COMP9417
Machine Learning and Data Mining

# 9417 Project Report

**z5199568** Changming ZHU

**z5241987** Yunfei CUI

**z5234253** Jiabao JU

# Contents

# 1. Introduction

In this assignment, our team decided to implement a prediction of weather in Australia, especially on whether tomorrow will rain or not. The dataset comes from the Bureau of Meteorology of Australia and provided by Kaggle, and it mainly reports daily weather observations from numerous Australian weather stations. From the study of rainfall events from Ashcroft and other researchers, the changes in the weather regimes can be affected by several different variables, such as number of raindays, humidity, pressure total rainfall received. The weather data is also valuable after applying some data analysis because these records may have the ability to provide more useful information on variations in water availability and the frequency of potentially damaging rainfall events [1], as well as predict some irregular climate phenomenon like El Niño–Southern Oscillation. These are the main motivations for the work of this project presented by our team.

Our Major works:

- Data analysis. Providing raw data analysis for this dataset.
- Feature engineering. Processing data in order to set missing value, fix outliers and do feature selection. We choose to use features with high correlation value.
- Model Training. Providing reasonable evidence of model selection and train each model, then get evaluation scores of each model.
- Result. This part includes the performance and scores of each selected model.
- Evaluation. Analysing model results and potential improvements of this data.

**The data we use and all relevant code files are in the Google Drive folder via this link:**
https://drive.google.com/drive/folders/10jMo_xt8pM88U4ru99nluQHjEqfGYjZf?usp=sharing

# 2. Data Preprocessing and Feature Engineering

As for data analysis and feature engineering, the aim is to continue processing the data to create an efficient dataset. Before modeling, it is essential to check and analyse each column value in order to create an efficient dataset. In this dataset, there are 24 columns with both text and number data types. We also need to create extra features and handle data with the object type. In this section, we will check the correlation between all features and 'RainTomorrow' repeatedly to estimate whether to ignore some features or to group some features, and the relevant heatmaps are attached in the Appendix.

## 2.1 MVC (Missing Values Completion)

Dealing with missing values is the first thing we encounter when we view the dataset. The top values represent in the following table:

Table 2.1: Top Missing Values

| **Variable** | **dataType** | **MissingValueRate** |
|---|---|---|
| Sunshine | float64 | 47.7 |
| Evaporation | float64 | 42.8 |
| Cloud3pm | float64 | 40.6 |
| Cloud9am | float64 | 37.7 |

From the table above, there are 4 features missing a relatively large number of data thus have large missing value rate. We use 35% as a threshold to drop the columns which have missing values with higher than this threshold. As deleting data with missing values may cause a huge waste of data, missing values are often filled [2]. Therefore, we fill the data whose missing values lower than 35% with different methods, and categorical features and numerical features are dealt with separately.

Table 2.2: Solution of missing values for categorical features

| **Feature(MissingValueRate)** | **Solution** |
|---|---|
| WindGustDir(6.6%)<br>WindDir9am(7.0%)<br>WindDir3pm(2.7%) | Drop |

Table 2.3: Solution of missing values for numerical features

| **Feature ( MissingValueRate )** | **Solution** |
|---|---|
| Sunshine ( 47.7% )<br>Evaporation ( 42.8% )<br>Cloud3pm ( 40.6% )<br>Cloud9am ( 37.7% ) | Drop |
| Rainfall ( 1.0% )<br>WindGustSpeed ( 6.5% )<br>WindSpeed9am ( 0.9% )<br>WindSpeed3pm ( 1.8% )<br>Humidity9am ( 0.8% )<br>Humidity3pm ( 1.2% ) | Interpolation technique |
| MaxTemp ( 0.2% )<br>MinTemp ( 0.4% ) | If both of two values are missing, drop the row.<br>MaxTemp = MinTemp + mean<br>MinTemp = MaxTemp - mean |
| Temp9am ( 0.3% )<br>Temp3pm ( 0.6% ) | If both of two values are missing, drop the row.<br>Temp9am = Temp3pm + mean<br>Temp3pm = Temp9am - mean |
| Pressure9am ( 7.4% )<br>Pressure3pm ( 7.3% ) | If both of two values are missing, drop the row.<br>Pressure9am = Pressure3pm + mean<br>Pressure3pm = Pressure9am - mean |

