May 19th, 2020

ANTALYA BİLİM
UNIVERSITY

Department of Computer Engineering

CS 447

Introduction to Data Science

Instructor

Dr. Sadi Evren SEKER

Spring, 2020

Project Report

Rain Prediction in Australia

Submitted by

190201038 – Ibrahim Berber

Contents

Abstract

In this report, a detail explanation and analysis of two approaches we have covered within the framework of CS447 – Introduction to Data Science course are presented. These two approaches are namely *Random Forrest Classifier* and *Logistic Regression*.

The dataset I have chosen for the project is *Rain Prediction in Australia*, which is publicly available on Kaggle Data Platform and be accessed via https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.

The goal of the project is attempting to obtain a model which has high accuracy, using the approaches of *Random Forrest Classifier* and *Logistic Regression*.

Performance evaluation of each machine learning model approaches included in the report, along with their comparison of each other. To have a fair comparison, both approaches are implemented on the same platform (KNIME Analytics Platform) and have been executed on the same computer.

The analysis of the data and how it is processed is explained in Section 1. To have a better understanding, some visualization techniques have been used in Section 2. Visualizations are implemented with KNIME and Anaconda3 Python programming language distribution (over Jupyter Notebook). Section 3 covers the implementation of two approaches. Comparison of two approaches in four categories is presented in Section 4. Lastly, Section 5 recaps the questions (which stated in Course webpage: http://sadievrenseker.com/wp/?page_id=2252) we needed to answer in this project. KNIME workflow SVG image is attach at the end of the report.

1. Introduction

One of the places where data plays a crucial role is weather predictions. Having appropriate data makes researchers to understand the overall trend, or the factors that affect weather the weather will be raining or not. This provides a chance to make a proper prediction on the status of the weather. Thanks to these predictions, we can shape our daily plans accordingly. More specifically, within the framework of this project, the data is concentrates on the classification task of whether a day is raining or not.

In this project, I have implemented *Random Forrest Classifier* and *Logistic Regression* approaches which we have coved in the lectures, with the ultimate goal of obtaining prediction values on the probability of raining.

Dataset and Data Availability

The dataset used in this project, contains about 10 years of daily weather observations from numerous Australian weather stations. The target variable is **`RainTomorrow`**, which represents the question of "did it rain the next day?". The answer is a binary respond of either **`Yes`** or **`No`**.

The dataset is published in Kaggle Data Platform, that is publicly available and can be accessed via following link: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.

The first five entries of dataset are given in Figure 1.1. Our dataset consists of 5 rows and 24 columns. The dataset contains continues data (e.g. **`Date`**), quantitative data (e.g. **`MinTemp`**), and categorical data (e.g. **`WindGustDir`**). Some detail of the features of data are as presented in Table 1.1.

```
In [1]: import pandas as pd

        weather_dataframe = pd.read_csv('weatherAUS.csv')
```

```
In [2]: weather_dataframe.head()
```

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...
0	12/1/2008	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...
1	12/2/2008	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...
2	12/3/2008	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...
3	12/4/2008	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...
4	12/5/2008	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...

5 rows × 24 columns

Figure 1.1. *An overview of data*

Columns	Description
Date	The date of observation
Location	The common name of the location of the weather station
MinTemp	The minimum temperature in degrees Celsius
Rainfall	The amount of rainfall recorded for the day in mm
Sunshine	The number of hours of bright sunshine in the day.
RainTomorrow	The target variable. Did it rain tomorrow?

Table 1.1. *Some columns in dataset and their descriptions*

Import and Review of Data in KNIME

First, dataset has been downloaded and saved in the same directory with KNIME project for convenience. If we look at the data, we see that its extension is **`.csv`**, which stands for *comma separated values*. In KNIME, *CSV Reader Node* has been added to workspace and in settings, we navigated where our dataset is located at. Notice that, we untick *Has Row Header*. Otherwise we get the error in Figure 1.3.

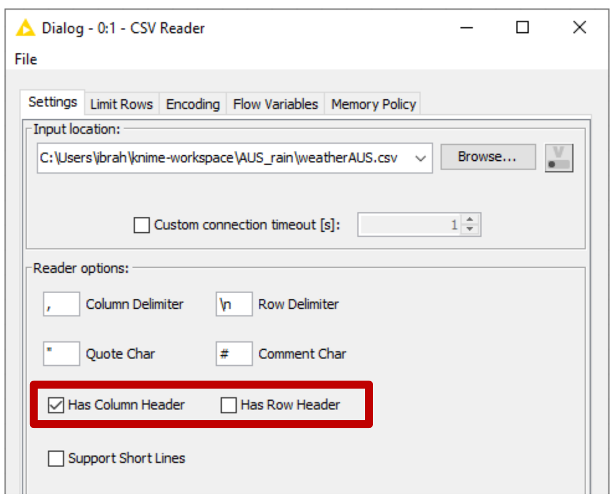


Figure 1.2. Import of data, (*Has Row Header* is unchecked)

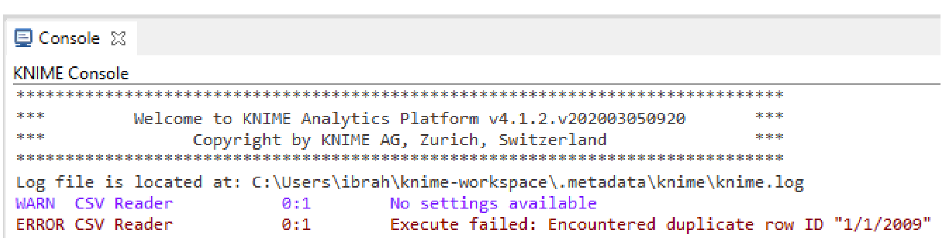
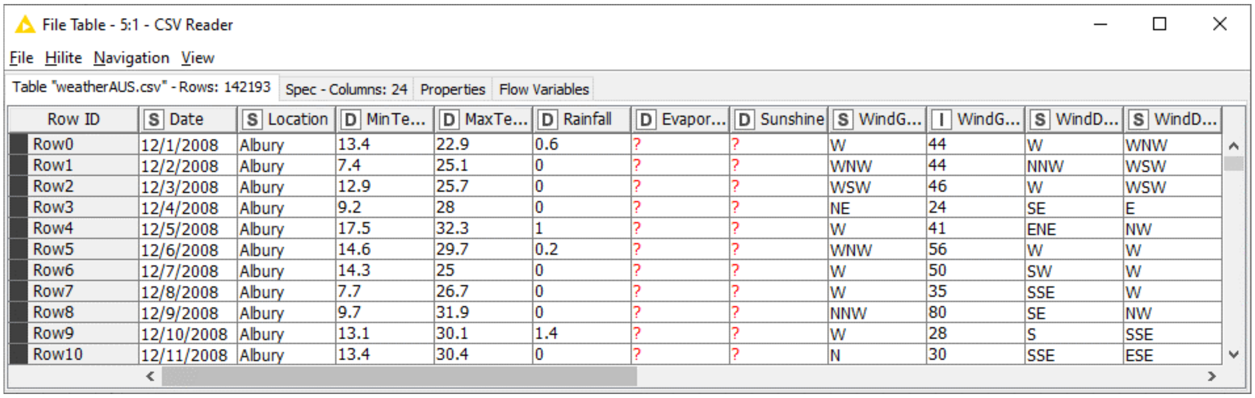


Figure 1.3. Error in importing data

In order to confirm that our data is successfully imported, we click on *File Table* in *CSV Reader Node*. As shown in Figure 1.4, our values are presented.


File Table - 5:1 - CSV Reader

File Hilite Navigation View

Table "weatherAUS.csv" - Rows: 142193 | Spec - Columns: 24 | Properties | Flow Variables

Row ID	S Date	S Location	D MinTe...	D MaxTe...	D Rainfall	D Evapor...	D Sunshine	S WindG...	I WindG...	S WindD...	S WindD...
Row0	12/1/2008	Albury	13.4	22.9	0.6	?	?	W	44	W	WNW
Row1	12/2/2008	Albury	7.4	25.1	0	?	?	WNW	44	NNW	WSW
Row2	12/3/2008	Albury	12.9	25.7	0	?	?	WSW	46	W	WSW
Row3	12/4/2008	Albury	9.2	28	0	?	?	NE	24	SE	E
Row4	12/5/2008	Albury	17.5	32.3	1	?	?	W	41	ENE	NW
Row5	12/6/2008	Albury	14.6	29.7	0.2	?	?	WNW	56	W	W
Row6	12/7/2008	Albury	14.3	25	0	?	?	W	50	SW	W
Row7	12/8/2008	Albury	7.7	26.7	0	?	?	W	35	SSE	W
Row8	12/9/2008	Albury	9.7	31.9	0	?	?	NNW	80	SE	NW
Row9	12/10/2008	Albury	13.1	30.1	1.4	?	?	W	28	S	SSE
Row10	12/11/2008	Albury	13.4	30.4	0	?	?	N	30	SSE	ESE

Figure 1.4. An overview of the AUS weather data in KNIME

A brief summary and related useful information of dataset can be obtained by use of Extract Table Dimension and Extract Table Spec nodes. Using these nodes, we see that there are 24 columns and properties of each column can be seen in Figure 1.5. The dimension of dataset is shown Figure 1.6.

