In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

In [2]:
```
df=pd.read_csv('weatherAUS.csv')
```

In [3]:
```
df
```

Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	Win
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	
...	...	...	...	...	...	...	...	...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	

145460 rows × 23 columns

In [4]:
```
df.drop(['Date'],axis=1,inplace=True)
```

In [5]:
```
df
```

Out[5]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustS
0	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	Albury	17.5	32.3	1.0	NaN	NaN	W	
...	...	...	...	...	...	...	...	
145455	Uluru	2.8	23.4	0.0	NaN	NaN	E	
145456	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	
145457	Uluru	5.4	26.9	0.0	NaN	NaN	N	
145458	Uluru	7.8	27.0	0.0	NaN	NaN	SE	
145459	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	

145460 rows × 22 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 22 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Location       145460 non-null  object 
 1   MinTemp        143975 non-null  float64
 2   MaxTemp        144199 non-null  float64
 3   Rainfall       142199 non-null  float64
 4   Evaporation    82670 non-null   float64
 5   Sunshine       75625 non-null   float64
 6   WindGustDir    135134 non-null  object 
 7   WindGustSpeed  135197 non-null  float64
 8   WindDir9am     134894 non-null  object 
 9   WindDir3pm     141232 non-null  object 
 10  WindSpeed9am   143693 non-null  float64
 11  WindSpeed3pm   142398 non-null  float64
 12  Humidity9am    142806 non-null  float64
 13  Humidity3pm    140953 non-null  float64
 14  Pressure9am    130395 non-null  float64
 15  Pressure3pm    130432 non-null  float64
 16  Cloud9am       89572 non-null   float64
 17  Cloud3pm       86102 non-null   float64
 18  Temp9am        143693 non-null  float64
 19  Temp3pm        141851 non-null  float64
 20  RainToday      142199 non-null  object 
 21  RainTomorrow   142193 non-null  object 
dtypes: float64(16), object(6)
memory usage: 24.4+ MB
```

Checking for Null values

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

As we can see there are a lot of missing values present in almost every column. We will drop the columns which have more than 30% missing values

In [8]:
```
df.drop(['Evaporation','Sunshine','Cloud9am','Cloud3pm'],axis=1,inplace=True)
```

We have dropped the columns which had more than 30% missing values

In [9]:
```
df.describe()
```

Out[9]:

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpee
count	143975.000000	144199.000000	142199.000000	135197.000000	143693.000000	142398.0(
mean	12.194034	23.221348	2.360918	40.035230	14.043426	18.6(
std	6.398495	7.119049	8.478060	13.607062	8.915375	8.8(
min	-8.500000	-4.800000	0.000000	6.000000	0.000000	0.0(
25%	7.600000	17.900000	0.000000	31.000000	7.000000	13.0(
50%	12.000000	22.600000	0.000000	39.000000	13.000000	19.0(
75%	16.900000	28.200000	0.800000	48.000000	19.000000	24.0(
max	33.900000	48.100000	371.000000	135.000000	130.000000	87.0(

Checking Skewness

In [10]:
```
from scipy.stats import skew
```

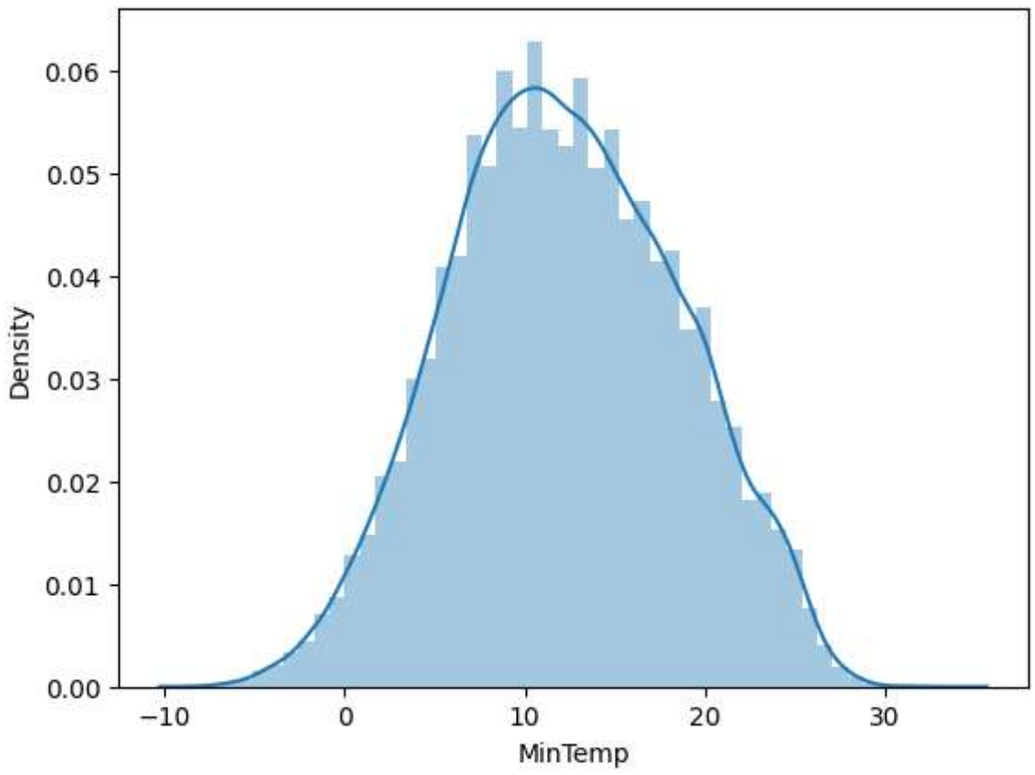
In [11]:
```
colname=df.select_dtypes('float64').columns
```

In [12]:
```
colname
```

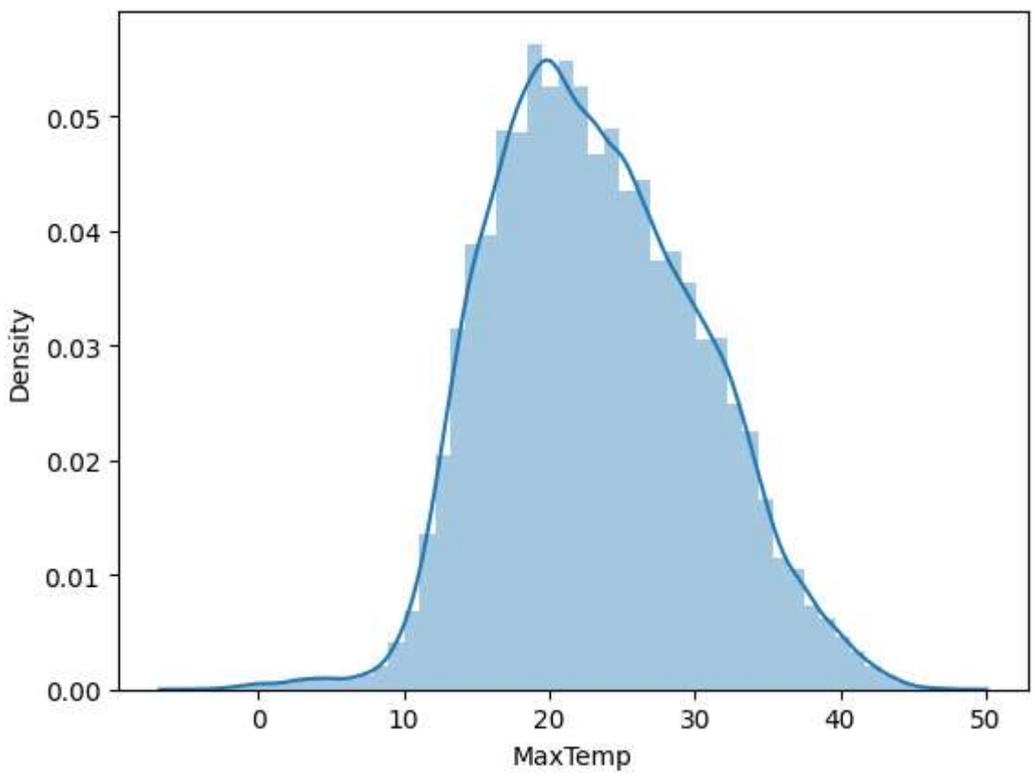
Out[12]:
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am',
       'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Temp9am', 'Temp3pm'],
      dtype='object')
```

