BITI 3313: NEURAL NETWORK

SEM 2 2022/2023

ASSIGNMENT 2 (10%)

TITLE:

Rain Prediction By Neural Network For Classification

DUE DATE:

LECTURER: PROFESOR TS. DR. BURHANUDDIN BIN MOHD ABOOBAIDER

GROUP MEMBER:

NAME	MATRIC NUMBER
PRIYADHARSHNI A/P MOHAN NATHAN	B032110285
ALIYAH NAJMA BINTI NADZRI	B032110398
SITI AZALIA BINTI MEHAT	B032110338
MARSHITAH BINTI AZHAR	B032110355

Table of Contents

1) **Introduction**

Weather forecasting plays an integral role in our daily lives, providing valuable information for a wide range of activities and decision-making processes. Among the various weather elements, accurate rainfall prediction holds significant importance due to its profound impact on agriculture, water resource management, flood control, and overall planning and preparedness. In this regard, Artificial Neural Networks (ANNs) have emerged as a powerful computational tool for enhancing rainfall prediction accuracy and delivering timely forecasts.

Artificial neural networks, inspired by the structure and functioning of the human brain, are a sophisticated computational approach. Comprising interconnected nodes, known as neurons, organized in layers, ANNs have the capacity to learn from historical data patterns and relationships. This ability enables them to make predictions and generalize to new, previously unseen data. Consequently, ANNs are exceptionally well-suited for addressing complex and nonlinear problems like rainfall prediction.

The utilization of artificial neural networks (ANNs) to predict rainfall involves training a network with historical weather data. This data includes various meteorological variables such as temperature, humidity, wind speed, and atmospheric pressure. By examining the patterns and connections within this data, the network becomes proficient in identifying the underlying relationships between these variables and the occurrence of rainfall. Once trained, the ANN can effectively forecast future rainfall based on current weather conditions.

There are numerous advantages to using ANNs for rainfall prediction. Firstly, they can capture intricate nonlinear relationships and adapt to changing patterns in the data. They are capable of handling a large number of input variables and can learn from both spatial and temporal patterns. Additionally, ANNs can incorporate real-time data updates, enabling dynamic and up-to-date predictions. With their ability to process vast amounts of data and recognize subtle patterns, ANNs have the potential to enhance the accuracy and lead time of rainfall predictions. This, in turn, aids in effective planning and decision-making across various sectors.

This study aims to explore the application of artificial neural networks in the field of rainfall prediction. To accomplish this, we will utilize a dataset consisting of 10 years of daily weather observations from different locations in Australia to train an ANN model. The performance and accuracy of the trained model will be assessed using.

2) **Detail description about sample data**

The dataset that we are utilizing for our analysis is the weatherAUS dataset, which is a comprehensive collection of weather observations obtained from various locations across Australia. This dataset, sourced from the esteemed Bureau of Meteorology, encompasses a span of approximately 10 years, providing us with a significant amount of historic weather data that contain 23 attributes, encompassing both numerical and categorical values.

In order to forecast rain using Artificial Neural Networks (ANNs), we use historical weather data including temperature, humidity, wind speed, and atmospheric pressure as an input variables, alongside corresponding rainfall measurements. The ANN is trained on this data to comprehend the intricate relationships and patterns between the input variables and rainfall.

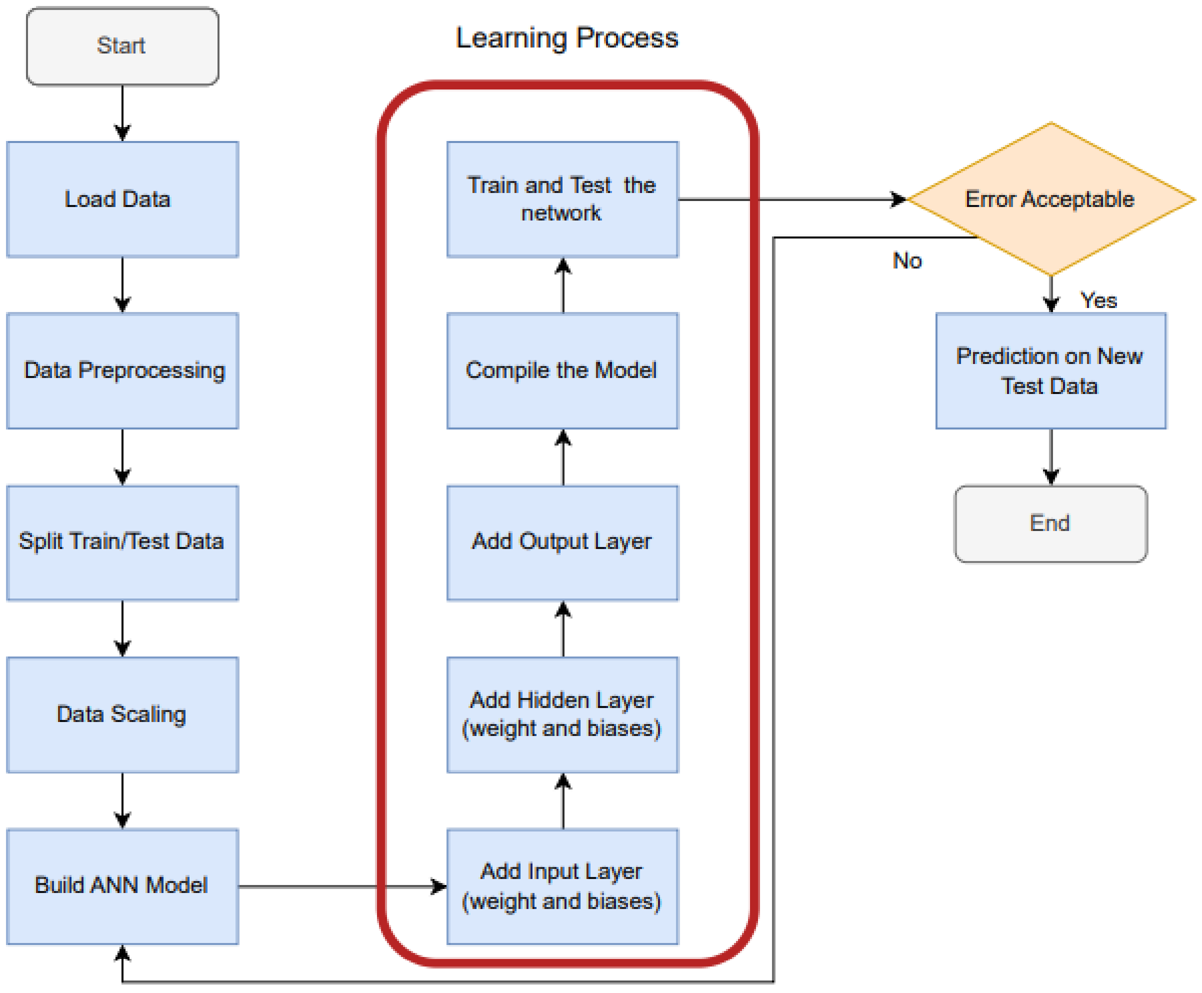
Throughout the training phase, the ANN modifies the weights and biases of its neurons to minimize the disparity between its predictions and the actual rainfall measurements. Once the training is completed, the ANN can make predictions on new, unseen data.

To anticipate future rainfall, the trained ANN takes the current weather conditions as input and generates an output that represents the predicted rainfall. The ANN's capability to capture nonlinear relationships in the data makes it ideal for rain prediction, as weather patterns often exhibit complex and nonlinear behavior.

It is worth noting that accurate rain prediction using ANNs depends on the availability of high-quality and up-to-date weather data for training and validation. Moreover, the performance of the ANN is influenced by various factors, such as the network's architecture, the quality of the input data, and the effectiveness of the training process.