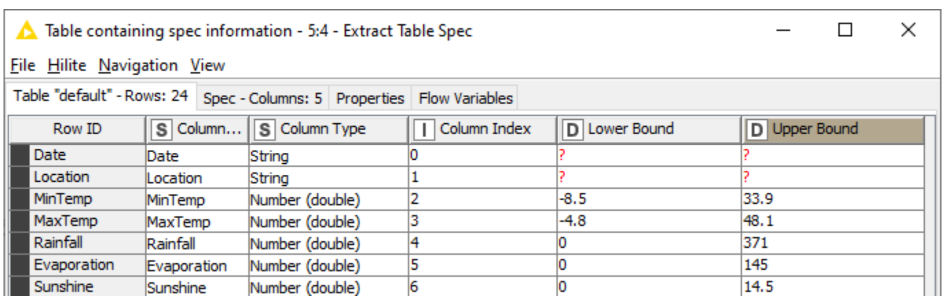


Table containing spec information - 5:4 - Extract Table Spec

File Hilite Navigation View

Table "default" - Rows: 24 | Spec - Columns: 5 | Properties | Flow Variables

Row ID	S Column...	S Column Type	I Column Index	D Lower Bound	D Upper Bound
Date	Date	String	0	?	?
Location	Location	String	1	?	?
MinTemp	MinTemp	Number (double)	2	-8.5	33.9
MaxTemp	MaxTemp	Number (double)	3	-4.8	48.1
Rainfall	Rainfall	Number (double)	4	0	371
Evaporation	Evaporation	Number (double)	5	0	145
Sunshine	Sunshine	Number (double)	6	0	14.5

Figure 1.5. Column info is obtained

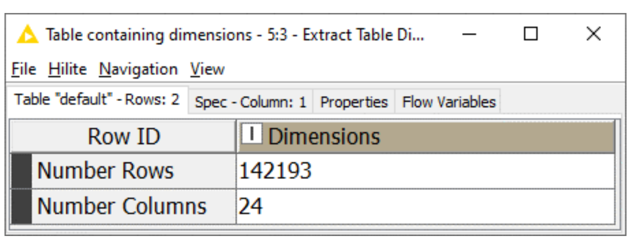


Table containing dimensions - 5:3 - Extract Table Di...

File Hilite Navigation View

Table "default" - Rows: 2 | Spec - Column: 1 | Properties | Flow Variables

Row ID	I Dimensions
Number Rows	142193
Number Columns	24

Figure 1.6. Dataset size is obtained

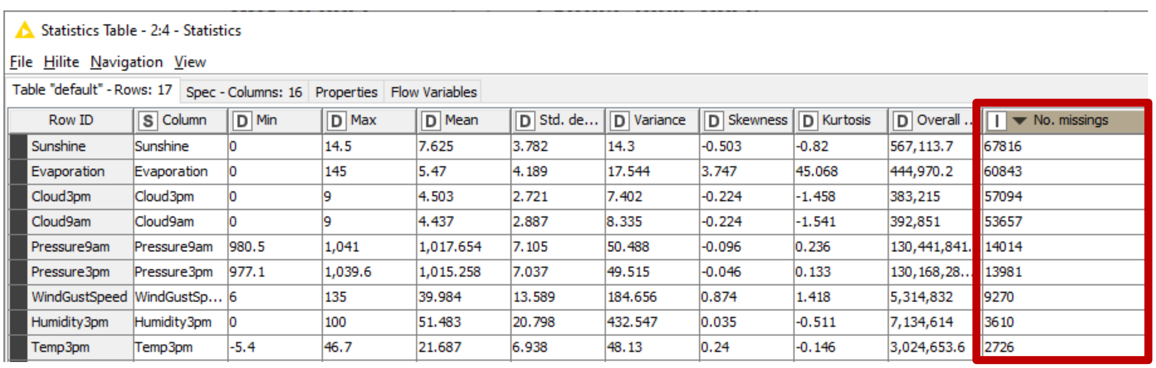
Data Preprocessing in KNIME

As seen in Figure 1.4. above, we realize our dataset contains *missing values*. In the implementation of models, missing values are problematic. Hence, they are needed to be extracted from data. For each column, we can count the number of missing values. For that purpose, we use *Statistics > Statistics table*.

In the Figure 1.7. below, there is *No. missings* column, we sort descending. That will sort the missing values of each column in decreasing manner. We observe that **`Sunshine`**, **`Evaporation`**, **`Cloud3pm`** and **`Cloud9am`** have larger number of missing values (more than one third of total data is missing) compared to missing values of other columns. Therefore, we can ignore these columns.

Additionally, in the Kaggle where the dataset is available at, it is stated that **`RISK_MM`** should be excluded since it contains information directly about the target variable.

Lastly, my aim is to make a prediction in whole country. Therefore, the location column will be dropped as well, because it represents cities.


Statistics Table - 2:4 - Statistics
File Hilite Navigation View
Table "default" - Rows: 17 | Spec - Columns: 16 | Properties | Flow Variables

Row ID	S Column	D Min	D Max	D Mean	D Std. de...	D Variance	D Skewness	D Kurtosis	D Overall ..	I No. missings
Sunshine	Sunshine	0	14.5	7.625	3.782	14.3	-0.503	-0.82	567,113.7	67816
Evaporation	Evaporation	0	145	5.47	4.189	17.544	3.747	45.068	444,970.2	60843
Cloud3pm	Cloud3pm	0	9	4.503	2.721	7.402	-0.224	-1.458	383,215	57094
Cloud9am	Cloud9am	0	9	4.437	2.887	8.335	-0.224	-1.541	392,851	53657
Pressure9am	Pressure9am	980.5	1,041	1,017.654	7.105	50.488	-0.096	0.236	130,441,841.	14014
Pressure3pm	Pressure3pm	977.1	1,039.6	1,015.258	7.037	49.515	-0.046	0.133	130,168,28...	13981
WindGustSpeed	WindGustSp...	6	135	39.984	13.589	184.656	0.874	1.418	5,314,832	9270
Humidity3pm	Humidity3pm	0	100	51.483	20.798	432.547	0.035	-0.511	7,134,614	3610
Temp3pm	Temp3pm	-5.4	46.7	21.687	6.938	48.13	0.24	-0.146	3,024,653.6	2726

Figure 1.7. Number of missing values for each column

These dropping operation are actually ignore of the columns we have mentioned. To do so, we can filter out these columns using Column Filter node in KNIME. My first attempt in running the two algorithms was with extraction of above-mentioned variables, however, the column **`date`** was creating problems. Hence, later, I excluded **`date`** as well.

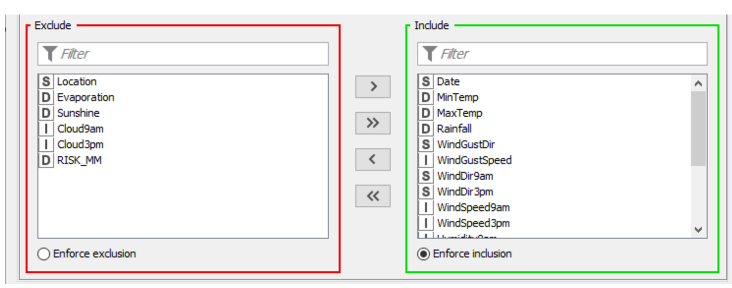
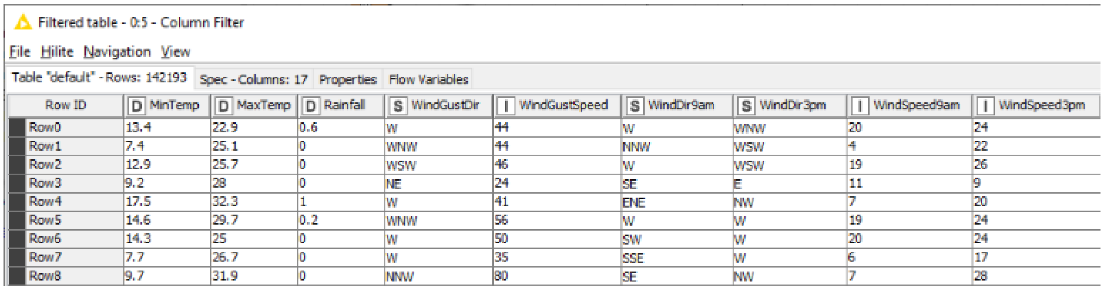


Figure 1.8. Filter of columns using Column Filter node

After ignoring the columns which either had missing values or ones we did not need, we can check resulting data in Column Filter node. Indeed, as shown partially in Figure 1.9, red question marks no longer exists.



