Twitter Sentiment Analysis

Group 13: Chionh Wan Sim, Hiew Shani, Kushioka Nodoka, Yeo Fu Kai Marcus

OUR TEAM

Marcus

Y2 Data Science & Analytics

Shani

Y2 Data Science & Economics

Wan Sim

Y2 Data Science & Analytics

Nodoka

Y2 Data Science & Analytics

Flow of Presentation

1 Nodoka

1. Introduction of the topic and dataset
2. Issues with dataset

2 Marcus

1. Data Preprocessing
2. Random Forest
3. Logistic Regression

3 Shani

1. k-Nearest Neighbours
2. Hyperparameter Tuning
3. Evaluation Metrics

4 Wan Sim

1. Results
2. Future Improvements

Why?

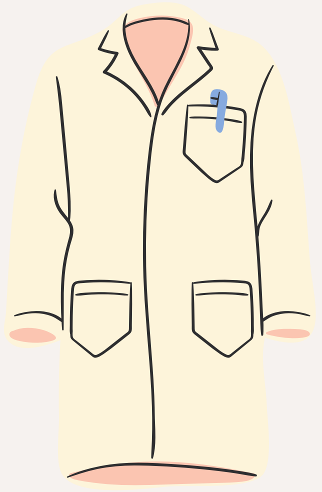
Understand public sentiment on diverse subjects

1. Business Owners

Gauge customer satisfaction and receive feedback

2. Politicians

Gauge sentiment on policies and understand societal concerns



3. Researchers

Especially useful in public health research

Reports Used and their Limitations

1. "Analytics of machine learning-based algorithms for text classification,"
   - ✓ Clear comparison on algorithms
   - ✗ Little details on preprocessing

2. "A Comparative Analysis of Machine Learning Classifiers for Twitter Sentiment Analysis,"
   - ✓ Detailed pre-processing steps
   - ✗ Trained on minimal data points

3. "Sentiment Analysis of Twitter Data,"
   - ✓ Multiple approaches, including non-machine learning
   - ✗ Why is the prediction accuracy of models is highly dependent on the classifiers that were taught by the target domain?

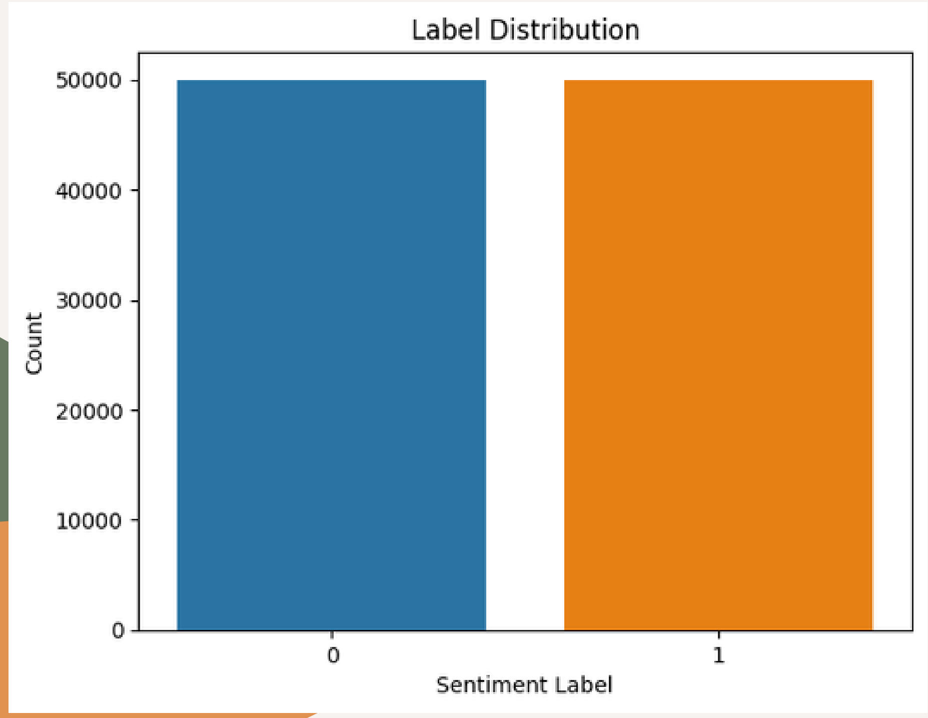
An Introduction

100,000 tweets

Sentiment label

1: ‘Positive sentiment’

