California State University, Long Beach

Computer Engineering and Computer Science Department

Weather Data Training in Australia

Submitted by

Jeff Kim

Abstract

The abstract is a *clear, concise, and complete* summary of the project, including the purpose, methodology, results, and major conclusions. Although the abstract is the first section that the audience reads, it is usually the last section that the author writes. The abstract should be one or possibly two paragraphs in length and appear on the title page.

The abstract will repeat information provided in the report. The abstract *is not* an introductory section of the report, but must be able to stand alone. Some people may read it and continue on with the report for details; others may never read the rest of the report. Your job is to entice the reader to continue reading the report, but summarize all key results in the abstract.

Introduction

Recently, the pattern of the weather has become inconsistent and erratic due to global warming and destruction of habitats. This led to frequent natural disasters and rapid increase in temperature and unpredictable weather patterns. As global warming accelerates and threatens the environment, the study of weather patterns has become essential and critical.

This research paper analyzes methods and models used to study weather pattern data which consists of weather pattern observed in Australia for approximately 10 years. It examines minimum temperature, max temperature, rainfall, evaporation, sunshine, wind gust speed at different times of a day, humidity at different times of a day, pressure at different times of a day, and cloud at different times of a day. Data consists of whether it rained on that day and on the next day.

This experiment's purpose is to study weather data and obtain better understanding of weather patterns to predict weather by using various classification algorithms such as Decision Trees, Support Vector Machines (SVM) under various kernels, Multi-layer Perception (MLP), Gaussian Naive Bayes, and Random Forest, Logistic RegressionCV. The report gives insight into how the data was preprocessed, and various models and training used to examine the weather data. To increase models' accuracy, hyperparameter tuning was applied in all models by using halvingGridsearch. By doing so, the best parameter is obtained to compare to previously designed model by showing classification report, area under receiving order charactersitics (ROC), confusion matrix, and classification report. These analysis and metrics prove which model has highest accuracy and how optimization affected the accuracy.

- a roadmap (overview, preview) of what information is provided subsequently in the individual report sections.

Problem Statement

In the process of predict the weather in the future, it is significant to analyze the pattern of the weather and find best way to predict the future weather. The project focuses to examine effectiveness of optimization techniques in several classifications and find best suited classification to predict the future weather.

Model Improvement technique description.

As mentioned earlier, halvingGridSearch was used to optimize all models. This technique works by setting parameter grid, which is a dictionary holding names as key and lists of parameters that the optimization will iterate through [1]. As shown in Fig 1 HalvingGridSearch will loop through 27 possible classifiers (n * n * n = 3 * 3 * 3) to find the best_estimator. In the figure, it tries different kernels, gamma, and C to find best classification.

```
param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01],'kernel': ['linear','rbf','sigmoid']}
tree = SVC(probability = True)
half_Grid = HalvingGridSearchCV(tree,param_grid,n_jobs=-1)
half_Grid.fit(X_train_scaled, y_train)
```

Figure 1HalvingGridSearch in SVC

Once the HavingGridSearch finds the best_estimator, one could find the parameter that belongs to it as shown in Fig 2

```
print("Half Grid Best Param: ", half_Grid.best_params_)
print("Half Grid Best Score:", half_Grid.best_score_)

best_model = half_Grid.best_estimator_
train_score = best_model.score(X_train_scaled, y_train)
test_score = best_model.score(X_test_scaled, y_test)
print("Best Model's Train Score:", test_score)
print("Best Model's Test Score:", test_score)
predict = best_model.predict(X_test_scaled)
```

```
Half Grid Best Param:  {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Half Grid Best Score: 0.8424005681818182
Best Model's Train Score: 0.8290455546869447
Best Model's Test Score: 0.8290455546869447
```

Figure 2Train/Test score and parameters of best_estimator of SVC

Experimental Results

For Decision Tree Classifier, the Train Accuracy and Test Accuracy was 1.0 (0.9999~) and 0.7479, respectively as shown in Fig 3. Also, the best_estimator's Classification report and confusion matrix is shown in Fig 4, followed by the ROC curve of it in Fig 5.

```
Decision Tree Train Accuracy : 0.9999556127657686
Decision Tree Test Accuracy : 0.7479567905621491
```

Figure 3Decision Tree Train/Test Accuracy before optimization

Classification report:

	precision	recall	f1-score	support
No	0.84	0.83	0.84	21866
Yes	0.44	0.47	0.45	6276
accuracy			0.75	28142
macro avg	0.64	0.65	0.64	28142
weighted avg	0.75	0.75	0.75	28142