Filtered table - 0:5 - Column Filter

File Hilite Navigation View

Table "default" - Rows: 142193 | Spec - Columns: 17 | Properties | Flow Variables

Row ID	D MinTemp	D MaxTemp	D Rainfall	S WindGustDir	I WindGustSpeed	S WindDir9am	S WindDir3pm	I WindSpeed9am	I WindSpeed3pm
Row0	13.4	22.9	0.6	W	44	W	WNW	20	24
Row1	7.4	25.1	0	WNW	44	NNW	WSW	4	22
Row2	12.9	25.7	0	WSW	46	W	WSW	19	26
Row3	9.2	28	0	NE	24	SE	E	11	9
Row4	17.5	32.3	1	W	41	ENE	NW	7	20
Row5	14.6	29.7	0.2	WNW	56	W	W	19	24
Row6	14.3	25	0	W	50	SW	W	20	24
Row7	7.7	26.7	0	W	35	SSE	W	6	17
Row8	9.7	31.9	0	NNW	80	SE	NW	7	28

Figure 1.9. Filtered data, missing values and unnecessary columns are removed

Unsurprisingly, the number of columns and rows have been reduced since we have excluded some of them. Updated dimensions and table specs can be obtained by connecting yet another Extract Table Spec and Extract Table Dimension, as shown in Figure 1.10. For *nan* value in rows, Missing Value Node is used.

Now that we have completed pre-processing, we can move on to the analysis of the data.

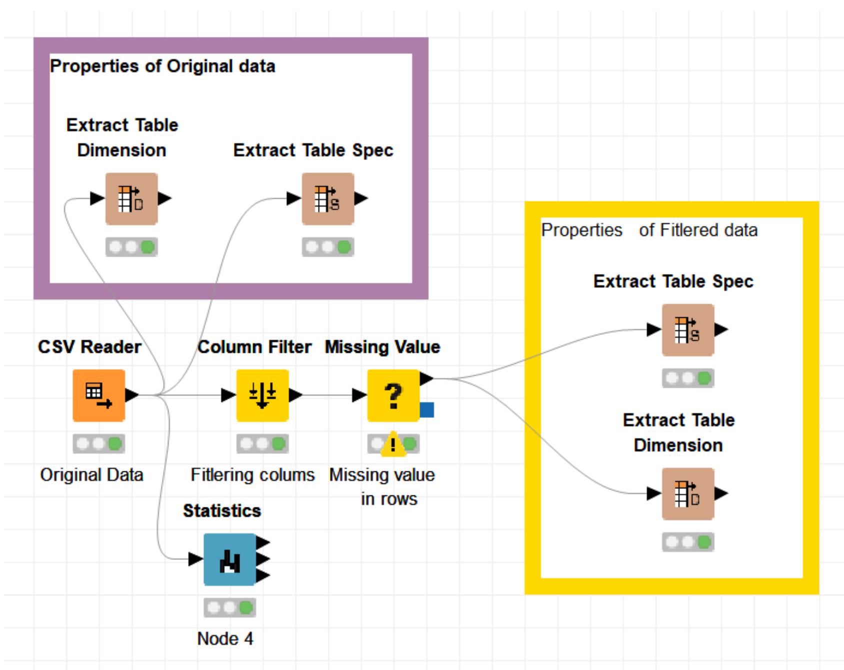
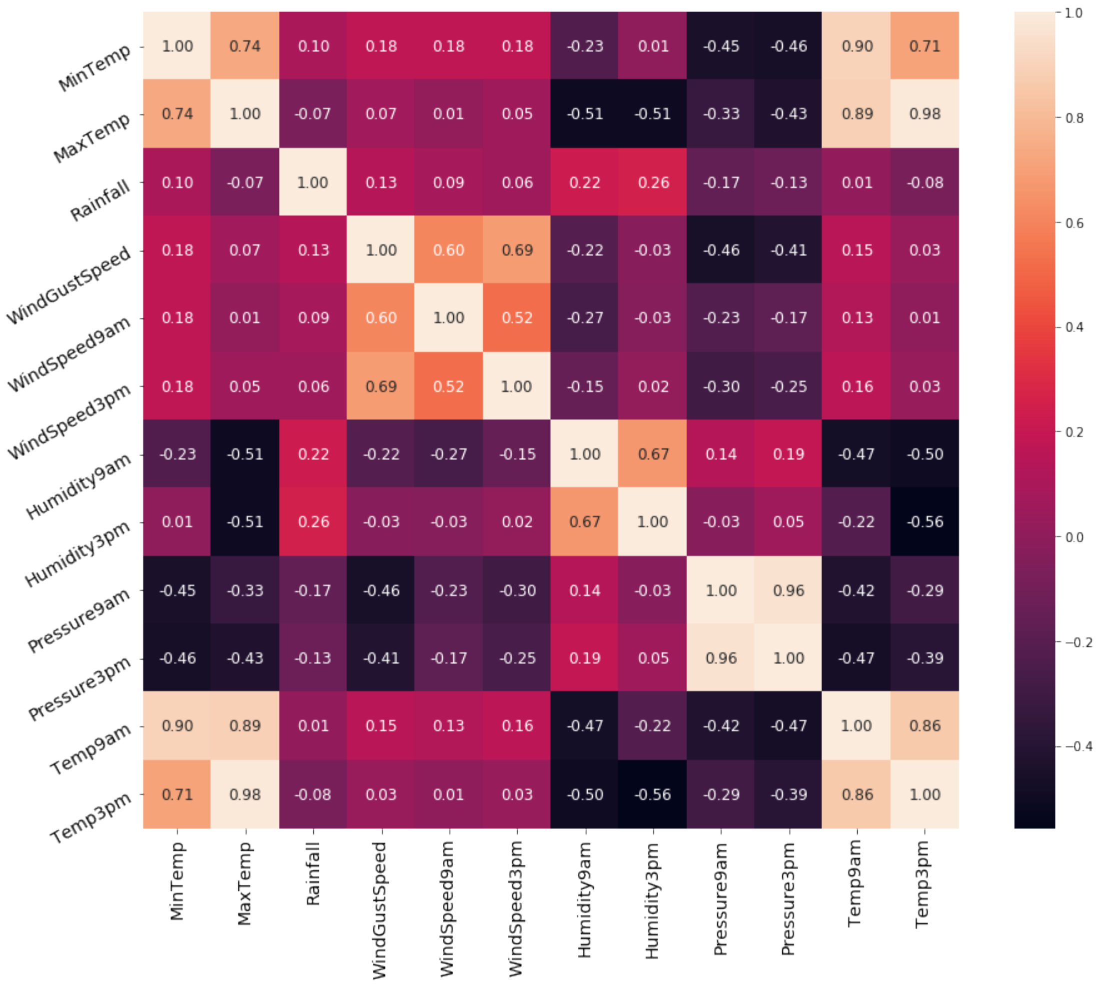


Figure 1.10. Nodes in KNIME

2. Analysis of Data

Before going into the implementation of two approaches, a good understanding of data in columns and their relation or correlation of each other is important. The patterns and relationships among filtered columns are best viewed by heat map, which is given in Figure 2.1.a below. From the graph, we can conclude that columns of temperatures have high correlation among themselves, e.g. **`MaxTemp`** and **`Temp9am`**. Same is true for general column types of wind and humanity. **`Rainfall`**, on the other hand, has no correlation except itself.

Figure 2.1.a Correlation Heatmap of Rain in Australia Dataset



Some visualization of data using KNIME environment is presented in Figures 2.1.b and 2.1.c.

