Kingdom of Saudi Arabia
Ministry of Education
Al-Imam Mohammed Ibn Saud Islamic University
College of Computer & Information Science
Department of Computer Science
Course: Machine Learning (CS364)

# RAIN IN AUSTRALIA

## CLASSIFICATION IF RAIN TOMORROW OR NOT
## MACHINE LEARNING
## CS364 -372

SUBMISSION DATE: 11 FEBRUARY 2023

# TABLE OF CONTENTS

# TABLE OF CONTENTS

# T A B L E  O F  F I G U R E S

# INTRODUCTION

**1.Introduction**

In this project, we aim to design and implement supervised machine learning models to classify rain tomorrow or not **IN AUSTRALIA .**

In this section, we are going to discuss the problem we are going to address followed by the project timeline and our qualifications as a team.

**1.1 Problem Description**

This project is about weather in Australia, particulary regarding prediction if rain tomorrow or not, by training classification model on the target variable.

The target is RainTomorrow. which means did it rain the next day, Yes or No?

This problem is binary classification because the target "RainTomorrow" contains two classes yes or No. The classification operation based on a relationship between a known class assignment and characteristics of the entity to be classified. supervised learning predictive method, which means the data set is labeled.

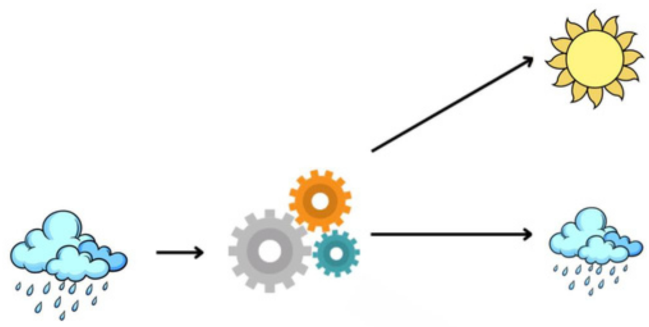


Figure 1: Classifier Model

# PROJECT TIMELINE

We proposed the following timeline to complete our project phases :

**weak 1**

### DATA ORGANIZATION

prepare and analysis the data set.

### SPLIT DATA FOR TRAINING AND TESTING

It defines a set of training data as input for learning algorithms, to have the ability to generalize well to new data and to determine the size of a test sample.

**weak 2**

### APPLIED SVM CLASSIFICATION

(Support Vector Machine)

**weak 3**

### WRITING OUR PROJECT

writing report and prepare our project for final submission

**weak 4**

# D A T A S E T

**2. Dataset**
In this section, we are going to present the dataset we selected and its attributes.

**2.1 Dataset Acquisition**
These data set we found on Kaggle
https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
It's about observation the daily weather "Instances" of ten years from many locations across Australia.

Number of rows : 145461
Number of attributes : 22
Number of Independent Columns: 22
Number of Dependent Column: 1

**2.2 Dataset Attributes**
Column Description "Attribute":

- Date : The date of observation
- Location : The common name of the location of the weather station
- MinTemp : The minimum temperature in degrees celsius
- MaxTemp : The maximum temperature in degrees celsius
- Rainfall : The amount of rainfall recorded for the day in mm
- Evaporation : The so-called Class A pan evaporation (mm) in the 24 hours to 9am
- Sunshine : The number of hours of bright sunshine in the day.
- WindGustDir : The direction of the strongest wind gust in the 24 hours to midnight
- WindGustSpeed : The speed (km/h) of the strongest wind gust in the 24 hours to midnight
- WindDir9am : Direction of the wind at 9am
- WindDir3pm : Direction of the wind at 3pm
- WindSpeed9am : Wind speed (km/hr) averaged over 10 minutes prior to 9am
- WindSpeed3pm : Wind speed (km/hr) averaged over 10 minutes prior to 3pm
- Humidity9am : Humidity (percent) at 9am
- Humidity3pm : Humidity (percent) at 3pm
- Pressure9am : Atmospheric pressure (hpa) reduced to mean sea level at 9am
- Pressure3pm : Atmospheric pressure (hpa) reduced to mean sea level at 3pm
- Cloud9am : Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
- Cloud3pm : Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values
- Temp9am : Temperature (degrees C) at 9am
- Temp3pm : Temperature (degrees C) at 3pm
- RainToday : Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
- RainTomorrow : The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".

# DATASET

| W | V | U | T | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | E | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RainTomo | RainToday | Temp3pm | Temp9am | Cloud3pm | Cloud9am | Pressure3 | Pressure9 | Humidity3 | Humidity9 | WindSpee | WindSpee | WindDir3p | WindDir9a | WindGust | WindGustI | Sunshine | Evaporatio | Rainfall | MaxTemp |
| No | No | 21.8 | 16.9 | NA | 8 | 1007.1 | 1007.7 | 22 | 71 | 24 | 20 | WNW | W | 44 | W | NA | NA | 0.6 | 22.9 |
| No | No | 24.3 | 17.2 | NA | NA | 1007.8 | 1010.6 | 25 | 44 | 22 | 4 | WSW | NNW | 44 | WNW | NA | NA | 0 | 25.1 |
| No | No | 23.2 | 21 | 2 | NA | 1008.7 | 1007.6 | 30 | 38 | 26 | 19 | WSW | W | 46 | WSW | NA | NA | 0 | 25.7 |
| No | No | 26.5 | 18.1 | NA | NA | 1012.8 | 1017.6 | 16 | 45 | 9 | 11 | E | SE | 24 | NE | NA | NA | 0 | 28 |
| No | No | 29.7 | 17.8 | 8 | 7 | 1006 | 1010.8 | 33 | 82 | 20 | 7 | NW | ENE | 41 | W | NA | NA | 1 | 32.3 |
| No | No | 28.9 | 20.6 | NA | NA | 1005.4 | 1009.2 | 23 | 55 | 24 | 19 | W | W | 56 | WNW | NA | NA | 0.2 | 29.7 |
| No | No | 24.6 | 18.1 | NA | 1 | 1008.2 | 1009.6 | 19 | 49 | 24 | 20 | W | SW | 50 | W | NA | NA | 0 | 25 |
| No | No | 25.5 | 16.3 | NA | NA | 1010.1 | 1013.4 | 19 | 48 | 17 | 6 | W | SSE | 35 | W | NA | NA | 0 | 26.7 |
| Yes | No | 30.2 | 18.3 | NA | NA | 1003.6 | 1008.9 | 9 | 42 | 28 | 7 | NW | SE | 80 | NNW | NA | NA | 0 | 31.9 |
| No | Yes | 28.2 | 20.1 | NA | NA | 1005.7 | 1007 | 27 | 58 | 11 | 15 | SSE | S | 28 | W | NA | NA | 1.4 | 30.1 |
| Yes | No | 28.8 | 20.4 | NA | NA | 1008.7 | 1011.8 | 22 | 48 | 6 | 17 | ESE | SSE | 30 | N | NA | NA | 0 | 30.4 |
| Yes | Yes | 17 | 15.9 | 8 | 8 | 1004.2 | 1010.5 | 91 | 89 | 13 | 15 | ENE | NE | 31 | NNE | NA | NA | 2.2 | 21.7 |
| Yes | Yes | 15.8 | 17.4 | 8 | 8 | 993 | 994.3 | 93 | 76 | 28 | 28 | NNW | NNW | 61 | W | NA | NA | 15.6 | 18.6 |
| No | Yes | 19.8 | 15.8 | 7 | NA | 1001.8 | 1001.2 | 43 | 65 | 20 | 24 | SSW | W | 44 | SW | NA | NA | 3.6 | 21 |
| NA | No | 23.5 | 15.9 | NA | NA | 1008.7 | 1009.7 | 32 | 57 | 30 | 4 | WNW | S | NA | NA | NA | NA | 0 | 24.6 |
| No | NA | 26.2 | 17.3 | NA | 0 | 1010.3 | 1013.4 | 28 | 50 | 22 | NA | WNW | NA | 50 | WNW | NA | NA | NA | 27.7 |
| Yes | No | 18.1 | 17.2 | 1 | 8 | 1010.4 | 1012.2 | 82 | 69 | 9 | 11 | E | SSW | 22 | ENE | NA | NA | 0 | 20.9 |
| Yes | Yes | 21.5 | 18 | 1 | 8 | 1002.2 | 1005.8 | 65 | 80 | 20 | 6 | WNW | N | 63 | W | NA | NA | 16.8 | 22.9 |
| No | Yes | 21 | 15.5 | 2 | NA | 1009.7 | 1009.4 | 32 | 47 | 17 | 24 | SW | WSW | 43 | SSE | NA | NA | 10.6 | 22.5 |
| No | No | 23.2 | 15.8 | NA | NA | 1017.1 | 1019.2 | 26 | 45 | 6 | 17 | NNW | SE | 26 | SSE | NA | NA | 0 | 25.6 |
| No | No | 27.3 | 19.1 | NA | NA | 1014.8 | 1019.3 | 28 | 56 | 9 | 9 | SE | SE | 24 | S | NA | NA | 0 | 29.3 |
| No | No | 31.6 | 24.5 | 1 | NA | 1008.1 | 1013.6 | 28 | 38 | 22 | 17 | N | NE | 43 | NE | NA | NA | 0 | 33 |
| No | No | 30.8 | 23.8 | NA | NA | 1005.7 | 1007.8 | 24 | 54 | 20 | 19 | W | W | 41 | WNW | NA | NA | 0 | 31.8 |
| No | No | 29 | 20.9 | NA | 5 | 1008.2 | 1011 | 23 | 55 | 13 | 6 | NW | ESE | 33 | N | NA | NA | 0 | 30.9 |
| No | No | 31.2 | 21.5 | NA | NA | 1010.1 | 1012.9 | 17 | 49 | 19 | 4 | W | E | 43 | W | NA | NA | 0 | 32.4 |
| No | No | 33 | 23.2 | 1 | NA | 1007.6 | 1010.9 | 19 | 45 | 13 | 9 | WSW | SE | 35 | WSW | NA | NA | 0 | 33.9 |
| No | No | 31.2 | 26.6 | 1 | NA | 1003.6 | 1006.8 | 28 | 41 | 26 | 0 | W | NA | 57 | WSW | NA | NA | 0 | 33 |
| No | No | 32.1 | 24.6 | NA | NA | 1001.7 | 1005.2 | 15 | 56 | 30 | 13 | WNW | N | 48 | WNW | NA | NA | 0 | 32.7 |
| Yes | No | 26.1 | 21.6 | NA | NA | 1004.2 | 1004.8 | 22 | 49 | 30 | 19 | WSW | NW | 46 | WNW | NA | NA | 0 | 27.2 |
| No | Yes | 18.2 | 12.5 | 8 | 8 | 1003.4 | 1005.6 | 70 | 78 | 22 | 11 | SW | WSW | 50 | WNW | NA | NA | 1.2 | 24.2 |

