Rain Prediction ANN

Import Libraries ¶

In [7]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [8]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
```

In [9]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dropout
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error , mean_absolute_percentage_er
```

Loading Data

In [11]:
```
data = pd.read_csv(r"C:\Users\hp\OneDrive\Documents\Desktop\weatherAUS.csv")
df = data.copy()
```

In [12]:
```
data.head()
```

Out[12]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGu
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 23 columns

In [15]:
```
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
```

In [14]:
```
data.describe()
```

Out[14]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpe
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000

In [15]:
```
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
```

In [14]:
```
data.describe()
```

Out[14]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpe
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000

In [16]:
```
# finding null value in Data
data.isnull().sum()
```

Out[16]:
```
Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
```

In [19]:
```
# adding different columns for days month and year
data["year"] = data["Date"].dt.year
data["month"] = data["Date"].dt.month
data["day"] = data["Date"].dt.day
```

In [17]:
```
# data is object thus changing it to datatime
```

In [20]:
```
data["Date"] = pd.to_datetime(data["Date"])
data.head()
```

Out[20]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGu
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 26 columns

In [18]:
```
# checking the data view
data["Date"]
```

Out[18]:
```
0        2008-12-01
1        2008-12-02
2        2008-12-03
3        2008-12-04
4        2008-12-05
            ...
145455   2017-06-21
145456   2017-06-22
145457   2017-06-23
145458   2017-06-24
145459   2017-06-25
Name: Date, Length: 145460, dtype: datetime64[ns]
```

In [19]:
```
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
# adding different columns for days month and year
data["year"] = data["Date"].dt.year
data["month"] = data["Date"].dt.month
data["day"] = data["Date"].dt.day
```

In [17]:
```
# data is object thus changing it to datatime
```

In [20]:
```
data.head()
data["Date"] = pd.to_datetime(data["Date"])
```

Out[20]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGu
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 26 columns

In [18]:
```
# checking the data view
data["Date"]
```

Out[18]:
```
0        2008-12-01
1        2008-12-02
2        2008-12-03
3        2008-12-04
4        2008-12-05
            ...
145455   2017-06-21
145456   2017-06-22
145457   2017-06-23
145458   2017-06-24
145459   2017-06-25
Name: Date, Length: 145460, dtype: datetime64[ns]
```

In [22]:
```
# deopping te date column
data.drop("Date",axis=1,inplace=True)
```

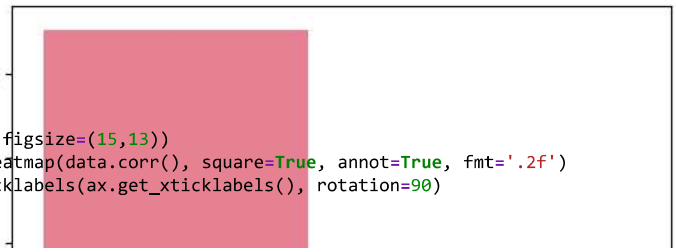
In [24]:
```
# listing the column
data.columns
```

