CSE 142 Machine Learning - Assignment II

Anushka Hada

10-25-2025

1 Introduction

This project explores important linear classification models such as **Perceptron**, **Logistic Regression**, and **Linear Support Vector Machine (SVM).** Each model is implemented, trained, and evaluated on two binary classification tasks: **Spambase** and **Language**.

The **Spambase** dataset contains many features regarding emails, including information like word frequencies of certain phrases that may appear in suspicious emails. The goal is to classify emails into two categories, Spam or Not Spam. The **Language** dataset contains short phrases in Dutch and English, with Dutch being labeled 0 and English 1. The goal is to distinguish between the two. In the end, the overall objective is to find the most optimal hyperparameter and compare the accuracy of the models.

2 Pre-Training Preparation

2.0.1 Tools used

The following libraries were used for implementing the models: **Pandas** and **NumPy** were used for data handling, while **Scikit-learn** tools were used for splitting data into training, development, and testing sets. It also provides classification models such as LinearSVC and LogisticRegression. Each models performance was evaluated using metrics such as accuracy score and confusion matrix. Also, **CountVectorizer** was used for the Language dataset to convert strings into vectors.

2.0.2 Spambase dataset

The spambase data was stored in the file spambase.data, while the corresponding column names were listed in spambase.names. Hence, the first step involved reading in the file and getting the column names. The next step involved the dataset being loaded using Pandas, and the retrieved column names were added as headers to the dataframe for easier analysis. After this, the data was split into training(80%), development (10%), and test(10%) sets. An equal proportion of Spam to Not Spam data was maintained.

2.0.3 Language dataset

The language dataset required more preproccesing before it could be used by the models. First, the Dutch and English phrases were kept in different files. Not only that, 40 extra phrases for each language were needed for the development and testing sets. The Dutch and English files were combined and labeled as 0's and 1's. This was assigned to the training set. The same process was repeated for the other 40 phrases. Once everything was combined and evenly split, the sets were shuffled to prevent bias.

The experiments conducted on the dataset required feature engineering to improve performance, as well as, convert text to vectors for easier processing by models. The 1st feature simply treats each word as a feature and counts the number of occurrences. This approach works well as Dutch and English do not have many words in common. The 2nd feature recognizes that Dutch contains more repeated vowels such as "ee" or "uu". Hence, the number of repeated vowles were counted to test if this feature could distinguish the languages better. The 3rd feature simply tested if a combination of both could lead to a high accuracy.

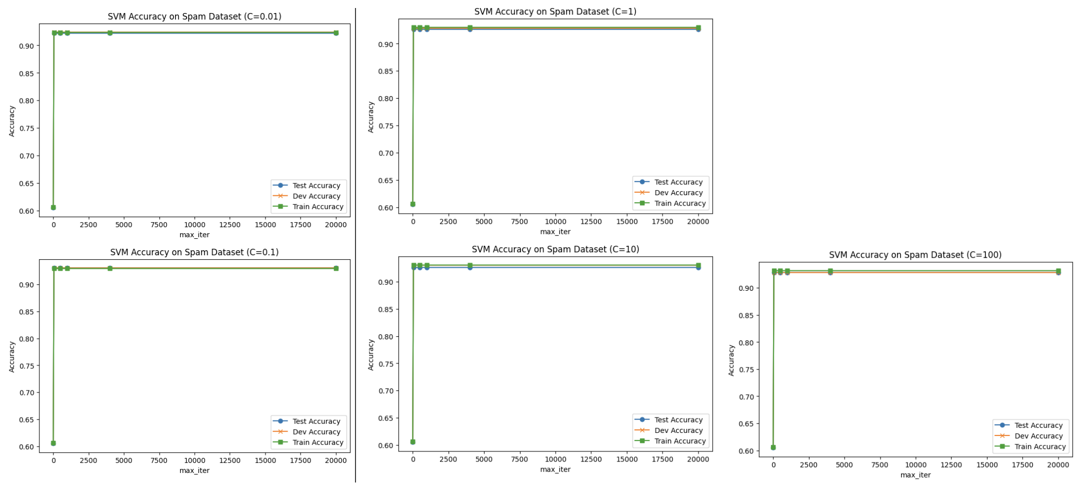
3 Models for Spambase dataset

3.0.1 Linear Support Vector Machine (SVM)

The Linear Support Vector Machine (LinearSVC) is fitted on the training set using the parameters max iteration and regularization parameter C. The parameter C was tested using the values 0.01, 0.1, 1, 10, and 100. Additionally, the maximum number of iterations was varied across 0, 50, 500, 1000, 4000, and 20000 to ensure the model converges. After training, the model was used to classify training, development, and test sets. The accuracy scores were calculated for all sets, while the confusion matrix was printed for the test set.

