**Development of Classification Techniques**

Author: Jose Manuel Alvarez Gonzalez

Email Id: jmanuel.ag89@gmail.com

Date: 22/5/2022

Contents

[Introduction 3](#_Toc135649940)

[Data Preparation 3](#_Toc135649941)

[Data Cleaning 3](#_Toc135649942)

[Data Splitting 4](#_Toc135649943)

[Model Preparation 4](#_Toc135649944)

[Feature Selection 4](#_Toc135649945)

[KNN 5](#_Toc135649946)

[Regression 7](#_Toc135649947)

[Decision Tree 8](#_Toc135649948)

[SVC 10](#_Toc135649949)

[Performance Measures 10](#_Toc135649950)

[Demo Data Test 11](#_Toc135649951)

[Mode Interpretation 12](#_Toc135649952)

[Logistic Regression 12](#_Toc135649953)

[Decision Tree 13](#_Toc135649954)

[Model comparison 14](#_Toc135649955)

# Introduction

For this project, we are going to continue the work with a dataset that contains daily weather observations for various Australian weather stations across 2019 and 2020. The data set has 25 features and 1 target variable which tells if it rained that day or not.

First, we will recap the process used to prepare the data set. Continue with the use of four different predictive models, detailing the stages used to prepare each model. Finalizing by comparing the models performance in terms of precision, recall, F1-score and balanced accuracy

# Data Preparation

## Data Cleaning

This stage was carried out in the first part of the project, the following is a description of the stages of the process:

The first stage involved descriptive statistical properties for each feature, revealing differing value ranges, units, and distribution types across features. Some data had null values which required addressing. Features "Cld\_9am" and "Cld\_3pm" with more than half of the values missing and "AVG\_Rain\_mm" with only one unique value were removed. Furthermore, two weather stations with a high percentage of null values across multiple features were eliminated from the dataset.

Missing data was handled by replacing null values with the median value for interval features and the mode for non-interval features, depending on the state the weather station belonged to. This decision was informed by an observed variation in feature medians across states.

The next step was feature transformation. Since most features weren't normally distributed, a logarithmic transformation was attempted but it didn't significantly reduce outliers or skewness. Therefore, Min-Max Scaler was applied to normalize interval features.

Categorical data was then transformed to numerical. The LabelEncoder function was applied to nominal features, and the OrdinalEncoder function was applied to ordinal features after establishing the relevant order of values.

Temperature-based interval features were binned into categories with a width of 5c, providing a better fit for the data spread. This features won’t be considered for the models.

The project progressed into feature selection, using four different strategies: Correlation Analysis, SelectKbest by chi2, SelectKbest by mutual info classifier, and SelectFromModel. The common important features were "Rain\_mm", "RH\_9am", and "RH\_3pm", while "Day" and "Year" were less significant.

Scatter plots and heatmaps were created for visual analysis. These graphics confirmed the importance of the previously mentioned features and allowed the correlation between variables to be visually evaluated.

In conclusion, through data exploration, cleaning, transformation, normalization, categorization, and feature selection, the dataset was prepared for future predictive modelling. The features "Rain\_mm", "RH\_9am", and "RH\_3pm" were identified as most relevant for predicting rain, thus supporting model performance.

## Data Splitting

For use of the data in the models, we first looked at the distribution of our target variable, concluding that 70% of the results show that didn’t rain. This indicates that we need to split all data keeping same distribution, at the same time we will use the 70% as a baseline for any predictive model.  
  
For the splits, we follow an initial 25% split for the Test data, with the remaining been further split to 25% Validation and the remaining to a Training data set, with the last data sets as the ones we will use to prepare the models. From the final training data, we randomly selected 20 samples for a demo data to be used at the end of the project.

# Model Preparation

## Feature Selection

In the data preparation stage, we used the function SelectKbest by mutual info classifier, creating a ranking of the most important features to use. Now I want to investigate how many of those features we need to use per model.

I created a learning curve plot for each model, to visualize how the accuracy will increase or decrease as we add more features on the model. The features were added based on the previous SelectKbest ranking, from the most to less important.   
  
Figure 1, shows the results we have, we can see that after 6 features we don’t have better results in any model. This is just an initial test of the models, we are aiming to get better results once we fine tune the parameters. From the previous result, we will only work with the top 6 features for each model.