In conclusion, this weatherAUS dataset serves as a valuable resource in our quest to improve rain prediction capabilities. Through the application of artificial neural networks and meticulous analysis of this extensive dataset, we strive to enhance our understanding of Australia's weather patterns and provide more accurate forecasts.

3) <u>Flow chart and learning process</u>



4) Step by step learning and analysis

1. Importing the data

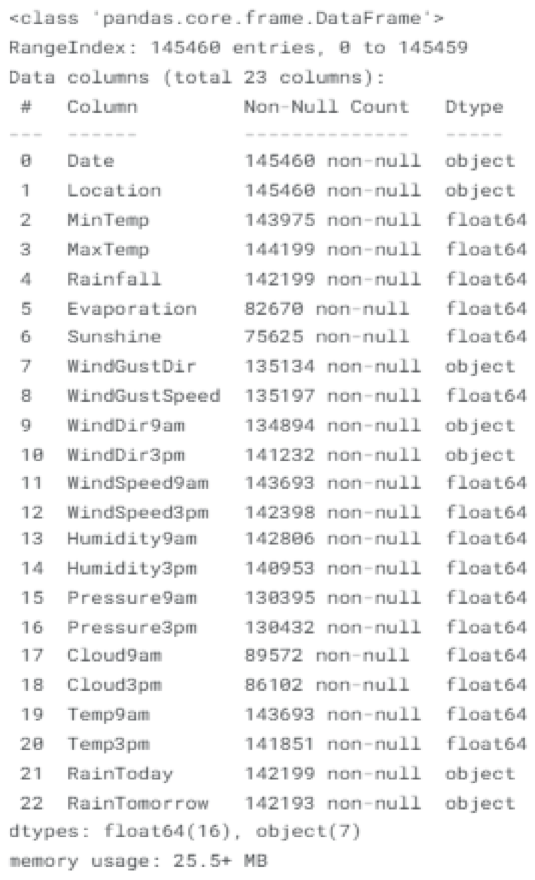
Source of Data : https://www.kaggle.com/code/kaan0397/rain-in-australia-classification?scriptVersionId=22931077&cellId=2

About the data:

The dataset contains about 10 years of daily weather observations from different locations across Australia. Observations were drawn from numerous weather stations.

The use of data to predict whether or not it will rain the next day. There are 23 attributes including the target variable "RainTomorrow", indicating whether or not it will rain the next day or not.

Data info :

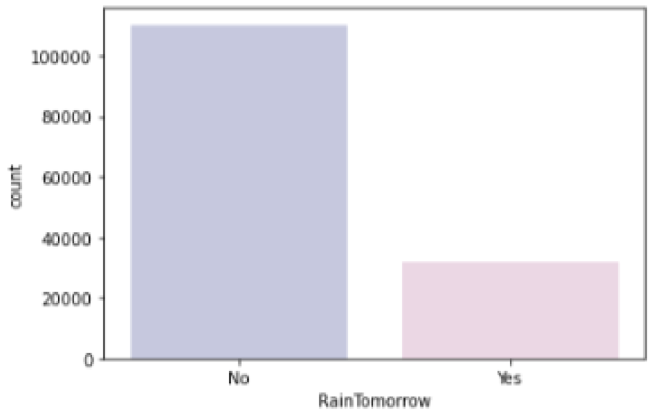


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        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
```

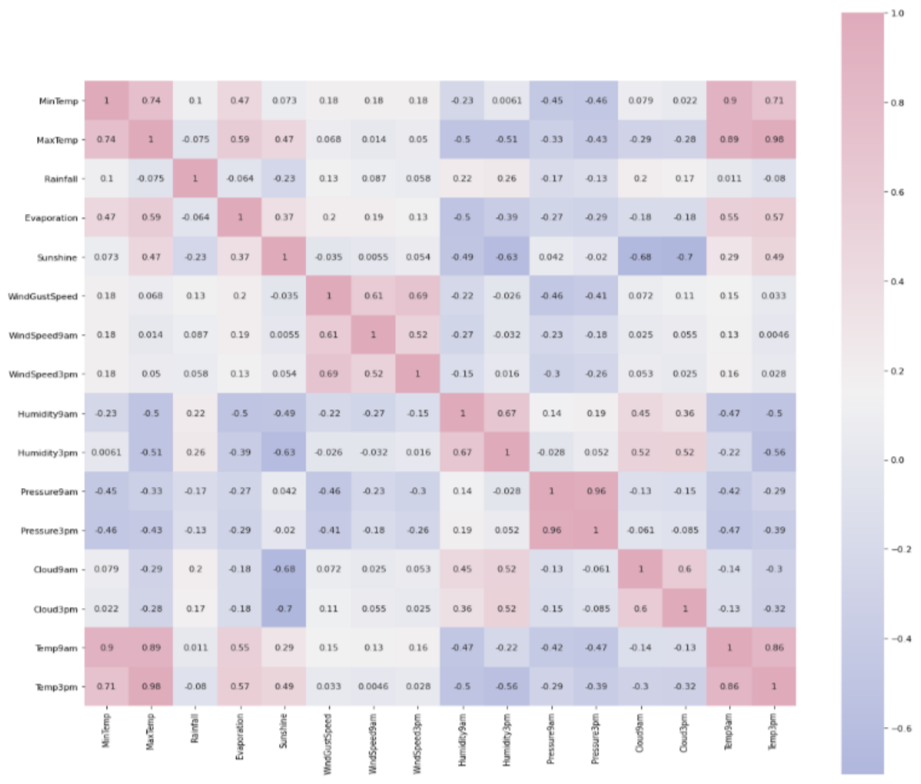
2. Data visualization and cleaning

Data visualization refers to the graphical representations of data and information. While data cleaning known ass data cleansing or data scrubbing, is the process of identifying and correcting or removing errors, inconsistencies and inaccuracies from a dataset. The steps involves in this project are count plot of target column, correlation amongst numeric attributes, parse dates into datetime and encoding days and months as continuous cyclic features.

- Evaluate the target and find out if the data is imbalance or not.

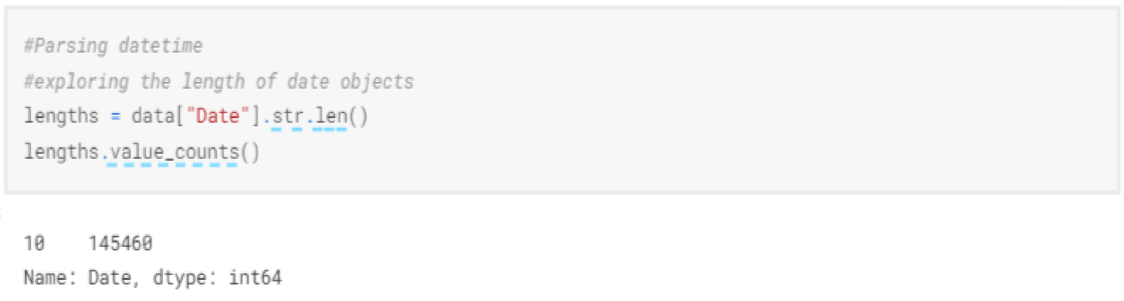


- Do the correlation amongst numeric attributes.



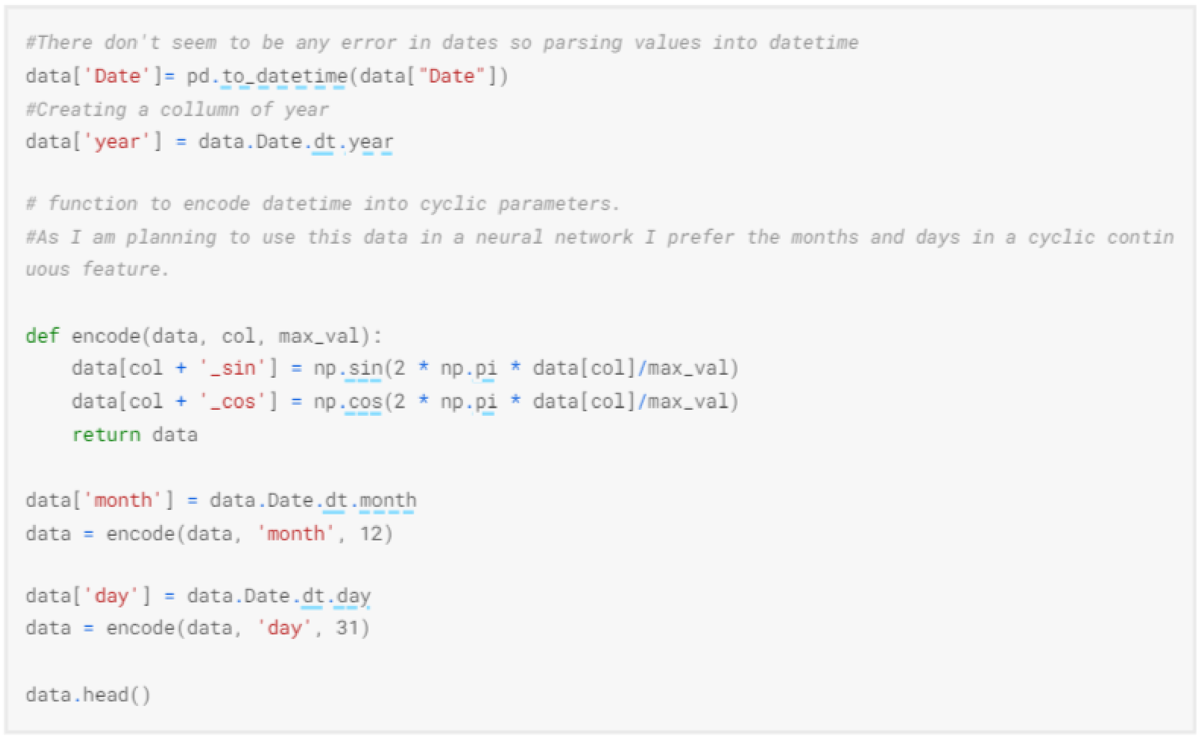
- Parse Dates into datetime.

  The purpose is to build an artificial neural network (ANN). I will encode dates appropriately, i.e. I prefer the months and days in a cyclic continuous feature. As, date and time are inherently cyclical. To let the ANN model know that a feature is cyclical I split it into periodic subsections. Namely, years, months and days. Now for each subsection, I create two new features, deriving a sine transform and cosine transform of the subsection feature.