**3. Machine Learning Algorithms Selection**
In this section, we are going to discuss the candidate machine learning algorithms we are going to follow to solve the problem addressed in this report.

**3.1 Models Selection**
we are going to select **Liner Support Vector Machine Algorithm,** Used for Classification problems, error-based learning, and parametric model.
In SVM the goal is to find the best separation "line" between different classes in the features space.

The "Rain in Australia" dataset has two different categories that are classified, there can be multiple lines/decision boundaries to separate the class yes and class No in n-dimensional space.
Finding the best decision boundary with the maximum possible distance" Margin" that helps to classify the features is known as the hyperplane of SVM. The closest to the hyperplane and which affect the position of the hyperplane are termed Support Vectors, Since these vectors support the hyperplane.

How to find the best separation "line" ?

Must understand the basic terminologies with SVM in the figure:

SVM parametric model summarizes data with a set of parameters of fixed size (independent of the number of training examples). This model more efficient classifier than the other models since some points is more important than others and takes them into account.

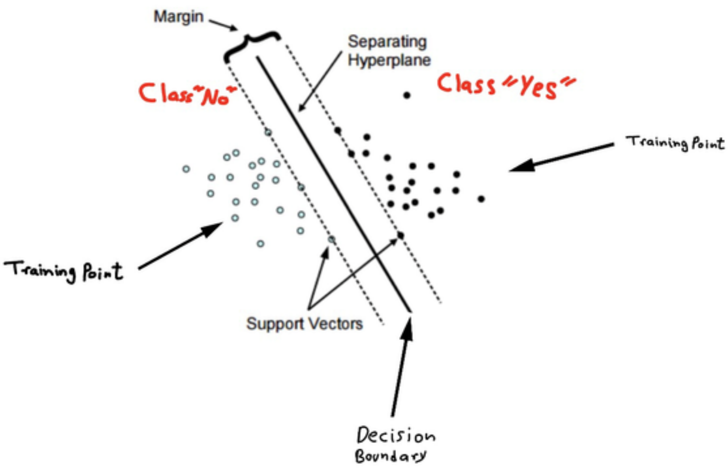



Figure 3: SVM model

**4. Implementation**
In this section, we going to explain the steps of our work on the problem that we previously explained.

**4.1 Importing Libraries**
we import these necessary libraries in our project.

- Scikit-learn (sklearn) : is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy. [14]

- from sklearn import metrics: to evaluate your machine learning algorithms
from sklearn.metrics import accursce_score: calculates the accuracy score for a set of predicted labels against the true labels.[12]

- from sklearn.metrics import confusion_matrix: It measures the quality of predictions from a classification model by looking at how many predictions are True and how many are False.[11]

- from sklearn.model_selection import train_test_split: Split arrays or matrices into random train and test subsets.[11]

- Import panadas as pd: It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array.[13]

- Import numpy as np: Onceimported NumPy, we can then use the functions built in it to quickly create and analyze data. [13]

- Import seaborn as sns: Seaborn is a Python data visualization library built on top of Matplotlib, Once imported Seaborn, we can then use the functions built in it to quickly visualize data. [13]

- Import matplotlib.pyplot as plt: matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. [13]

- from sklearn.linear_model import SGDOneClassSVM: Solves linear One-Class SVM using Stochastic Gradient Descent.[11]

- om sklearn import preprocessing: The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. 1

- from sklearn.ensemble import BaggingClassifier: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. 1

- from sklearn.multiclass import OneVsRestClassifier: this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. 1

- from sklearn.svm import SVC: This class is responsible for multi-class support using a one-to-one mechanism. 7

libraries

```
[1] # lib for linear algebra
    import numpy as np
    import matplotlib.pyplot as plt
    # processing
    import pandas as pd
    # data visualization
    # to see data
    import seaborn as sns
    from sklearn.linear_model import SGDOneClassSVM
    from sklearn import preprocessing
    # to measure accuracy
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import confusion_matrix
    from sklearn import metrics
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import BaggingClassifier
    from sklearn.multiclass import OneVsRestClassifier
    from sklearn.svm import SVC
```

**4.2 Load Data Set**

we loaded dataset by a method read_csv() from library pandas.

read data

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# use pandas to process data in our dataset
weather_data= pd.read_csv('drive/MyDrive/Colab Notebooks/dataset/weatherAUS.csv')
# to print dataset
weather_data
```

| | Date | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | ... | Humidity9am | Humidity3pm |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2008-12-01 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | ... | 71.0 | 22.0 |
| 1 | 2008-12-02 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | ... | 44.0 | 25.0 |
| 2 | 2008-12-03 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | ... | 38.0 | 30.0 |
| 3 | 2008-12-04 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | ... | 45.0 | 16.0 |
| 4 | 2008-12-05 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | ... | 82.0 | 33.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 145455 | 2017-06-21 | Uluru | 2.8 | 23.4 | 0.0 | NaN | NaN | E | 31.0 | SE | ... | 51.0 | 24.0 |

### 4.3 Data Pre processing

Data Pre processing is read and understand that data to makes sure data is clean and organizing to the Machine Learning model, which means the real-world data is often messy, incomplete, unstructured, inconsistent, redundant. so should be converting data to a suitable format to extract insights. [5]

```
size of weather data frame is : (145460, 23)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

**we observe :**

- Dataset has two data types: float64, object
- all column has missing values Except the Date, Location columns, .