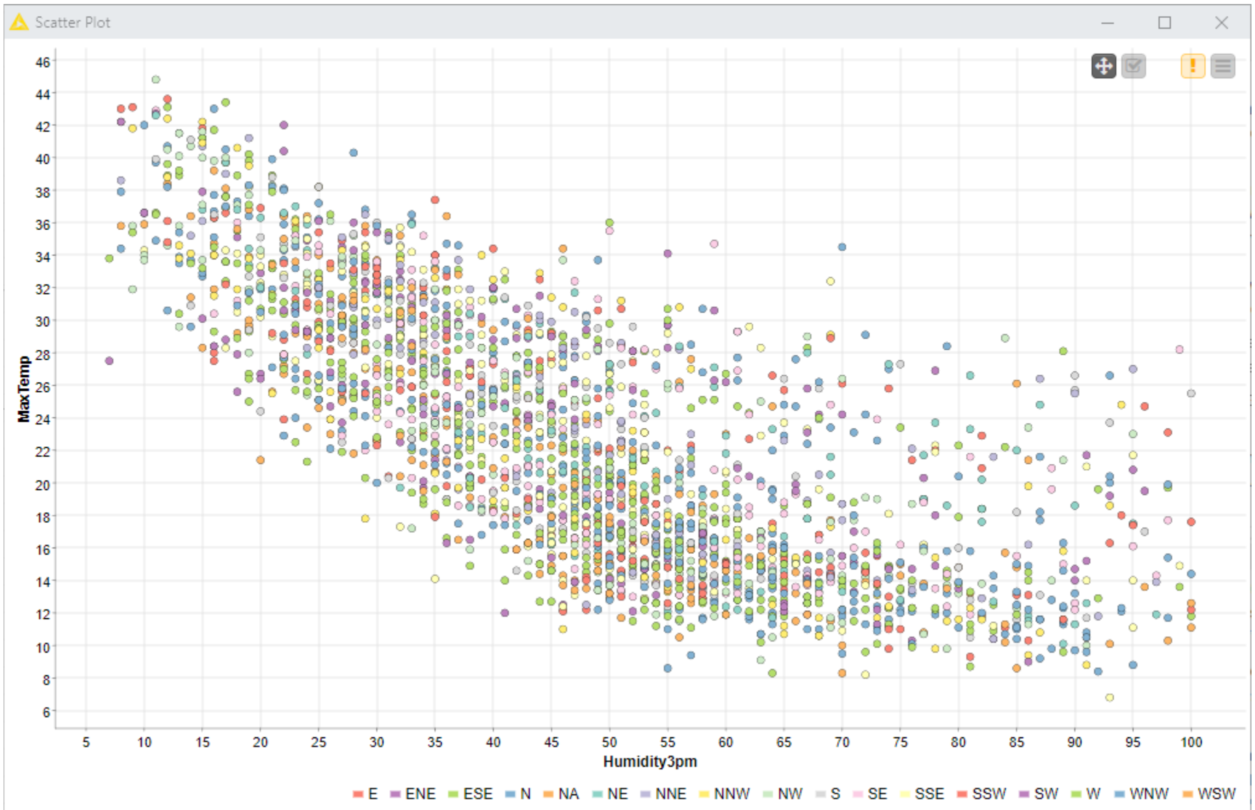


Figure 2.1.b. Scatter plot of MaxTemp vs Humidity3pm over Wind directions

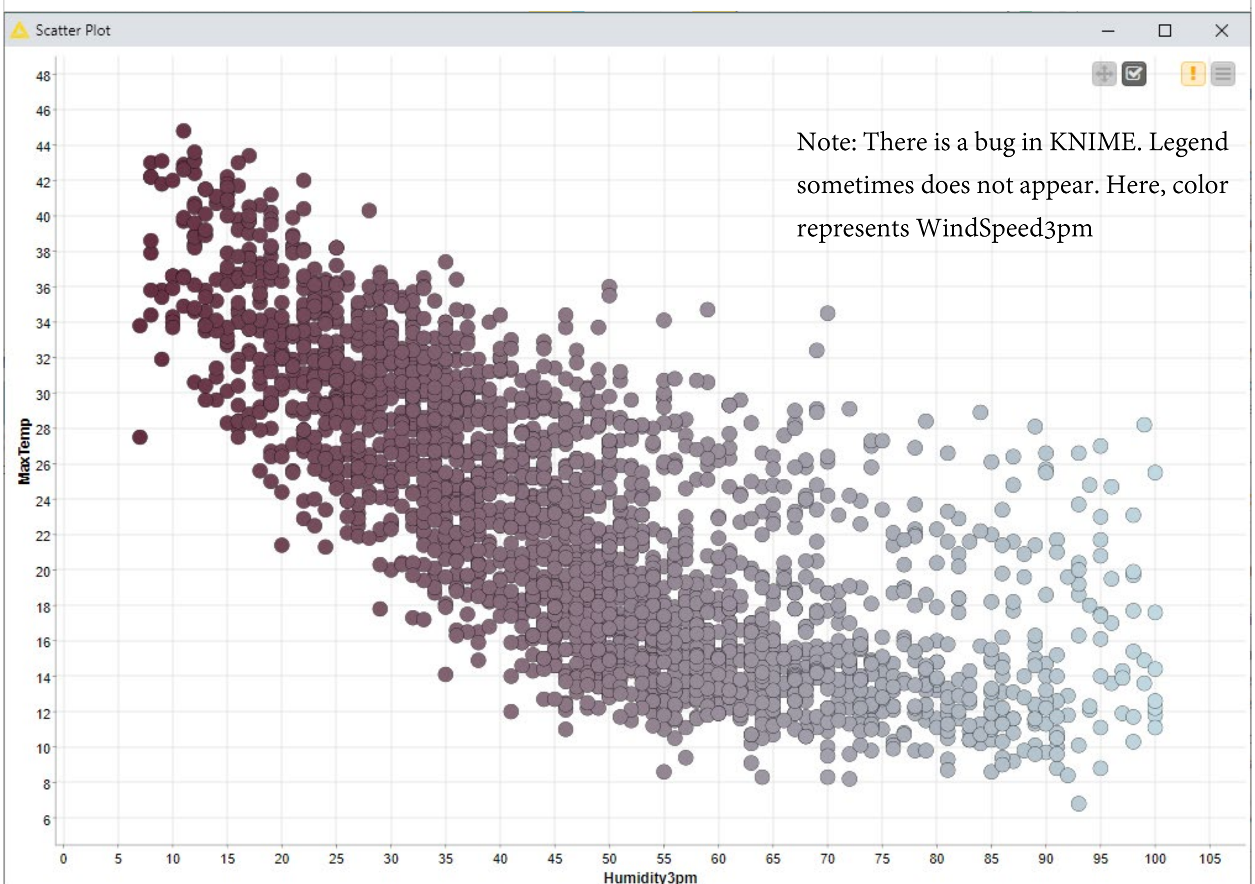
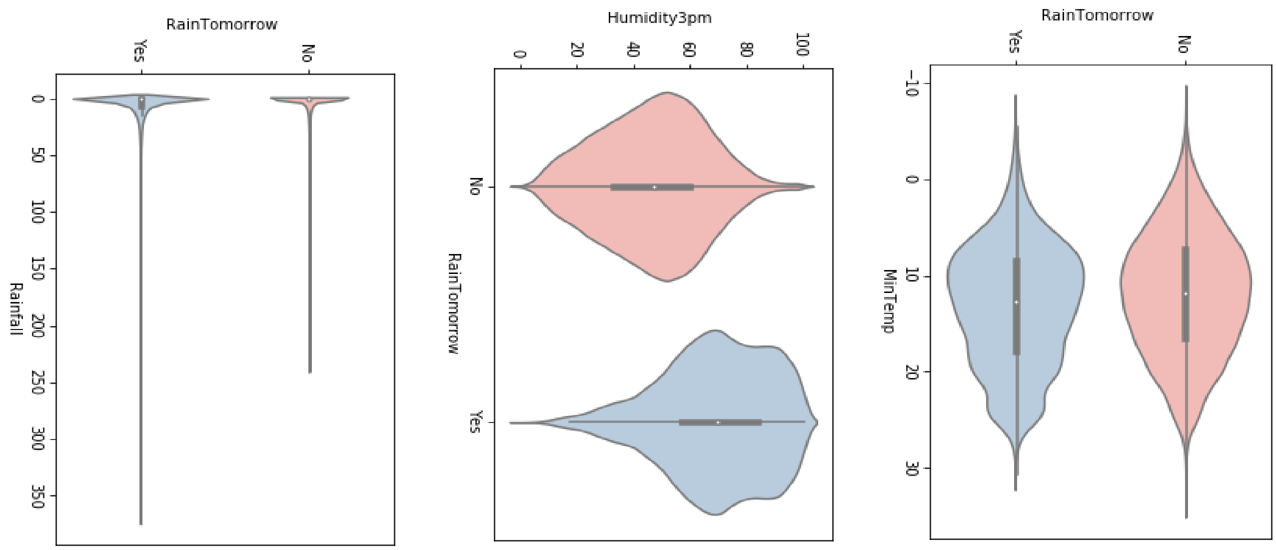


Figure 2.1.c. Scatter plot of MaxTemp vs Humidity3pm over WindSpeed3pm

Further analysis can be made by use of Violin plots as shown Figure 2.2.



(a) **(b)** **(c)**

Figure 2.2. Violin plots of various distribution in our dataset

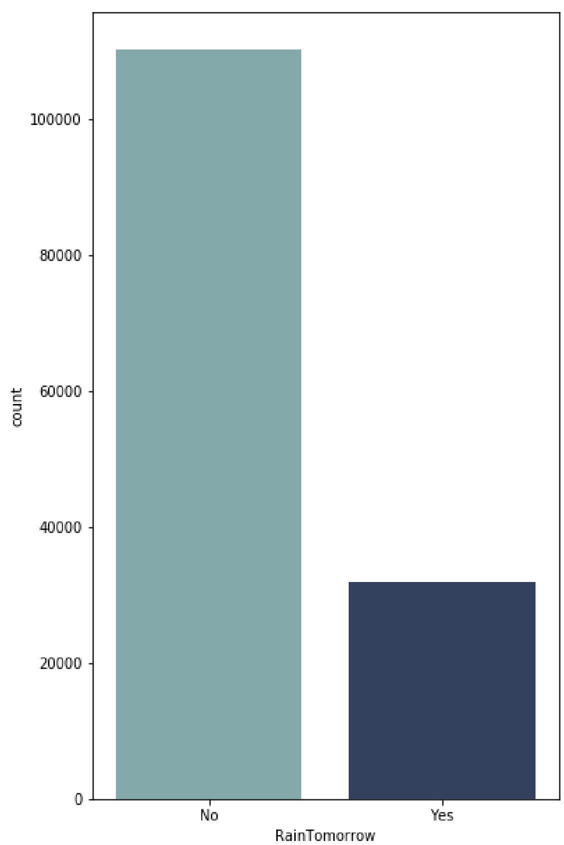


Figure 2.3. Visualization number of *yes* and *no* in AUS Weather Data

From statistical point of view, similar shape is observed in Figure 2.2.a, except **`Yes`** values in **`Rainfall`** seems to have longer tale towards higher values. In Figure 2.2.b., distributions seem to be mirrored for **`RainTomorrow`** in Huminity3pm. Both **`Yes`** and **`No`** in **`RainTomorrow`** has similar distribution in terms of its number in **`MinTemp`**. When we look at the Figure 2.3, we see that number of **`No`** in **`RainTomorrow`**, in general count, are overwhelmingly larger compared to **`Yes`** count. Hence, the data is *imbalanced*. Same info can be obtained in KNIME as well, using Value Counter node (See Figure 2.4 for exact occurrences).