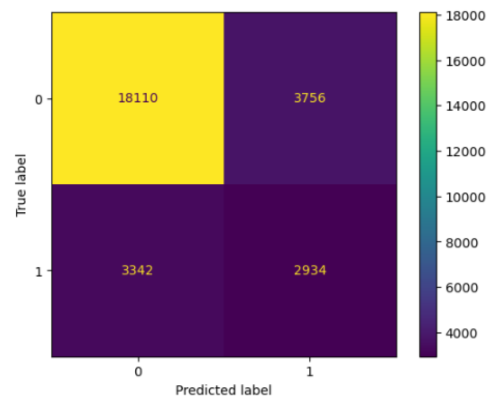

Figure 4Classification Report and confusion matrix of Decision Tree

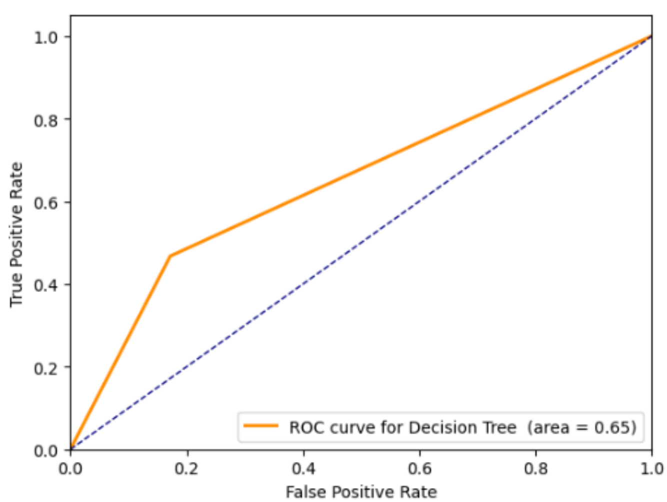

Figure 5 ROC curve of Decision Tree before optimization

```
Half Grid Best Param:  {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 10}
Half Grid Best Score: 0.8255351274535927
Best Model's Train Score: 0.8233601023381423
Best Model's Test Score: 0.8233601023381423
```

Figure6 Decision Tree after optimization

After optimizing the Decision Tree with HalvingGridsearch the Decision Train and Test accuracy are both 82.3360 as shown in Figure 6. Then, its classification report and confusin matrix is shown in Fig 7 and ROC curve in Figure 8.

Classification report:

	precision	recall	f1-score	support
No	0.85	0.94	0.89	21866
Yes	0.66	0.42	0.52	6276
accuracy			0.82	28142
macro avg	0.76	0.68	0.70	28142
weighted avg	0.81	0.82	0.81	28142

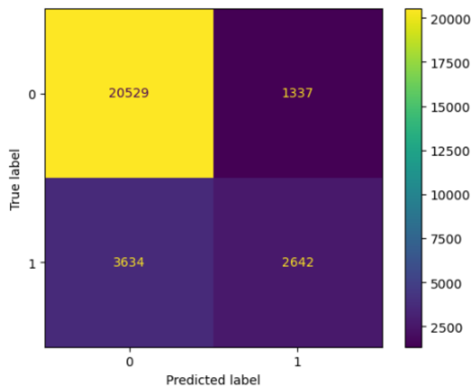

Figure 7Classification and Confusion Matrix for optimized Decision Tree

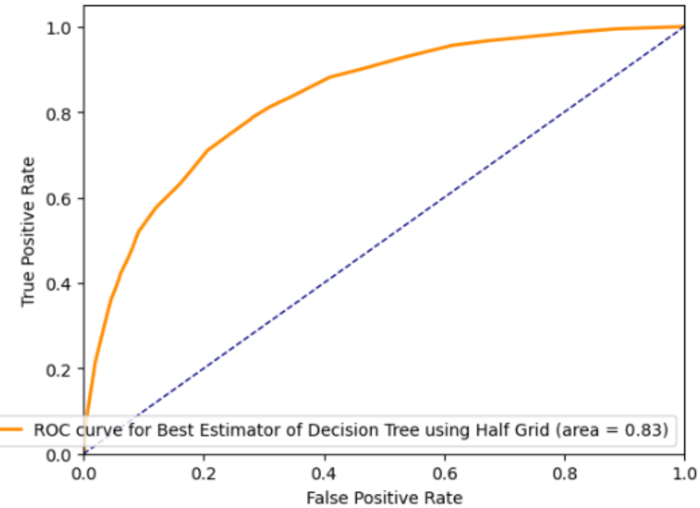

Figure 8 ROC curve for Optimized Decision Tree

For SVC classifier, it is using Linear as its kernel before the optimization and its Train and Test accuracy, 82.50 for both, is shown in Figure 9. Classification report and Confusion matrix in Figure 10 and ROC curve in Figure 11.

```
Linear SVM Train Score: 83.45599005725953
Linear SVM Test Score: 82.5030203965603
Linear SVM Accuracy: 82.5030203965603
```

