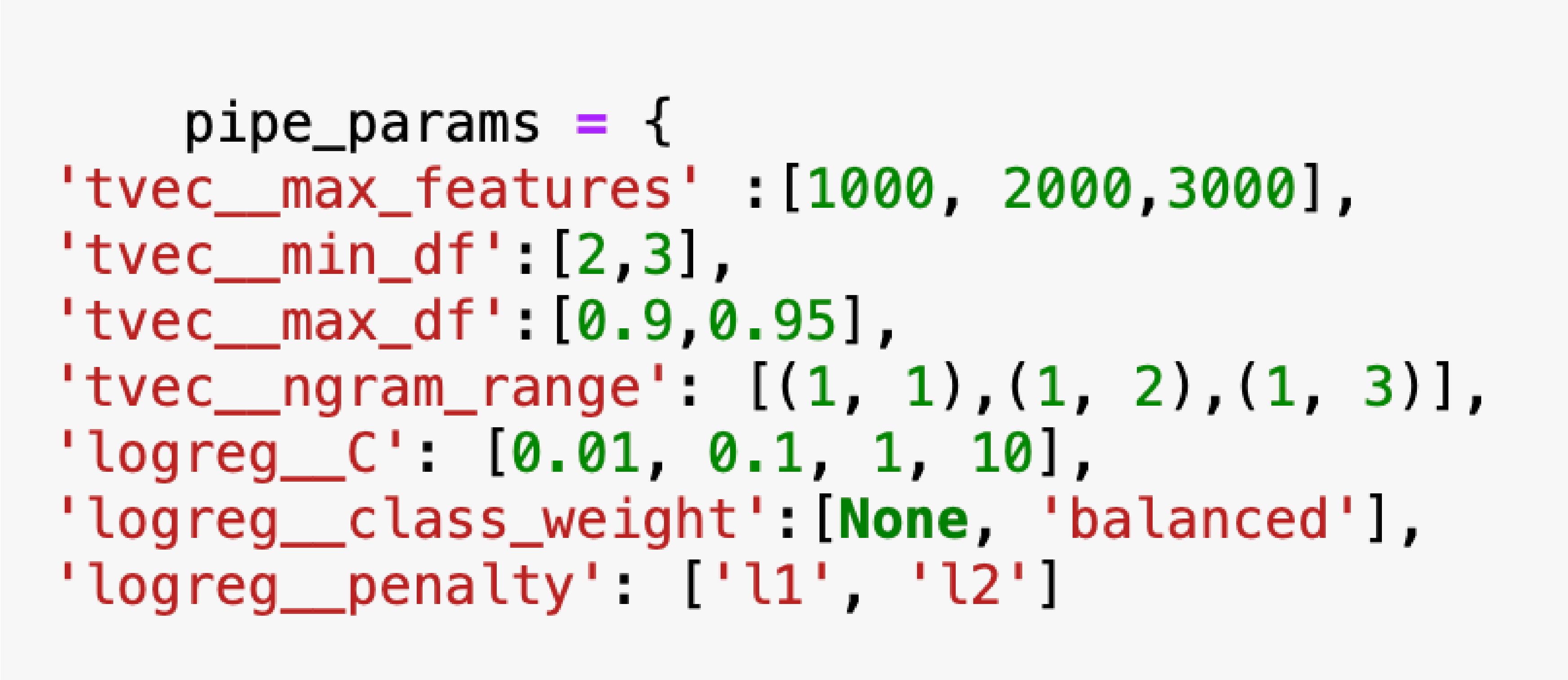
CountVectorizer => Logistic Regression



```
pipe_params = {
'cvec__max_features' :[1000,2000, 3000],
'cvec__min_df':[2,3],
'cvec__max_df':[0.9,0.95],
'cvec__ngram_range': [(1, 1),(1, 2),(1, 3)],
'logreg__C': [0.01, 0.1, 1, 10],
'logreg__class_weight':[None, 'balanced'],
'logreg__penalty': ['l1', 'l2']
```

Best params = {'cvec__max_df': 0.9, 'cvec__max_features': 1000, 'cvec__min_df': 2, 'cvec__ngram_range': (1, 2), 'logreg__C': 0.01, 'logreg__class_weight': None, 'logreg__penalty': 'l2'}

