```
# data analysis
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
# read data
df=pd.read_csv("weatherAUS.csv")
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGus
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	
...	...	...	...	...	...	...	...	
140580	2012-08-08	Darwin	17.8	31.4	0.0	7.0	11.1	
140581	2012-08-09	Darwin	18.4	32.2	0.0	7.4	11.1	
140582	2012-08-10	Darwin	17.3	32.8	0.0	7.6	10.9	
140583	2012-08-11	Darwin	17.7	31.6	0.0	7.4	11.1	
140584	2012-08	NaN	NaN	NaN	NaN	NaN	NaN	

140585 rows × 23 columns

```
# data information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140585 entries, 0 to 140584
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           140585 non-null  object 
 1   Location       140584 non-null  object 
 2   MinTemp        139184 non-null  float64
 3   MaxTemp        139371 non-null  float64
 4   Rainfall       137397 non-null  float64
 5   Evaporation    79490 non-null   float64
 6   Sunshine       73907 non-null   float64
 7   WindGustDir    130377 non-null  object 
 8   WindGustSpeed  130436 non-null  float64
 9   WindDir9am     130137 non-null  object 
 10  WindDir3pm     136409 non-null  object 
 11  WindSpeed9am   138855 non-null  float64
 12  WindSpeed3pm   137559 non-null  float64
 13  Humidity9am    138026 non-null  float64
 14  Humidity3pm    136918 non-null  float64
 15  Pressure9am    125530 non-null  float64
 16  Pressure3pm    125566 non-null  float64
 17  Cloud9am       85968 non-null   float64
 18  Cloud3pm       83112 non-null   float64
 19  Temp9am        138874 non-null  float64
 20  Temp3pm        137704 non-null  float64
 21  RainToday      137397 non-null  object 
 22  RainTomorrow   137394 non-null  object 
dtypes: float64(16), object(7)
memory usage: 24.7+ MB
```

```
# data contain float64(16), object(7) and 140585 entries are available in dataset
# check null values available or not
df.isnull().sum()
```

```
Date                 0
Location             1
MinTemp           1401
MaxTemp           1214
Rainfall          3188
Evaporation      61095
Sunshine         66678
WindGustDir      10208
WindGustSpeed    10149
WindDir9am       10448
WindDir3pm        4176
WindSpeed9am      1730
WindSpeed3pm      3026
Humidity9am       2559
Humidity3pm       3667
Pressure9am      15055
Pressure3pm      15019
Cloud9am         54617
Cloud3pm         57473
Temp9am           1711
Temp3pm           2881
RainToday         3188
RainTomorrow      3191
dtype: int64
```

```
# data shows that so many null values are available in dataset
```

```
# handling missing values
from sklearn.impute import SimpleImputer
si= SimpleImputer(missing_values=np.nan, strategy='mean')
```

```
num_col=df.select_dtypes(["float","int"]).columns
```

```
df[num_col]=si.fit_transform(df[num_col])
```

```
cat_col=df.select_dtypes(["O"]).columns
```

```
si= SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df[cat_col]=si.fit_transform(df[cat_col])
```

```
df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir
0	2008-12-01	Albury	13.4	22.9	0.6	5.413627	7.585605	W
1	2008-12-02	Albury	7.4	25.1	0.0	5.413627	7.585605	WNW
2	2008-12-03	Albury	12.9	25.7	0.0	5.413627	7.585605	WSW
3	2008-12-04	Albury	9.2	28.0	0.0	5.413627	7.585605	NE
4	2008-12-05	Albury	17.5	32.3	1.0	5.413627	7.585605	W

