Australia Rain Prediction

```
from google .colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df=pd.read_csv("/content/drive/MyDrive/Dataset/weatherAUS.csv")
```

```
df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindD
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	

5 rows × 23 columns

```
df.isnull().sum()/len(df)*100
```

```
Date            0.000000
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
dtype: float64
```

```
df["Location"].value_counts()
```

```
Canberra            3436
Sydney              3344
Darwin              3193
Melbourne           3193
Brisbane            3193
Adelaide            3193
Perth               3193
Hobart              3193
Albany              3040
MountGambier        3040
Ballarat            3040
Townsville          3040
GoldCoast           3040
Cairns              3040
Launceston          3040
AliceSprings        3040
Bendigo             3040
Albury              3040
MountGinini         3040
Wollongong          3040
Newcastle           3039
Tuggeranong         3039
Penrith             3039
Woomera             3009
Nuriootpa           3009
Cobar               3009
CoffsHarbour        3009
Moree               3009
Sale                3009
PerthAirport        3009
PearceRAAF          3009
```

```
Witchcliffe         3009
BadgerysCreek       3009
Mildura             3009
NorfolkIsland       3009
MelbourneAirport    3009
Richmond            3009
SydneyAirport       3009
WaggaWagga          3009
Williamtown         3009
Dartmoor            3009
Watsonia            3009
Portland            3009
Walpole             3006
NorahHead           3004
SalmonGums          3001
Katherine           1578
Nhil                1578
Uluru               1578
Name: Location, dtype: int64
```

```
df["Location"].duplicated()
```

```
0         False
1          True
2          True
3          True
4          True
          ...
145455     True
145456     True
145457     True
145458     True
145459     True
Name: Location, Length: 145460, dtype: bool
```

```
n_cols=df.select_dtypes(include=['int','float']).columns
n_cols
```

```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
       'Temp9am', 'Temp3pm'],
      dtype='object')
```

```
c_cols=df.select_dtypes(include=['object']).columns
c_cols
```

```
Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm',
       'RainToday', 'RainTomorrow'],
```

```
          dtype='object')
```

```
for i in c_cols:
    print(i, df[i].isnull().sum())
```

```
Date 0
Location 0
WindGustDir 10326
WindDir9am 10566
WindDir3pm 4228
RainToday 3261
RainTomorrow 3267
```

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

```
for i in n_cols:
    print(i, df[i].isnull().sum())
```

```
MinTemp 1485
MaxTemp 1261
Rainfall 3261
Evaporation 62790
Sunshine 69835
WindGustSpeed 10263
WindSpeed9am 1767
WindSpeed3pm 3062
Humidity9am 2654
Humidity3pm 4507
Pressure9am 15065
Pressure3pm 15028
Cloud9am 55888
Cloud3pm 59358
Temp9am 1767
Temp3pm 3609
```

```
for i in n_cols:
    df[i].fillna(df[i].median(), inplace=True)
```

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

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
```

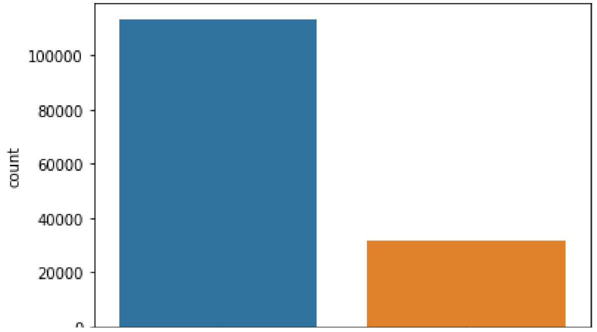
```
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