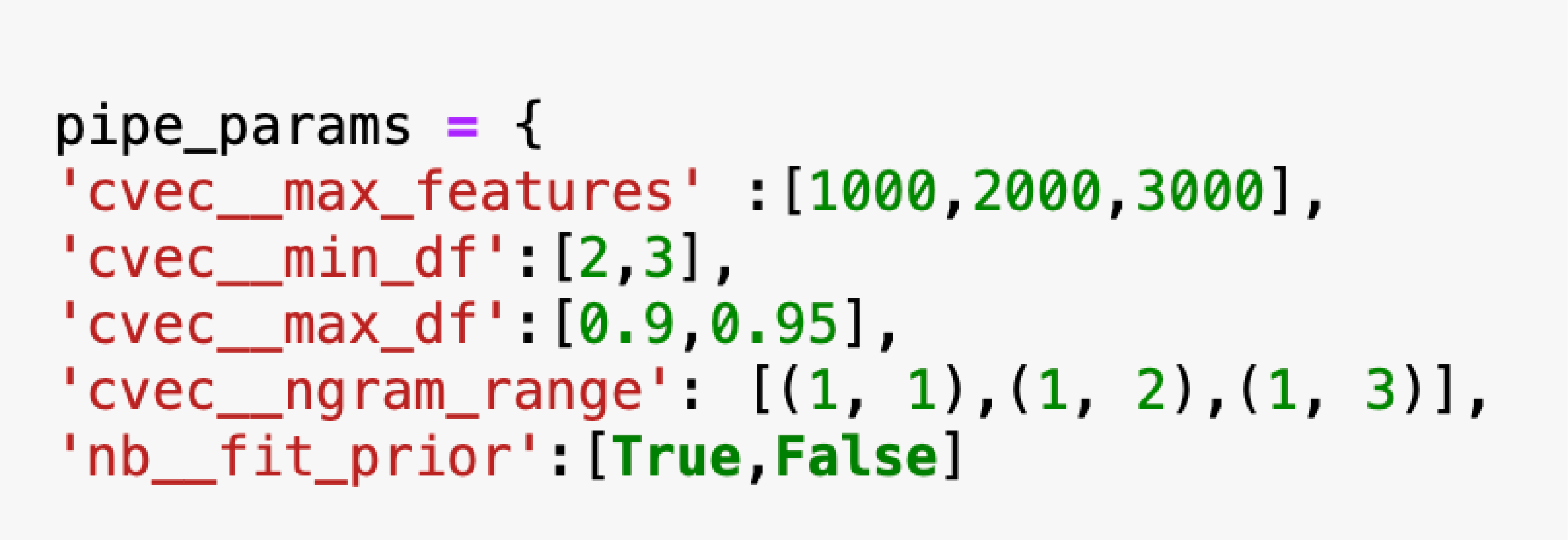
TfidfVectorizer => Logistic Regression



```
    pipe_params = {
'tvec__max_features' :[1000, 2000,3000],
'tvec__min_df':[2,3],
'tvec__max_df':[0.9,0.95],
'tvec__ngram_range': [(1, 1),(1, 2),(1, 3)],
'logreg__C': [0.01, 0.1, 1, 10],
'logreg__class_weight':[None, 'balanced'],
'logreg__penalty': ['l1', 'l2']
```

Best params = {'logreg__C': 1, 'logreg__class_weight': 'balanced', 'logreg__penalty': 'l2', 'tvec__max_df': 0.9, 'tvec__max_features': 1000, 'tvec__min_df': 2, 'tvec__ngram_range': (1, 2)}

