In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
raw_df = pd.read_csv(r"E:\Data Science Projects\Project Files\Australia Rain\weatherAUS.csv\weatherAUS.csv")
```

In [3]:
```
raw_df.head()
```

Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9a
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	101
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	100
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	101
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	101

5 rows × 23 columns

In [4]:
```
raw_df.tail()
```

Out[4]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	56.0	21.0	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	53.0	24.0	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	51.0	24.0	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	NaN	ESE	...	62.0	36.0	

5 rows × 23 columns

In [5]:
```
raw_df.shape
```

Out[5]:
```
(145460, 23)
```

In [6]:
```
raw_df.info()
# here i have targeted to the Target Columns(Rain Today and Rain Tmrw).
#This 2 are Imp columns
```

```
<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
```

In [7]:
```
raw_df.dropna(subset=['RainToday','RainTomorrow'], inplace=True)
```

In [8]:
```
raw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140787 entries, 0 to 145458
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           140787 non-null  object 
 1   Location       140787 non-null  object 
 2   MinTemp        140319 non-null  float64
 3   MaxTemp        140480 non-null  float64
 4   Rainfall       140787 non-null  float64
 5   Evaporation    81093 non-null   float64
 6   Sunshine       73982 non-null   float64
 7   WindGustDir    131624 non-null  object 
 8   WindGustSpeed  131682 non-null  float64
 9   WindDir9am     131127 non-null  object 
 10  WindDir3pm     137117 non-null  object 
 11  WindSpeed9am   139732 non-null  float64
 12  WindSpeed3pm   138256 non-null  float64
 13  Humidity9am    139270 non-null  float64
 14  Humidity3pm    137286 non-null  float64
 15  Pressure9am    127044 non-null  float64
 16  Pressure3pm    127018 non-null  float64
 17  Cloud9am       88162 non-null   float64
 18  Cloud3pm       84693 non-null   float64
 19  Temp9am        140131 non-null  float64
 20  Temp3pm        138163 non-null  float64
 21  RainToday      140787 non-null  object 
 22  RainTomorrow   140787 non-null  object 
dtypes: float64(16), object(7)
memory usage: 25.8+ MB
```

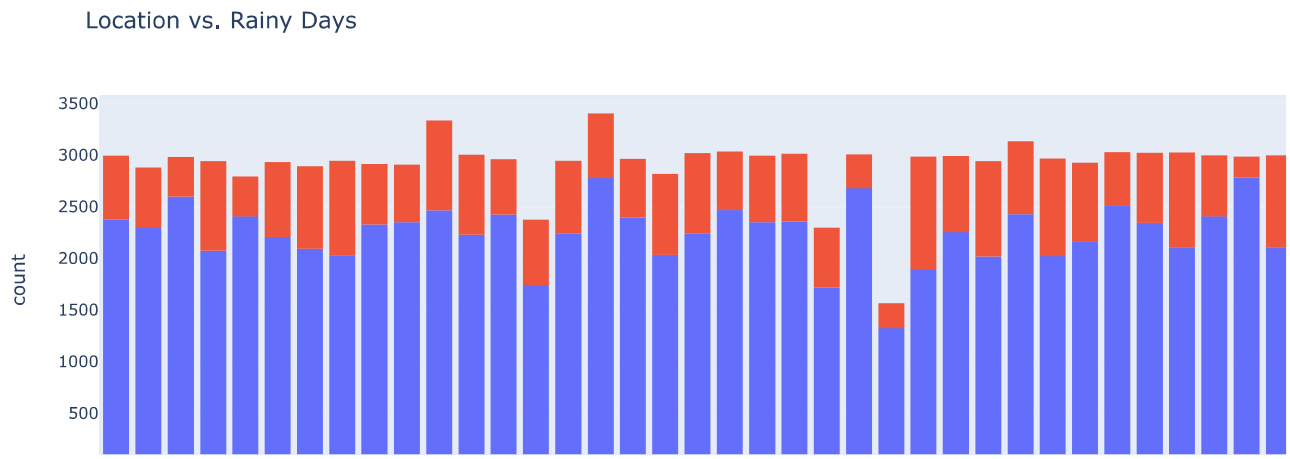
EDA - Exploratory Data Analysis and Visualization

In [9]:
```
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline



sns.set_style('darkgrid')
matplotlib.rcParams['font.size']=14
matplotlib.rcParams['figure.figsize']=(10,6)
matplotlib.rcParams['figure.facecolor']='#00000000'
```

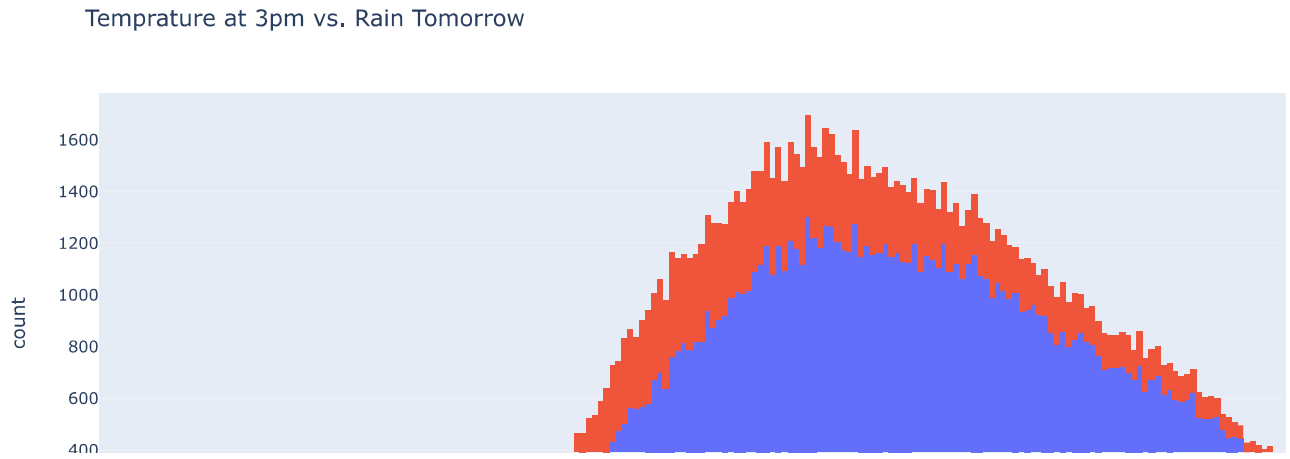
In [10]:
```
raw_df.Location.nunique()
```

Out[10]:
```
49
```

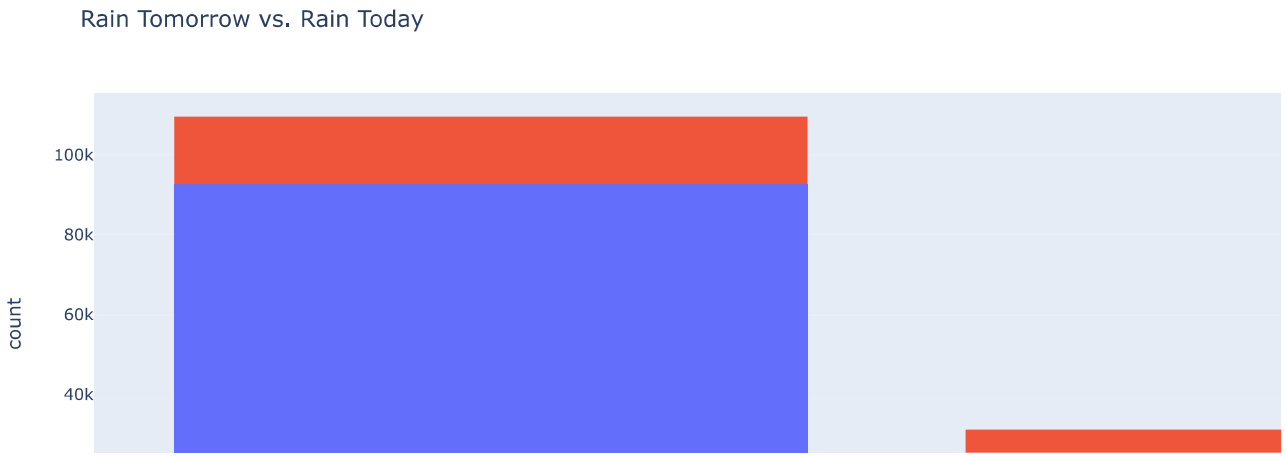
In [11]:
```
px.histogram(raw_df,x='Location', title='Location vs. Rainy Days', color = 'RainToday')
#in sns - we use hue instead of color
```



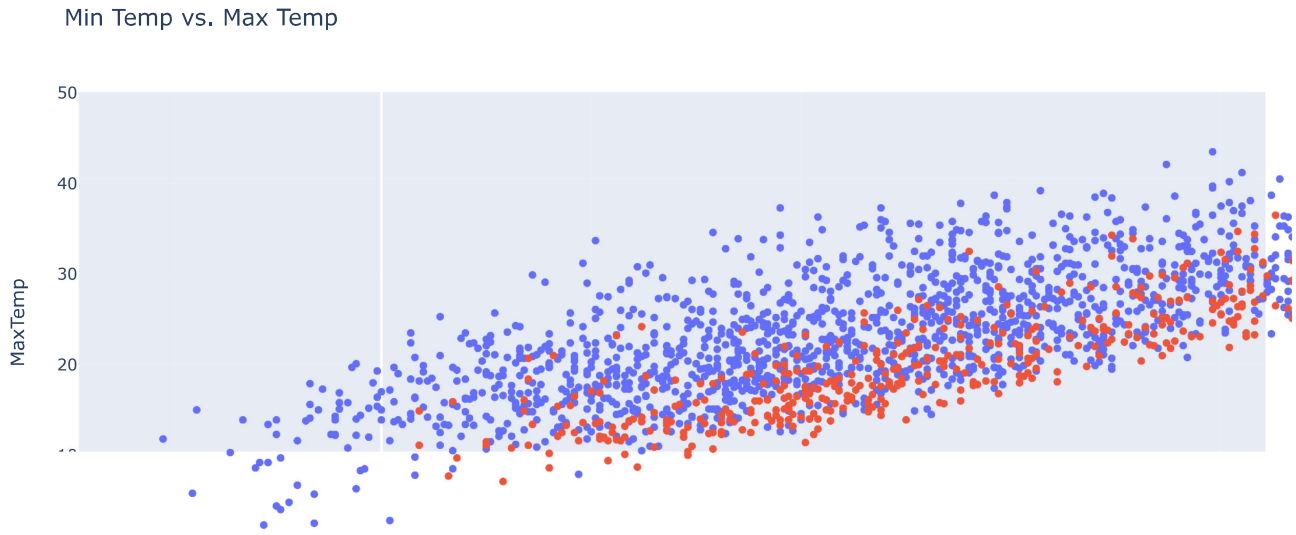
In [12]:
```
px.histogram(raw_df,x='Temp3pm',title='Temprature at 3pm vs. Rain Tomorrow',color='RainTomorrow')
```



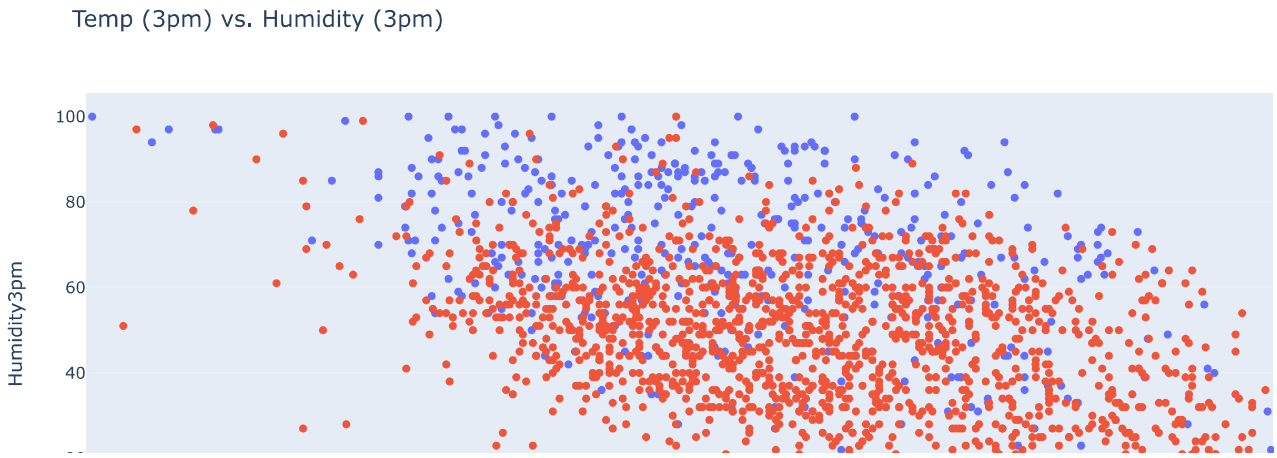
In [13]:
```
px.histogram(raw_df,x= 'RainTomorrow', title='Rain Tomorrow vs. Rain Today',color='RainToday')
```



In [14]:
```
px.scatter(raw_df.sample(2000),
           title='Min Temp vs. Max Temp',
           x ='MinTemp',
           y='MaxTemp',
           color='RainToday')
```



In [15]:
```
px.strip(raw_df.sample(2000),
        title="Temp (3pm) vs. Humidity (3pm)",
        x='Temp3pm',
        y='Humidity3pm',
        color='RainTomorrow')
```



In [ ]:

In [ ]:

In [ ]:

Training, Validation and Test Sets

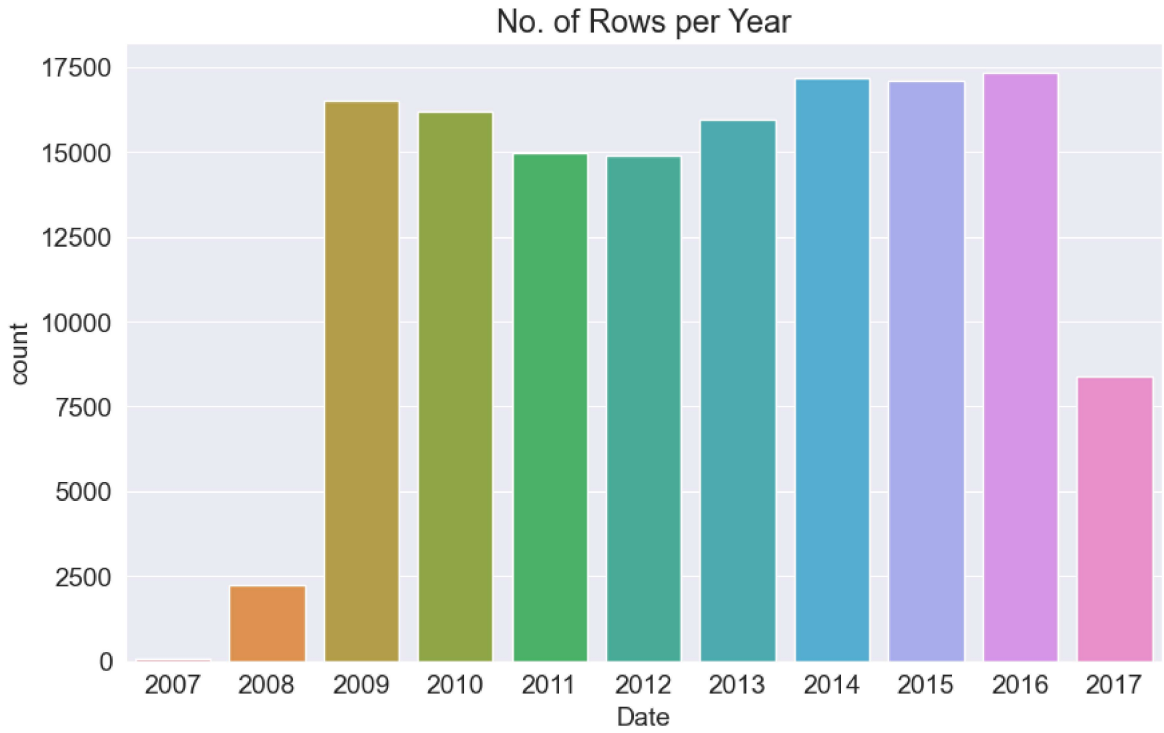
In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
train_val_df, test_df = train_test_split(raw_df,test_size=0.2, random_state=42)
train_df, val_df = train_test_split(train_val_df, test_size=0.25, random_state=42)
```

In [18]:
```
print('train_df.shape : ', train_df.shape)
print('val_df.shape : ', val_df.shape)
print('test_df.shape : ',test_df.shape)
```

```
train_df.shape :  (84471, 23)
val_df.shape :  (28158, 23)
test_df.shape :  (28158, 23)
```

In [19]:
```
plt.title('No. of Rows per Year')
sns.countplot(x=pd.to_datetime(raw_df.Date).dt.year)
plt.show()
```



In [20]:
```
year = pd.to_datetime(raw_df.Date).dt.year
train_df= raw_df[year<2015]
val_df=raw_df[year==2015]
test_df=raw_df[year>2015]
```

In [21]:
```
print('train_df.shape : ', train_df.shape)
print('val_df.shape : ', val_df.shape)
print('test_df.shape : ',test_df.shape)
```

```
train_df.shape :  (97988, 23)
val_df.shape :  (17089, 23)
test_df.shape :  (25710, 23)
```

In [22]:
```
train_df
```

Out[22]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
144548	2014-12-27	Uluru	16.9	33.2	0.0	NaN	NaN	SSE	43.0	ESE	...	22.0	13.0	
144549	2014-12-28	Uluru	15.1	36.8	0.0	NaN	NaN	NE	31.0	ENE	...	16.0	8.0	
144550	2014-12-29	Uluru	17.3	37.8	0.0	NaN	NaN	ESE	39.0	ESE	...	15.0	8.0	
144551	2014-12-30	Uluru	20.1	38.5	0.0	NaN	NaN	ESE	43.0	ESE	...	22.0	9.0	
144552	2014-12-31	Uluru	22.5	39.6	0.0	NaN	NaN	WNW	76.0	ENE	...	16.0	9.0	

97988 rows × 23 columns

In [23]:
```
val_df
```

Out[23]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
2133	2015-01-01	Albury	11.4	33.5	0.0	NaN	NaN	WSW	30.0	ESE	...	45.0	14.0	
2134	2015-01-02	Albury	15.5	39.6	0.0	NaN	NaN	NE	56.0	ESE	...	45.0	12.0	
2135	2015-01-03	Albury	17.1	38.3	0.0	NaN	NaN	NNE	48.0	NE	...	35.0	19.0	
2136	2015-01-04	Albury	26.0	33.1	0.0	NaN	NaN	NNE	41.0	ESE	...	46.0	37.0	
2137	2015-01-05	Albury	19.0	35.2	0.0	NaN	NaN	E	33.0	SSE	...	60.0	34.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
144913	2015-12-27	Uluru	20.5	34.7	0.0	NaN	NaN	E	52.0	ESE	...	23.0	12.0	
144914	2015-12-28	Uluru	18.0	36.4	0.0	NaN	NaN	ESE	54.0	E	...	17.0	7.0	
144915	2015-12-29	Uluru	17.5	37.1	0.0	NaN	NaN	E	56.0	E	...	12.0	7.0	
144916	2015-12-30	Uluru	20.0	38.9	0.0	NaN	NaN	E	59.0	E	...	12.0	12.0	
144917	2015-12-31	Uluru	19.3	37.4	0.0	NaN	NaN	SE	56.0	ESE	...	46.0	18.0	