```
df.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        145460 non-null  float64
 3   MaxTemp        145460 non-null  float64
 4   Rainfall       145460 non-null  float64
 5   Evaporation    145460 non-null  float64
 6   Sunshine       145460 non-null  float64
 7   WindGustDir    145460 non-null  object 
 8   WindGustSpeed  145460 non-null  float64
 9   WindDir9am     145460 non-null  object 
 10  WindDir3pm     145460 non-null  object 
 11  WindSpeed9am   145460 non-null  float64
 12  WindSpeed3pm   145460 non-null  float64
 13  Humidity9am    145460 non-null  float64
 14  Humidity3pm    145460 non-null  float64
 15  Pressure9am    145460 non-null  float64
 16  Pressure3pm    145460 non-null  float64
 17  Cloud9am       145460 non-null  float64
 18  Cloud3pm       145460 non-null  float64
 19  Temp9am        145460 non-null  float64
 20  Temp3pm        145460 non-null  float64
 21  RainToday      145460 non-null  object 
 22  RainTomorrow   145460 non-null  object 
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

```
sns.countplot(df.RainTomorrow)
```

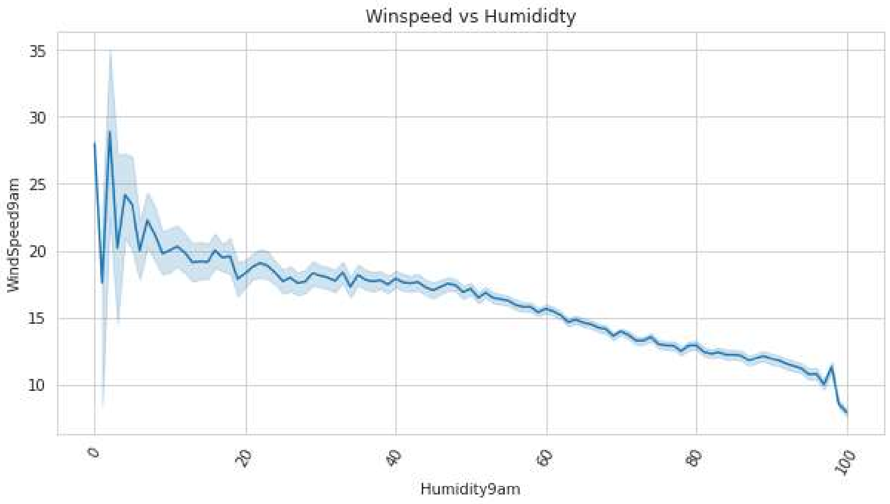
```
<AxesSubplot:xlabel='RainTomorrow', ylabel='count'>
```




```
#Dataset is imbalanced
```

What is the Relationship between Windspeed vs Humididty ?

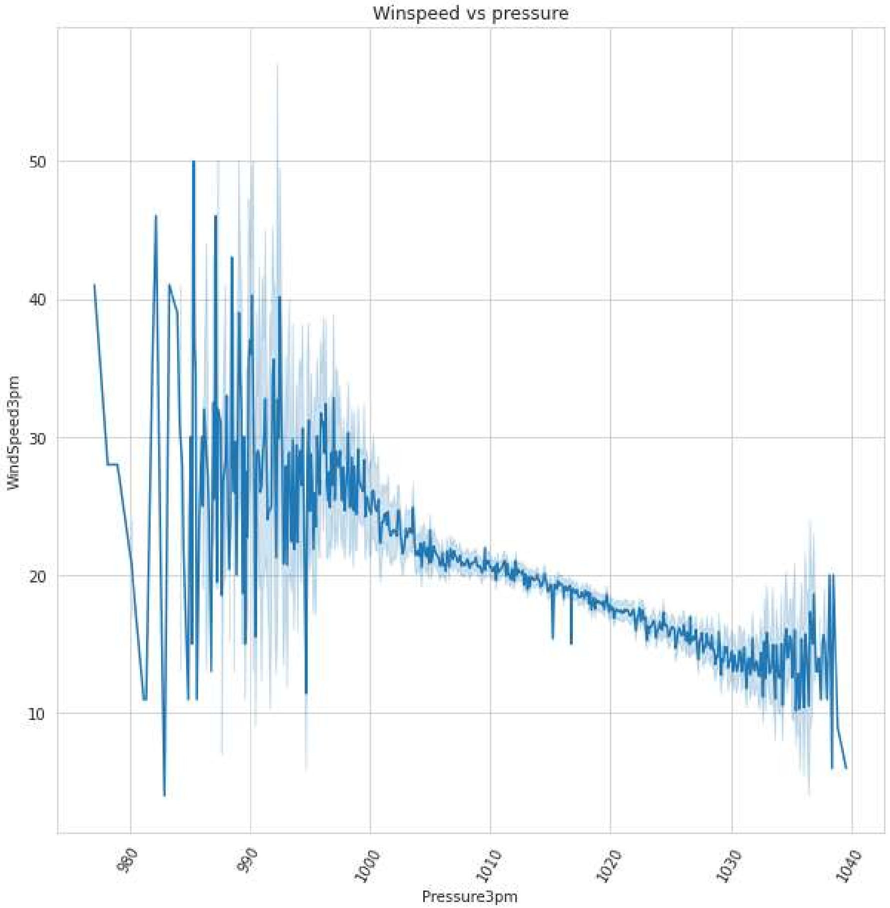
```
sns.set_style(style="whitegrid")
fig_dims= (10,5)
fig, ax=plt.subplots(figsize = fig_dims)
fig=sns.lineplot(y=df["WindSpeed9am"],x=df["Humidity9am"],ax=ax).set(title="Winspeed vs Humididty")
plt.xticks(rotation=60)
plt.show()
```