Out[24]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshi
ne',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow', 'year', 'month', 'day'],
      dtype='object')
```

Data Visualization And Cleaning

In [26]:
```
cols = ["#C2C4E2", "#EED4E5"]
sns.countplot(x= data["RainTomorrow"], palette= "husl")
```

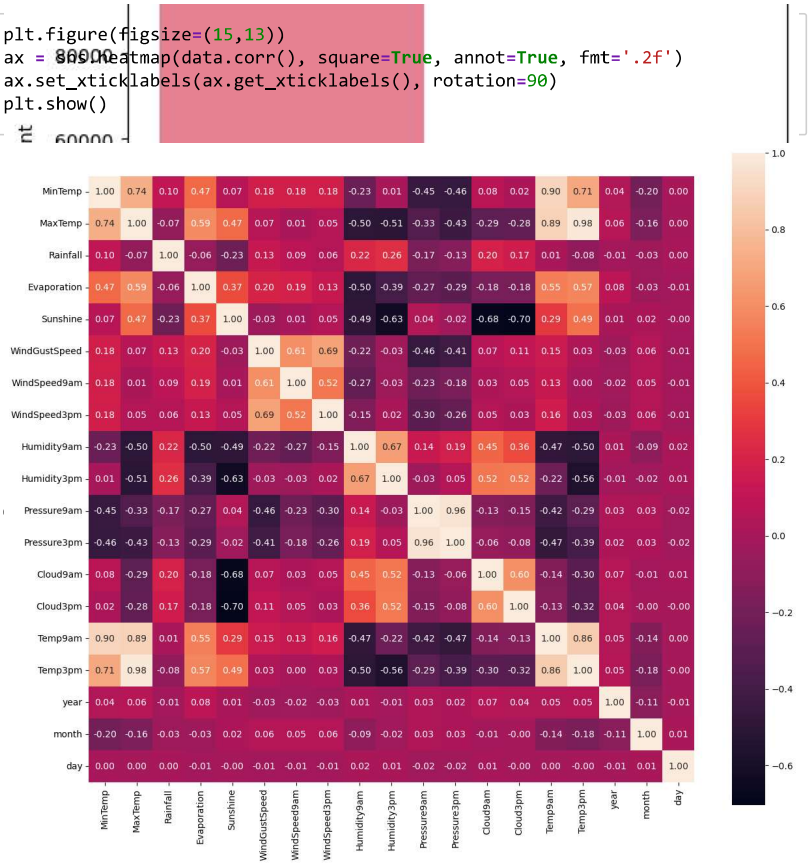
Out[26]:
```
<Axes: xlabel='RainTomorrow', ylabel='count'>
```



In [27]:
```
plt.figure(figsize=(15,13))
ax = sns.heatmap(data.corr(), square=True, annot=True, fmt='.2f')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
```

The

In [27]:
```
plt.figure(figsize=(15,13))
ax = sns.heatmap(data.corr(), square=True, annot=True, fmt='.2f')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
```

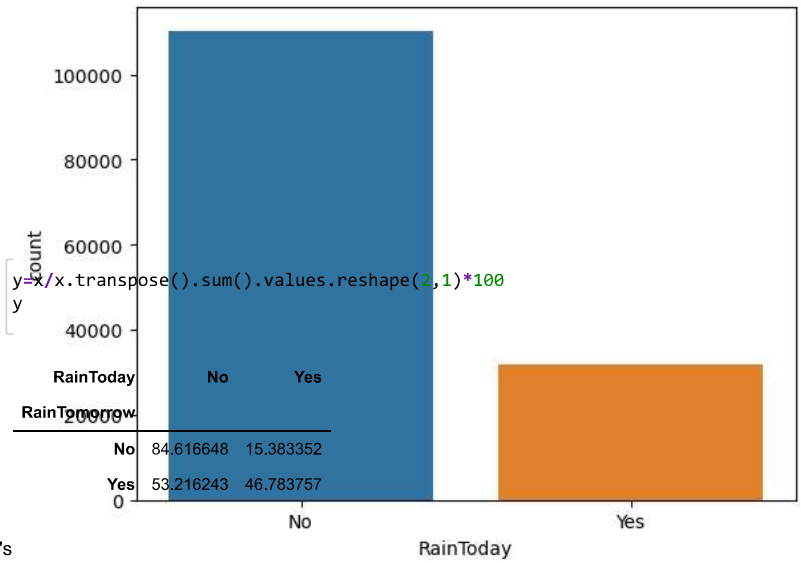


The

Rain today ~ Rain tomorrow

In [29]:
```
sns.countplot(x= data["RainToday"])
```

Out[29]: <Axes: xlabel='RainToday', ylabel='count'>



In [32]:
```
y=x/x.transpose().sum().values.reshape(2,1)*100
y
```

Out[32]:

RainToday	No	Yes
RainTomorrow		
No	84.616648	15.383352
Yes	53.216243	46.783757

If it's
chance of raining tomorrow = 46%

In [31]:
```
x=pd.crosstab(data["RainTomorrow"],data["RainToday"])
```

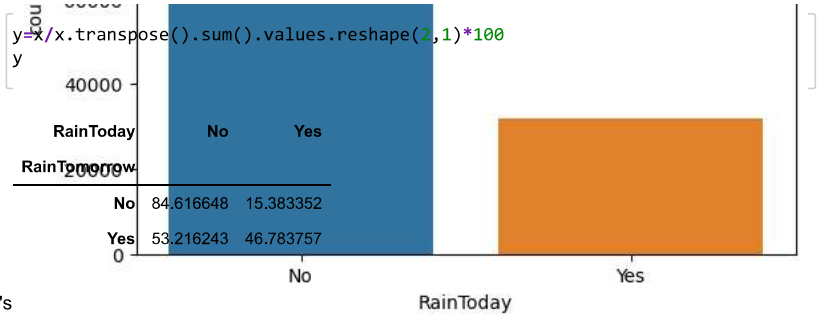
In [33]:
```
x.plot(kind="bar", figsize=(8,6),color=["#006666","#d279a6"])
```

Out[33]: <Axes: xlabel='RainTomorrow'>

In [32]:
```
y=x/x.transpose().sum().values.reshape(2,1)*100
y
```

Out[32]:

RainToday	No	Yes
RainTomorrow		
No	84.616648	15.383352
Yes	53.216243	46.783757

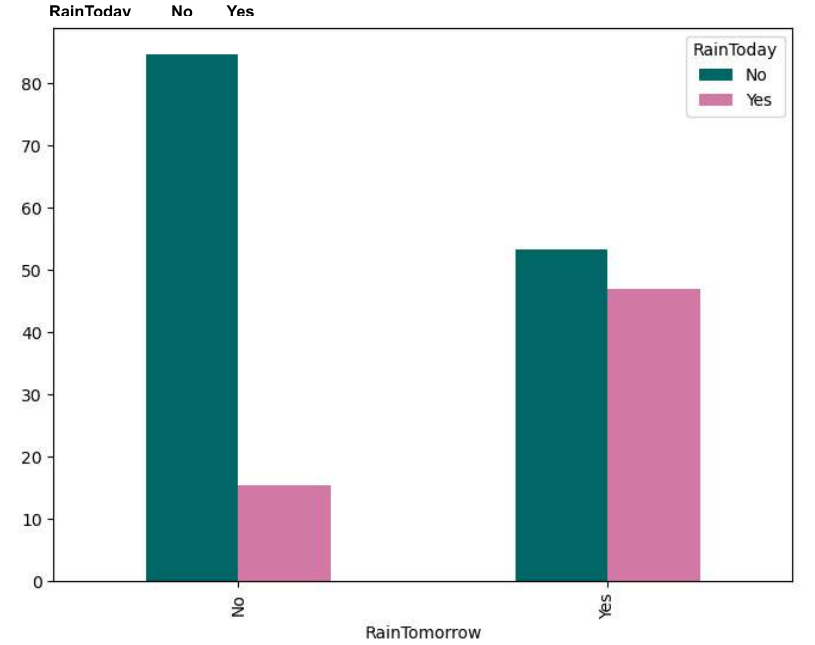


If it's
chance of raining tomorrow = 46%

In [31]:
```
x=pd.crosstab(data["RainTomorrow"],data["RainToday"])
```

In [33]:
```
x.plot(kind="bar", figsize=(8,6),color=["#006666","#d279a6"])
```

Out[33]: <Axes: xlabel='RainTomorrow'>



Location ~ Rain Today

In [37]:
```
x=pd.crosstab(data["Location"],data["RainToday"])
x
```

Out[37]:

RainToday	No	Yes
Location		
Adelaide	2402	689
Albany	2114	902
Albury	2394	617
AliceSprings	2788	244
BadgerysCreek	2345	583
Ballarat	2247	781
Bendigo	2472	562
Brisbane	2452	709
Cairns	2038	950
Canberra	2789	629
Cobar	2602	386
CoffsHarbour	2084	869
Dartmoor	2021	921

In [37]:
```
x=pd.crosstab(data["Location"],data["RainToday"])
x
```

Out[37]:

Location	No	Yes
Adelaide	2402	689
Albany	2114	902
Albury	2394	617
AliceSprings	2788	244
BadgerysCreek	2345	583
Ballarat	2247	781
Bendigo	2472	562
Brisbane	2452	709
Cairns	2038	950
Canberra	2789	629
Cobar	2602	386
CoffsHarbour	2084	869
Dartmoor	2021	921
Darwin	2341	852
GoldCoast	2205	775
Hobart	2426	762
Katherine	1295	265
Launceston	2328	700
Melbourne	1799	636
MelbourneAirport	2356	653
Mildura	2680	327
Moree	2460	394
MountGambier	2110	921
MountGinini	2088	819
Newcastle	2224	731
Nhil	1327	242
NorahHead	2121	808
NorfolkIsland	2045	919
Nuriootpa	2411	592
PearceRAAF	2257	505
Penrith	2369	595
Perth	2548	645
PerthAirport	2442	567
Portland	1902	1054
Richmond	2391	560
Sale	2357	643
SalmonGums	2483	472
Sydney	2471	866
SydneyAirport	2231	774
Townsville	2513	520
Tuggeranong	2430	568
Uluru	1406	116
WaggaWagga	2440	536
Walpole	1870	949
Watsonia	2261	738
Williamtown	1853	700
Witchcliffe	2073	879
Wollongong	2269	713
Woomera	2789	202

In [38]:
```
# getting percentage of raining days and non raining days for each city
y=x/x.transpose().sum().values.reshape((-1, 1))*100
y
```

Out[38]:

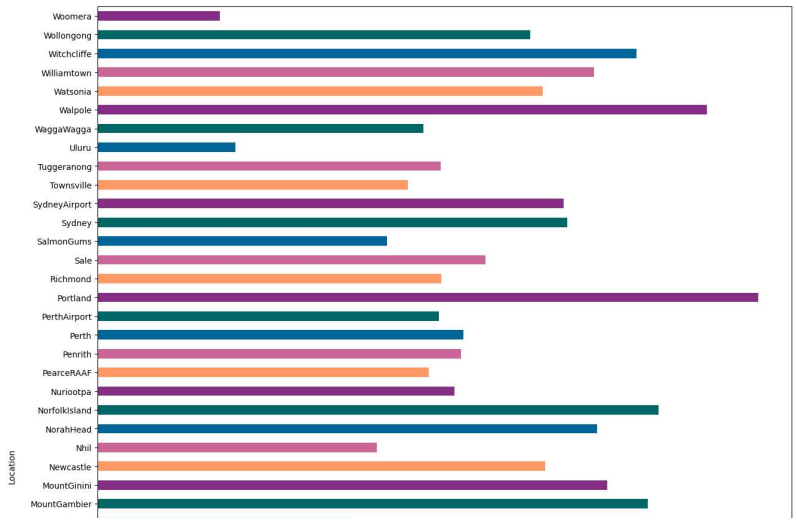
Location	No	Yes
Adelaide	77.709479	22.290521
Albany	70.092838	29.907162
Albury	79.508469	20.491531
AliceSprings	91.952507	8.047493
BadgerysCreek	80.088798	19.911202
Ballarat	74.207398	25.792602
Bendigo	81.476599	18.523401
Brisbane	77.570389	22.429611
Cairns	68.206158	31.793842
Canberra	81.597425	18.402575
Cobar	87.081660	12.918340
CoffsHarbour	70.572299	29.427701

In [38]:
```
# getting percentage of raining days and non raining days for each city
y=x/x.transpose().sum().values.reshape((-1, 1))*100
y
```

Out[38]:

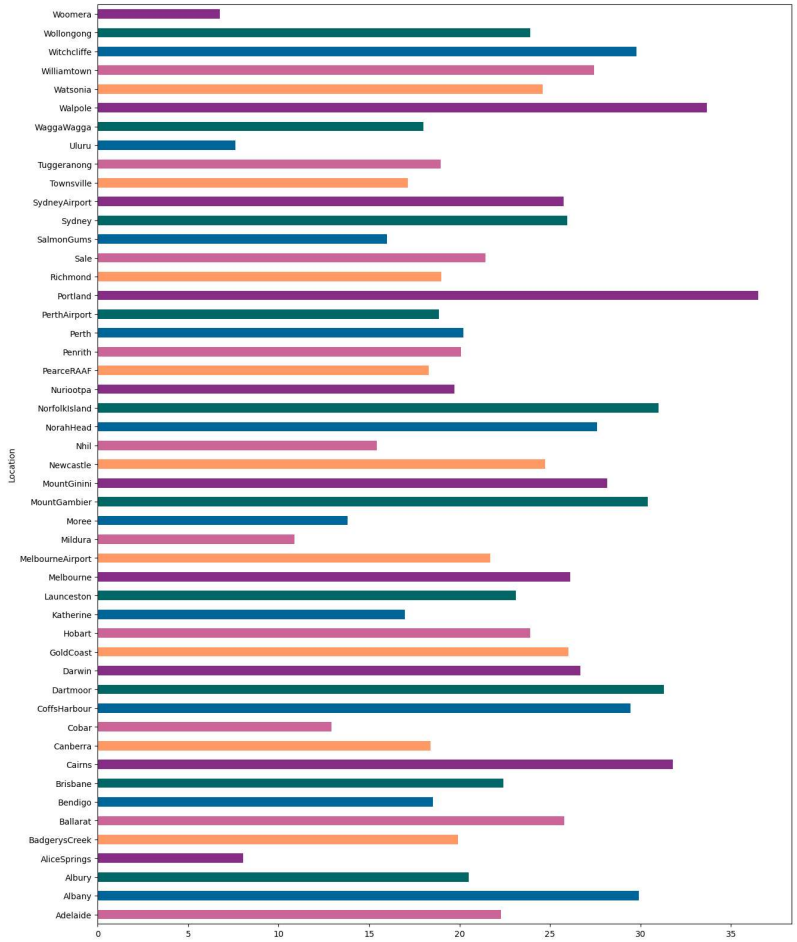
RainToday	No	Yes
Location		
Adelaide	77.709479	22.290521
Albany	70.092838	29.907162
Albury	79.508469	20.491531
AliceSprings	91.952507	8.047493
BadgerysCreek	80.088798	19.911202
Ballarat	74.207398	25.792602
Bendigo	81.476599	18.523401
Brisbane	77.570389	22.429611
Cairns	68.206158	31.793842
Canberra	81.597425	18.402575
Cobar	87.081660	12.918340
CoffsHarbour	70.572299	29.427701
Dartmoor	68.694765	31.305235
Darwin	73.316630	26.683370
GoldCoast	73.993289	26.006711
Hobart	76.097867	23.902133
Katherine	83.012821	16.987179
Launceston	76.882431	23.117569
Melbourne	73.880903	26.119097
MelbourneAirport	78.298438	21.701562
Mildura	89.125374	10.874626
Moree	86.194814	13.805186
MountGambier	69.613989	30.386011
MountGinini	71.826625	28.173375
Newcastle	75.262267	24.737733
Nhil	84.576163	15.423837
NorahHead	72.413793	27.586207
NorfolkIsland	68.994602	31.005398
Nuriootpa	80.286380	19.713620
PearceRAAF	81.716148	18.283852
Penrith	79.925776	20.074224
Perth	79.799562	20.200438
PerthAirport	81.156530	18.843470
Portland	83.484848	16.515152
Richmond	81.023382	18.976618

In [39]:
```
color=[ '#cc6699', '#006699', '#006666','#862d86','#ff9966'  ]
y.Yes.plot(kind="barh",figsize=(15,20),color=color);
```



In [39]:
```
color=[ "#cc6699", "#006699", "#006666",'#862d86','#ff9966'  ]
y.Yes.plot(kind="barh",figsize=(15,20),color=color);
```