17089 rows × 23 columns

In [24]:
```
test_df
```

Out[24]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
2498	2016-01-01	Albury	20.4	37.6	0.0	NaN	NaN	ENE	54.0	NaN	...	46.0	17.0	
2499	2016-01-02	Albury	20.9	33.6	0.4	NaN	NaN	SSE	50.0	SSE	...	54.0	30.0	
2500	2016-01-03	Albury	18.4	23.1	2.2	NaN	NaN	ENE	48.0	ESE	...	62.0	67.0	
2501	2016-01-04	Albury	17.3	23.7	15.6	NaN	NaN	SSE	39.0	SE	...	74.0	65.0	
2502	2016-01-05	Albury	15.5	22.9	6.8	NaN	NaN	ENE	31.0	SE	...	92.0	63.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145454	2017-06-20	Uluru	3.5	21.8	0.0	NaN	NaN	E	31.0	ESE	...	59.0	27.0	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	56.0	21.0	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	53.0	24.0	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	51.0	24.0	

25710 rows × 23 columns

Identifying Input and Target Columns

In [25]:
```
raw_df
```

Out[25]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145454	2017-06-20	Uluru	3.5	21.8	0.0	NaN	NaN	E	31.0	ESE	...	59.0	27.0	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	56.0	21.0	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	53.0	24.0	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	51.0	24.0	

140787 rows × 23 columns

In [26]:
```
input_cols = list(train_df.columns)[1:-1]
target_cols= 'RainTomorrow'
```

In [27]:
```
print(input_cols)
```

```
['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday']
```

In [28]:
```
target_cols
```

Out[28]:
```
'RainTomorrow'
```

We can now create input and targets for the training, validation and test sets for further processing and model training

In [29]:
```
train_inputs = train_df[input_cols].copy()
train_targets = train_df[target_cols].copy()
```

In [30]:
```
val_inputs = val_df[input_cols].copy()
val_targets = val_df[target_cols].copy()
```

In [31]:
```
test_inputs = test_df[input_cols].copy()
test_targets = test_df[target_cols].copy()
```

In [32]:
```
train_inputs
```

Out[32]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	WindSpeed3pm	Humidity9
0	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	...	24.0	7
1	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	...	22.0	4
2	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	...	26.0	3
3	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	...	9.0	4
4	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	...	20.0	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...
144548	Uluru	16.9	33.2	0.0	NaN	NaN	SSE	43.0	ESE	SSE	...	26.0	2
144549	Uluru	15.1	36.8	0.0	NaN	NaN	NE	31.0	ENE	SW	...	20.0	1
144550	Uluru	17.3	37.8	0.0	NaN	NaN	ESE	39.0	ESE	SSE	...	9.0	1
144551	Uluru	20.1	38.5	0.0	NaN	NaN	ESE	43.0	ESE	SSW	...	17.0	2
144552	Uluru	22.5	39.6	0.0	NaN	NaN	WNW	76.0	ENE	SSW	...	13.0	1

97988 rows × 21 columns

In [33]:
```
val_inputs
```

Out[33]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	WindSpeed3pm	Humidity9
2133	Albury	11.4	33.5	0.0	NaN	NaN	WSW	30.0	ESE	W	...	11.0	4
2134	Albury	15.5	39.6	0.0	NaN	NaN	NE	56.0	ESE	ESE	...	9.0	4
2135	Albury	17.1	38.3	0.0	NaN	NaN	NNE	48.0	NE	N	...	20.0	3
2136	Albury	26.0	33.1	0.0	NaN	NaN	NNE	41.0	ESE	W	...	7.0	4
2137	Albury	19.0	35.2	0.0	NaN	NaN	E	33.0	SSE	SE	...	9.0	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
144913	Uluru	20.5	34.7	0.0	NaN	NaN	E	52.0	ESE	E	...	20.0	2
144914	Uluru	18.0	36.4	0.0	NaN	NaN	ESE	54.0	E	ESE	...	31.0	1
144915	Uluru	17.5	37.1	0.0	NaN	NaN	E	56.0	E	SE	...	22.0	1
144916	Uluru	20.0	38.9	0.0	NaN	NaN	E	59.0	E	SSE	...	17.0	1
144917	Uluru	19.3	37.4	0.0	NaN	NaN	SE	56.0	ESE	S	...	28.0	4

17089 rows × 21 columns

In [34]:
```
test_inputs
```

Out[34]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	WindSpeed3pm	Humidity9
2498	Albury	20.4	37.6	0.0	NaN	NaN	ENE	54.0	NaN	ESE	...	7.0	4
2499	Albury	20.9	33.6	0.4	NaN	NaN	SSE	50.0	SSE	SE	...	17.0	5
2500	Albury	18.4	23.1	2.2	NaN	NaN	ENE	48.0	ESE	ENE	...	39.0	6
2501	Albury	17.3	23.7	15.6	NaN	NaN	SSE	39.0	SE	SSE	...	17.0	7
2502	Albury	15.5	22.9	6.8	NaN	NaN	ENE	31.0	SE	SSE	...	9.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...
145454	Uluru	3.5	21.8	0.0	NaN	NaN	E	31.0	ESE	E	...	13.0	5
145455	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	ENE	...	11.0	5
145456	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	N	...	9.0	5
145457	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	WNW	...	9.0	5
145458	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	N	...	7.0	5

25710 rows × 21 columns

In [35]:
```
train_targets
```

Out[35]:
```
0         No
1         No
2         No
3         No
4         No
         ..
144548    No
144549    No
144550    No
144551    No
144552    No
Name: RainTomorrow, Length: 97988, dtype: object
```

In [36]:
```
test_targets
```

Out[36]:
```
2498       No
2499      Yes
2500      Yes
2501      Yes
2502       No
         ...
145454     No
145455     No
145456     No
145457     No
145458     No
Name: RainTomorrow, Length: 25710, dtype: object
```

In [37]:
```
val_targets
```

Out[37]:
```
2133      No
2134      No
2135      No
2136      No
2137      No
          ..
144913    No
144914    No
144915    No
144916    No
144917    No
Name: RainTomorrow, Length: 17089, dtype: object
```

Let's also identify which of the columns are numerical and which ones are categorical.

In [38]:
```
numerical_cols= train_inputs.select_dtypes(include=np.number).columns.tolist()
categorical_cols = train_inputs.select_dtypes('object').columns.tolist()
```

In [39]:
```
print(numerical_cols)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3p
m', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm']
```

In [40]:
```
print(categorical_cols)
```

```
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday']
```

In [41]:
```
train_inputs[numerical_cols].describe()
```

Out[41]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pres
count	97674.000000	97801.000000	97988.000000	61657.000000	57942.000000	91160.000000	97114.000000	96919.000000	96936.000000	96872.000000	8887
mean	12.007831	23.022202	2.372935	5.289991	7.609004	40.215873	14.092263	18.764608	68.628745	51.469547	101
std	6.347175	6.984397	8.518819	3.952010	3.788813	13.697967	8.984203	8.872398	19.003097	20.756113	
min	-8.500000	-4.100000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	98
25%	7.500000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000	37.000000	101
50%	11.800000	22.400000	0.000000	4.600000	8.500000	39.000000	13.000000	19.000000	70.000000	52.000000	101
75%	16.600000	27.900000	0.800000	7.200000	10.600000	48.000000	19.000000	24.000000	83.000000	66.000000	102
max	33.900000	48.100000	371.000000	82.400000	14.300000	135.000000	87.000000	87.000000	100.000000	100.000000	104

In [42]:
```
train_inputs[categorical_cols].describe()
```

Out[42]:

	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday
count	97988	91120	90969	96036	97988
unique	49	16	16	16	2
top	Canberra	W	N	SE	No
freq	2506	6672	8012	7603	76002

In [43]:
```
train_inputs[categorical_cols].nunique()
```

Out[43]:
```
Location       49
WindGustDir    16
WindDir9am     16
WindDir3pm     16
RainToday       2
dtype: int64
```

Imputing Missing Numeric Data

In [44]:
```
from sklearn.impute import SimpleImputer
```

In [45]:
```
?SimpleImputer
```

In [46]:
```
imputer = SimpleImputer(strategy='mean')
```

Before we perform imputation, let's check the no. of missing values in each numeric columns

In [47]:
```
train_inputs[numerical_cols].isnull().sum().sort_values()
```

Out[47]:
```
Rainfall            0
MaxTemp           187
MinTemp           314
Temp9am           574
Temp3pm           596
WindSpeed9am      874
Humidity9am      1052
WindSpeed3pm     1069
Humidity3pm      1116
WindGustSpeed    6828
Pressure9am      9112
Pressure3pm      9131
Cloud9am        34988
Cloud3pm        36022
Evaporation     36331
Sunshine        40046
dtype: int64
```

In [48]:
```
raw_df.shape
```

Out[48]:
```
(140787, 23)
```

The 1st step in imputation is to fit the imputer to data

In [49]:
```
imputer.fit(raw_df[numerical_cols])
```

Out[49]:
```
SimpleImputer
SimpleImputer()
```

In [50]:
```
list(imputer.statistics_)
```

Out[50]:
```
[12.18482386562048,
 23.235120301822324,
 2.349974074310839,
 5.472515506887154,
 7.630539861047281,
 39.97051988882308,
 13.990496092519967,
 18.631140782316862,
 68.82683277087672,
 51.44928834695453,
 1017.6545771543717,
 1015.2579625879797,
 4.431160817585808,
 4.499250233195188,
 16.98706638787991,
 21.69318269001107]
```

The missing values in the training, test and validation sets can now be filled in using the transform method of imputer.

In [51]:
```
train_inputs[numerical_cols] = imputer.transform(train_inputs[numerical_cols])
val_inputs[numerical_cols] = imputer.transform(val_inputs[numerical_cols])
test_inputs[numerical_cols] = imputer.transform(test_inputs[numerical_cols])
```

The missing values are now filled in with the mean of each column.