## 2.2 Outliers

When we analyse the description of dataset, we can find 3 columns,*'Rainfall'*, *'WindSpeed9am'*, *'WindSpeed3pm'*, have small mean and standard deviation value. However, the max values of these three columns are quite larger than this mean and standard deviation, which means they may have outliers. We draw the histogram to identify them, the graph is attached in the Appendix. From the graph, the skewness of distribution of data distribution can be found, so we will find IQR (interquartile range) to discern outliers and make the correction. The range of outliers for *'Rainfall'*, *'WindSpeed9am'* and *'WindSpeed3pm'* are listed in the following:

Table 2.4: the range of outliers

| Feature | the range of outliers |
|---|---|
| Rainfall | < -2.4 or > 3.2 |
| WindSpeed9am | < -29 or > 55 |
| WindSpeed3pm | < -20 or > 57 |

From the table above, we have known that the maximum of normal values for *'Rainfall'*, *'WindSpeed9am'* and *'WindSpeed3pm'* which are 3.2, 55.0 and 57.0 respectively. Insetead of dropping all outliers, capping maximum values and remove the outliers is a better choice because we can keep the data size.

## 2.3 Other Issues

**Continuous Value Analysis**

Another important issue is marking and checking continuous value. From the graph in Appendix, we can see these four features use date as the horizontal axis. Therefore, when doing data processing, we decided to use the theory of CFS (Correlation-based Feature selection) to select features and apply this idea to fix missing value which was mentioned by Hall in 2000 [3].

**Feature Grouping**

After checking the correlation among all features, we decide to group two or three features having high correlation and create a new column. The process of grouping is shown in the following:

Table 2.5: Process of grouping features

| New feature | the way to group old features |
|---|---|
| temp_range | MaxTemp - MinTemp |
| mean_temp | ( Temp9am + Temp3pm ) / 2 |
| std_temp | \| Temp9am - Temp3pm \| |

As for *'Humidity9am'* and *'Humidity3pm'*, computing neither mean nor standard deviation can get better correlation, therefore, we keep the better one *'Humidity3pm'*. In the same situation, we keep *'Pressure9am'*. We plot the heatmap of the correlation matrix for all features, except *'Location'*, the performance is much better than the original dataset, which means the updated dataset is more efficient.

**One-hot Encoding**

It is challenging for algorithms to understand categorical features directly, so we choose to change them to numerical format. Checking the columns in our dataset, we notice that, except *'RainTomorrow'*, there is only two columns are categorical now, named *'Location'* and *'RainToday'*. However, from the heatmap, the correlation between *'Location'* and *'RainTomorrow'* is pretty low, so we dropped it. After encoding *'RainToday'*, two additional variables *'RainToday_0'*and *'RainToday_1'* are created from *'RainToday'* variable.

**Data Splitting**

There are two main components in creating a machine learning model: training and testing. Training is the foundation of machine learning, but testing is also as important. Before splitting the final dataset into training and testing datasets, we need to randomly shuffle it. This is because the dataset is currently sorted by data and store, which is systematic trend that we need to remove. In our test case, we choose a 80-20 split, meaning that training set comprises 80% of the final dataset while the testing set comprises 20%.

# 3. Model Training

Four classifiers which are Decision Tree, Random Forest, Logistic Regression and Support Vector Machine are trained and tested in this project. The training and optimizing process of each model is similar. First, we use base model to fit the train data, then analyse the appropriate way to optimize each model in order to achieve better performance. Finally, test optimized model, evaluate and examine whether the model is overfitting or not.

## 3.1 Decision Tree

As decision tree model performs well when predict and express any Boolean function. It also combines logical tests with a probability-based decision. This dataset combines numeric features and discrete attribute–value pairs, and the target function is discrete valued, so that this model may perform well and should be considered.

**step 1. Base model without parameter turning**

In this step, we did not adjust any parameters in the function model Decision Tree Classifier and fit data directly. However, we get the train_score is 1.0, test_score is only 0.77, also the bias is low and variance is high, thus overfitting may occur. From Andrew's research [4], the model, which is overfitted may perform perfectly on training data but is likely to perform poorly, and counter to expectation with the test dataset. This difference occurs in both classification and regression problems.

**step 2. Set and find the best parameters**

To achieve this, we define a "grid" of parameters that we would want to test out in the model and select the best model using 'GridSearchCV'. The following 4 parameters are tuned in the decision tree model:

- *“criterion”*: The function to measure the quality of a split.
- *“max_depth”*: The maximum depth of the tree. It should not be None in this case as the nodes are expanded until the tree has achieved this value.
- *“min_samples_split”*: The minimum number of samples required to split an internal node:
- *“random_state”*: Controls the randomness of the estimator. To obtain a deterministic behaviour and eliminate the randomness problem during fitting, this parameter has to be fixed to an integer.