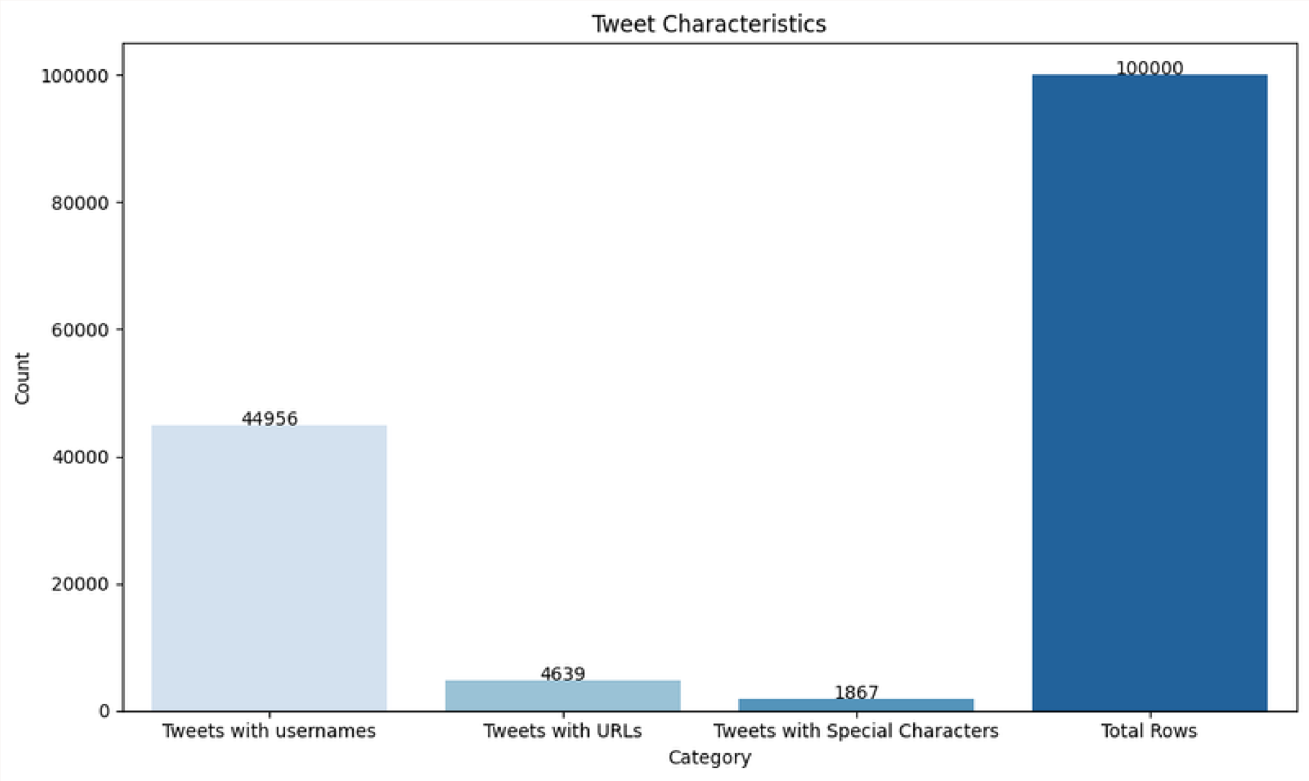
0: ‘Negative Sentiment’