Figure 9Train and Test Accuracy of SVM before Optimization

Classification report:

	precision	recall	f1-score	support
No	0.84	0.96	0.89	21866
Yes	0.71	0.36	0.48	6276
accuracy			0.83	28142
macro avg	0.77	0.66	0.69	28142
weighted avg	0.81	0.83	0.80	28142

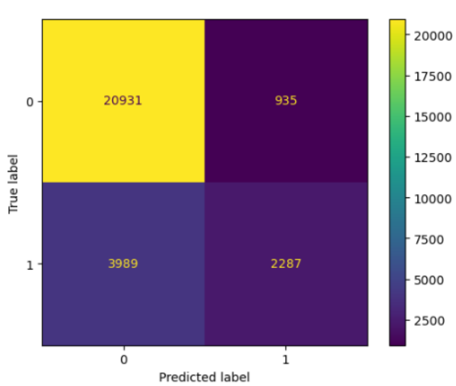

Figure 10 Classification report and Confusion maxtrix of SVM before optimization

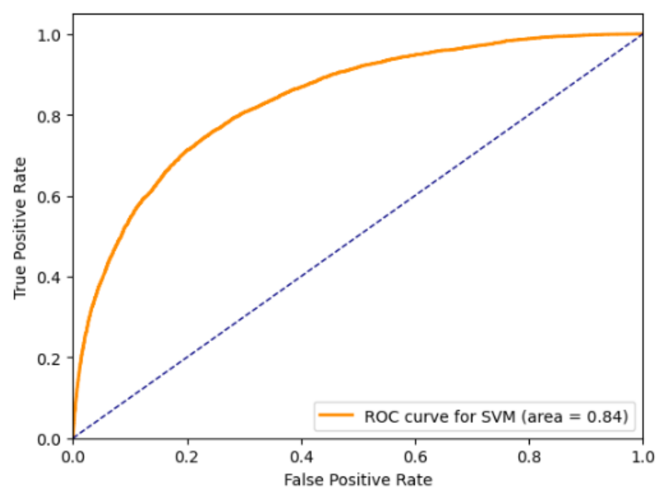

Figure 11 ROC Curve of SVM before optimization

After optimization of SVM, it prints the best_estimator's parameters and its accuracy in training data and test data (Fig11.1) Then, its confusion matrix and classification report is returned (Fig 11.2) followed by ROC curve (Fig 11.3).

```
Half Grid Best Param:  {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Half Grid Best Score: 0.8424005681818182
Best Model's Train Score: 0.8290455546869447
Best Model's Test Score: 0.8290455546869447
```

Figure 11.1 Best_estimator of svm's parameter and accuracy

```
Classification report:              precision    recall  f1-score   support

          No       0.84      0.96      0.90     21866
         Yes       0.71      0.39      0.50      6276

    accuracy                           0.83     28142
   macro avg       0.78      0.67      0.70     28142
weighted avg       0.82      0.83      0.81     28142
```

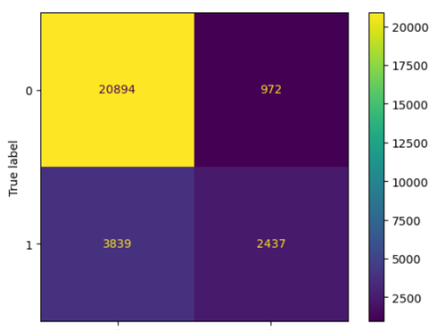

Figure 11.2 Confusion matrix and classification report of SVM after optimization

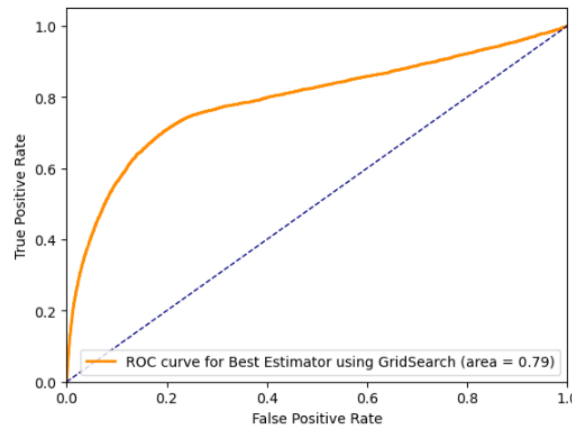

Figure 11.3 ROC curve of SVM after optimization

For Multi-layer Perception before optimization, Train and Test Accuracy was 84.7876 and 82.9827, respectively as shown in Figure 12. Its Classification report is shown in Figure 13 and ROC curve is shown in Figure 14.

```
MLP Classifier Train Score: 84.78760708420259
MLP Classifier Test Score: 82.98273043849052
MLP Accuracy: 82.98273043849052
```