Based on the graph, and the full values printed during experimentation shows that across all regularization strengths C, the SVM quickly converges after a small number of iterations(50). The performance stabilizes afterwards. When max iterations is 0 the model underfits, achieving only about a 60% accuracy. This is because it is too regularized and predicts mainly one class. For max iterations greater than 0 the test accuracy stabilizes to 92–93%. Furthermore, increasing C beyond 1 provides little improvement, and in some cases shows convergence warnings. This suggests that higher C values make optimization harder without increasing accuracy. The best hyperparameter for this model ended up being max iterations greater than 0 and C being 0.1.

SVM Accuracy for Different Regularization Strengths (C values)

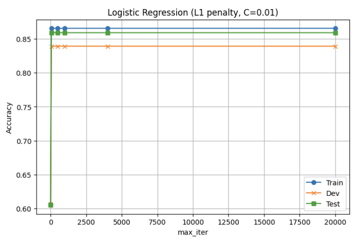


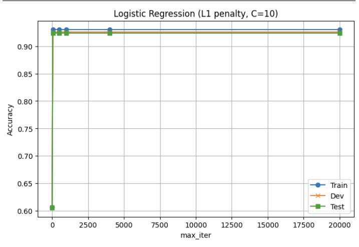
3.0.2 Logistic Regression

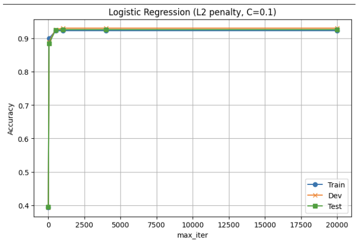
The Logistic Regression model was trained just like the SVM model. Two types of regularization penalties were tested, L1 and L2, with their corresponding solvers 'liblinear' and 'lbfgs'. The regularization parameter C was varied across values 0.01, 0.1, 1, 10, and 100 to test the effect of regularization strength on model performance. Furthermore, the maximum number of iterations was varied across 0, 50, 500, 1000, 4000, and 20000 to ensure proper convergence. After fitting the model, predictions were made on the training, development, and test sets. The accuracy was calculated for each set and the confusion matrix was generated.

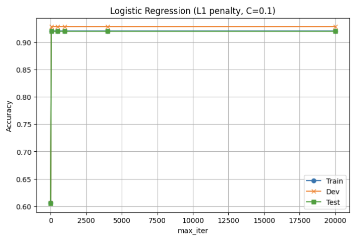
As shown in the graph below, L1 regularization performs worst at C = 0.01 for the development set, wheres L2 regularization remains consistent across the development, test, and training sets. The highest highest accuracy is achieved with L1 at C = 1 and L2 at most C values greater than 0.01. Both L1 and L2 show similar convergence trends and increasing the maximum number of iterations after 50 does not lead to significantly better results. Overall, C = 0.01 constantly under performs compared to other C values. Since the regularization increases with a smaller C value it leads to the model ignoring most features and underfitting.

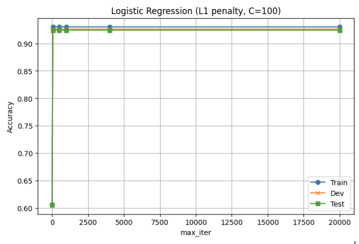
Logistic Regression Accuracy for Different Penalty and Regularization Strengths (C values)