**step 3. Test Normalization Influence**

After tuning parameters, the influence of normalization should be considered and tested in this step. The function StandardScaler() is applied to the training set and test set. Then, we fit the data and apply the model with the same parameters to get the result, and compare with accuracy score which generated without data normalization.

After normalization, the accuracy score of Decision Tree has no big change.

## 3.2 Random Forest

Random Forest is an ensemble method which could generate collections of trees using different subsets of the training data. Predictive accuracy is obtained by aggregating over the predictions of individual members of each decision tree classifiers[5]. This ensemble learning could also reduce variance by voting/averaging, thus reducing the overall expected error, and whether the datasets are all dependent or not is not crucial to the modeling process.

**Step 1. Compare with single decision tree model**

In general, the better performance of the fitting in a single decision tree model, the better performance will happen when fitting the same dataset in random forest model. Therefore, it is essential to compare score and performance in decision tree model with these value of random forest model. From the graph, the cross-validation score of random forest model is higher than decision tree model, which means the data fits better in this ensemble model. The graph of comparison attached in Appendix.

**step 2. Set and find the best parameters**

These 4 parameters, “criterion”, “max_depth”, “min_samples_split” and “random_state” are the same as decision tree model. The following parameter is what we need to tune:

- *“n_estimators”*: Defines the number of trees in the forest.

**step 3. Test Normalization Influence**

After normalization, the accuracy score of Random Forest has no big change.

## 3.3 Logistic Regression

Logistic Regression model is commonly used in a classification problem, and it is a supervised learning classification algorithm used to predict the probability of a target variable. The simplest form of logistic regression is binary or binomial logistic regression in which the target or dependent variable can have only 2 possible types either 1 or 0, the same as this dataset.

**step 1. Set and find the best parameters**

The following parameters are tuned in the logistic regression model:

- *"C"*: Inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization.
- *"class_weight"*: Weights associated with classes in the form. If not given, all classes are supposed to have weight one.
- *"Solver"*: Algorithm to use in the optimization problem, such as ‘newton-cg’, ‘lbfgs’ and ‘liblinear’.
- *"max_iter"*: Maximum number of iterations taken for the solvers to converge

**step 2. Test Normalization Influence**

After normalization, the accuracy score of logistic regression increases slightly, from 0.8425 to 0.8432.

**step 3. Adjusting the threshold level**

The default threshold is 0.5, we predicted it will rain tomorrow if probability $>$ 0.5. In this part, we will check that the threshold level should be increased or decreased. We plot the distribution of predicted probabilities which shows that the distribution is highly positive skewed, and only small number of observations predict that there will be no rain tomorrow. Therefore, we try to lower the threshold. After computing, however, we find that 0.5 get the best model performance.

## 3.4 SVM

Support vector machine is a learn linear classifiers. It can avoid overfitting by learning a form of decision boundary called the maximum margin hyperplane. In this dataset, SVM may perform well because it has different Kernel parameters and easy to compare performance of each kernel. This model can also be modified to deal with numeric prediction problems which related to this project.

**step 1. Base model with different Kernel**

In this step, three kernels, including "linear", "rbf", "sigmoid" are compared together. The accuracy values are 0.84, 0.78, 0.78. From these scores, we can find only "linear" kernel achieve the accuracy over 80%. The test of tune hyper parameters and normalization is necessary.

**step 2. Test Normalization Influence**

After normalization, the accuracy score of Support Vector Machine raised dramatically, from 0.8005 to 0.8467. Therefore, the improvement of normalization to rbf kernel SVM model is considerable, and we decide to use this kernel in the following step.

**step 3. Set and find the best parameters**

The most important adjustment of this model is allowing margin errors, which is to introduce slack variables $\xi_i$. This also results in the soft margin of SVM and allow limited misclassification. The constraints are changed to $w \cdot x_i - t \geq 1 - \xi_i$, and minimise the objective function in order to achieve soft margin optimisation, which shows in the following equation.

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w}^*, t^*, \xi_i^*}{\arg\min} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1 - \xi_i$ and $\xi_i \geq 0, 1 \leq i \leq n$.

- *“gamma”*: Get Kernel coefficient for ’rbf’, ’poly’ and ’sigmoid’.
- *“C”*: Regularization parameter, which can be also known as the coefficient of slack variable. A high value of C means that margin errors incur a high penalty, while a low value could achieve a large margin because it permits more margin errors[6].

## 3.5 Summary of Hyper-parameters

From the test of each result and observe output graph in Appendix, we can get the optimal hyper-parameters of each model. These models also have been tested to avoid overfitting.

