DATA ANALYSIS PROJECT

Unit Chair: Dr Musa Mammadov
Submission Date: 5:00PM Friday of Week 6

Table of Contents

Dataset Name: Rain in Australia

Group Name: Mon-13 (FANH) **On Campus/Cloud**: On Campus

STUDENT ID	STUDENT FULL NAME	Individual contribution*
218401269	ALEXANDER PAK YU LAI	5
218241616	HARRY WILLIAM LODGE	5
218459058	VIET NAM NGUYEN	5
218271795	JARROD KENG YEN YONG	5

* 5 - Contributed significantly, attended all meetings
4 - Partial contribution, attended all meetings
3 - Partial contribution, attended few meetings
2 - No contribution, attended few meetings
1 - No contribution, did not attended any meetings

Section 1: Introduction and Data Description

The Australian weather broadcast or the appropriate agencies, comprehends tomorrow's weather, specifically whether it will be raining or not, to deliver updated weather news to citizens or propose proper plans for social activities respectively. Therefore, the dataset of day-to-day weather was collected across various stations over a period of 10 years. The observations extracted from the Australian weather dataset aim to return prediction of tomorrow rain by producing the possibilities of it or two binary labels (Yes and No, 1 and 0 in that order).

The “Rain in Australia” dataset consists of 24 variables in total. Among these variables, there are two variable types, categorical variable and numerical variable.

For the categorical variable, five variables are normal, including “Date”, “Location”, “WindGustDir”, “WindDir9am” and “WindDir3pm” while the remaining two binary variables are “RainToday” and “RainTomorrow”, especially “RainTomorrow” is a target variable. Also, these categorical variables are defined with nominal type. In addition, the “WindDir9am” and “WindGustDir” variables experience the biggest percentages of missing value with around 7 percent (Young 2017). Furthermore, the variable having the highest cardinality, which means that a variable has the largest number of labels, is “Date” with 3436 labels and the second is “Location” with 49.

For the numerical variable, there are seventeen variables, which refer to continuous type, namely “MinTemp”, “MaxTemp”, “Rainfall”, “Evaporation”, “Sunshine”, “WindGustSpeed”, “WindSpeed9am”, “WindSpeed3pm”, “Humidity9am”, “Humidity3pm”, “Pressure9am”, “Pressure3pm”, “Cloud9am”, “Cloud3pm”, “Temp9am”, “Temp3pm” and “RISK_MM”. According the figures in the table indicated the dataset provided by Young (2017), the variables witnessing the most missing values are “Evaporation” and “Sunshine” with roughly 43 and 48 percent respectively, followed by “Cloud9am” and “Cloud3pm” with nearly 38 and 40 percent in that order. Also, the “RISK_MM” variable is in consideration for dropping out of the dataset. According to Young (2017), this variable should be eliminated if it aims to train a classification model instead of regression one or the “RainTomorrow” variable is considered as a target because including this variable which indicates the further information of rain impacts negatively to the predicted values of the trained model or a lower accuracy score.

The observation about the “Rain in Australia” is that the method of training model to return a high accuracy score can be a regression algorithm instead of classification, but the null accuracy, where the accuracy is gained by preferring the most frequent value, should be compared with it to guarantee that the applied model results in the higher accuracy score.

Another observation is that many pairs of variables would have strong or high correlation in the positive trend since they are intuitively correlated in terms of related fields of rain prediction.

Before the dataset is analyzed thoroughly and its comprehensive patterns are extracted, the dataset ensures to be evaluated carefully and cleaned to get rid of missing values or incorrect data by finding the frequency of null values for each variable, calculating z scores or exploring the inner problem of categorical and numerical variables. After finishing the process of data cleansing, the cleaned dataset should be tested, followed by discovering patterns which define relationships between variables due to their correlation coefficient. Eventually, the comprehensive analysis is finalized to sum up as well as conclude findings. Throughout the exploratory data analysis, the appropriate visualizations of data evaluation or results are illustrated.