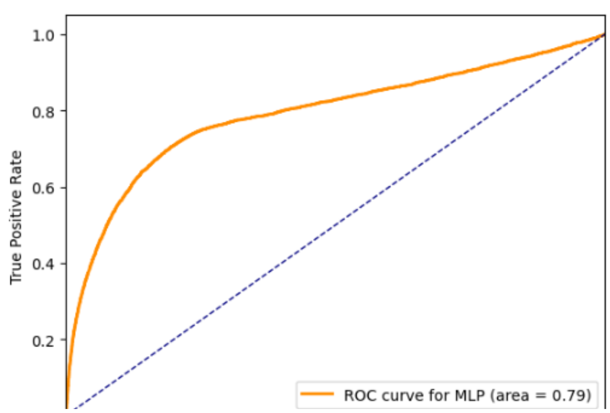

Figure 14 ROC Curve of MLP before optimization

```
Classification report:              precision    recall  f1-score   support

          No       0.85      0.94      0.90     21866
         Yes       0.68      0.44      0.54      6276

    accuracy                           0.83     28142
   macro avg       0.77      0.69      0.72     28142
weighted avg       0.82      0.83      0.82     28142
```

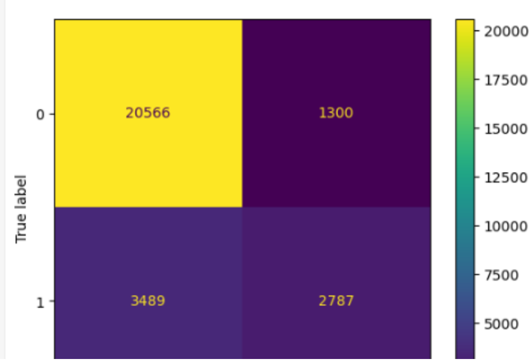

Figure 13 Classification report and Confusion Matrix before optimization

After optimization of MLP we retrieve the parameter that was used in best_estimator and get its train and test accuracy, 81.19 for both. Classification report

```
MLP Best Param:  {'alpha': 0.001, 'hidden_layer_sizes': (100, 50), 'learning_rate_init': 0.001}
MLP Best Score: 0.8377752130681818
Best Model's Train Score: 81.19536635633573
Best Model's Test Score: 81.19536635633573
MLP Best model Accuracy: 81.19536635633573
          No       0.86      0.90      0.88     21866
```

Figure 15 After optimization of MLP; Train and Test Accuracy and Params

```
weighted avg       0.80      0.81      0.81     28142
```

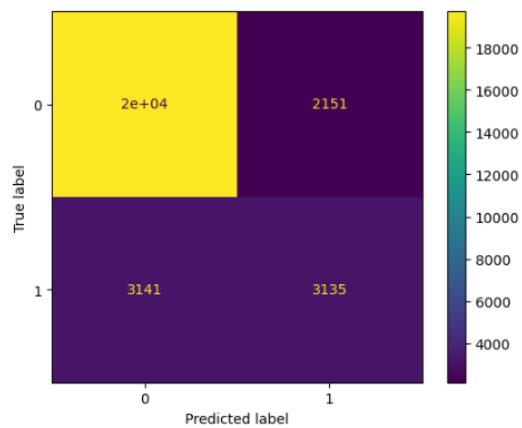

Figure 16Classficiation Report and Confusion matrix of MLP after optimizaiton

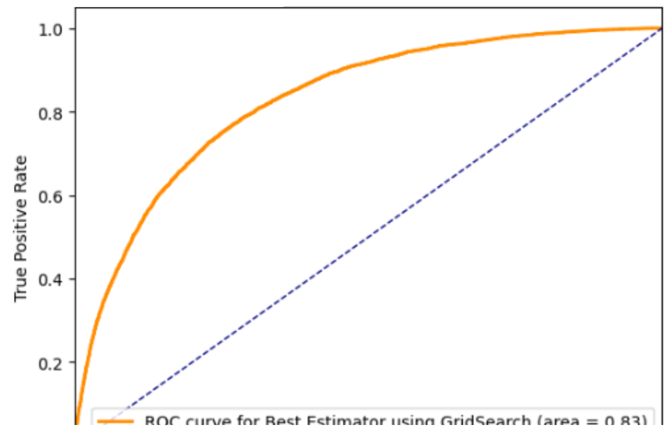

Figure 17 ROC curve of MLP after optimization

As for Gaussian Naïve Bayes, Train and Test Score was 81.11 and 80.25, respectively as shown in figure 15 and Classification report

```
Gaussian NB Train score:  0.8111145634515513
Gaussian NB Test score:  0.8025371331106531
```