Table 3.1: Optimal Hyperparameters

| **Model** | **Parameter** |
|---|---|
| Decision Tree | criterion = "gini", random_state = 30, max_depth= 8, min_samples_split = 7 |
| Random Forest | criterion = "gini", random_state = 30, max_depth= 8, min_samples_split = 7, n_estimators = 35 |
| Logistic Regression | C=1, max_iter=100, solver='newtoncg' |
| SVM | kernel = "rbf", gamma = 0.0359381366380464, cache_size = 5000, C= 12 |

# 4. Results and Evaluation

## 4.1 Results

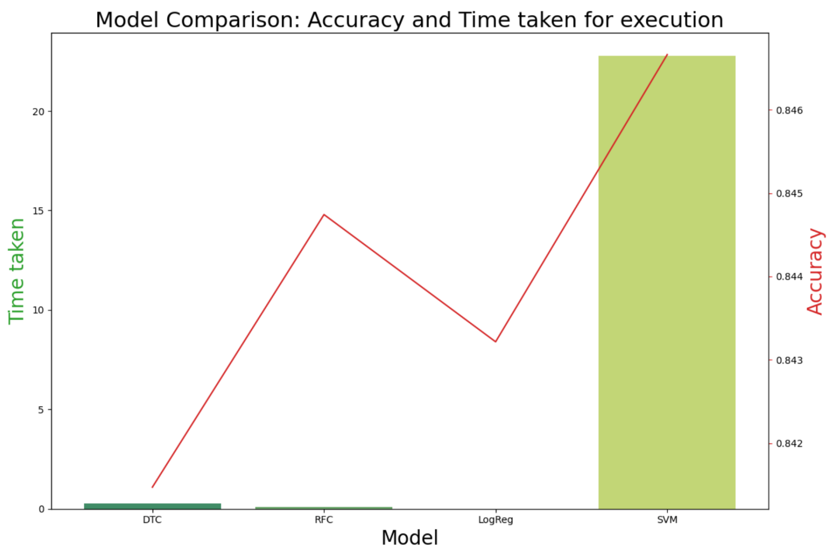
After training models and set appropriate parameters, it is important to compare test results horizontally among these models. We mainly test Misclassification rate, Precision rate, Recall rate, F1 and Test set Accuracy. In this case, the confusion matrix is introduced when calculating each score. Also, the formulations of each rate calculation and confusion matrix are both in Appendix.

The following table shows the results of each equation as well as the performance of each model with the best parameters respectively.

Table 4.1: The Comparision of Modeling Results

| | Score | | | | |
|---|---|---|---|---|---|
| Model | Precision rate | Recall rate | F1 | Train score | Test score |
| Decision Tree | 0.86 | 0.94 | 0.90 | 0.85 | 0.83 |
| Random Forest | 0.86 | 0.96 | 0.90 | 0.85 | 0.84 |
| Logistic Regression | 0.86 | 0.95 | 0.90 | 0.84 | 0.84 |
| SVM | 0.85 | 0.89 | 0.87 | 0.99 | 0.79 |

If we mainly consider the accuracy and speed, the following plot can show us the details about them.



## 4.2 Evaluation

In the result, the performances of these four models are stable, so the prediction result of test data is relatively reliable. We notice that these models are not overfitting as their test set accuracy score is slightly lower than the training set accuracy score. These models may worth to be used in weather report when the weather stations predict whether it will rain tomorrow.

However, there are some aspects that not be considered. Firstly, the objective-type data are underutilized, such as *Date* and *Location*. We tried to use date information to analyse the influences of time and season to raining but the correlation score is negative, which means this data is irrelevant to the value to be predicted. Time Seris Analysis should be used to experiment with these learning algorithms. Another experiment is the Neural Network model. It is not performed as expected when testing this dataset so that we abandoned it. The artificial neural network mainly use when processing graphic data or data vector, and it requires higher-dimensional input and the output may not have human readability[7], which conflicts with the purpose of this case. As for the target value of this dataset, we find that 'No' value is larger than 'Yes' value, it is not a balanced distribution. Therefore, the oversampling issue should be considered when we test data and evaluate accuracy score, and the recall rate of these 2 class is not the same as well.

# 5. Conclusion

In this project, we trained 4 machine learning classifiers: Decision Tree, Random Forest, Logistic Regression and Support Vector Machine. Each model performs similarly that they all achieve high accuracy around 0.8450. As for Decision Tree and Random Forest, we used the accuracy score without normalising features, since there is no big difference before and after normalisation. Concerning the rest of two models, we normalised all features firstly, then computed the accuracy score of them respectively. Among these four models, the SVM can achieve the highest accuracy score at 0.8480.

