Rainfall in Australia

JARROD KENG YEN YONG (218271795)

HARRY WILLIAM LODGE (218241616)

VIET NAM NGUYEN (218459058)

ALEXANDER PAK YU LAI (218401269)

Introduction

The Australian weather broadcast or the appropriate agencies, comprehends tomorrow's weather, specifically whether it will be raining or not.

Dataset of day-to-day weather collected across various stations over a period of 10 years.

Aim to return prediction of tomorrow rain by producing the possibilities.

Data Description

Categorical Variables
Date
Location
WindGustDir
WindDir9am
WindDir3pm
RainToday
RainTomorrow

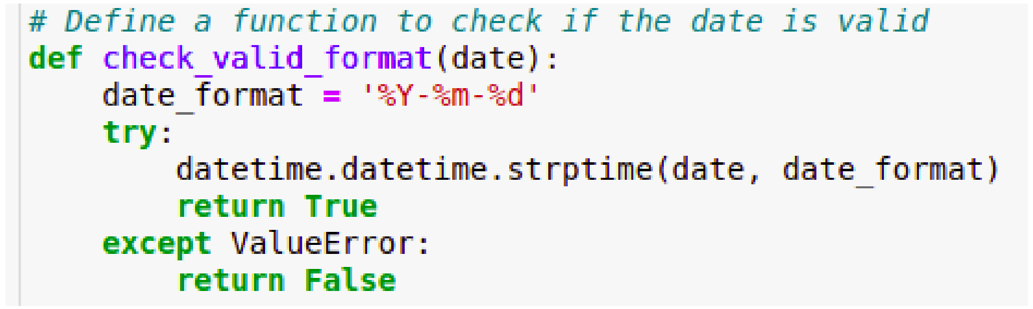
Numerical Variable	
MinTemp	WindSpeed3pm
MaxTemp	Humidity9am
Rainfall	Humidity3pm
Evaporation	Pressure9am
Sunshine	Pressure3pm
WindGustSpeed	Cloud9am
WindSpeed9am	Cloud3pm
Temp9am	RISK_MM
Temp3pm	

Data Cleansing

A	Find and correct errors in categorical variables
B	Count the total number of missing values for each column
C	Drop columns with more than 10% missing values
D	Drop rows with more than 20% missing values
E	Calculate z-score for numerical variables and replace outliers
F	Replace all missing values by implementing appropriate algorithms

A. Find and correct errors in categorical variables

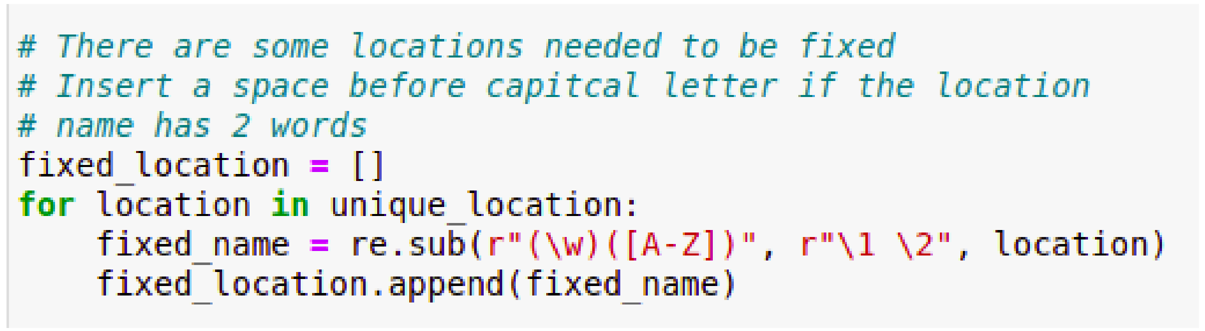
DATE VARIABLE



```
# Define a function to check if the date is valid
def check_valid_format(date):
    date_format = '%Y-%m-%d'
    try:
        datetime.datetime.strptime(date, date_format)
        return True
    except ValueError:
        return False
```

A. Find and correct errors in categorical variables

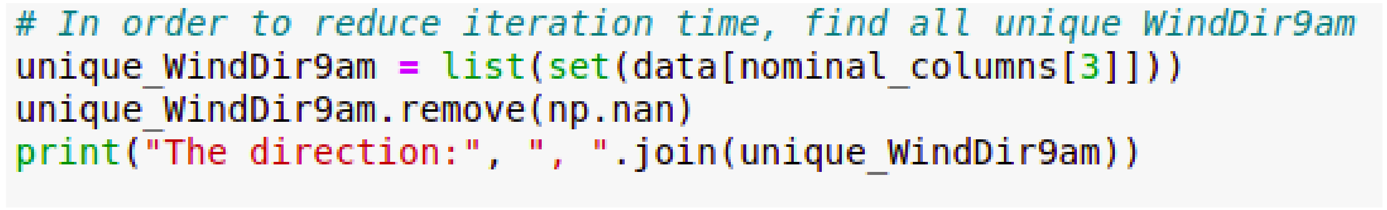
LOCATION VARIABLE



```
# There are some locations needed to be fixed
# Insert a space before capitcal letter if the location
# name has 2 words
fixed_location = []
for location in unique_location:
    fixed_name = re.sub(r"(\w)([A-Z])", r"\1 \2", location)
    fixed_location.append(fixed_name)
```

A. Find and correct errors in categorical variables

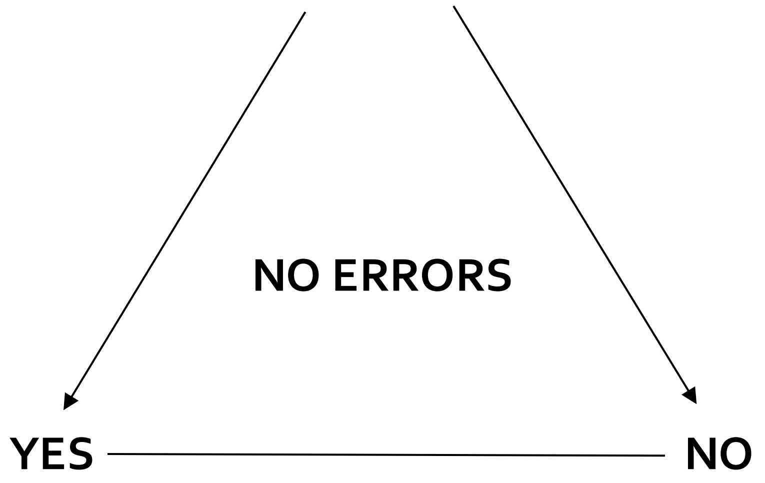
WindGustDir| WindDir9am|WinDir3pm



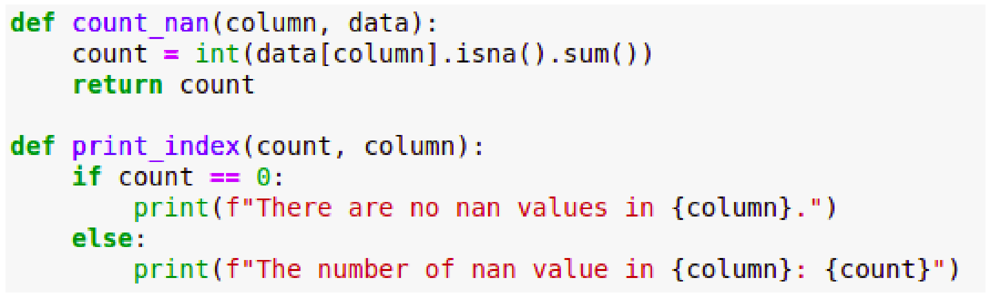
```
# In order to reduce iteration time, find all unique WindDir9am
unique_WindDir9am = list(set(data[nominal_columns[3]]))
unique_WindDir9am.remove(np.nan)
print("The direction:", ", ".join(unique_WindDir9am))
```