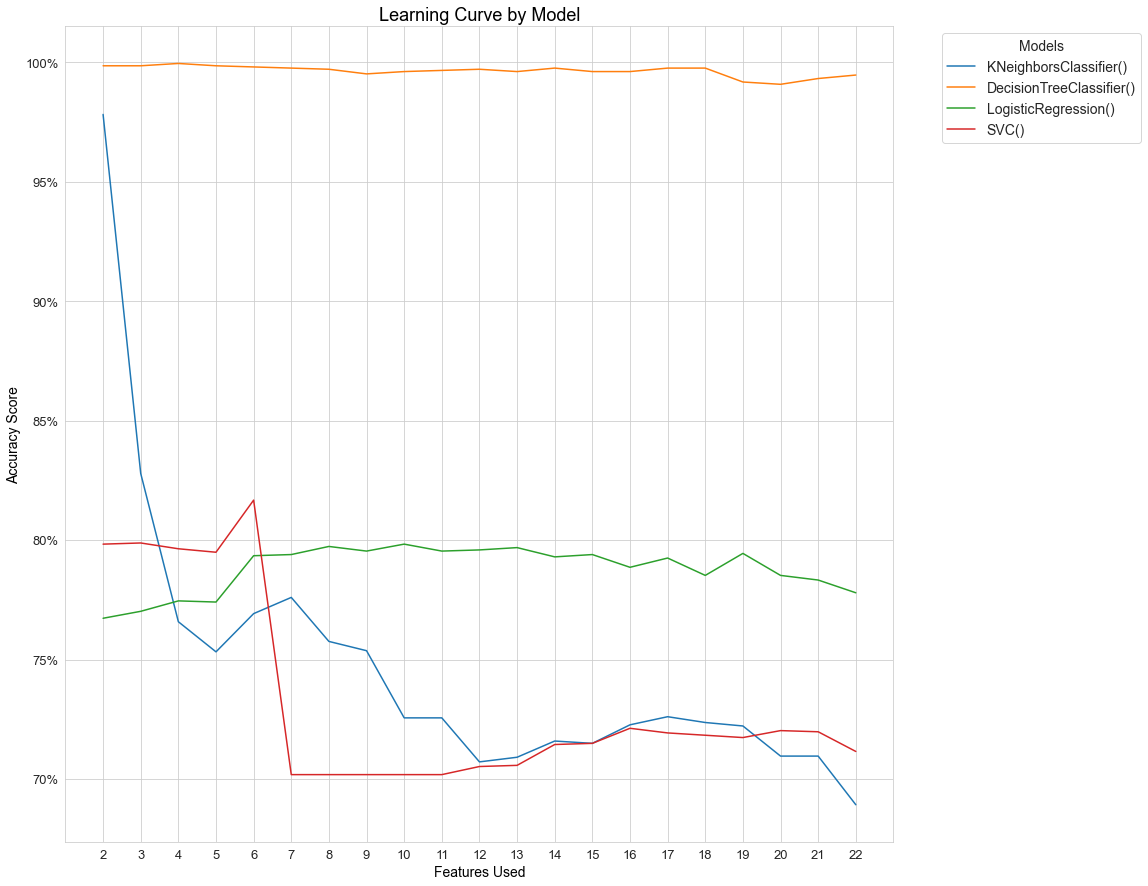


Figure , learning curve for each of the 4 models and the features used.

## KNN

For this model, we have two important parameters to decide: first we have the number of neighbours the model is going to choose to make a decision (we need an odd number), secondly, we have the measurement for distance used (Euclidean or Manhattan).   
  
To identify the best parameters, I tested the model with each distance and a range of 3 to 60 neighbours. For testing, we used the validation data set, the results showed that using the 6 features, 31 neighbours and Manhattan measurement we obtain 81% accuracy. This is better than the 70% baseline but not much improvement, looking back at figure 1, we can see that with only two features the model is doing a really good job.

I decided to test the model with 2 features and we obtained a 97.81% with the default settings, I also tested if we can improve the model by adding more k-values and we have the following figure:

A picture containing text, plot, diagram, line

Description automatically generated

Figure , KNN learning curve of k-values with only top 2 features

To understand why this works, I created figure 3. Here we can see a clear visual distinction on the right side of the table for samples when it rained.

