ECSE 551 Mini-Project 2

Samuel Dayo - 260889723 Aman Sidhu - 260885556 Aydin Karsidag - 260954749

Abstract

This project explored and evaluated the effectiveness of various machine learning (ML) models in performing text classification. The text classification task was to classify Reddit posts and comments to one of four cities: Montreal, Melbourne, London, and Paris. A training dataset was provided by the course for the purpose of this exploration and a corresponding Kaggle competition was used to return test results. Before training the models, the following text pre-processing steps were investigated: stopword removal, lemmatization, tokenization, and text vectorization. Out of four models, Stochastic Gradient Descent (SGD) with Huber loss and linear Support Vector Machine classifiers yielded the best results with accuracies of 72.8% and 71.1% respectively in the Kaggle competition. The other models which did not yield optimal results were Naive Bayes and Logistic Regression. Naive Bayes was implemented without the use of pre-existing ML libraries for this assignment. The optimization of each model was done using SK Learn's grid search class and the previously implemented 10-fold cross-validation code in mini-project 1. We found that text pre-processing had a positive effect on model performance, particularly since the dataset had samples with dirty characters. Confusion matrices and additional metrics are reported. Notably, the F1 score, precision, and recall were the greatest for the Montreal class for the two superior models, indicating that the other classes were incorrectly identified more frequently. The model with the best validation accuracy was SGD with Huber loss and is the same model which scored the highest in the Kaggle competition.

1 Introduction

In this project, various machine learning models were developed to analyze text from Reddit, a popular social media forum. The developed models were designed to identify the subreddit that a given post or comment originated from. Each post or comment was classified to one of four possible subreddits: Melbourne, Paris, London or Montreal. Four classifiers were developed using various machine learning models: Naive Bayes, Logistic Regression, Support Vector Machines, and Stochastic Gradient Descent. Each model was evaluated using 10-fold cross-validation on the pre-processed and vectorized dataset. The pre-processing steps taken in this exploration were stopword removal, lemmatization, tokenization, and text vectorization. Text pre-processing is very data dependant which requires careful attention and is elaborated further in the following section 2.3.

2 Datasets

2.1 Reddit Post Text Classification Dataset

The text classification dataset contains posts from four subreddits: Montreal, Paris, Melbourne, and Paris. The corpus has 1400 total samples, with 350 posts belonging to each subreddit. Based on the langdetect library, 1039 posts were written in English and 361 posts in French. Using the regex pattern *r"(?u)\b(\w+)\b"* to count the number of unique instances of alphanumeric characters across the dataset, we found there were approximately 14913 unique words.

2.2 Data Cleaning

When first loading the train.csv file, we used the "ISO-8859-1 encoding format since the default UTF-8 encoding raised errors when reading certain characters across the training corpus. While most text samples were expressed within the ASCII character set, tokens such as \x82 and \x85 appeared in place of special characters like é and à respectively. Additionally, it was also common to see question marks replacing apostrophes within words, often in words with contractions like "isn't". In such cases, we created a mapping between what we think are the original characters and these token substitutes and replaced them manually to help clean the data.

2.3 Text Preprocessing

With the cleaned data, we performed stopword removal, lemmatization, and tokenization on the input data before passing it to the corresponding model. Since the training corpus and test corpus include English and French words, we processed each sample according to the language. For stopwords, we included both common English and French stopwords from the NLTK and Spacy libraries, as well as, a short list of other stopwords including www and http(s) which created a list of 949 total stopwords. Subsets of these were considered in our following experiments. Similarly, to apply the correct lemmatization we used langdetect to determine the language before applying either Wordnet lemmatizer or French LEFFF (Lexique des Formes Fléchies du Français) lemmatizer on English and French samples, respectively. We also use NLTK's Part-Of-Speech Tag function to apply the corresponding conjugation. Stemming was also considered, but we experimented with lemmatization for better accuracy. Lastly, for word tokenization, we used the regex pattern *r"(?u)*$(?u)$*\b([a-z][a-z]+)\b")* and the re library function findall() to split each sample into words that contained at least two alphabetical characters and separated by whitespace or any punctuation.