In [13]:
```
for i in df[colname]:
    print(i)
    print(skew(df[i]))
    sns.distplot(df[i])
    plt.show()
```

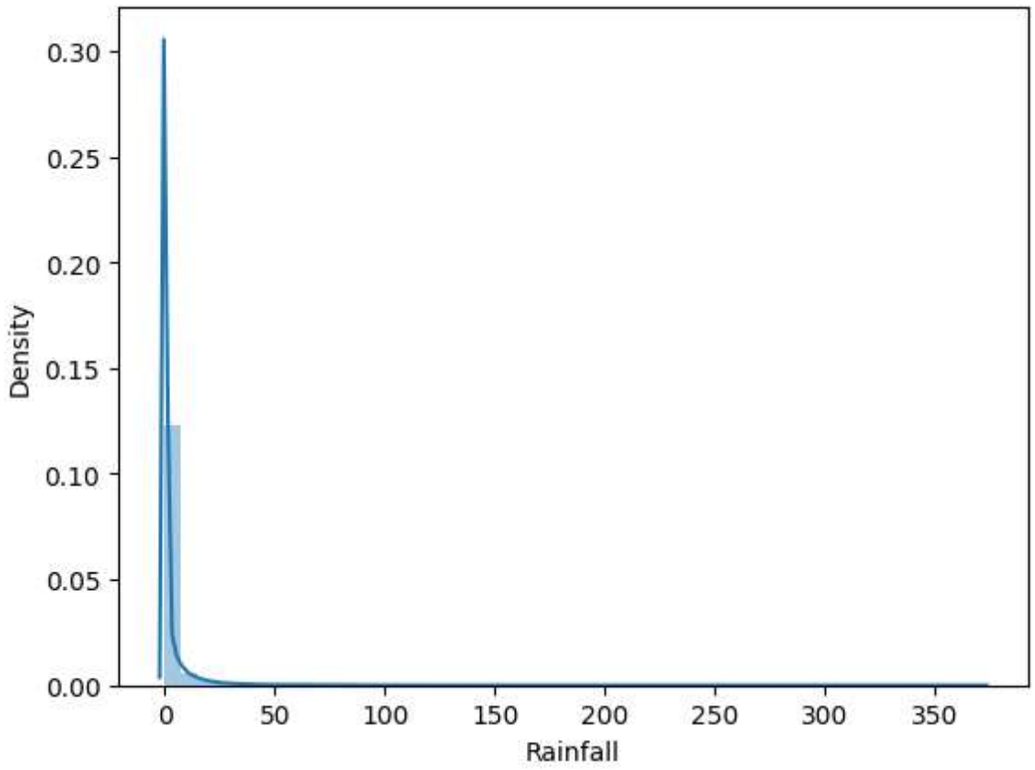
```
MinTemp
nan
```



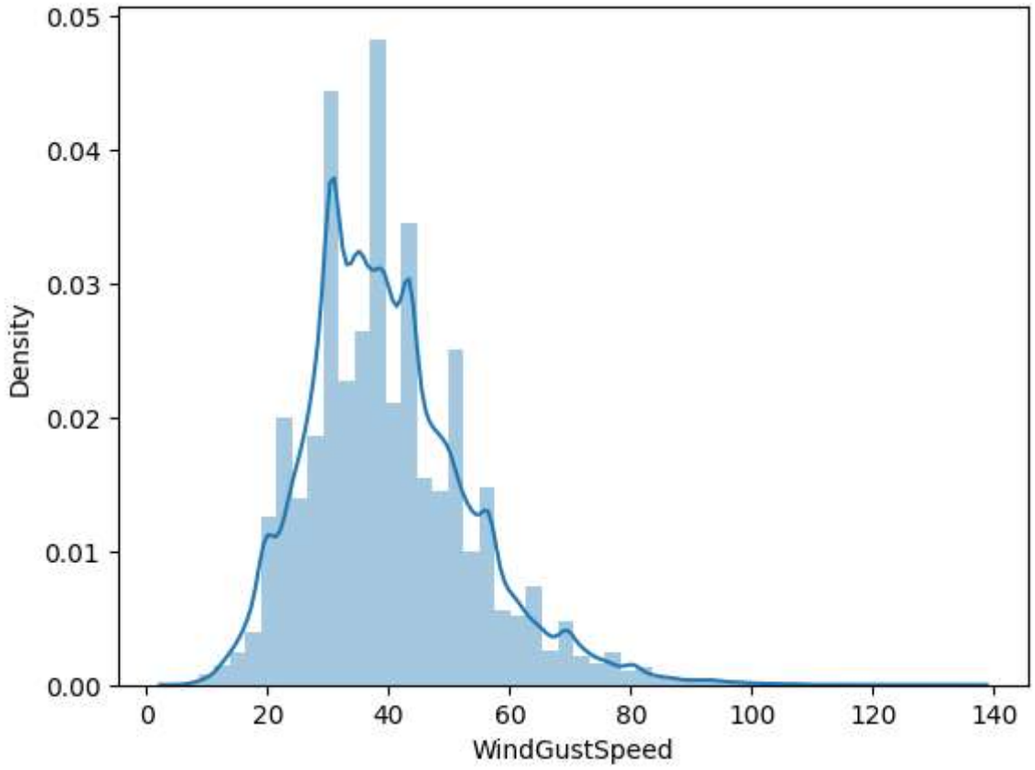

```
MaxTemp
nan
```



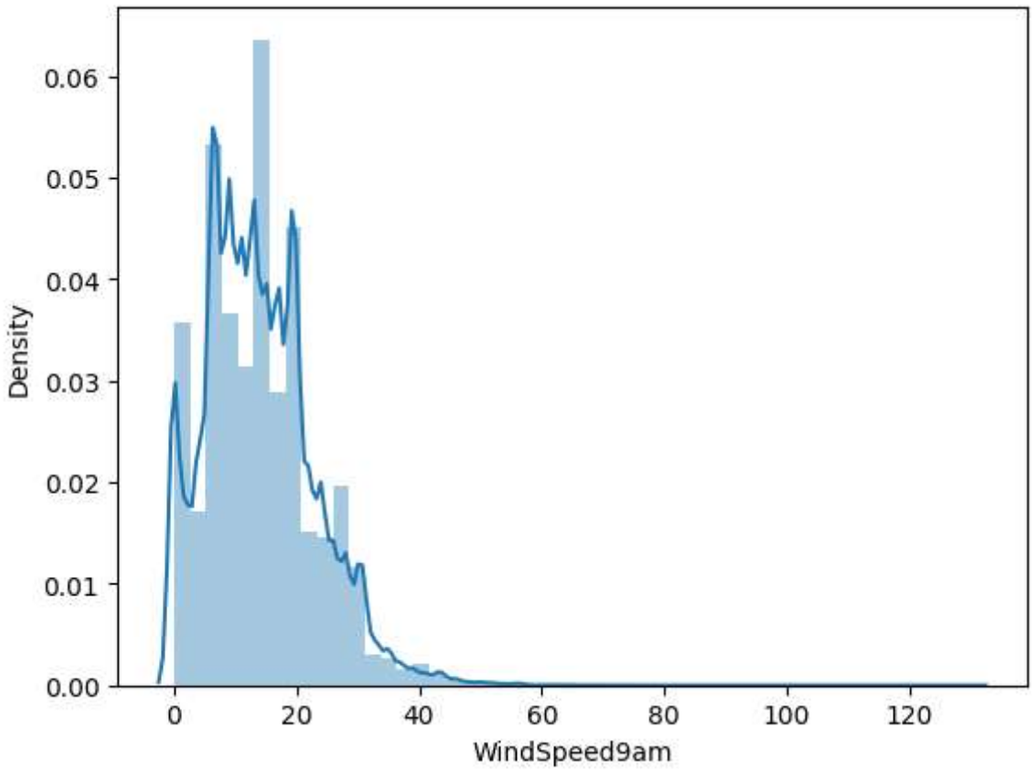

```
Rainfall
nan
```



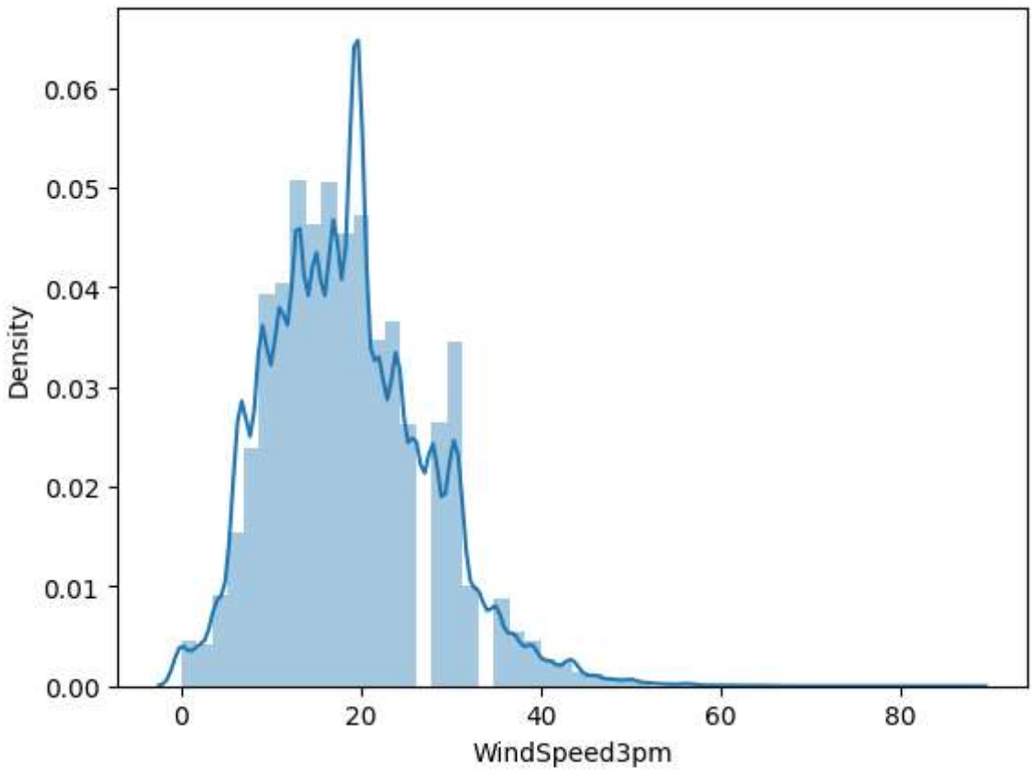

```
WindGustSpeed
nan
```