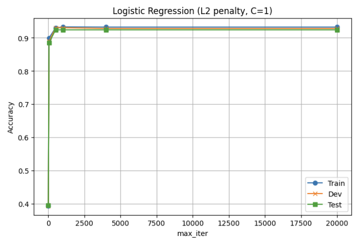


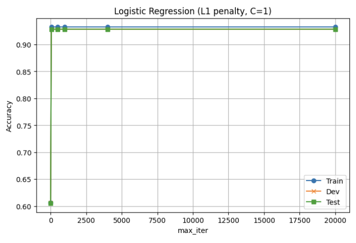


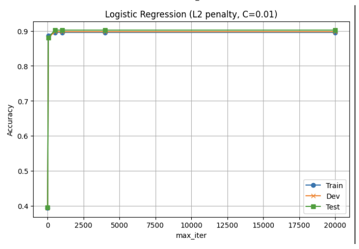


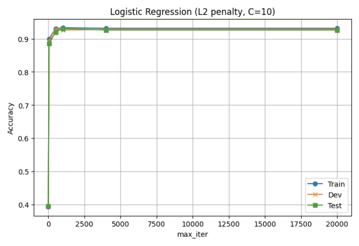


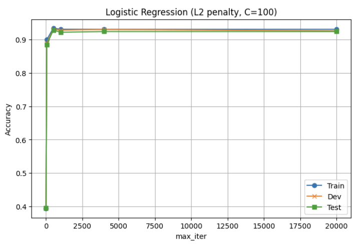










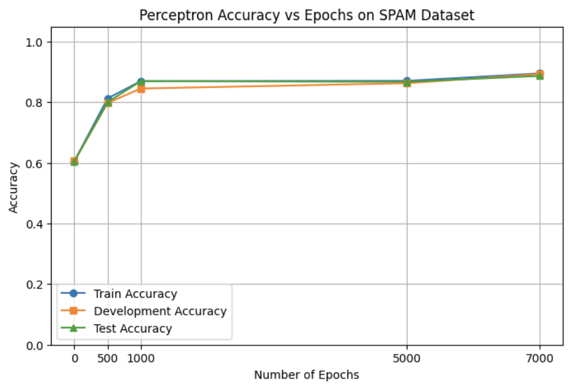


3.0.3 Perceptron Model

The Perceptron model is trained on the dataset just like other models. The model separates two classes by learning the weight vector w and the bias b. For an input x, the predicted label is 1 if w·x+b>0, otherwise it's -1. Labels are converted to 1 and -1 to simplify updates. When a point is misclassified, the weights and bias are adjusted to pick the correct class. Training happens over multiple epochs with shuffled data. After training, the model returns the accuracy scores for the training, development, and test sets. The confusion matrix is calculated for the test set using the original 0 and 1 labels. The experiment is conducted over multiple epochs with values being 0, 500, 1000, 5000, and 7000. This is to observe if performance improves or stabilizes over time.

Based on the plotted data, the model shows clear learning progress as the number of epochs increases. At 0 epochs, the model predicts mostly one class. Yielding around 60% accuracy across all sets. By 500 to 1000 epochs the accuracy improves significantly, reaching 82%. This shows that the model is beginning to learn patterns in the data. As the training continues to 5000–7000 epochs the accuracy improves to 92%. This demonstrates good convergence and generalization. The confusion matrices also reflects this trend. Early epochs show many misclassifications for one class. Later epochs have significantly fewer errors for both classes. Overall, the Perceptron model effectively learned over time. Diminishing gains were seen as the model converges due to large epoches.