In [52]:
```
train_inputs[numerical_cols].isnull().sum()
```

Out[52]:
```
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustSpeed    0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
dtype: int64
```

In [53]:
```
imputer_cat = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
```

In [54]:
```
imputer_cat.fit(raw_df[categorical_cols])
```

Out[54]:
```
SimpleImputer
SimpleImputer(strategy='most_frequent')
```

In [55]:
```
list(imputer_cat.statistics_)
```

Out[55]:
```
['Canberra', 'W', 'N', 'SE', 'No']
```

In [56]:
```
train_inputs[categorical_cols] = imputer_cat.transform(train_inputs[categorical_cols])
val_inputs[categorical_cols] = imputer_cat.transform(val_inputs[categorical_cols])
test_inputs[categorical_cols] = imputer_cat.transform(test_inputs[categorical_cols])
```

In [57]:
```
train_inputs[categorical_cols].isnull().sum()
```

Out[57]:
```
Location       0
WindGustDir    0
WindDir9am     0
WindDir3pm     0
RainToday      0
dtype: int64
```

Scaling Numeric Features

Another good practice to scale numeric features to a small range of values e.g. (0,1) or (-1,1). Scaling numeric features ensure that no particular feature has a disproportionate impact on the model's loss.

In [58]:
```
raw_df[numerical_cols].describe()
```

Out[58]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	1
count	140319.000000	140480.00000	140787.000000	81093.000000	73982.000000	131682.000000	139732.000000	138256.000000	139270.000000	137286.000000	
mean	12.184824	23.23512	2.349974	5.472516	7.630540	39.970520	13.990496	18.631141	68.826833	51.449288	
std	6.403879	7.11450	8.465173	4.189132	3.781729	13.578201	8.886210	8.798096	19.063650	20.807310	
min	-8.500000	-4.80000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	
25%	7.600000	17.90000	0.000000	2.600000	4.900000	31.000000	7.000000	13.000000	57.000000	37.000000	
50%	12.000000	22.60000	0.000000	4.800000	8.500000	39.000000	13.000000	19.000000	70.000000	52.000000	
75%	16.800000	28.30000	0.800000	7.400000	10.700000	48.000000	19.000000	24.000000	83.000000	66.000000	
max	33.900000	48.10000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.000000	

Let's use MinMaxScaler from sklearn.preprocessing to scale values to the (0,1) range.

In [59]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [60]:
```
?MinMaxScaler
```

In [61]:
```
scaler= MinMaxScaler()
```

First, I fit the scaler to the data i.e. compute the range of values for each numeric column.

In [62]:
```
scaler.fit(raw_df[numerical_cols])
```

Out[62]:
```
▾ MinMaxScaler
MinMaxScaler()
```

We can now inspect the minimum and maximum values in each column

In [63]:
```
print('Minimum : ')
list(scaler.data_min_)
```

Out[63]:
```
Minimum :
[-8.5,
 -4.8,
 0.0,
 0.0,
 0.0,
 6.0,
 0.0,
 0.0,
 0.0,
 0.0,
 980.5,
 977.1,
 0.0,
 0.0,
 -7.2,
 -5.4]
```

In [64]:
```
print('Maximum : ')
list(scaler.data_max_)
```

Out[64]:
```
Maximum :
[33.9,
 48.1,
 371.0,
 145.0,
 14.5,
 135.0,
 130.0,
 87.0,
 100.0,
 100.0,
 1041.0,
 1039.6,
 9.0,
 9.0,
 40.2,
 46.7]
```

We can now separately scale the training,validation and test sets using the transform method of scaler.

In [65]:
```
train_inputs[numerical_cols] = scaler.transform(train_inputs[numerical_cols])
val_inputs[numerical_cols] = scaler.transform(val_inputs[numerical_cols])
test_inputs[numerical_cols] = scaler.transform(test_inputs[numerical_cols])
```

In [66]:
```
train_inputs[numerical_cols]
```

Out[66]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pres
0	0.516509	0.523629	0.001617	0.037741	0.526244	0.294574	0.153846	0.275862	0.71	0.22	0.449587	
1	0.375000	0.565217	0.000000	0.037741	0.526244	0.294574	0.030769	0.252874	0.44	0.25	0.497521	
2	0.504717	0.576560	0.000000	0.037741	0.526244	0.310078	0.146154	0.298851	0.38	0.30	0.447934	
3	0.417453	0.620038	0.000000	0.037741	0.526244	0.139535	0.084615	0.103448	0.45	0.16	0.613223	
4	0.613208	0.701323	0.002695	0.037741	0.526244	0.271318	0.053846	0.229885	0.82	0.33	0.500826	
...	...	...	...	...	...	...	...	...	...	...	...	...
144548	0.599057	0.718336	0.000000	0.037741	0.526244	0.286822	0.184615	0.298851	0.22	0.13	0.555372	
144549	0.556604	0.786389	0.000000	0.037741	0.526244	0.193798	0.146154	0.229885	0.16	0.08	0.530579	
144550	0.608491	0.805293	0.000000	0.037741	0.526244	0.255814	0.200000	0.103448	0.15	0.08	0.519008	
144551	0.674528	0.818526	0.000000	0.037741	0.526244	0.286822	0.215385	0.195402	0.22	0.09	0.553719	
144552	0.731132	0.839319	0.000000	0.037741	0.526244	0.542636	0.230769	0.149425	0.16	0.09	0.522314	

97988 rows × 16 columns

In [67]:
```
val_inputs[numerical_cols]
```

Out[67]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Press
2133	0.469340	0.724008	0.0	0.037741	0.526244	0.186047	0.053846	0.126437	0.45	0.14	0.545455	
2134	0.566038	0.839319	0.0	0.037741	0.526244	0.387597	0.069231	0.103448	0.45	0.12	0.586777	
2135	0.603774	0.814745	0.0	0.037741	0.526244	0.325581	0.153846	0.229885	0.35	0.19	0.618182	
2136	0.813679	0.716446	0.0	0.037741	0.526244	0.271318	0.053846	0.080460	0.46	0.37	0.547107	
2137	0.648585	0.756144	0.0	0.037741	0.526244	0.209302	0.053846	0.103448	0.60	0.34	0.609917	
...	...	...	...	...	...	...	...	...	...	...	...	...
144913	0.683962	0.746692	0.0	0.037741	0.526244	0.356589	0.269231	0.229885	0.23	0.12	0.540496	
144914	0.625000	0.778828	0.0	0.037741	0.526244	0.372093	0.230769	0.356322	0.17	0.07	0.565289	
144915	0.613208	0.792060	0.0	0.037741	0.526244	0.387597	0.253846	0.252874	0.12	0.07	0.530579	
144916	0.672170	0.826087	0.0	0.037741	0.526244	0.410853	0.153846	0.195402	0.12	0.12	0.441322	
144917	0.655660	0.797732	0.0	0.037741	0.526244	0.387597	0.153846	0.321839	0.46	0.18	0.442975	

17089 rows × 16 columns

In [68]:
```
test_inputs[numerical_cols]
```

Out[68]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pres
2498	0.681604	0.801512	0.000000	0.037741	0.526244	0.372093	0.000000	0.080460	0.46	0.17	0.543802	
2499	0.693396	0.725898	0.001078	0.037741	0.526244	0.341085	0.069231	0.195402	0.54	0.30	0.505785	
2500	0.634434	0.527410	0.005930	0.037741	0.526244	0.325581	0.084615	0.448276	0.62	0.67	0.553719	
2501	0.608491	0.538752	0.042049	0.037741	0.526244	0.255814	0.069231	0.195402	0.74	0.65	0.618182	
2502	0.566038	0.523629	0.018329	0.037741	0.526244	0.193798	0.046154	0.103448	0.92	0.63	0.591736	
...	...	...	...	...	...	...	...	...	...	...	...	...
145454	0.283019	0.502836	0.000000	0.037741	0.526244	0.193798	0.115385	0.149425	0.59	0.27	0.730579	
145455	0.266509	0.533081	0.000000	0.037741	0.526244	0.193798	0.100000	0.126437	0.51	0.24	0.728926	
145456	0.285377	0.568998	0.000000	0.037741	0.526244	0.124031	0.100000	0.103448	0.56	0.21	0.710744	
145457	0.327830	0.599244	0.000000	0.037741	0.526244	0.240310	0.069231	0.103448	0.53	0.24	0.669421	
145458	0.384434	0.601134	0.000000	0.037741	0.526244	0.170543	0.100000	0.080460	0.51	0.24	0.642975	

25710 rows × 16 columns

We can now varify that values in each column lie in the range of (0,1).

In [69]:
```
train_inputs[numerical_cols].describe()
```

Out[69]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pres
count	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	97988.000000	9798
mean	0.483689	0.525947	0.006396	0.036949	0.525366	0.265107	0.108395	0.215668	0.686309	0.514693	
std	0.149458	0.131904	0.022962	0.021628	0.200931	0.102420	0.068800	0.101424	0.189008	0.206376	
min	0.000000	0.013233	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.377358	0.429112	0.000000	0.026207	0.517241	0.193798	0.053846	0.149425	0.570000	0.370000	
50%	0.478774	0.514178	0.000000	0.037741	0.526244	0.255814	0.100000	0.218391	0.690000	0.520000	
75%	0.591981	0.618147	0.002156	0.038621	0.634483	0.310078	0.146154	0.275862	0.830000	0.650000	
max	1.000000	1.000000	1.000000	0.568276	0.986207	1.000000	0.669231	1.000000	1.000000	1.000000	

Encoding Categorical Data

In [70]:
```
raw_df[categorical_cols].nunique()
```

Out[70]:
```
Location       49
WindGustDir    16
WindDir9am     16
WindDir3pm     16
RainToday       2
dtype: int64
```

We can perform one hot encoding using the OneHotEncoder class from sklearn.preprocessing

In [71]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [72]:
```
?OneHotEncoder
```

In [73]:
```
encoder = OneHotEncoder(sparse = False, handle_unknown='ignore')
```

First, we fit the encoder to the data i.e. identify the full list of categories across all categorical columns.