```
PerthAirport   81.156530   18.843470
Portland       63.484646   36.515354
Richmond       81.023382   18.976618
```



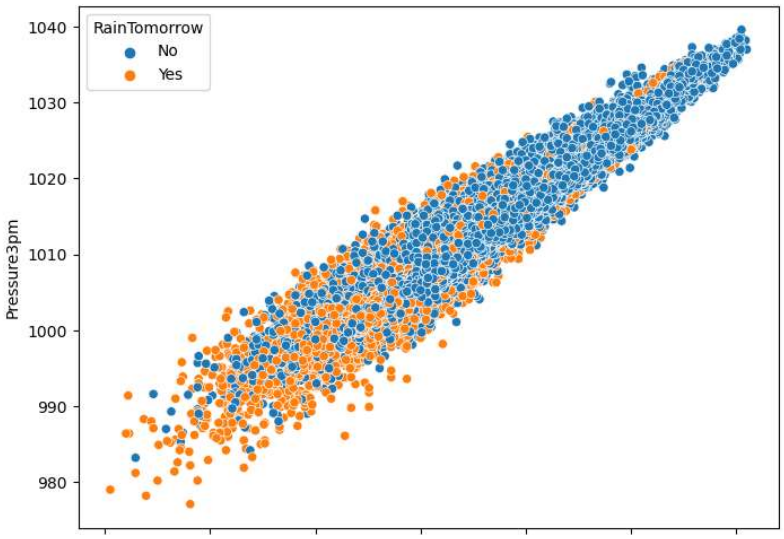
location affects the raining as for Portland, it's raining 36% of days and for Woomers it's raining only 6% of the days

Effects of humidity and pressure on raining tomorrow

In [40]:
```
data.columns
```

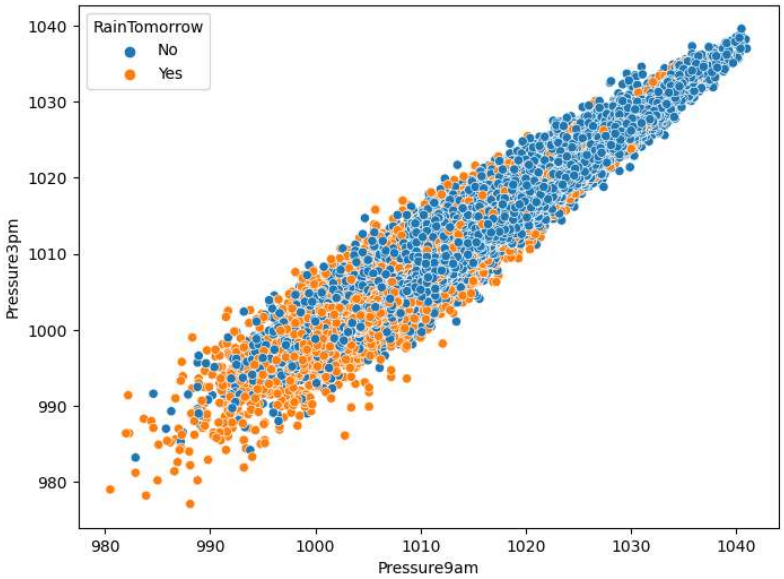
Out[40]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
```