A. Find and correct errors in categorical variables

RAINTODAY | RAINTOMORROW



B. Count the total number of missing values for each column

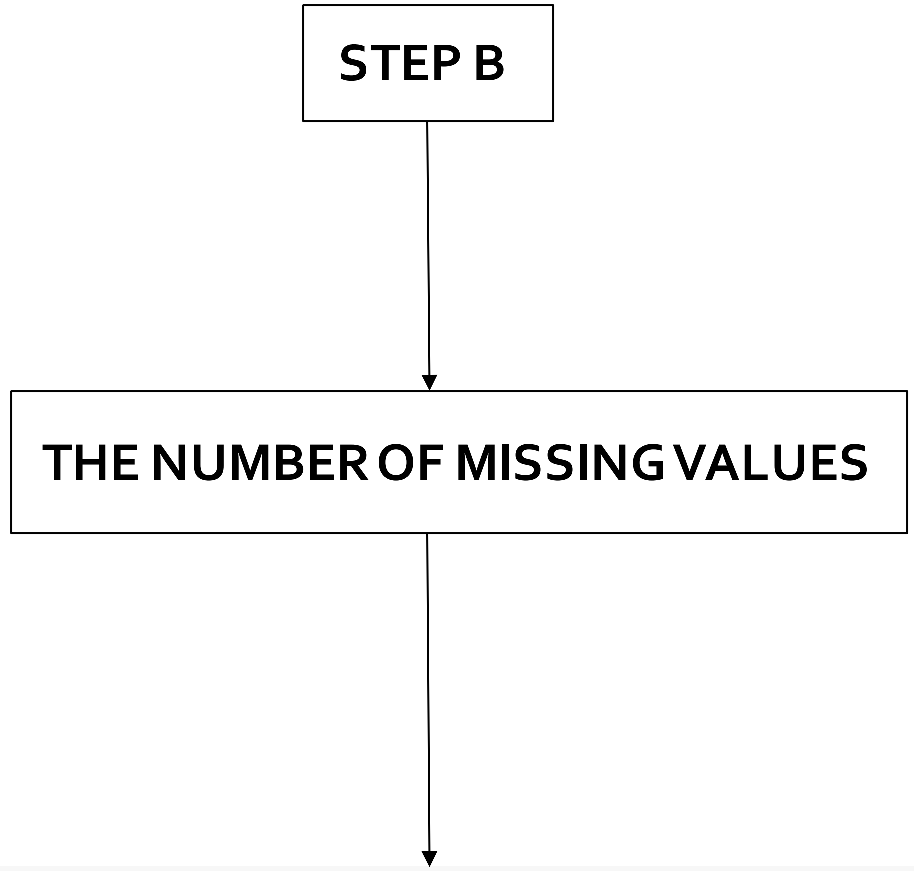


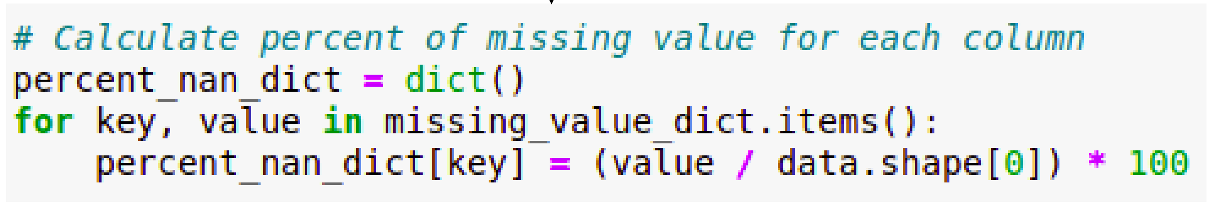
```python
def count_nan(column, data):
    count = int(data[column].isna().sum())
    return count

def print_index(count, column):
    if count == 0:
        print(f"There are no nan values in {column}.")
    else:
        print(f"The number of nan value in {column}: {count}")
```

Variables	Total Missing Values
Date	0
Location	0
MinTemp	637
MaxTemp	322
Rainfall	1406
Evaporation	60843
Sunshine	67816
WindGustDir	9330
WindGustSpeed	9270
WindDir9am	10013
WindDir3pm	3778
WindSpeed9am	1348
WindSpeed3pm	2630
Humidity9am	1774
Humidity3pm	3610
Pressure9am	14014
Pressure3pm	13981
Cloud9am	53657
Cloud3pm	57094
Temp9am	904
Temp3pm	2726
RainToday	1406
RISK_MM	0
RainTomorrow	0

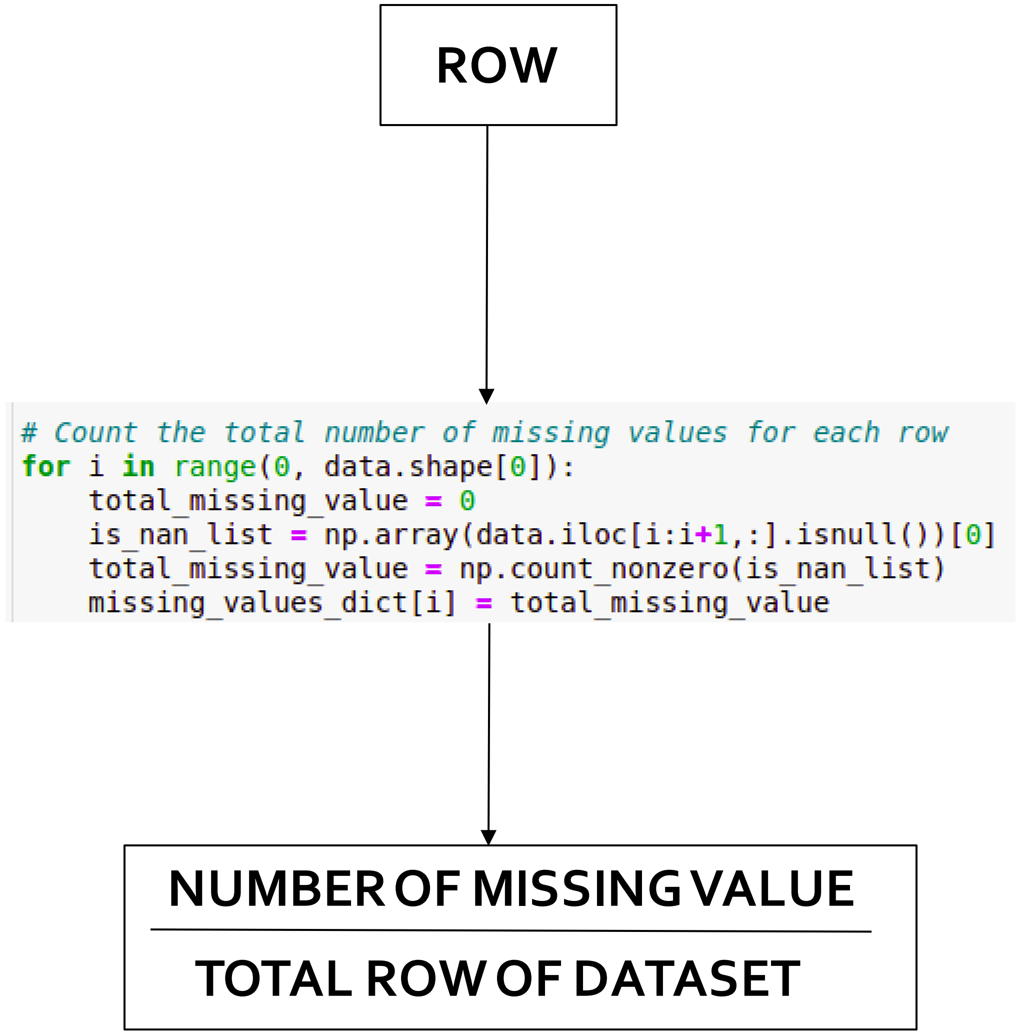
C. Drop columns with more than 10% missing values





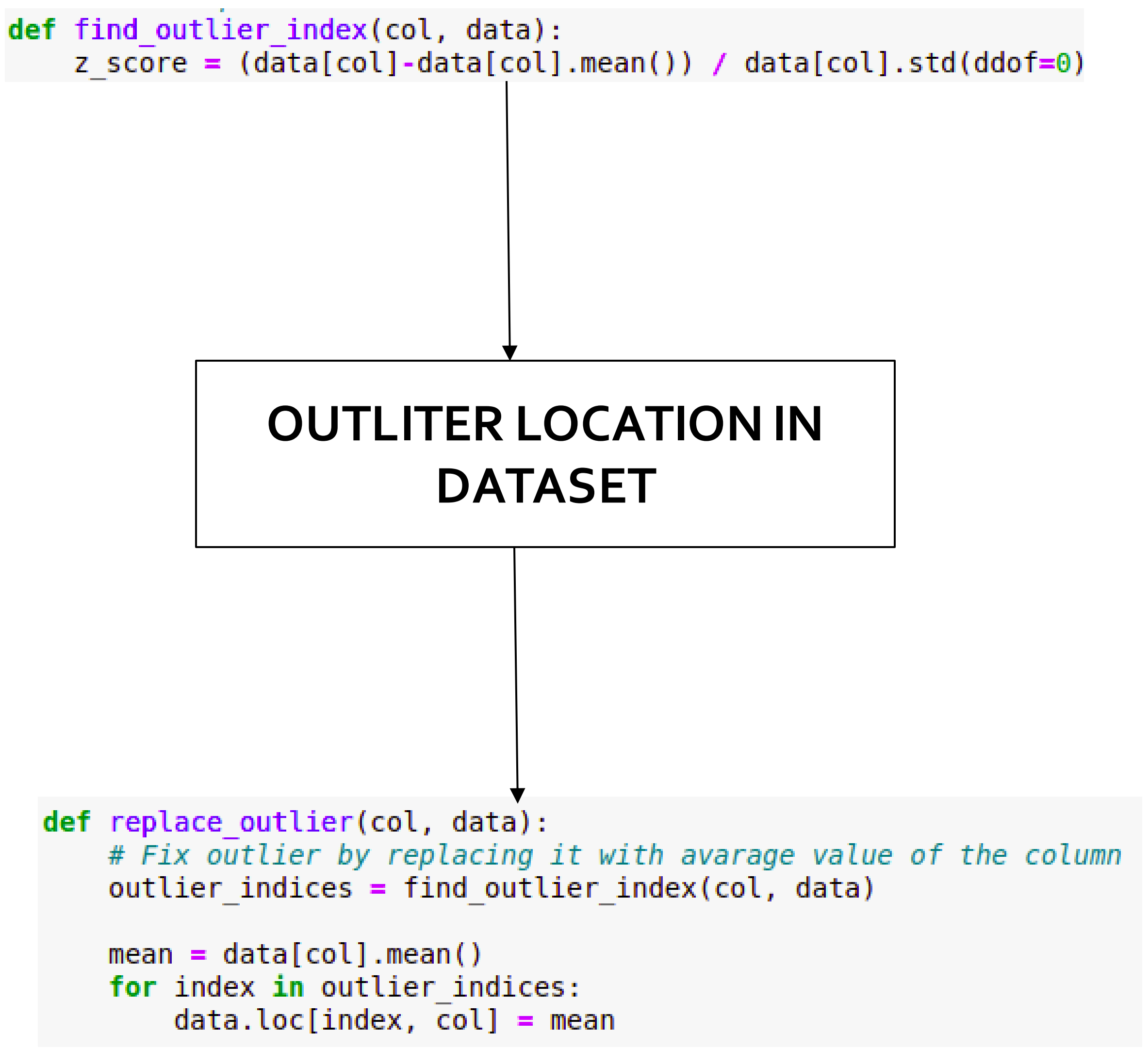
```python
# Calculate percent of missing value for each column
percent_nan_dict = dict()
for key, value in missing_value_dict.items():
    percent_nan_dict[key] = (value / data.shape[0]) * 100
```