### 4.4 Finding Categorical and Numerical Features in a Data set

here we going to find "categorical features" and "Numerical Features" in Dataset
by check dtype per column, if type of column=0 then the column is have categorical features otherwise the column have Numerical Features.[4]
dtype is an instance of numpy.dtype class, can describes type of data (integer, object, etc..) [6]

after we read dataset we going to prapere data + handling the missing values

finding categorical features

```
# Categorical features in Dataset
features1 = [column_name for column_name in weather_data.columns if weather_data[column_name].dtype == 'O']
print("Number of Categorical Features is: {}".format(len(features1)))
print("Categorical Features is: ",features1)
```

```
Number of Categorical Features is: 7
Categorical Features is:  ['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

```
[6] # Numerical Features in Dataset
features2 = [column_name for column_name in weather_data.columns if weather_data[column_name].dtype != 'O']
print("Number of Numerical Features is: {}".format(len(features2)))
print("Numerical Features is: ",features2)
```

```
Number of Numerical Features is: 16
Numerical Features is:  ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9
```

**4.5 Cardinality check for Categorical features**

because the accuracy of a classifier model "SVM" is depend on what kind of data we are feeding to the classifier model to learn, we should to find the cardinality. which means the number of unique values in each categorical feature is known as cardinality.

- **feature with a high unique values have high cardinality**

This is not good for the model because cause problems like makes the model doesn't generalise well to unseen examples (x_test,y_text) and big problem as curse of dimensionality.

simple summarization of  curse of dimensionality: when the dimensionality increases the volume of the space increases fast then the data become sparse and because the data come sparsity and dissimilar then prevents data organization strategies from being efficient. [7]

-to show high cardinality..

by calling data frame "weather_data" we call nunique() function this give the unique value for each category then we will plot the unique category, in the bar() we will defined the size of figure as parameter, finally we will make ylable and xlable as shown below:

```
weather_data.nunique().plot.bar(figsize=(12,6))
plt.ylabel("number of unique catrgories")
plt.xlabel("variables")
plt.title("cardinality check")
plt.show()
```

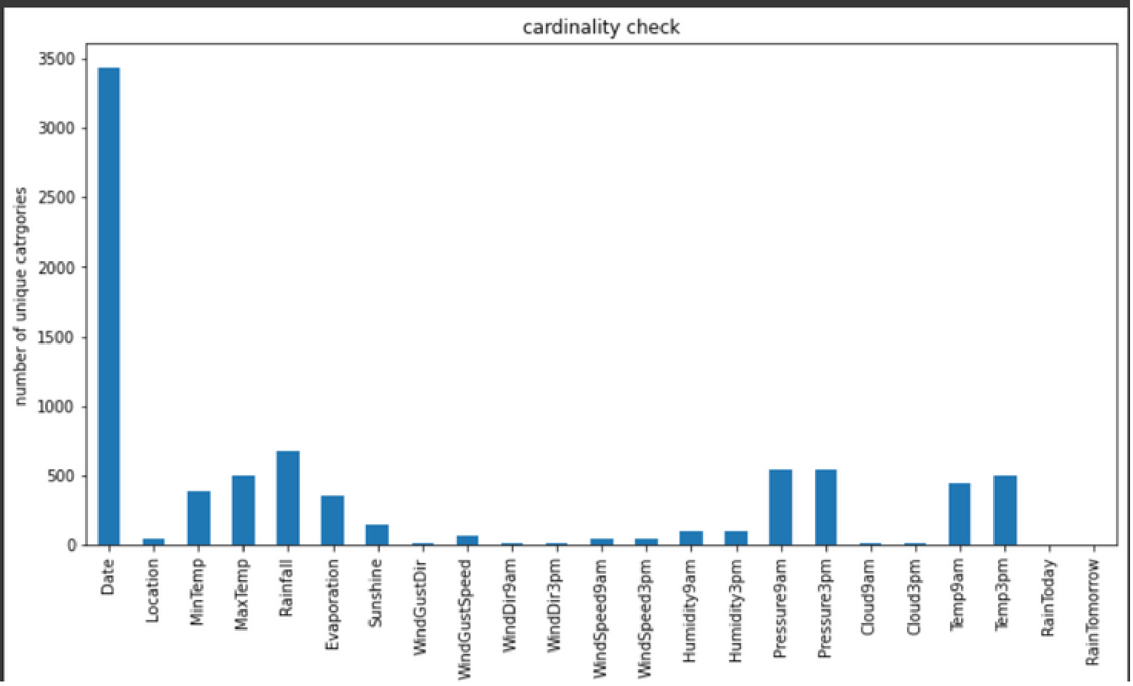



Figure 4 : cardinality check

```python
#Cardinality check
for each_feature in features1:
    value = len(weather_data[each_feature].unique())
    print("Cardinality of {} are: {}".format(each_feature, value))
```

```
Cardinality of Date are: 3436
Cardinality of Location are: 49
Cardinality of WindGustDir are: 17
Cardinality of WindDir9am are: 17
Cardinality of WindDir3pm are: 17
Cardinality of RainToday are: 3
Cardinality of RainTomorrow are: 3
```

we observe the higher cardinality is date have 3436 unique values "more number of category" which poses several problems to the model in terms of efficiency and dimensions of data increase when encoded to numerical data, for this reason we handling cardinality before encoding process. [5]
after we show the high cardinality we going to handling by :

- feature engineering

refers to manipulation addition, deletion, combination, mutation of your data set to improve the model training, leading to better accuracy.
feature engineering have several types we going to apply **Feature construction** which is creates new features from one feature. here, using the date we add a feature that indicates the date:[8]

| year | month | day |
|---|---|---|
| 2008 | 12 | 1 |
| 2008 | 12 | 2 |
| 2008 | 12 | 3 |
| 2008 | 12 | 4 |
| 2008 | 12 | 5 |

- dropping that feature if it doesn’t add any value to the model.

after we add features that indicates to the date we going to drop the date..

```python
#Feature Engineering of Date column to decrease high cardinality
weather_data['Date'] = pd.to_datetime(weather_data['Date'])
weather_data['year'] = weather_data['Date'].dt.year
weather_data['month'] = weather_data['Date'].dt.month
weather_data['day'] = weather_data['Date'].dt.day
```

```python
#Drop Date column
weather_data.drop('Date', axis = 1, inplace = True)
weather_data.head()
```

| | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | ... | Pressure3pm | Cloud9am | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | WNW | ... | 1007.1 | 8.0 | |
| 1 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | WSW | ... | 1007.8 | NaN | |
| 2 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | WSW | ... | 1008.7 | NaN | |
| 3 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | E | ... | 1012.8 | NaN | |
| 4 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | NW | ... | 1006.0 | 7.0 | |

5 rows × 25 columns

### 4.6 Handling Missing Values

Missing values occur due to various factors such as completely missing at random, missing at random, or non-existent at random. All this may result from a system glitch during data collection or human error during data pre-processing. Missing data handling is very important while pre-processing a dataset because many machine learning algorithms do not support missing values.[15]

If a dataset contains missing values and loaded using pandas, then missing values get replaced with NaN(Not a Number) values. These NaN values can be identified using methods like isna() or isnull() and they can be imputed using fillna(). This process is known as Missing Data Imputation. [15]

#### 4.6.1 Handling Missing values in categorical features

A categorical features(sometimes called a nominal features) is one that has two or more categories, but there is no intrinsic ordering to the categories. [16]

```
categorical_features = [column_name for column_name in weather_data.columns if weather_data[column_name].dtype == 'O']
weather_data[categorical_features].isnull().sum()
```

```
Location            0
WindGustDir     10326
WindDir9am      10566
WindDir3pm       4228
RainToday        3261
RainTomorrow     3267
dtype: int64
```