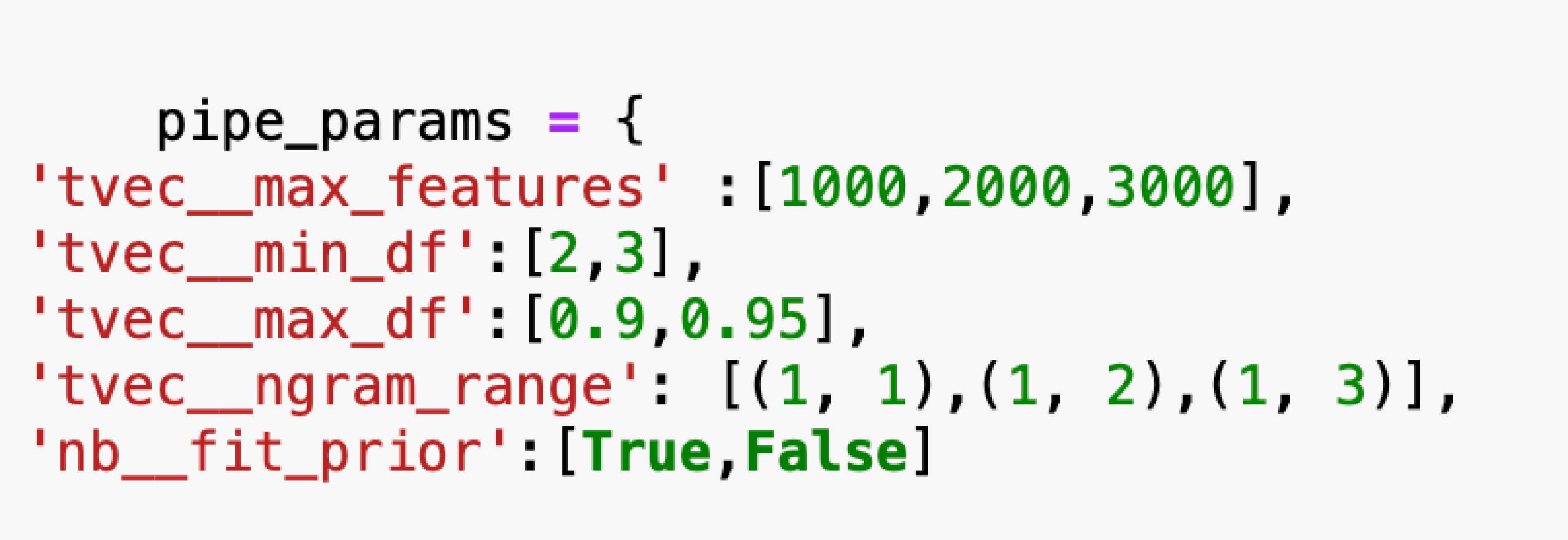
CountVectorizer => MultinomialNB



```
pipe_params = {
'cvec__max_features' :[1000,2000,3000],
'cvec__min_df':[2,3],
'cvec__max_df':[0.9,0.95],
'cvec__ngram_range': [(1, 1),(1, 2),(1, 3)],
'nb__fit_prior':[True,False]
```

Best params = {'cvec__max_df': 0.9, 'cvec__max_features': 2000,
'cvec__min_df': 2, 'cvec__ngram_range': (1, 3),
'nb__fit_prior': False}

TfidfVectorizer => MultinomialNB



```
     pipe_params = {
'tvec__max_features' :[1000,2000,3000],
'tvec__min_df':[2,3],
'tvec__max_df':[0.9,0.95],
'tvec__ngram_range': [(1, 1),(1, 2),(1, 3)],
'nb__fit_prior':[True,False]
```

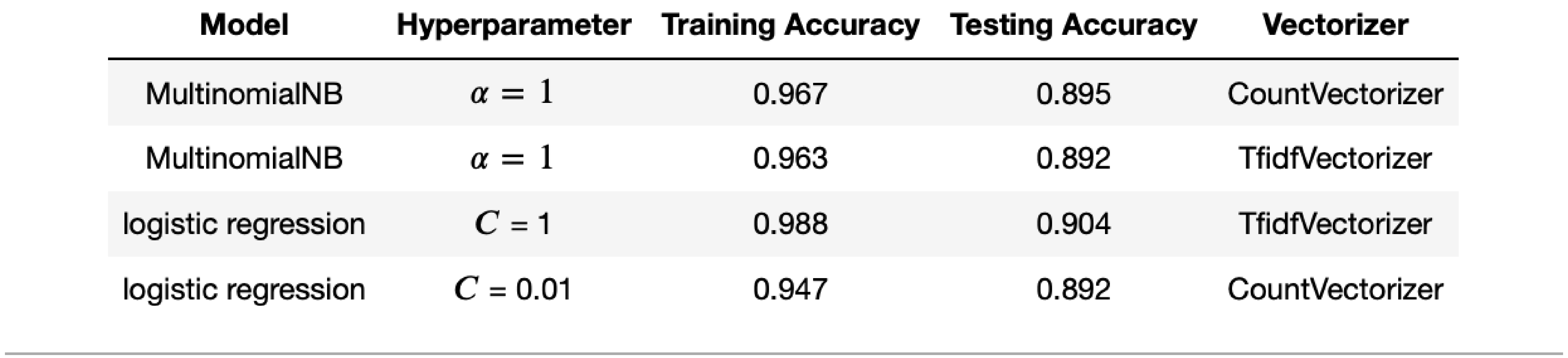
Best params = {'nb__fit_prior': False, 'tvec__max_df': 0.9,
'tvec__max_features': 1000, 'tvec__min_df': 3,
'tvec__ngram_range': (1, 3)}