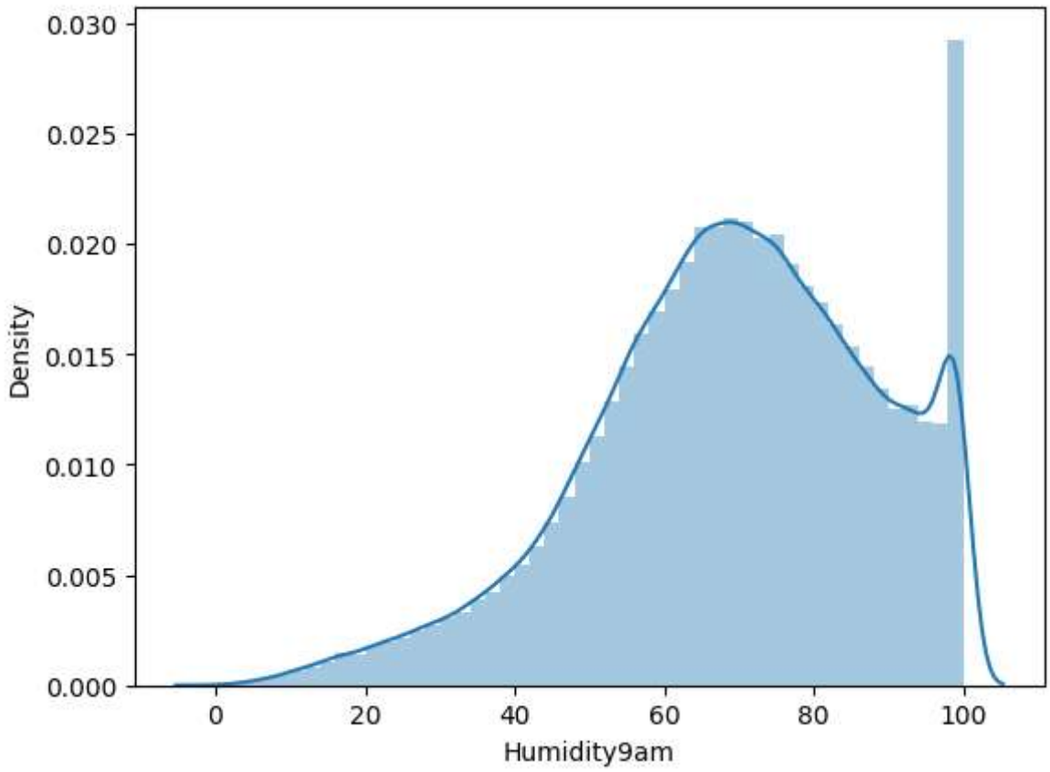

```
WindSpeed9am
nan
```



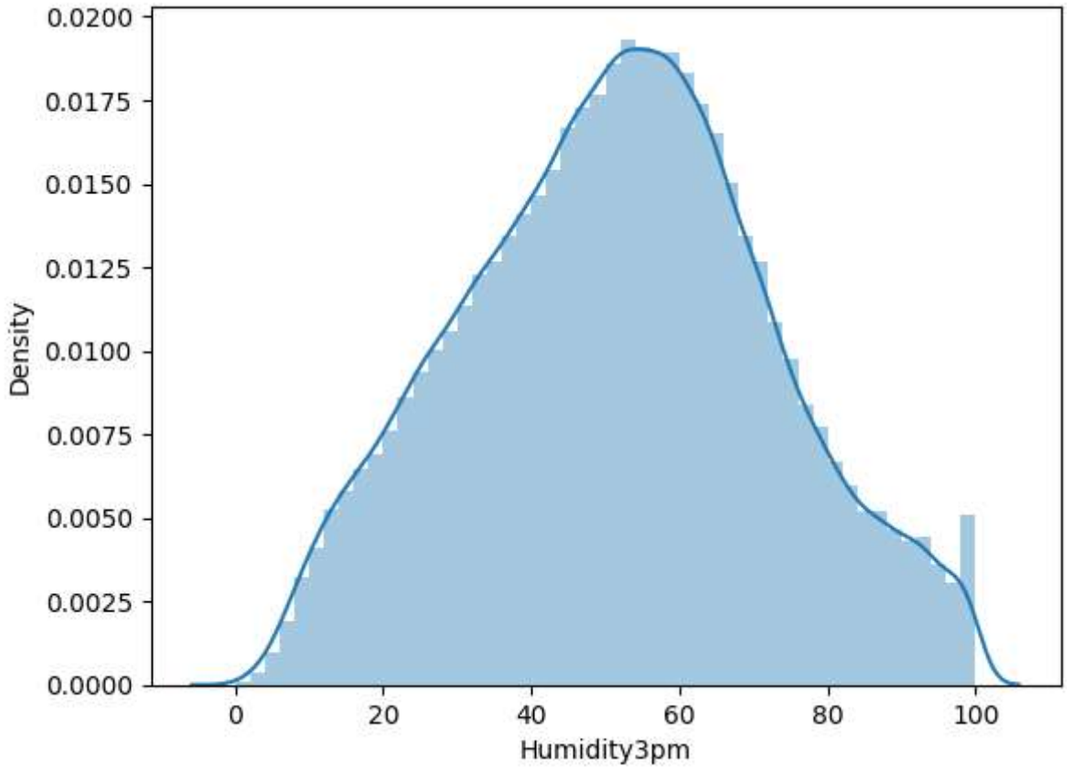

```
WindSpeed3pm
nan
```



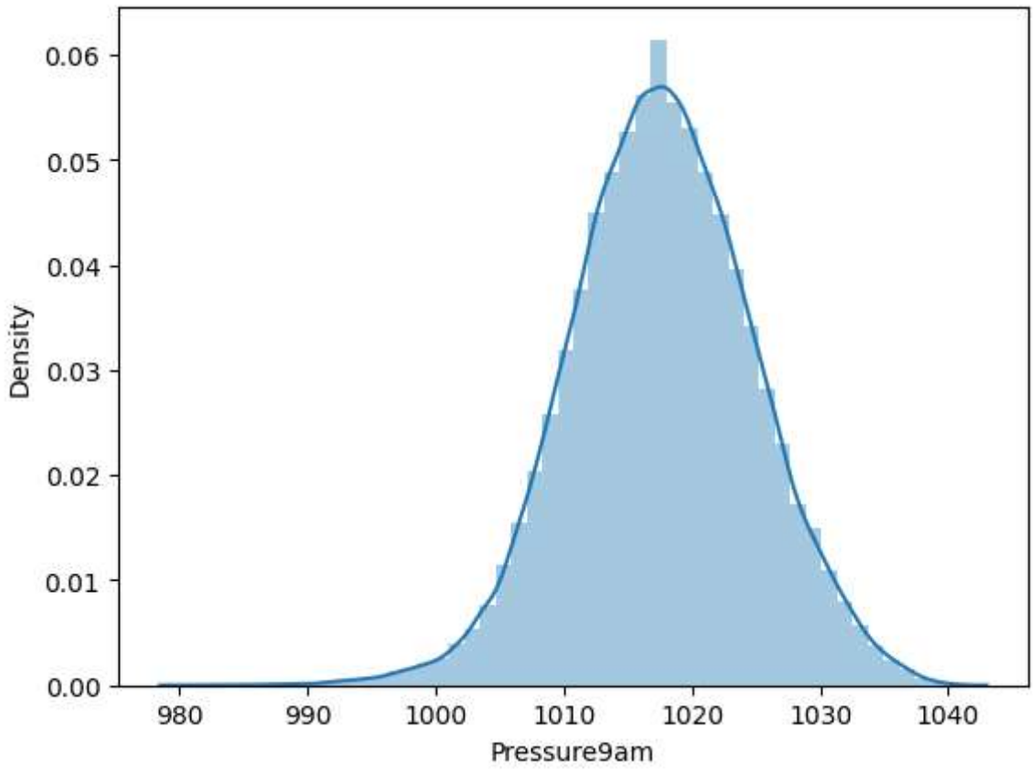

```
Humidity9am
nan
```



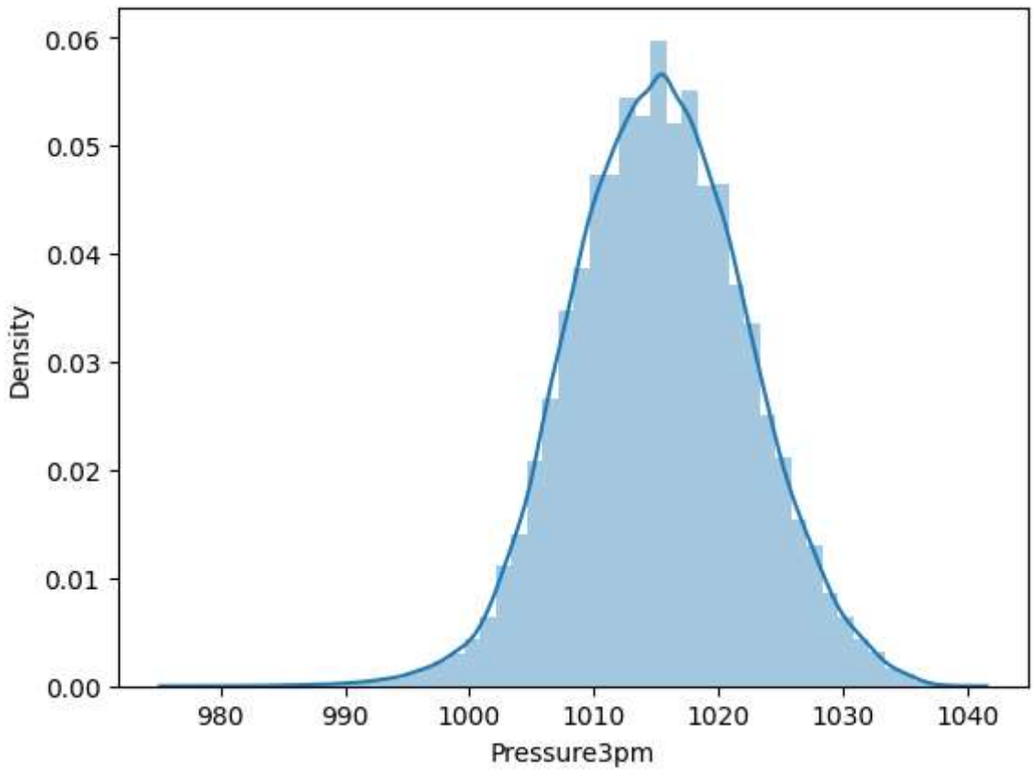

```
Humidity3pm
nan
```



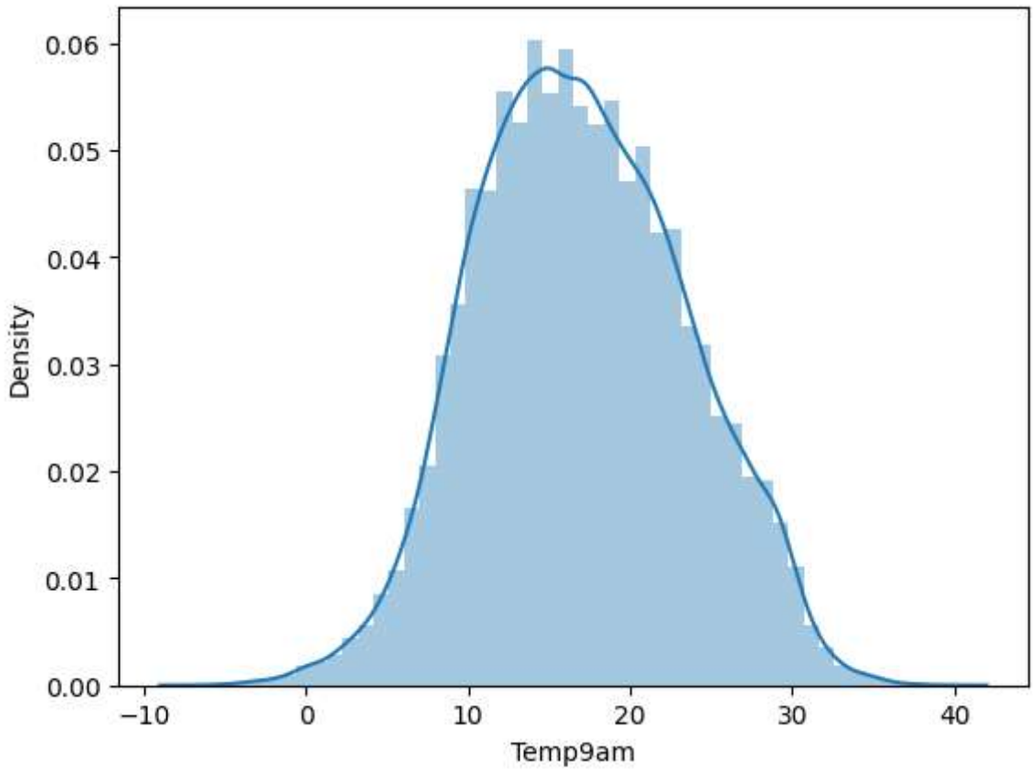

```
Pressure9am
nan
```



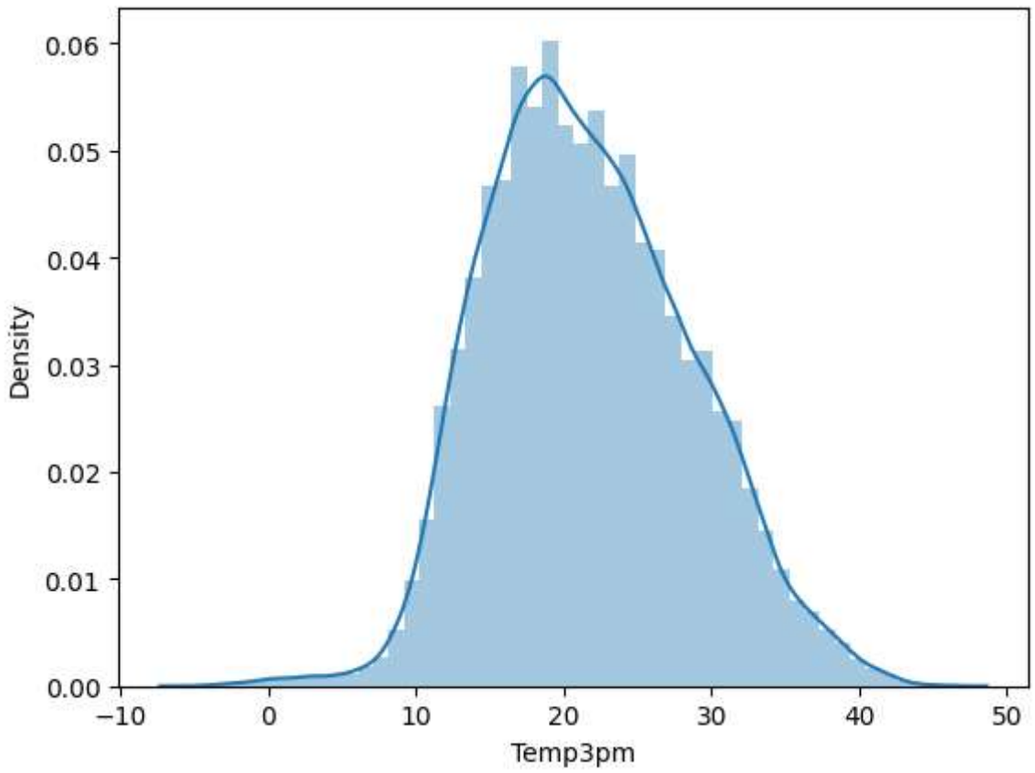

```
Pressure3pm
nan
```




```
Temp9am
nan
```




```
Temp3pm
nan
```



Fill the missing values in columns which lies between 3% to 30% with Simple Imputer

In [14]:
```
from sklearn.impute import SimpleImputer
si=SimpleImputer(missing_values=np.nan,strategy='mean')
df[['WindGustSpeed','Pressure9am','Pressure3pm']]=si.fit_transform(df[['WindGu
```

In [16]:
```
si1=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
df[['WindGustDir','WindDir9am']]=si1.fit_transform(df[['WindGustDir','WindDir9
```

As all remaining columns have missins values less than 3% we will drop the missing values

using dropna method

In [17]:
```
df.dropna(inplace=True)
```

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Here all the missing values have been dropped and now no missing values are present in the dataset