A picture containing text, screenshot, diagram, line

Description automatically generated

Figure , dot plot of RH\_9am vs Rain\_mm

Now we have the model, with the final parameters:

* Metric: Euclidean
* N\_neighbors = 3

Testing the final model on the test data set, gives us an accuracy rate of 97.67% and we have figure 4 with the results of each sample, we will look at the results later and compare them between all four models.

A picture containing text, screenshot, number, diagram

Description automatically generated

Figure , Confusion matrix of KNN on Test data

## Regression

For the regression model, we followed the same approach. In this case we looked at the following parameters:

* C: Is an hyperparameter that regularizes the data and gives a penalty to the loss function.
* Penalty: The type of regularization applied
* Solver: Algorithm used to optimize the loss function.

In simpler terms, these parameters will tell the model how to mathematically handle values that are to far away from what the model predicts and find the best fit.

To test the parameters, we used a function call GridsearchCV, what this function does, it creates a model with each single combination of the parameters we choose. Then for testing which one works better, it creates a cross validation with the Training data, I chose 5 cross validation tests for each model. This technique can take a long time if we use a high number of parameters, but it will be the most reliable.

The resulting parameters:

* C: 10
* Penalty: L1
* Solver: liblinear

Now we test on the Test data, and we obtain a 90.3% accuracy, with the following results:

A picture containing text, screenshot, number

Description automatically generated

Figure , Confusion matrix of Regression on Test data

## Decision Tree

For the decision tree the most important part to consider, it becomes easy to overfit the data into the model. To prevent overfitting, I will start with a small tree and slowly increase the size of it and compare results. The parameters that will help building the tree are as follow:

* Criterion: The measurement of information gain by choosing a feature to split data
* Max\_depth: Maximum levels of the tree.
* Max\_leaf\_nodes: Maximum number of leaf nodes in the tree.
* Min\_samples\_leaf: Minimum number of samples on each leaf node.

Using a GridsearchCV with conservative parameters, we obtained the following best fit: {'criterion': 'gini', 'max\_depth': 2, 'max\_leaf\_nodes': 2, 'min\_samples\_leaf': 50}. Giving the following tree:

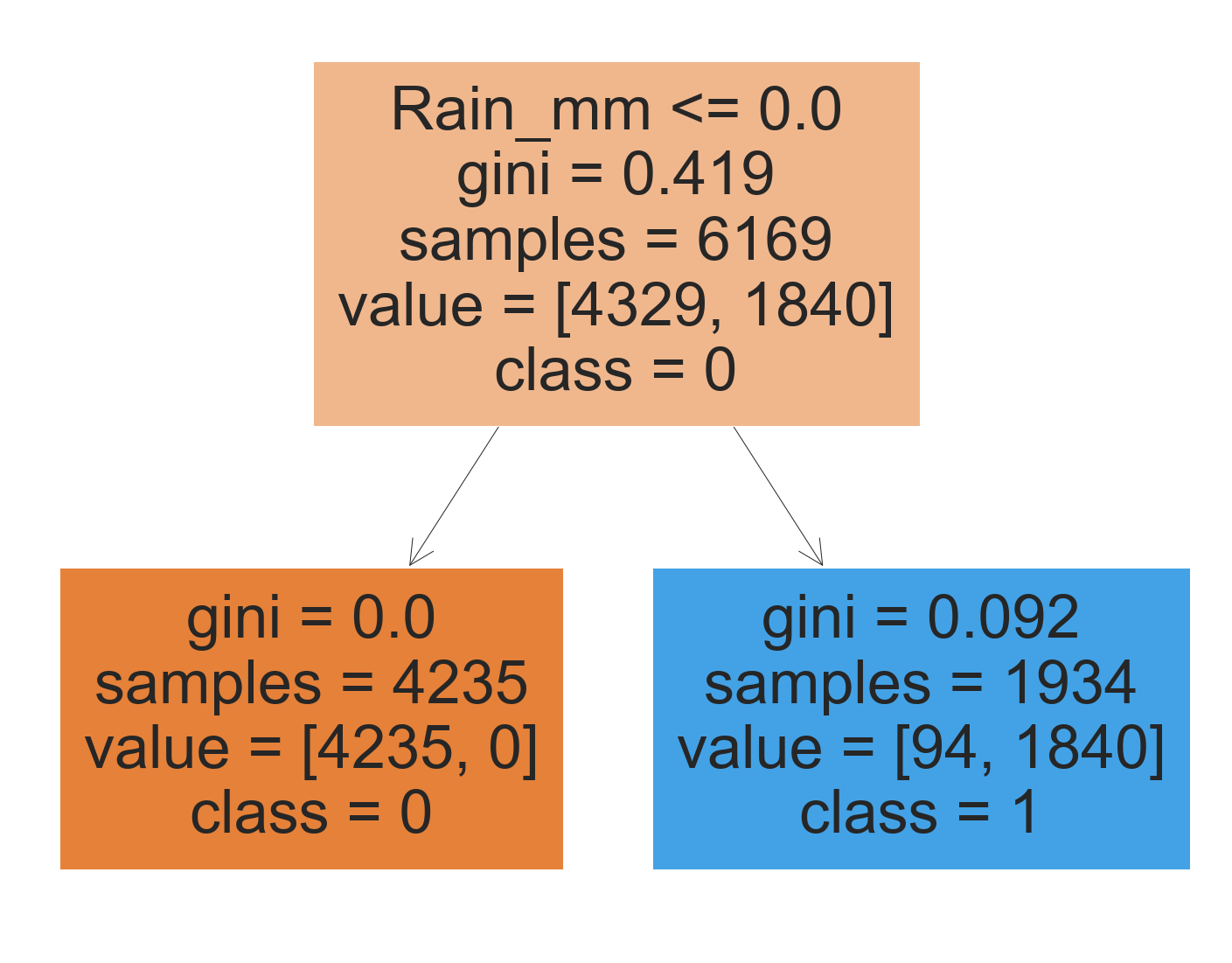


Figure , simple decision tree

We can see that only the feature of “Rain\_mm” is used to split the data, with a 100% accuracy at predicting days that didn’t rain. I will try to get a better result by increasing the depth of the tree, following a GridsearchCV I obtained: {'criterion': 'entropy', 'max\_depth': 8, 'max\_leaf\_nodes': 8}