Row ID	I count
No	110316
Yes	31877

Figure 2.4. Occurrences of *yes* and *no* in KNIME

Although we have got rid of the columns which included most missing values, we also need to check if our data is has missing values in rows, using Missing Value node in KNIME If so, we must remove them.

Operation	# of Rows	# of Columns
Original (No operation applied)	142,193	24
Filtered (7 columns excluded)	142,193	17
Missing Value node	119590	17

Figure 2.5. Dimension of AUS Weather Dataset after each operation

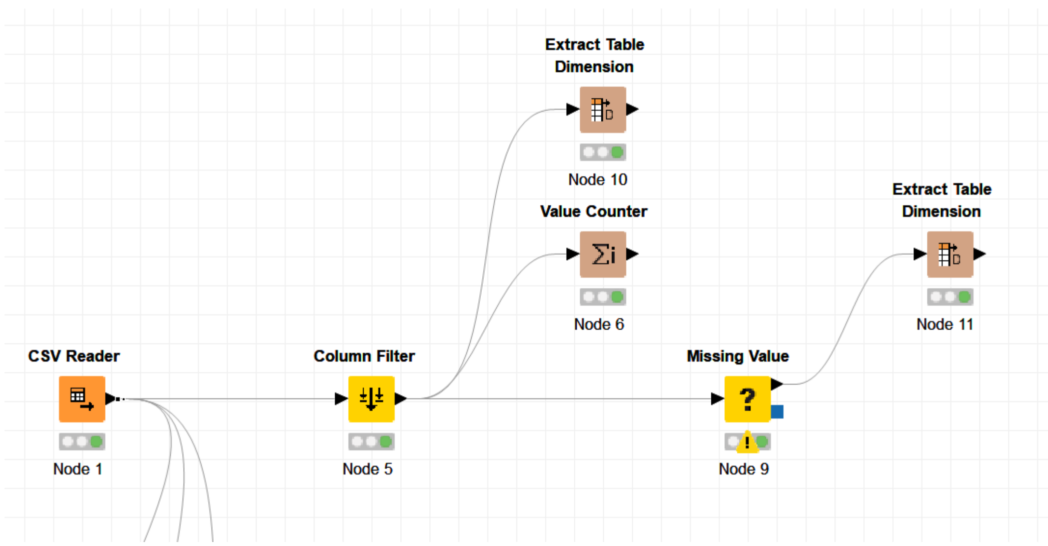


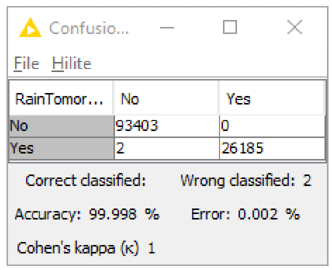
Figure 2.6. Related nodes in KNIME

3. Implementation of Two Approaches

Approach I: Random Forest Classifier

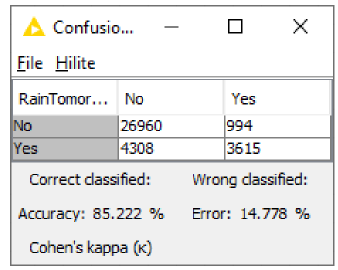
As first approach, let us implement Random Forest Classifier. In KNIME Analytics Platform, this can be done by Random Forest Learner and Random Forest Predictor nodes. To view the accuracy, Scorer node is appended at the end of Random Forest Predictor. Here, we need to configure Scorer such that the scoring will be performed over the columns of **`RainTomorrow`** and **`Prediction (RainTomorrow)`** variables.

In my first attempt, I did not use any partitioning node. As seen by Confusion matrix in Figure 3.1., this resulting in an exceptionally high accuracy – 99.998%, which could be an indication of *overfitting*.



RainTomor...	No	Yes
No	93403	0
Yes	2	26185

Figure 3.1. Confusion matrix in without partitioning

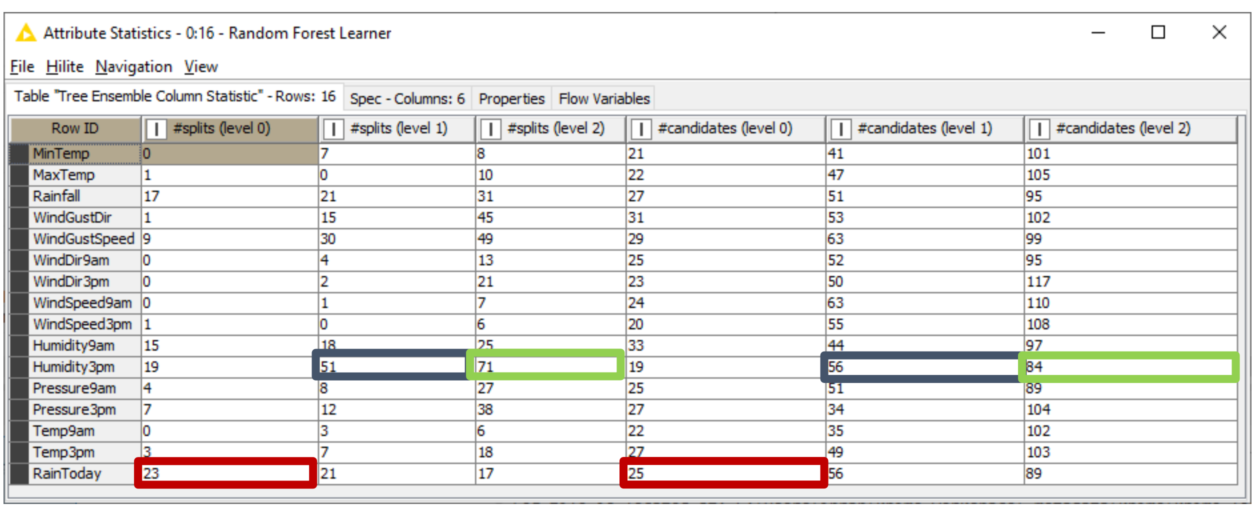


RainTomor...	No	Yes
No	26960	994
Yes	4308	3615

Figure 3.2. Confusion matrix with use of Partitioning

Therefore, I added Partitioning node to see its accuracy on testing set. The result is seen in Figure 3.2. The partitioning was 70% but changing it did not affect much. (±7%)

Another interesting finding is that, in level zero, **`RainToday`** seems to be the most important feature, whereas **`Humidity3pm`** is used for first and second layers.



Row ID	#splits (level 0)	#splits (level 1)	#splits (level 2)	#candidates (level 0)	#candidates (level 1)	#candidates (level 2)
MinTemp	0	7	8	21	41	101
MaxTemp	1	0	10	22	47	105
Rainfall	17	21	31	27	51	95
WindGustDir	1	15	45	31	53	102
WindGustSpeed	9	30	49	29	63	99
WindDir9am	0	4	13	25	52	95
WindDir3pm	0	2	21	23	50	117
WindSpeed9am	0	1	7	24	63	110
WindSpeed3pm	1	0	6	20	55	108
Humidity9am	15	18	25	33	44	97
Humidity3pm	19	51	71	19	56	84
Pressure9am	4	8	27	25	51	89
Pressure3pm	7	12	38	27	34	104
Temp9am	0	3	6	22	35	102
Temp3pm	3	7	18	27	49	103
RainToday	23	21	17	25	56	89

Figure 3.3. Attribute Statistics

Using Receiver Operating Characteristic (ROC) Curve in KNIME, we confirm that Humidity3pm has larger area under the curve, as shown in Figure 3.4. ROC Curve node is appended to the output of Random Forest Predictor.