Windspeed is inversely proportional to as Humidity increases windspeed decreases

What is the Relationship between Windspeed and Pressure ?

```
fig_dims= (10,10)
fig, ax=plt.subplots(figsize = fig_dims)
fig=sns.lineplot(x=df["Pressure3pm"],y=df["WindSpeed3pm"],ax=ax).set(title="Winspeed vs pressure")
plt.xticks(rotation=60)
plt.show()
```

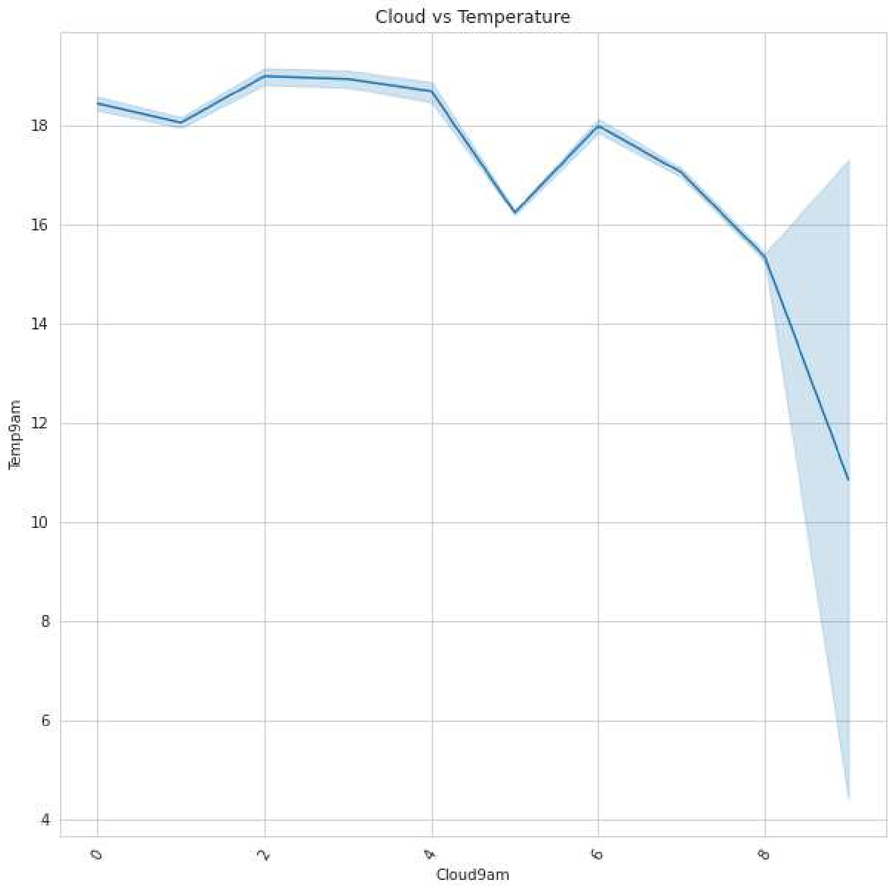


Windspeed is inversely proportional to pressure. As prerssure increases windspeed decreases.