In [42]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='Pressure9am',y='Pressure3pm',hue='RainTomorrow')
```

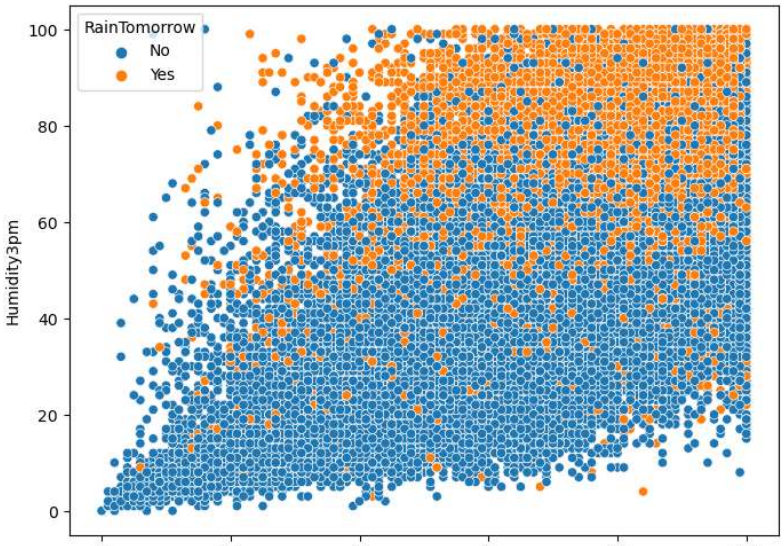


In [42]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='Pressure9am',y='Pressure3pm',hue='RainTomorrow')
```

Out[40]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
```

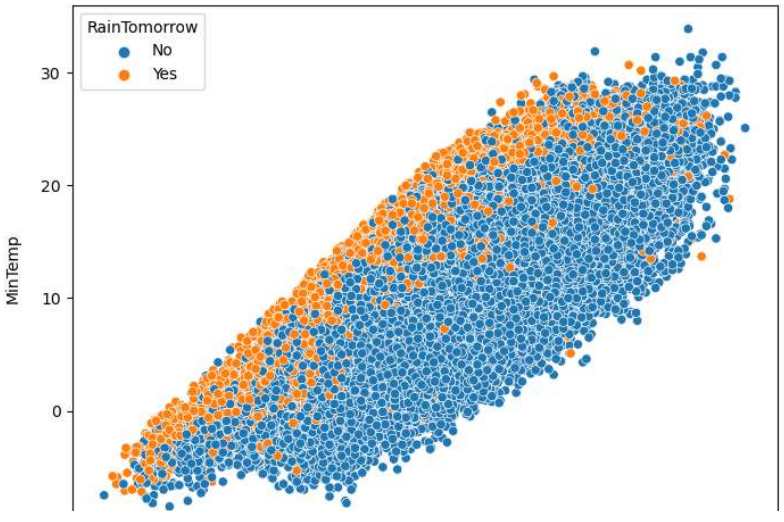


In [44]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='Humidity9am',y='Humidity3pm',hue='RainTomorrow')
```



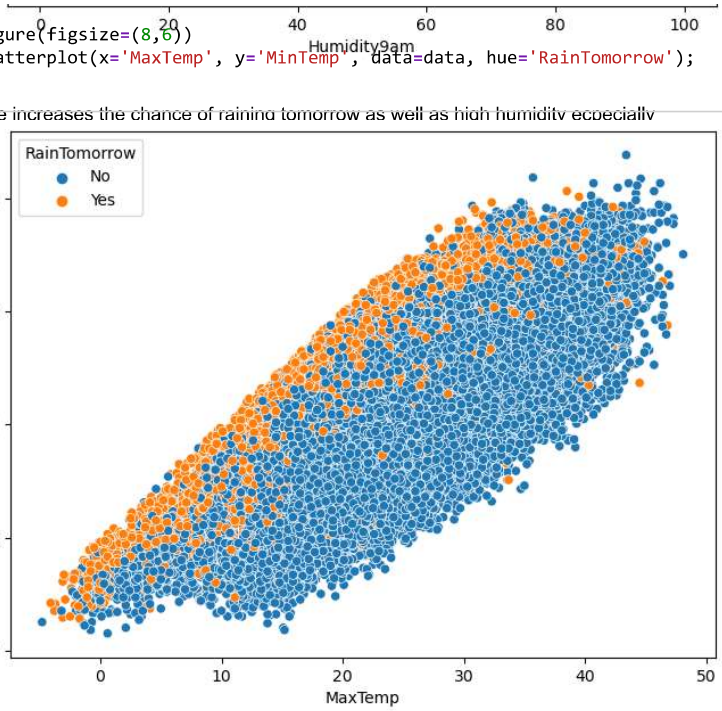
In [45]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x='MaxTemp', y='MinTemp', data=data, hue='RainTomorrow');
```

Low pressure increases the chance of raining tomorrow as well as high humidity especially humi



Dc

In [45]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x='MaxTemp', y='MinTemp', data=data, hue='RainTomorrow');
```