Perceptron Model Accuracy for Different Epoches



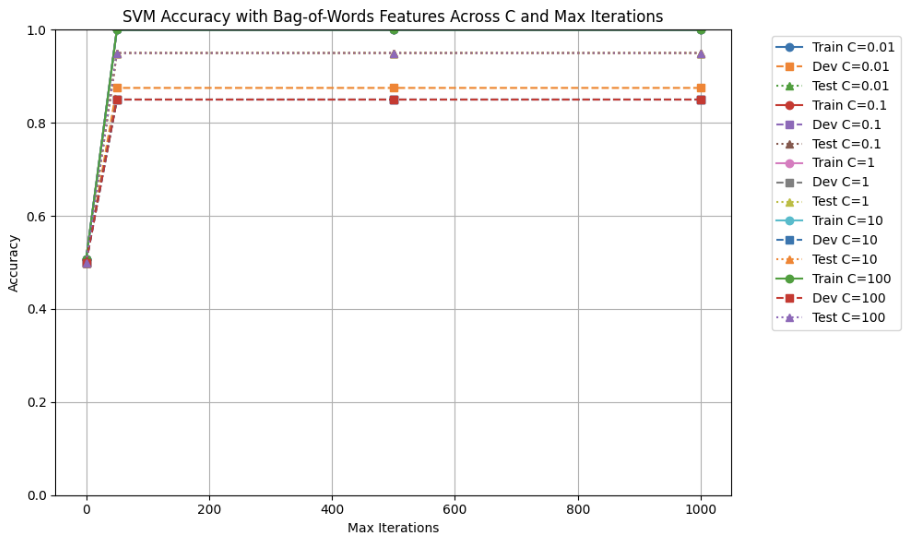
4 Models for Language dataset dataset

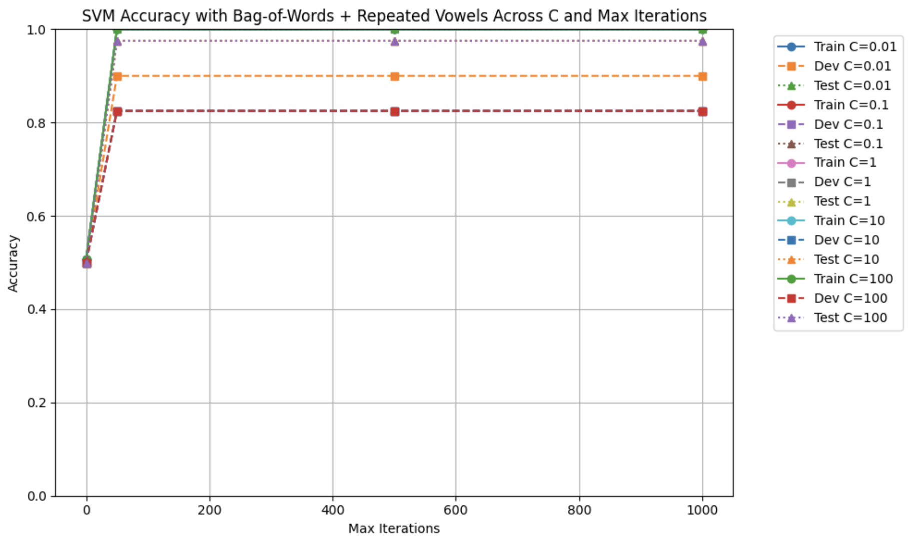
The models for the language dataset are the exact same as for the spambase dataset. Hence, only the experiments shall be discussed, taking into account the effects of different features on performance.

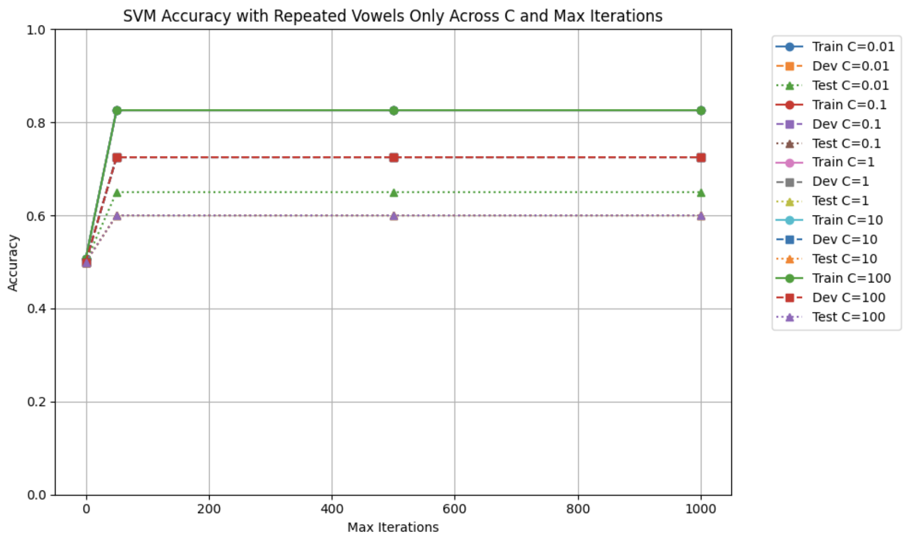
4.0.1 Graphs for Linear Support Vector Machine (SVM), Perceptron Model and Logistic Regression

The graphs are shown below:

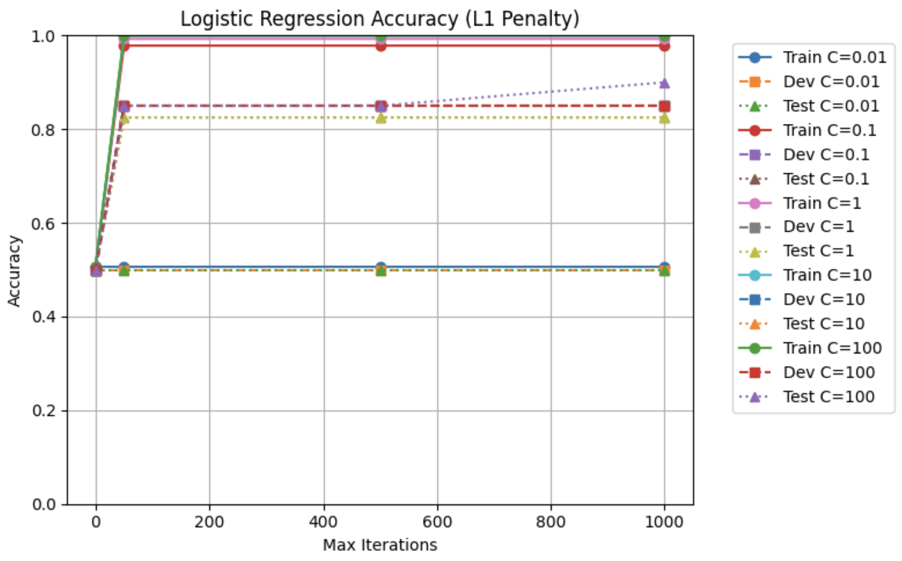
SVM Accuracy for Different Features

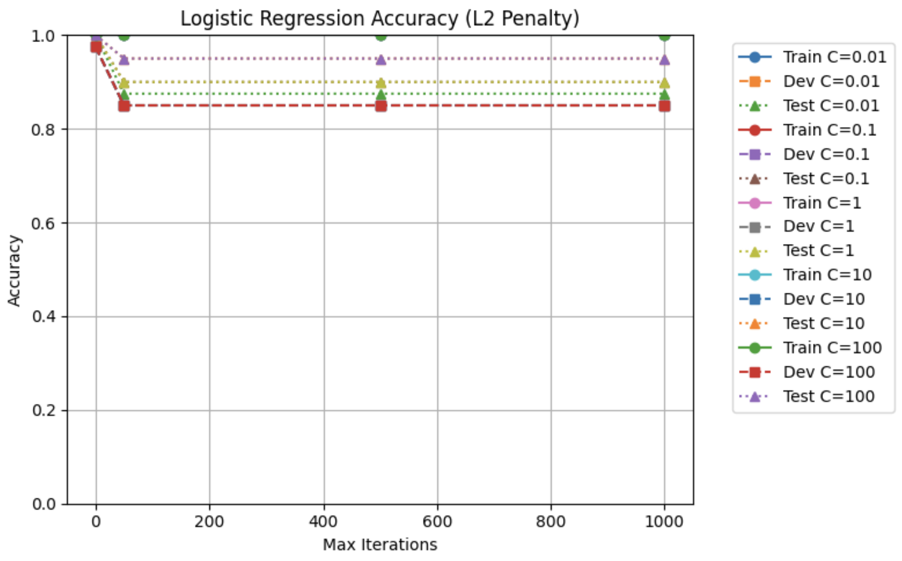




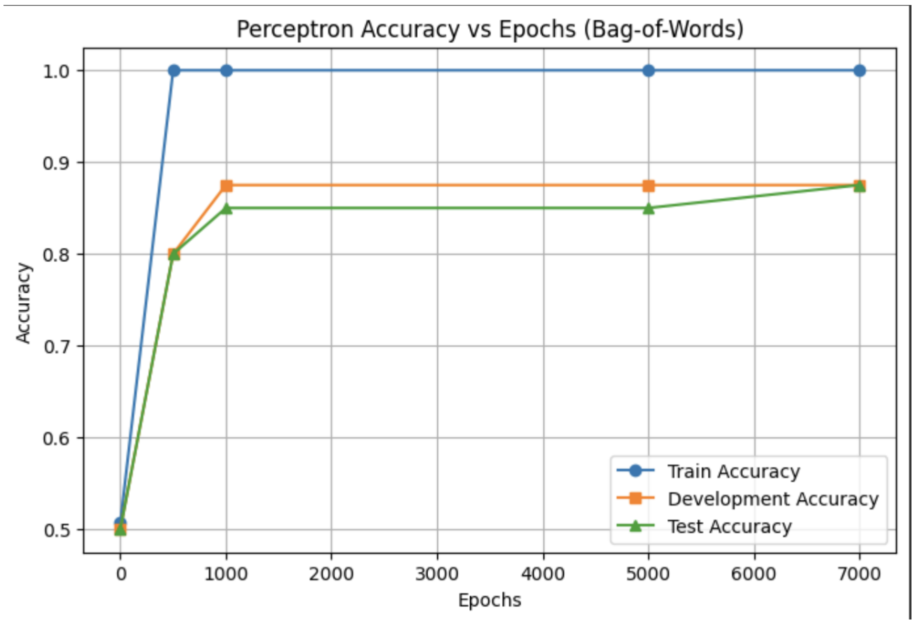


Logistic Regression Accuracy for Different Features on L1 and L2 penalty





Perceptron Model Accuracy for Different Epoches on Bag- of-Words Feature



4.0.2 Feature Engineering and Model Performance

For **SVM**, the results show that the performance of the feature, repeated vowels, is worse with only about a 60% test accuracy. In contrast the feature, word frequencies or bag-of-words, yields about a 97.5% test accuracy, while the combination of both is 95%. This indicates that word frequency provides a stronger basis for classification between Dutch and English, likely due to the fact that Dutch and English share fewer words in common compared to patterns like repeated vowels. The repeated vowels may not generalize well if the training, development, and test set do not contain enough Dutch phrases with repeated vowels to make it a viable option as a high performing feature. Overall, though word frequency is a simpler approach, it has proven to be the best performing. High accuracy was seen across all regularization C values and max iterations greater than 0.

For **logistic regression**, the graph shows the bag-of-words feature being tested against L1 and L2 penalties. For L1, the regularization parameter C at 0.01 results in a consistently worse accuracy of 50% for all max iterations. The model soon converges between 0 and 200 max iterations, combined with the regularization value C being greater than 0.01, leading to an accuracy of 80% to 87% for the test set. The best parameter with the highest accuracy for L1 resulted in 87.5% at C = 100.

Moving onto the L2 penalty, the graph depicts less variation between the test accuracies at just 87% to 97.5%. The accuracies for L2 penalty was higher across all C values at max iteration 0. This behavior is different from the other models, with max iteration 0 usually being the worst performing. Overall, the graph suggests that L2 penalty provides more consistent results, higher accuracies, and better generalizations. One interesting thing to note is that the confusion matrix for L1 and L2 always resulted in no false positives, English phrases were never misclassified as Dutch. The reason for this phenomenon is unknown.