05

Model fit and evaluation

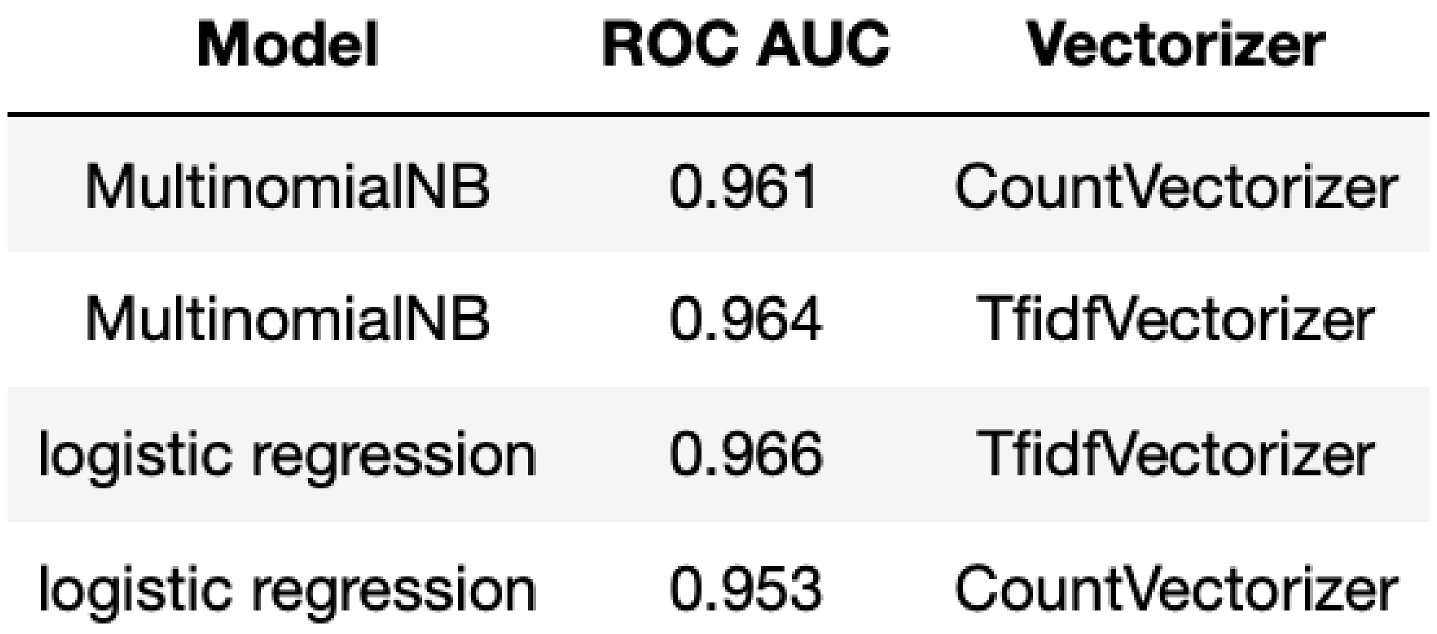
Metrics: accuracy, precision, f1-score and ROC AUC score is used to identify the best model

Accuracy for train vs test



Model	Hyperparameter	Training Accuracy	Testing Accuracy	Vectorizer
MultinomialNB	$\alpha = 1$	0.967	0.895	CountVectorizer
MultinomialNB	$\alpha = 1$	0.963	0.892	TfidfVectorizer
logistic regression	C = 1	0.988	0.904	TfidfVectorizer
logistic regression	C = 0.01	0.947	0.892	CountVectorizer