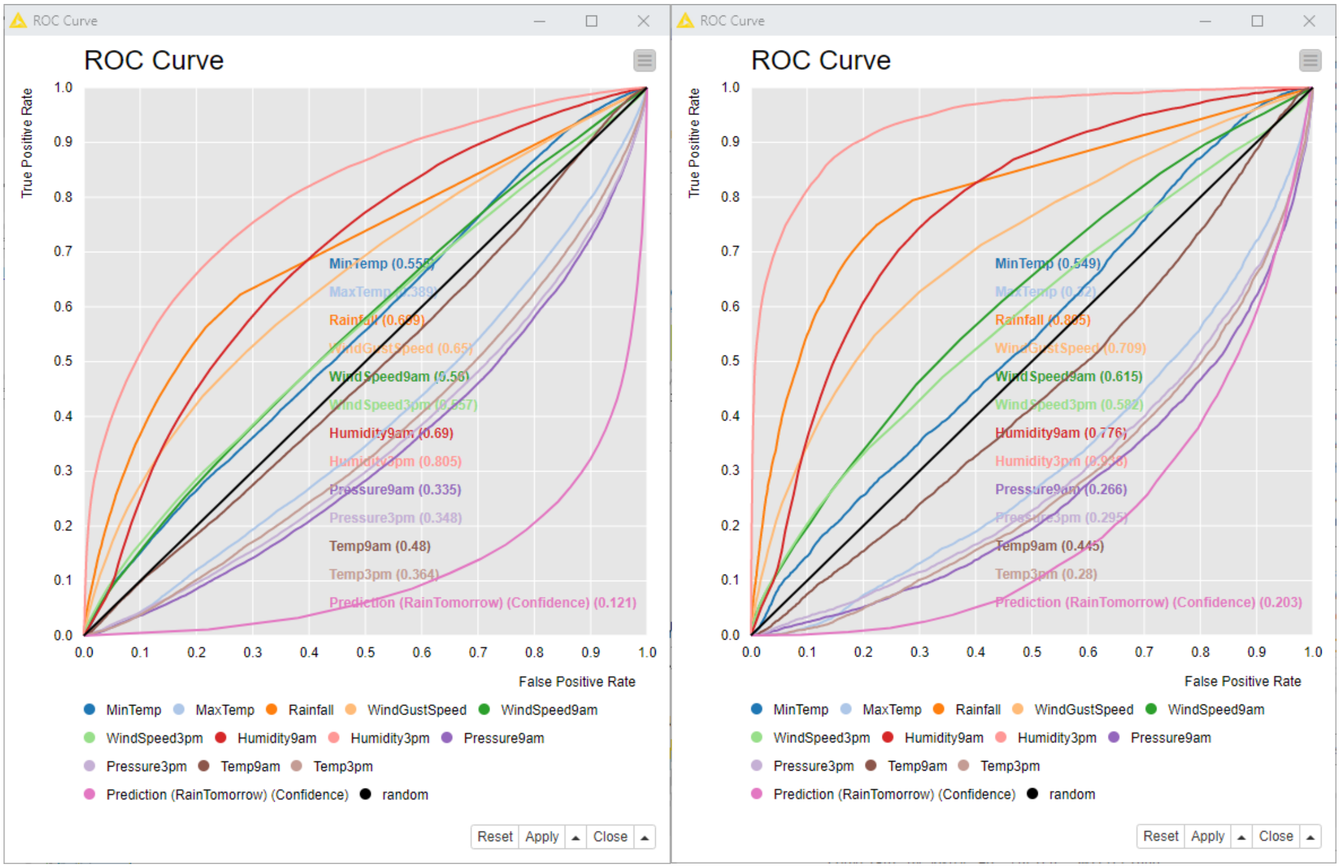


Figure 3.4. ROC curves of setups of no partition (on the left) and partition of 70% (on the right)

I have selected different options in Random Forest Learner with various parameters, and in each setup, the accuracies are shown in Table 3.1, on text page. Unsurprisingly, use of larger parameters of tree depth, minimum node size, or even unlimiting them, with higher number of models yields higher accuracy. However, the overall accuracy seems to not increase after having more than 100 model.

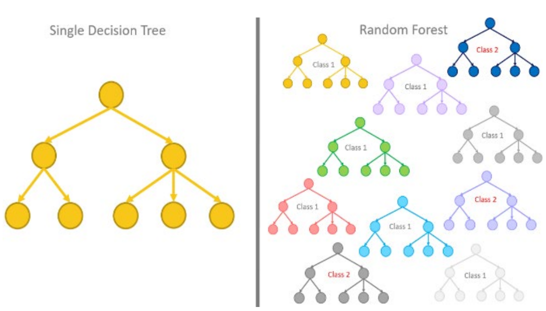


Figure 3.5. Illustration of random forest: More decision trees

Tree options of *Information gain*, *Information gain index*, or *Gini index* did not make much impact in terms of accuracy.

Lastly, for Random Forest Approach, I attempted to analyze the effect of normalization on accuracies. Hence, min-max normalization and Z-score normalization have been applied. However, the results were almost identical of without

Tree Options	Tree depth	Min Node size	# of Model	Accuracy
Information Gain	2	1	2	79.463 %
Information Gain	5	2	20	83.747 %
Information Gain	10	2	100	85.071 %
Information Gain	Not limited	Not specified	2	81,771 %
Information Gain	Not limited	Not specified	100	85.236 %
Information Gain	Not limited	Not specified	500	85.328 %
Information Gain Ratio	2	1	2	79.315 %
Information Gain Ratio	5	2	20	83.354 %
Information Gain Ratio	10	2	100	84.756 %
Information Gain Ratio	Not limited	Not specified	2	82.069 %
Information Gain Ratio	Not limited	Not specified	100	85.222 %
Information Gain Ratio	Not limited	Not specified	500	85.358 %
Gini Index	2	1	2	79.616 %
Gini Index	5	2	20	83.987 %
Gini Index	10	2	100	84.993 %
Gini Index	Not limited	Not specified	2	81.520 %
Gini Index	Not limited	Not specified	100	85.135 %
Gini Index	Not limited	Not specified	500	85.303 %

Table 3.1. Random Forest Classification Accuracies

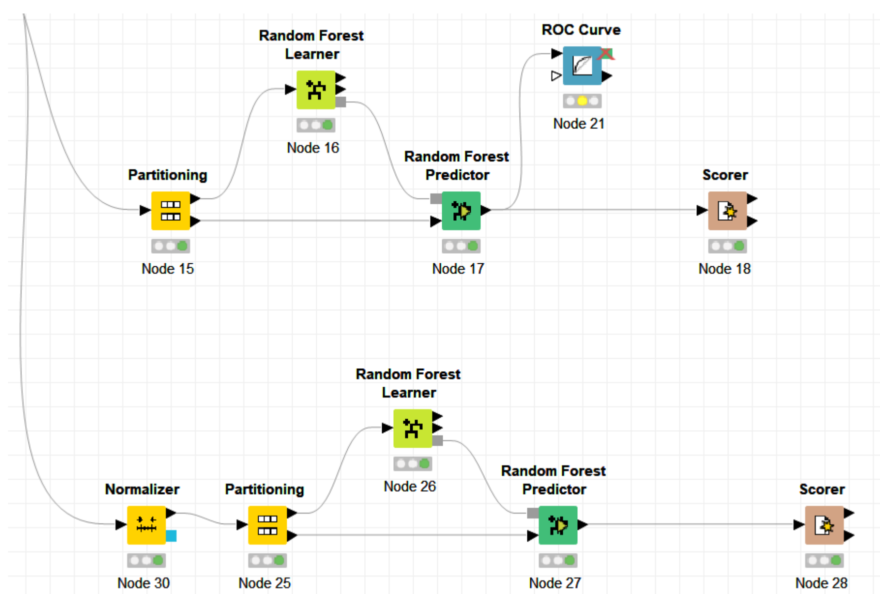


Figure 3.6. Random Forest Classifier in KNIME

Approach II: Logistic Regression

Implementation of Logistic Regression, which is our second approach can be done by Logistic Regression. To view the accuracy, Scorer node is appended at the end of Logistic Regression Predictor. As we did before, Scorer is configured for **`RainTomorrow`** and **`Prediction (RainTomorrow)`** variables.

Without getting into the parameter selection, I have executed Logistic Regression Learner and Predictor nodes in default settings, without and with partitioning our data, and their accuracies are shown in Figures 3.7, 3.8 and 3.9, respectively.

Unlike before, it seems that changing partitioning percentage impacts the accuracy dramatically. However, I realized that this is **not correct**. When I executed the nodes several times, I have obtained averages as shown in Table 3.2. Here, I have taken the average of three execution, to be precise. In each step, the partitioning was kept being performed randomly.

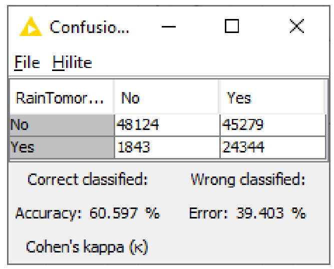


Figure 3.7. Confusion matrix in without partitioning

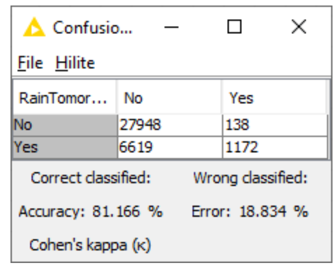


Figure 3.8. Confusion matrix with use of Partitioning of 70%

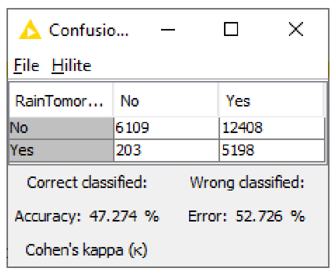


Figure 3.9. Confusion matrix with use of Partitioning of 80%

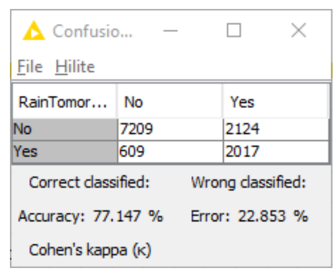


Figure 3.10. Confusion matrix with use of Partitioning of

Partitioning	Execution 1	Execution 2	Execution 3	Avg Accuracy
None	82.718 %	72.678 %	60.597 %	71.998 %
60%	84.196 %	83.175 %	83.552 %	83.641 %
70%	81.166 %	84.485 %	83.000 %	82.883 %
80%	47.274 %	83.895 %	83.744 %	71.638 % (err)
90%	77.147 %	78.803 %	81.704 %	79.218 %

Table 3.2. Average accuracy of three executions of Logistic Regression with default parameters

To my knowledge, the value highlighted with blue in Table 3.2, was related with partitioning random value and I believe it is safe to ignore it, as in second and third execution of partitioning of 80 % resulted in a lot higher accuracy. On the other hand, the table clarifies that as partitioning percentage increases, the accuracy decreases slightly. Since the difference is ignorable, we can keep partitioning percentage as 70%.