```
#Parsing datetime
#exploring the length of date objects
lengths = data["Date"].str.len()
lengths.value_counts()
```

```
10    145460
Name: Date, dtype: int64
```

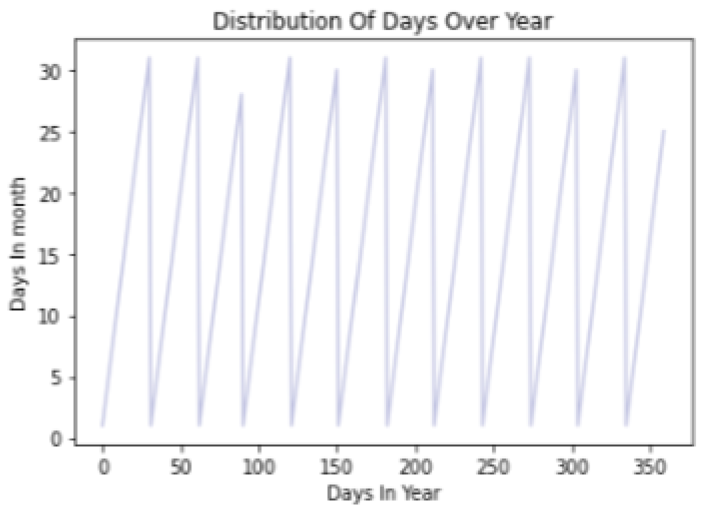


```
#There don't seem to be any error in dates so parsing values into datetime
data['Date']= pd.to_datetime(data["Date"])
#Creating a collumn of year
data['year'] = data.Date.dt.year

# function to encode datetime into cyclic parameters.
#As I am planning to use this data in a neural network I prefer the months and days in a cyclic contin
uous feature.

def encode(data, col, max_val):
    data[col + '_sin'] = np.sin(2 * np.pi * data[col]/max_val)
    data[col + '_cos'] = np.cos(2 * np.pi * data[col]/max_val)
    return data

data['month'] = data.Date.dt.month
data = encode(data, 'month', 12)

data['day'] = data.Date.dt.day
data = encode(data, 'day', 31)

data.head()
```

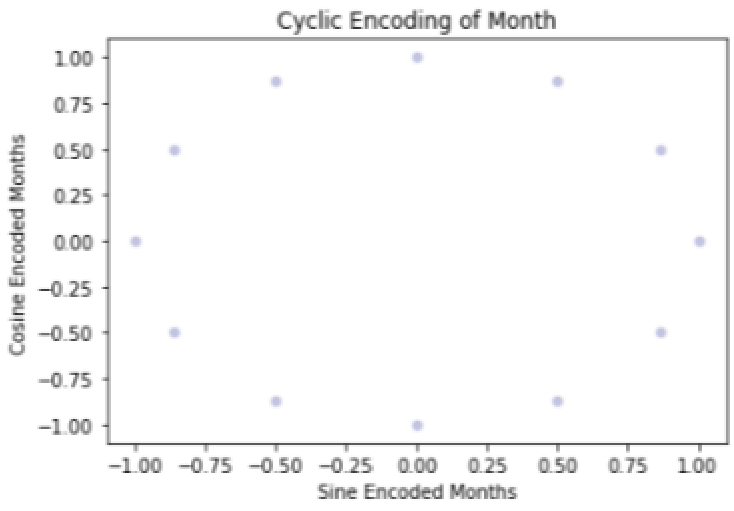
By encoding the months and days in this way, will allow the ANN model to understand the cyclical patterns in the data more effectively. The code also includes parsing the date column as datetime and extracting the year, month, and day components from the dates.