In [74]:
```
raw_df2=raw_df[categorical_cols].fillna('Unknown')
```

In [75]:
```
encoder.fit(raw_df2)
```

Out[75]:
```
OneHotEncoder(handle_unknown='ignore', sparse=False, sparse_output=False)
```

In [76]:
```
categorical_cols
```

Out[76]:
```
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday']
```

In [77]:
```
encoder.categories_
```

Out[77]:
```
[array(['Adelaide', 'Albany', 'Albury', 'AliceSprings', 'BadgerysCreek',
        'Ballarat', 'Bendigo', 'Brisbane', 'Cairns', 'Canberra', 'Cobar',
        'CoffsHarbour', 'Dartmoor', 'Darwin', 'GoldCoast', 'Hobart',
        'Katherine', 'Launceston', 'Melbourne', 'MelbourneAirport',
        'Mildura', 'Moree', 'MountGambier', 'MountGinini', 'Newcastle',
        'Nhil', 'NorahHead', 'NorfolkIsland', 'Nuriootpa', 'PearceRAAF',
        'Penrith', 'Perth', 'PerthAirport', 'Portland', 'Richmond', 'Sale',
        'SalmonGums', 'Sydney', 'SydneyAirport', 'Townsville',
        'Tuggeranong', 'Uluru', 'WaggaWagga', 'Walpole', 'Watsonia',
        'Williamtown', 'Witchcliffe', 'Wollongong', 'Woomera'],
       dtype=object),
 array(['E', 'ENE', 'ESE', 'N', 'NE', 'NNE', 'NNW', 'NW', 'S', 'SE', 'SSE',
        'SSW', 'SW', 'Unknown', 'W', 'WNW', 'WSW'], dtype=object),
 array(['E', 'ENE', 'ESE', 'N', 'NE', 'NNE', 'NNW', 'NW', 'S', 'SE', 'SSE',
        'SSW', 'SW', 'Unknown', 'W', 'WNW', 'WSW'], dtype=object),
 array(['E', 'ENE', 'ESE', 'N', 'NE', 'NNE', 'NNW', 'NW', 'S', 'SE', 'SSE',
        'SSW', 'SW', 'Unknown', 'W', 'WNW', 'WSW'], dtype=object),
 array(['No', 'Yes'], dtype=object)]
```

In [78]:
```
encoded_cols = list(encoder.get_feature_names_out(categorical_cols))
print(encoded_cols)
```

```
['Location_Adelaide', 'Location_Albany', 'Location_Albury', 'Location_AliceSprings', 'Location_BadgerysCreek', 'Location_Ballarat', 'Loca
tion_Bendigo', 'Location_Brisbane', 'Location_Cairns', 'Location_Canberra', 'Location_Cobar', 'Location_CoffsHarbour', 'Location_Dartmoo
r', 'Location_Darwin', 'Location_GoldCoast', 'Location_Hobart', 'Location_Katherine', 'Location_Launceston', 'Location_Melbourne', 'Locat
ion_MelbourneAirport', 'Location_Mildura', 'Location_Moree', 'Location_MountGambier', 'Location_MountGinini', 'Location_Newcastle', 'Loca
tion_Nhil', 'Location_NorahHead', 'Location_NorfolkIsland', 'Location_Nuriootpa', 'Location_PearceRAAF', 'Location_Penrith', 'Location_Pe
rth', 'Location_PerthAirport', 'Location_Portland', 'Location_Richmond', 'Location_Sale', 'Location_SalmonGums', 'Location_Sydney', 'Loca
tion_SydneyAirport', 'Location_Townsville', 'Location_Tuggeranong', 'Location_Uluru', 'Location_WaggaWagga', 'Location_Walpole', 'Locatio
n_Watsonia', 'Location_Williamtown', 'Location_Witchcliffe', 'Location_Wollongong', 'Location_Woomera', 'WindGustDir_E', 'WindGustDir_EN
E', 'WindGustDir_ESE', 'WindGustDir_N', 'WindGustDir_NE', 'WindGustDir_NNE', 'WindGustDir_NNW', 'WindGustDir_NW', 'WindGustDir_S', 'WindG
ustDir_SE', 'WindGustDir_SSE', 'WindGustDir_SSW', 'WindGustDir_SW', 'WindGustDir_Unknown', 'WindGustDir_W', 'WindGustDir_WNW', 'WindGustD
ir_WSW', 'WindDir9am_E', 'WindDir9am_ENE', 'WindDir9am_ESE', 'WindDir9am_N', 'WindDir9am_NE', 'WindDir9am_NNE', 'WindDir9am_NNW', 'WindDi
r9am_NW', 'WindDir9am_S', 'WindDir9am_SE', 'WindDir9am_SSE', 'WindDir9am_SSW', 'WindDir9am_SW', 'WindDir9am_Unknown', 'WindDir9am_W', 'Wi
ndDir9am_WNW', 'WindDir9am_WSW', 'WindDir3pm_E', 'WindDir3pm_ENE', 'WindDir3pm_ESE', 'WindDir3pm_N', 'WindDir3pm_NE', 'WindDir3pm_NNE',
'WindDir3pm_NNW', 'WindDir3pm_NW', 'WindDir3pm_S', 'WindDir3pm_SE', 'WindDir3pm_SSE', 'WindDir3pm_SSW', 'WindDir3pm_SW', 'WindDir3pm_Unkn
own', 'WindDir3pm_W', 'WindDir3pm_WNW', 'WindDir3pm_WSW', 'RainToday_No', 'RainToday_Yes']
```

In [79]:
```
train_inputs[encoded_cols] = encoder.transform(train_inputs[categorical_cols])
val_inputs[encoded_cols] = encoder.transform(val_inputs[categorical_cols])
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

In [80]:
```
train_inputs[encoded_cols]
```

Out[80]:

	Location_Adelaide	Location_Albany	Location_Albury	Location_AliceSprings	Location_BadgerysCreek	Location_Ballarat	Location_Bendigo	Location_Brisbane
0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.(
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.(
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.(
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.(
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.(
...	...	...	...	...	...	...	...	..
144548	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
144549	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
144550	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
144551	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
144552	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(

97988 rows × 102 columns

In [81]:
```
val_inputs[encoded_cols]
```

Out[81]:

	Location_Adelaide	Location_Albany	Location_Albury	Location_AliceSprings	Location_BadgerysCreek	Location_Ballarat	Location_Bendigo	Location_Brisbane
2133	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2134	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2135	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2136	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2137	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
144913	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
144914	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
144915	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
144916	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
144917	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

17089 rows × 102 columns

In [82]:
```
test_inputs[encoded_cols]
```

Out[82]:

	Location_Adelaide	Location_Albany	Location_Albury	Location_AliceSprings	Location_BadgerysCreek	Location_Ballarat	Location_Bendigo	Location_Brisbane
2498	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2499	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2500	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2501	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2502	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
145454	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145455	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145456	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145457	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145458	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

25710 rows × 102 columns

In [83]:
```
pd.set_option('Display.max_columns', None)
```

In [84]:
```
test_inputs
```

Out[84]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3p
2498	Albury	0.681604	0.801512	0.000000	0.037741	0.526244	ENE	0.372093	N	ESE	0.000000	0.0804
2499	Albury	0.693396	0.725898	0.001078	0.037741	0.526244	SSE	0.341085	SSE	SE	0.069231	0.1954
2500	Albury	0.634434	0.527410	0.005930	0.037741	0.526244	ENE	0.325581	ESE	ENE	0.084615	0.4482
2501	Albury	0.608491	0.538752	0.042049	0.037741	0.526244	SSE	0.255814	SE	SSE	0.069231	0.1954
2502	Albury	0.566038	0.523629	0.018329	0.037741	0.526244	ENE	0.193798	SE	SSE	0.046154	0.1034
...	...	...	...	...	...	...	...	...	...	...	...	...
145454	Uluru	0.283019	0.502836	0.000000	0.037741	0.526244	E	0.193798	ESE	E	0.115385	0.1494
145455	Uluru	0.266509	0.533081	0.000000	0.037741	0.526244	E	0.193798	SE	ENE	0.100000	0.1264
145456	Uluru	0.285377	0.568998	0.000000	0.037741	0.526244	NNW	0.124031	SE	N	0.100000	0.1034
145457	Uluru	0.327830	0.599244	0.000000	0.037741	0.526244	N	0.240310	SE	WNW	0.069231	0.1034
145458	Uluru	0.384434	0.601134	0.000000	0.037741	0.526244	SE	0.170543	SSE	N	0.100000	0.0804

25710 rows × 123 columns

In [85]:
```
print('train_inputs : ',train_inputs.shape)
print('train_targets : ',train_targets.shape)
print('val_inputs : ',val_inputs.shape)
print('val_target : ',val_targets.shape)
print('test_inputs : ',test_inputs.shape)
print('test_target : ',test_targets.shape)
```

```
train_inputs :  (97988, 123)
train_targets :  (97988,)
val_inputs :  (17089, 123)
val_target :  (17089,)
test_inputs :  (25710, 123)
test_target :  (25710,)
```

Parquet

In [86]:
```
train_inputs.to_parquet('train_inputs.parquet')
val_inputs.to_parquet('val_inputs.parquet')
test_inputs.to_parquet('test_inputs.parquet')
```

In [87]:
```
%%time
pd.DataFrame(train_targets).to_parquet('train_targets.parquet')
pd.DataFrame(val_targets).to_parquet('val_targets.parquet')
pd.DataFrame(test_targets).to_parquet('test_targets.parquet')
```

```
CPU times: total: 46.9 ms
Wall time: 205 ms
```

We can read the data back using pd.read_parquet

In [88]:
```
%%time
train_inputs = pd.read_parquet('train_inputs.parquet')
val_inputs = pd.read_parquet('val_inputs.parquet')
test_inputs = pd.read_parquet('test_inputs.parquet')
```

```
CPU times: total: 1.66 s
Wall time: 2.28 s
```

In [89]:
```
%%time
train_targets = pd.read_parquet('train_targets.parquet')[target_cols]
val_targets = pd.read_parquet('val_targets.parquet')[target_cols]
test_targets = pd.read_parquet('test_targets.parquet')[target_cols]
```

```
CPU times: total: 109 ms
Wall time: 475 ms
```

Lets verify that Data was loaded properly

In [90]:
```
print('train_inputs : ',train_inputs.shape)
print('train_targets : ',train_targets.shape)
print('val_inputs : ',val_inputs.shape)
print('val_targets : ',val_targets.shape)
print('test_inputs : ',test_inputs.shape)
print('test_targets : ',test_targets.shape)
```