D. Drop rows with more than 20% missing values



```python
# Count the total number of missing values for each row
for i in range(0, data.shape[0]):
    total_missing_value = 0
    is_nan_list = np.array(data.iloc[i:i+1,:].isnull())[0]
    total_missing_value = np.count_nonzero(is_nan_list)
    missing_values_dict[i] = total_missing_value
```

E. Calculate z-score for numerical variables and replace outliers

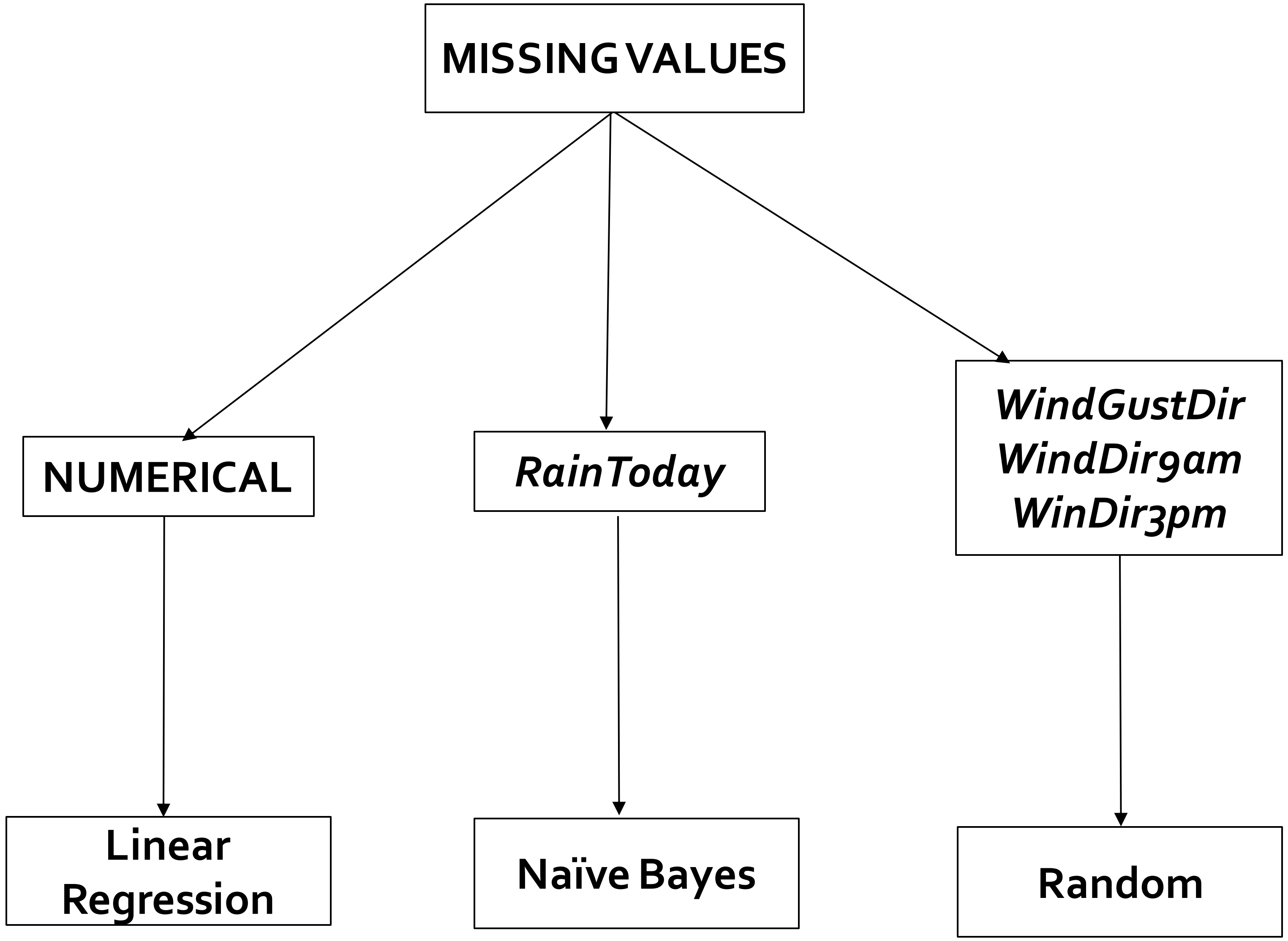


```
def find_outlier_index(col, data):
    z_score = (data[col]-data[col].mean()) / data[col].std(ddof=0)
```

```
def replace_outlier(col, data):
    # Fix outlier by replacing it with avarage value of the column
    outlier_indices = find_outlier_index(col, data)

    mean = data[col].mean()
    for index in outlier_indices:
        data.loc[index, col] = mean
```

F. Replace all missing values by implementing appropriate algorithms



Main Structure and Key Variables

A. Information used in analysis

B. Date and location

C. Min temp and Max temp

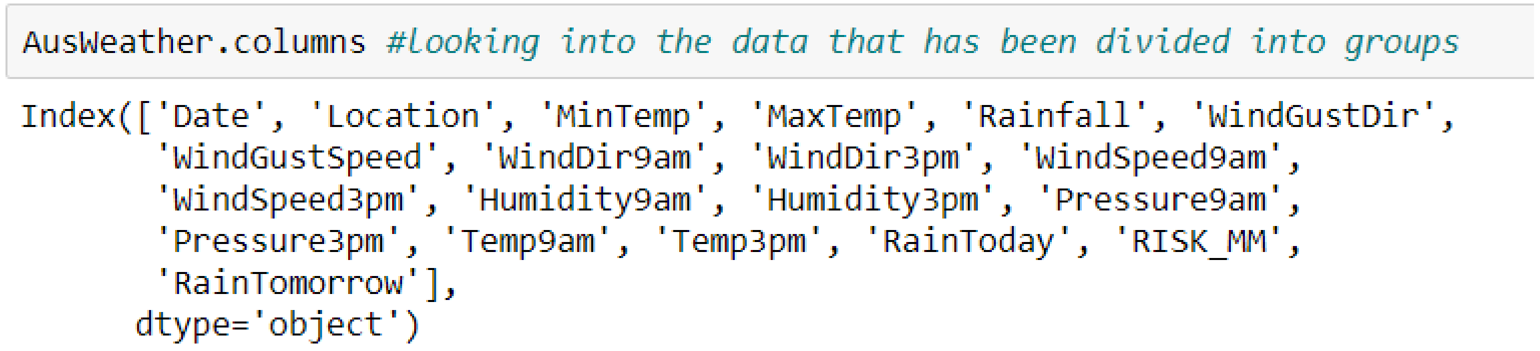
D. Pressure, Humidity and Temp for 9am and 3pm

E. Rain today and tomorrow

G. The other values

A. Information used in analysis

- The list shown is the different values/columns that we could use when analysing the information. It was our job to look at the list of different variables that we could use to help us analyse patterns and trends in the information.