```
# Imputing the missing values in categorical features using the most frequent value which is mode
categorical_features_with_null = [feature for feature in categorical_features if weather_data[feature].isnull().sum()]
for each_feature in categorical_features_with_null:
    mode_val = weather_data[each_feature].mode()[0]
    weather_data[each_feature].fillna(mode_val,inplace=True)
```

##### 4.6.1.1 Imputing the missing values in categorical features using the most frequent value which is mode

Missing values is from categorical columns such as string or numerical then the missing values can be replaced with the most frequent category. [17]

```
weather_data.head()
```

| on | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | ... | Pressure3pm | Cloud9am | Cloud3pm | Te |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | WNW | ... | 1007.1 | 8.0 | NaN | |
| ury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | WSW | ... | 1007.8 | NaN | NaN | |
| ury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | WSW | ... | 1008.7 | NaN | 2.0 | |
| ury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | E | ... | 1012.8 | NaN | NaN | |
| ury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | NW | ... | 1006.0 | 7.0 | 8.0 | |

columns

```
# the null values after Imputing the missing values in categorical features
weather_data[categorical_features].isnull().sum()
```

```
Location        0
WindGustDir     0
WindDir9am      0
WindDir3pm      0
RainToday       0
RainTomorrow    0
dtype: int64
```

### 4.6.2 Handling Missing values in Numerical features

Missing values in Numerical Features can be imputed using Mean and Median. Mean is sensitive to outliers and median is immune to outliers. We want to impute the missing values with mean values, then outliers in numerical features need to be addressed properly. [18]

```
[ ] #Handling Missing values in Numerical features
    numerical_features = [column_name for column_name in weather_data.columns if weather_data[column_name].dtype != 'O']
    weather_data[numerical_features].isnull().sum()
```

```
MinTemp           1485
MaxTemp           1261
Rainfall          3261
Evaporation      62790
Sunshine         69835
WindGustSpeed    10263
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am       2654
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
year                 0
month                0
day                  0
dtype: int64
```

#### 4.6.2.1 Outliers detection and treatment

Outliers are those data points that are significantly different from the rest of the dataset. To ensure that the trained model generalizes well to the valid range of test inputs, it’s important to detect and remove outliers. [19]

They can be detected using visualization box plots which displays the five-number summary of a set of data. [20]

We shown in screenshot process of detection outliers and handling missing values ..

```
[ ] # after we find the missing values in Numerical Features can be imputed using Mean and Median
    # before should handle the outliers
    features_with_outliers = ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'WindGustSpeed','WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Pres:
    for feature in features_with_outliers:
        q1 = weather_data[feature].quantile(0.25)
        q3 = weather_data[feature].quantile(0.75)
        IQR = q3-q1
        lower_limit = q1 - (IQR*1.5)
        upper_limit = q3 + (IQR*1.5)
        weather_data.loc[weather_data[feature]<lower_limit,feature] = lower_limit
        weather_data.loc[weather_data[feature]>upper_limit,feature] = upper_limit
```

```
numerical_features_with_null = [feature for feature in numerical_features if weather_data[feature].isnull().sum()]
for feature in numerical_features_with_null:
    mean_value = weather_data[feature].mean()
    weather_data[feature].fillna(mean_value,inplace=True)
```

```
[ ] #show data frame after handling missing values by mean
    weather_data.head()
```

| | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | ... | Pressure3pm | Cloud9am | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Albury | 13.4 | 22.9 | 0.6 | 5.318667 | 7.611178 | W | 44.0 | W | WNW | ... | 1007.1 | 8.000000 | 4 |
| 1 | Albury | 7.4 | 25.1 | 0.0 | 5.318667 | 7.611178 | WNW | 44.0 | NNW | WSW | ... | 1007.8 | 4.447461 | 4 |
| 2 | Albury | 12.9 | 25.7 | 0.0 | 5.318667 | 7.611178 | WSW | 46.0 | W | WSW | ... | 1008.7 | 4.447461 | 2 |
| 3 | Albury | 9.2 | 28.0 | 0.0 | 5.318667 | 7.611178 | NE | 24.0 | SE | E | ... | 1012.8 | 4.447461 | 4 |
| 4 | Albury | 17.5 | 32.3 | 1.0 | 5.318667 | 7.611178 | W | 41.0 | ENE | NW | ... | 1006.0 | 7.000000 | 8 |

5 rows × 25 columns

-to show Outliers..
The points that lie beyond the whiskers are detected as outliers.

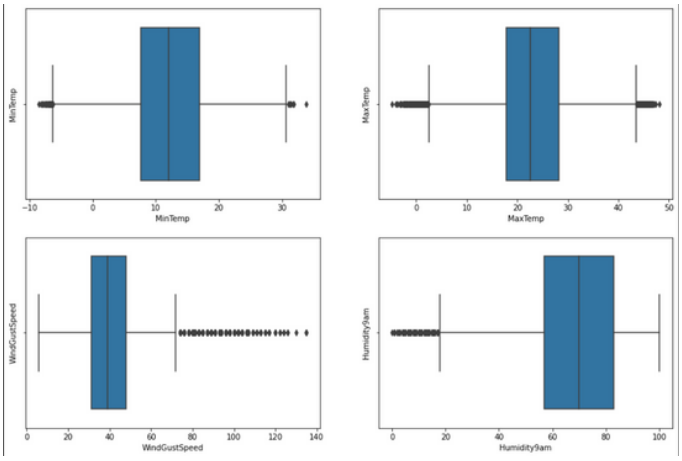


Figure 5: Outliers

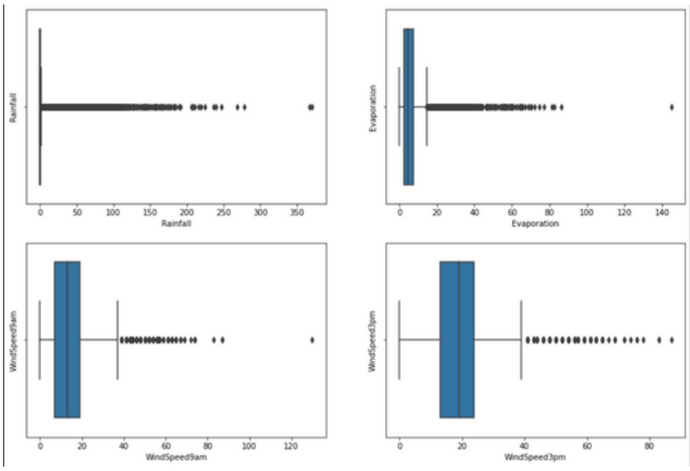


Figure 6: Outliers

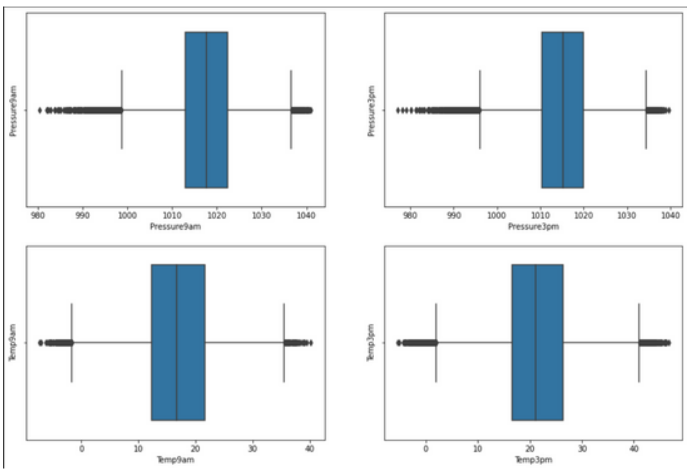


Figure 7: Outliers

**-after handling all null values**

```
#count missing values for each column >> by function isnull()
print('the missing values for each column is :')
print('_____________________________________________')
weather_data.isna().sum()
```

```
the missing values for each column is :
_____________________________________________
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
year             0
month            0
day              0
dtype: int64
```

