Final Project Written Report

CS-UY 4563

2:00PM Section

05/02/2022

Prof. Linda M. Sellie

Zijie Xiong, Aditya Dheer

**Introduction**

The data set that we used contains 10 years of daily weather observations from many locations across Australia. This dataset contains information about whether based on the current weather conditions, it rained the next day or didn’t. The data, when downloaded, hasn’t been separated into a training set and validation set, and contains a total of 145,461 records in total with 23 different features.

The goal of this project is to utilize logistic regression, support vector machine (SVM), and neural networks to predict whether it will rain the next day or not. This would involve yielding the highest testing accuracy, as opposed to the highest training accuracy. Each model was tested multiple times with varying hyperparameters in order to find the ones that led to the greatest accuracy on the test set.

**Preprocessing**

The data were provided to use in a csv file that contains one sheet with 23 columns. The first column indicates the date of the record, and the second column indicates the region of the record. The last column indicates whether the next day of the record rained or not. And the columns between them are the weather observation data. Every line contains the recorded data of weather observations of a singular region on a specific date.

In order to preprocess the data for training models, we took a number of steps. Firstly we separated out the last column as the target variable, and drop out the first and the second column to exclude them in features. We also remove all the columns pertaining to the wind direction, since including them didn’t impact our observations significantly. For other columns with null value, we use the mean value of that column to replace it. Then we separated the data into two sets, a training set containing 105,590 lines of records, and a test set containing 35,197 lines of records. And lastly we created a standard scaler using “sklearn.preprocessing”, and put all the data set we created above in it to fit them to the model.

**Logistic Regression**

The first model that we ran was Logistic Regression. This was done using the implementation provided by the “sklearn.linear\_model” library. We created a total of 26 different models, with varying regularization and C-values. The first was the base model with no penalty term and thus no regularization. Then, both L1 and L2 regularization were tested, each with the following C-values: [0.01, 0.1, 1, 10, 100, 1000]. The models were also run after changing the prediction threshold to the following: 0 if probability(y=0) > 80% else 1. The results for the same are shown below.

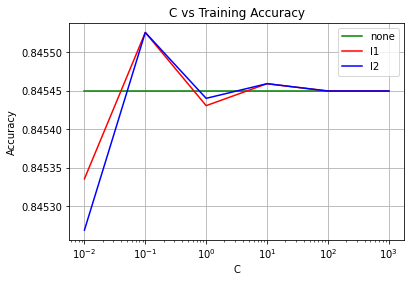


Figure: C-value vs Training Accuracy for Logistic Regression

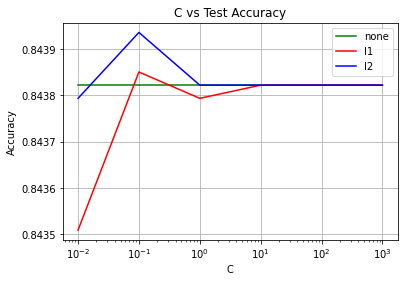


Figure: C-value vs Testing Accuracy for Logistic Regression

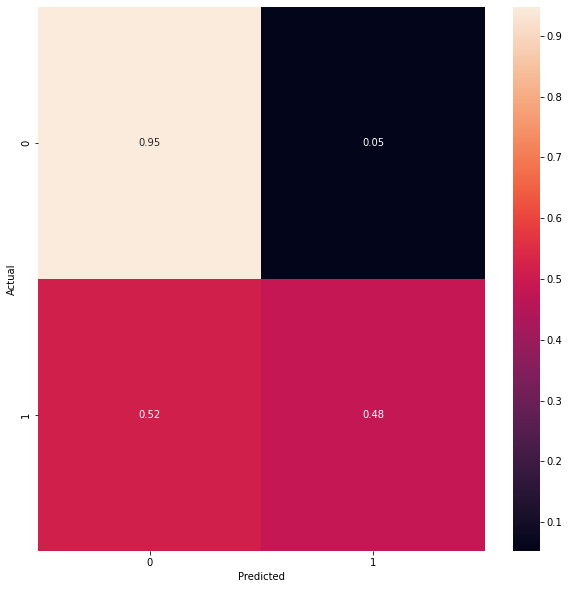


Figure: Normalized (per row) confusion matrix for Logistic Regression parameters with the highest test accuracy (Regularization = “l2”, C = 0.01)