Figure 15 Test and Train Score of Gaussian NB before optimization

```
Classification report:          precision    recall  f1-score   support

          No       0.85      0.91      0.88     21866
         Yes       0.57      0.44      0.50      6276

    accuracy                           0.80     28142
   macro avg       0.71      0.67      0.69     28142
weighted avg       0.79      0.80      0.79     28142
```

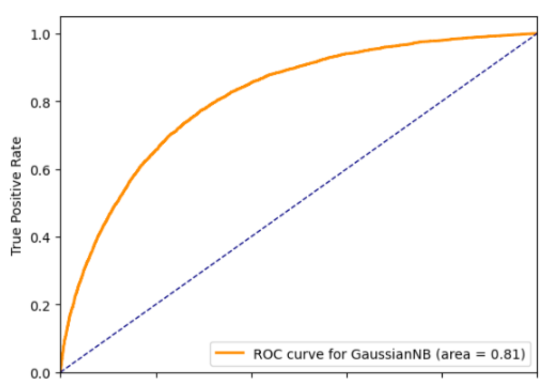

Figure 17 ROC curve of Gaussian NB before optimization

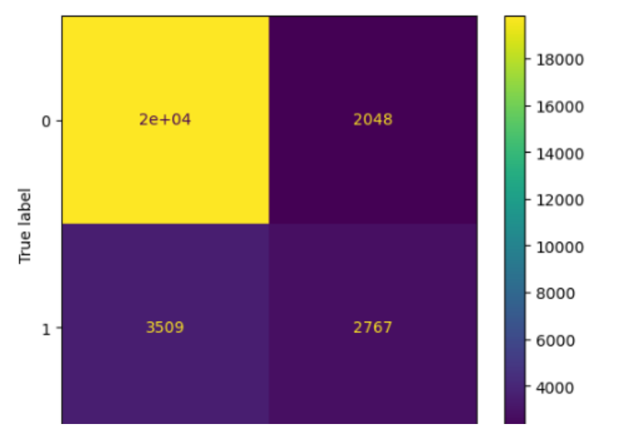

Figure 16 Classification and Confusion Matrix of Gaussian NB before optimization

After optimizing Gaussian NB we get train Accuracy and Test Accuracy of 80.25 by tuning ‘var_smoothing’(Fig 18). Classification report in Figure 19 and ROC curve is shown in Figure 20

```
GaussianNB Best Param:  {'var_smoothing': 1e-08}
GaussianNB Best Score: 0.8109108664772726
Best Model's Train Score: 80.25371331106531
Best Model's Test Score: 80.25371331106531
Best model Accuracy: 80.25371331106531
```

Figure 18 Train and Test accuracy and parameter after optimization of Gaussian NB

Classification report:

	precision	recall	f1-score	support
No	0.85	0.91	0.88	21866
Yes	0.57	0.44	0.50	6276
accuracy			0.80	28142
macro avg	0.71	0.67	0.69	28142
weighted avg	0.79	0.80	0.79	28142

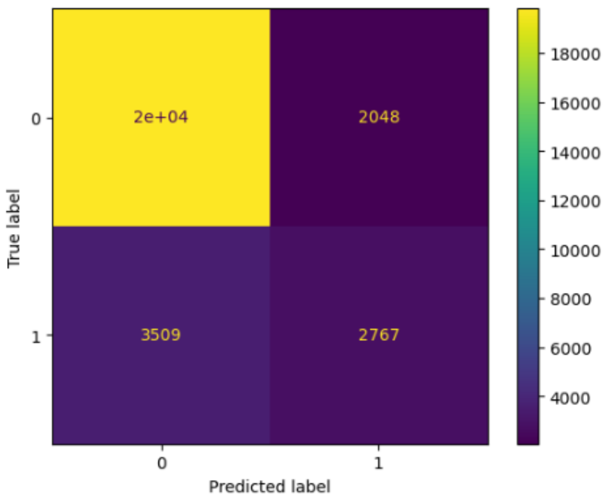

Figure 19 Classfication report and Confusion matrix after optimization of Gaussian NB