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134590 entries, 0 to 145458
Data columns (total 18 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Location       134590 non-null  object 
 1   MinTemp        134590 non-null  float64
 2   MaxTemp        134590 non-null  float64
 3   Rainfall       134590 non-null  float64
 4   WindGustDir    134590 non-null  object 
 5   WindGustSpeed  134590 non-null  float64
 6   WindDir9am     134590 non-null  object 
 7   WindDir3pm     134590 non-null  object 
 8   WindSpeed9am   134590 non-null  float64
 9   WindSpeed3pm   134590 non-null  float64
 10  Humidity9am    134590 non-null  float64
 11  Humidity3pm    134590 non-null  float64
 12  Pressure9am    134590 non-null  float64
 13  Pressure3pm    134590 non-null  float64
 14  Temp9am        134590 non-null  float64
 15  Temp3pm        134590 non-null  float64
 16  RainToday      134590 non-null  object 
 17  RainTomorrow   134590 non-null  object 
dtypes: float64(12), object(6)
memory usage: 19.5+ MB
```

Converting categorical into numerical data using encoding

In [20]:
```
df
```

Out[20]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	Winc
0	Albury	13.4	22.9	0.6	W	44.0	W	
1	Albury	7.4	25.1	0.0	WNW	44.0	NNW	
2	Albury	12.9	25.7	0.0	WSW	46.0	W	
3	Albury	9.2	28.0	0.0	NE	24.0	SE	
4	Albury	17.5	32.3	1.0	W	41.0	ENE	
...	...	...	...	...	...	...	...	
145454	Uluru	3.5	21.8	0.0	E	31.0	ESE	
145455	Uluru	2.8	23.4	0.0	E	31.0	SE	
145456	Uluru	3.6	25.3	0.0	NNW	22.0	SE	
145457	Uluru	5.4	26.9	0.0	N	37.0	SE	
145458	Uluru	7.8	27.0	0.0	SE	28.0	SSE	

134590 rows × 18 columns

In [22]:
```
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
df[['Location','WindGustDir','WindDir9am','WindDir3pm','RainToday']]=
oe.fit_transform(df[['Location','WindGustDir','WindDir9am','WindDir3pm','RainT
```

In [23]:
```
df
```

Out[23]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	Winc
0	2.0	13.4	22.9	0.6	13.0	44.0	13.0	
1	2.0	7.4	25.1	0.0	14.0	44.0	6.0	
2	2.0	12.9	25.7	0.0	15.0	46.0	13.0	
3	2.0	9.2	28.0	0.0	4.0	24.0	9.0	
4	2.0	17.5	32.3	1.0	13.0	41.0	1.0	
...	...	...	...	...	...	...	...	
145454	41.0	3.5	21.8	0.0	0.0	31.0	2.0	
145455	41.0	2.8	23.4	0.0	0.0	31.0	9.0	
145456	41.0	3.6	25.3	0.0	6.0	22.0	9.0	
145457	41.0	5.4	26.9	0.0	3.0	37.0	9.0	
145458	41.0	7.8	27.0	0.0	9.0	28.0	10.0	

134590 rows × 18 columns

In [24]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['RainTomorrow']=le.fit_transform(df['RainTomorrow'])
```

In [25]:
```
df['RainTomorrow'].unique()
```

Out[25]:
```
array([0, 1])
```

In [26]:
```
df
```

Out[26]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	Winc
0	2.0	13.4	22.9	0.6	13.0	44.0	13.0	
1	2.0	7.4	25.1	0.0	14.0	44.0	6.0	
2	2.0	12.9	25.7	0.0	15.0	46.0	13.0	
3	2.0	9.2	28.0	0.0	4.0	24.0	9.0	
4	2.0	17.5	32.3	1.0	13.0	41.0	1.0	
...	...	...	...	...	...	...	...	
145454	41.0	3.5	21.8	0.0	0.0	31.0	2.0	
145455	41.0	2.8	23.4	0.0	0.0	31.0	9.0	
145456	41.0	3.6	25.3	0.0	6.0	22.0	9.0	
145457	41.0	5.4	26.9	0.0	3.0	37.0	9.0	
145458	41.0	7.8	27.0	0.0	9.0	28.0	10.0	

134590 rows × 18 columns

Splitting the dataset

In [27]:
```
x=df.iloc[:,0:-1].values
x
```

Out[27]:
```
array([[ 2. , 13.4, 22.9, ..., 16.9, 21.8,  0. ],
       [ 2. ,  7.4, 25.1, ..., 17.2, 24.3,  0. ],
       [ 2. , 12.9, 25.7, ..., 21. , 23.2,  0. ],
       ...,
       [41. ,  3.6, 25.3, ..., 10.9, 24.5,  0. ],
       [41. ,  5.4, 26.9, ..., 12.5, 26.1,  0. ],
       [41. ,  7.8, 27. , ..., 15.1, 26. ,  0. ]])
```

In [28]:
```
y=df.iloc[:,-1].values
y
```

Out[28]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [29]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

In [30]:
```
x
```

Out[30]:
```
array([[-1.5337994 ,  0.18986408, -0.05108984, ..., -0.01212636,
         0.00729032, -0.5340412 ],
       [-1.5337994 , -0.75122871,  0.2604199 , ...,  0.03423448,
         0.36840422, -0.5340412 ],
       [-1.5337994 ,  0.11143968,  0.3453771 , ...,  0.6214718 ,
         0.2095141 , -0.5340412 ],
       ...,
       [ 1.19646447, -1.34725415,  0.28873897, ..., -0.93934318,
         0.39729333, -0.5340412 ],
       [ 1.19646447, -1.06492631,  0.5152915 , ..., -0.69208536,
         0.62840622, -0.5340412 ],
       [ 1.19646447, -0.68848919,  0.52945103, ..., -0.29029141,
         0.61396167, -0.5340412 ]])
```

Splitting the dataset into training and testing dataset

In [31]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

Building the model

In [32]:
```
ann=Sequential()
```

Adding hidden layers to the model

In [33]:
```
ann.add(Dense(150,activation='relu'))
ann.add(Dropout(rate=0.2))

ann.add(Dense(150,activation='relu'))
ann.add(Dropout(rate=0.2))

ann.add(Dense(1,activation='sigmoid'))
```

Establishing the connection between the layers

In [34]:
```
ann.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

Importing earlystopping library

In [35]:
```
from tensorflow.keras.callbacks import EarlyStopping
es=EarlyStopping(monitor='val_loss',mode='min',patience=25)
```

Training the model

In [36]:
```
ann.fit(xtrain,ytrain,validation_data=(xtest,ytest),epochs=300,batch_size=100,
        callbacks=[es])
```

```
Epoch 1/300
1077/1077 [==============================] - 3s 2ms/step - loss: 0.3658 - acc
uracy: 0.8437 - val_loss: 0.3456 - val_accuracy: 0.8523
Epoch 2/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3515 - acc
uracy: 0.8502 - val_loss: 0.3389 - val_accuracy: 0.8537
Epoch 3/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3484 - acc
uracy: 0.8513 - val_loss: 0.3380 - val_accuracy: 0.8566
Epoch 4/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3452 - acc
uracy: 0.8522 - val_loss: 0.3347 - val_accuracy: 0.8568
Epoch 5/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3438 - acc
uracy: 0.8532 - val_loss: 0.3358 - val_accuracy: 0.8571
Epoch 6/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3415 - acc
uracy: 0.8543 - val_loss: 0.3329 - val_accuracy: 0.8573
Epoch 7/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3400 - acc
uracy: 0.8550 - val_loss: 0.3305 - val_accuracy: 0.8593
Epoch 8/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3391 - acc
uracy: 0.8550 - val_loss: 0.3309 - val_accuracy: 0.8598
Epoch 9/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3378 - acc
uracy: 0.8560 - val_loss: 0.3328 - val_accuracy: 0.8582
Epoch 10/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3363 - acc
uracy: 0.8564 - val_loss: 0.3297 - val_accuracy: 0.8599
Epoch 11/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3355 - acc
uracy: 0.8559 - val_loss: 0.3308 - val_accuracy: 0.8599
Epoch 12/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3350 - acc
uracy: 0.8568 - val_loss: 0.3298 - val_accuracy: 0.8588
Epoch 13/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3339 - acc
uracy: 0.8576 - val_loss: 0.3287 - val_accuracy: 0.8598
Epoch 14/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3337 - acc
uracy: 0.8575 - val_loss: 0.3283 - val_accuracy: 0.8604
Epoch 15/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3320 - acc
uracy: 0.8582 - val_loss: 0.3305 - val_accuracy: 0.8590
Epoch 16/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3322 - acc
uracy: 0.8578 - val_loss: 0.3274 - val_accuracy: 0.8596
Epoch 17/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3314 - acc
uracy: 0.8586 - val_loss: 0.3283 - val_accuracy: 0.8592
Epoch 18/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3307 - acc
uracy: 0.8589 - val_loss: 0.3272 - val_accuracy: 0.8596
Epoch 19/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3297 - acc
uracy: 0.8587 - val_loss: 0.3280 - val_accuracy: 0.8603
Epoch 20/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3292 - acc
uracy: 0.8600 - val_loss: 0.3276 - val_accuracy: 0.8591
Epoch 21/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3290 - acc
uracy: 0.8592 - val_loss: 0.3273 - val_accuracy: 0.8590
Epoch 22/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3284 - acc
uracy: 0.8601 - val_loss: 0.3266 - val_accuracy: 0.8598
Epoch 23/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3277 - acc
uracy: 0.8599 - val_loss: 0.3276 - val_accuracy: 0.8608
Epoch 24/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3281 - acc
uracy: 0.8596 - val_loss: 0.3269 - val_accuracy: 0.8601
Epoch 25/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3264 - acc
```