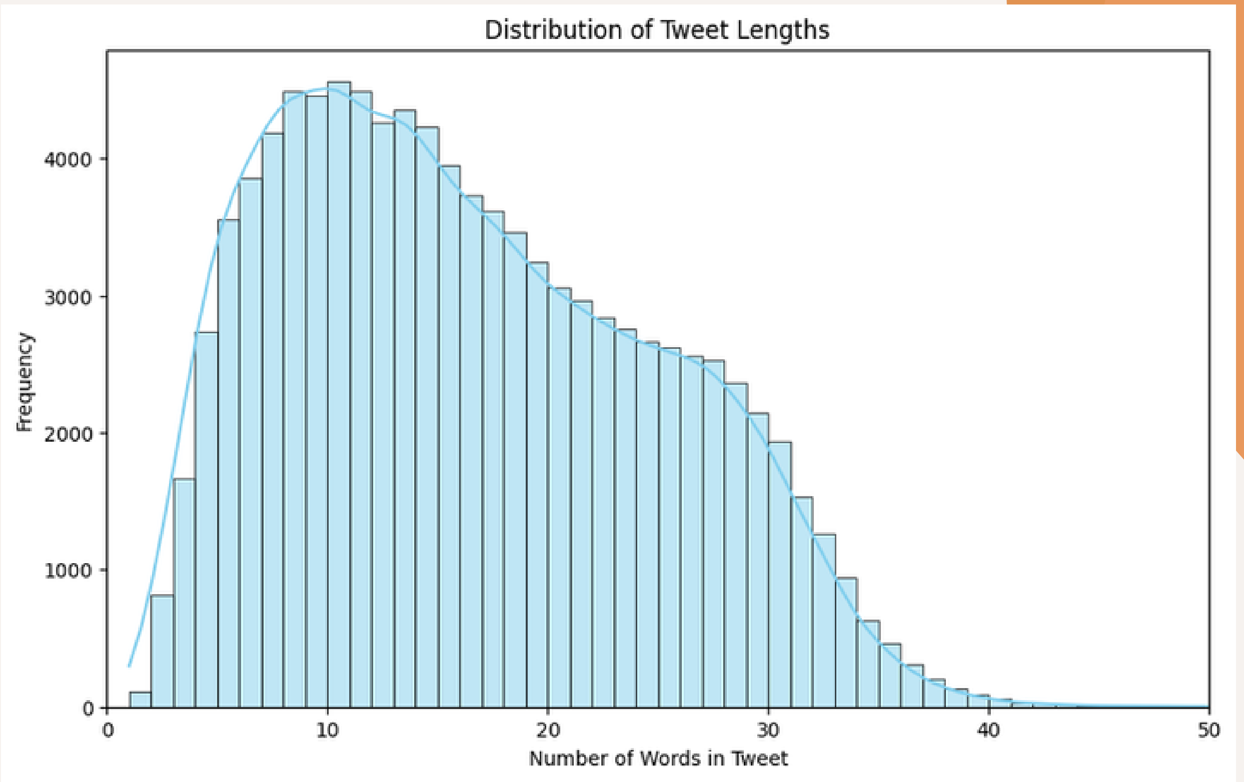


Equal distribution

50,000 instances for
both positive and
negative numbers

Limitations





1. Noise

Many tweets with irrelevant text

2. Short tweets

Too short to understand the context

3. Slangs and Abbreviations

NLP struggle with shorthand text

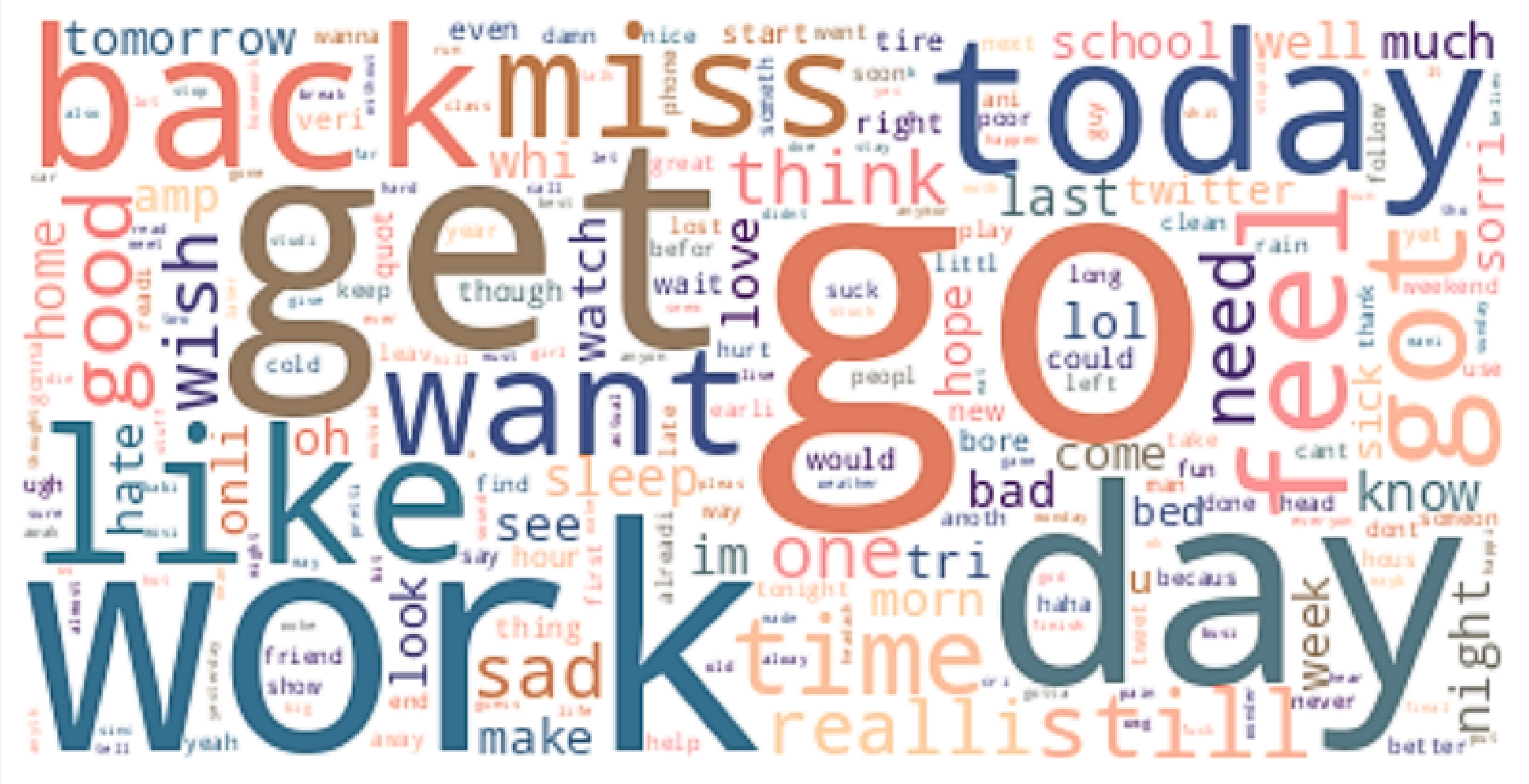
Data Preprocessing

No.	Steps
1	Removed Usernames
2	Removed URLs
3	Stripped Punctuations, Numbers, and Special Characters
4	Tokenization
5	Stemming and Lemmatization
6	(Remove stop words)

"@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D"

[david, day, got, carr, bummer, shoulda, third]

Word Cloud After Preprocessing



“Negative” Sentiments

“Positive” Sentiments

Splitting into Training & Test Set

sklearn.model_selection.train_test_split

`sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)` [source]

80%

Training Set

20%

Test Set

Tf-idf Vectorizer

sklearn.feature_extraction.text.TfidfVectorizer

class sklearn.feature_extraction.text.**TfidfVectorizer**(*, *input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False*) [source]

```
[ ]  vectorizer = TfidfVectorizer(ngram_range=(1, 3), max_features = 10000, stop_words='english')
```