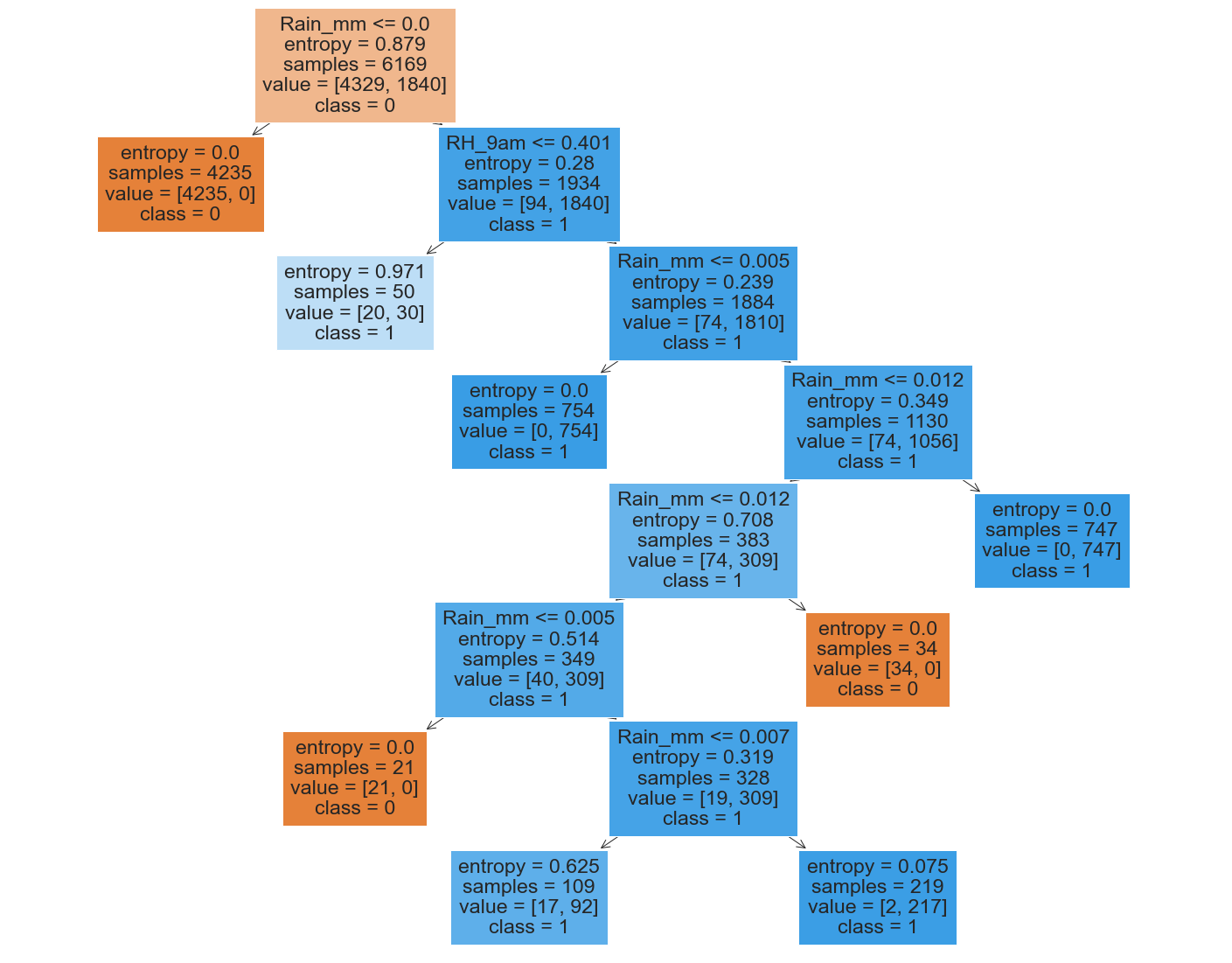


Figure , complex decision tree

Here we can identify that the tree is starting to overfit the data across the variable “Rain\_mm” and the feature “RH\_9am” is only used once. To have a better understanding of the model, I also included a tree with the default settings, this resulted in a really complex tree with more than 10 leaf nodes a depth of 14.

Now I want to compare the last three models with the validation set:

* Simle\_tree: 98.64%
* Complex\_tree: 99.36%
* Default\_tree: 99.9%

For this case, I will choose the simplest tree as this doesn’t overfit the data and still has the best score so far with only one feature. We now use this model on the Test data to obtain 98.58%, with the following results:

A picture containing text, screenshot, number, diagram

Description automatically generated

Figure , Confusion matrix of Decision Tree on Test data

## SVC

This model uses a plane and tries to visually separate the classes across them. To be able to handle the 6 dimensions (top 6 features) we need to look for the parameters that better help at separating the data we have. The parameters we used:

* Kernel: The mathematical function used to transform and separate the data in the plane
* Gamma: Used for the 'rbf', 'poly', and 'sigmoid' kernels. Used to control the decision boundary.
* Degree: This is for the 'poly' kernel, used for the polynomial degree of the plane equation.
* C: Penalty parameter of the error term, need to be careful to don’t add a really high value as it can overfit the data.

As we can see not all parameters work together, I also excluded C after testing as it makes the model incredible slow. I used the function GridsearchCV and obtained the following best fit: {kernel: ‘poly’, degree: 4}.  
  
I decided to also include the ‘rbf’ and ‘linear’ kernel to compare with the validation data, we obtained the following:

* Poly: 87.15%
* RBF: 81%
* Linear: 74.16%

Clearly the polynomial with degree 4 is the best results, we now use this model on the Test data to obtain 88.66%, with the following results:

A picture containing text, screenshot, number

Description automatically generated

Figure , Confusion matrix of SVC on Test data

## Performance Measures

Now that we have tested each final model on the Test data, we can compare each model performance. As we are interested if the model predicted if it Rained, we will be comparing the performance across that result. We are going to use the measurements of precision, recall, F1-score and balanced accuracy. The results are summarized on the following table:



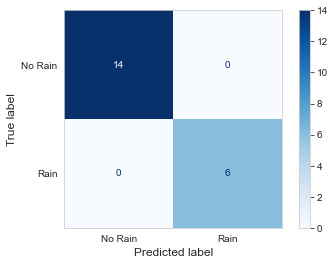
Figure , performance measure of four models

From the table we can see that SVC has the lowest scores on performance, F1 and BCR (Balanced Classification Rate). All of the models scored at least a 95% on their precision, is on the recall where we can clearly see the difference between them. For the best performing we have KNN and the Decision Tree, with the tree performing 2% better on the BCR and F-1 Score.