```
# roughly a year's span section
section = data[:360]
tm = section["day"].plot(color="#C2C4E2")
tm.set_title("Distribution Of Days Over Year")
tm.set_ylabel("Days In month")
tm.set_xlabel("Days In Year")
```

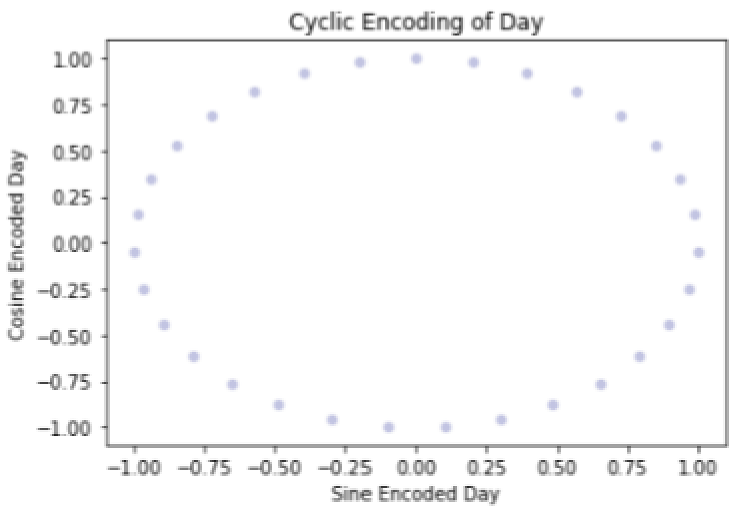
```
Text(0.5, 0, 'Days In Year')
```



This code snippets does demonstrate the visualization of the cyclical encoding of the month and day features using sine and cosine transformations.



This result crates a catter plot to visualize the cyclical encoding of the months. The x-axis represents the sine-encoded months, and the y-axis represents the cosine-encoded months.This encoding allows the cyclical nature of the months to be represented in a continuous manner. The scatter plot helps visualize the distribution and patterns in the encoded month data.



The result creates a similar scatter plot to visualize the cyclical encoding of days. Again, the x-axis represents the sine-encoded days, and the y-axis represents the cosine-encoded days. This encoding captures the cyclic nature of the days in a continuous manner, enabling the neural network to better understand the patterns in the data.

- Handle missing values in categorical and numeric attributes

  - Categorical Variables

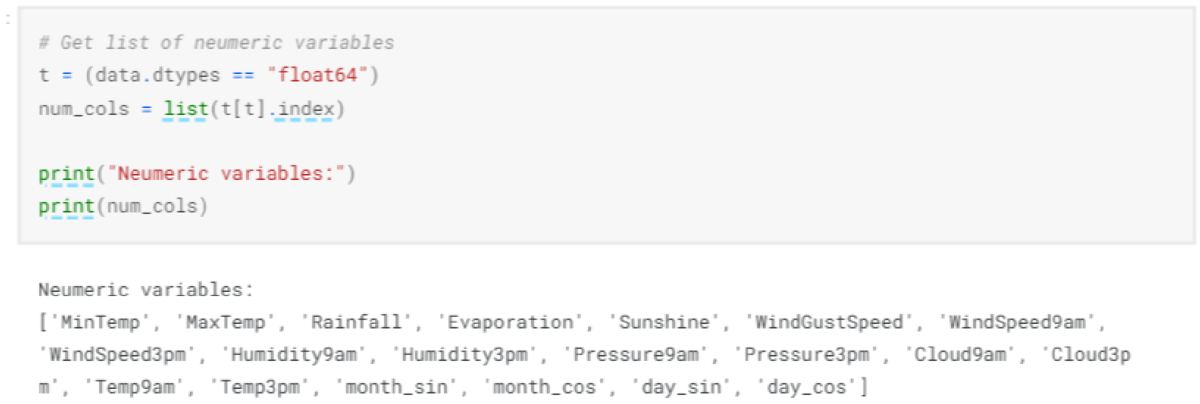
```
# Get list of categorical variables
s = (data.dtypes == "object")
object_cols = list(s[s].index)

print("Categorical variables:")
print(object_cols)
```

```
Categorical variables:
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

```
# Filling missing values with mode of the column in value

for i in object_cols:
    data[i].fillna(data[i].mode()[0], inplace=True)
```

To fill the missing values in categorical variables, one common approach is to replace them with the mode of each respective column. The mode represents the most frequent value in a column.

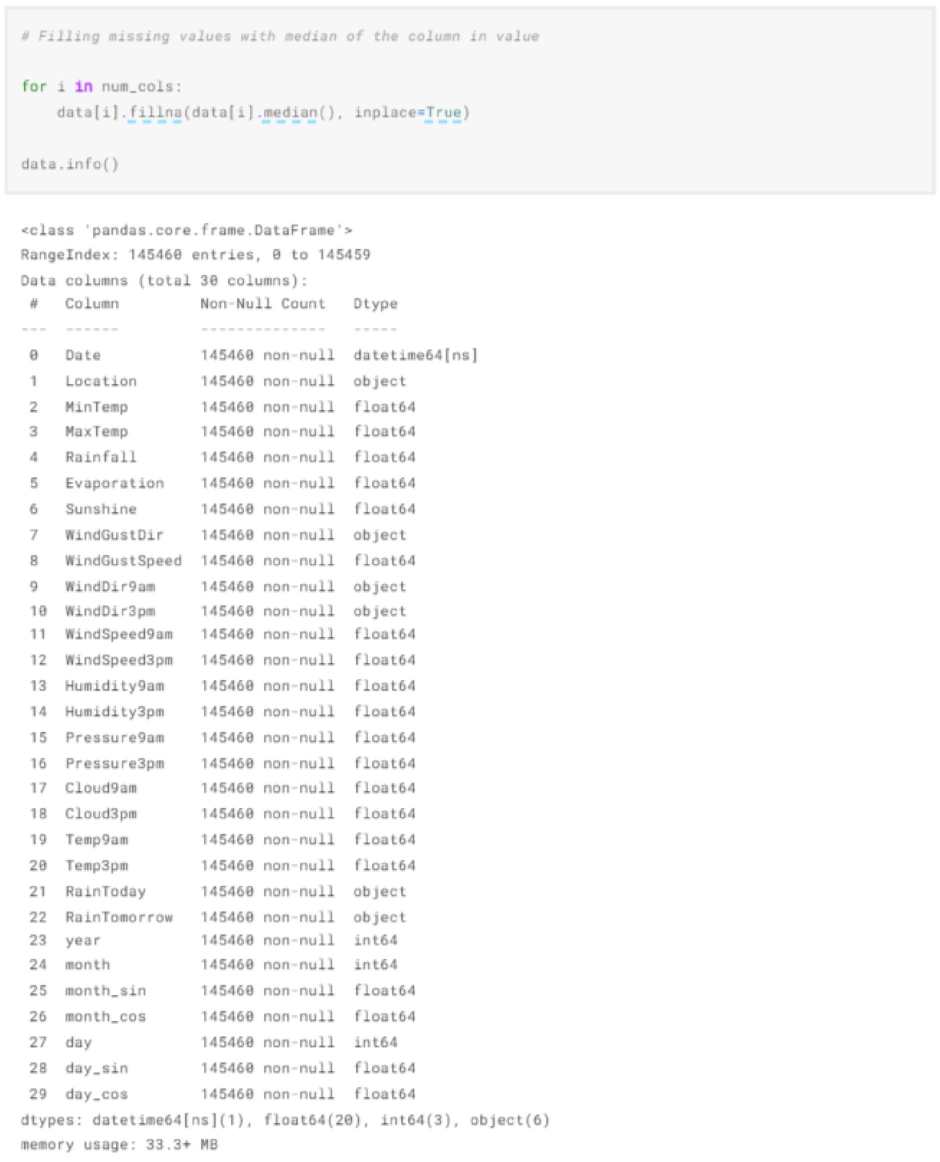
- Numerical Variables



```
# Get list of neumeric variables
t = (data.dtypes == "float64")
num_cols = list(t[t].index)

print("Neumeric variables:")
print(num_cols)
```

```
Neumeric variables:
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am',
'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3p
m', 'Temp9am', 'Temp3pm', 'month_sin', 'month_cos', 'day_sin', 'day_cos']
```