Section 2: Exploratory Data Analysis and Results

1. Data Cleansing

To achieve a machine learning model with better performance and higher test accuracy score, the “Rain in Australia” dataset should be clean, which removes all missing values, known as null or nan, fixing incorrect data in terms of grammar or format and finding outliers. Therefore, the process of data cleansing is split up into six steps.

a. Find and correct errors in categorical variables

The first stage of cleaning the dataset is to engineer errors in categorical variables, known as nominal columns. There are eight of them in total, “Date”, “Location”, “WindGustDir”, “WindDir9am”, “WinDir3pm”, “RainToday” and “RainTomorrow”, but the important variable scrutinized thoroughly during training and testing process is “RainTomorrow” since it is used as target to produce accuracy scores. Starting finding invalid date format in “Date” is the first step in eliminating errors in the categorical variables and the following function supported by the “datetime” library is applied on this variable to assess every single element.

```
# 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
```

As a result, the “Date” column witnesses no incorrect value in regard to its standard format, e.g. “2008-12-01”.

Focusing on the “Location” variable is the next categorical variable. To evaluate this column, all unique locations should be explored, and one small flaw on the dataset might be fixed. The small flaw is missing proper whitespace in some location names which contains two words. Therefore, the solution is simply that the whitespaces are placed appropriately in two-word locations. The lines of code below illustrates this approach.

```
# 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)
```

However, this flaw in the “Location” column has insignificant impact on the dataset because the uniqueness of locations are still preserved.

The next three categorical variables whose names are “WindGustDir”, “WindDir9am” and “WinDir3pm” are defined with sixteen compass directions including four caridnal directions, four intercardinal directions and eight secondary intercardinal directions. The fortunate results after processing them through some lines of code shows no incorrect directions in terms of format and the following figure indicates the typical checking.

```
# 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))
```

The two remaining variables, “RainToday” and “RainTomorrow”, are represented by two unique elements, “Yes” and “No” which are two binary values and can be considered as 1 and 0 respectively. The positive results gained after these variables experience the validation process are no invalid data.

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

After the nominal columns finish being corrected, the total number of nan or null values of each column should be found, which becomes a base for the following steps. The below figure illustrates two functions which count the missing values and find their corresponding indices in columns.

```
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}")
```

The following table gives information on the total number of missing values for each variable.

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

Exploiting the result of the total number of missing values in the previous assists in calculating the variables having more than 10 percent nan or null data. Therefore, the percentages of missing values in these variables are easily computed by applying the lines of code in the following figure.

```
# 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
```

Consequently, there are four variables holding the percentage of missing values exceeding 10, “Evaporation”, “Sunshine”, “Cloud9am” and “Cloud3pm”, corresponding 44, 48, 48 and 40 percent, which encourages them to be dropped out of dataset by one function of pandas library, “pandas.DataFrame.drop”.

d. Drop rows with more than 20% missing values

Removing the rows which percentages of missing values are greater than 20 has the same strategy with the previous step. However, one more computation which leads to percentage calculation is counting the total number of missing values for each row by adding the following extra lines of code to figure it out.

```
# 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
```

The same process of computing the percentage of missing values with the prior step. However, the opposite results are achieved because of no rows with more than 20 percent nan or null data.

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

Only numerical variables, known as continuous columns, are able to be applied to the z-score formula to find outliers, and these outliers should be substituted by the mean of appropriate columns. The following figure indicates the computation of z-scores for the entire dataset.

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

Addressing all outliers assists the process of replacing them with possible proper values which are the average of the corresponding columns.Removing all outliers is the preparation for filling missing values existing in the entire dataset. The function in the below figure gives information on fixing outliers.

```
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

To engineer all nan or null data with the possibly highest accuracy, different data types should be applied to replace them by the most proper algorithm. Therefore, filling missing values in numerical variables utilizes the Linear Regression algorithm, “RainToday” uses Naive Bayes”, and “WindGustDir”, “WindDir9am” and “WindDir3pm” simply takes random algorithm provided by “numpy” library.

The reasons why choosing Linear Regression is for numerical variables are that they are continuous and each of these variable arms themselves with independence, which encourages to figure out their relationships by exploiting the linear equation which provides one coefficient and one intercept. Based on the coefficient and intercept, the final purpose is to form a linear equation presenting the relationship between two variables, which are a column containing missing values and the selected stable one which is “RISK_MM”. However, to fit the model successfully, all null or nan should be substituted by 0. The following figure illustrates the process of training the model and producing the linear equation.