```
#count valid values for each column
print('the valid values for each column is :')
print(weather_data.count().sort_values())
```

```
the valid values for each column is :
Location         145460
year             145460
RainTomorrow     145460
RainToday        145460
Temp3pm          145460
Temp9am          145460
Cloud3pm         145460
Cloud9am         145460
Pressure3pm      145460
Pressure9am      145460
Humidity3pm      145460
month            145460
Humidity9am      145460
WindSpeed9am     145460
WindDir3pm       145460
WindDir9am       145460
WindGustSpeed    145460
WindGustDir      145460
Sunshine         145460
Evaporation      145460
Rainfall         145460
MaxTemp          145460
MinTemp          145460
WindSpeed3pm     145460
day              145460
dtype: int64
```

```
print('result of each column, after handling missing values :')
#weather_data.isnull().
weather_data.isnull().sum().sort_values(ascending=False)
print('___________________________________________________________')
weather_data.head
```

```
result of each column, after handling missing values :
___________________________________________________________
<bound method NDFrame.head of         Location  MinTemp    MaxTemp  Rainfall  Evaporation  Sunshine  \
0         Albury     13.4  22.900000       0.6     5.318667  7.611178
1         Albury      7.4  25.100000       0.0     5.318667  7.611178
2         Albury     12.9  25.700000       0.0     5.318667  7.611178
3         Albury      9.2  28.000000       0.0     5.318667  7.611178
4         Albury     17.5  32.300000       1.0     5.318667  7.611178
...          ...      ...        ...       ...          ...       ...
145455     Uluru      2.8  23.400000       0.0     5.318667  7.611178
145456     Uluru      3.6  25.300000       0.0     5.318667  7.611178
145457     Uluru      5.4  26.900000       0.0     5.318667  7.611178
145458     Uluru      7.8  27.000000       0.0     5.318667  7.611178
145459     Uluru     14.9  23.224781       0.0     5.318667  7.611178

       WindGustDir  WindGustSpeed WindDir9am WindDir3pm  ...  Pressure3pm  \
0                W      44.000000          W        WNW  ...       1007.1
1              WNW      44.000000        NNW        WSW  ...       1007.8
2              WSW      46.000000          W        WSW  ...       1008.7
3               NE      24.000000         SE          E  ...       1012.8
4                W      41.000000        ENE         NW  ...       1006.0
...            ...            ...        ...        ...  ...          ...
145455           E      31.000000         SE        ENE  ...       1020.3
145456         NNW      22.000000         SE          N  ...       1019.1
145457           N      37.000000         SE        WNW  ...       1016.8
145458          SE      28.000000        SSE          N  ...       1016.5
145459           W      39.837792        ESE        ESE  ...       1017.9
```

**4.7 Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is a technigue used to analyze the frequency and other such characteristics of data, visualize the relationship that may exist between different variables, Understand the trends and patterns of data, and Know the distribution of the variables in the data. [5]

**4.7.1 Univariate Analysis**

**4.7.1.1 Exploring target variable**

The term univariate analysis refers to the analysis of one variable, which is target variable RainTomorow.

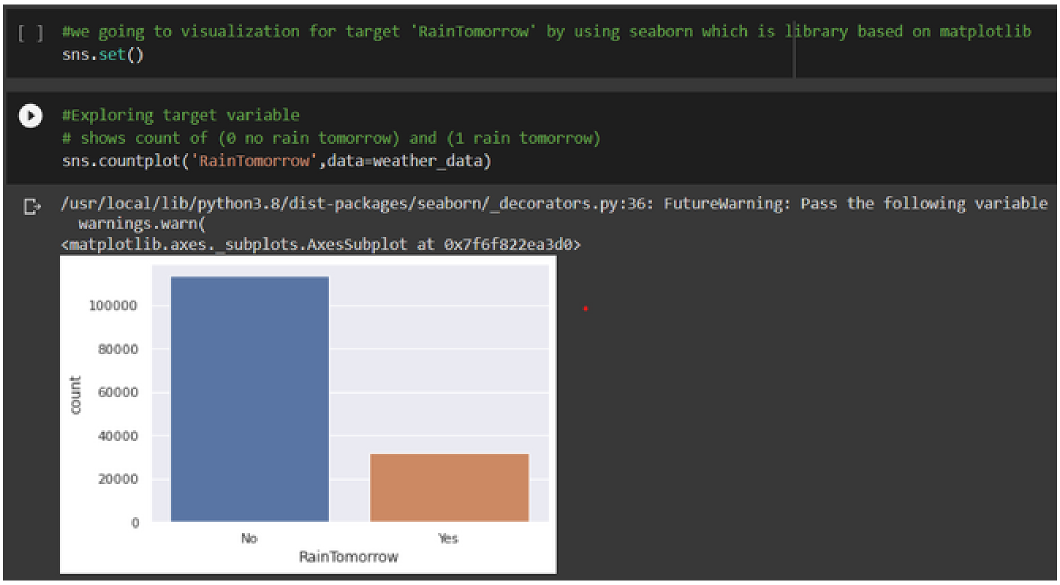



Figure 8:univariate analysis

Target variable is imbalanced. It has more ‘No’ values. If data is imbalanced, then it might decrease the performance of the model. since the target is related to meteorology, there will be no credibility if we balance the data.

Another variable we used is RainToday , and we visualization RainTomorrow according to it.


Figure 9: RainToday vs RainTomorrow

we observe if RainToday '0' then RainTomorrow is '0' almost 38000 from 43993 and '1' almost 5800 from 12427
If RainToday '1' then RainTomorrow is '0' almost 5800 from 43993 and '1' almost 5100 from 12427 , which is this logical division corresponds to visualization for RainTomorrow on the above.

### 4.7.2 Bi-variate Analysis

Bivariate analysis lets you study the relationship that exists between two variables. This has a lot of use in real life. It helps to find out if there is an association between the variables.

#### 4.7.2.1 Sunshine vs Rainfall

```
#show relationship between Sunshine and Rainfall
sns.lineplot(data=weather_data,x='Sunshine',y='Rainfall',color='green')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6f8228a730>
```

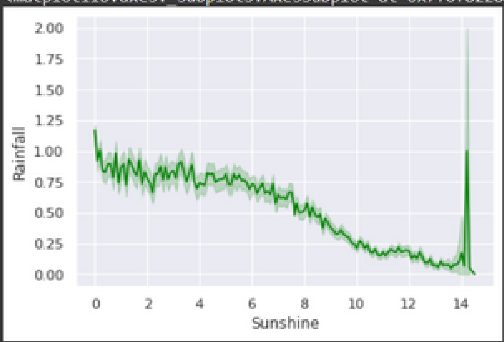



Figure 10 : Sunshine vs Rainfall

In the above line plot, the sunshine feature is inversely proportional to the rainfall feature, and this makes a lot of sense since if the sun is shining, there will be no precipitation and vice versa.

#### 4.7.2.2 Sunshine vs Evaporation

```
[ ] #show relationship between Sunshine and Evaporation
sns.lineplot(data=weather_data,x='Sunshine',y='Evaporation',color='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6f821cb550>
```

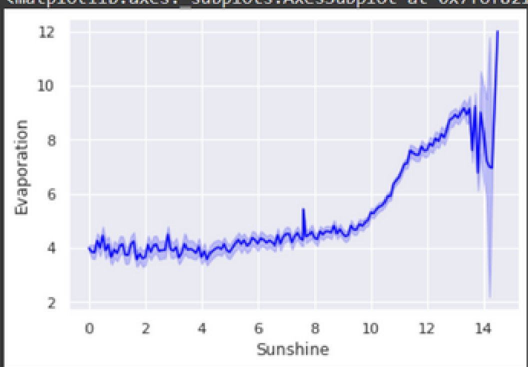



Figure 11 : Sunshine vs Evaporation

In the above line plot, we also see that the Sunshine feature is proportional to the Evaporation feature.[5]

### 4.8 Feature Encoding

SVM can’t handle categorical data, these categorical data need to be converted to numerical data for modeling, which is called  Feature Encoding.

There are many feature encoding techniques, we use two techniques:

A- replace() function to encode binary categorical data to numerical data.

convet string values to numerical values