windspeed is maximum when pressure is between in 990-1000

What is relationship between cloud and temperature ?

```
fig_dims= (10,10)
fig, ax=plt.subplots(figsize = fig_dims)
fig=sns.lineplot(x=df["Cloud9am"],y=df["Temp9am"],ax=ax).set(title="Cloud vs Temperature")
plt.xticks(rotation=60)
plt.show()
```



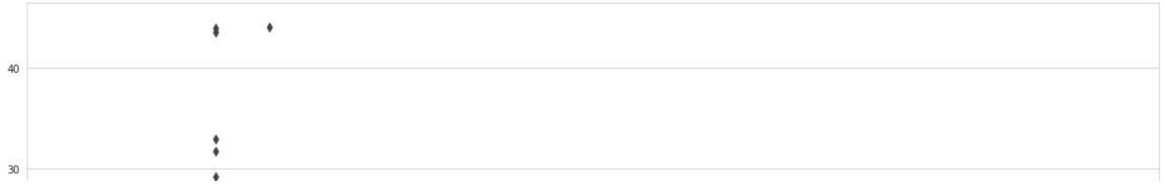
As no. of clouds increases in sky Temperature decreases

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for i in c_cols:
    df[i] = label_encoder.fit_transform(df[i])
```

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

```
col_names = list(X.columns)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X=sc.fit_transform(X)
X= pd.DataFrame(X, columns=col_names)
```

```
plt.figure(figsize=(20,10))
sns.boxenplot(data =X)
plt.xticks(rotation=90)
plt.show()
```



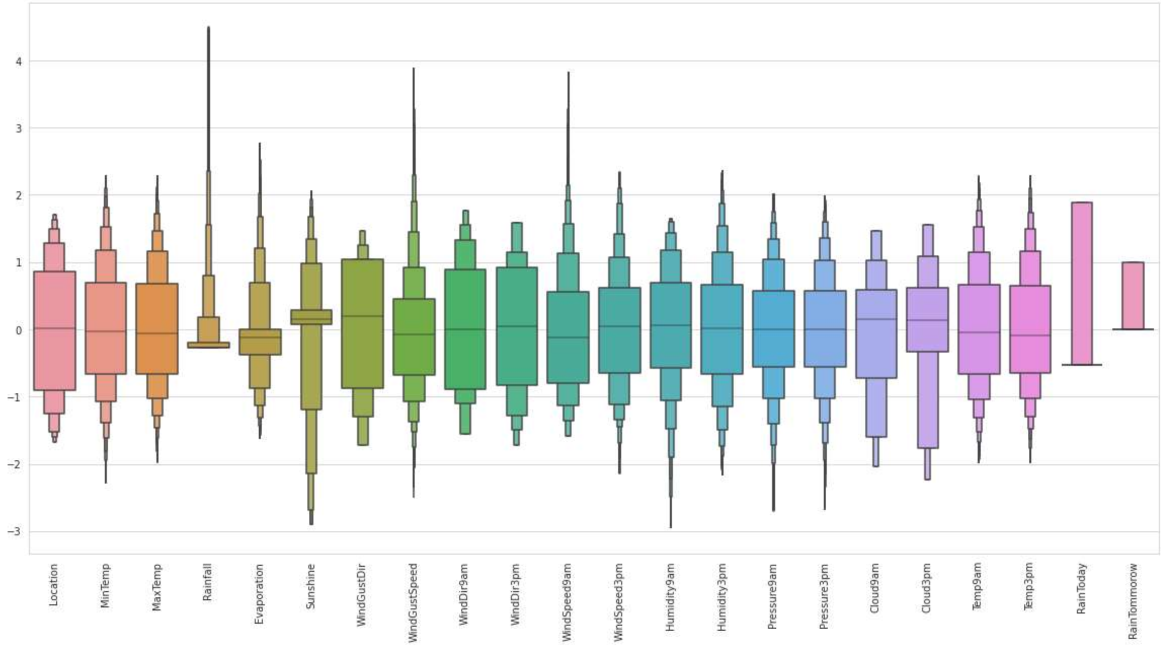