```
# # Fit the model
lr = linear_model.LinearRegression()
lr.fit(x,y)

# Gain the model parameters
coef = lr.coef_
intercept = lr.intercept_
```

Filling the missing values in “WindGustDir”, “WindDir9am” and “WindDir3pm” is simplified by the random method of choosing a random direction in sixteen compass directions. The two lines of code in the following figure.

```
random_number = random.randint(0, len(unique_direction)-1)
random_direction = unique_direction[random_number]
```

The “RainToday” variable consists of only two unique values, “Yes” and “No” experiences through Gaussian Classifier which is based on Naive Bayes Theorem to replace all missing values with one of two choices. The first reason why the last variable should be trained by this algorithm is that it is quick, highly accurate and reliable regardless of the size of the dataset, but not simple. Another one is that it emphasizes the independence of each variable and it has high performance on categorically-typed variables. Therefore, after all selected data are encoded to numbers corresponding to labels, they are fitted into Gaussian Classifier to train the model which supports finding the most appropriate values for the missing values. The following lines of code shows the implementation of this classifier technique.

```
#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(features,encoded_column)
```

2. Main Structure and Key Variables

a. Information used in analysis

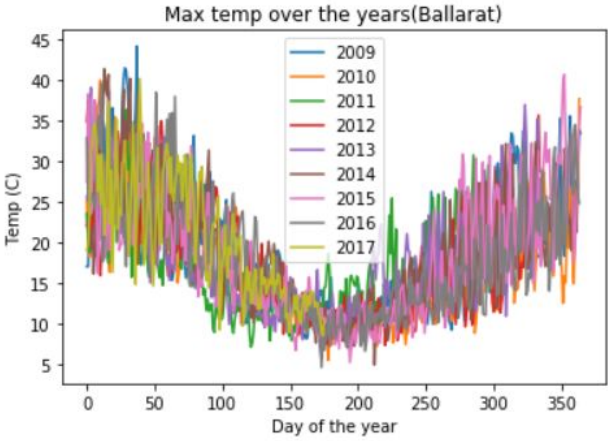
```
AusWeather.columns #looking into the data that has been divided into groups
```

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

Above is the list of 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.