Low pressure increases the chance of raining tomorrow as well as high humidity ecpecially humi

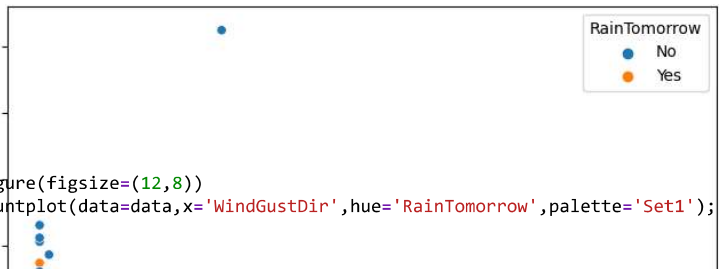
Dc

Chance of raining tomorrow increses when the maximum temperature of the day and minimum one are close to each other

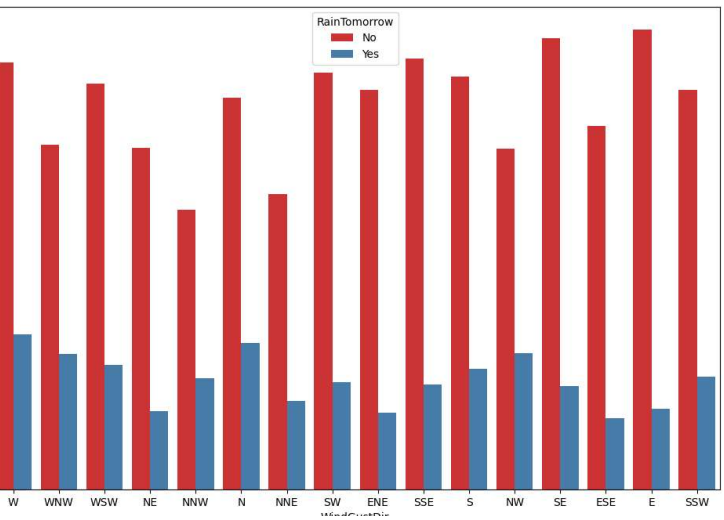
```
It's a rainy season in June, July and August as these are the coldest months
of the year in Australia
```

Effect of rain fall and evaporation on raining tomorrow

In [51]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x='Rainfall',y='Evaporation',hue='RainTomorrow');
```

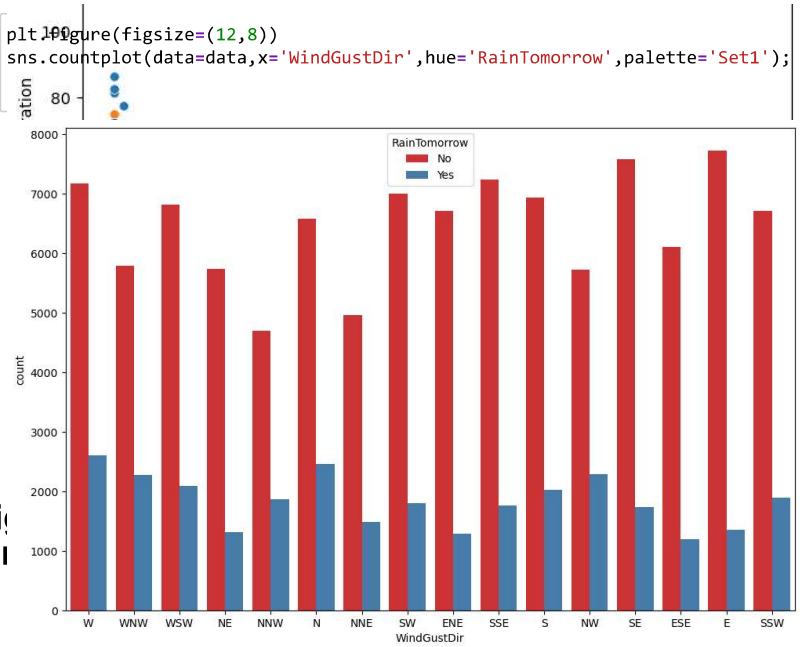


In [52]:
```
plt.figure(figsize=(12,8))
sns.countplot(data=data,x='WindGustDir',hue='RainTomorrow',palette='Set1');
```



Hi
to

In [52]:
```
plt.figure(figsize=(12,8))
sns.countplot(data=data,x='WindGustDir',hue='RainTomorrow',palette='Set1');
```



Handling Missing Values

In [54]:
```
# percentage of missing data in each column
data.isnull().sum()/data.shape[0]*100
```

Out[54]:
```
Location          0.000000
MinTemp           1.020899
MaxTemp           0.866905
Rainfall          2.241853
Evaporation      43.166506
Sunshine         48.009762
WindGustDir       7.098859
WindGustSpeed     7.055548
WindDir9am        7.263853
WindDir3pm        2.906641
WindSpeed9am      1.214767
WindSpeed3pm      2.105046
Humidity9am       1.824557
Humidity3pm       3.098446
Pressure9am      10.356799
Pressure3pm      10.331363
Cloud9am         38.421559
Cloud3pm         40.807095
Temp9am           1.214767
Temp3pm           2.481094
RainToday         2.241853
RainTomorrow      2.245978
year              0.000000
month             0.000000
day               0.000000
dtype: float64
```

In [57]:
```
#numeric features
t = (data.dtypes == "float64")
num_cols = list(t[t].index)
```

In [58]:
```
for i in num_cols:
    data[i].fillna(data[i].median(), inplace=True)
```

In [55]:
```
# filling feature with large percentage of missing data using random choice
lst=['Evaporation','Sunshine','Cloud9am','Cloud3pm']
for col in lst:
    fill_list = data[col].dropna()
    data[col] = data[col].fillna(pd.Series(
        np.random.choice(fill_list , size = len(data.index))))
```

In [59]:
```
data.isnull().sum()
```

Out[59]:
```
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
```

In [56]:
```
#categorical features
s = (data.dtypes == "object")
object_cols = list(s[s].index)
for i in object_cols:
    data[i].fillna(data[i].mode()[0], inplace=True)
```

In [57]:
```
#numeric features
t = (data.dtypes == "float64")
num_cols = list(t[t].index)
```

