# COMP 4900 — Mini-Project 2 Report

[redacted], Rongqiang Zhang, and [redacted]

November 5, 2020

## Abstract

In this project we implemented Bernoulli Naïve Bayes from scratch to analyze text dataset from the website Reddit. We investigated the performance of Bernoulli Naïve Bayes and three additional classification models from the SciKit learn package using k-fold cross-validation. We found that the Linear Support Vector Classificatio approach achieved better accuracy.

## 1 Introduction

This report presents the experiments using four classifiers: Bernoulli Naïve Bayes, Logistic Regression, Linear Support Vector Classification and Decision Tree. In this project, we implemented the Bernoulli Naïve Bayes classifier from scratch, did research on classification models from the SciKit learn and used them to identify the subreddit where post or comment originally came from.

After the experiments, we found that the Bernoulli Naïve Bayes achieved around 83 percent accuracy, the Logistic Regression achieved around 87 percent accuracy, the LinearSVC achieved around 88 percent accuracy, and the Decision Tree achieved around 71 percent accuracy.

Number of features affect run time and accuracy significantly. With adjusting max features limitation, the accuracy is improved for LinearSVC and Decision Tree while the run time is longer. The LinearSVC achieved around 90 percent accuracy, and Decision Tree achieved around 72 percent accuracy.

## 2 Datasets

We are using different text classification models to identify the subreddit where post or comment originally came from. There are a total number of 11582 training examples in the training dataset. Each training example includes one field of text and one field of the subreddit it was posted on. Among the training examples, 2382 of them are from 'datascience' subreddit, 2064 of them are from 'cars' subreddit, 1988 of them are from 'rpg' subreddit, 1617 of them are from 'hardware' subreddit, 1304 of them are from 'anime' subreddit, 1016 of them are from 'gamedev' subreddit, 784 of them are from 'gamernews' subreddit, and 427 of them are from 'computer' subreddit.

To prepare the feature vectors, we shuffle the rows randomly in the training set firstly and separate text and subreddit into two series. Next, we create a CountVectorizer or TfidfVectorizer based on

the type of the classification model being used. Then, we set the parameters of the vectorizer, including stop_words, binary, and max_feature and apply the fit_transform function of the vectorizer on the text series. Finally, we receive the return value which is the feature vector.

## 3 Proposed Approach

We apply the fit_transform of the vectorizer on the text series, and we get the document-term matrix as return value which is the feature vector. Each feature in the feature vector presents a token count or a TF-IDF feature depending on the type of vectorizer being used.

We implement Bernoulli Naïve Bayes [1] from scratch. Firstly, according to the Naïve Bayes assumption, features are conditionally independent.

$$P(x_j|y) = P(x_j|y, x_k) \tag{1}$$

Secondly, we define $\theta_k = P(y = k)$ and $\theta_{j,k} = P(x_j = 1|y = k)$. Therefore

$$\begin{aligned} P(y = k|x) \propto \delta_k(x) &= \log P(y = k)P(x|y = k) \\ &= \log(P(y = k) \prod_{j=1}^{m} P(x_j|y = k)) \\ &= \log(\theta_k \prod_{j=1}^{m} \theta_{j,k}^{x_j}(1 - \theta_{j,k})^{1-x_j}) \\ &= \log(\theta_k) + \sum_{j=1}^{m} x_j \log(\theta_{j,k}) + (1 - x_j) \log(1 - \theta_{j,k}) \end{aligned} \tag{2}$$

Additionally, when we fit the model on the training set, we are able to compute a list of $\theta_k$ and a matrix of $\theta_{j,k}$.

Finally, given a testing point $x$, for each class $k$, calculate its probability by combining (2), input $x$, already known $\theta_k$ list and $\theta_{j,k}$ matrix. Return the class with the highest probability.

Besides, we did additional experiments using three additional classifiers from the SciKit learn package, including Logistic Regression, Linear Support Vector Classification and Decision Tree.

Logistic Regression models the log-odds with a linear function.

$$a = ln\frac{P(y = 1|x)}{P(y = 0|x)} = w_0 + w_1x_1 + \ldots + w_mx_m = w^T x \tag{3}$$

We can get the probability of $y$ by feeding $a$ to the logistic sigmoid function.

$$\hat{P}(y = 1|x) = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}} \tag{4}$$

The logistic sigmoid maps the whole real axis into a finite interval, and it plays an important role in classification algorithms. [2]

Linear Support Vector Classification is an implementation of Support Vector Machine classifier. The support vector machine produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. [3] The determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum. [2]

Decision Tree is a decision tree classifier, and it makes predictions by recursively splitting on different features according to a tree structure. Firstly, we start with empty decition tree and complete training set. The tree splits on the highest Information Gain.

$$\begin{aligned} H(Y) &= -\sum_{y \in Y} p(y) \log_2 p(y) \\ H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \\ IG(Y|X) &= H(Y) - H(Y|X) \\ &= -\sum_{y \in Y} p(y) \log_2 p(y) + \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \end{aligned} \tag{5}$$

Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions. [3]

We implement a model validation using k-fold cross-validator from SciKit learn package and use it to test the performance of Naïve Bayes model and the SciKit learn models. We get the best model on the training data and perform classification on the test dataset.