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 below.



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

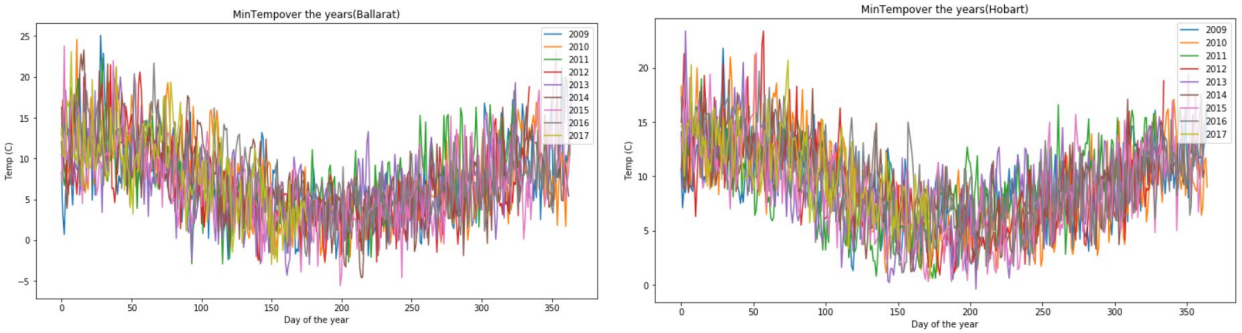
The rest of the values, while useful, weren't used too much in our calculations and meant that we didn't have a way to use them during the analysis, other than the generic analysis that we had used during testing that we had used so that we could make sure all our calculations for both data cleaning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

3. Data Patterns

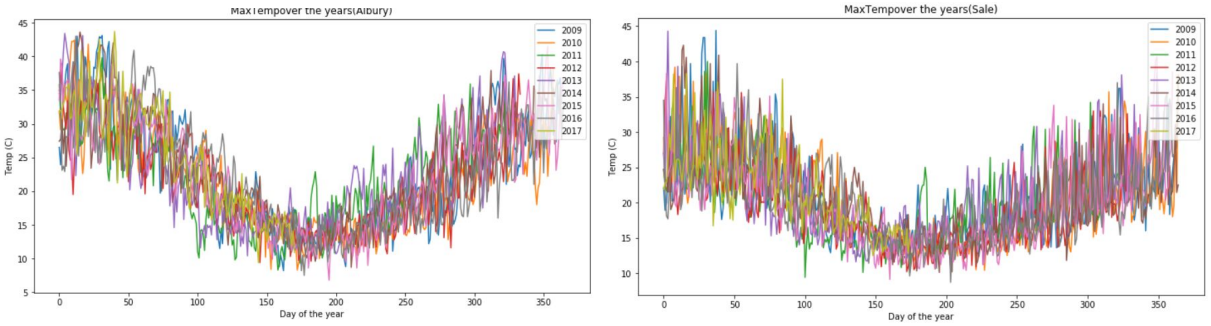
Temp based patterns:

Max temp over the year:

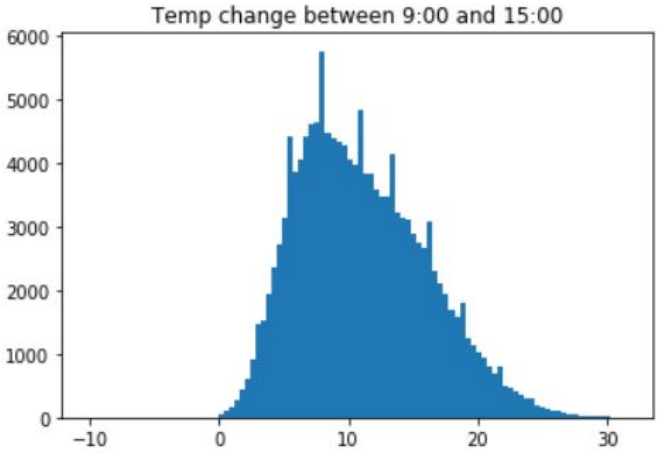
One of the patterns we observed was when we observed when looking at the max temp and min temp of locations, there was a fairly erratic to all the information as shown by the graphs. It was however noted that while it does fall into a curve that is common throughout the year, causing a sine wave.



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.

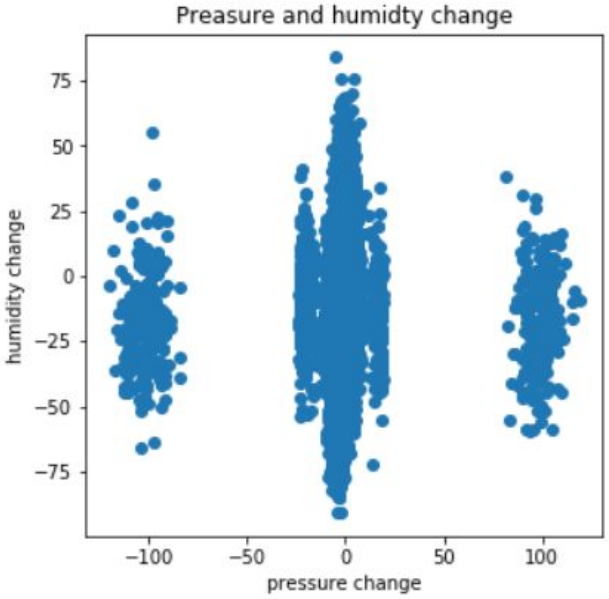


The final pattern that was plotted and observed in relation to the temperature was when we plotted the change in temp between 9AM and 3PM for all entries in the table. When we plotted the information, we managed to get a plot vaguely matching the bell curve.

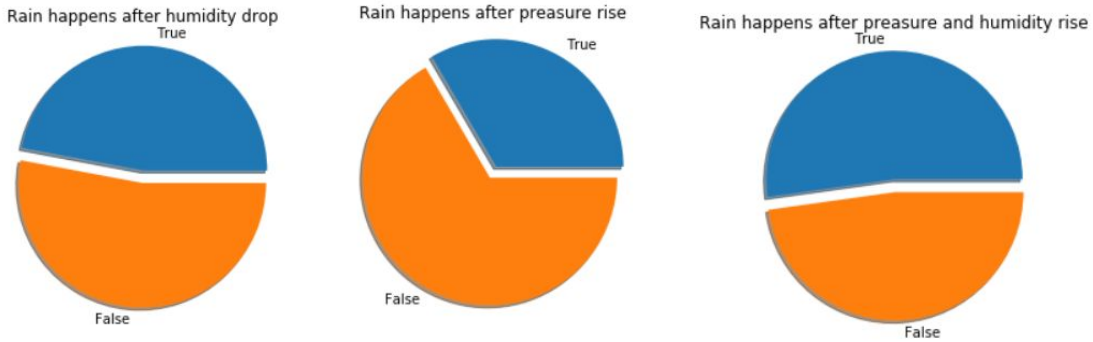


Humidity and pressure:

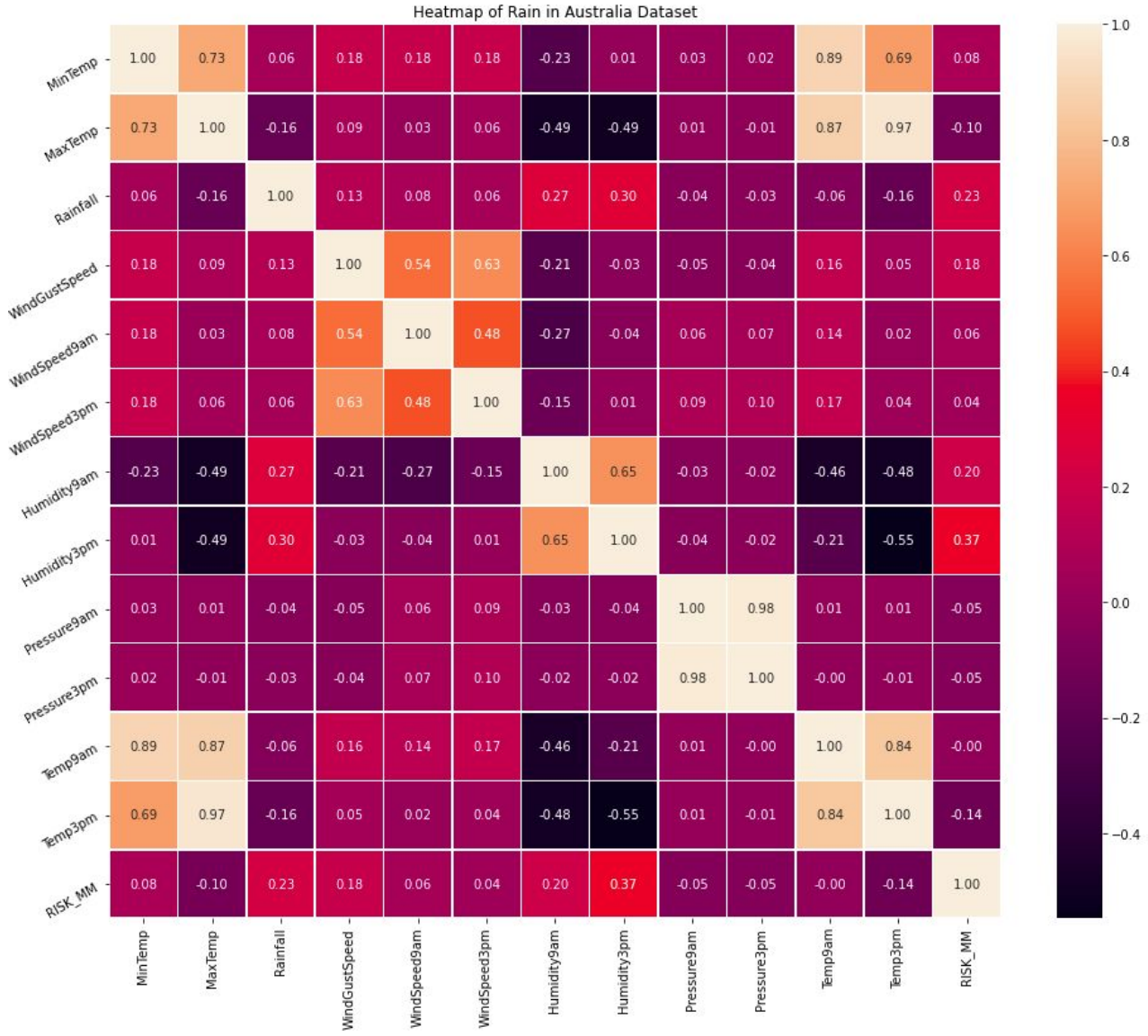
Having looked at the trends that we observed in which we saw that we could predict rain >50% of the time, we decided to see if there was a trend in the rise in humidity that correlated with the rise in pressure, while we didn't manage to get usable information, we were able to notice that when plotted,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