5 rows × 23 columns

```
df.isnull().sum()
```

```
Date             0
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
dtype: int64
```

```
# convert object columns in numerical form
from sklearn.preprocessing import LabelEncoder
lr=LabelEncoder()
```

```
cat_col
```

```
Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm',
       'RainToday', 'RainTomorrow'],
      dtype='object')
```

```
for i in cat_col:
  df[i]=lr.fit_transform(df[i])
```

```
# spliting data into train_test_split
x=df.iloc[:,:-1].values
x
```

```
array([[ 396.        ,    2.        ,   13.4       , ...,   16.9       ,
           21.8       ,    0.        ],
       [ 397.        ,    2.        ,    7.4       , ...,   17.2       ,
           24.3       ,    0.        ],
       [ 398.        ,    2.        ,   12.9       , ...,   21.        ,
           23.2       ,    0.        ],
       ...,
       [1715.        ,   13.        ,   17.3       , ...,   24.1       ,
           32.        ,    0.        ],
       [1716.        ,   13.        ,   17.7       , ...,   21.7       ,
           30.4       ,    0.        ],
       [1705.        ,    9.        ,   11.93831978, ...,   16.70854588,
           21.40531865,    0.        ]])
```

```
y=df.iloc[:,-1].values
y
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
# scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=145)
```

```
# creating a model
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
# using early stoping concept
from tensorflow.keras.callbacks import EarlyStopping
```

```
early_stop=EarlyStopping(monitor='val_loss',mode='min',verbose=1,patience=25)
```

```
# initialize the model
ann=Sequential()
```

```
# add layers to the model
ann.add(Dense(60,activation='relu'))
ann.add(Dense(60,activation='relu'))
# output layer
ann.add(Dense(1,activation='sigmoid'))
```

```
# established the connection between the layers
```

```
ann.compile(optimizer='sgd',loss='binary_crossentropy',metrics=['accuracy'])

# train the model
ann.fit(x_train,y_train,epochs=200,validation_data=(x_test,y_test),batch_size=125,callbacks=[early_stop])
```

```
Epoch 7/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2979 - accuracy: 0.8730 - val_loss: 0.3578 - val_accuracy: 0.848
Epoch 8/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2979 - accuracy: 0.8730 - val_loss: 0.3578 - val_accuracy: 0.848
Epoch 9/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2979 - accuracy: 0.8730 - val_loss: 0.3577 - val_accuracy: 0.848
Epoch 10/200
900/900 [==============================] - 2s 3ms/step - loss: 0.2979 - accuracy: 0.8729 - val_loss: 0.3577 - val_accuracy: 0.848
Epoch 11/200
900/900 [==============================] - 2s 3ms/step - loss: 0.2979 - accuracy: 0.8727 - val_loss: 0.3577 - val_accuracy: 0.848
Epoch 12/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2979 - accuracy: 0.8730 - val_loss: 0.3578 - val_accuracy: 0.848
Epoch 13/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2978 - accuracy: 0.8730 - val_loss: 0.3580 - val_accuracy: 0.848
Epoch 14/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2978 - accuracy: 0.8727 - val_loss: 0.3578 - val_accuracy: 0.848
Epoch 15/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2978 - accuracy: 0.8726 - val_loss: 0.3578 - val_accuracy: 0.848
Epoch 16/200
900/900 [==============================] - 2s 3ms/step - loss: 0.2978 - accuracy: 0.8729 - val_loss: 0.3580 - val_accuracy: 0.848
Epoch 17/200
900/900 [==============================] - 2s 3ms/step - loss: 0.2978 - accuracy: 0.8728 - val_loss: 0.3579 - val_accuracy: 0.847
Epoch 18/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2978 - accuracy: 0.8729 - val_loss: 0.3579 - val_accuracy: 0.848
Epoch 19/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8726 - val_loss: 0.3579 - val_accuracy: 0.847
Epoch 20/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8729 - val_loss: 0.3579 - val_accuracy: 0.848
Epoch 21/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8731 - val_loss: 0.3580 - val_accuracy: 0.848
Epoch 22/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8729 - val_loss: 0.3580 - val_accuracy: 0.848
Epoch 23/200
900/900 [==============================] - 2s 3ms/step - loss: 0.2977 - accuracy: 0.8732 - val_loss: 0.3582 - val_accuracy: 0.848
Epoch 24/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8729 - val_loss: 0.3581 - val_accuracy: 0.848
Epoch 25/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2977 - accuracy: 0.8731 - val_loss: 0.3582 - val_accuracy: 0.847
Epoch 26/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2976 - accuracy: 0.8729 - val_loss: 0.3581 - val_accuracy: 0.848
Epoch 27/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2976 - accuracy: 0.8727 - val_loss: 0.3580 - val_accuracy: 0.847
Epoch 28/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2976 - accuracy: 0.8731 - val_loss: 0.3582 - val_accuracy: 0.847
Epoch 29/200
900/900 [==============================] - 3s 3ms/step - loss: 0.2976 - accuracy: 0.8728 - val_loss: 0.3582 - val_accuracy: 0.848
Epoch 30/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2975 - accuracy: 0.8730 - val_loss: 0.3584 - val_accuracy: 0.848
Epoch 31/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2975 - accuracy: 0.8728 - val_loss: 0.3582 - val_accuracy: 0.847
Epoch 32/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2975 - accuracy: 0.8729 - val_loss: 0.3585 - val_accuracy: 0.847
Epoch 33/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2975 - accuracy: 0.8730 - val_loss: 0.3584 - val_accuracy: 0.848
Epoch 34/200
900/900 [==============================] - 2s 2ms/step - loss: 0.2975 - accuracy: 0.8727 - val_loss: 0.3582 - val_accuracy: 0.847
Epoch 34: early stopping
<keras.callbacks.History at 0x7f27fc4049a0>
```

```
ann.history.history
```

```
        0.35819903016090393],
       'val_accuracy': [0.8496994972229004,
        0.8493082523345947,
        0.8483124375343323,
        0.8493082523345947,
        0.8482056856155396,
        0.8484191298484802,
        0.8486680388450623,
        0.8489170074462891,
        0.8483479619026184,
        0.8485613465309143,
        0.8488814830780029,
        0.8486325144767761,
        0.848845899105072,
        0.8486325144767761,
        0.8485613465309143,
        0.8484546542167664,
        0.8475655317306519,
        0.8485613465309143,
        0.8478856086730957,
        0.8485258221626282,
        0.8481701612472534,
        0.8480989933013916,
        0.8484902381896973,
        0.848774790763855,
        0.8479211926460266,
        0.8486325144767761,
        0.8477789163589478,
        0.8478500843048096,
        0.8481701612472534,
        0.8488814830780029,
        0.8474944233894348,
        0.8477078080177307,
        0.848845899105072,
        0.8472809791564941]}
```

```
ypred=ann.predict(x_test)
ypred=ypred>0.5
```

```
     879/879 [==============================] - 1s 1ms/step
```

```
# evaluation on test data
from sklearn.metrics import classification_report
print(classification_report(ypred,y_test))
```

```
                   precision    recall  f1-score   support

            False       0.93      0.88      0.90     23224
             True       0.55      0.70      0.61      4893

         accuracy                           0.85     28117
        macro avg       0.74      0.79      0.76     28117
     weighted avg       0.87      0.85      0.85     28117
```

```
# model predicted 85% accuracy.
```