Takes into account
unigrams, **bigrams**, and **trigrams**

Analyse the **top 10,000**
most frequent terms

Remove common
English stop words

Fit Transform

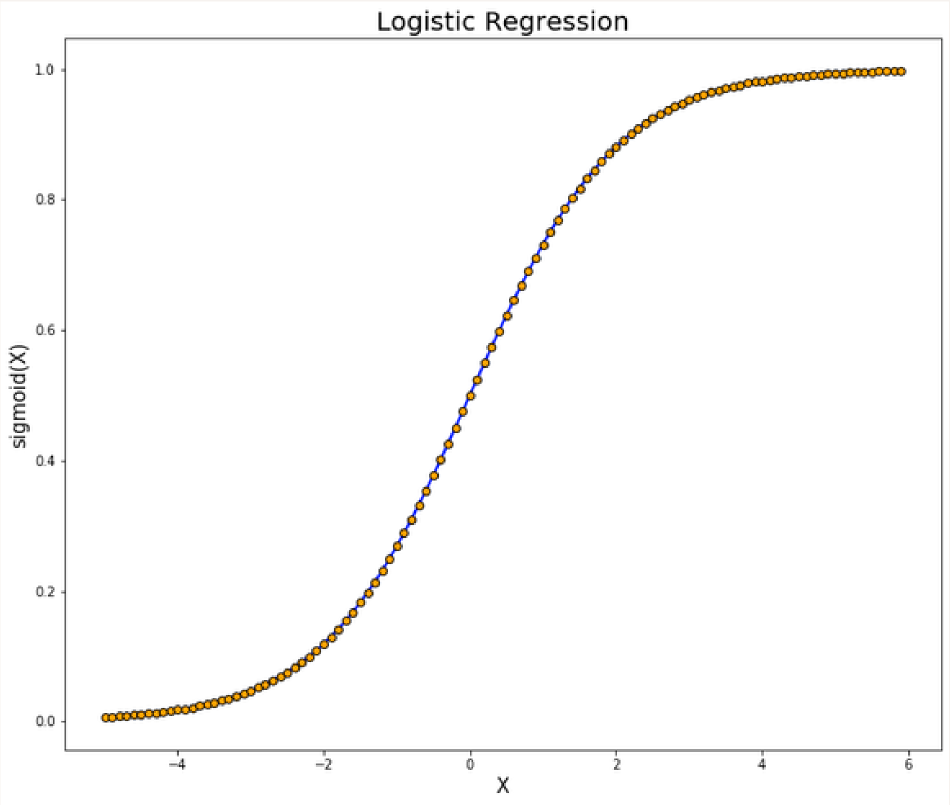
fit_transform(*raw_documents, y=None*) [source]

Learn the vocabulary dictionary and return document-term matrix.

This is equivalent to fit followed by transform, but more efficiently implemented.

Parameters:	**raw_documents : *iterable*** An iterable which generates either str, unicode or file objects. **y : *None*** This parameter is ignored.
Returns:	**X : *array of shape (n_samples, n_features)*** Document-term matrix.

Logistic Regression



sklearn.linear_model.LogisticRegression

class sklearn.linear_model.**LogisticRegression**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

What is Logistics Regression?

A statistical model for binary classification, estimates the probability of a data point belonging to a specific class through a logistic function