3 Proposed Approach

3.1 Model Selection

For this project we investigated four supervised learning models: Bernoulli Naive Bayes, Logistic Regression, Support Vector Machine (SVM), and a Stochastic Gradient Descent (SGD) classifier. The last three models are from the SK Learn library. Naive Bayes was chosen as per assignment instructions, while the other models were chosen due to their popularity as robust text classification models. Each model also represents different levels of complexity and with varying number of hyperparameters to tune which can potentially help fit to the data better.

Bernoulli Naive Bayes model was implemented from scratch with the help of SK Learn's CountVectorizer() function and based on notes from lecture 9. Naive Bayes assumes all features x_i are conditionally independent given $Y = y$. When applied to Bayes' Theorem, we can perform classification by determining which label/class maximizes the log-likelihood function, where the log-likelihood is proportional to $\prod_{i=1}^{n}(P(y_i)\prod_{j=1}^{m}P(x_{i,j}|y_i))$. Logistic Regression uses gradient descent to iteratively find the optimal weights per class, where the class with the highest probability is chosen. SVM aims to find a hyperplane that separates the classes and maximizes the distance to the closest point from either class known as the support vectors. Its kernel hyperparameter gives the model flexibility to adapt to different training sets. Lastly, SGD pairs regularized linear models with stochastic gradient descent learning, meaning the model's weight is updated after every sample, with a decreasing learning rate. The model offers more hyperparameters than logistic regression and has a wide range of possible loss functions.

3.2 Experimental Setup

With the goal of getting the best test accuracy on the Kaggle test dataset, in each of the following experiments we investigated which model hyperparameters or text preprocessing could result in the best-performing model in terms of accuracy and not training latency. For model validation, we reused our version of K-fold and implemented K-fold cross-validation based on functions from the Scikit Learn (SK Learn) library when performing hyperparameter optimization. After shuffling the training

data, grid search helps find the best model based on the highest, average validation accuracy across 10 folds. We chose 10 folds as we saw that average cross-validation accuracy was indicative of the relative performance on the Kaggle test set. For each model, the best hyperparameters are selected and the model then makes predictions on the real test set. In each case, the test data is processed in the same way as the best model during training.

4 Results

4.1 Naive Bayes

Naive Bayes hyperparameter optimization centered on text preprocessing and parameters of the SK Learn CountVectorizer(). For the text processing, we looked at a subset of the stopwords described in 2.3, specifically English and French stopwords from NLTK, English and French stopwords from Spacy, and a union of both in addition to other stopwords. Another text processing step we examined was lemmatizing and not lemmatizing the training corpus, but keeping the tokenization the same as discussed in 2.3. In CountVectorizer, we experimented with different combinations of n-grams and max features. Lastly, we modified the value of alpha used in the Laplace smoothing with values between 1, 2, and 3. The best Naive Bayes model used spacy stopwords, lemmatized data, (1,2) n-grams, and 5000 features to give a validation error of 36% and test accuracy of 65% as seen Table 1.

4.2 Logistic Regression

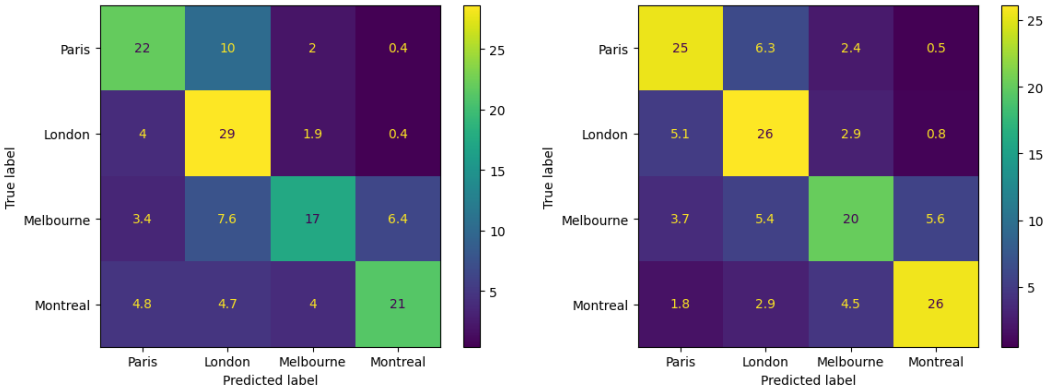
Logistic Regression is a common model in ML and has a few parameters to tune. The training penalty, regularization strength, and model solving algorithm can have a large impact on model performance. Our exploration concluded that a "liblinear" solver and inverse regularization strength of 10 resulted in the best model when trained on unlemmatized data. In Table 1 the model got a validation error of 30.1% and 68.9% test accuracy. Unlike with other models, Logistic Regression worked better with no max feature limit. We also found that the TF-IDF vectorizer gave much better results than the normal count vectorizer, with a min DF of 2 and max DF of 0.5.