ROC AUC



Model	ROC AUC	Vectorizer
MultinomialNB	0.961	CountVectorizer
MultinomialNB	0.964	TfidfVectorizer
logistic regression	0.966	TfidfVectorizer
logistic regression	0.953	CountVectorizer

Precision, Recall

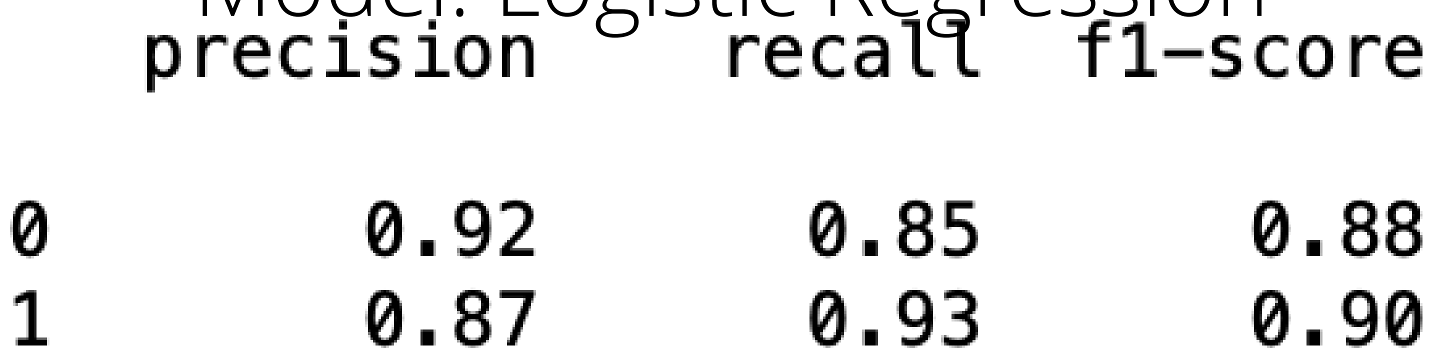
Precision: The logistic regression classifier performed better

Recall: The naive bayes classifier performed better

f1-score

Vectoriser: CountVectoriser

Model: Logistic Regression

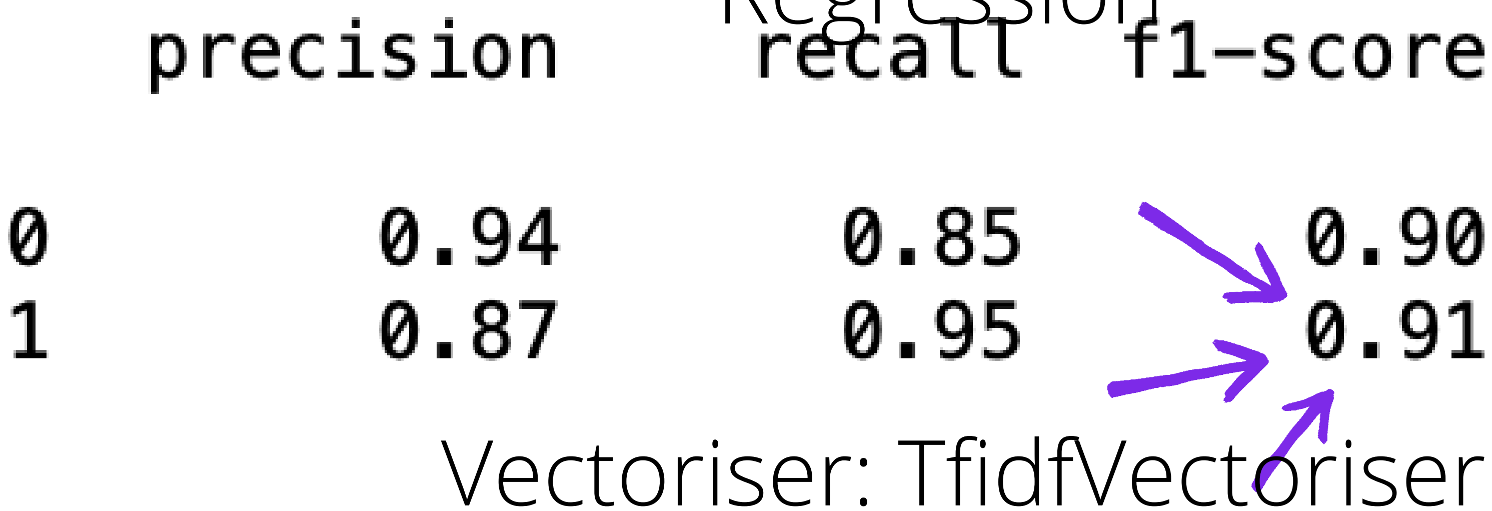


```
     precision    recall  f1-score

0         0.92      0.85      0.88
1         0.87      0.93      0.90
```