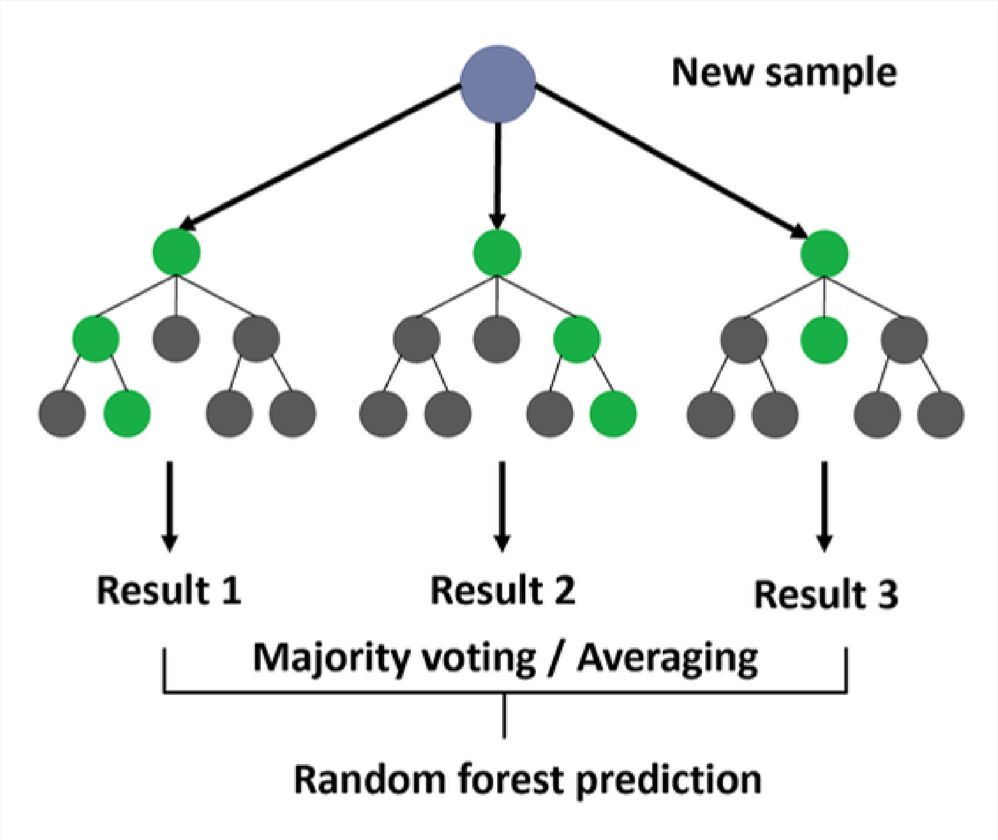
- Simplistic
- Computational Efficient
- Reveals most influential words

Key Parameters

C : *float, default=1.0*

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Random Forest



sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

What is Random Forest?

t is an ensemble learning method. It constructs multiple decision trees and outputs the mode class

- Ability to handle large, high-dimensional datasets
- Robustness against overfitting

Key Parameters

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

n_estimators : *int, default=100*
The number of trees in the forest.

K Nearest Neighbours (KNN)

sklearn.neighbors.KNeighborsClassifier ¶

class sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5*, ***, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=None*) [source]

What is K-Nearest Neighbour?

An instance-based learning algorithm that classifies data points based on the majority class among their k nearest neighbours in a feature space

- Makes no explicit assumptions
- Flexibility in modelling complex relationships

Key Parameters

n_neighbors : *int, default=5*
Number of neighbors to use by default for `kneighbors` queries.

p : *float, default=2*
Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

Hyperparameter Tuning

RandomizedSearchCV

sklearn.grid_search.RandomizedSearchCV

```
class sklearn.grid_search.RandomizedSearchCV(estimator, param_distributions, n_iter=10, scoring=None, fit_params=None, n_jobs=1, iid=True, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', random_state=None, error_score='raise')
```

[source]

- Faster: only covers a subset of hyperparameter combinations
- Risk of Missing Optimal Values

GridSearchCV

sklearn.model_selection.GridSearchCV

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False)
```

[source]

- Exhaustively explores all parameter combinations
- Computationally expensive

Hyperparameter Tuning

RandomizedSearchCV

sklearn.grid_search.RandomizedSearchCV