## 4 Results

We start with decision of vectorizer. For each model, we test it with a CountVectorizer and a TfidfVectorizer and use 10-fold cross validation to estimate performance in the model training. We record a test result of 10-fold cross validation in the Table 1 below, and the result shows that Logistic Regression and LinearSVC perform better with TfidfVectorizer, and BernoulliNB and Decision Tree perform better with the CountVectorizer.

| | Our BernoulliNB | Logistic Regression | Linear SVC | Decision Tree |
|---|---|---|---|---|
| CountVectorizer accuracy | 83.45 | 86.82 | 84.29 | 71.75 |
| runtime | 36.13 s | 15.37 s | 13.19 s | 23.36 s |
| TfidfVectorizer accuracy | 77.30 | 87.56 | 88.27 | 70.40 |
| runtime | 35.30 s | 11.99 s | 9.73 s | 30.07 s |

Table 1: CountVectorizer with parameters stop_words='english', binary=True, max_features=5000 TfidfVectorizer with parameters stop_words='english', max_features=5000

After we make decision of which vectorizer to be used for each classification model. We test different parameters of the vectorizer and use 10-fold cross validation to estimate performance in the model training. We first adjust the max_features parameter for the vectorizers. We record a test result of 10-fold cross validation in the Table 2 below, and the result shows that LinearSVC and Decision Tree perform better with no max features limitation, while BernoulliNB and Logistic Regression perform better with a max features limitation of 5000.

| | Our BernoulliNB | Logistic Regression | Linear SVC | Decision Tree |
|---|---|---|---|---|
| max_features=4000 accuracy | 82.68 | 87.30 | 87.56 | 71.47 |
| runtime | 30.17 s | 11.40 s | 9.47 s | 22.55 s |
| max_features=5000 accuracy | 83.45 | 87.56 | 88.27 | 71.84 |
| runtime | 35.62 s | 11.71 s | 9.52 s | 22.96 s |
| max_features=None accuracy | 75.42 | 87.54 | 90.00 | 72.13 |
| runtime | 201.06 s | 13.85 s | 10.16 s | 35.54 s |

Table 2: Each model is using the better vectorizer decided above

We then adjust the stop_words parameter for the vectorizers. We record a test result of 10-fold cross validation in the Table 3 below, and the result shows that Logistic Regression and LinearSVC perform better with the stop words provided by the NLTK corpus package, while other two models perform better with SciKit learn package built-in english stop list.

| | Our BernoulliNB | Logistic Regression | Linear SVC | Decision Tree |
|---|---|---|---|---|
| stop_words=None accuracy | 79.05 | 87.11 | 90.17 | 67.97 |
| runtime | 37.12 s | 18.94 s | 12.24 s | 43.90 s |
| stop_words='english' accuracy | 83.45 | 87.56 | 90.00 | 72.16 |
| runtime | 36.32 s | 11.69 s | 10.18 s | 35.12 s |
| stop_words= nltk.corpus.stopwords.words('english') accuracy | 82.70 | 87.65 | 90.23 | 71.78 |
| runtime | 36.56 s | 12.47 s | 10.50 s | 37.08 s |

Table 3: Each model is using the better vectorizer and max_features parameter decided above

By analyzing the results from cross validations, we obtain the best model which is LinearSVC from the SciKit learn package. We improved its performance by adjusting TfidfVectorizer's parameters. Finally, we run the best model on the test dataset and get 91.024 percent leaderboard accuracy on Kaggle.

## 5 Discussion and Conclusion

We implemented the Bernoulli Naïve Bayes classification model from scratch and analyzed it as well as three other classification models from the SciKit learn package, including Logistic Regression, LinearSVC and Decision Tree. We improved the accuracy on the training dataset by selecting the suitable vectorizer for each model. And we improved the accuracy on t by adding adjusting parameters of the vectorizer. We get better understanding of the Bernoulli Naïve Bayes classifier, which will help us to extend analyses to other models.

## References


[1] Komeili Majid. "Introduction to Machine Learning Lecture 11". Lecture Slides. 2020.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science & Business Media, 2006.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.


## Statement of Contributions

[redacted] implemented Bernoulli Naïve Bayes classifier and helped with the report.

Rong did experiments using Logistic Regression, Linear Support Vector Classification as well as Decision Tree and implemented a model validation function for performance Analysis.

[redacted] tested the models with different parameters and analyzed the results in the report.

# Appendix

```
class BernoulliNaiveBayes:
  def __init__(self):
    pass

  def fit(self, X, y):
    self.X = X.toarray()
    self.num_examples = X.shape[0]
    self.float_num_examples = float(self.num_examples)
    self.num_features = X.shape[1]
    self.y = y
    self.y_value_counts = y.value_counts()
    self.y_values = self.y_value_counts.index.values
    self.num_y_values = self.y_value_counts.size
    self.num_y_eq_k = self.y_value_counts.values
    self.array_theta_k = self.num_y_eq_k / self.float_num_examples
    self.matrix_theta_j_k = np.array([[None] * self.num_features] * self.num_y_values)
    for k in range(self.num_y_values):
      self.matrix_theta_j_k[k] =
      (self.X[self.y == self.y_values[k]].sum(axis=0).T.reshape(-1) + 1)
      / (self.num_y_eq_k[k] + 2)

  def _predict_single(self, x):
    class_prob = [None] * self.num_y_values
    for k in range(self.num_y_values):
      feature_likelihood = (x.toarray() * np.log(self.matrix_theta_j_k[k].astype(np.float32))
      + (1-x.toarray()) * np.log(1 - self.matrix_theta_j_k[k].astype(np.float32))).sum()
      class_prob[k] = feature_likelihood + np.log(self.array_theta_k[k])
    return self.y_values[np.argmax(class_prob)]

  def predict(self, X):
    return np.array(list(map(self._predict_single, X)))
```