4.3 SVM

The main hyper parameters for support vector machines/classifiers we looked at are the kernel and the regularization. We found that a linear kernel with a regularization parameter of 3 yielded the best results for this type of classifier. The other hyper parameters such as stopping tolerance and decision function shape had a marginal impact on the model performance. The pre-processing parameters for the TF-IDF vectorizer used in this model were max DF of 0.1, max features of 10 000 and considering monograms and bigrams. The accuracy is reported in Table 1 and is 71.6% and 71.1% for validation and test.

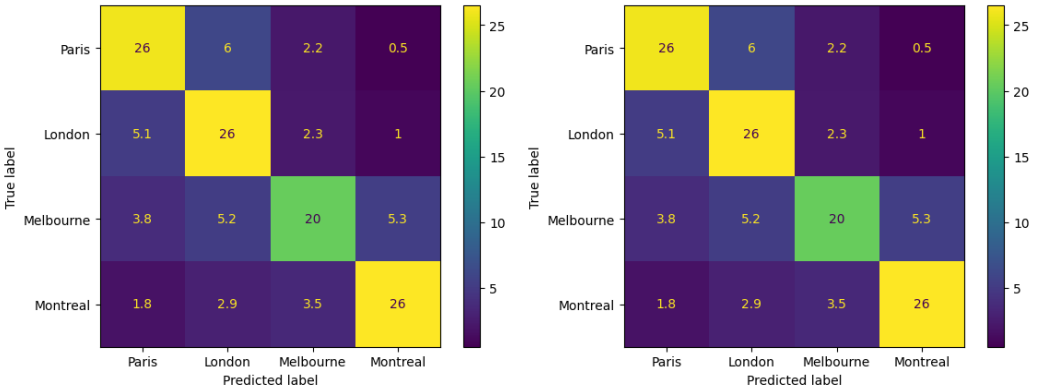
4.4 SGD

Stochastic gradient descent is a model in the Scikit Learn library which uses a stochastic approximation of gradient descent to optimize a loss function. The loss function is a hyper parameter which largely influences the model behavior. For example, the hinge loss will result in a linear SVM classifier while the log loss gives a logistic regression classifier. Other hyper parameters are the regularization type, regularization coefficient, and stopping tolerance. We found that the Huber loss function with an L2 regularization and 10^{-5} coefficient resulted in the best model. The stopping tolerance did not improve performance. This model also used the TF-IDF vectorizer with max DF of 0.2, max features of 20 000 and considered monograms and bigrams. The accuracy is reported in Table 1 and is 71.4% and 72.8% for validation and test.



(a) Naive Bayes confusion matrix.

(b) Logistic Regression confusion matrix.



(c) Linear Support Vector Machine confusion matrix.

(d) Huber Loss SGD confusion matrix.

Figure 1: Confusion Matrices for all four models: Logistic Regression, SVM, Naive Bayes, and SGD. All confusion matrices produced by averaging 10 fold cross-validation classification results on validation sets.

5 Discussion and conclusion

The stochastic gradient descent model utilizing huber loss and the linear SVM models yielded the best performance in K-fold cross-validation displaying 28.6% and 28.4% average validation errors across 10 folds. Thus, these two models were the two most superior of the four models developed. When compared to each other and for the Kaggle competition, the Huber loss SGD model is our best performing model and will be used in the competition. This is mainly supported by the improved Kaggle accuracy, with SGD yielding approximately 2% higher classification accuracy. When evaluating the classification performance of the models in more depth, there is a marginal difference. Both models exhibit superior performance classifying Reddit posts belonging to the Montreal subreddit, each reporting F1 scores of 78%. Both models experienced similar degradation in the classification performance of posts belonging to the Paris and London subreddits, reporting F1 scores of approximately 73% (72% for SGD) and 70%, respectively. Posts belonging to the Melbourne subreddit, however, easily reported the lowest classification performance for both models, reporting F1 score of 64%. With the near identical F1 scores and average validation error, we used the kaggle accuracy as the deciding factor in our selection.