```
# the model to work, we must give it numeric values
# first, we convert data has tow binary variables 'RainToday' and 'RainTomorrow' (yea-no) into quantitative variables (0-1)

# Replace No and Yes for 0 and 1 in RainToday and RainTomorrow by using replace function
weather_data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
weather_data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)

#show column after converting
weather_data.RainTomorrow
```

```
0         0
1         0
2         0
3         0
4         0
         ..
145455    0
145456    0
145457    0
145458    0
145459    0
Name: RainTomorrow, Length: 145460, dtype: int64
```

B- A dummy (binary) variable just takes the value 0 or 1 to indicate the exclusion or inclusion of a category. [22]

```
#Categorical variables WindGustDir, WindDir3pm and WindDir9am
#each of them Categorical variables with 16 categories, we going to conver each category into additional variable by dummy variables.
# we use dummy because have several Categorical variables not just a binary as raintoday can we use function replace()
categoric_values = ['WindGustDir', 'WindDir3pm', 'WindDir9am','Location']
weather_finaldata = pd.get_dummies(weather_data, columns=categoric_values)
weather_data.RainTomorrow
print('______________________________________________________')
display(weather_finaldata.head)
```

```
______________________________________________________
<bound method NDFrame.head of         MinTemp    MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0          13.4  22.900000       0.6     5.318667  7.611178      44.000000
1           7.4  25.100000       0.0     5.318667  7.611178      44.000000
2          12.9  25.700000       0.0     5.318667  7.611178      46.000000
3           9.2  28.000000       0.0     5.318667  7.611178      24.000000
4          17.5  32.300000       1.0     5.318667  7.611178      41.000000
...         ...        ...       ...          ...       ...            ...
145455      2.8  23.400000       0.0     5.318667  7.611178      31.000000
145456      3.6  25.300000       0.0     5.318667  7.611178      22.000000
145457      5.4  26.900000       0.0     5.318667  7.611178      37.000000
145458      7.8  27.000000       0.0     5.318667  7.611178      28.000000
145459     14.9  23.224781       0.0     5.318667  7.611178      39.837792

        WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  ...  \
0               20.0          24.0         71.0         22.0  ...
1                4.0          22.0         44.0         25.0  ...
2               19.0          26.0         38.0         30.0  ...
3               11.0           9.0         45.0         16.0  ...
4                7.0          20.0         82.0         33.0  ...
```

**4.9 Correlation**

here we going to apply Correlation heatmap which is visualize the strength of relationships between two features, means how each column related to each other, we use seabron library to create heatmap then we calculated correlation  by method corr() in pandas.

so now we explain the figure below, the Correlation can take any value between -1 and 1, and there is tow type of Correlation the first one is positive occurs when tow variables move in the same direction here the maximum positive Correlation is +1.

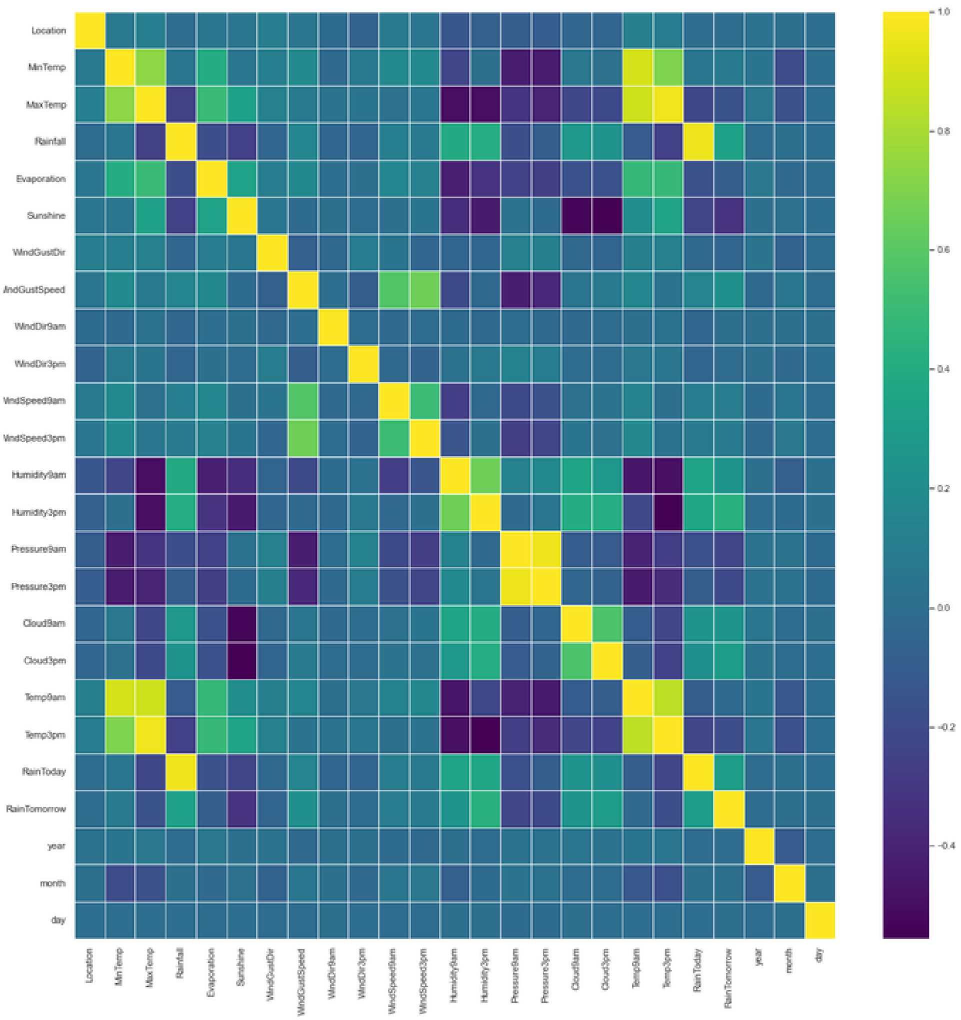
- there is positive Correlation between Timp9am and Mintemp because is greater than 0.7.

and the second one is negative occurs when tow variables move in opposite directions "one increases and other is decreases here the negative Correlation is -0.4

- there is negative Correlation between cloud9am and sunshine because have purple color which mean -0.4
- also as shown in the figure there is several variables have no Correlation and whose correlation value near to 0 [9]

```
[ ]  plt.figure(figsize=(20,20))
     sns.heatmap(weather_data.corr(), linewidths=0.5, annot=False, fmt=".2f", cmap = 'viridis')
```

Figure 12 :correlation



### 4.10 Splitting data into Independent Features and Dependent Features

For feature importance and feature scaling, we need to split data into independent and dependent features.

* X – Independent Features or Input features
* y – Dependent Features or target label

```
value of y is :
 0        0
1        0
2        0
3        0
4        0
         ..
145455    0
145456    0
145457    0
145458    0
145459    0
Name: RainTomorrow, Length: 145460, dtype: int64
```

```
values of x is
:         MinTemp    MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0          13.4  22.900000       0.6     5.318667  7.611178      44.000000
1           7.4  25.100000       0.0     5.318667  7.611178      44.000000
2          12.9  25.700000       0.0     5.318667  7.611178      46.000000
3           9.2  28.000000       0.0     5.318667  7.611178      24.000000
4          17.5  32.300000       1.0     5.318667  7.611178      41.000000
...         ...        ...       ...          ...       ...            ...
145455      2.8  23.400000       0.0     5.318667  7.611178      31.000000
145456      3.6  25.300000       0.0     5.318667  7.611178      22.000000
145457      5.4  26.900000       0.0     5.318667  7.611178      37.000000
145458      7.8  27.000000       0.0     5.318667  7.611178      28.000000
145459     14.9  23.224781       0.0     5.318667  7.611178      39.837792

        WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  ...  \
0               20.0          24.0         71.0         22.0  ...
1                4.0          22.0         44.0         25.0  ...
2               19.0          26.0         38.0         30.0  ...
3               11.0           9.0         45.0         16.0  ...
4                7.0          20.0         82.0         33.0  ...
...              ...           ...          ...          ...  ...
145455          13.0          11.0         51.0         24.0  ...
145456          13.0           9.0         56.0         21.0  ...
145457           9.0           9.0         53.0         24.0  ...
145458          13.0           7.0         51.0         24.0  ...
```

### 4.10.1 Splitting Data into training and testing set

train_test_split() is a method of model_selection class used to split data into training and testing sets.