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.



4. Correlations and Assumptions



The above heat map illustrates the correlation coefficient between variables. The figure for correlation coefficient, 0.73, indicates high positive correlation between “MinTemp” and “MaxTemp”. The correlation coefficient of “MinTemp” and “Temp9am” variables are 0.89, which demonstrates the strong positive correlation. The “MinTemp” and “Temp3pm” variables have a quite high positive correlation with their coefficient, 0.69. Then, the number of correlation coefficient, 0.87, reveal the strong positive correlation between “MaxTemp” and “Temp3am” variables. Then, the correlations between “MaxTemp” and “Temp3pm”, “Pressure9am” and “Pressure3pm”, “Temp9am” and “Temp3pm” are positively strong with coefficients, 0.97, 0.98 and 0.84 respectively.

Therefore, the pairs of variable who have strong positively such as “MaxTemp” and “Temp3pm”, “Pressure9am” and “Pressure3pm” presents proportional increases which means if one variable rises, another go up in the same direction with the variable. They can assist the prediction of tomorrow's rain to be more accurate by filtering the best variable to be training or testing set for training models, especially Linear Regression, which improves their performance and accuracy score.

5. Data Testing

Once all the data has been cleansed of its inconsistent values whether it had been replaced or changed it was now time to train/test the dataset in order to see if the dataset and model is good enough to be evaluated.

a. Find and identify the values in question

As predicting the rainfall tomorrow was being questioned, establishing the values in the dataset was important to starting testing.