The accuracies we have obtain is not inadequate, however as prompted in KNIME console, we can try different parameters and normalization techniques.

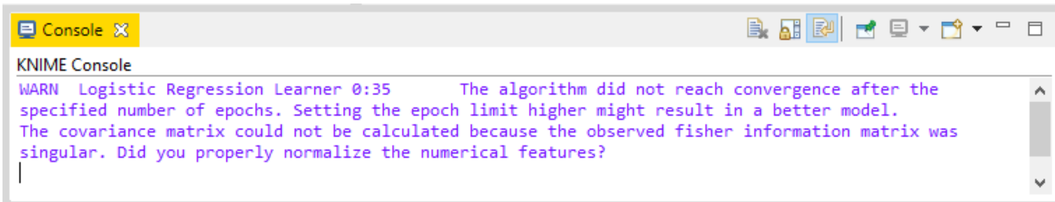


Figure 3.11. Warning in console about convergence and normalization.

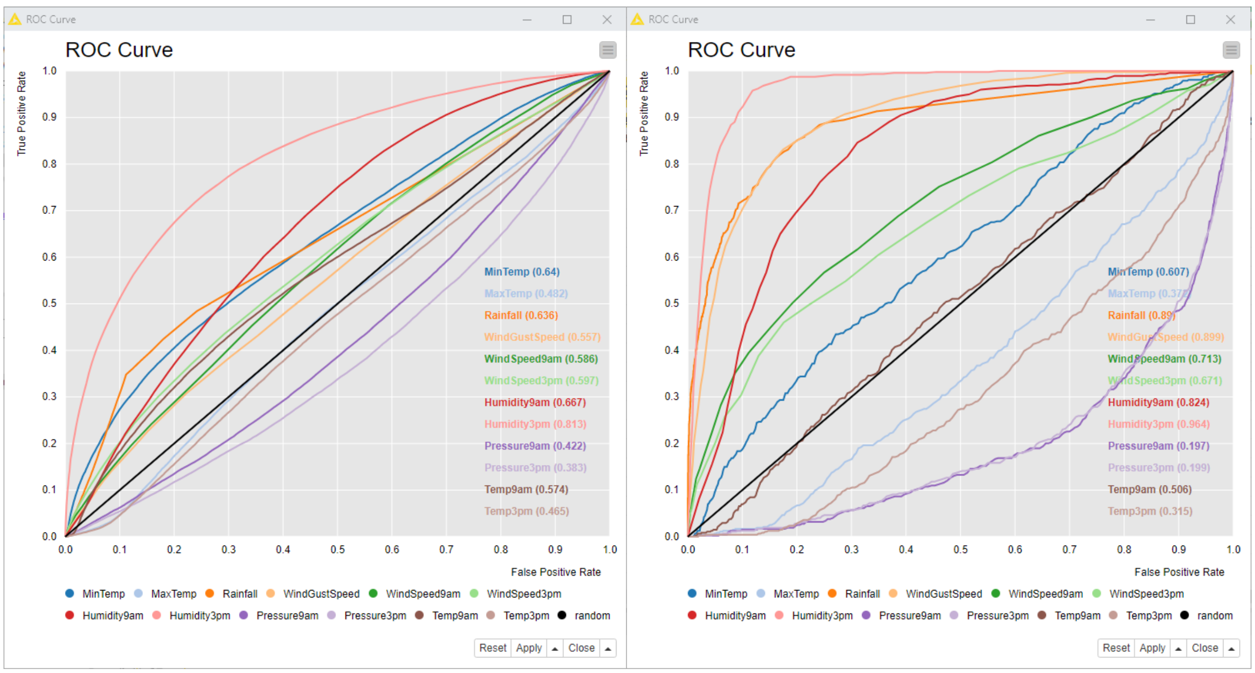


Figure 3.12. ROC curves of setups of no partition (on the left) and partition of 70% (on the

Like previous approach, for Logistic regression, I have selected different options and parameters in configuration of Logistic Regression Learner. Those are maximal number of epochs, stop size and regularization options.

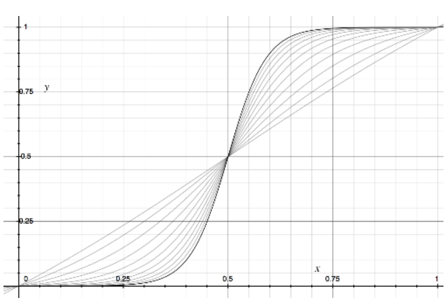


Figure 3.13. Illustration of Logistic Regression

As reminded by KNIME console in Figure 3.11, I have applied normalization. Table 3.3 on the next page gives a detailed information of the effect of normalization and different regularization types.

Without normalization, the accuracies vary. That is why I had to execute each setup three times to make sure the value obtained is not an outlier (shown with orange).

Gauss and Laplace regularization variances were kept at 0.1 and the epsilon value in determines termination condition was set to 1.0E-5 throughout all operations in Table 3.3.

Normalization	# of Epochs	Learning Rate	Regularization	Accuracy 1	Accuracy 2	Accuracy 3
Not applied	100	Fixed: 0.01	Uniform	78 %	80 %	84 %
Not applied	100	Fixed: 0.01	Gauss	84 %	65 %	79 %
Not applied	100	Fixed: 0.01	Laplace	83 %	84 %	83 %
Not applied	100	Fixed: 0.10	Uniform	84 %	70 %	81 %
Not applied	100	Fixed: 0.10	Gauss	77 %	83 %	75 %
Not applied	100	Fixed: 0.10	Laplace	82 %	73 %	80 %
Not applied	500	Fixed: 0.01	Uniform	81 %	82 %	82 %
Not applied	500	Fixed: 0.01	Gauss	83 %	78 %	81 %
Not applied	500	Fixed: 0.01	Laplace	78 %	84 %	81 %
Not applied	500	Fixed: 0.10	Uniform	78 %	85 %	45 %
Not applied	500	Fixed: 0.10	Gauss	52 %	79 %	73 %
Not applied	500	Fixed: 0.10	Laplace	81 %	82 %	78 %
Min-Max	100	Fixed: 0.01	Uniform	85 %	85 %	85 %
Min-Max	100	Fixed: 0.01	Gauss	85 %	85 %	84 %
Min-Max	100	Fixed: 0.01	Laplace	85 %	85 %	85 %
Min-Max	100	Fixed: 0.10	Uniform	85 %	85 %	85 %
Min-Max	100	Fixed: 0.10	Gauss	84 %	85 %	85 %
Min-Max	100	Fixed: 0.10	Laplace	85 %	85 %	85 %
Min-Max	500	Fixed: 0.01	Uniform	85 %	85 %	85 %
Min-Max	500	Fixed: 0.01	Gauss	85 %	84 %	85 %
Min-Max	500	Fixed: 0.01	Laplace	85 %	85 %	85 %
Min-Max	500	Fixed: 0.10	Uniform	85 %	85 %	85 %
Min-Max	500	Fixed: 0.10	Gauss	84 %	85 %	85 %
Min-Max	500	Fixed: 0.10	Laplace	85 %	85 %	85 %
Z-Score	100	Fixed: 0.01	Uniform	85 %	85 %	85 %
Z-Score	100	Fixed: 0.01	Gauss	85 %	85 %	84 %
Z-Score	100	Fixed: 0.01	Laplace	85 %	84 %	85 %
Z-Score	100	Fixed: 0.10	Uniform	85 %	85 %	85 %
Z-Score	100	Fixed: 0.10	Gauss	85 %	85 %	84 %
Z-Score	100	Fixed: 0.10	Laplace	84 %	85 %	85 %
Z-Score	500	Fixed: 0.01	Uniform	85 %	85 %	85 %
Z-Score	500	Fixed: 0.01	Gauss	85 %	85 %	85 %
Z-Score	500	Fixed: 0.01	Laplace	85 %	84 %	85 %
Z-Score	500	Fixed: 0.10	Uniform	85 %	85 %	84 %
Z-Score	500	Fixed: 0.10	Gauss	84 %	85 %	85 %
Z-Score	500	Fixed: 0.10	Laplace	85 %	84 %	84 %

Table 3.3. Logistic Regression accuracies (in three execution) on various settings

Use of normalization not only have removed the warning in console, but allowed us to obtain stable accuracies, as presented in Table 3.3.