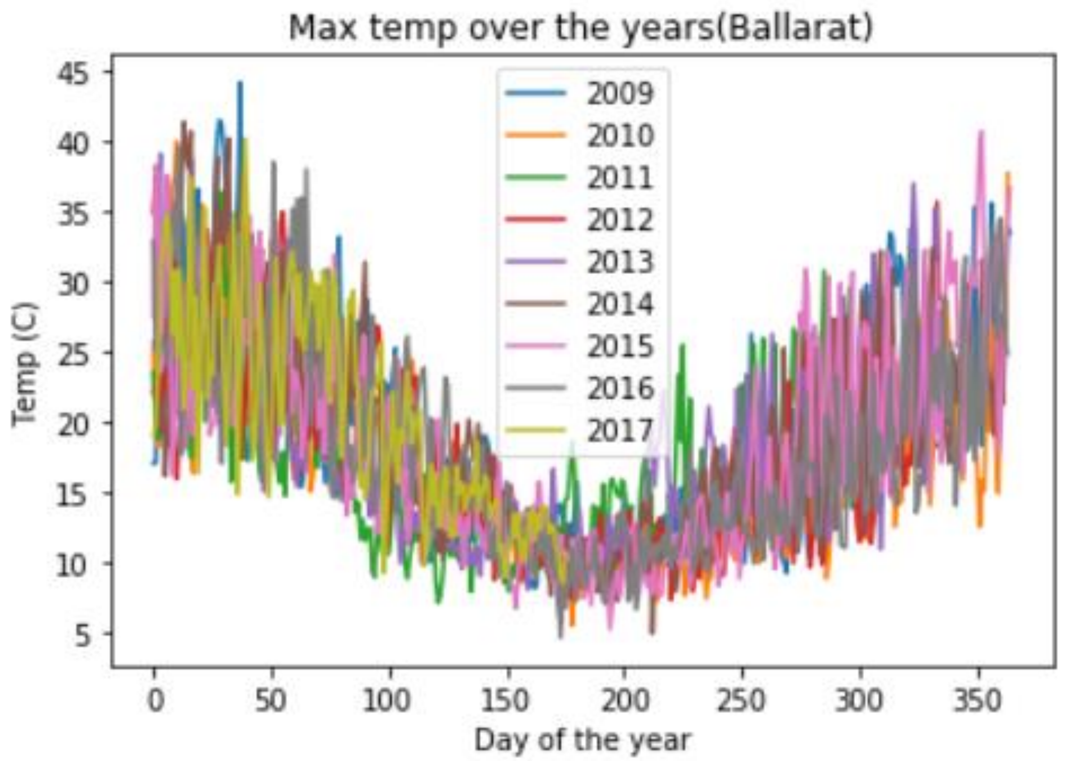


```
AusWeather.columns #looking into the data that has been divided into groups
```