Vectoriser: TfidfVectoriser

Model: Logistic Regression

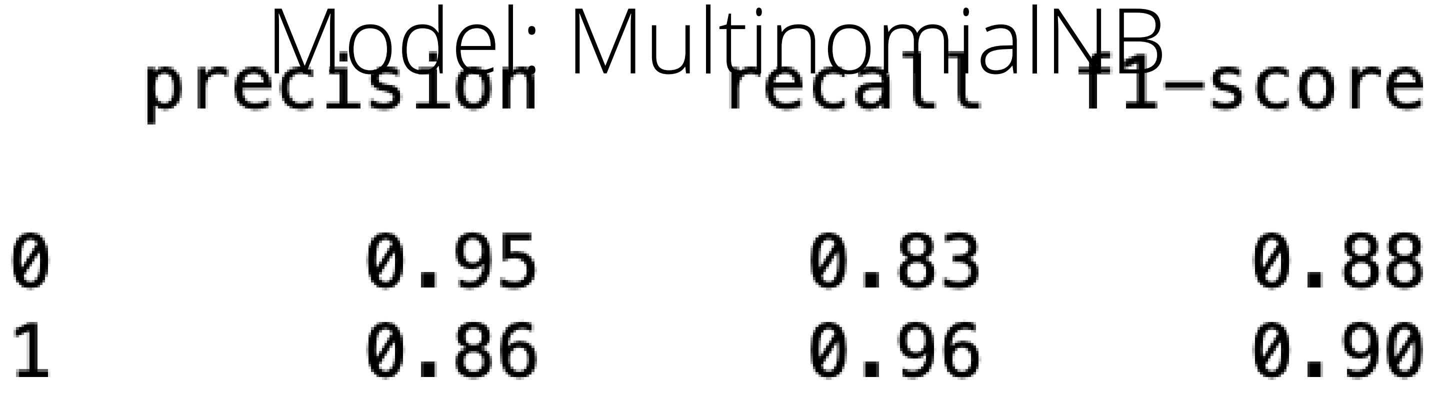


```
     precision    recall  f1-score

0         0.94      0.85      0.90
1         0.87      0.95      0.91
```

Vectoriser: CountVectoriser

Model: MultinomialNB

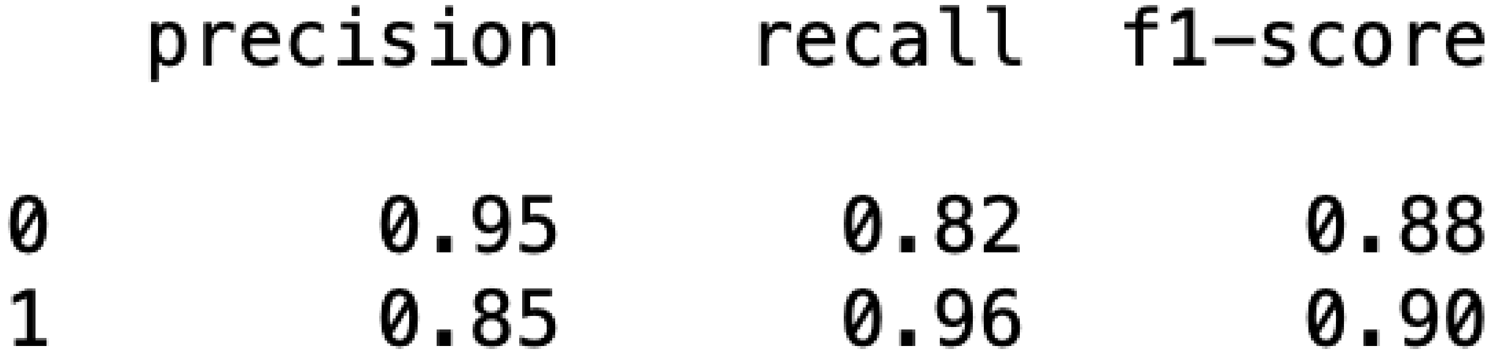


```
     precision    recall  f1-score

0         0.95      0.83      0.88
1         0.86      0.96      0.90
```