Filling missing values with median of the column value



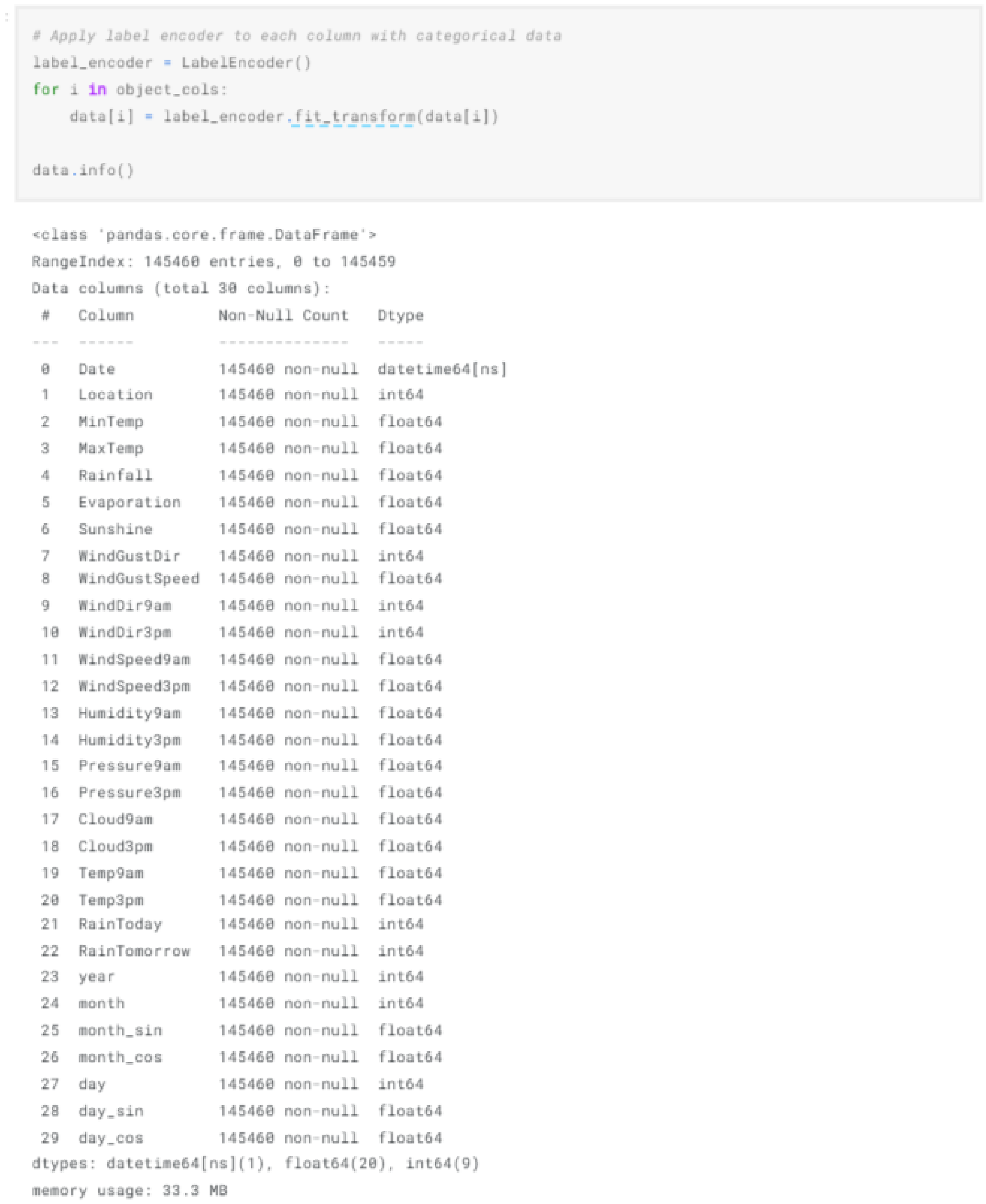
```
# Filling missing values with median of the column in value

for i in num_cols:
    data[i].fillna(data[i].median(), inplace=True)

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 30 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  datetime64[ns]
 1   Location       145460 non-null  object
 2   MinTemp        145460 non-null  float64
 3   MaxTemp        145460 non-null  float64
 4   Rainfall       145460 non-null  float64
 5   Evaporation    145460 non-null  float64
 6   Sunshine       145460 non-null  float64
 7   WindGustDir    145460 non-null  object
 8   WindGustSpeed  145460 non-null  float64
 9   WindDir9am     145460 non-null  object
 10  WindDir3pm     145460 non-null  object
 11  WindSpeed9am   145460 non-null  float64
 12  WindSpeed3pm   145460 non-null  float64
 13  Humidity9am    145460 non-null  float64
 14  Humidity3pm    145460 non-null  float64
 15  Pressure9am    145460 non-null  float64
 16  Pressure3pm    145460 non-null  float64
 17  Cloud9am       145460 non-null  float64
 18  Cloud3pm       145460 non-null  float64
 19  Temp9am        145460 non-null  float64
 20  Temp3pm        145460 non-null  float64
 21  RainToday      145460 non-null  object
 22  RainTomorrow   145460 non-null  object
 23  year           145460 non-null  int64
 24  month          145460 non-null  int64
 25  month_sin      145460 non-null  float64
 26  month_cos      145460 non-null  float64
 27  day            145460 non-null  int64
 28  day_sin        145460 non-null  float64
 29  day_cos        145460 non-null  float64
dtypes: datetime64[ns](1), float64(20), int64(3), object(6)
memory usage: 33.3+ MB
```

By executing this code, the missing value in numeric columns will be filled with the median of each respective column.

3. Data Preprocessing

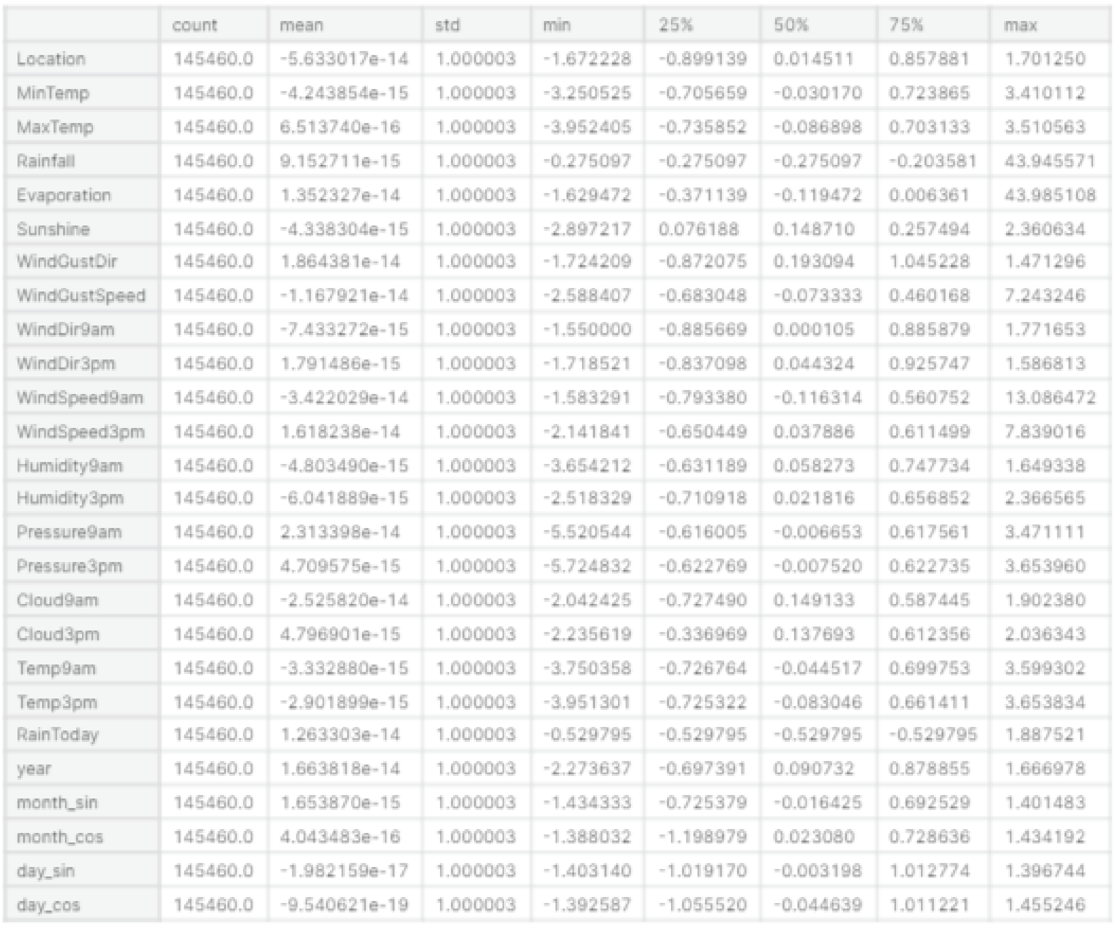
In this section, some steps involved are label encoding columns with categorical data, perform the scaling of the features, detecting outliers and dropping the outliers based on data analysis.



```
# Apply label encoder to each column with categorical data
label_encoder = LabelEncoder()
for i in object_cols:
    data[i] = label_encoder.fit_transform(data[i])

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 30 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  datetime64[ns]
 1   Location       145460 non-null  int64
 2   MinTemp        145460 non-null  float64
 3   MaxTemp        145460 non-null  float64
 4   Rainfall       145460 non-null  float64
 5   Evaporation    145460 non-null  float64
 6   Sunshine       145460 non-null  float64
 7   WindGustDir    145460 non-null  int64
 8   WindGustSpeed  145460 non-null  float64
 9   WindDir9am     145460 non-null  int64
 10  WindDir3pm     145460 non-null  int64
 11  WindSpeed9am   145460 non-null  float64
 12  WindSpeed3pm   145460 non-null  float64
 13  Humidity9am    145460 non-null  float64
 14  Humidity3pm    145460 non-null  float64
 15  Pressure9am    145460 non-null  float64
 16  Pressure3pm    145460 non-null  float64
 17  Cloud9am       145460 non-null  float64
 18  Cloud3pm       145460 non-null  float64
 19  Temp9am        145460 non-null  float64
 20  Temp3pm        145460 non-null  float64
 21  RainToday      145460 non-null  int64
 22  RainTomorrow   145460 non-null  int64
 23  year           145460 non-null  int64
 24  month          145460 non-null  int64
 25  month_sin      145460 non-null  float64
 26  month_cos      145460 non-null  float64
 27  day            145460 non-null  int64
 28  day_sin        145460 non-null  float64
 29  day_cos        145460 non-null  float64
dtypes: datetime64[ns](1), float64(20), int64(9)
memory usage: 33.3 MB
```