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustDir',
       'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am',
       'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RISK_MM',
       'RainTomorrow'],
      dtype='object')
```

B. Date and location

- The main use for these two variables was to help us know the differences between each row of information. Of the 14000+ entries that we had to look into, the combination of the two variables allow us to break all entries into unique rows.
- The values could also be broken up when looking into each individual location like when we used the location and date to help us know a specific area's max temp as posted throughout the year. By separating the location and then separating by time, we could get area-based analysis that we could compare against each year's record as shown in our analysis.



C. Min temp and Max temp

The max and min temp are used to show us the maximum temperature and minimum temperature that was recorded on that day. We used these temperatures when we wanted to identify if there was any correlation to temperature rising and falling and any of the other variables.

D. Pressure, Humidity and Temp for 9am and 3pm

- Perhaps one of the most useful sets of variables that we could use so that we could observe the amount difference between the morning and afternoon.

- As we originally didn't find much information in the section where there was a rise in temperature during the day. We used pressure and humidity as the key values we looked into for the analysis and helped us reach a usable conclusion with our studies.

E. Rain today and tomorrow

These were the key values that we used to make sure our assumptions were correct. The aim of these variables were to help us identify if there was rain on the following day. By using that information we could predict based on the values given to us if there would be “rain today” on the following day without the need for additional analysis.

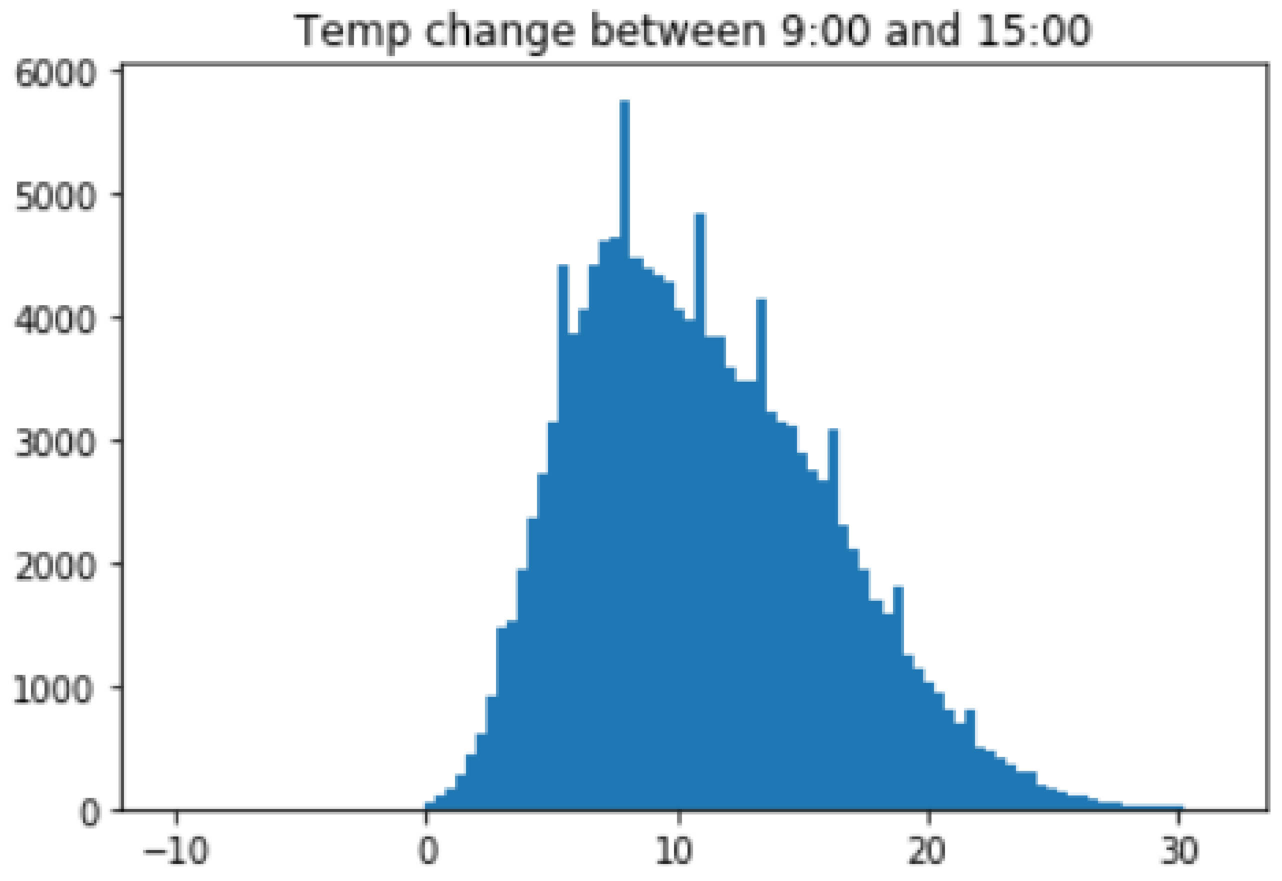
G. The other values

- The rest of the values, while useful, weren't used too much in our calculations and meant that we didn’t have a way to use them during the analysis, other than the generic analysis that we had used during testing that we had used so that we could make sure all our calculations for both data cleaning and predicting are correct.

Data Patterns

Temp Based Patterns

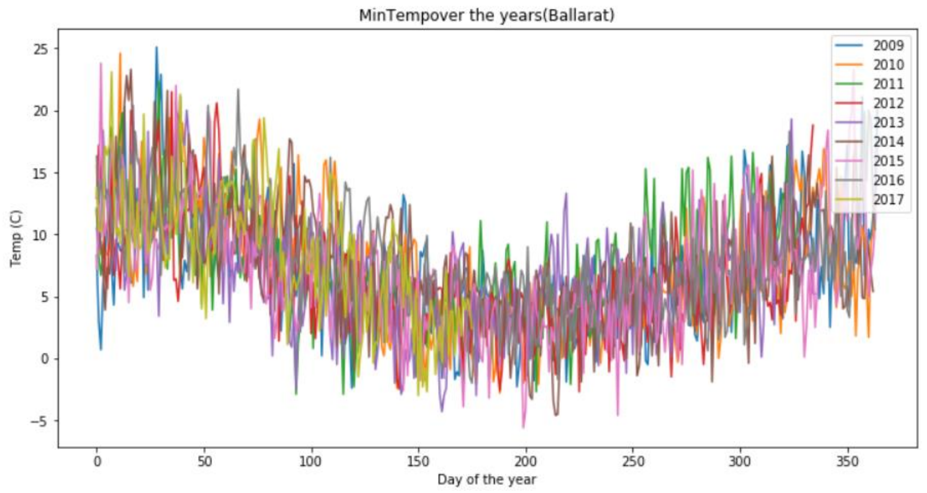
- The final pattern that was plotted and observed in relation to the temperature was when we plotted the change in temp between 9AM and 3PM for all entries in the table. When we plotted the information, we managed to get a plot vaguely matching the bell curve.

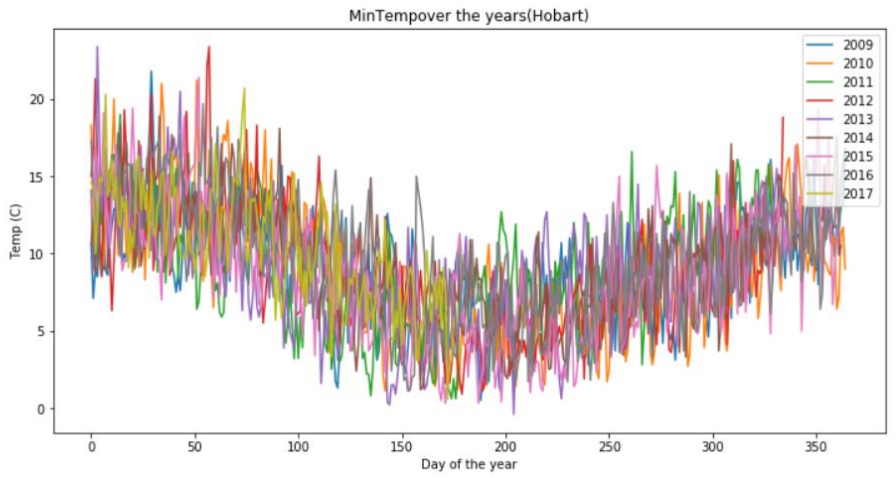


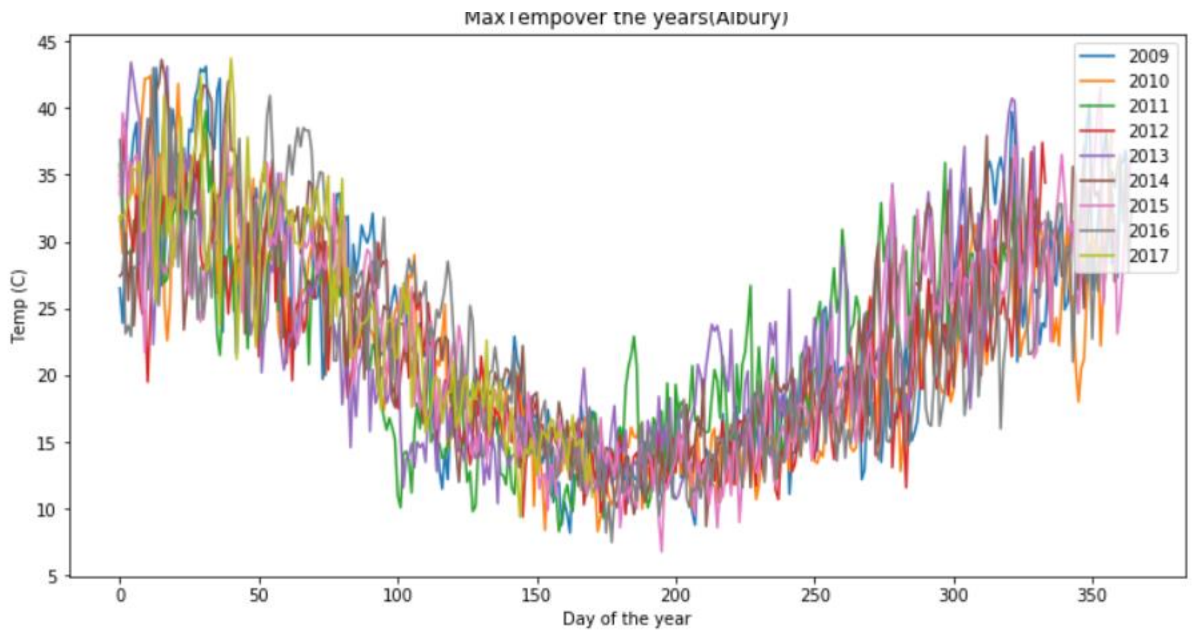
Temp Based Patterns

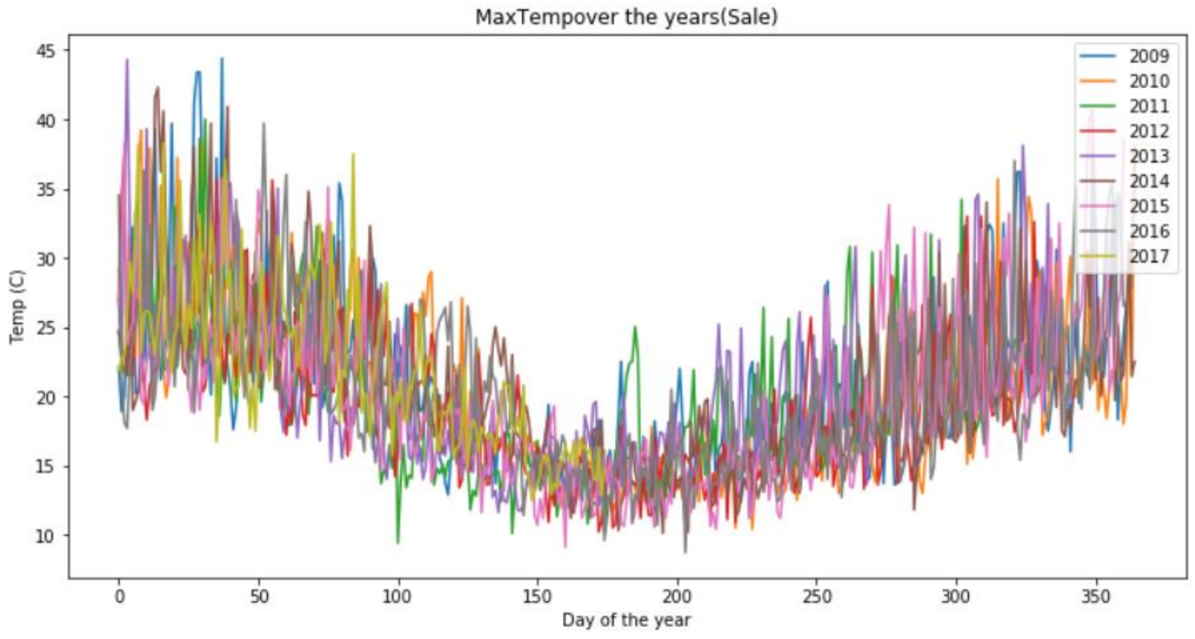
Max temp over the year:

- One of the patterns we observed was when we observed when looking at the max temp and min temp of locations, there was a fairly erratic to all the information as shown by the graphs. It was however noted that while it does fall into a curve that is common throughout the year, causing a sine wave.







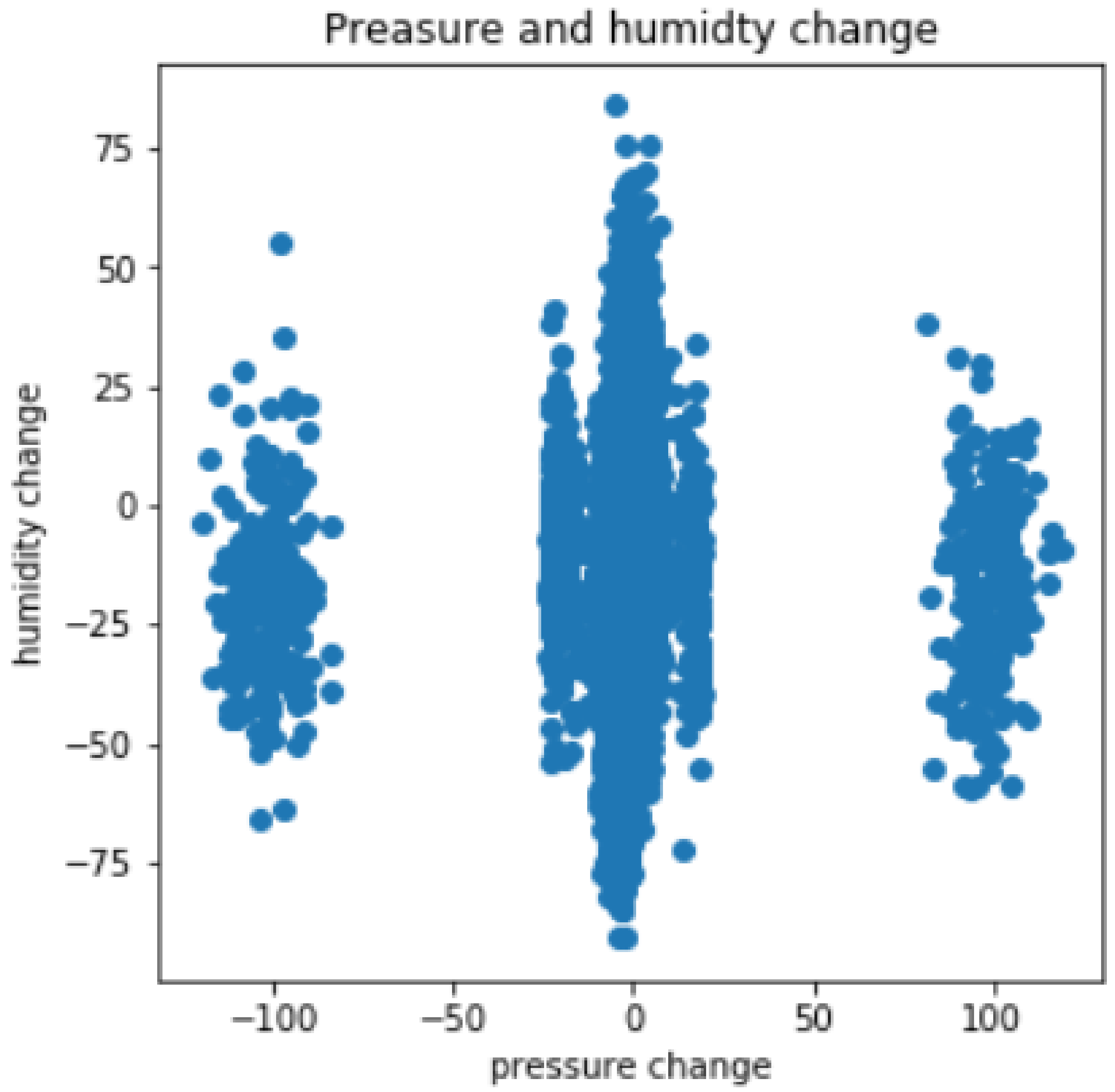


Temp Based Patterns

- It was noticed however that the max temp manages to shorten the curve's variation over the winter months. As shown by the graphs below show, there is a dip in the variation during the winter months with that trend being observed with all the locations.

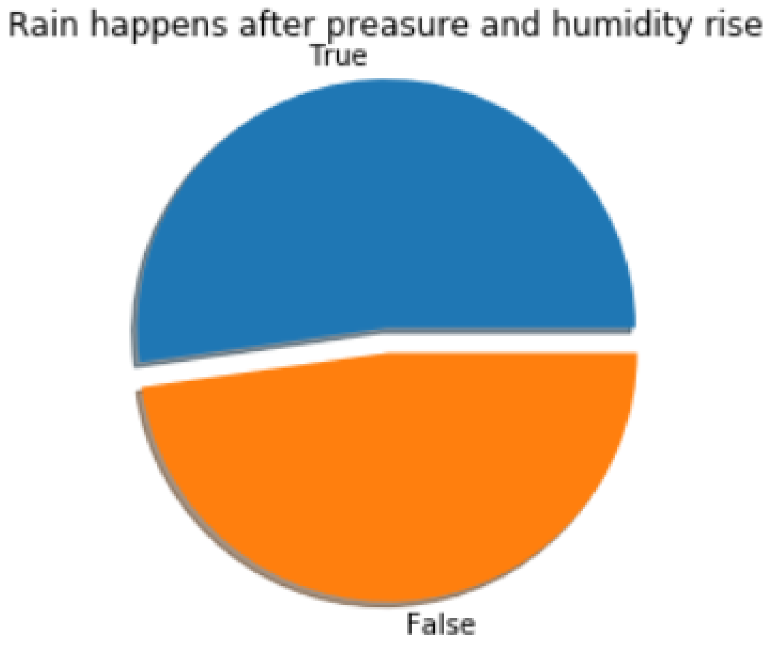
Humidity and pressure

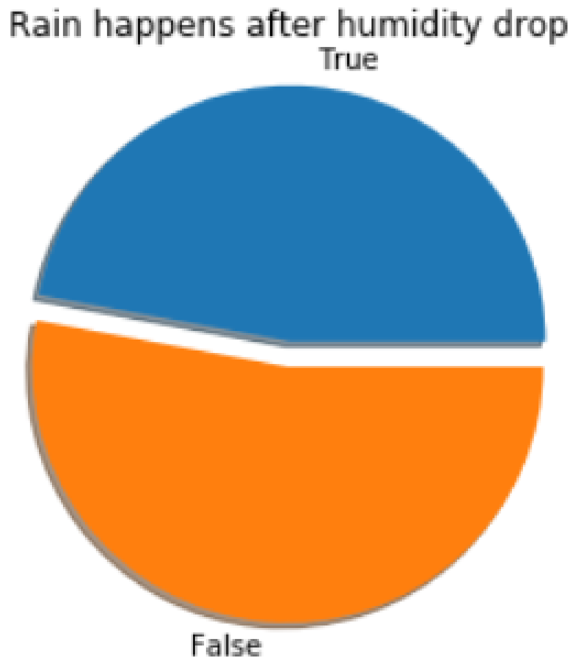
- Having looked at the trends that we observed in which we saw that we could predict rain >50% of the time, we decided to see if there was a trend in the rise in humidity that correlated with the rise in pressure, while we didn't manage to get usable information, we were able to notice that when plotted, the information made a graph that was interesting nonetheless and made the information somewhat predictable.

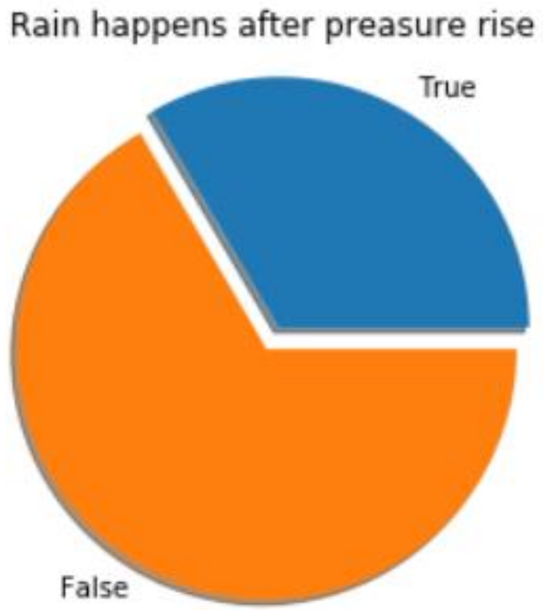


Humidity and pressure

- The other main observation that was observed while looking at the information was us following our assumptions.
- We had presumed that the humidity was related to rainfall, specifically an increase in humidity during the day meant that there was a change for rain, as we later observed that while this was true, pressure also correlated when there is a rise in both.
- While this was good news it was found that pressure only mattered in predicting with humidity as the pressure rise in relation to rain was almost negligible.







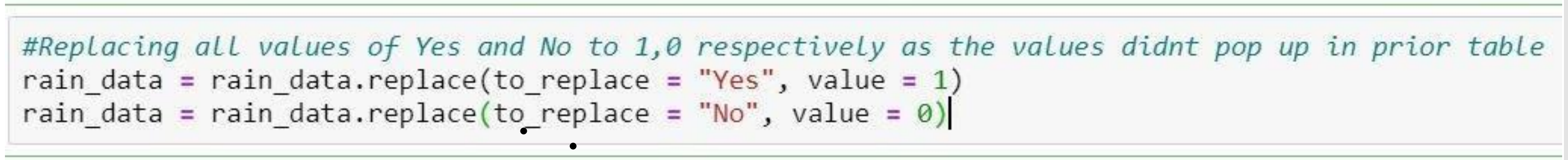
Data Testing

a. Find and identify the values in question

b. Setting up the DataFrame and preparing for Training and Testing

c. Training and Testing the dataset

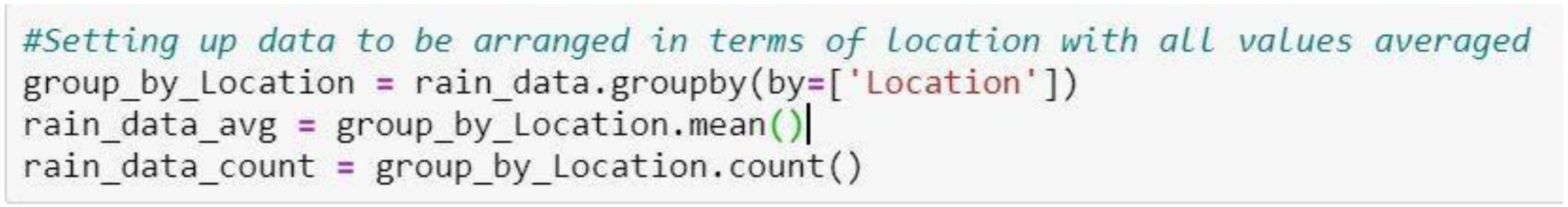
a. Find and identify the values in question