```
X["RainTommorow"]=y
```

Removing Outliers

```
X = X[(X["MinTemp"]<2.3)&(X["MinTemp"]>-2.3)]
X = X[(X["MaxTemp"]<2.3)&(X["MaxTemp"]>-2)]
X = X[(X["Rainfall"]<4.5)]
X = X[(X["Evaporation"]<2.8)]
X = X[(X["Sunshine"]<2.1)]
X = X[(X["WindGustSpeed"]<4)&(X["WindGustSpeed"]>-4)]
X = X[(X["WindSpeed9am"]<4)]
X = X[(X["WindSpeed3pm"]<2.5)]
X = X[(X["Humidity9am"]>-3)]
X = X[(X["Humidity3pm"]>-2.2)]
X = X[(X["Pressure9am"]< 2)&(X["Pressure9am"]>-2.7)]
X = X[(X["Pressure3pm"]< 2)&(X["Pressure3pm"]>-2.7)]
X = X[(X["Cloud9am"]<1.8)]
X = X[(X["Cloud3pm"]<2)]
X = X[(X["Temp9am"]<2.3)&(X["Temp9am"]>-2)]
X = X[(X["Temp3pm"]<2.3)&(X["Temp3pm"]>-2)]
X.shape
```

```
(127536, 22)
```

```
plt.figure(figsize=(20,10))
sns.boxenplot(data = X)
plt.xticks(rotation=90)
plt.show()
```



MODEL BUILDING

```
x=X.drop(["RainTommorow"],axis=1)
y=X["RainTommorow"]
```

```
#from imblearn.over_sampling import SMOTE
#sm=SMOTE(sampling_strategy="auto")
#X,y=sm.fit_resample(x,y)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
```

```
X_train.shape
```

```
(89275, 21)
```

```
X_test.shape
```

```
(38261, 21)
```

```
y_train.shape
```

```
(89275,)
```

```
y_test.shape
```

```
(38261,)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Dropout
from sklearn.metrics import classification_report
from tensorflow.keras.callbacks import EarlyStopping
```

```
ann=Sequential()
```

```
early_stop=EarlyStopping(monitor="val_loss",mode="min",verbose=2,patience=30)
```

```
ann.add(Dense(units=32,activation="relu"))
ann.add(Dense(units=32,activation="relu"))
ann.add(Dense(units=16,activation="relu"))
ann.add(Dense(units=8,activation="relu"))
ann.add(Dense(units=1,activation="sigmoid"))
ann.compile(optimizer="Adam",loss="binary_crossentropy",metrics=["accuracy"])
history = ann.fit(X_train, y_train, batch_size = 32, epochs = 150, callbacks=[early_stop], validation_split=0.2)
```