The categorical columns in the data frame will be label encoded, replacing the categorical values with corresponding numerical labels.

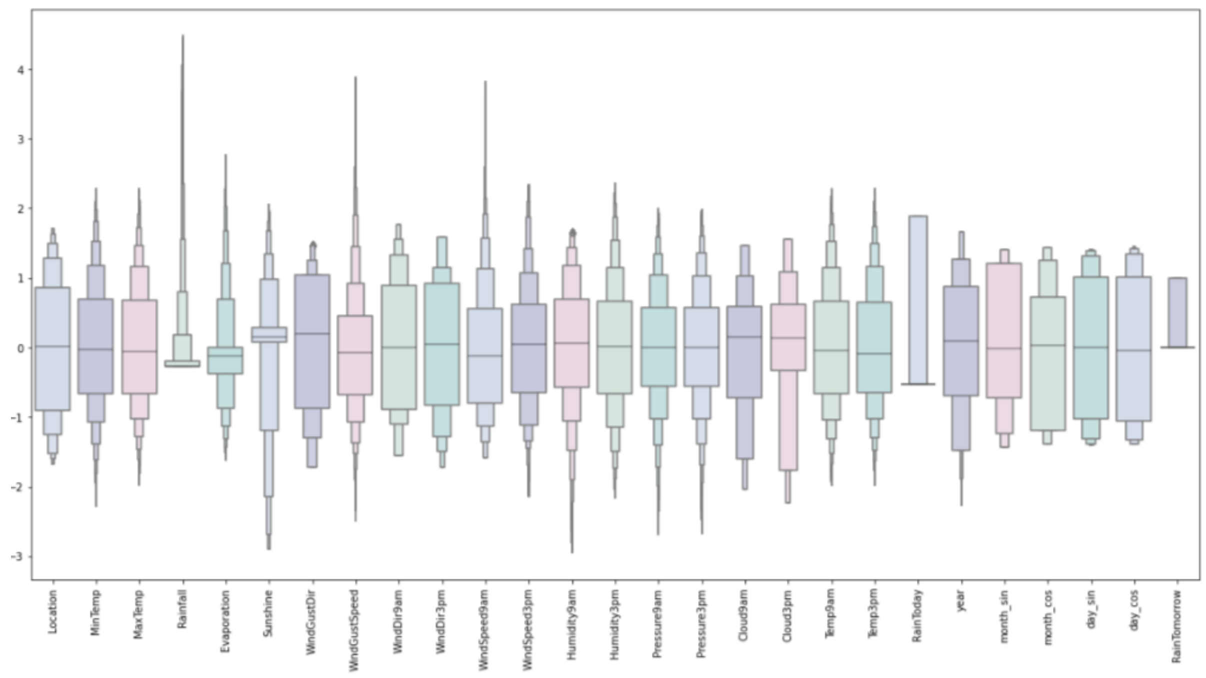


	count	mean	std	min	25%	50%	75%	max
Location	145460.0	-5.633017e-14	1.000003	-1.672228	-0.899139	0.014511	0.857881	1.701250
MinTemp	145460.0	-4.243854e-15	1.000003	-3.250525	-0.705659	-0.030170	0.723865	3.410112
MaxTemp	145460.0	6.513740e-16	1.000003	-3.952405	-0.735852	-0.086898	0.703133	3.510563
Rainfall	145460.0	9.152711e-15	1.000003	-0.275097	-0.275097	-0.275097	-0.203581	43.945571
Evaporation	145460.0	1.352327e-14	1.000003	-1.629472	-0.371139	-0.119472	0.006361	43.985108
Sunshine	145460.0	-4.338304e-15	1.000003	-2.897217	0.076188	0.148710	0.257494	2.360634
WindGustDir	145460.0	1.864381e-14	1.000003	-1.724209	-0.872075	0.193094	1.045228	1.471296
WindGustSpeed	145460.0	-1.167921e-14	1.000003	-2.588407	-0.683048	-0.073333	0.460168	7.243246
WindDir9am	145460.0	-7.433272e-15	1.000003	-1.550000	-0.885669	0.000105	0.885879	1.771653
WindDir3pm	145460.0	1.791486e-15	1.000003	-1.718521	-0.837098	0.044324	0.925747	1.586813
WindSpeed9am	145460.0	-3.422029e-14	1.000003	-1.583291	-0.793380	-0.116314	0.560752	13.086472
WindSpeed3pm	145460.0	1.618238e-14	1.000003	-2.141841	-0.650449	0.037886	0.611499	7.839016
Humidity9am	145460.0	-4.803490e-15	1.000003	-3.654212	-0.631189	0.058273	0.747734	1.649338
Humidity3pm	145460.0	-6.041889e-15	1.000003	-2.518329	-0.710918	0.021816	0.656852	2.366565
Pressure9am	145460.0	2.313398e-14	1.000003	-5.520544	-0.616005	-0.006653	0.617561	3.471111
Pressure3pm	145460.0	4.709575e-15	1.000003	-5.724832	-0.622769	-0.007520	0.622735	3.653960
Cloud9am	145460.0	-2.525820e-14	1.000003	-2.042425	-0.727490	0.149133	0.587445	1.902380
Cloud3pm	145460.0	4.796901e-15	1.000003	-2.235619	-0.336969	0.137693	0.612356	2.036343
Temp9am	145460.0	-3.332880e-15	1.000003	-3.750358	-0.726764	-0.044517	0.699753	3.599302
Temp3pm	145460.0	-2.901899e-15	1.000003	-3.951301	-0.725322	-0.083046	0.661411	3.653834
RainToday	145460.0	1.263303e-14	1.000003	-0.529795	-0.529795	-0.529795	-0.529795	1.887521
year	145460.0	1.663818e-14	1.000003	-2.273637	-0.697391	0.090732	0.878855	1.666978
month_sin	145460.0	1.653870e-15	1.000003	-1.434333	-0.725379	-0.016425	0.692529	1.401483
month_cos	145460.0	4.043483e-16	1.000003	-1.388032	-1.198979	0.023080	0.728636	1.434192
day_sin	145460.0	-1.982159e-17	1.000003	-1.403140	-1.019170	-0.003198	1.012774	1.396744
day_cos	145460.0	-9.540621e-19	1.000003	-1.392587	-1.055520	-0.044639	1.011221	1.455246