## Demo Data Test

Now we are going to use each model on the Demo Data Test and display the results on the confusion Matrix, this gives us the full idea of true positives, true negatives, false positives and false negatives.

KNN – 100% Accuracy: Regression – 90% Accuracy:

 A picture containing text, screenshot, rectangle, diagram

Description automatically generated

Decision Tree – 100% Accuracy: SVC – 90% Accuracy:

A screenshot of a graph

Description automatically generated with low confidence A picture containing text, screenshot, rectangle, diagram

Description automatically generated

From the last results, we can see how the SVC and Regression failed with false negative at misplacing two samples as “Not Rain” when it should have been “Rain”. Decision Tree and KNN both scored a 100%.

# Mode Interpretation

## Logistic Regression

The logistic regression model is one of the predictive models used in this project. Used to predict whether it would rain based on various weather observations from Australian weather stations.

Logistic regression, in this project, works by taking a set of features from the dataset ("Rain\_mm", "RH\_9am", "RH\_3pm", 'Temp\_3pm', 'Max\_Temp', 'Temp\_9am') and learns how these features can predict if it rained on a particular day. The features used were determined based on a feature selection process that ranked their importance.

In terms of parameters, the regression model used in this project considered a few key parameters:

C: This is a hyperparameter that helps the model control how much it adjusts to individual observations. This helps prevent overfitting by allowing the model to make mistakes and thereby learn the general pattern instead of memorizing the training data.

Penalty: This refers to the type of regularization applied to the model, another measure to prevent overfitting.

Solver: This is the specific algorithm used by the model to optimize the loss function and find the best fit for the data.

These parameters were tuned using a method called GridSearchCV, which tests different combinations of parameters to find the ones that give the best results.

After the model was trained and fine-tuned, it was tested using a separate dataset. The performance of the model scored an accuracy of 90.3%.

In terms of performance measures, it was found that the logistic regression model had high precision, meaning it was very good at accurately predicting when it was going to rain. However, the recall, which measures the model's ability to find all the data points of interest, was not as high compared to other models. This difference was particularly noticeable in the Balanced Classification Rate and F-1 Score, which both take precision and recall into account.

We also tested the regression model on a dataset of 20 samples, where it achieved an accuracy rate of 90%.

Overall, the logistic regression model was able to predict whether it would rain based on the provided weather observations, but its performance was not the best compared to the other models used in the project. Despite this, it still performed reasonably well and could be useful in a practical setting.

## Decision Tree

The Decision Tree can be told as a series of yes and no questions based on the amount of information gained from each answer. In this project we learnt that for the data we have, the simplest tree is more than good enough to proceed.