For the **Perceptron Model**, a single graph was produced depicting the bag-of-words feature or word frequencies over multiple epoches. As can be seen in the graph, the test accuracy quickly rises after 500 and stabilized after 1000 epoches, with little to no improvement afterwards. At 0 epoches, the model has a 50% accuracy, revealing that little learning was happening with most classifications being left to chance. The best performance was shown at 500 and 1000 epoches which had a 90% test accuracy. This demonstrates that the bag of words feature effectively captured the difference between English and Dutch phrases.

4.0.3 Conclusion

In conclusion, the experiments conducted on the three models using the two datasets, Spambase and Language, revealed several trends. For the Spambase dataset, a maximum iteration greater than 0 and C greater than 0.01 were necessary for proper convergence and strong performance across all models. Overall, the accuracy was similar across all models, ranging from 90% to 93%, with the SVM performing slightly better than the others. For the Language dataset, the Perceptron reached 90% accuracy, SVM 92%, and Logistic Regression achieved the highest accuracy at 97%. Logistic Regression demonstrated better binary classification abilities. The most effective feature was the simple word frequency of all words in the dataset. Across both datasets, the Language dataset showed the highest overall accuracy. Analysis of the confusion matrices indicated a general bias towards English, with models classifying English phrases more accurately than Dutch. In the future, the hyperparameter shall be tuned further to account for the confusion matrix. Currently mostly the accuracies of the test, development and training sets were considered for issues such as underfitting or over fitting.