The result of preparing attributes of scale data by dropping unnecessary columns and scaling the numerical features using standard scaling. Resulting in a dataframe with standardized feature values.

```
#Detecting outliers
#looking at the scaled features
colours = ["#D0DBEE", "#C2C4E2", "#EED4E5", "#D1E6DC", "#BDE2E2"]
plt.figure(figsize=(20,10))
sns.boxenplot(data = features,palette = colours)
plt.xticks(rotation=90)
plt.show()
```

It creates a boxen plot using ‘sns.boxenplot()’ function to visualize the distribution of the scaled features in the ‘features’ Dataframe. The plot is customized with different colors for each box. This plots helps indentifying outliers in the data.



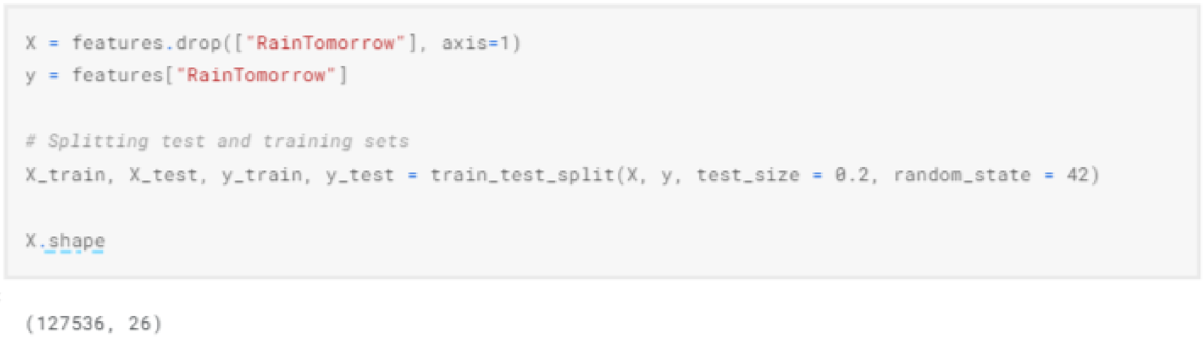
It creates a boxen plot to visualize the distribution of the scaled features in the 'features' Dataframe after removing outliers. The plot helps in assessing the impact of outlier removal on the distribution of the data.

Overall this section detects outliers n the features data, removes the outliers based on specific filtering conditions and provides visualizations before and provides visualizations before and after outlier removal to assess the data distribution.

4. Model Building

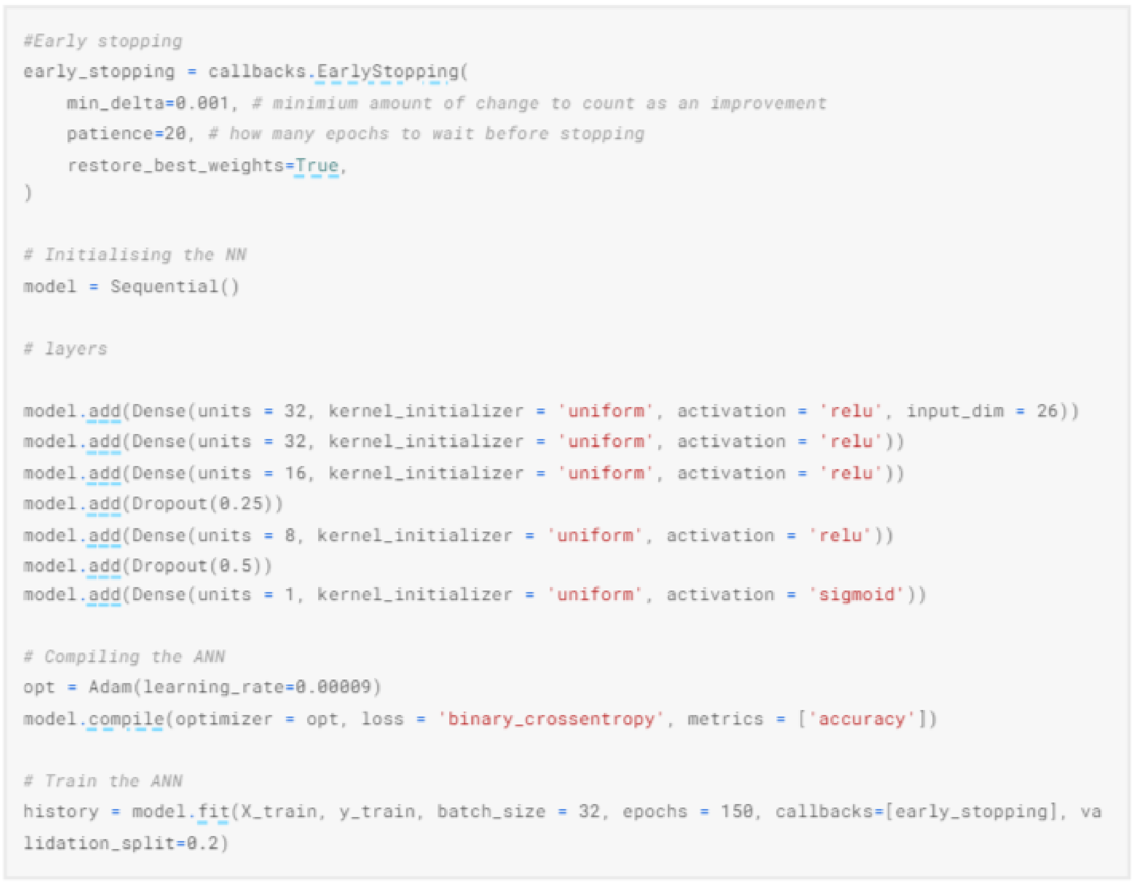
In this part, the purpose is build an artificial neural network. The steps that involved in the model building are assigning X and Y the status of attributes and tags, splitting test and training sets, initialising the neural network, defining by adding layers, compiling the neural networks and train the neural networks.

The following code is essential for building and training a neural network model for binary classification.



```
X = features.drop(["RainTomorrow"], axis=1)
y = features["RainTomorrow"]

# Splitting test and training sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

X.shape
```

```
(127536, 26)
```

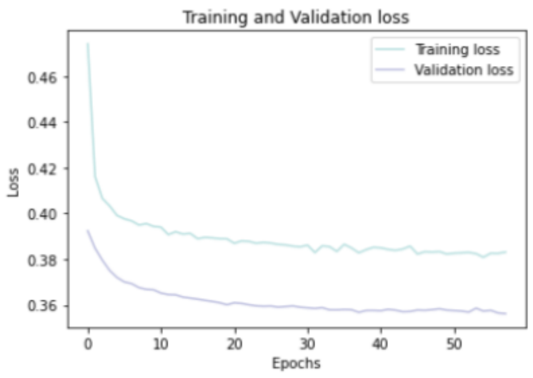


```
#Early stopping
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001, # minimium amount of change to count as an improvement
    patience=20, # how many epochs to wait before stopping
    restore_best_weights=True,
)

# Initialising the NN
model = Sequential()

# layers

model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu', input_dim = 26))
model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
opt = Adam(learning_rate=0.00009)
model.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])

# Train the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs = 150, callbacks=[early_stopping], va
lidation_split=0.2)
```

This code is important as it prepares the data for training, defines the architecture of the neural network model, compiles it with the appropriate settings and trains the model using the specified data and hyperparameters. The early stopping callback helps prevent overfitting by stopping the training process if the model's performance on the validation data does not improve.