```
RainTomorrow      2.245978
year              0.000000
month             0.000000
day               0.000000
dtype: float64
```

In [58]:
In [55]:
```
# filling feature with large percentage of missing data using random choice
lst=['Evaporation','Sunshine','Cloud9am','Cloud3pm']
for col in lst:
    fill_list = data[col].dropna()
    data[col] = data[col].fillna(pd.Series(
        np.random.choice(fill_list , size = len(data.index))))
```

In [59]:
```
data.isnull().sum()
```

Out[59]:
```
Location          0
MinTemp           0
MaxTemp           0
Rainfall          0
Evaporation       0
Sunshine          0
WindGustDir       0
WindGustSpeed     0
WindDir9am        0
WindDir3pm        0
WindSpeed9am      0
WindSpeed3pm      0
Humidity9am       0
Humidity3pm       0
Pressure9am       0
Pressure3pm       0
Cloud9am          0
Cloud3pm          0
Temp9am           0
Temp3pm           0
RainToday         0
RainTomorrow      0
year              0
month             0
day               0
dtype: int64
```

In [56]:
```
#categorical features
s = (data.dtypes == "object")
object_cols = list(s[s].index)
```

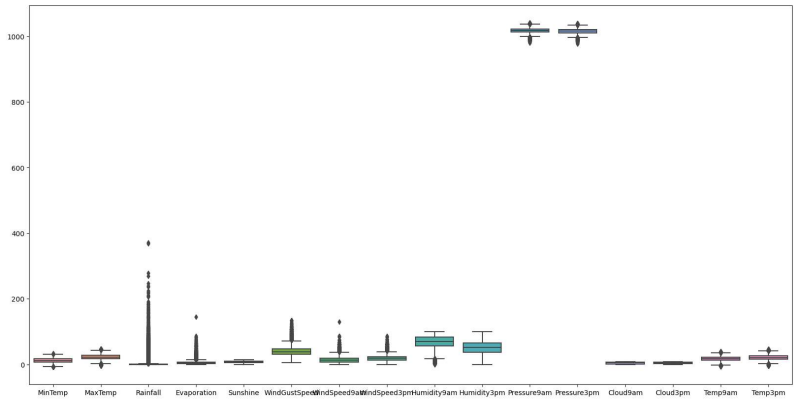
```
for i in object_cols:
    data[i].fillna(data[i].mode()[0], inplace=True)
```

Headline Outliers

In [60]:
```
num_cols
```

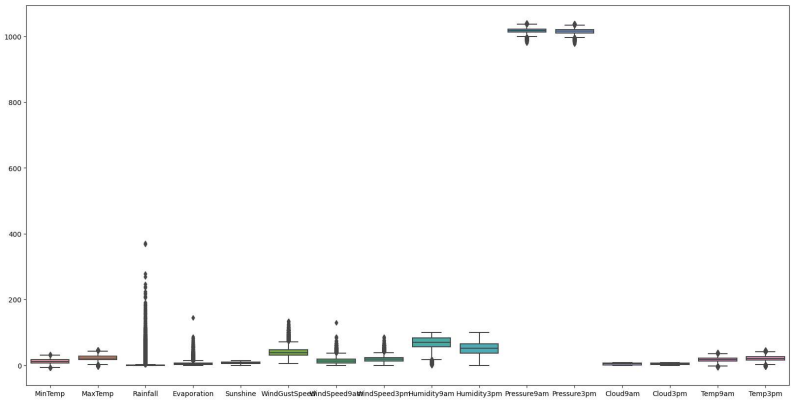
Out[60]:
```
['MinTemp',
 'MaxTemp',
 'Rainfall',
 'Evaporation',
 'Sunshine',
 'WindGustSpeed',
 'WindSpeed9am',
 'WindSpeed3pm',
 'Humidity9am',
 'Humidity3pm',
 'Pressure9am',
 'Pressure3pm',
 'Cloud9am',
 'Cloud3pm',
 'Temp9am',
 'Temp3pm']
```

In [61]:
```
# show all the outlier as a graph using boxplot for each columns
plt.figure(figsize=(20,10))
sns.boxplot(data=df,orient="v")
plt.show()
```



In [62]:
```
# MinTemp
x=data.loc[(data['MinTemp']>30)|(data['MinTemp']<-6)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(ro
```

In [61]:
```
'Cloud9am',
 'Cloud3pm',
 'Temp9am',
 'Temp3pm']
#,show all the outlier as a graph using boxplot for each columns
plt.figure(figsize=(20,10))
sns.boxplot(data=df,orient="v")
plt.show()
```



In [62]:
```
# MinTemp
x=data.loc[(data['MinTemp']>30)|(data['MinTemp']<-6)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['MinTemp']<=30)&(data['MinTemp']>=-6)]
```

```
NO. of outliers = 76
percentage of the outliers = 0.05%
```

In [63]:
```
# MaxTemp
x=data.loc[(data['MaxTemp']>45)|(data['MaxTemp']<1)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['MaxTemp']<=45)&(data['MaxTemp']>=1)]
```

```
NO. of outliers = 215
percentage of the outliers = 0.15%
```

In [64]:
```
# Rainfall
x=data.loc[data['Rainfall']>55]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['Rainfall']<=55]
```

```
NO. of outliers = 647
percentage of the outliers = 0.45%
```

In [66]:
```
# Evaporation
x=data.loc[(data['Evaporation']>30)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Evaporation']<=30)]
```

```
NO. of outliers = 378
percentage of the outliers = 0.26%
```

In [69]:
```
# WindSpeed3pm
x=data.loc[data['WindSpeed3pm']>45]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindSpeed3pm']<=45]
```

```
NO. of outliers = 0
percentage of the outliers = 0.0%
```

In [67]:
```
# WindGustSpeed
x=data.loc[data['WindGustSpeed']>80]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindGustSpeed']<80]
```

```
NO. of outliers = 445
percentage of the outliers = 0.31%
```

In [70]:
```
# Humidity9am
x=data.loc[data['Humidity9am']<=19]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['Humidity9am']>19]
```

```
NO. of outliers = 1703
percentage of the outliers = 1.21%
```

In [68]:
```
# WindSpeed9am
x=data.loc[data['WindSpeed9am']>40]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindSpeed9am']<=40]
```

```
NO. of outliers = 972
percentage of the outliers = 0.68%
```

In [71]:
```
# Pressure9am
x=data.loc[(data['Pressure9am']>1035)  |  (data['Pressure9am'] <1000)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Pressure9am']<=1035)  &  (data['Pressure9am'] >=1000)]
```