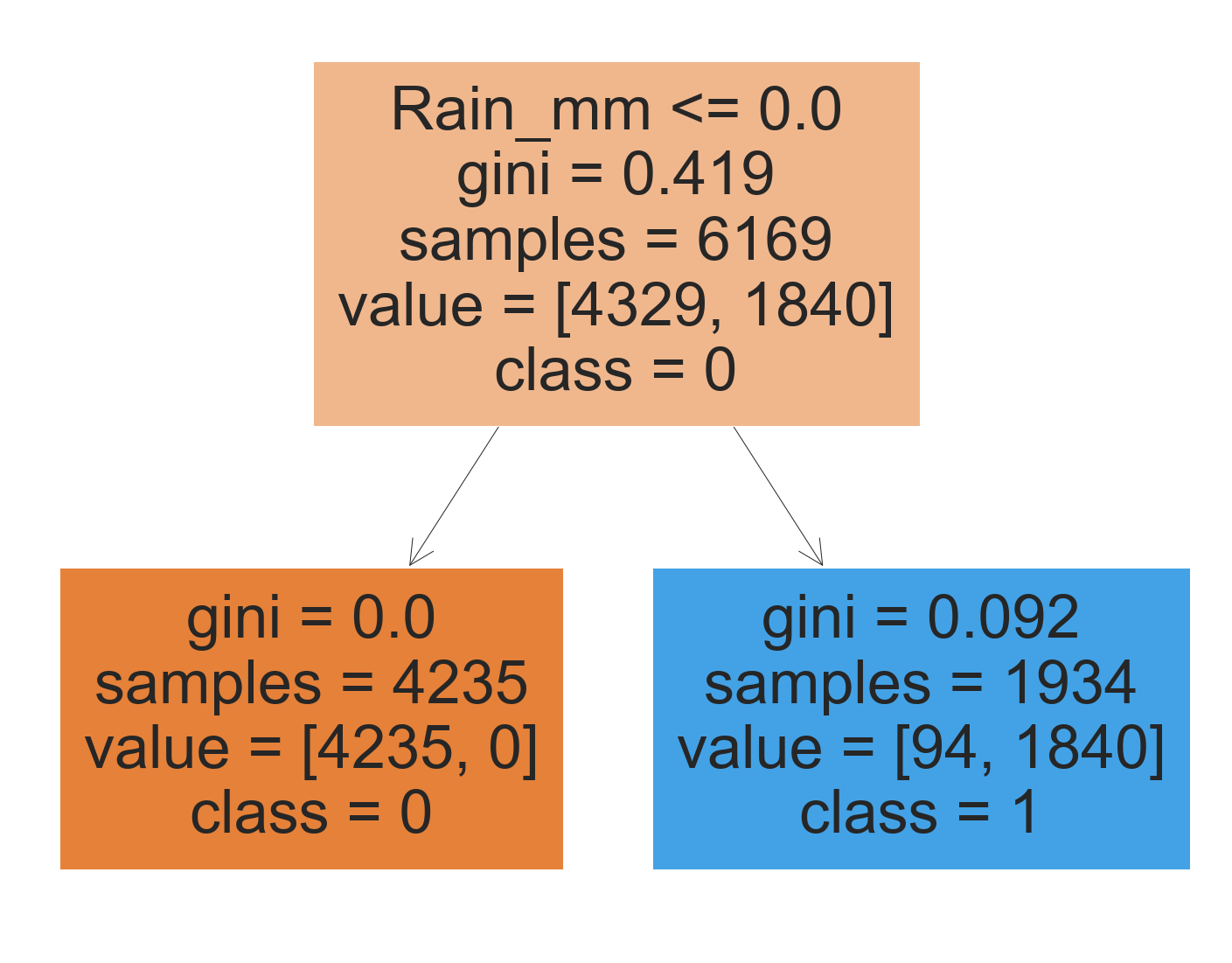


Figure , simple decision tree

As we can see on figure 6, all we need to know If the feature “Rain\_mm” scored more than 0 on the day, with a 98% confidence we can accurately predict if it rained or not that day.

# Model comparison

In this project, we worked with daily weather data collected from various Australian weather stations to predict if it would rain on a particular day. The analysis involved data cleaning, transformation, normalization, categorization, and feature selection processes. We worked with four predictive models: KNN, Logistic Regression, Decision Tree, SVC.

Based on precision, recall, F1-score, and balanced accuracy metrics, we observed that the Decision Tree and KNN models were the top performers. These models delivered an accuracy of 98.58% and 97.67%, respectively, doing better than the baseline of 70%. The simplicity of the Decision Tree model, gives it a major advantage, as it only relies on the feature "Rain\_mm" to predict whether it would rain or not.

The worst-performing model was the SVC, which only managed an accuracy of 88.66% on the test data. Still better than the 70% baseline.

The results were further confirmed when the models were tested on a smaller demo data set. The KNN and Decision Tree models correctly predicted all samples, the Logistic Regression and SVC models failed to correctly predict a couple of samples, resulting in 90% accuracy.

In conclusion, the Decision Tree model demonstrated the best performance in predicting whether it would rain or not on a given day. The combination of its simplicity, interpretability, and high predictive accuracy makes it an excellent choice for this particular application. The SVC model was the least effective, probably caused by the noise and overlap of result, making it difficult to find a non-linear decision boundary.