```
train_inputs :  (97988, 123)
train_targets :  (97988,)
val_inputs :  (17089, 123)
val_targets :  (17089,)
test_inputs :  (25710, 123)
test_targets :  (25710,)
```

Training a Logistic Regression Model

In [91]:
```
from sklearn.linear_model import LogisticRegression
```

In [92]:
```
?LogisticRegression
```

In [93]:
```
model=LogisticRegression(solver='liblinear')
```

We can train the model using model.fit

In [94]:
```
model.fit(train_inputs[numerical_cols + encoded_cols],train_targets)
```

Out[94]:
```
LogisticRegression
LogisticRegression(solver='liblinear')
```

Let's check the weights and biases of the trained model.

In [95]:
```
print(numerical_cols + encoded_cols)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3p
m', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'Location_Adelaide', 'Location_Albany', 'Location_Albur
y', 'Location_AliceSprings', 'Location_BadgerysCreek', 'Location_Ballarat', 'Location_Bendigo', 'Location_Brisbane', 'Location_Cairns',
'Location_Canberra', 'Location_Cobar', 'Location_CoffsHarbour', 'Location_Dartmoor', 'Location_Darwin', 'Location_GoldCoast', 'Location_H
obart', 'Location_Katherine', 'Location_Launceston', 'Location_Melbourne', 'Location_MelbourneAirport', 'Location_Mildura', 'Location_Mor
ee', 'Location_MountGambier', 'Location_MountGinini', 'Location_Newcastle', 'Location_Nhil', 'Location_NorahHead', 'Location_NorfolkIslan
d', 'Location_Nuriootpa', 'Location_PearceRAAF', 'Location_Penrith', 'Location_Perth', 'Location_PerthAirport', 'Location_Portland', 'Loc
ation_Richmond', 'Location_Sale', 'Location_SalmonGums', 'Location_Sydney', 'Location_SydneyAirport', 'Location_Townsville', 'Location_Tu
ggeranong', 'Location_Uluru', 'Location_WaggaWagga', 'Location_Walpole', 'Location_Watsonia', 'Location_Williamtown', 'Location_Witchclif
fe', 'Location_Wollongong', 'Location_Woomera', 'WindGustDir_E', 'WindGustDir_ENE', 'WindGustDir_ESE', 'WindGustDir_N', 'WindGustDir_NE',
'WindGustDir_NNE', 'WindGustDir_NNW', 'WindGustDir_NW', 'WindGustDir_S', 'WindGustDir_SE', 'WindGustDir_SSE', 'WindGustDir_SSW', 'WindGus
tDir_SW', 'WindGustDir_Unknown', 'WindGustDir_W', 'WindGustDir_WNW', 'WindGustDir_WSW', 'WindDir9am_E', 'WindDir9am_ENE', 'WindDir9am_ES
E', 'WindDir9am_N', 'WindDir9am_NE', 'WindDir9am_NNE', 'WindDir9am_NNW', 'WindDir9am_NW', 'WindDir9am_S', 'WindDir9am_SE', 'WindDir9am_SS
E', 'WindDir9am_SSW', 'WindDir9am_SW', 'WindDir9am_Unknown', 'WindDir9am_W', 'WindDir9am_WNW', 'WindDir9am_WSW', 'WindDir3pm_E', 'WindDir
3pm_ENE', 'WindDir3pm_ESE', 'WindDir3pm_N', 'WindDir3pm_NE', 'WindDir3pm_NNE', 'WindDir3pm_NNW', 'WindDir3pm_NW', 'WindDir3pm_S', 'WindDi
r3pm_SE', 'WindDir3pm_SSE', 'WindDir3pm_SSW', 'WindDir3pm_SW', 'WindDir3pm_Unknown', 'WindDir3pm_W', 'WindDir3pm_WNW', 'WindDir3pm_WSW',
'RainToday_No', 'RainToday_Yes']
```

In [96]:
```
print(model.coef_.tolist())
```

```
[[0.8759427121123949, -2.9146034988379115, 3.181148028059171, 0.8080038566918112, -1.669513513867109, 6.743836016973369, -0.6802696386868
67, -1.487665533037215, 0.29770156896077243, 6.005970408629825, 5.557936465472615, -9.294497866745052, -0.16263605493884503, 1.2882093692
232979, 0.5284208712081634, 2.0523199948957362, 0.597831828026479, -0.3359886522711448, 0.4412265669013653, -0.009898775501273188, 0.3309
460097184362, -0.35133175026186464, 0.1653381477734191, 0.432061363048786, -0.04105112370979982, 0.04247761016067064, 0.257492565587498,
0.006379436382661839, -0.06162435512729727, -0.4908691171748553, -0.15800108081887917, -0.5906773772771582, -0.8017991533842034, -0.27564
745643543137, -0.3203360323469535, -0.5592334309914397, 0.069796985882100 69, 0.004676780171886963, 0.06112934289554019, -0.89303290324595
27, -0.28943214351851265, -0.0016150502464875888, -0.46361637277386997, -0.4902187004493721, -0.08280889709949585, 0.2085141199492651, 0.
4358646999105958, 0.6011647731908243, 0.42789272342049683, -0.03549407923563424, 0.2125311040869512, -0.34431164887561433, 0.417284488205
18654, 0.021975312812179293, -0.11879455996422021, -0.7421474293021451, 0.33701497645485623, 0.1635296882394633, 0.17032419393567833, 0.1
71074827975738 28, -0.25080915038963464, 0.013045137865874818, 0.6911855921562511, -0.8120632674189728, -0.19997063439763338, -0.181235743
52265387, -0.16992554026121706, -0.09049097009347515, -0.20040702745400898, -0.23529846874477425, -0.3143158441527274, -0.143691182954625
32, -0.13459051896494478, -0.12408782531985864, -0.0677530498423623, -0.07444188387025358, -0.09800649523265582, -0.08175395820469261, 0.
0, -0.11839014791769539, -0.22920325941211073, -0.17642295152057721, -0.3224970426685626, -0.03693659791288521, -0.33485940854281787, -0.
0777890810512166, -0.026111801865727726, 0.12768752574274508, -0.08119800419996868, -0.06675218421978682, -0.41361532685639824, -0.316949
88350939823, -0.4144279154583496, -0.19724570411785153, -0.06831542396184098, 0.0, -0.10567595428826022, -0.07516974854227522, -0.0301583
160125476, -0.22367926690421108, -0.14502379877053617, -0.21618872444201334, 0.045448939761485736, -0.30508953543192363, -0.0777901651351
0408, 0.28687218734042885, 0.21724673123682445, -0.2553230246937691, -0.1589355543092586, -0.3504464823492939, -0.31743854634484425, -0.3
7824932832714075, 0.0, -0.21161023669591783, -0.052458016188125525, -0.2973500462125173, -1.4690710483159646, -0.9709438191400028]]
```

In [97]:
```
weight_df = pd.DataFrame({
    'feature' : numerical_cols + encoded_cols,
    'weight' : model.coef_.tolist()[0]
})
```

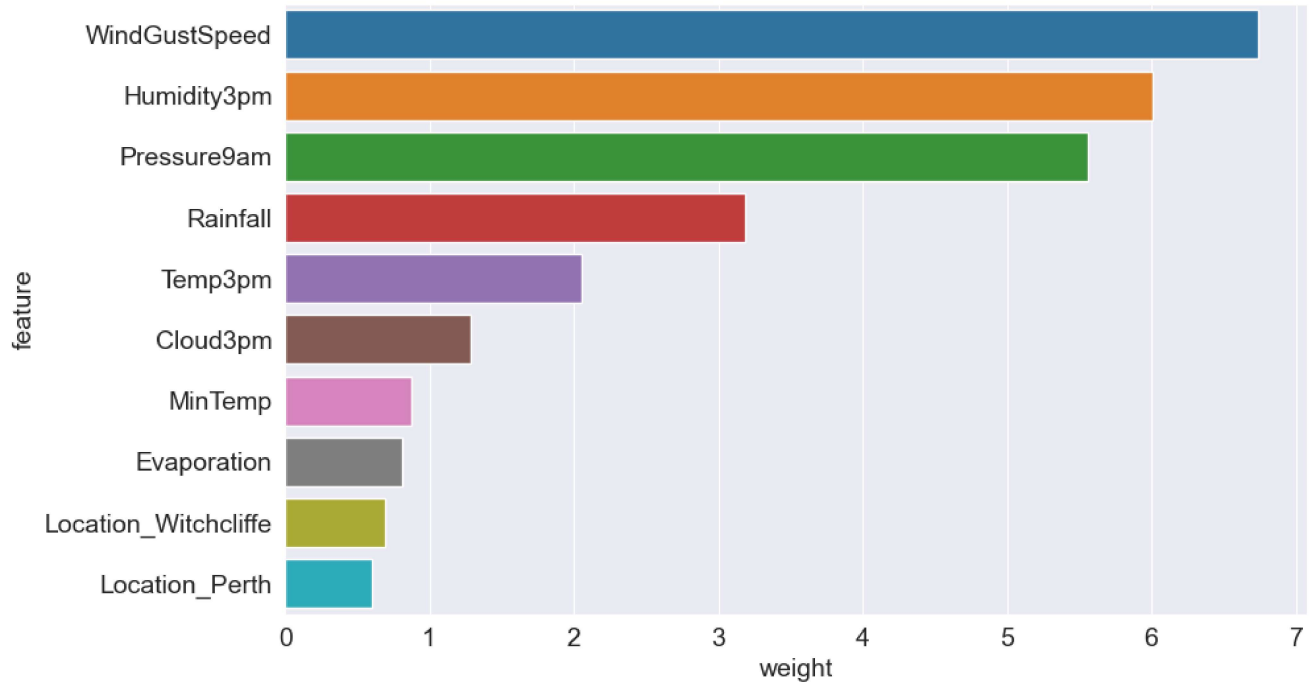
In [98]:
```
weight_df
```

Out[98]:

	feature	weight
0	MinTemp	0.875943
1	MaxTemp	-2.914603
2	Rainfall	3.181148
3	Evaporation	0.808004
4	Sunshine	-1.669514
...	...	...
113	WindDir3pm_W	-0.211610
114	WindDir3pm_WNW	-0.052458
115	WindDir3pm_WSW	-0.297350
116	RainToday_No	-1.469071
117	RainToday_Yes	-0.970944

118 rows × 2 columns

In [99]:
```
sns.barplot(data=weight_df.sort_values('weight',ascending=False).head(10), x = 'weight',y = 'feature')
plt.show()
```



In [100…
```
print(model.intercept_)
```