```
#Replacing all values of Yes and No to 1,0 respectively as the values didnt pop up in prior table
rain_data = rain_data.replace(to_replace = "Yes", value = 1)
rain_data = rain_data.replace(to_replace = "No", value = 0)
```

RainTomorrow was target variable

- Once all those values are 1 and 0 respectively the column of “RainTomorrow” can be used within the modelling data further on. Staging the next steps involved ensuring that each value could be grouped and averaged to create a more generalised set. This would contain locations to see averages in each area and to see if values counted are also consistent or slightly imbalanced.



```
#Setting up data to be arranged in terms of location with all values averaged
group_by_Location = rain_data.groupby(by=['Location'])
rain_data_avg = group_by_Location.mean()
rain_data_count = group_by_Location.count()
```

b. Setting up the Data Frame and preparing for Training and Testing

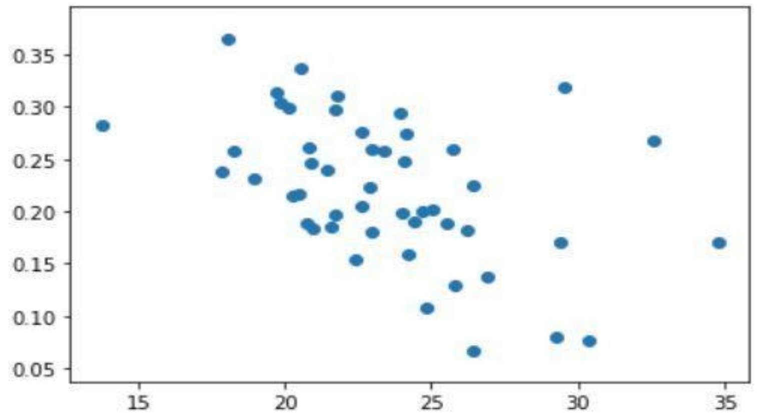
- As this model will be looking at locations and their relation to “RainTomorrow” the DataFrame will be set to the group by location done earlier.