```
# split x,y into test and train by sklearn.model_selection and give train_test_split x,y and the percentage given to each train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.10, random_state=2)

print('total x', X.shape)
print('X train shape: ', X_train.shape)
print('X test shape: ', X_test.shape)

print('________________________________________ \n')

print('total y', Y.shape)
print('Y train shape: ', Y_train.shape)
print('Y test shape: ', Y_test.shape)

#then give x,y to the model
# build model by x train and y train
# and test model by x test and y test ??

# take 5642 for test from total x 56420 -- and take 50778 for train from 56420
# take from total y 56420 -- 50778 for training and 5642 for test
```

```
total x (145460, 117)
X train shape:  (130914, 117)
X test shape:  (14546, 117)
________________________________________

total y (145460,)
Y train shape:  (130914,)
Y test shape:  (14546,)
```

### 4.11 Model Building

#### 4.11.1 Model Training

we going to build svm model by sklearn library then we will give the model tha training set that we specified previously as (x_train and y_train) for training. we use instance of learning model "svmla" to call method fit().
fit() method is train the algorithm on training data after we initialized the svm model, by accept tow argument, one for sample data x and other in our case "supervised" is accept also argument for labels y, once the svm model is trained then we can use it to make predictions based on learning parameter (x_train and y_train). [10]

```
#Import svm model
from sklearn.svm import SVC
#Create a svm Classifier
svmcla = SVC(kernel = 'linear', random_state =0)
#Train the model using the training sets only
svmcla.fit(X_train, Y_train)

SVC(kernel='linear', random_state=0)
```

after we train SVM model on training set we going to make prediction based on same data and save result of this prediction in "y_predict2" then we check on accuracy of this prediction by using accuracy_score to compare the prediction result from learning model and the actual result for training data.
-the accuracy is 0.8486 .

```
[ ] #We predict result of X_train by the model
    # run the model on same data for built, which means on X_train not x_test
    Y_predict2 = svmcla.predict(X_train)
    print(Y_predict2)

    [0 0 0 ... 1 0 0]

[ ] #check accuracy on training data by accuracy_score function
    #give the function real result y_train and result that our model is predicted Y_predict2 to compare between them to determine the accuracy
    # accuracy may be high because we predict same values that model built over it

    accuracy= accuracy_score(Y_train,Y_predict2)
    print('the accuracy on seen data is : ',accuracy)

    the accuracy on seen data is :  0.8486028996134867
```

### 4.11.2 Model Testing

here we are going to test learning model by unseen data, which means we didn't give it learning model during the training phase.
we give x_test to the learning model as unseen data and save prediction result in y_predict2 then we print the result [000 ... 010] as shown in screenshot, (0 --> no rain and 1 --> rain).
after that we check the accuracy by using accuracy_score to compare between prediction result and unseen target y_test.
-the accuracy is 0.8453 .

```
[ ] # We predict values for X_test by svm model
    # which is the target values is x_test --> 5642 .. the model is not built on these values

    Y_predict2 = svmcla.predict(X_test)
    print(Y_predict2)

    [0 0 0 ... 0 1 0]

[ ] # check accuracy on test data
    #give the function unseen data

    accuracy= accuracy_score(Y_test,Y_predict2)
    print('the accuracy on unseen data is : ',accuracy)

    the accuracy on unseen data is :  0.8453183005637288
```

### 4.12 More evaluating Model Performance

accuracy score which is based on measure how many fails and how many true e.g. when have 5 prediction true and 5 prediction fails then result of the accuracy from accuracy score will be 50%, sometimes this measure not enough to model accuracy is judged !
we going to apply other metrices in following sections.

### 4.12.1 Confusion Matrix

we going to show a table that define the performance of classification algorithm.
we obtained using confusion_matrix() function which is part from sklearn library.

true negative : 10780
number of negative examples classified accurately.
true positive : 1516
number of positive example classified accurately
false negative : 1670
number of positive example classified as negative
false positive : 580
number of negative example classified as positive

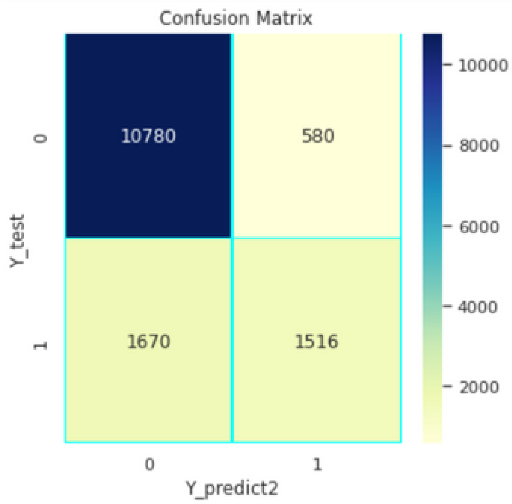


Figure 13 : confusion metric

we observed the learning model is achieves the lowest percentage in (FP) and (FN) , while the (TN) and (TP) is increase.

we going to manual calculated the accuracy according the values in confusion matrix by this function: $\text{Accuracy} = \frac{TN+TP}{TN+FP+FN+TP}$

(10780+1516)/(10780+1516+580+1670) = 0.845 "same by accuracy_score"

also can compute misclassification = (FN)/(TP+TN+FP+FN) = 1670/(580+1516+10780)=0.129

now we going to compute recall or sensitivity and precision :

- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)

TP/(TP+FP) = 1516/1516+580=0.723

TP/(TP+FN)=1516/1516+1670=0.475

compute F1 score =2*precision*recall/precision + recall =2*0.723*0.475/0.723+0.475=0.573

"can be compute by function in python as we shown in following sections"

### 4.12.2 Classification-report

here we going to show how we can compute recall, precision and F1 in python by using functions from sklearn. [21]

```
[59] recall=metrics.recall_score(Y_test, Y_predict2)
     print('the recall score is : \n ',recall)

     the recall score is :
       0.4758317639673572

     precision=metrics.precision_score(Y_test, Y_predict2)
     print('the precision score is : \n ',precision)

     the precision score is :
       0.7232824427480916

[61] f1=metrics.f1_score(Y_test, Y_predict2)
     print('the f1 score is : \n ',f1)

     the f1 score is :
       0.5740249905338888

[62] report=metrics.classification_report(Y_test, Y_predict2)
     print('give whole report of the model \n:',report)

     give whole report of the model
     :               precision    recall  f1-score   support

                0       0.87      0.95      0.91     11360
                1       0.72      0.48      0.57      3186

         accuracy                           0.85     14546
        macro avg       0.79      0.71      0.74     14546
     weighted avg       0.83      0.85      0.83     14546
```

### 4.12.3 Cross-validation

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

```
# Cross-Validation
from sklearn.model_selection import cross_val_score
scores = cross_val_score(svmcla, X_train, Y_train, cv = 5, scoring='accuracy')
print('Cross-validation scores:{}'.format(scores))
print('Average cross-validation score: {}'.format(scores.mean()))
```

The mean accuracy score of cross-validation is almost the same as the original model accuracy score which is 0.845. So, the accuracy of the model may not be improved using Cross-validation.

### 4.13 checking for underfitting and overfitting

Overfitting models produce good predictions for data points in the training set but perform poorly on new samples. Underfitting occurs when the machine learning model is not well-tuned to the training set. The resulting model is not capturing the relationship between input and output well enough.

show if have overfitting and underfitting