```
[-2.44001487]
```

Each weight is applied to the value in a specific column of the input. Higher the weight, greater the impact of the column on the prediction.

Making Prediction And Evaluating The Model

We can now use the trained model to make predictions on the training, test

In [101…
```
X_train = train_inputs[numerical_cols + encoded_cols]
X_val = val_inputs[numerical_cols + encoded_cols]
X_test = test_inputs[numerical_cols + encoded_cols]
```

In [102…
```
train_preds = model.predict(X_train)
```

In [103…
```
list(train_preds)
```

```
Out[103]: ['No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'Yes',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
```

```
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
```

```
'No',
'No',
'Yes',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'Yes',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'Yes',
```

```
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'Yes',
'No',
'No',
'Yes',
```

```
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'Yes',
'No',
'Yes',
```

```
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'Yes',
```

```
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'Yes',
```

```
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'Yes',
```

```
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
```

```
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
```

```
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'Yes',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'Yes',
'Yes',
'Yes',
'Yes',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
'No',
```

```
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 'No',
 ...]
```

In [104…]
```
train_targets
```

Out[104]:
```
0         No
1         No
2         No
3         No
4         No
          ..
144548    No
144549    No
144550    No
144551    No
144552    No
Name: RainTomorrow, Length: 97988, dtype: object
```

We can output a probabilistic prediction using predict_proba.

In [105…]
```
train_probs = model.predict_proba(X_train)
train_probs
```

Out[105]:
```
array([[0.94157918, 0.05842082],
       [0.94348473, 0.05651527],
       [0.96210254, 0.03789746],
       ...,
       [0.98712455, 0.01287545],
       [0.98320559, 0.01679441],
       [0.86686691, 0.13313309]])
```

In [106…]
```
model.classes_
```

Out[106]:
```
array(['No', 'Yes'], dtype=object)
```

We can test the accuracy of the model's predictions by computing the percentage of matching values in train_preds and train_targets. This can be done using the accuracy_score function from sklearn.metrics.

In [107…]
```
from sklearn.metrics import accuracy_score
```

In [108…]
```
accuracy_score(train_targets, train_preds)
```

Out[108]:
```
0.8518185900314325
```

It gives 85.1% on the training set.

Confusion Matrix

In [109…]
```
from sklearn.metrics import confusion_matrix
```

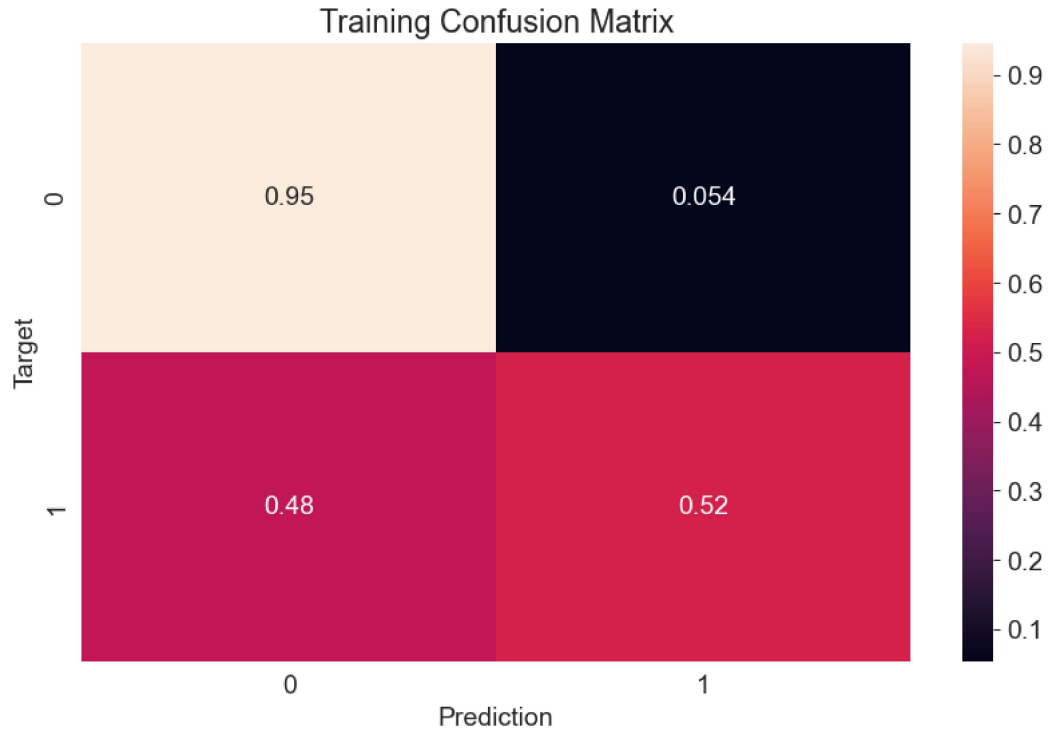
In [110…]
```
confusion_matrix(train_targets, train_preds, normalize = 'true')
```

Out[110]:
```
array([[0.94609529, 0.05390471],
       [0.47770438, 0.52229562]])
```

In [111…]
```
def predict_and_plot (inputs, targets, name=''):
    preds=model.predict(inputs)

    accuracy= accuracy_score(targets,preds)
    print('Accuracy : {:.2f}%'.format(accuracy*100))

    cf = confusion_matrix(targets,preds, normalize='true')
    plt.figure()
    sns.heatmap(cf,annot=True)
    plt.xlabel('Prediction')
    plt.ylabel('Target')
    plt.title('{} Confusion Matrix'. format(name))

    return preds
```

In [112…]
```
train_preds=predict_and_plot(X_train, train_targets, 'Training')
```

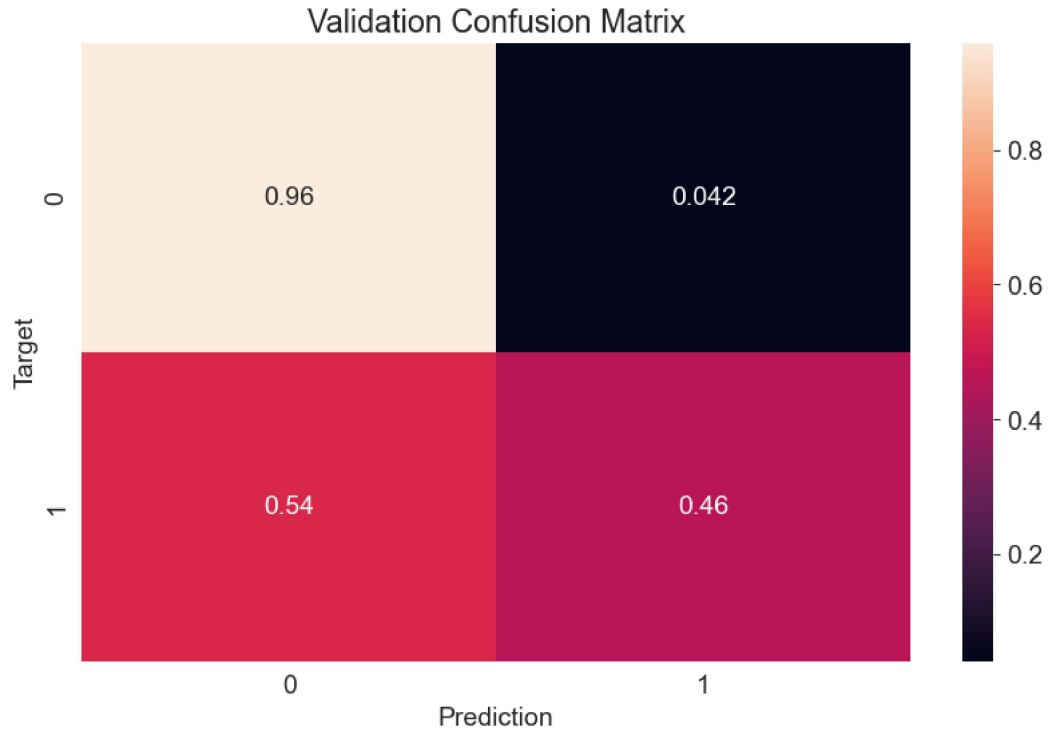
```
Accuracy : 85.18%
```



Let's compute the model's accuracy on the validation and test sets too.

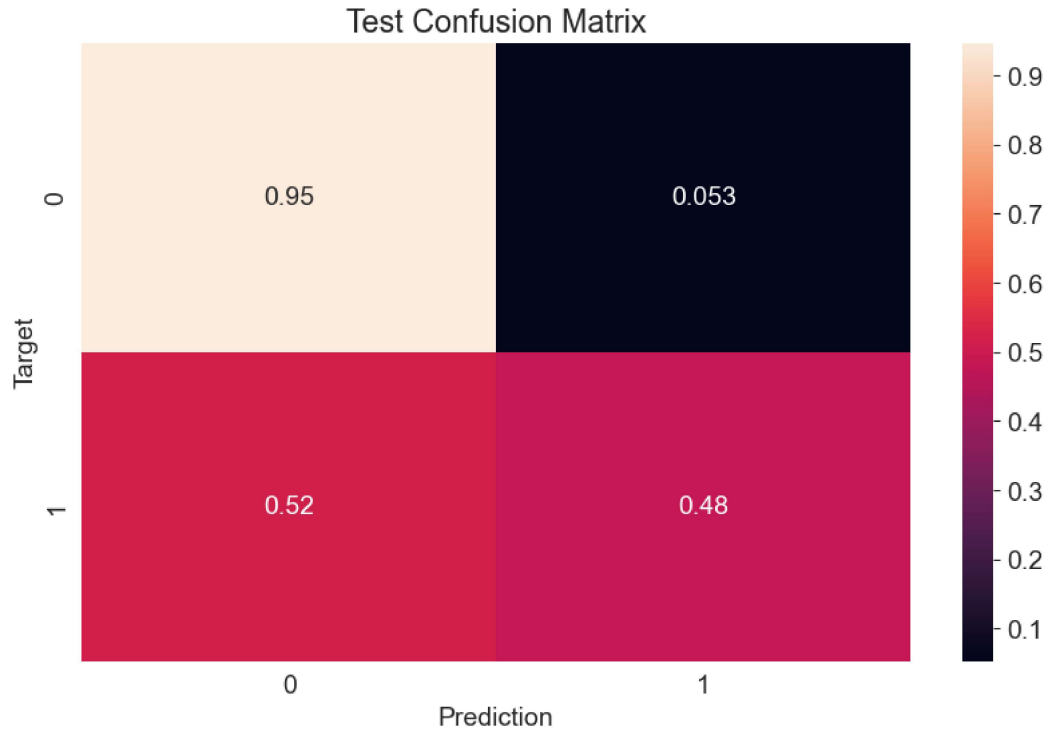
In [113…
```
val_preds = predict_and_plot(X_val,val_targets,'Validation')
```