Vectoriser: TfidfVectoriser

Model: MultinomialNB



```
     precision    recall  f1-score

0         0.95      0.82      0.88
1         0.85      0.96      0.90
```

06

Business Recommendations and Conclusion

→

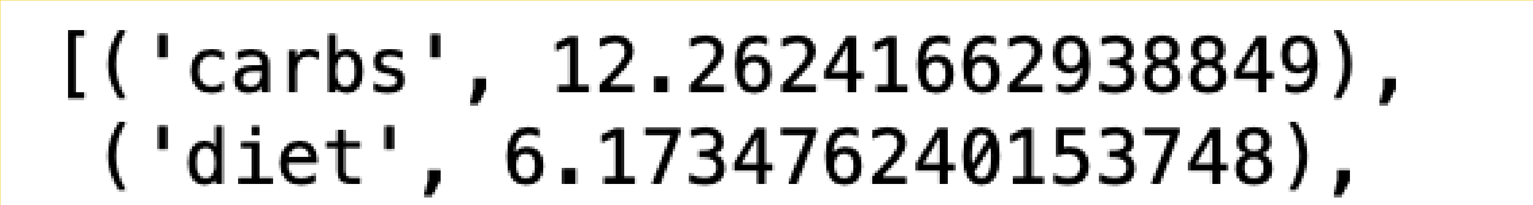
Best Model

TfidfVectorizer => Logistic Regression

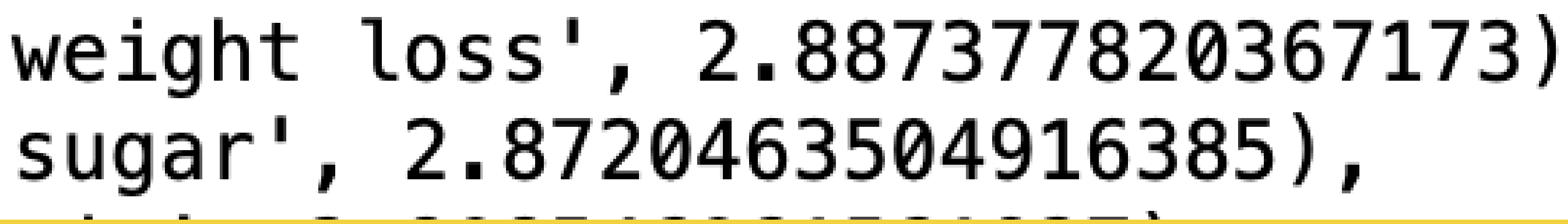
1. Identify words that contain more predictive power- Words that occur often in one document but don't occur in many documents
2. Separates the feature space into classes and typically works reasonably well even when some of the variables are correlated

Food for thought

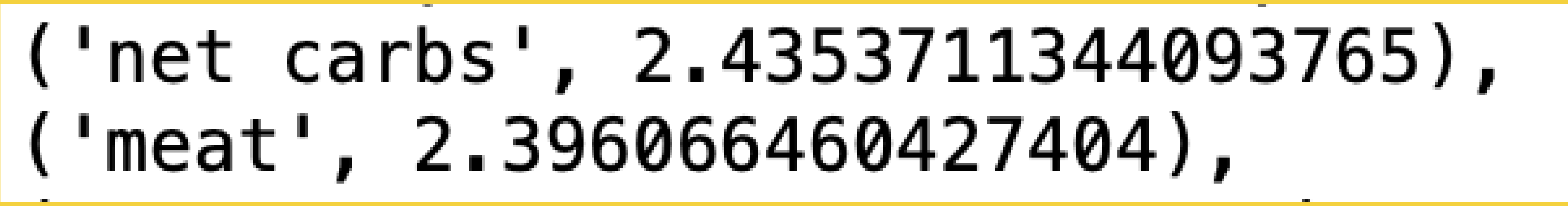
The community in general are concerned about reducing the net carb intake