However, if speed is an important thing to consider, we would stick to RFC instead of SVM, since it took a long time for the SVM model, approximately 24 min, whereas less than 1 min for the RFC model.

## References

[1] Linden Ashcroft, David J. Karoly, and Andrew J. Dowdy. Historical extreme rainfall events in southeastern australia. *Weather and Climate Extremes*, 25:100210, 2019.

[2] Arnaud Ragel. Preprocessing of missing values using robust association rules. In Jan M. Żytkow and Mohamed Quafafou, editors, *Principles of Data Mining and Knowledge Discovery*, pages 414–422, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[3] Mark Hall. Correlation-based feature selection for discrete and numeric class machine learning. pages 359–366, 01 2000.

[4] Andrew W Moore. Cross-validation for detecting and preventing overfitting. *School of Computer Science Carneigie Mellon University*, 2001.

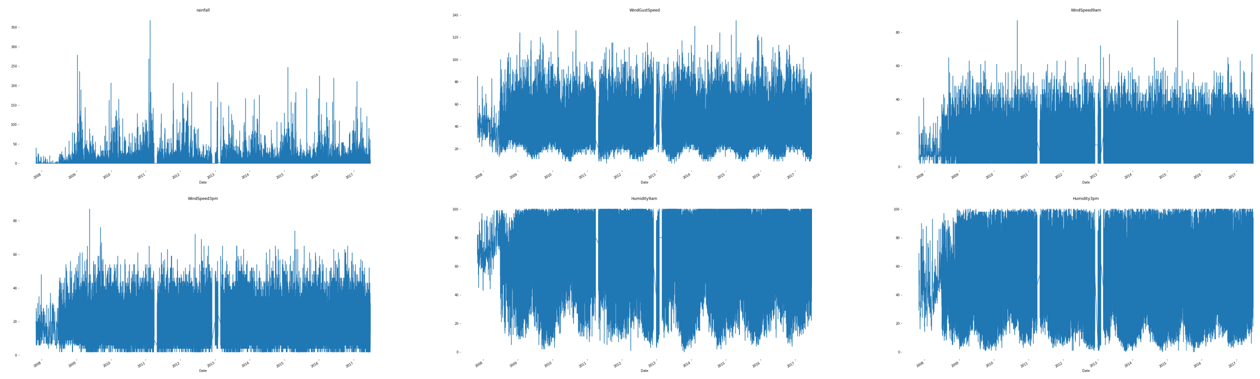
[5] Nan Lin, Douglas Noe, and Xuming He. *Tree-Based Methods and Their Applications*, pages 551–570. Springer London, London, 2006.

[6] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* 01 2018.

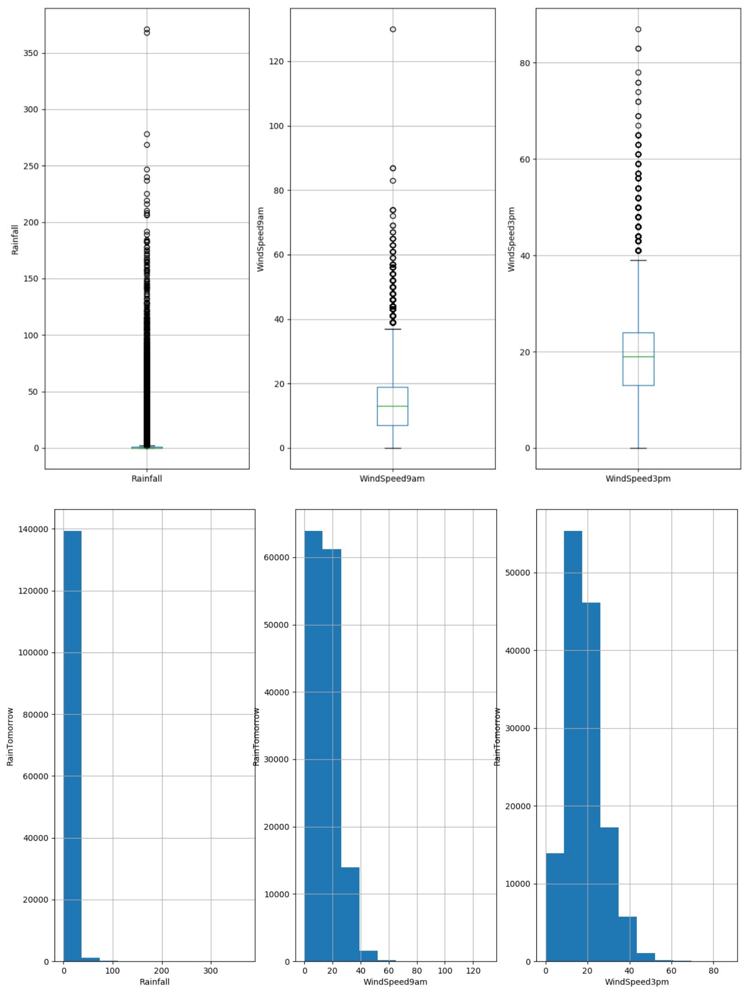
[7] Simon S Cross, Robert F Harrison, and R Lee Kennedy. Introduction to neural networks. *The Lancet*, 346(8982):1075–1079, 1995.