```
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3353 - accuracy: 0.8562 - val_loss: 0.3446 - val_accuracy: 0.8517
Epoch 17/150
2232/2232 [==============================] - 8s 3ms/step - loss: 0.3344 - accuracy: 0.8564 - val_loss: 0.3430 - val_accuracy: 0.8530
Epoch 18/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3345 - accuracy: 0.8575 - val_loss: 0.3435 - val_accuracy: 0.8529
Epoch 19/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3332 - accuracy: 0.8574 - val_loss: 0.3446 - val_accuracy: 0.8506
Epoch 20/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3329 - accuracy: 0.8583 - val_loss: 0.3418 - val_accuracy: 0.8531
Epoch 21/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3321 - accuracy: 0.8583 - val_loss: 0.3434 - val_accuracy: 0.8515
Epoch 22/150
2232/2232 [==============================] - 9s 4ms/step - loss: 0.3317 - accuracy: 0.8587 - val_loss: 0.3420 - val_accuracy: 0.8530
Epoch 23/150
2232/2232 [==============================] - 6s 3ms/step - loss: 0.3310 - accuracy: 0.8593 - val_loss: 0.3426 - val_accuracy: 0.8518
Epoch 24/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3310 - accuracy: 0.8583 - val_loss: 0.3442 - val_accuracy: 0.8511
Epoch 25/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3304 - accuracy: 0.8586 - val_loss: 0.3486 - val_accuracy: 0.8484
Epoch 26/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3293 - accuracy: 0.8601 - val_loss: 0.3444 - val_accuracy: 0.8521
Epoch 27/150
2232/2232 [==============================] - 6s 3ms/step - loss: 0.3296 - accuracy: 0.8593 - val_loss: 0.3475 - val_accuracy: 0.8506
Epoch 28/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3289 - accuracy: 0.8595 - val_loss: 0.3448 - val_accuracy: 0.8516
Epoch 29/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3282 - accuracy: 0.8608 - val_loss: 0.3437 - val_accuracy: 0.8519
Epoch 30/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3277 - accuracy: 0.8599 - val_loss: 0.3447 - val_accuracy: 0.8509
Epoch 31/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3275 - accuracy: 0.8606 - val_loss: 0.3434 - val_accuracy: 0.8521
Epoch 32/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3269 - accuracy: 0.8604 - val_loss: 0.3492 - val_accuracy: 0.8520
Epoch 33/150
2232/2232 [==============================] - 7s 3ms/step - loss: 0.3264 - accuracy: 0.8608 - val_loss: 0.3478 - val_accuracy: 0.8500
Epoch 34/150
2232/2232 [==============================] - 8s 4ms/step - loss: 0.3263 - accuracy: 0.8610 - val_loss: 0.3471 - val_accuracy: 0.8511
Epoch 35/150
2232/2232 [==============================] - 6s 3ms/step - loss: 0.3257 - accuracy: 0.8615 - val_loss: 0.3484 - val_accuracy: 0.8496
Epoch 36/150
2232/2232 [==============================] - 9s 4ms/step - loss: 0.3252 - accuracy: 0.8619 - val_loss: 0.3467 - val_accuracy: 0.8502
Epoch 37/150
2232/2232 [==============================] - 6s 3ms/step - loss: 0.3253 - accuracy: 0.8615 - val_loss: 0.3466 - val_accuracy: 0.8525
Epoch 37: early stopping
```