```
#Importing cleaned data
rain_data = pd.read_csv('data/cleaned_data.csv')
rain_data = rain_data.drop(columns=["Unnamed: 0"])
columns = rain_data.columns
print(columns)
rain_data.describe()
```

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

Following identifying all the columns and values obtained, the next situation was changing all values of “Yes” and “No” into binary in order to use “RainTomorrow” as the basis of the testing in comparison to other values.

```
#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)
```

Once all those values are 1 and 0 respectively the column of “RainTomorrow” can be used within the modeling 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 DataFrame and preparing for Training and Testing

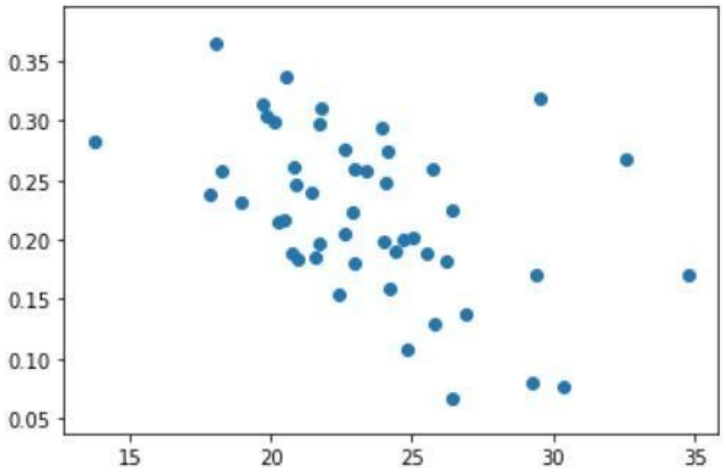
The next step was crucial to do correctly as the dataset must be set up to ensure the value as taken for training and testing to see the model accuracy. As this model will be looking at locations and their relation to “RainTomorrow” the DataFrame will be set to the group by location done earlier.

```
#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. This part of testing used the values of Minimum and Maximum Temperatures and compared it to RainTomorrow.

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

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



Each point represents the chance of RainTomorrow in a particular location based on the average Max temp in that area. 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 modeling 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

With the training data set as 80% and testing 20% it is now time to train and test.