# Appendix

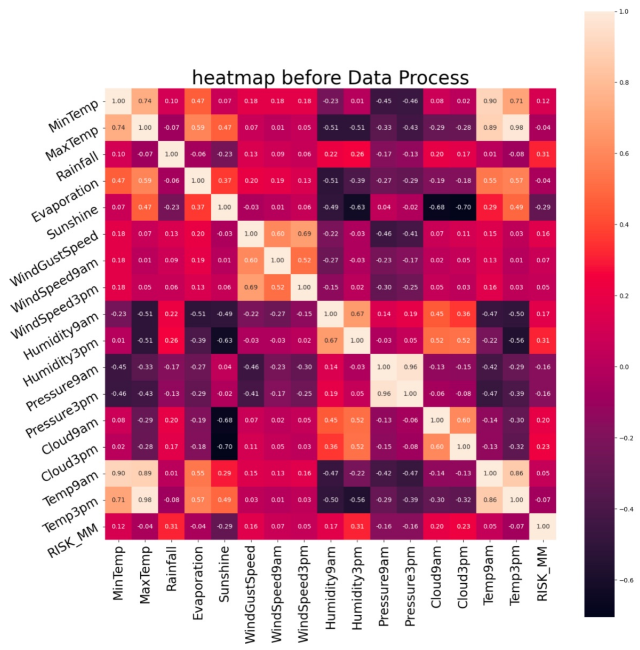
## Continuous Values

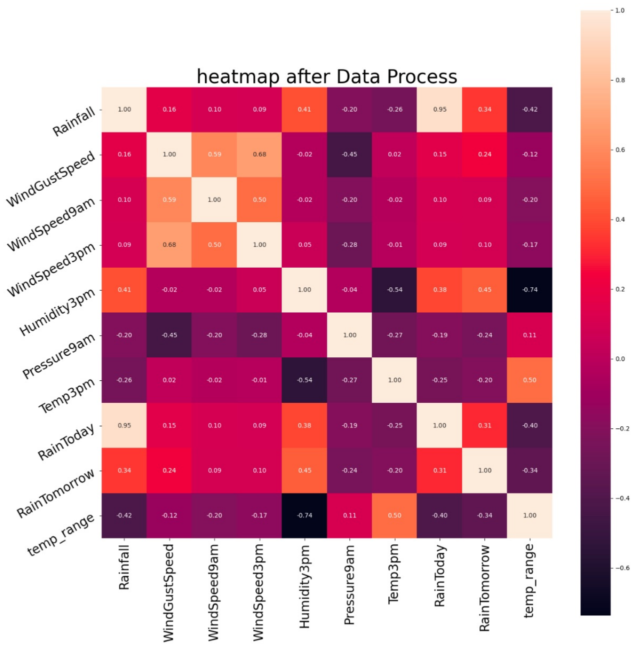


## The distribution of *'Rainfall'*, *'WindSpeed9am'* and *'WindSpeed3pm'*

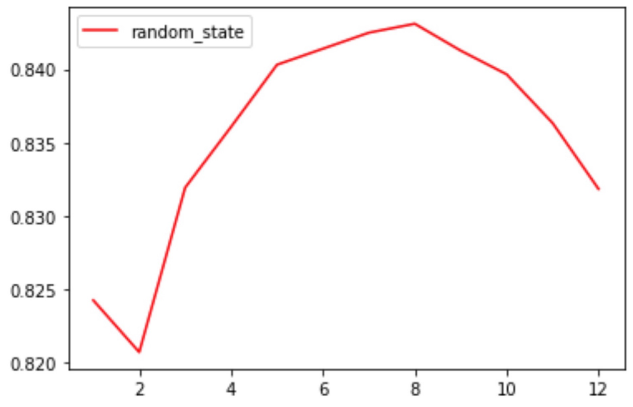