```
#Checking the first 15 values to ensure right average set was grabbed
df = group_by_Location.mean()
df.head(15)
```

- The application of head was used to check that the first 15 values corresponded with the prior group_by_location.mean() table and no errors had occurred in between.

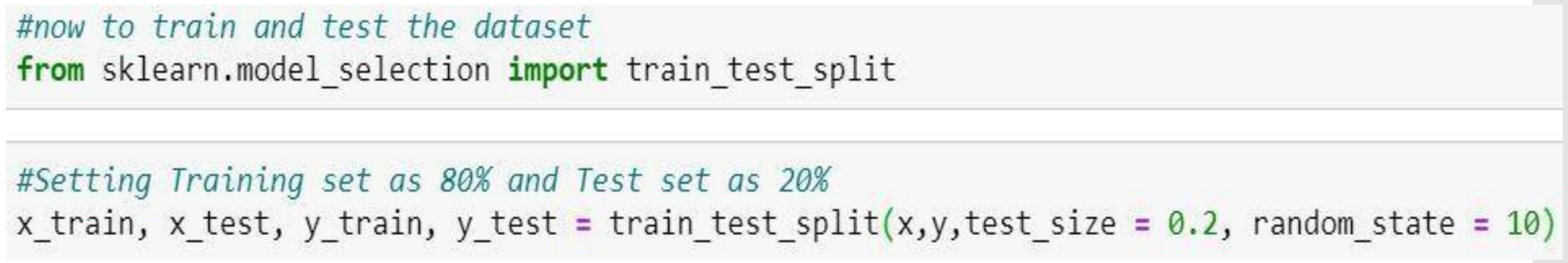
```
#Graph of Max Temp vs RainTomorrow
plt.scatter(df['MaxTemp'],df['RainTomorrow'])
```

```
<matplotlib.collections.PathCollection at 0x1d790b651c8>
```



b. Setting up the Data Frame and preparing for Training and Testing

- To prepare for Train and Test Split both x and y values must be defined. In this case Max and Min Temp will be “x” and RainTomorrow will be the “y” values. The machine learning library was to be imported and used next. Since it uses numpy extensively for high-performance linear algebra and array operations it’ll assist with modelling the data and checking its accuracy via Testing and Training.



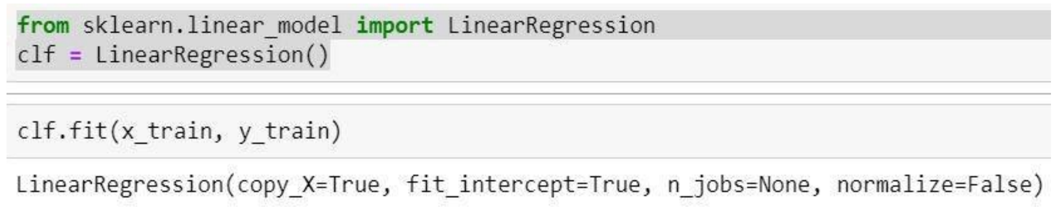
```
#now to train and test the dataset
from sklearn.model_selection import train_test_split
```

```
#Setting Training set as 80% and Test set as 20%
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 10)
```

- The random state was set at a specific number in order to keep pulling from that segment in the training set for consistency reasons.

c. Training and Testing the dataset

- First off, importing linear regression as Min and Max Temp want to be tested as a potential variable to the dependent variable RainTomorrow.

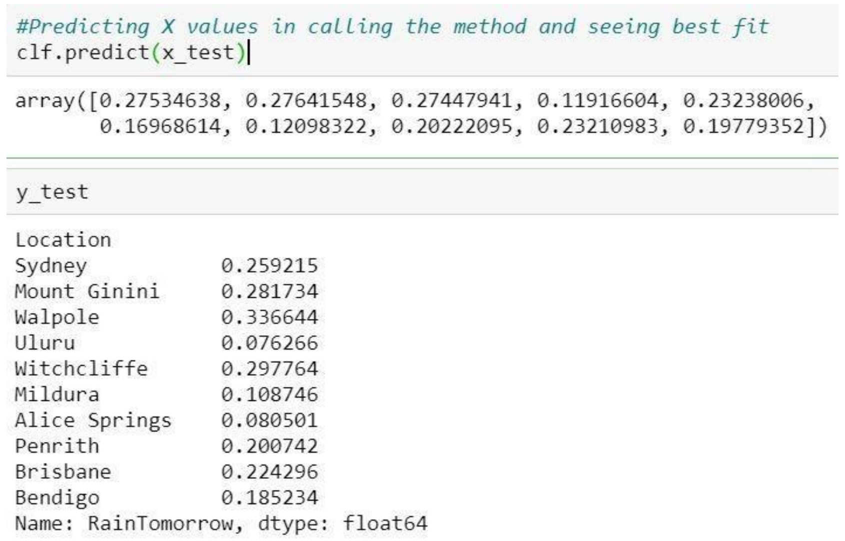


```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
```

```
clf.fit(x_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

- This takes all our values into the train set and checks to see if there is a potential fit and to determine values for testing.



```
#Predicting X values in calling the method and seeing best fit
clf.predict(x_test)
```

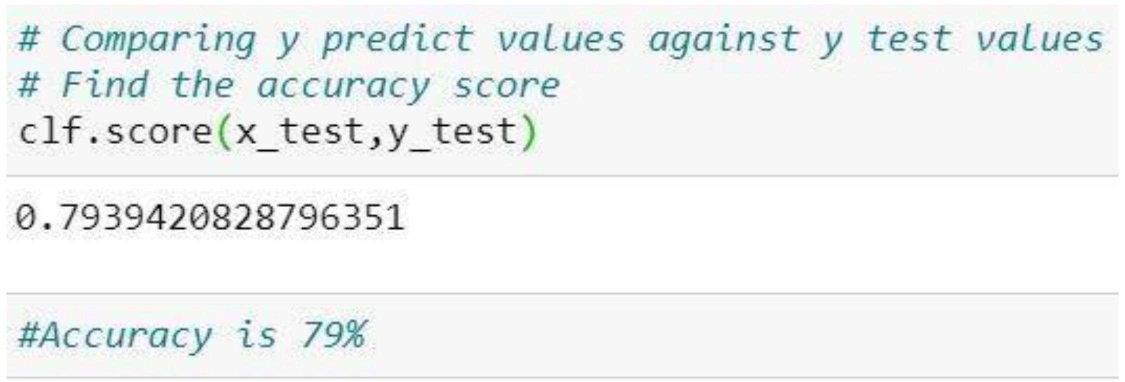
```
array([0.27534638, 0.27641548, 0.27447941, 0.11916604, 0.23238006,
       0.16968614, 0.12098322, 0.20222095, 0.23210983, 0.19779352])
```

```
y_test
```

```
Location
Sydney           0.259215
Mount Ginini     0.281734
Walpole          0.336644
Uluru            0.076266
Witchcliffe      0.297764
Mildura          0.108746
Alice Springs    0.080501
Penrith          0.200742
Brisbane         0.224296
Bendigo          0.185234
Name: RainTomorrow, dtype: float64
```

c. Training and Testing the dataset

- To summarize, the train_test_split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building the model. The testing subset is for using the model on unknown data to evaluate the performance of the model. This in turn allows us to see the accuracy and properties potentially of the dataset.



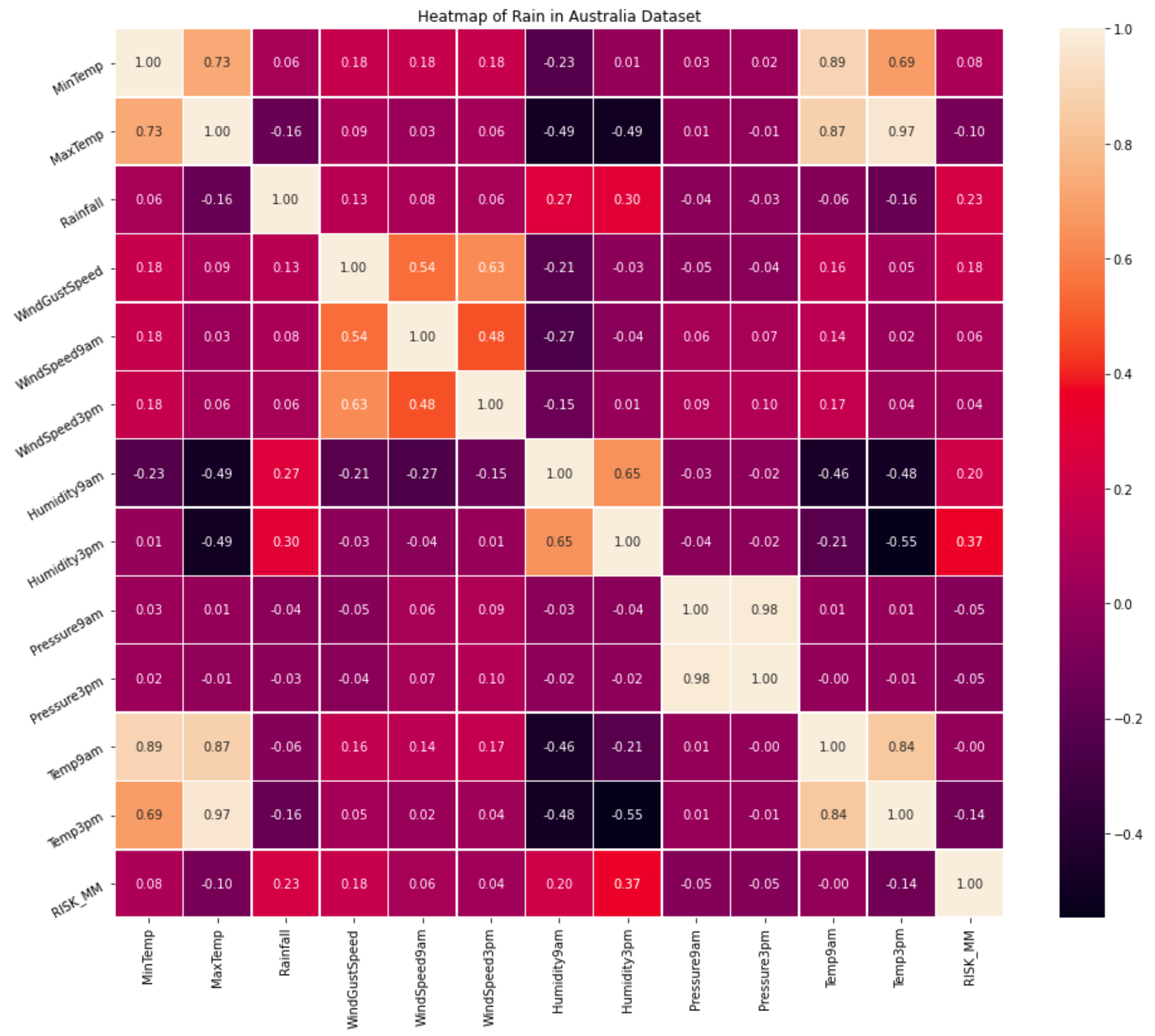
```
# Comparing y predict values against y test values
# Find the accuracy score
clf.score(x_test,y_test)
```

```
0.7939420828796351
```

```
#Accuracy is 79%
```