```
history.history
```

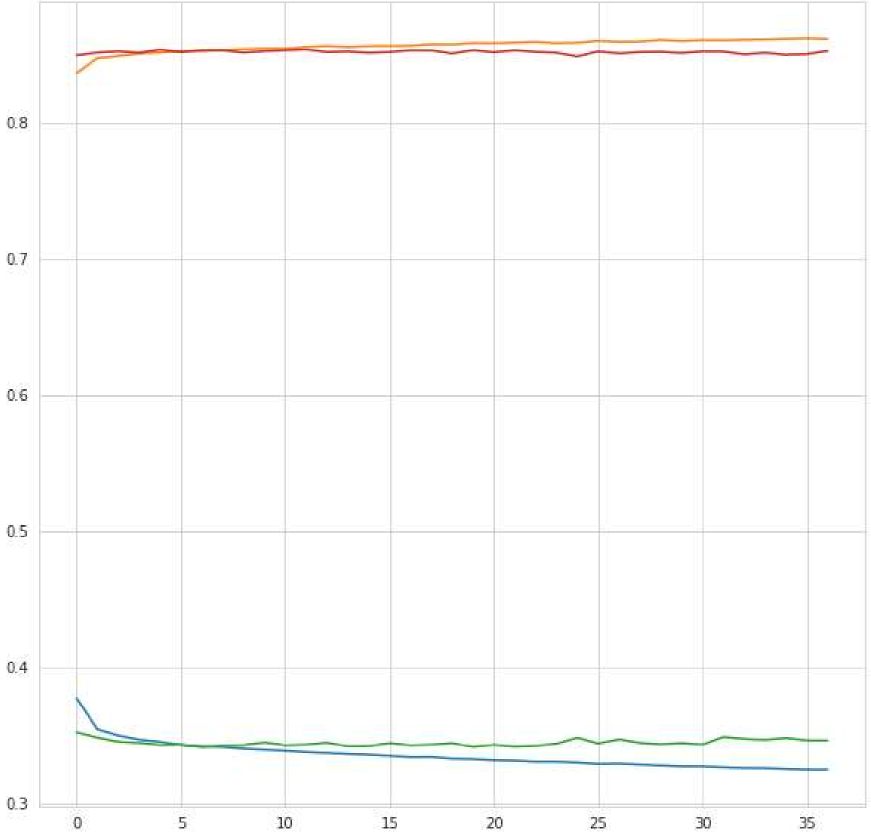
```
        0.348621666431427,
        0.34437859058380127,
        0.34748080372810364,
        0.3448055386543274,
        0.3436725437641144,
        0.3447219133377075,
        0.34342554211616516,
        0.34919166564941406,
        0.3477797508239746,
        0.3470659852027893,
        0.3484296202659607,
        0.3466717302799225,
        0.3466234505176544],
       'val_accuracy': [0.8492299318313599,
        0.8513021469116211,
        0.8521982431411743,
        0.8511341214179993,
        0.8535424470901489,
        0.8516381978988647,
        0.8530383706092834,
        0.8529823422431946,
        0.851246178150177,
        0.8523662686347961,
        0.8530383706092834,
        0.853598415851593,
        0.8516942262649536,
        0.8520862460136414,
        0.8510781526565552,
        0.8516942262649536,
        0.8529823422431946,
        0.8529263734817505,
        0.8505740761756897,
        0.8530943989753723,
        0.8514701724052429,
        0.8530383706092834,
        0.8517501950263977,
        0.8511341214179993,
        0.8483898043632507,
        0.8520862460136414,
        0.8506301045417786,
        0.8516381978988647,
        0.8518622517585754,
        0.8509101271629333,
        0.8521422743797302,
        0.8520302176475525,
        0.8499580025672913,
        0.8511341214179993,
        0.8496219515800476,
        0.8501819968223572,
        0.8524783253669739]}
```

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

	loss	accuracy	val_loss	val_accuracy
0	0.377514	0.836222	0.352580	0.849230
1	0.354860	0.847172	0.348762	0.851302
2	0.350205	0.848740	0.345587	0.852198
3	0.347281	0.850406	0.344894	0.851134
4	0.345506	0.851316	0.343107	0.853542
5	0.343365	0.852408	0.343373	0.851638
6	0.342280	0.852366	0.341738	0.853038
7	0.341732	0.853150	0.342804	0.852982
8	0.340510	0.853864	0.343072	0.851246
9	0.339752	0.854285	0.345163	0.852366
10	0.338998	0.854257	0.342990	0.853038
11	0.337967	0.855447	0.343480	0.853598
12	0.337329	0.856175	0.344989	0.851694
13	0.336670	0.855447	0.342314	0.852086
14	0.336069	0.856175	0.342381	0.851078
15	0.335250	0.856245	0.344591	0.851694
16	0.334365	0.856385	0.342997	0.852982
17	0.334465	0.857491	0.343453	0.852926
18	0.333191	0.857379	0.344590	0.850574
19	0.332880	0.858345	0.341850	0.853094
20	0.332074	0.858303	0.343374	0.851470
21	0.331749	0.858695	0.341997	0.853038
22	0.330958	0.859269	0.342613	0.851750
23	0.331020	0.858289	0.344165	0.851134
24	0.330393	0.858569	0.348622	0.848390
25	0.329319	0.860053	0.344379	0.852086
26	0.329581	0.859311	0.347481	0.850630
27	0.328904	0.859493	0.344806	0.851638
28	0.328241	0.860753	0.343673	0.851862
29	0.327658	0.859913	0.344722	0.850910