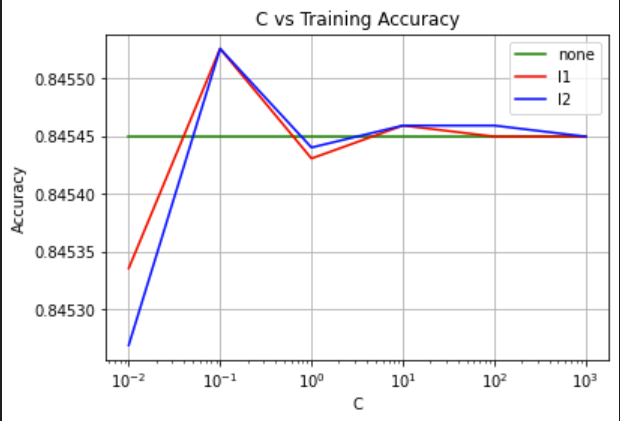


Figure: C-value vs Training Accuracy for Logistic Regression (lower threshold)

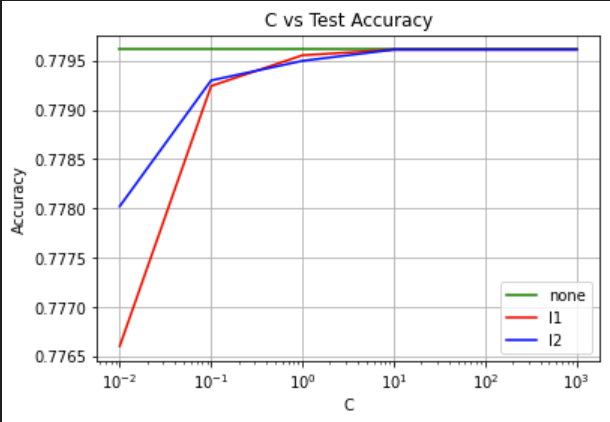


Figure: C-value vs Testing Accuracy for Logistic Regression (lower threshold)

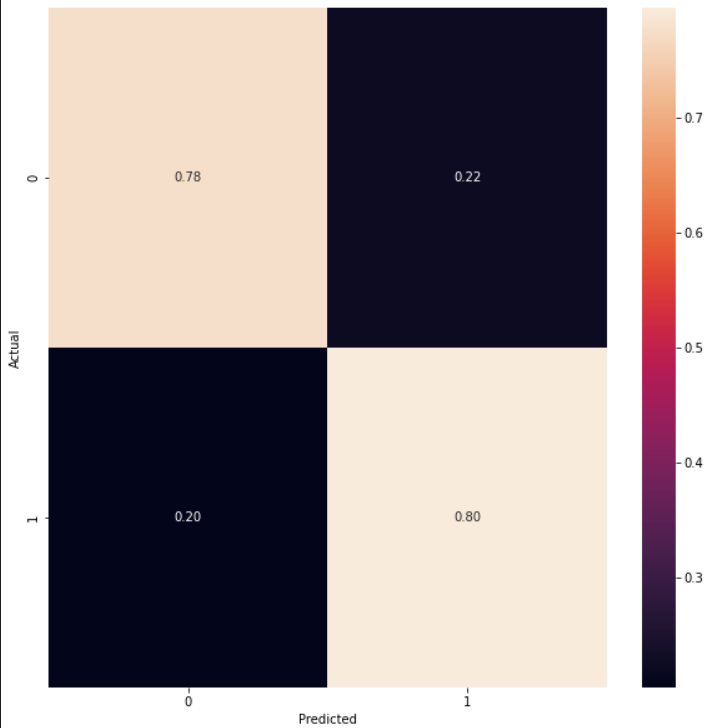
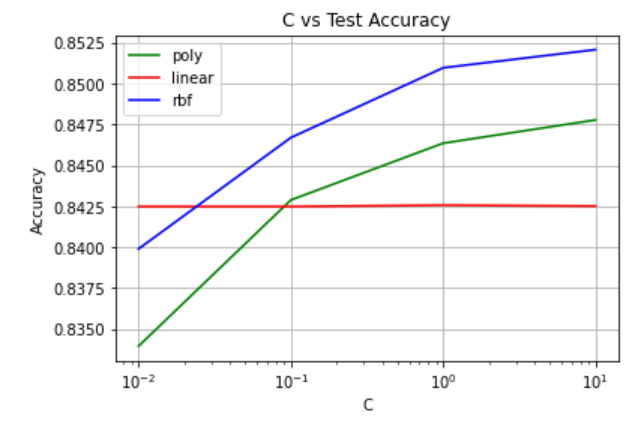
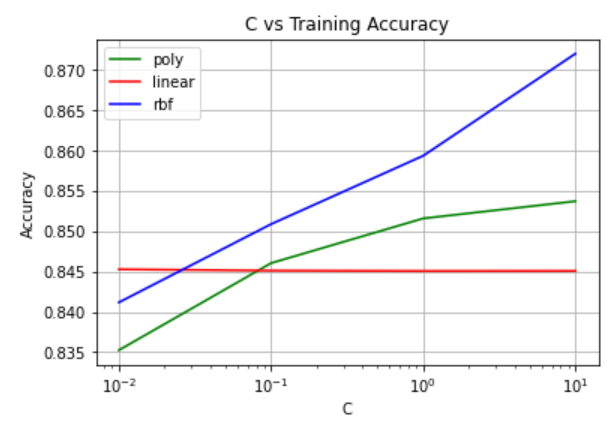