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. The entire point of this process is to take values from the dataset to obtain and verify the Datasets overall accuracy and given the independent variables find a model that best fits the dependent variable

```
#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
```

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
```

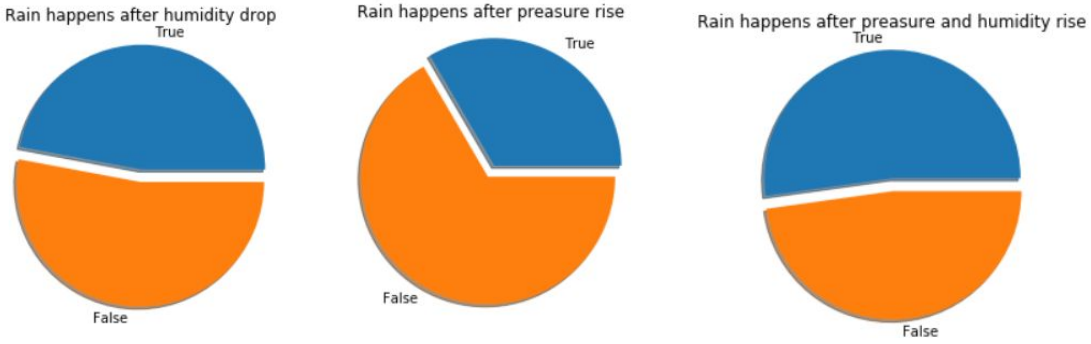
```
#Accuracy is 79%
```

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.

6. Visualization

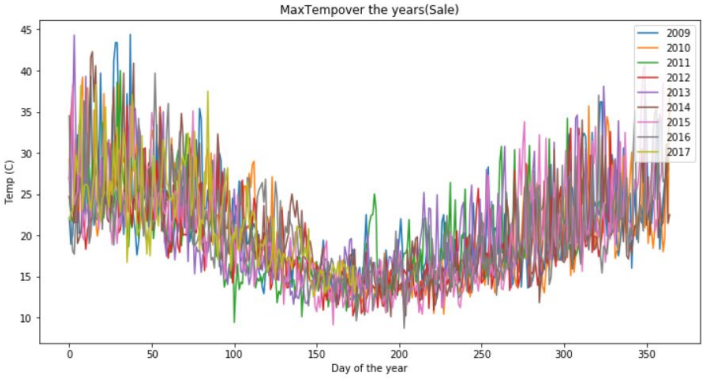
Pie chart:

The main visualization used was the pie charts that we used to check our assumptions. We picked these as they show the relative size of both sections. As we can see below, it's clear to see that in the intal two charts, there is a clear trend showing that "no" is greater where in the third, it's clear that "yes" is greater.



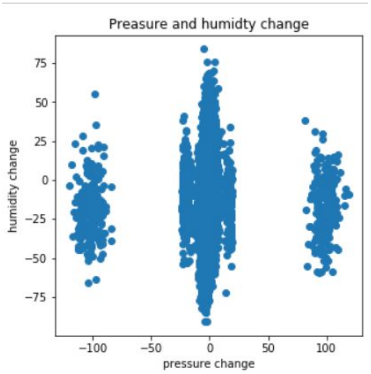
Plot graphs:

These are a graph that we used to show the linking of variables in the same group. We used this when analyzing the temp over years graphs. Below we can see an example of this where each year's daily temps are linked up so that we can see the differences for each year.

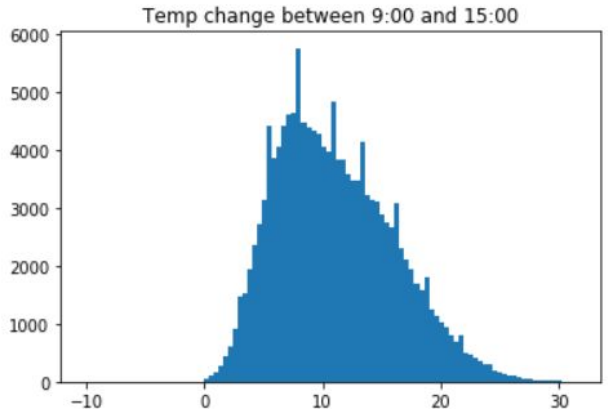


Scatter plot:

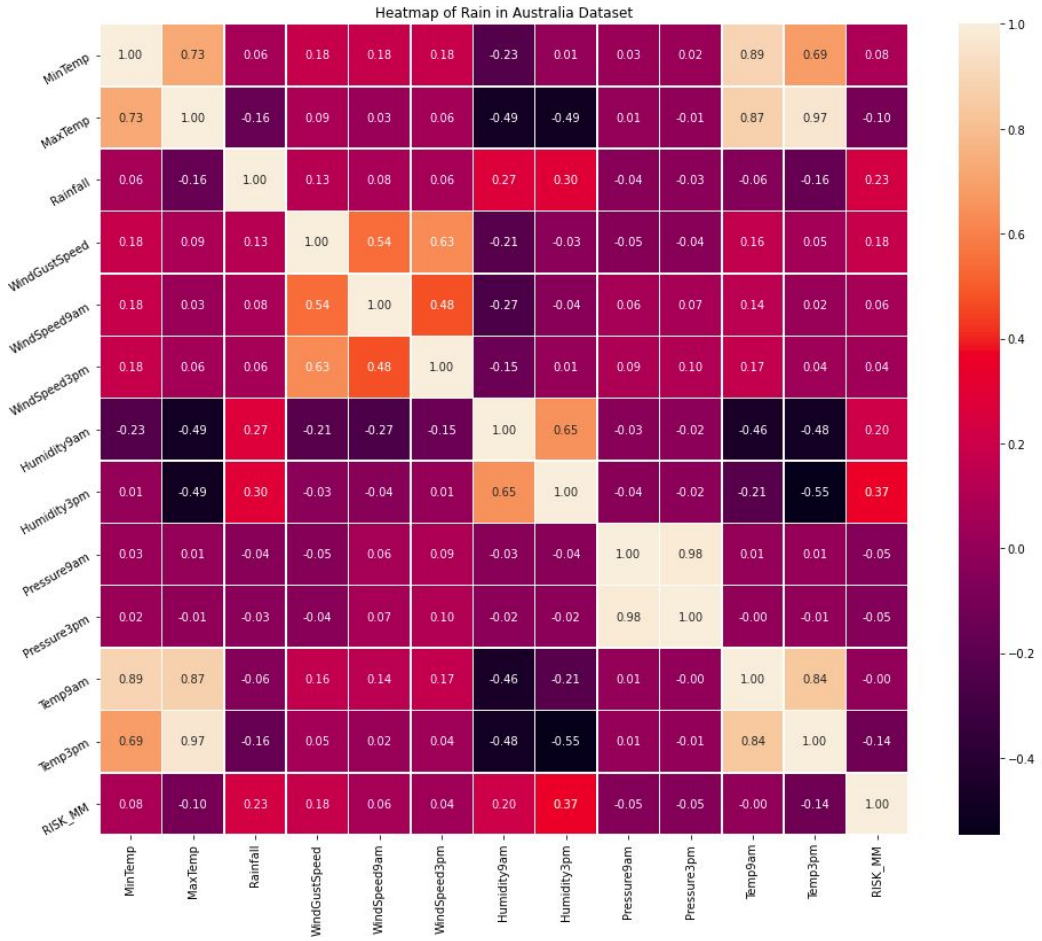
This graph was used when we wanted to make it so we could compare 2 values that are correlating. Our application of this was to show groupings with the change in temp and change in humidity that we had earlier looked at when we compared the variables in relation to rain.



Histogram:
The use of histograms in this case was to show the quantity of the temp rises in the morning to afternoon in an attempt to intitaly see if there was a negative, it was however observed that the information shaped into the bell curve that we can see below.



Heatmap:
The heatmap helps show the correlation between the different pairs of variables.



Section 3: Conclusion

Observations in data:

The observations that were formed from the testing and analysis of the “Rain in Australia” data-set was a series of useful prepared information visualised to be used in real life scenarios like environmental science or prediction of weather forecast. The visualization of the data shows patterns in temperatures throughout Australia were erratic in nature but fit in line with the year's seasonal changes and with the data on humidity and pressure, we were able to view trends that would help predict rain more than 50% of the time.

Obstacles:
Some factors that proved to be obstacles were:

- The struggle in implementing Naive Bayes to fill the missing values in categorical variables like encoding and decoding values.
- Dealing with dimension of matrix to fit Linear Regression to fill the missing values in numerical variables.
- Deciding 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 incorrectly.

Solutions and resolve:
Some workarounds these obstacles consisted of:

- Going back through learned content like practical classes was critical to dealing with many of the problems.
- Reading and using online sources and articles to help understand the steps and process to get the desired result.
- Using other students and colleagues to help find a method or process to figure out the solution.

Section 4: 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>.