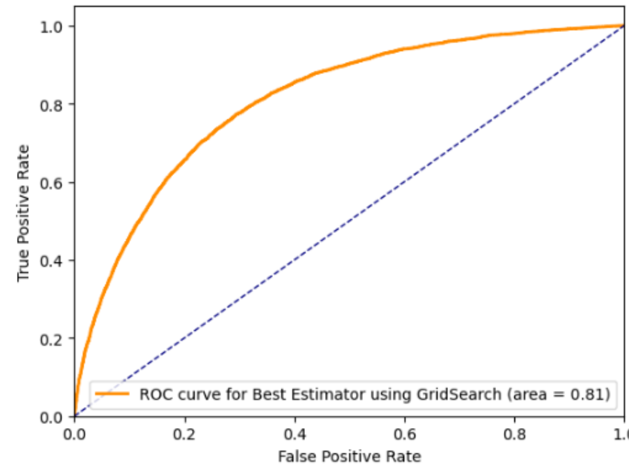

Figure 20 ROC curve after optimization of Gaussian NB

For RandomForestClassifier, its Train and Test Accuracy of is shown in Fig 21 followed by Classification Report(Fig22) and ROC Curve(Fig 23)

```
RandomForest Classifier Train Score: 99.94762306360691
RandomForest Classifier Test Score: 82.75175893682041
RandomForest Classifier Accuracy: 82.75175893682041
```

Figure 21 Tran and Test accuracy of RandomForest before optimization

Classification report:

	precision	recall	f1-score	support
No	0.86	0.94	0.89	21866
Yes	0.67	0.45	0.54	6276
accuracy			0.83	28142
macro avg	0.76	0.69	0.71	28142
weighted avg	0.81	0.83	0.81	28142

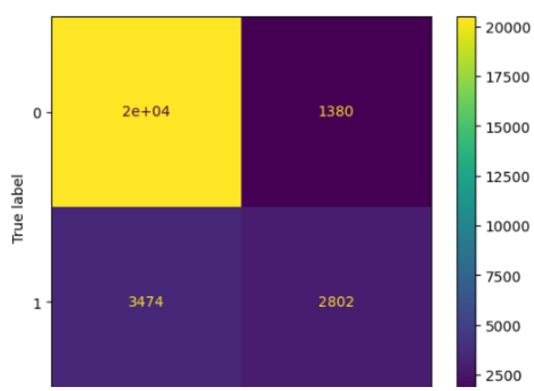

Figure22 Classfication Report and Confusion Matrix before optimization of RandomForest

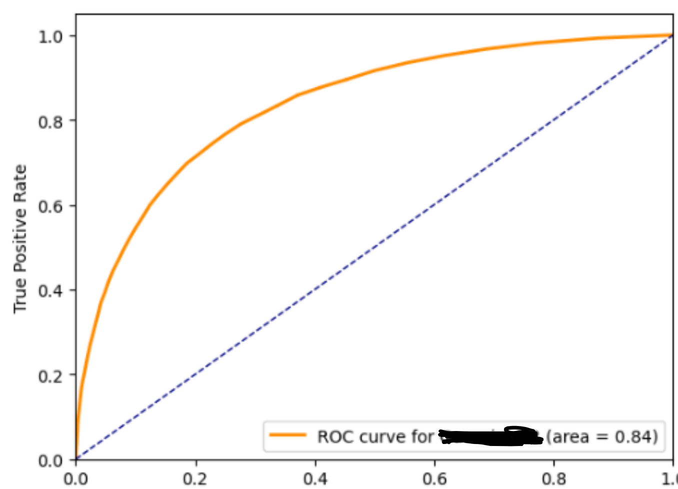

Figure 23ROC curve of Random Forest before optimization

After optimization of Random Forest Classifier, its best parameter, Train Accuracy, and Test Accuracy is returned (Fig 24). Followed by it's ROC Curve (Fig 26)

```
RandomForest Best Param:  {'max_depth': None, 'max_features': 'sqrt', 'min_samples_split': 20, 'n_estimators': 100}
RandomForest Best Score: 0.8421085353939072
Best Model's Train Score: 92.28727417994584
Best Model's Test Score: 83.11775993177457
Best model Accuracy: 83.11775993177457
```

Figure 24 Best Param and Train + Test Accuracy of Random Forest After optimization

Classification report:

	precision	recall	f1-score	support
No	0.86	0.94	0.90	21866
Yes	0.68	0.46	0.55	6276
accuracy			0.83	28142
macro avg	0.77	0.70	0.72	28142
weighted avg	0.82	0.83	0.82	28142