However, the algorithm still does not converge which there might be a room for improvement in terms of accuracy. Hence, I tried various (relatively extreme) values of parameters. One of the setting is as follows:

- *Maximal number of epochs* was set to 5000, 10K and 15K.
- *Epsilon*: 1.0E-6 and 1.0E-7
- *Learning rate*: LineSearch and fixed of values of 1.0E-3, 1.0E-4
- *Regularization*: Laplace and Gauss

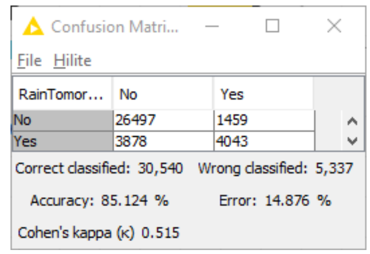


Figure 3.14. Accuracy example

With these settings, computational times was more than 20 minutes for each. However, the accuracy was 85.124 %, which is not any better than what we have previously obtained in Table 3.3. with normalization applied. This concludes that, the maximum accuracy I could obtained in Logistic Regression in AUS_Weather dataset was around 85%.

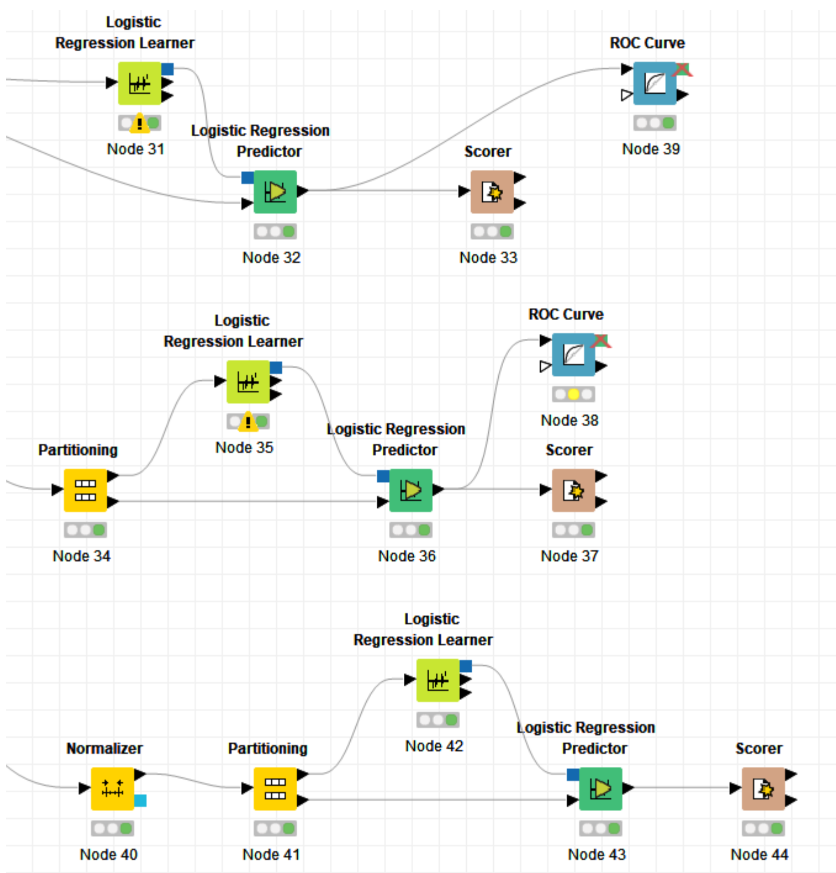
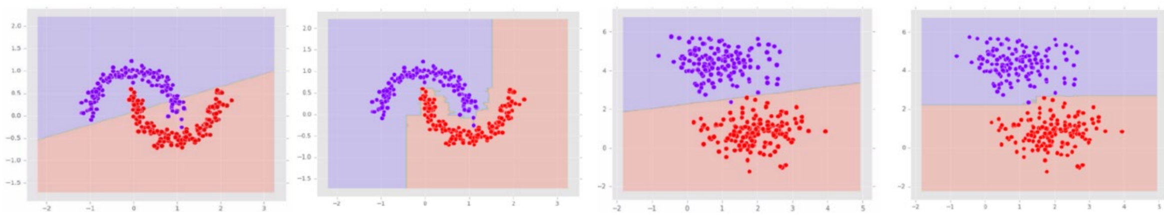


Figure 3.15. KNIME workflow for Logistic Regression

4. Performance Evaluation and Comparison

Both Logistic regression and Random Forest approaches are used in binary classification problems. As I have learned from literature that, in an example dataset shown in Figure 4.1 (not the dataset I have used), Random Forest is able to make more accurate decision boundary, depending on the complexity of the data. However, for our problem, the accuracy of each approach was almost the *same*.



(a) (b) (c) (d)

Figure 4.1. Decision boundaries between binary classes for an exemplary data
(a) and (c): Logistic Regression; (b) and (d): Random Forest

► ***Accuracy-wise comparison***

The best accuracy we obtained for Random Forest approach was 85.328%, using 500 model and without limiting Tree depth and Min Node size. On the other hand, for Logistic Regression, it was 85.567. However, in Logistic Regression, we observe that only having values normalized increases the accuracy significantly. Other parameters, even the model is converged successfully with different regularizations applied, did not make as huge impact as normalization. Various normalization did not affect the accuracy in Random Forest approach.

► ***Memory-wise (space complexity)***

Memory wise Random Forest allocated much more memory (in some settings, 10+ GB was being used by just KNIME). For the case of Logistic Regression, the memory usage was a lot less, around 1GB. My assumption is that Logistic Regression used less memory because it relies more of calculations.

► ***Computation power-wise***

No heavy background task was running, all other utilizations were less than 1%. The total CPU usage was around 20% during the execution of Logistic Regression. On the other hand, in Random Forest, it jumped to 100%.

► ***Computation timewise (time complexity)***

The time required for Random Forrest algorithm was less, in general, compared to Logistic Regression. Even with parameters which forces program to spend more time, Random Forrest algorithm was producing the output in less than 5 minutes in my setups. In Logistic regression, execution time was linearly increasing with some parameters, e.g. number of epochs.

5. Discussion and Conclusion

With the issues in dataset, I have learned the ways in which I could correct the data. Although there were columns which we needed to exclude due to both our purpose (prediction country-wise, not over cities) and its highly included missing values, excluding the columns were not enough, because there were missing values in the columns we needed as well. For that purpose, those entries are removed. Fortunately, there were no dirty or noisy data. However, the target variable **`RainTomorrow`** was imbalanced, in other words, number of **`no`**'s in **`RainTomorrow`** was overwhelmingly larger than number of **`yes`**'s (approximately four times, as explained in Section 2: Analysis of Data).

The impact of normalization (Z-score and Min-Max) are observed in both algorithms. While having or not having normalization did not make much impact in accuracy in Random Forest, the same is not true for Logistic Regression. Without normalization (any type), the accuracies were not consistent and lower, compared to the accuracies obtained with normalized data. Detail analyses are in relative sections.

The dimension transformation such as PCA or LDA is not used. For both approaches, overfitting possibility had been considered. As I have explained in Section 3, Hence, in the Random Forest approach, maximum allowable tree depth was increased while the accuracy was being monitored. For Logistic regression, regularization options of *Uniform*, *Gauss* and *Laplace* was performed in various setup.

The data science management method I attempted was *Knowledge Discovery in Database* (KDD in short), because the goal of prediction is assessed based on the given dataset. Hence, the answer could lay inside the data itself. From observing the results, and interpretations were made (properties and relation among column data, in Section 2). Segmentation or clustering were not needed.

The most frustrating part was exhaustive search on of different parameters, especially in Logistic Regression. It not only took a large portion of the time I have efforted over this project, it also confused me with strange and unexpected results it produced. I had to run several times to make sure that the result is not some randomization problem (partitioning etc.) I had encounter with.

As I mentioned, parameter selection mostly based on trial and error, and exhaustively testing as many combinations of parameters as possible (see Table 3.3. Logistic Regression accuracies in Section 3.). I don't think we can use the same parameters on a different dataset, without trying. Depending on the complexity, for example, we might need different number of model in Random Forest.

This Project help me to get familiar with KNIME environment. I have gained the experience of applying concepts that have been covered in related courses such as Artificial Intelligence, Data Science, and Data Visualization. Additionally, reading terms used in data science literature allowed me to have an acknowledge of related active research areas.

References

[1] Sadi Evren Seker, *Lecture Notes*, Introduction to Artificial Intelligence and Introduction to Data Science course. (2019 Fall and 2020 Spring)

[2] Rain Prediction in Australia Dataset Available at: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.

[3] KNIME Analytic Platform. Available at: https://www.knime.com

[4] KNIME TV Youtube Channel. Available at: https://www.youtube.com/user/KNIMETV

[5] KNIME Documentation. Available at: https://docs.knime.com

[6] Anaconda Python3 Distribution. Available at: https://www.anaconda.com

[7] Sarang N, *AUC – ROC Curve*. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[8] Seaborn: Statistical Data Visualization. Available at: https://seaborn.pydata.org/examples/index.html

[9] The Ultimate Python Seaborn Tutorial. Available at: https://elitedatascience.com/python-seaborn-tutorial

[10] Rosarila Silipo, *From a Single Decision Tree to a Random Forest*, (2019). Available at: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

[11] Kirasich, Kaitlin, Trace Smith, and Bivin Sadler. "Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets." SMU Data Science Review 1.3 (2018).

[12] The python Graph Gallery. Available at: https://python-graph-gallery.com/category/seaborn/

Appendix: KNIME Workflow