```
print("Train Data Score: {}".format(svmcla.score(X_train, Y_train)))
print("Test Data Score: {}".format(svmcla.score(X_test, Y_test)))
```

```
Train Data Score: 0.8486028996134867
Test Data Score: 0.8453183005637288
```

In the beginning, we trained the model on the training data regardless of the presence of outliers and deleted them, so we obtained the accuracy: “0.839”.
But then we arrived SVM accuracy = “0.845” after applying:
1- Handling Missing Values.
2- Outliers detection and treatment.
3- dropping that feature if it doesn’t add any value to the model using Feature construction.

After that, we compared the performance of SVM with other algorithms using confusion matrix , as follows: [1]

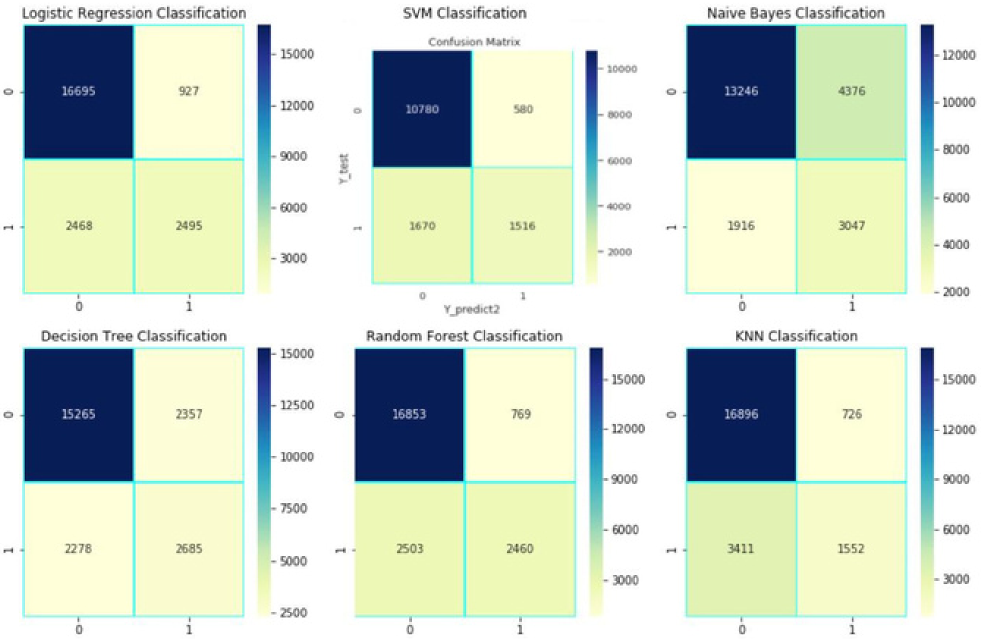



Figure 14 : confusion metrics of different algorithms

We noticed SVM performance approach to another algorithm performance in terms of accuracy, so we got these results:

Naive bayes classification = 0.7214

Decision tree classification = 0.794

Random forest classification = 0.855

K-Nearest Neighbor classification = 0.816

# CONCLUSION

In this report, we have presented our work on a precipitation prediction dataset in Australia using the SVM learning algorithm, as it has relatively high performance if it includes a large dataset to generalize a problem. The main strength of SVM is that training the data is relatively easy, since the goal of training is to predict whether it will RainTomorrow with a set of important properties such as RainToday, Sunshine, and so on.

After training the algorithm on the existing data, we got an accuracy of 0.84 compared to the training results, which means that the algorithm was good at working on this type of data.

In this project, we encountered some challenges and problems that deteriorated the flow of work which is:

1-The SVM model took a lot of time to run, which led to a delay in work for some time

2-The emergence of many outliers that affected the results

3-The difference in free time between group members, which made it difficult to synchronize work

However, we were able to address these problems, learn from them, and complete this project in the time allotted.

# REFERENCES

[1] jperezm9. “Rain Classification in Australia.” Kaggle, Kaggle, 7 June 2019, https://www.kaggle.com/code/jperezm9/rain-classification-in-australia.

[2] Raj, Ravish. “Classification and Regression Problems in Machine Learning.” Enjoyalgorithms, https://www.enjoyalgorithms.com/blogs/classification-and-regression-in-machine-learning.

[3] “Support Vector Machine (SVM) Algorithm - Javatpoint.” Www.javatpoint.com, https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[4] venkat0107venkat0107 24011 silver badge1212 bronze badges, et al. “How to Find No of Categorical Columns and Numerical Columns in Dataset.” Stack Overflow, 1 June 1967, https://stackoverflow.com/questions/62455152/how-to-find-no-of-categorical-columns-and-numerical-columns-in-dataset.

[5] Rain Prediction in Australia | Predictive Modelling Using Python. https://www.analyticsvidhya.com/blog/2021/06/predictive-modelling-rain-prediction-in-australia-with-python/.

[6] “Data Type Objects (Dtype)#.” Data Type Objects (Dtype) - NumPy v1.24 Manual, https://numpy.org/doc/stable/reference/arrays.dtypes.html.

[7] Medium â€“ Where Good Ideas Find You., https://link.medium.com/xkstRFpVixb.

[8] “What Is Feature Engineering?: Domino Data Science Dictionary.” What Is Feature Engineering? | Domino Data Science Dictionary, https://www.dominodatalab.com/data-science-dictionary/feature-engineering.

[9] Kumar, Ajitesh. “Correlation Concepts, Matrix & Heatmap Using Seaborn.” Data Analytics, 16 Apr. 2022, https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/amp/.

[10] Medium â€“ Where Good Ideas Find You., https://link.medium.com/0WDOdT2kjxb.

# REFERENCES

[11] “Sklearn.metrics.confusion_matrix.” Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

[12] “What Is the accuracy_score Function in Sklearn?” Educative, https://www.educative.io/answers/what-is-the-accuracyscore-function-in-sklearn.

[13] Zach. “The Easiest Way to Use Seaborn: Import Seaborn as SNS.” Statology, 16 June 2021, https://www.statology.org/import-seaborn-as-sns.

[14] Pal, Satyabrata. “Scikit-Learn Tutorial: Machine Learning in Python.” Dataquest, 8 Feb. 2023, https://www.dataquest.io/blog/sci-kit-learn-tutorial.

[15] Emmanuel, Tlamelo, et al. “A Survey on Missing Data in Machine Learning - Journal of Big Data.” SpringerOpen, Springer International Publishing, 27 Oct. 2021, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00516-9.

[16] “Home.” OARC Stats, https://stats.oarc.ucla.edu/other/mult-pkg/whatstat/what-is-the-difference-between-categorical-ordinal-and-interval-variables/.

[17] “How Do You Handle Missing Values, Categorical Data and Feature Scaling in Machine Learning - Pianalytix - Machine Learning.” Pianalytix, 4 Nov. 2020, https://pianalytix.com/how-do-you-handle-missing-values-categorical-data-and-feature-scaling-in-machine-learning/.

[18] Rain Prediction in Australia | Predictive Modelling Using Python. https://www.analyticsvidhya.com/blog/2021/06/predictive-modelling-rain-prediction-in-australia-with-python/.

[19] C, Bala Priya. “How to Detect Outliers in Machine Learning – 4 Methods for Outlier Detection.” FreeCodeCamp.org, FreeCodeCamp.org, 11 July 2022, https://www.freecodecamp.org/news/how-to-detect-outliers-in-machine-learning/amp/.

# R E F E R E N C E S

[20] “Box Plot Review (Article).” Khan Academy, Khan Academy, https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review.

[21] “Confusion Matrix.” Confusion Matrix - an Overview | ScienceDirect Topics, https://www.sciencedirect.com/topics/engineering/confusion-matrix.

[22] Pramoditha, Rukshan. “Encoding Categorical Variables: One-Hot vs Dummy Encoding.” Medium, Towards Data Science, 16 Dec. 2021, https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db.