```
plt.figure(figsize=(10,10))
plt.plot(loss_df)
```

```
[<matplotlib.lines.Line2D at 0x7f74e4573dc0>,
 <matplotlib.lines.Line2D at 0x7f74e4573e20>,
 <matplotlib.lines.Line2D at 0x7f74e4573f40>,
 <matplotlib.lines.Line2D at 0x7f74e7fca0a0>]
```




```
y_pred=ann.predict(X_test)
```

```
1196/1196 [==============================] - 2s 2ms/step
```

```
y_pred
```

```
array([[0.00664576],
       [0.00179289],
       [0.10076873],
       ...,
       [0.02231799],
```

```
       [0.9395832 ],
       [0.02993768]], dtype=float32)
```

```
y_pred=np.where(y_pred<0.5,0,1)
```

```
print(classification_report(y_test,y_pred))
```

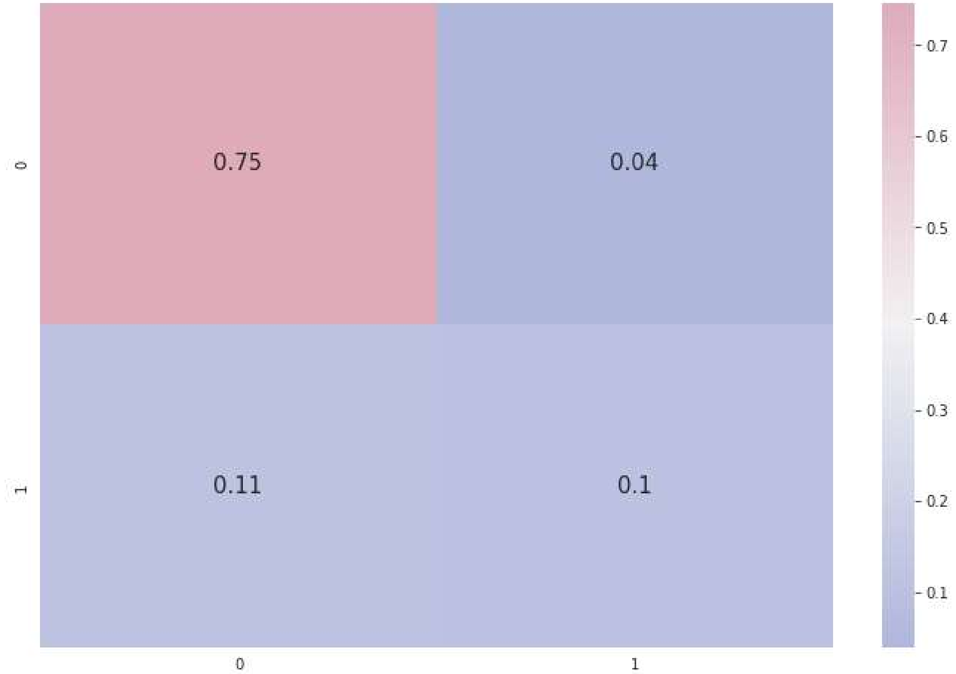
```
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     30065
           1       0.72      0.49      0.58      8196

    accuracy                           0.85     38261
   macro avg       0.80      0.72      0.75     38261
weighted avg       0.84      0.85      0.84     38261
```

```
# confusion matrix
cmap1 = sns.diverging_palette(260,-10,s=50, l=75, n=5, as_cmap=True)
plt.subplots(figsize=(12,8))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), cmap = cmap1, annot = True, annot_kws = {'size':15})
```

```
<AxesSubplot:>
```



This shows that Model has Good True positive Rate

0s completed at 3:03 PM