Figure: Normalized (per row) confusion matrix for Logistic Regression (lower threshold) parameters with the highest test accuracy (Regularization = “l1”, C = 10)

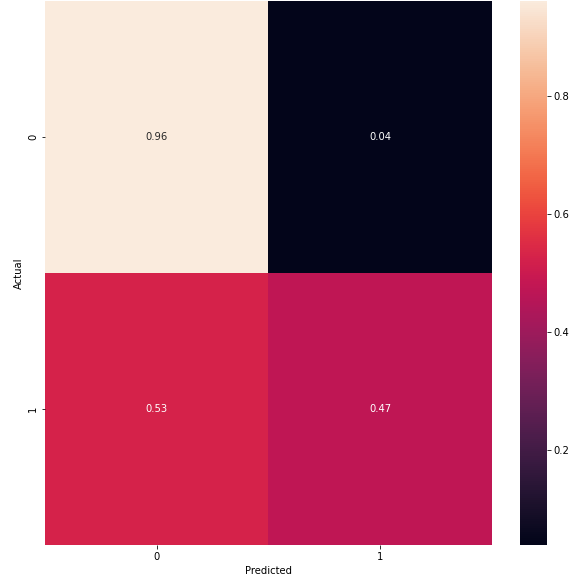
**Support Vector Machine(SVM)**

After the preprocessing steps, we created SVC object from the “sklearn.svm” library. We created 12 different models with varying kernel functions and regularization values. The three kernel functions we use are linera, polynomial, and radial-basis-funciton(rbf). The polynomial models were of degree 2. With each kernel, we used C-values of [0.01, 0.1, 1, 10] The results are shown below.



**Figure 6:** The training and test accuracy of SVM;

Key Describes the kernel used

Lin=Linear, Poly=Polynomial, RBF=Radial-Basis-Function

**Figure 7:** Normalized (per row) confusion matrix for SVM parameters with highest test accuracy (Radial-Basis-Function, C = 1)

**Neural Networks**

Lastly, after the same preprocessing steps, we created a neural network model using the implementation provided by the sklearn linear regression model. The shape of the neural network was set to [17,10,1] to appropriately train over the number of features in the dataset which is 17 and account for the 0 or 1 result depending on whether it would rain or not. The model was tested using 3 different activation functions: ReLU, tanh, and sigmoid. For each of the activation functions, 6 different alpha values were tried which were: [0, 0.001, 0.01, 0.1, 1, 10]. For each alpha value rate, we maintained constant parameters for the learning rate and the max number of iterations which were 0.01 and 2000 respectively. The solver used in this case was “sdg” which stands for stochastic gradient descent. The results are shown below. In total 18 different models were tried.