```
class sklearn.grid_search.RandomizedSearchCV(estimator, param_distributions, n_iter=10,
scoring=None, fit_params=None, n_jobs=1, iid=True, refit=True, cv=None, verbose=0,
pre_dispatch='2*n_jobs', random_state=None, error_score='raise')
```
[source]

- Faster: only covers a subset of hyperparameter combinations
- Risk of Missing Optimal Values

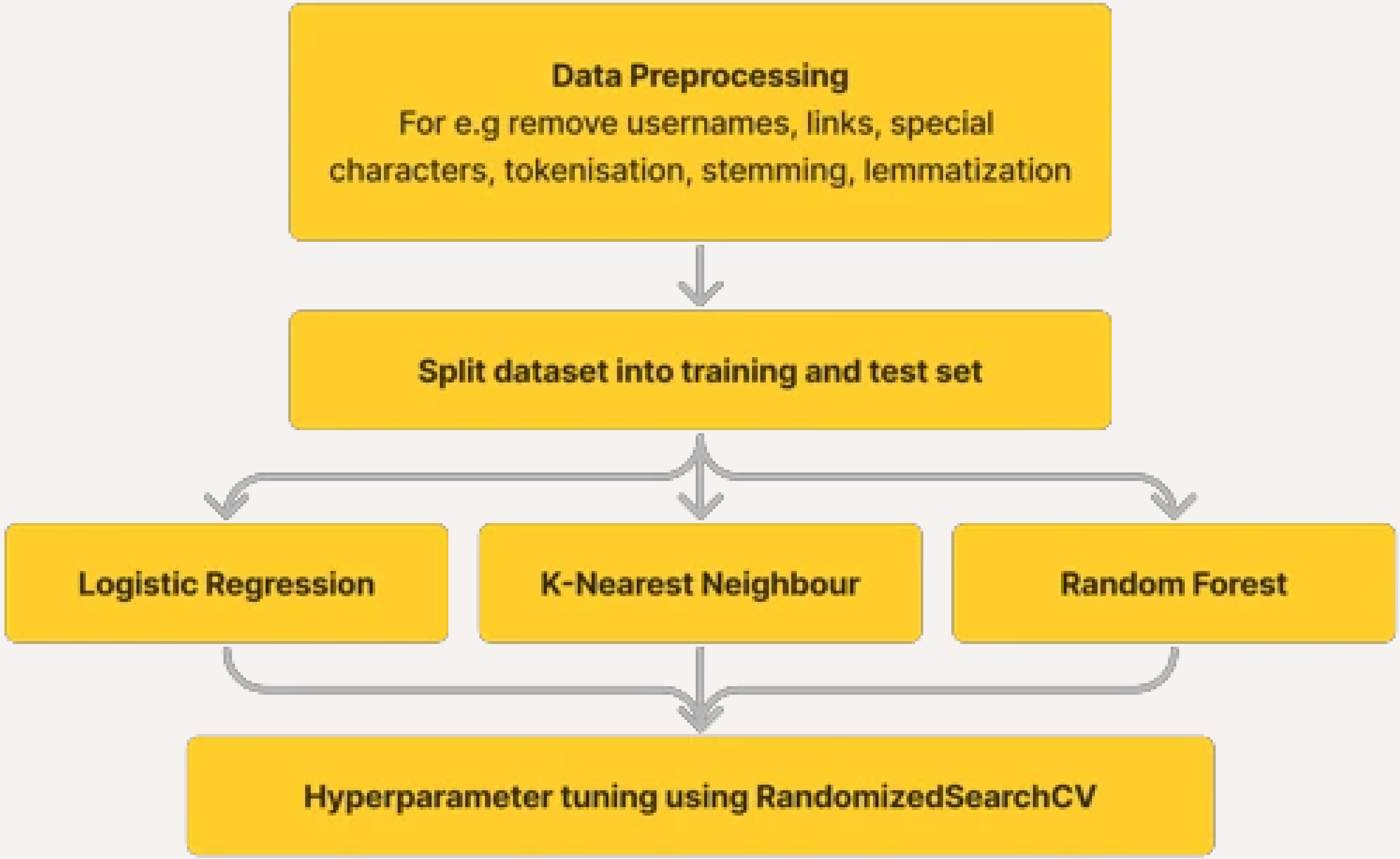
GridSearchCV

sklearn.model_selection.GridSearchCV

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, *,
scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs',
error_score=nan, return_train_score=False)
```
[source]

- Exhaustively explores all parameter combinations
- Computationally expensive

Overall Methodology



Evaluation Metrics

Accuracy

$$\frac{TP+TN}{TP+TN+FP+FN}$$

a measure of the overall correctness of a classification model

Precision

$$\frac{TP}{TP+FP}$$

assesses the model's accuracy when it predicts a positive class

Recall

$$\frac{TP}{TP+FN}$$

measures the model's ability to correctly identify all relevant instances of a specific class.