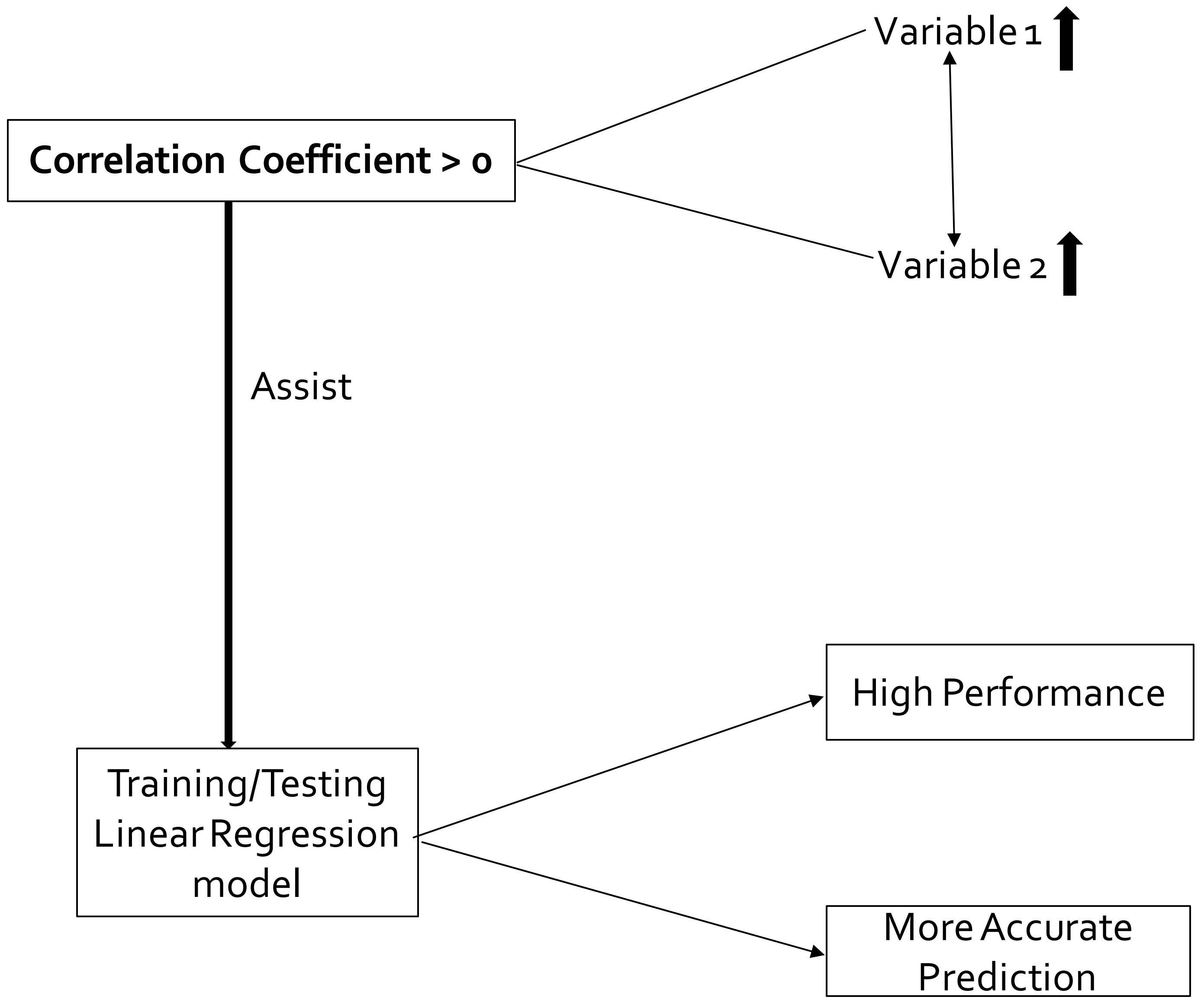
- As seen in the picture above the models accuracy was 79% however this was only given the data for Max and Min Temperatures and can be further used to test other variables.

Correlations and Assumptions



	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm	RISK_MM
MinTemp	1.00	0.73	0.06	0.18	0.18	0.18	-0.23	0.01	0.03	0.02	0.89	0.69	0.08
MaxTemp	0.73	1.00	-0.16	0.09	0.03	0.06	-0.49	-0.49	0.01	-0.01	0.87	0.97	-0.10
Rainfall	0.06	-0.16	1.00	0.13	0.08	0.06	0.27	0.30	-0.04	-0.03	-0.06	-0.16	0.23
WindGustSpeed	0.18	0.09	0.13	1.00	0.54	0.63	-0.21	-0.03	-0.05	-0.04	0.16	0.05	0.18
WindSpeed9am	0.18	0.03	0.08	0.54	1.00	0.48	-0.27	-0.04	0.06	0.07	0.14	0.02	0.06
WindSpeed3pm	0.18	0.06	0.06	0.63	0.48	1.00	-0.15	0.01	0.09	0.10	0.17	0.04	0.04
Humidity9am	-0.23	-0.49	0.27	-0.21	-0.27	-0.15	1.00	0.65	-0.03	-0.02	-0.46	-0.48	0.20
Humidity3pm	0.01	-0.49	0.30	-0.03	-0.04	0.01	0.65	1.00	-0.04	-0.02	-0.21	-0.55	0.37
Pressure9am	0.03	0.01	-0.04	-0.05	0.06	0.09	-0.03	-0.04	1.00	0.98	0.01	0.01	-0.05
Pressure3pm	0.02	-0.01	-0.03	-0.04	0.07	0.10	-0.02	-0.02	0.98	1.00	-0.00	-0.01	-0.05
Temp9am	0.89	0.87	-0.06	0.16	0.14	0.17	-0.46	-0.21	0.01	-0.00	1.00	0.84	-0.00
Temp3pm	0.69	0.97	-0.16	0.05	0.02	0.04	-0.48	-0.55	0.01	-0.01	0.84	1.00	-0.14
RISK_MM	0.08	-0.10	0.23	0.18	0.06	0.04	0.20	0.37	-0.05	-0.05	-0.00	-0.14	1.00

Correlations and Assumptions


Correlation Coefficient > 0
Variable 1
Variable 2
Assist
Training/Testing Linear Regression model
High Performance
More Accurate Prediction

Conclusion

Observations in data

- Testing and analysis of the “Rain in Australia” data-set visualized to be utilized in real life scenarios
- Patterns in temperatures in Australia were erratic and data on humidity and pressure, able to view trends that would help.

Conclusion

Obstacles:

- Struggle in implementing Naive Bayes .
- Dealing with dimension of matrix to fit Linear Regression .
- To calculate z-score first before drop columns with 20% of the values missing.
- Not getting the heatmap to work for a specific location after 3 attempts to recreate it from scratch.
- Having the data displayed/visualized incorrectly.

Conclusion

Solutions:

- Learned content and previous tasks
- Online sources and articles
- Other students and collogues

References

- Bureau of Meteorology n.d., *Note to accompany Daily Weather Observations*, Australian Government, retrieved 18 April 2020, <http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml>.

- Hunter, J, Dale, D, Firing, E & Droettboom, M n.d., *The Matplotlib development team*, retrieved 18 April 2020, <https://matplotlib.org/>.

- Young, J 2017, *Rain in Australia*, Kaggle, retrieved 18 April 2020, <https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>.