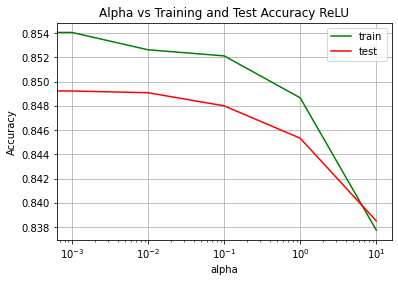


Figure: Alpha vs Training and Test Accuracy for ReLU activation function

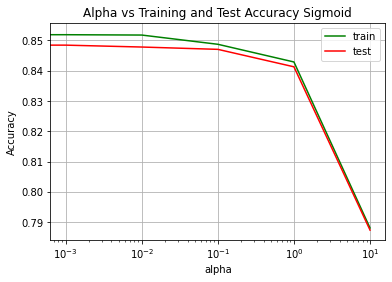


Figure: Alpha vs Training and Test Accuracy Sigmoid

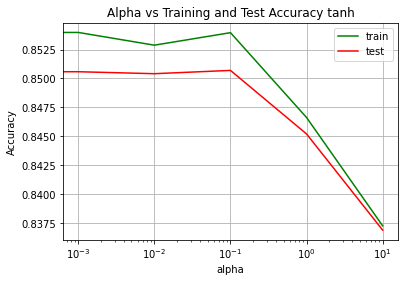


Figure: Alpha vs Training and Test Accuracy Tanh

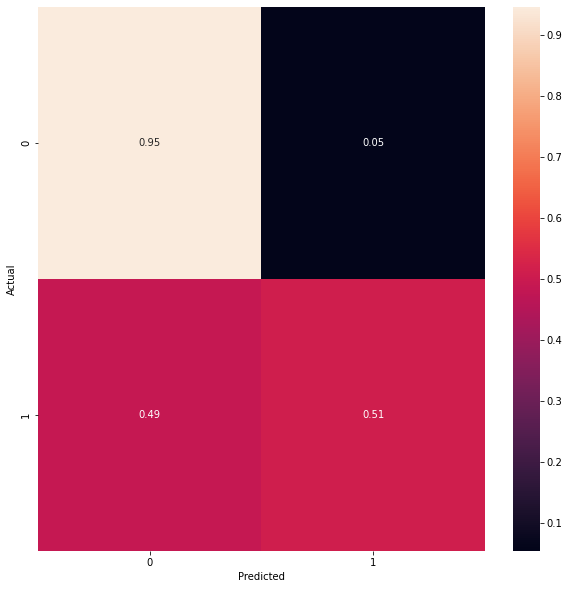


Figure: Normalized (per row) confusion matrix for neural network with the highest test accuracy (Regularization = “l2”, alpha = 0.1, activation = “tanh”)

**Table of Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0.01 | 0.1 | 1 | 10 | 100 | 1000 |
| None, train | 0.845449 | 0.845449 | 0.845449 | 0.845449 | 0.845449 | 0.845449 |
| None, test | 0.843822 | 0.843822 | 0.843822 | 0.843822 | 0.843822 | 0.843822 |
| L1, train | 0.845336 | 0.845525 | 0.84543 | 0.845459 | 0.845449 | 0.845449 |
| L1, test | 0.843509 | 0.84385 | 0.843794 | 0.843822 | 0.843822 | 0.843822 |
| L2, train | 0.845269 | 0.845525 | 0.84544 | 0.845459 | 0.845449 | 0.845449 |
| L2, test | 0.843794 | 0.843936 | 0.843822 | 0.843822 | 0.843822 | 0.843822 |

Figure: Table of Results for Logistic Regression;

Top row details the C values;

Leftmost column details the regularization type in the form (None, L1, L2);

Training rows are highlighted in Orange, test rows are highlighted in yellow, highest test value is highlighted in blue;

All values are rounded to 6 decimal places

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0.01 | 0.1 | 1 | 10 | 100 | 1000 |
| None, train | 0.845449 | 0.845449 | 0.845449 | 0.845449 | 0.845449 | 0.845449 |
| None, test | 0.779612 | 0.779612 | 0.779612 | 0.779612 | 0.779612 | 0.779612 |
| L1, train | 0.845336 | 0.845525 | 0.84543 | 0.845459 | 0.845449 | 0.845449 |
| L1, test | 0.776600 | 0.779243 | 0.779555 | 0.779612 | 0.779612 | 0.779612 |
| L2, train | 0.845269 | 0.845525 | 0.84544 | 0.845459 | 0.845449 | 0.845449 |
| L2, test | 0.778021 | 0.779299 | 0.779498 | 0.779612 | 0.779612 | 0.779612 |

Figure: Table of Results for Logistic Regression (lower threshold);

Top row details the C values;

Leftmost column details the regularization type in the form (None, L1, L2);

Training rows are highlighted in Orange, test rows are highlighted in yellow, highest test value is highlighted in blue;