```
NO. of outliers = 1659
percentage of the outliers = 1.19%
```

```
percentage of the outliers = 0.26%
```

In [69]:
```
# WindSpeed3pm
x=data.loc[data['WindSpeed3pm']>45]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindSpeed3pm']<=45]
```

In [67]:
```
# WindGustSpeed
x=data.loc[data['WindGustSpeed']>=80]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindGustSpeed']<80]
```

```
NO. of outliers = 445
percentage of the outliers = 0.31%
NO. of outliers = 0
percentage of the outliers = 0.0%
```

In [70]:
```
# Humidity9am
x=data.loc[data['Humidity9am']<=19]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['Humidity9am']>19]
```

In [68]:
```
# WindSpeed9am
x=data.loc[data['WindSpeed9am']>40]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[data['WindSpeed9am']<=40]
```

```
NO. of outliers = 1703
percentage of the outliers = 1.21%
NO. of outliers = 972
percentage of the outliers = 0.68%
```

In [71]:
```
# Pressure9am
x=data.loc[(data['Pressure9am']>1035)  |  (data['Pressure9am'] <1000)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Pressure9am']<=1035)  &  (data['Pressure9am'] >=1000)]
```

```
NO. of outliers = 1659
percentage of the outliers = 1.19%
```

In [72]:
```
# Pressure3pm
x=data.loc[(data['Pressure3pm']>1032)  |  (data['Pressure3pm'] <1000)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Pressure3pm']<=1032)  &  (data['Pressure3pm'] >=1000)]
```

```
NO. of outliers = 1147
percentage of the outliers = 0.83%
```

In [73]:
```
# Temp9am
x=data.loc[(data['Temp9am']>=35) | (data['Temp9am']<=0)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Temp9am']<35) & (data['Temp9am']>0)]
```

```
NO. of outliers = 334
percentage of the outliers = 0.24%
```

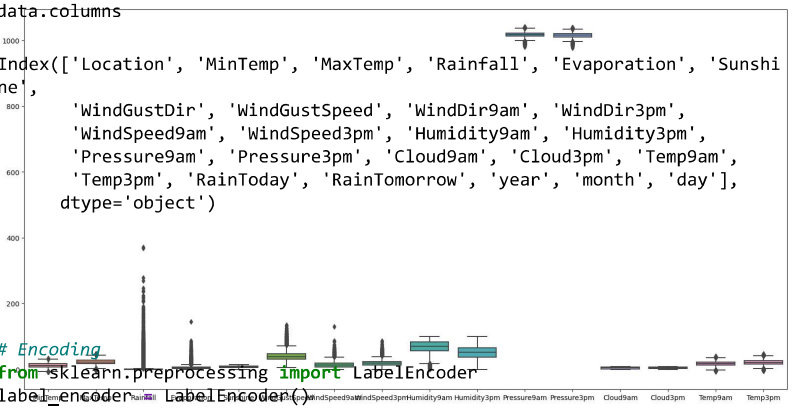
In [74]:
```
# Temp3pm
x=data.loc[(data['Temp3pm']>40) | (data['Temp3pm']<=0)]
print('NO. of outliers =',x.shape[0],'\npercentage of the outliers =',str(rou
data=data.loc[(data['Temp3pm']<=40) & (data['Temp3pm']>0)]
```

```
NO. of outliers = 364
percentage of the outliers = 0.27%
```

In [75]:
```
# show all the outliers as a graph using boxplot for each column in the datas
plt.figure(figsize=(20,10))
sns.boxplot(data=df,orient='v')
plt.show()#
```

In [77]:
```
data.columns
```

Out[77]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshi
ne',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow', 'year', 'month', 'day'],
      dtype='object')
```



En

In [81]:
```
# Encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for i in object_cols:
    data[i] = label_encoder.fit_transform(data[i])
```

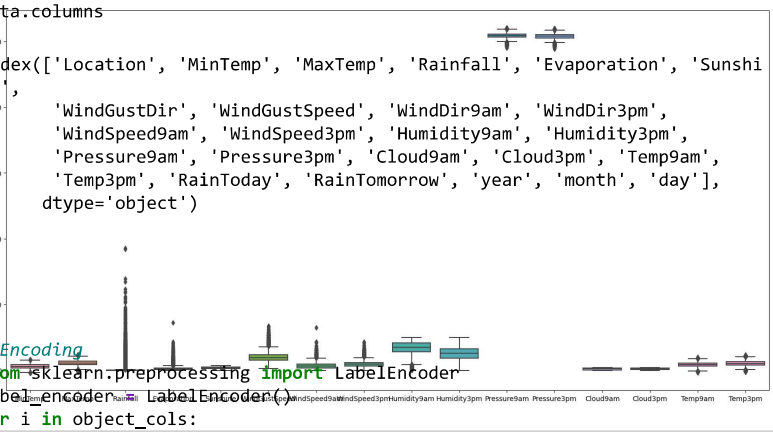
In [76]:
```
data.shape
```

Out[76]:
```
(135877, 25)
```

In [82]:
```
x = data.drop(["RainTomorrow","day"],axis=1).values
y = data["RainTomorrow"].values
```

In [77]:
```
data.columns
```

Out[77]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshi
ne',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow', 'year', 'month', 'day'],
      dtype='object')
```



En

In [81]:
```
# Encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for i in object_cols:
    data[i] = label_encoder.fit_transform(data[i])
```

In [76]:
```
data.shape
```

Out[76]:
```
(135877, 25)
```

In [82]:
```
x = data.drop(["RainTomorrow","day"],axis=1).values
y = data["RainTomorrow"].values
```

Modeling

In [87]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random
```

In [88]:
```
scaler = MinMaxScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [89]:
```
from tensorflow.keras.optimizers import Adam
from keras import regularizers
model = Sequential()

model.add(Dense(units=24,activation='tanh',))

#model.add(Dropout(0.2))

model.add(Dense(units=18,activation='tanh',kernel_regularizer=regularizers.l2

#model.add(Dropout(0.2))

model.add(Dense(units=23,activation='tanh',kernel_regularizer=regularizers.l2

model.add(Dropout(0.5))

model.add(Dense(units=12,activation='tanh'))

model.add(Dropout(0.2))
model.add(Dense(units=1,activation='sigmoid'))
# For a binary classification problem
opt = Adam(learning_rate=0.001)
model.compile(loss='binary_crossentropy', optimizer=opt)
```

In [93]:
```
model.fit(x=x_train,
          y=y_train,
          validation_data=(x_test, y_test), verbose=1,
          callbacks=[early_stop],
          epochs=150,
          batch_size=32
          )
```

In [91]:
```
early_stop = EarlyStopping(monitor='val_loss', mode='min',min_delta=0.001,ve
```