```
Accuracy : 85.44%
```



In [114…
```
test_preds = predict_and_plot(X_test, test_targets,'Test')
```

```
Accuracy : 84.22%
```



In [115…
```
def random_guess (inputs):
    return np.random.choice(['No','Yes'],len(inputs))
```

In [116…
```
def all_no(inputs):
    return np.full(len(inputs),'No')
```

In [117…
```
random_guess(X_val)
```

Out[117]:
```
array(['Yes', 'No', 'No', ..., 'Yes', 'Yes', 'No'], dtype='<U3')
```

In [118…
```
all_no(X_val)
```

Out[118]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype='<U2')
```

In [119…
```
accuracy_score(test_targets, random_guess(X_test))
```

Out[119]:
```
0.4973551147413458
```

In [120…
```
accuracy_score(test_targets, all_no(X_test))
```

Out[120]:
```
0.7734344612991054
```

Making Prediction On A Single Input

In [121…
```
new_input = {'Date' : '2021-06-19',
             'Location' : 'Katherine',
             'MinTemp' : 23.2,
             'MaxTemp' : 33.2,
             'Rainfall' : 10.2,
             'Evaporation' : 4.2,
             'Sunshine' : np.nan,
             'WindGustDir' : 'NNW',
             'WindGustSpeed' : 10.0,
             'WindDir9am' : 'NW',
             'WindDir3pm' : 'NNE',
             'WindSpeed9am' : 13.0,
             'WindSpeed3pm' :20.0 ,
              'Humidity9am' : 89.0,
              'Humidity3pm' : 58.0,
              'Pressure9am' :1004.8 ,
              'Pressure3pm' : 1001.5,
              'Cloud9am' : 8.0,
              'Cloud3pm' : 5.0,
              'Temp9am' : 25.7,
              'Temp3pm' : 33.0,
              'RainToday' : 'Yes'

             }
```

In [122…
```
new_input_df = pd.DataFrame([new_input])
```

In [123…
```
new_input_df
```

Out[123]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3
0	2021-06-19	Katherine	23.2	33.2	10.2	4.2	NaN	NNW	10.0	NW	NNE	13.0	

In [124…
```
new_input_df[numerical_cols] = imputer.transform(new_input_df[numerical_cols])
new_input_df[numerical_cols] = scaler.transform(new_input_df[numerical_cols])
new_input_df[encoded_cols] = encoder.transform(new_input_df[categorical_cols])
```

In [125…
```
X_new_input = new_input_df[numerical_cols + encoded_cols]
X_new_input
```

Out[125]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3p
0	0.747642	0.718336	0.027493	0.028966	0.526244	0.031008	0.1	0.229885	0.89	0.58	0.401653	0.39

In [126…
```
prediction = model. predict(X_new_input)[0]
```

In [127…
```
prediction
```

Out[127]:
```
'No'
```

In [128…
```
prob = model.predict_proba(X_new_input)[0]
```

In [129…
```
prob
```

Out[129]:
```
array([0.89145062, 0.10854938])
```

In [130…
```
def predict_input(single_input):
    input_df = pd.DataFrame([single_input])
    input_df[numerical_cols] = imputer.transform(input_df[numerical_cols])
    input_df[numerical_cols] = scaler.transform(input_df[numerical_cols])
    input_df[encoded_cols] = encoder.transform(input_df[categorical_cols])
    X_input = input_df[numerical_cols + encoded_cols]
    pred = model.predict(X_input)[0]
    prob = model.predict_proba(X_input)[0][list(model.classes_).index(pred)]
    return pred, prob
```

In [131…
```
new_input = {'Date' : '2021-06-19',
             'Location' : 'Launceston',
             'MinTemp' : 23.2,
             'MaxTemp' : 33.2,
             'Rainfall' : 10.2,
             'Evaporation' : 4.2,
             'Sunshine' : np.nan,
             'WindGustDir' : 'NNW',
             'WindGustSpeed' : 52.0,
             'WindDir9am' : 'NW',
             'WindDir3pm' : 'NNE',
             'WindSpeed9am' : 13.0,
             'WindSpeed3pm' :20.0 ,
              'Humidity9am' : 89.0,
              'Humidity3pm' : 58.0,
              'Pressure9am' :1004.8 ,
              'Pressure3pm' : 1001.5,
              'Cloud9am' : 8.0,
              'Cloud3pm' : 5.0,
              'Temp9am' : 25.7,
              'Temp3pm' : 33.0,
              'RainToday' : 'Yes'

             }
```

In [132…
```
predict_input(new_input)
```

Out[132]:
```
('Yes', 0.6493479784180273)
```

In [133…
```
raw_df.Location.unique()
```

Out[133]:
```
array(['Albury', 'BadgerysCreek', 'Cobar', 'CoffsHarbour', 'Moree',
       'Newcastle', 'NorahHead', 'NorfolkIsland', 'Penrith', 'Richmond',
       'Sydney', 'SydneyAirport', 'WaggaWagga', 'Williamtown',
       'Wollongong', 'Canberra', 'Tuggeranong', 'MountGinini', 'Ballarat',
       'Bendigo', 'Sale', 'MelbourneAirport', 'Melbourne', 'Mildura',
       'Nhil', 'Portland', 'Watsonia', 'Dartmoor', 'Brisbane', 'Cairns',
       'GoldCoast', 'Townsville', 'Adelaide', 'MountGambier', 'Nuriootpa',
       'Woomera', 'Albany', 'Witchcliffe', 'PearceRAAF', 'PerthAirport',
       'Perth', 'SalmonGums', 'Walpole', 'Hobart', 'Launceston',
       'AliceSprings', 'Darwin', 'Katherine', 'Uluru'], dtype=object)
```

In [134…
```
new_input = {'Date' : '2021-06-19',
             'Location' : 'Albury',
             'MinTemp' : 23.2,
             'MaxTemp' : 33.2,
             'Rainfall' : 10.2,
             'Evaporation' : 4.2,
             'Sunshine' : np.nan,
             'WindGustDir' : 'NNW',
             'WindGustSpeed' : 52.0,
             'WindDir9am' : 'NW',
             'WindDir3pm' : 'NNE',
             'WindSpeed9am' : 13.0,
             'WindSpeed3pm' :20.0 ,
              'Humidity9am' : 89.0,
              'Humidity3pm' : 58.0,
              'Pressure9am' :1004.8 ,
              'Pressure3pm' : 1001.5,
              'Cloud9am' : 8.0,
              'Cloud3pm' : 5.0,
              'Temp9am' : 25.7,
              'Temp3pm' : 33.0,
              'RainToday' : 'Yes'

             }
```

In [135…
```
predict_input(new_input)
```

Out[135]:
```
('Yes', 0.7913444341932449)
```

In [136…
```
new_input = {'Date' : '2021-06-19',
             'Location' : 'Townsville',
             'MinTemp' : 23.2,
             'MaxTemp' : 33.2,
             'Rainfall' : 10.2,
             'Evaporation' : 4.2,
             'Sunshine' : np.nan,
             'WindGustDir' : 'NNW',
             'WindGustSpeed' : 52.0,
             'WindDir9am' : 'NW',
             'WindDir3pm' : 'NNE',
             'WindSpeed9am' : 13.0,
             'WindSpeed3pm' :20.0 ,
              'Humidity9am' : 89.0,
              'Humidity3pm' : 58.0,
              'Pressure9am' :1004.8 ,
              'Pressure3pm' : 1001.5,
              'Cloud9am' : 8.0,
              'Cloud3pm' : 5.0,
              'Temp9am' : 25.7,
              'Temp3pm' : 33.0,
              'RainToday' : 'Yes'

             }
```

In [137…
```
predict_input(new_input)
```

Out[137]:
```
('Yes', 0.5373489095301723)
```

Joblib

Let's first create a Dictionary containing all the required objects

In [138…
```
import joblib
```

In [139…
```
aussie_rain = {
    'model' : model,
    'imputer' : imputer,
    'scaler' : scaler,
    'encoder' : encoder,
    'input_cols' : input_cols,
    'target_cols' : target_cols,
    'numeric_cols' : numerical_cols,
    'categorical_cols' : categorical_cols,
    'encoder_cols' : encoded_cols
```

```
}
```

In [140…
```
joblib.dump(aussie_rain,'aussie_rain.joblib')
```

Out[140]:
```
['aussie_rain.joblib']
```

In [141…
```
aussie_rain2 = joblib.load('aussie_rain.joblib')
```

In [142…
```
test_preds2 = aussie_rain2['model'].predict(X_test)
```

In [143…
```
accuracy_score(test_targets,test_preds2)
```

Out[143]:
```
0.8422014780241152
```

In [144…
```
test_targets.shape,test_preds2.shape
```

Out[144]:
```
((25710,), (25710,))
```

In [ ]:
```

```