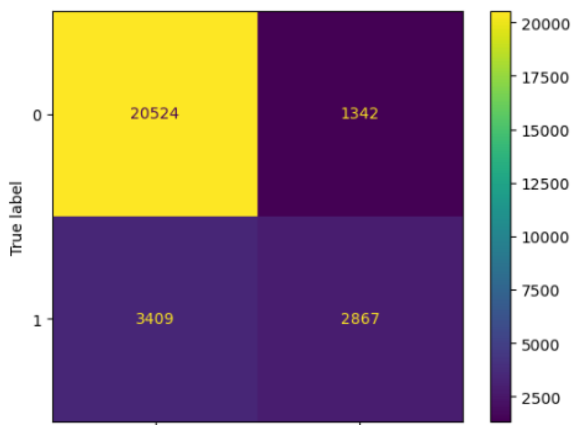

Figure 25 Classifcation and Confusion Matrix after optimization of RandomForest

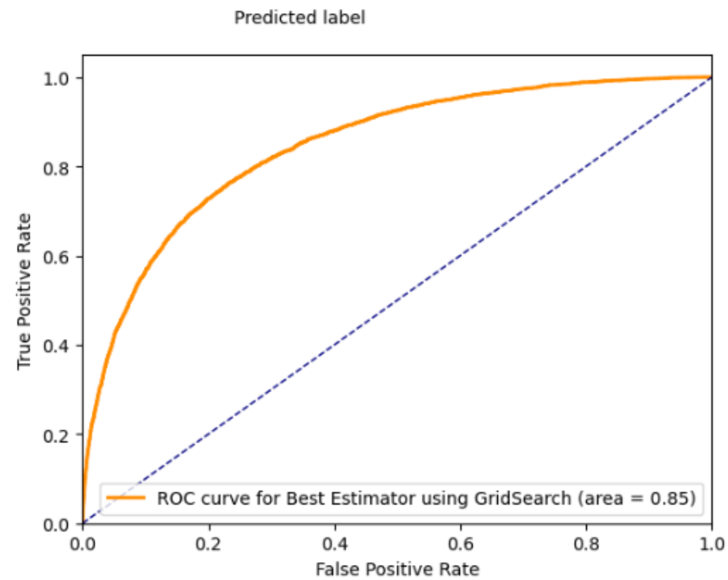

Figure 26 ROC curve for Random Forest After optimization

Lastly, Logistric Regression Classifier is used to train the data. Its accuracy is shown in Figure 27. Followed by Classification report and confusion matrix in Figure 28. Lastly, Figure 29 shows ROC curve of it.

```
LogisticRegression Train Score: 83.5509787385148
LogisticRegression Test Score: 82.7410987136664
LogisticRegression Accuracy: 82.7410987136664
```

Figure 27 Regression Accuracy before optimization

Classification report:

	precision	recall	f1-score	support
No	0.85	0.94	0.89	21866
Yes	0.69	0.42	0.52	6276
accuracy			0.83	28142
macro avg	0.77	0.68	0.71	28142
weighted avg	0.81	0.83	0.81	28142

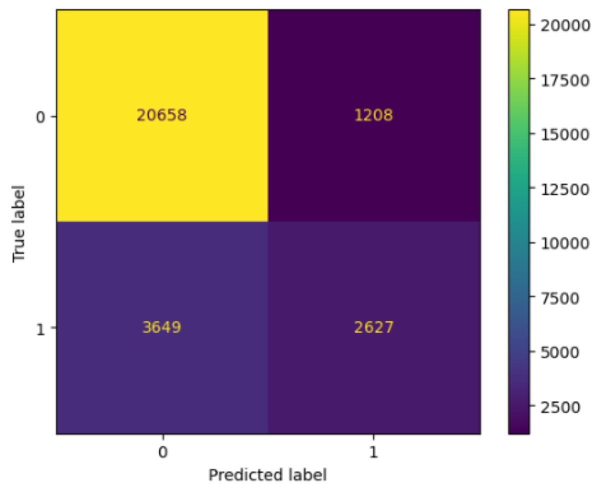

Figure 28 Classification report and Confusion matrix before optimization

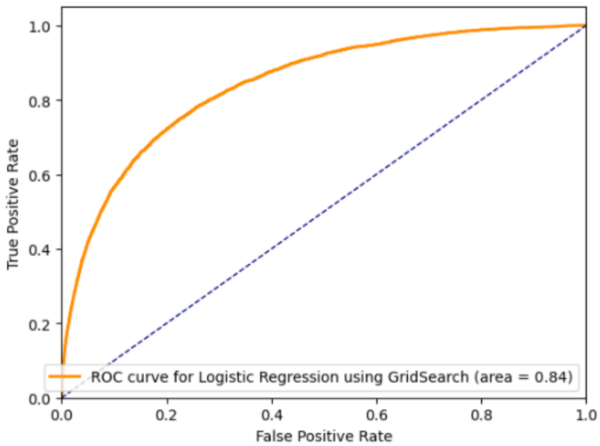

Figure 29 ROC curve before optimization

After optimizing Logistic Regression Classifier, we get the best_estimator's hyper parameter and its accuracy (Fig 30).