```
Epoch 1/150
3185/3185 [==============================] - 9s 2ms/step - loss: 0.4187 - v
al_loss: 0.3734
Epoch 2/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3817 - v
al_loss: 0.3716
Epoch 3/150
3185/3185 [==============================] - 8s 2ms/step - loss: 0.3784 - v
al_loss: 0.3685
Epoch 4/150
3185/3185 [==============================] - 8s 3ms/step - loss: 0.3757 - v
al_loss: 0.3648
Epoch 5/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3739 - v
al_loss: 0.3708
Epoch 6/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3724 - v
al_loss: 0.3613
```

In [93]:
```
model.add(Dropout(0.2))
model.add(Dense(units=1,activation='sigmoid'))
# For a binary classification problem
opt = Adam(learning_rate=0.001)
model.compile(loss='binary_crossentropy', optimizer=opt)
```

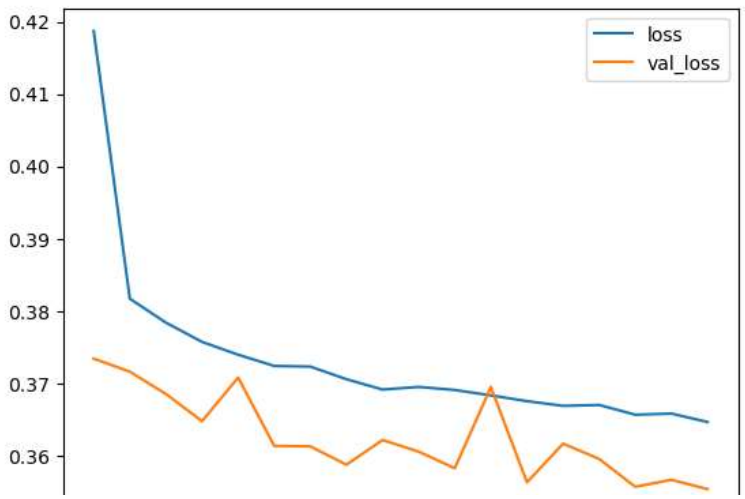
In [91]:
```
early_stop = EarlyStopping(monitor='val_loss', mode='min', min_delta=0.001, ve
model.fit(x=x_train,
          y=y_train,
          validation_data=(x_test, y_test), verbose=1,
          callbacks=[early_stop],
          epochs=150,
          batch_size=32
)
```

```
Epoch 1/150
3185/3185 [==============================] - 9s 2ms/step - loss: 0.4187 - v
al_loss: 0.3734
Epoch 2/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3817 - v
al_loss: 0.3716
Epoch 3/150
3185/3185 [==============================] - 8s 2ms/step - loss: 0.3784 - v
al_loss: 0.3685
Epoch 4/150
3185/3185 [==============================] - 8s 3ms/step - loss: 0.3757 - v
al_loss: 0.3648
Epoch 5/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3739 - v
al_loss: 0.3708
Epoch 6/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3724 - v
al_loss: 0.3613
Epoch 7/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3723 - v
al_loss: 0.3613
Epoch 8/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3706 - v
al_loss: 0.3587
Epoch 9/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3691 - v
al_loss: 0.3622
Epoch 10/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3695 - v
al_loss: 0.3605
Epoch 11/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3691 - v
al_loss: 0.3583
Epoch 12/150
3185/3185 [==============================] - 8s 2ms/step - loss: 0.3683 - v
al_loss: 0.3695
Epoch 13/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3675 - v
al_loss: 0.3563
Epoch 14/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3669 - v
al_loss: 0.3617
Epoch 15/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3670 - v
al_loss: 0.3595
Epoch 16/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3657 - v
al_loss: 0.3557
Epoch 17/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3658 - v
al_loss: 0.3567
Epoch 18/150
3185/3185 [==============================] - ETA: 0s - loss: 0.3646Restorin
g model weights from the end of the best epoch: 13.
```

In [94]:
```
model_loss = pd.DataFrame(model.history.history)
model_loss.plot()
```

Out[94]: <AxesSubplot:>

Out[93]:

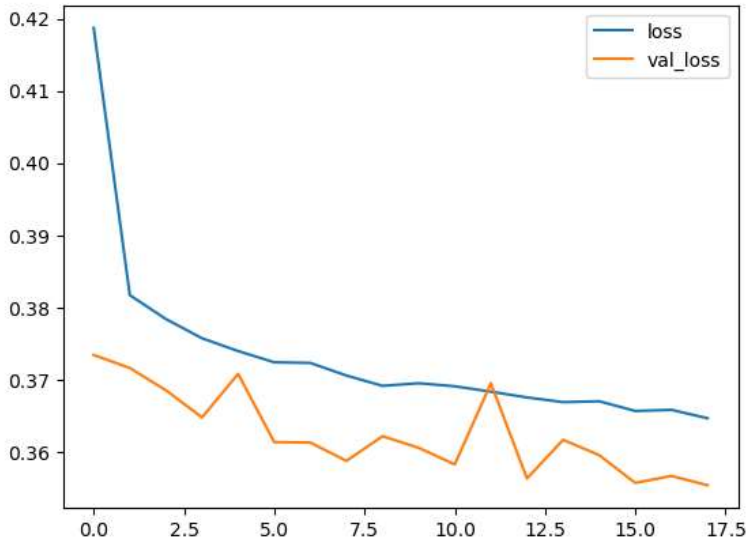


In [94]:
```
model_loss = pd.DataFrame(model.history.history)
model_loss.plot()
```

```
Epoch 17/150
3185/3185 [==============================] - 7s 2ms/step - loss: 0.3658 - val_loss: 0.3567
Epoch 18/150
3185/3185 [==============================] - ETA: 0s - loss: 0.3646Restoring model weights from the end of the best epoch: 13.
```

Out[94]: <Axes: >

Out[93]:



In [95]:
```
prediction = model.predict(x_test)
prediction = (prediction>0.5)
```

```
1062/1062 [==============================] - 1s 1ms/step
```

In [98]:
```
confusion_matrix(y_test,prediction)
```

Out[98]:
```
array([[25359,  1478],
       [ 3626,  3507]], dtype=int64)
```

In [99]:
```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91     26837
           1       0.70      0.49      0.58      7133

    accuracy                           0.85     33970
   macro avg       0.79      0.72      0.74     33970
weighted avg       0.84      0.85      0.84     33970
```

In [101]:
```
mean_squared_error(y_test ,prediction)
```

Out[101]:
```
0.15025022078304387
```

In [102]:
```
mean_absolute_error(y_test , prediction)
```

Out[102]:
```
0.15025022078304387
```

In [103]:
```
r2_score(y_test , prediction)
```

Out[103]:
```
0.09426735242587214
```

In [ ]:
```

```