When analyzing the classification performance metrics, interesting observations on the performance of all four models in classifying posts from the different subreddits come to light. Concerning the Melbourne subreddit in particular, all four developed models reported the worst classification performance when classifying posts belonging to this sub-reddit. F1 scores ranged between 58% to 64%, with recall scores of 51%, 58%, 59%, and 59% for the Naive Bayes, Logistic Regression, SVM, and SGD models, respectively. Precision scores were significantly better at 69%, 67%, 71%, and 72% for the same models. This suggests that, concerning the Melbourne class specifically,

Naive Bayes						
Ave. Valid Error	Acc. Kaggle	Train Time (ms)	Class	Precision	Recall	F1-Score
0.359	0.650	147	Montreal	0.75	0.62	0.67
			Melbourne	0.69	0.51	0.58
			London	0.56	0.82	0.66
			Paris	0.64	0.64	0.64
Logistic Regression						
Avg. Valid Error	Acc. Kaggle	Train Time (ms)	Class	Precision	Recall	F1-Score
0.301	0.689	170	Montreal	0.79	0.74	0.76
			Melbourne	0.67	0.58	0.62
			London	0.64	0.75	0.69
			Paris	0.70	0.73	0.72
Linear Support Vector Machine						
Avg. Valid Error	Acc. Kaggle	Train Time (ms)	Class	Precision	Recall	F1-Score
0.284	0.711	118	Montreal	0.80	0.77	0.78
			Melbourne	0.71	0.59	0.64
			London	0.65	0.76	0.70
			Paris	0.71	0.75	0.73
Huber Loss SGD						
Avg. Valid Error	Acc. Kaggle	Train Time (ms)	Class	Precision	Recall	F1-Score
0.286	0.728	147	Montreal	0.80	0.77	0.78
			Melbourne	0.72	0.59	0.64
			London	0.65	0.76	0.70
			Paris	0.71	0.74	0.72

Table 1: Performance Metrics of Naive Bayes, Logistic Regression, SVM, and Huber Loss SGD on: average validation error and training time during 10-fold cross-validation, Kaggle test accuracy. Classification metrics based on confusion matrices shown in Fig.1.

the four models are pessimistic in classifying posts to this subreddit, accurately classifying a low number of posts to this class, but missing a large number of posts that should have been classified there. For the London subreddit, the opposite behaviour is observed: models reported recall scores much higher than precision scores. This suggests that models were more liberal in classifying posts to the London subreddit, identifying a high number of positive cases, but also falsely classifying many posts. Potential explanations of this behavior include: ineffective data shuffling resulting in imbalanced datasets or ineffectual text pre-processing of primarily English words (likely why the Paris and Montreal posts have better performance as they're primarily in French). Different training approaches can be used to improve the overall performance: training two distinct models on the primarily French and English posts, respectively, and combining their outputs (weighted averaging or selecting a classifier based on the main language of the text), applying different threshold probabilities for classifying the output, employing higher thresholds to improve precision and lower thresholds to improve recall.

For future work, we would want to shift our focus toward feature extraction and cleaning instead of tweaking model complexity. We often wasted time running experiments only to rerun them once an issue with the training data was found. A better emphasis on input data extraction outside of stopwords and lemmatization could yield even better results. On the model side, we think other models could be worth exploring such as random forest, KNN, and ensemble classifiers such as SK Learn's VotingClassifier.

6 Statement of contributions

(**Sam**) - Ran hyperparameter search on SVM, french language lemmatization in preprocessing, modified K-fold code to collect confusion matrix and classification metric data

(**Aman**) - Programmed Naive Bayes and K-fold cross validation, made preprocessing methods, ran hyperparameter search for Naive Bayes and Logisitic Regression

(**Aydin**) - Ran hyperparameter search for Logistic Regression, Linear SVC, SVC, and SGD models. Explored language partitioned datasets.