All values are rounded to 6 decimal places

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1e-2 | 1e-1 | 1e0 | 1e1 |
| Linear, Train | 0.845260 | 0.845099 | 0.845052 | 0.845061 |
| Linear, Test | 0.842487 | 0.842487 | 0.842572 | 0.842515 |
| Poly, Train | 0.835240 | 0.846027 | 0.851567 | 0.853698 |
| Poly, Test | 0.833963 | 0.842884 | 0.846351 | 0.847771 |
| RBF, Train | 0.841159 | 0.850857 | 0.859333 | 0.871986 |
| RBF, Test | 0.839901 | 0.836601 | 0.850953 | 0.852061 |

**Figure 11:** Table of results of SVM;

Top row details C-values;

Leftmost column details model type in the form (Kernel, Train/Test);

Training rows highlighted in orange, test rows highlighted in yellow, highest test value highlighted in blue;

Poly = Polynomial, RBF=Radial-Basis-Funciton

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 0.001 | 0.01 | 0.1 | 1 | 10 |
| Train, ReLU | 0.853376 | 0.854049 | 0.852619 | 0.852117 | 0.84866 | 0.83775 |
| Test, ReLU | 0.850044 | 0.84922 | 0.849078 | 0.847998 | 0.845328 | 0.838509 |
| Train, sigmoid | 0.852647 | 0.851899 | 0.851785 | 0.848736 | 0.842892 | 0.788048 |
| Test, sigmoid | 0.848794 | 0.848453 | 0.847828 | 0.847061 | 0.841293 | 0.78734 |
| Train, tanh | 0.853878 | 0.853973 | 0.852874 | 0.853954 | 0.846633 | 0.837286 |
| Test, tanh | 0.849732 | 0.850584 | 0.850413 | 0.850698 | 0.845214 | 0.836918 |

Figure: Table of Results for Neural Networks;

Top row details Alpha values;

Leftmost column details activation function type in the form (ReLU, sigmoid, tanh);

Training rows are highlighted in Orange, test rows are highlighted in yellow, highest test value is highlighted in blue;

All values are rounded to 6 decimal places

**Conclusion**

First of all, for logistic regression, the overall accuracy that was achieved was pretty good (around 85%). However, looking at the confusion matrix we can see there are a disproportionate number of false negatives since 52% of the time the model predicted that it was not going to rain when in reality it did rain. The training accuracy in most cases is 0.002% higher than the test accuracy. It is expected that the training accuracy will be higher but such a slight difference doesn’t suggest any overfitting or underfitting. Even after trying both L1 and L2 regularization, only a marginal increase in the value of test accuracy was observed. Since no significant changes were observed after changing the hyperparameters, the only possible way to improve the accuracy would be to include more relevant features in the dataset and increase the number of entries.

When the logistic regression model was tested again with lower threshold to predict 1, there was a significant decrease in the number of false positives but there was around a 8% decrease in the test accuracy compared to the regular model. This means that the false negatives can be reduced but at the cost of overall test accuracy as it increases the number of false positives.

Next, for svm, the result seems divided. According to the confusion matrix, it is obvious that the model has an excellent performance in predicting non-rainy days, while it seems inconsistent in predicting rainy days. However, the model doesn’t appear to have a problem with excess bias or variance for the accuracy on the training set is always higher than the test set, and the difference is about 0.01. Thus, possible improvements of this model can be trying a different kernel function on the same data set or finding another data set with more and better features.

Last but not the least, for neural networks, 3 different activation functions were tried however there existed only a marginal difference between the test accuracies of the different activation functions for the same alpha values. The neural network with the highest test accuracy was the one using the tanh activation function with an alpha value 0.1. The neural networks also surprisingly yielded similar results to the logistic regression model and SVM, achieving the highest accuracy of around 85%.

Among all 3 models, all of them had the same problem of a high number of false negatives when predicting whether it would rain or not. This was, however, fixed to some extent by raising the threshold value in order to force the model to only classify as not raining if the probability is significant. If we look at only in terms of the test accuracy either of the three models could be used. However, if reducing the probability of false negatives is important, then the logistic regression model with a lower threshold for predicting 1 should be used, which has the accuracy of 77.9612%.

Works Cited

Rain in Australia Predict next-day rain in Australia:

<https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package>

Confusion Matrix:

<https://stackoverflow.com/questions/20927368/how-to-normalize-a-confusion-matrix>

Logistic Regression sklearn:

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

MLP classifier sklearn:

<https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html>

Replace missing observations with mean:

<https://thispointer.com/pandas-replace-nan-with-mean-or-average-in-dataframe-using-fillna/>