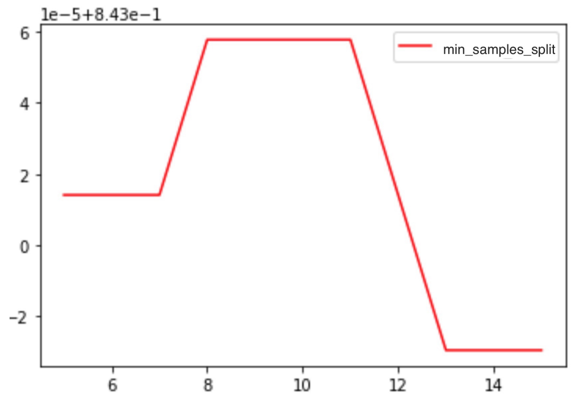
## Heatmaps before vs. after Feature Engineering



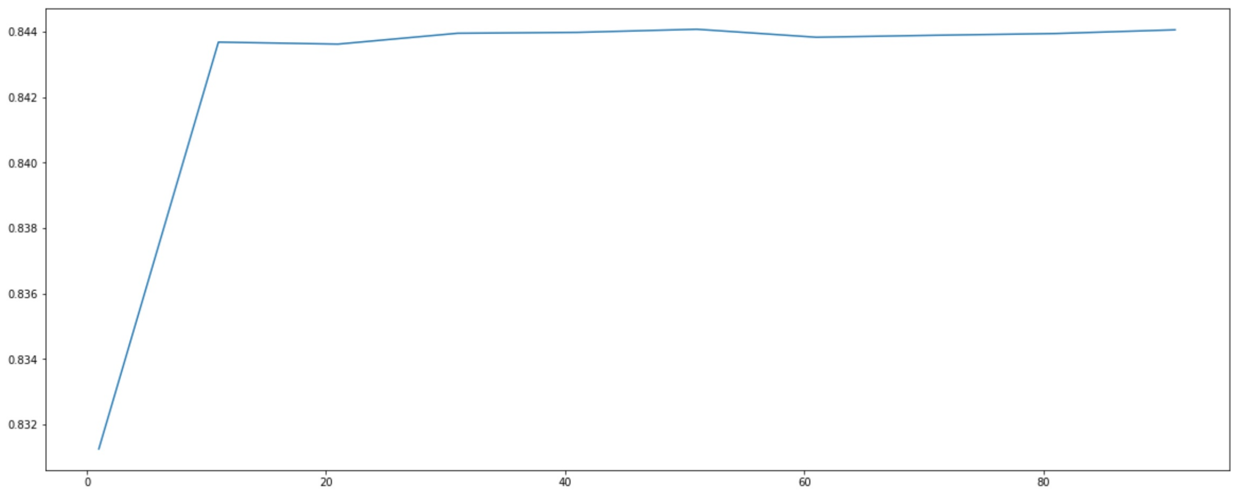


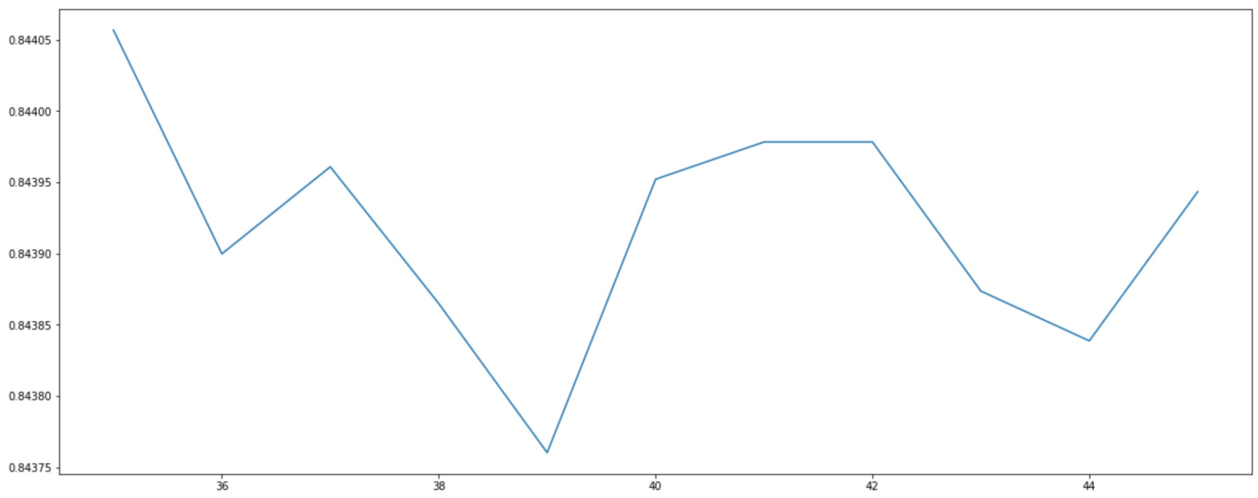
## Parameters tuning in Decision Tree model