F1-Score

$$\frac{2*precision*recall}{precision+recall}$$

It provides a balance between precision and recall, rewarding models that have both high precision and high recall.

Evaluation Metrics

Accuracy

$$\frac{TP+TN}{TP+TN+FP+FN}$$

a measure of the overall correctness of a classification model

Precision

$$\frac{TP}{TP+FP}$$

assesses the model's accuracy when it predicts a positive class

Recall

$$\frac{TP}{TP+FN}$$

measures the model's ability to correctly identify all relevant instances of a specific class.

F1-Score

$$\frac{2*precision*recall}{precision+recall}$$

It provides a balance between precision and recall, rewarding models that have both high precision and high recall.

Results

```
Model: Logistic Regression
Best Hyperparameters: {'C': 1}
Accuracy on Validation Set: 0.75

Evaluation on Test Set:
Accuracy on Test Set: 0.75

Classification Report on Test Set:
              precision    recall  f1-score   support

    Negative       0.77      0.73      0.75     10037
    Positive       0.74      0.78      0.76      9963

    accuracy                           0.75     20000
   macro avg       0.75      0.75      0.75     20000
weighted avg       0.75      0.75      0.75     20000
```

Results

```
Model: Random Forest
Best Hyperparameters: {'n_estimators': 175, 'max_depth': None}
Accuracy on Validation Set: 0.74
F1 Score on Validation Set: 0.74

Evaluation on Test Set:
Accuracy on Test Set: 0.74
F1 Score on Test Set: 0.74

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.73      0.74     10037
           1       0.74      0.75      0.74      9963

    accuracy                           0.74     20000
   macro avg       0.74      0.74      0.74     20000
weighted avg       0.74      0.74      0.74     20000
```

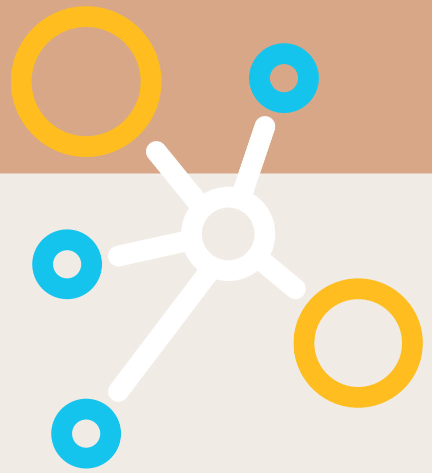
Results

```
Model: K-Nearest Neighbors
Best Hyperparameters: {'n_neighbors': 2}
Accuracy on Validation Set: 0.61

Evaluation on Test Set:
Accuracy on Test Set: 0.61

Classification Report on Test Set:
              precision    recall  f1-score   support

    Negative       0.57      0.85      0.68     10037
    Positive       0.71      0.36      0.48      9963

    accuracy                           0.61     20000
   macro avg       0.64      0.61      0.58     20000
weighted avg       0.64      0.61      0.58     20000
```

Future Improvements

Alternative Models

Ensemble Methods
Deep Learning models:
Recurrent Neural
Networks (RNNs)

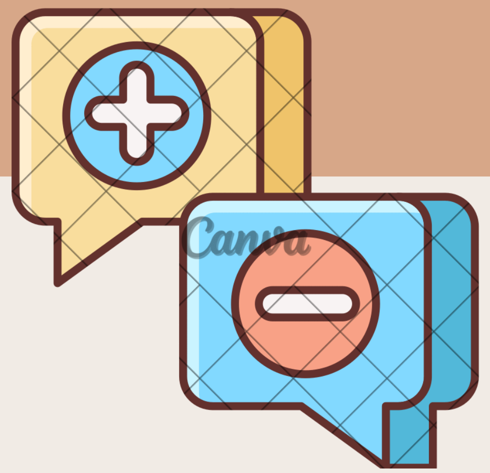


Data Manipulation

Oversampling
Undersampling
Synthetic data generation

Additional Information

incorporating annotated
lists of words with their
sentiment polarity




Q&A
?
!

Workload Breakdown

Marcus

40%

Shani

30%

Wan Sim

10%

Nodoka

20%