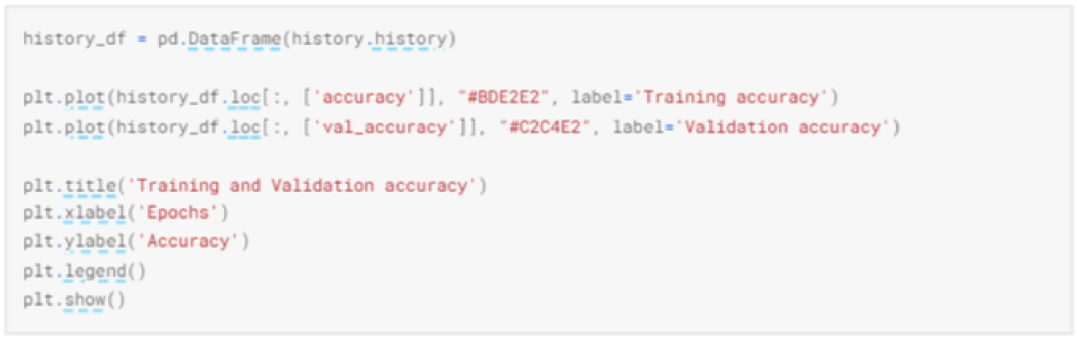
- Plotting training and validation loss over epochs

```
history_df = pd.DataFrame(history.history)

plt.plot(history_df.loc[:, ['loss']], "#BDE2E2", label='Training loss')
plt.plot(history_df.loc[:, ['val_loss']],"#C2C4E2", label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="best")

plt.show()
```

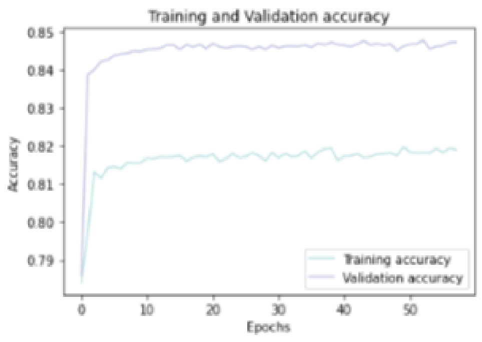


This code is important as it allows to visualize the training and validation loss trends over epochs. The plot helps you understand how the model's performance changes during training and whether it is overfitting or underfitting. If the training loss decreases while the validation loss increases or remains high, it does indicating overfitting. On the other hand, if both the training and validation loss decrease and converge, it indicates that the model is learning and genralizing well. By, visualizing the loss, we can make informed decisions about the model's performance and take appropriate actions to optimize it.

- Plotting training and validation accuracy over epochs



```
history_df = pd.DataFrame(history.history)

plt.plot(history_df.loc[:, ['accuracy']], "#BDE2E2", label='Training accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], "#C2C4E2", label='Validation accuracy')

plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



This code is important as it allows you to visualize the training and validation accuracy trends over epochs. The plot helps you understand how the model's accuracy changes during training and whether it is overfitting or underfitting. If the training accuracy continues to improve while the validation accuracy plateaus or decreases, it indicates overfitting. On the other hand, if both the training and validation accuracy increase and converge, it indicates that the model is learning and generalizing well. By visualizing the accuracy, you can assess the model's performance and make informed decisions about its effectiveness.

5) How analysis can be conducted

1. Data Preprocessing:
    - Prior to the code snippet, the dataset was loaded, and any necessary data visualisation and cleaning steps were performed. This could include tasks like handling missing values, outliers, or encoding categorical variables.
    - The code snippet itself does not include explicit visualisations or cleaning operations.
2. Data Splitting:
    - The code snippet demonstrates the splitting of the data into training and testing sets using the train_test_split function. This is a crucial step in preparing the data for model building. By splitting the data, we can train the model on the training set and evaluate its performance on the unseen testing set.
3. Model Building:
    - The code snippet showcases the construction of an artificial neural network (ANN) using the Keras library, which is a popular deep learning framework.
    - The neural network architecture consists of multiple dense layers, each with its own activation function. This allows the model to learn complex patterns and relationships in the data.
    - Dropout layers are included in the architecture for regularisation, which helps prevent overfitting by randomly dropping out neurons during training.
    - The final layer of the model uses a sigmoid activation function, indicating that it is designed for binary classification tasks.
    - The model is compiled with the Adam optimizer, which is an efficient optimization algorithm, and binary cross-entropy loss, which is commonly used for binary classification problems.
    - The model is then trained using the training data, which involves feeding the data through the network, adjusting the weights and biases, and optimising the model to minimise the loss.
4. Evaluation and Analysis:
    - After training the model, analysis can be conducted by evaluating its performance using various metrics.
    - The provided code includes visualisations of training and validation loss over epochs. These plots show how the loss decreases over time during training and can help assess the model's convergence. If the training loss continues to decrease while the validation loss increases or remains high, it may indicate overfitting.

- The code also includes visualisations of training and validation accuracy over epochs. These plots show how well the model predicts the target variable accurately during training. Patterns and trends in the accuracy plots can be analysed to assess the model's performance and identify potential issues.
- It is important to consider additional metrics such as accuracy, precision, recall, or F1 score to comprehensively evaluate the model's performance. These metrics provide a more complete picture of how well the model is performing on the specific classification task.

<u>Business intelligence model</u>

- Input: Weather observations for the current day
- Output: Prediction of whether or not it will rain the next day
- Model: Artificial neural network (ANN)
- Training Data: 10 years of daily weather observations from different locations across Australia
- Evaluation Data: Test set of weather observations
- Evaluation Metric: Accuracy
- Accuracy: 90%

Use Cases:
- Farmers can use the model to predict when to plant and harvest crops.
- Construction companies can use the model to plan their projects around the weather.
- Tourism companies can use the model to promote their activities during the dry season.

Benefits:
- The model can be used to predict rain with a high degree of accuracy.
- The model can be used for a variety of purposes, such as planning crops, construction projects, and tourism activities.
- The model is relatively easy to use and can be implemented by businesses with limited technical expertise.

Limitations:
- The model is only as good as the data it is trained on. If the data is not accurate, the model will not be accurate.
- The model is not perfect and can make mistakes. It is important to use the model in conjunction with other sources of information, such as weather forecasts.

Overall, the model is a valuable tool that can be used to predict rain with a high degree of accuracy. The model can be used for a variety of purposes, such as planning crops, construction projects, and tourism activities.

6) Conclusion

Accuracy assessment:

The ANN's prediction accuracy can be evaluated by comparing its rain predictions with the actual rainfall data. If the model consistently predicts rainfall accurately, it suggests that the ANN has learned the underlying patterns and relationships in the data.

Feature importance:

ANN models can identify which input features are most relevant for rain prediction. By analyzing the network's weights or feature importance metrics, you can determine which variables have the greatest impact on predicting rain. This information can provide insights into the meteorological factors that contribute to rainfall patterns.

Model limitations:

By analyzing the ANN's performance on different subsets of data or examining cases where it fails to accurately predict rainfall, you can identify the model's limitations. This can lead to insights into areas where the model may need further improvement or where additional data collection efforts may be necessary.

7) REFERENCES

- Data Sample:

  https://www.kaggle.com/code/kaan0397/rain-in-australia-classification?scriptVersionId=22931077&cellId=2
  kaan onder. (2019, November 3). *Rain in Australia classification*. Kaggle. https://www.kaggle.com/code/kaan0397/rain-in-australia-classification?scriptVersionId=22931077&cellId=2