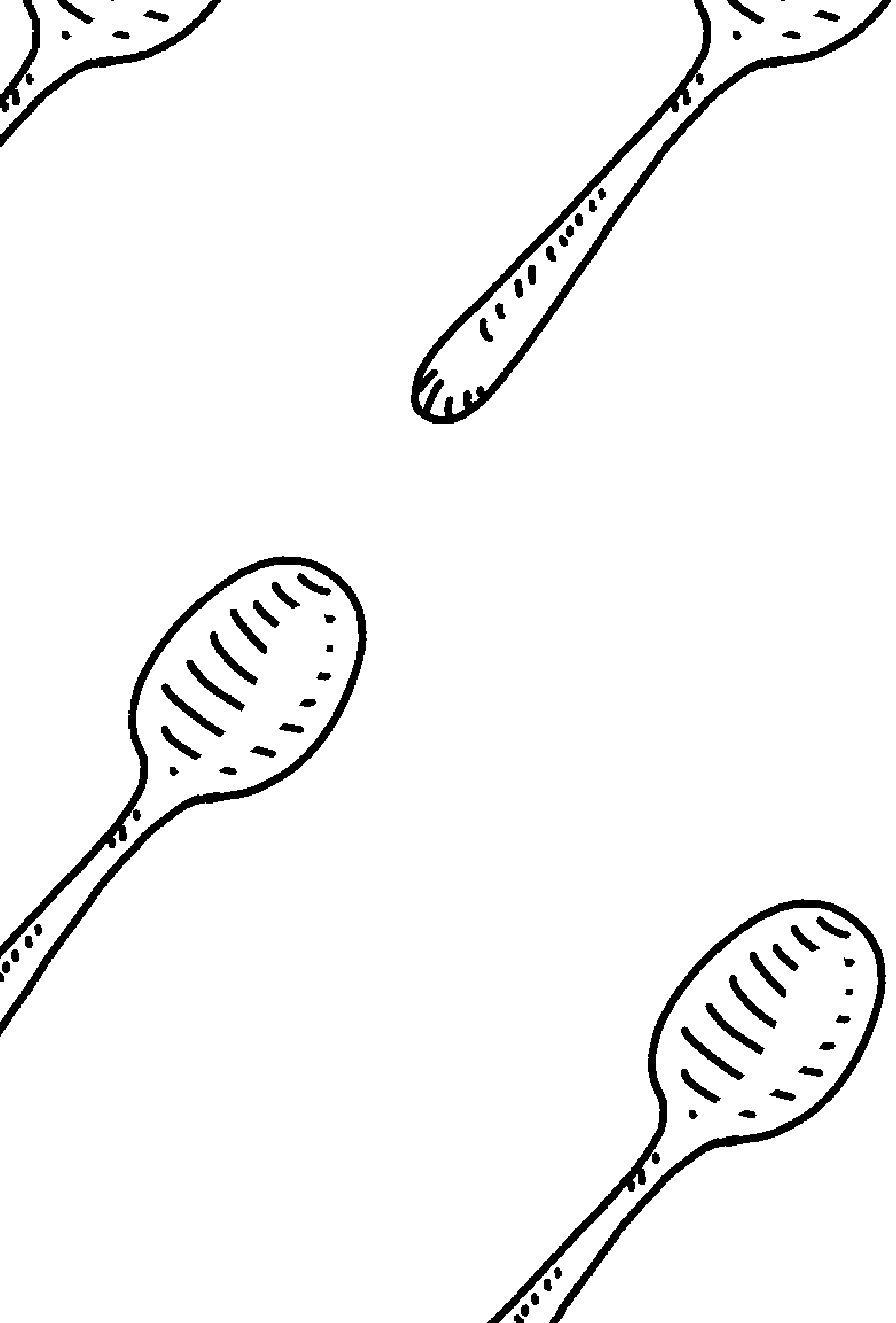
```
[('carbs', 12.26241662938849),
 ('diet', 6.173476240153748),
```



```
weight loss', 2.887377820367173)
sugar', 2.872046350491638 5),
```



```
('net carbs', 2.435371134409376 5),
('meat', 2.396066460427404),
```



Thank you.