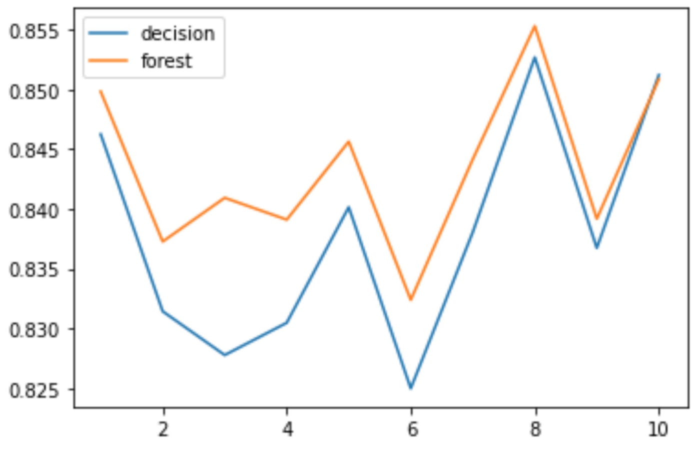


## Parameters tuning in Random Forest model (*"n_estimator"*)

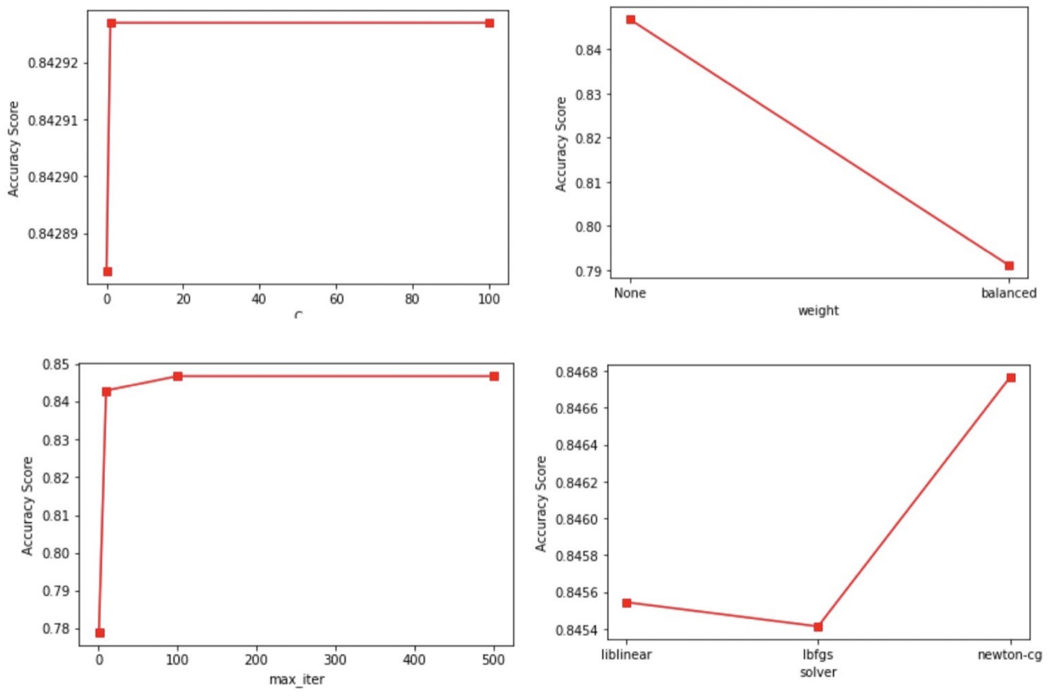




**Decision Tree performance vs. Random Forest Performance**



**Parameters tuning in Logistic Regression model (*"n_estimator"*)**



**The Confusion Matrix**

| Actual Class | Predicted Class | |
|---|---|---|
| | Yes | No |
| Yes | True Positive (TP) | False Negative (FN) |
| No | False Positive (FP) | True Negative (TN) |

**Score function formulations**

$$Classification\ error = \frac{FP + FN}{TP + TN + FP + FN}$$

$$Precision\ rate = \frac{TP}{TP + FP}$$

$$Recall\ rate = \frac{TP}{TP + FN}$$

$$F1\ score = \frac{2\ *\ precision\ rate\ *\ recall\ rate}{precision\ rate\ +\ recall\ rate}$$