```
import pandas as pd   #importing librariees
import numpy as np

import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense



df=pd.read_csv("weatherAUS.csv") #reading file data
df.head()
```

:ion	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
bury	13.4	22.9	0.6	NaN	NaN
bury	7.4	25.1	0.0	NaN	NaN
bury	12.9	25.7	0.0	NaN	NaN
bury	9.2	28.0	0.0	NaN	NaN

```
from sklearn.impute import SimpleImputer

si=SimpleImputer(missing_values=np.nan,strategy="mean")
col=df.select_dtypes(["float","int"]).columns



df[col]=si.fit_transform(df[col])#filled null values with mean


col=df.select_dtypes("object").columns
si=SimpleImputer(missing_values=np.nan,strategy="most_frequent")



df[col]=si.fit_transform(df[col])


from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
col=df.select_dtypes("object").columns



df[col]=oe.fit_transform(df[col])#performed encoding on object columns


x=df.iloc[:,:-1].values#selecting features
x
```

```
array([[3.960e+02, 2.000e+00, 1.340e+01, ..., 1.690e+01, 2.180e+01,
        0.000e+00],
       [3.970e+02, 2.000e+00, 7.400e+00, ..., 1.720e+01, 2.430e+01,
        0.000e+00],
       [3.980e+02, 2.000e+00, 1.290e+01, ..., 2.100e+01, 2.320e+01,
        0.000e+00],
       ...,
       [3.433e+03, 4.100e+01, 5.400e+00, ..., 1.250e+01, 2.610e+01,
        0.000e+00],
       [3.434e+03, 4.100e+01, 7.800e+00, ..., 1.510e+01, 2.600e+01,
        0.000e+00],
       [3.435e+03, 4.100e+01, 1.490e+01, ..., 1.500e+01, 2.090e+01,
        0.000e+00]])
```

```
y=df.iloc[:,-1].values #seperating target
y=y.astype(int)
y
```

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

```
from sklearn.preprocessing import StandardScaler#performing scalling standard scaler
sc=StandardScaler()
x=sc.fit_transform(x)


from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)#splitting data into test and training data


from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=25)#performing early stopping

#earlystopping: EarlyStopping  can be applied at the certain stages to reduce the overfiiting problem


ann = Sequential()



ann.add(Dense(20,activation='relu'))#hidden layer
ann.add(Dense(20,activation='relu'))
ann.add(Dense(1,activation='sigmoid')) #Output layer (Since it's a binary classification problem)



#Using accuracy as loss function
ann.compile(optimizer='sgd',loss='binary_crossentropy',metrics=['accuracy'])


ann.fit(xtrain,ytrain,epochs=200,batch_size=128,callbacks=[early_stop])
```

```
ann.history.history
```

```
       0.8501011729240417,
       0.849796712398529,
       0.8505823612213135,
       0.8504449129104614,
       0.8502091765403748,
       0.8502877354621887,
       0.8506020307540894,
       0.8503565192222595,
       0.8504449129104614,
       0.8503565192222595,
       0.8505529165267944,
       0.8504154086112976,
       0.8506118655204773,
       0.8505627512931824,
       0.8507689833641052,
       0.8508377075195312,
       0.8508279323577881,
       0.8505136370658875,
       0.850945770740509,
       0.8504841923713684,
       0.8508475422859192,
       0.8508475422859192,
       0.8509261012077332,
       0.8508377075195312,
       0.8505136370658875,
       0.8512600660324097,
       0.8507493734359741,
       0.8508475422859192,
       0.850877046585083,
       0.8506904244422913,
       0.8507591485977173,
       0.8508672118186951,
       0.8512305617332458,
       0.8511814475059509,
       0.8510931134223938,
       0.8511912822723389,
       0.8511225581169128,
       0.8512895107269287,
       0.8512207865715027,
       0.8512796759605408,
       0.85106360912323,
       0.8513189554214478,
       0.8510047197341919,
       0.8513484597206116,
       0.851102888584137,
       0.8513091206550598,
       0.8512895107269287,
       0.8512600660324097,
       0.8515350222587585,
       0.8516626954078674,
       0.8515743017196655,
       0.8519082069396973,
       0.8516234159469604,
       0.8513484597206116,
       0.8513680696487427,
       0.8516528606414795,
       0.8514957427978516,
       0.8515448570251465,
```

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

```
     1364/1364 [==============================] - 2s 1ms/step
```

```
ypred
```

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

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
                precision    recall  f1-score   support

             0       0.87      0.95      0.91     34215
             1       0.73      0.51      0.60      9423
```

```
              accuracy                           0.85     43638
             macro avg       0.80      0.73      0.75     43638
          weighted avg       0.84      0.85      0.84     43638
```