Classification report:

	precision	recall	f1-score	support
No	0.85	0.94	0.89	21866
Yes	0.69	0.42	0.52	6276
accuracy			0.83	28142
macro avg	0.77	0.68	0.71	28142
weighted avg	0.81	0.83	0.81	28142

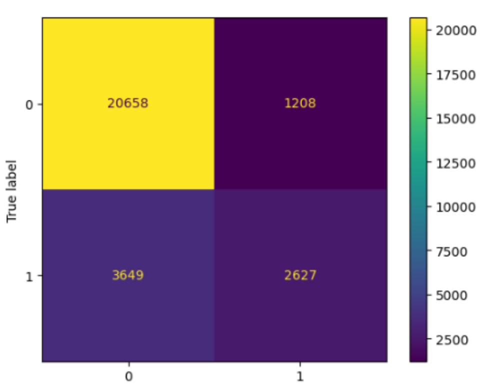

Figure 31 Classification report and confusion matrix of Logistic Regression after optimization

```
Logistic Regression Best Param:  {'Cs': 10, 'max_iter': 285, 'penalty': 'l2'}
Logistic Regressio  Best Score: 0.8354580965909092
Best Model's Train Score: 83.5509787385148
Best Model's Test Score: 82.7410987136664
Best model Accuracy: 82.7410987136664
```

Figure 30 Best_estimator's parameters and its accuracy after optimization

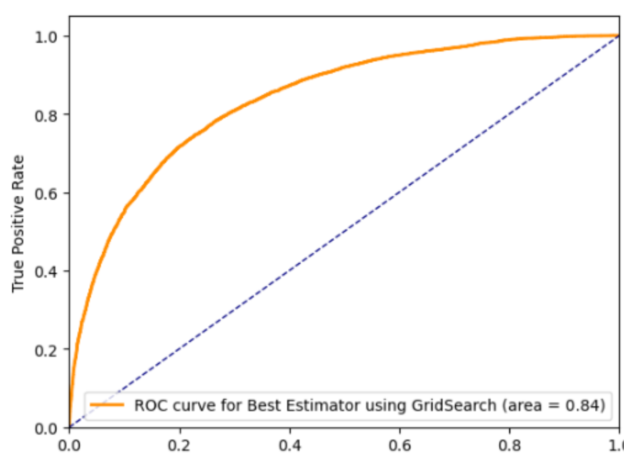

Figure 32 ROC Curve of Logistic Regression Classifier after optimization

Discussion/Analysis of Results

Overall, optimizing classification through hyperparameter tuning improved performance for most classification or performed similarly. The classification that stands out the most is Decision Tree. For Decision Tree, Confusion Matrix indicates that optimization increased True Positive by 13.2% and Classification matrix indicates that recall rate went up by 9%. Also, there was sharp decrease in false positive rate from 3743 to 1337 which is 64.3% decrease. The Area under ROC curve was increased by 27.6% indicating that optimization worked effectively. Most importantly, the decision tree before optimization had train score of 0.99, but 0.74 test score. This indicates that the tree is biased and memorized the training set. However, optimized decision tree had higher test accuracy showing that it works well with other data besides X_train.

Although the optimization significantly improved Decision Tree, it was not as effective in other classifications like MLP and RandomForest. For MLP, Area under ROC curve increased by 5%, true negative increased by 12.48%. Random Forest classification also had slight increase in its performance after optimization. For example, Area under ROC curve went up by 1.2%, accuracy_score went up by 0.005%, and True negative went up to 2867 from 2802 indicating increase in 2.3%.

On the other hand, for GaussianNB and Logistic Regression, the hyperparameter tuning had no effect in performance. This is result from the fact that the GaussianNB only has 'var_smoothing' parameter to tune. This indicates that hyperparameter tuning through halvingGridSearch is not effective and efficient for GaussianNB. As for Logistic regression, although it had multiple parameters to tune, it is hypothesized that param_grid is very similar to default value.

Unfortunately, the hyperparameter tuning did not work well in SVC classification. Although there was increase in the accuracy of estimator by 0.005% it is insignificant compared to the fact that area under ROC curve went down 6%, and false positive increased by 39%. This is caused by the fact that param_grid was limited to keep it small because of time restriction of the project and resources. However, the optimization would have significantly increased the performance if it iterated through more possible parameters.

Conclusions

In conclusion, hyperparameter tuning optimization was most effective in Decision Tree and Random Forest was most effective in predicting the weather with its highest accuracy, 83.11, and area under the ROC curve , 0.85.

References

1. *Sklearn.model_selection.HALVINGGRIDSEARCHCV*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html
2. GeeksforGeeks. *How decision tree depth impact on the accuracy*. https://www.geeksforgeeks.org/how-decision-tree-depth-impact-on-the-accuracy/,2024, February 26.