```
uracy: 0.8614 - val_loss: 0.3264 - val_accuracy: 0.8608
Epoch 26/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3266 - acc
uracy: 0.8604 - val_loss: 0.3270 - val_accuracy: 0.8595
Epoch 27/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3262 - acc
uracy: 0.8607 - val_loss: 0.3282 - val_accuracy: 0.8590
Epoch 28/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3252 - acc
uracy: 0.8608 - val_loss: 0.3267 - val_accuracy: 0.8607
Epoch 29/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3257 - acc
uracy: 0.8606 - val_loss: 0.3264 - val_accuracy: 0.8608
Epoch 30/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3242 - acc
uracy: 0.8609 - val_loss: 0.3262 - val_accuracy: 0.8604
Epoch 31/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3244 - acc
uracy: 0.8615 - val_loss: 0.3260 - val_accuracy: 0.8604
Epoch 32/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3236 - acc
uracy: 0.8620 - val_loss: 0.3261 - val_accuracy: 0.8600
Epoch 33/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3234 - acc
uracy: 0.8626 - val_loss: 0.3260 - val_accuracy: 0.8602
Epoch 34/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3220 - acc
uracy: 0.8615 - val_loss: 0.3258 - val_accuracy: 0.8609
Epoch 35/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3234 - acc
uracy: 0.8625 - val_loss: 0.3260 - val_accuracy: 0.8611
Epoch 36/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3229 - acc
uracy: 0.8632 - val_loss: 0.3255 - val_accuracy: 0.8614
Epoch 37/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3225 - acc
uracy: 0.8629 - val_loss: 0.3256 - val_accuracy: 0.8615
Epoch 38/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3209 - acc
uracy: 0.8638 - val_loss: 0.3258 - val_accuracy: 0.8620
Epoch 39/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3216 - acc
uracy: 0.8635 - val_loss: 0.3260 - val_accuracy: 0.8592
Epoch 40/300
1077/1077 [==============================] - 3s 2ms/step - loss: 0.3207 - acc
uracy: 0.8646 - val_loss: 0.3263 - val_accuracy: 0.8608
Epoch 41/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3211 - acc
uracy: 0.8637 - val_loss: 0.3261 - val_accuracy: 0.8606
Epoch 42/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3208 - acc
uracy: 0.8629 - val_loss: 0.3260 - val_accuracy: 0.8612
Epoch 43/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3202 - acc
uracy: 0.8640 - val_loss: 0.3292 - val_accuracy: 0.8599
Epoch 44/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3204 - acc
uracy: 0.8633 - val_loss: 0.3263 - val_accuracy: 0.8606
Epoch 45/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3192 - acc
uracy: 0.8635 - val_loss: 0.3251 - val_accuracy: 0.8607
Epoch 46/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3194 - acc
uracy: 0.8647 - val_loss: 0.3252 - val_accuracy: 0.8624
Epoch 47/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3189 - acc
uracy: 0.8646 - val_loss: 0.3255 - val_accuracy: 0.8610
Epoch 48/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3184 - acc
uracy: 0.8648 - val_loss: 0.3260 - val_accuracy: 0.8612
Epoch 49/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3185 - acc
uracy: 0.8651 - val_loss: 0.3252 - val_accuracy: 0.8604
Epoch 50/300
```

```
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3173 - acc
uracy: 0.8652 - val_loss: 0.3270 - val_accuracy: 0.8595
Epoch 51/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3176 - acc
uracy: 0.8648 - val_loss: 0.3256 - val_accuracy: 0.8625
Epoch 52/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3166 - acc
uracy: 0.8655 - val_loss: 0.3266 - val_accuracy: 0.8612
Epoch 53/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3169 - acc
uracy: 0.8651 - val_loss: 0.3264 - val_accuracy: 0.8598
Epoch 54/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3171 - acc
uracy: 0.8656 - val_loss: 0.3265 - val_accuracy: 0.8603
Epoch 55/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3170 - acc
uracy: 0.8647 - val_loss: 0.3254 - val_accuracy: 0.8608
Epoch 56/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3156 - acc
uracy: 0.8652 - val_loss: 0.3257 - val_accuracy: 0.8620
Epoch 57/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3162 - acc
uracy: 0.8660 - val_loss: 0.3264 - val_accuracy: 0.8596
Epoch 58/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3157 - acc
uracy: 0.8665 - val_loss: 0.3267 - val_accuracy: 0.8598
Epoch 59/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3157 - acc
uracy: 0.8659 - val_loss: 0.3274 - val_accuracy: 0.8613
Epoch 60/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3147 - acc
uracy: 0.8662 - val_loss: 0.3276 - val_accuracy: 0.8608
Epoch 61/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3148 - acc
uracy: 0.8660 - val_loss: 0.3277 - val_accuracy: 0.8608
Epoch 62/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3154 - acc
uracy: 0.8662 - val_loss: 0.3257 - val_accuracy: 0.8614
Epoch 63/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3148 - acc
uracy: 0.8661 - val_loss: 0.3259 - val_accuracy: 0.8619
Epoch 64/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3139 - acc
uracy: 0.8664 - val_loss: 0.3267 - val_accuracy: 0.8622
Epoch 65/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3142 - acc
uracy: 0.8658 - val_loss: 0.3266 - val_accuracy: 0.8612
Epoch 66/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3142 - acc
uracy: 0.8660 - val_loss: 0.3268 - val_accuracy: 0.8598
Epoch 67/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3139 - acc
uracy: 0.8666 - val_loss: 0.3273 - val_accuracy: 0.8599
Epoch 68/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3136 - acc
uracy: 0.8667 - val_loss: 0.3262 - val_accuracy: 0.8612
Epoch 69/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3137 - acc
uracy: 0.8667 - val_loss: 0.3265 - val_accuracy: 0.8608
Epoch 70/300
1077/1077 [==============================] - 2s 2ms/step - loss: 0.3132 - acc
uracy: 0.8669 - val_loss: 0.3256 - val_accuracy: 0.8609
```

Out[36]: <keras.callbacks.History at 0x2154b6ccf50>

In [37]:
```
ann.history.history
```

Out[37]:

```
{'loss': [0.36581382155418396,
  0.35145893692970276,
  0.3484371304512024,
  0.34516191482543945,
  0.3438326120376587,
  0.3414609432220459,
  0.3399929702281952,
  0.33912554383277893,
  0.3377738893032074,
  0.33634814620018005,
  0.33550792932510376,
  0.33499667048454285,
  0.3338972330093384,
  0.33372700214385986,
  0.3319515883922577,
  0.33218905329704285,
  0.3314156234264374,
  0.33066272735595703,
  0.3297240138053894,
  0.32915276288986206,
  0.3289799988269806,
  0.32835566997528076,
  0.32770320773124695,
  0.3281016945838928,
  0.32644808292388916,
  0.326649934053421,
  0.3261687159538269,
  0.3252173066139221,
  0.3256732225418091,
  0.3242429494857788,
  0.32441091537475586,
  0.3236311376094818,
  0.32341593503952026,
  0.3220312297344208,
  0.3234211206436157,
  0.32294338941574097,
  0.32248446345329285,
  0.32091379165649414,
  0.32162579894065857,
  0.320749968290329,
  0.32110440731048584,
  0.3208119571208954,
  0.3201605975627899,
  0.32038143277168274,
  0.31924358010292053,
  0.31936049461364746,
  0.31888651847839355,
  0.3183622658252716,
  0.3184589147567749,
  0.3172789216041565,
  0.3175828158855438,
  0.3166368007659912,
  0.3168753683567047,
  0.31708773970603943,
  0.3170190453529358,
  0.31561851501464844,
  0.31615686416625977,
  0.3157174289226532,
  0.3156595230102539,
  0.31471824645996094,
  0.3148355185985565,
  0.31536585092544556,
  0.31480005383491516,
  0.3139103949069977,
  0.31423333287239075,
  0.3142426609992981,
  0.31392738223075867,
  0.3135605454444885,
  0.3137194812297821,
  0.31320980191230774],
 'accuracy': [0.8437012434005737,
  0.8501653075218201,
  0.8512704968452454,
  0.8521807193756104,
```

```
 0.8531558513641357,
 0.8543261289596558,
 0.855022668838501,
 0.854957640171051,
 0.8559699654579163,
 0.8563507795333862,
 0.8559328317642212,
 0.8568058609962463,
 0.8575674295425415,
 0.8575395941734314,
 0.8581804037094116,
 0.8577624559402466,
 0.8585797548294067,
 0.8589419722557068,
 0.8586726188659668,
 0.8599821925163269,
 0.8591648936271667,
 0.860065758228302,
 0.8599264621734619,
 0.859592080116272,
 0.8613567352294922,
 0.860381543636322,
 0.8606880307197571,
 0.8607994914054871,
 0.8605579733848572,
 0.8609201908111572,
 0.8615424633026123,
 0.8619697093963623,
 0.8625547885894775,
 0.8614774346351624,
 0.8625269532203674,
 0.8632420897483826,
 0.8628612756729126,
 0.8637714385986328,
 0.8635113835334778,
 0.864644467830658,
 0.8636785745620728,
 0.8628612756729126,
 0.8640221953392029,
 0.8632792234420776,
 0.8634835481643677,
 0.8647280335426331,
 0.864644467830658,
 0.864802360534668,
 0.865053117275238,
 0.865173876285553,
 0.864802360534668,
 0.865489661693573,
 0.865108847618103,
 0.8655732274055481,
 0.8647094964981079,
 0.8651831746101379,
 0.8660097122192383,
 0.8665019869804382,
 0.8658611178398132,
 0.8661769032478333,
 0.8659632802009583,
 0.8661862015724182,
 0.8660933375358582,
 0.8664091229438782,
 0.8657589554786682,
 0.8660376071929932,
 0.8666412830352783,
 0.8667248487472534,
 0.8667434453964233,
 0.8669106364250183],
'val_loss': [0.34558382630348206,
 0.3388546109199524,
 0.3380020260810852,
 0.3347031772136688,
 0.3358044922351837,
 0.3328825831413269,
 0.3304961919784546,
 0.33091986179351807,
```

```
 0.3327895700931549,
 0.3296668529510498,
 0.3307962119579315,
 0.3298111855983734,
 0.32874971628189087,
 0.3282792568206787,
 0.33045822381973267,
 0.3273714780807495,
 0.3282526731491089,
 0.3271668553352356,
 0.327959269285202,
 0.32763099670410156,
 0.32725533843040466,
 0.3265800476074219,
 0.3276275396347046,
 0.32693931460380554,
 0.3263551890850067,
 0.3270237147808075,
 0.32820799946784973,
 0.32668599486351013,
 0.326384574174881,
 0.3261721730232239,
 0.3259594142436981,
 0.3261428773403168,
 0.3260476291179657,
 0.3258073031902313,
 0.32604077458381653,
 0.325471431016922,
 0.32555055618286133,
 0.32584571838378906,
 0.32603469491004944,
 0.32629847526550293,
 0.32610854506492615,
 0.3260366916656494,
 0.3292379677295685,
 0.3262820839881897,
 0.325095534324646,
 0.3252159655094147,
 0.32545870542526245,
 0.32598286867141724,
 0.32516157627105713,
 0.3270433247089386,
 0.325645387172699,
 0.3266477882862091,
 0.3264387845993042,
 0.32647356390953064,
 0.32535088062286377,
 0.32570627331733704,
 0.3264455199241638,
 0.3266542851924896,
 0.3273744285106659,
 0.32758593559265137,
 0.3277381956577301,
 0.32573238015174866,
 0.32594382762908936,
 0.326700359582901,
 0.32662320137023926,
 0.3268437385559082,
 0.3273135721683502,
 0.3262421488761902,
 0.32645609974861145,
 0.32563474774360657],
'val_accuracy': [0.8522921204566956,
 0.8537409901618958,
 0.8565644025802612,
 0.8567872643470764,
 0.8570844531059265,
 0.8572702407836914,
 0.8593134880065918,
 0.8598335981369019,
 0.8581990003585815,
 0.8599449992179871,
 0.859907865524292,
 0.8588305115699768,
```

```
0.8597592711448669,
0.860390841960907,
0.8590162992477417,
0.8596106767654419,
0.8592020273208618,
0.8596106767654419,
0.860279381275177,
0.8590905666351318,
0.8589791059494019,
0.859796404838562,
0.8607622981071472,
0.860056459903717,
0.8607994914054871,
0.859536349773407,
0.8589791059494019,
0.8606880307197571,
0.8607622981071472,
0.8603536486625671,
0.860390841960907,
0.8599821925163269,
0.8602050542831421,
0.8609480857849121,
0.8611338138580322,
0.8613567352294922,
0.8615424633026123,
0.8620254397392273,
0.8592391610145569,
0.8607994914054871,
0.8606137037277222,
0.8611709475517273,
0.8599449992179871,
0.8606137037277222,
0.8606508374214172,
0.8623597621917725,
0.8609852194786072,
0.8612081408500671,
0.860427975654602,
0.8594992160797119,
0.8625454902648926,
0.8612081408500671,
0.859796404838562,
0.8603165149688721,
0.8608366250991821,
0.8619511127471924,
0.8595735430717468,
0.859796404838562,
0.8613195419311523,
0.8608366250991821,
0.8608366250991821,
0.8613567352294922,
0.8619139790534973,
0.8621740341186523,
0.8612081408500671,
0.859796404838562,
0.8598707318305969,
0.8612081408500671,
0.8607622981071472,
0.8609108924865723]}
```

Making Predictions

In [39]:
```
ypred=ann.predict(xtest)
```

```
842/842 [==============================] - 0s 526us/step
```

In [40]:
```
ypred
```

Out[40]:
```
array([[0.01425193],
       [0.03892713],
       [0.11097739],
       ...,
       [0.1671862 ],
       [0.31435546],
       [0.00717696]], dtype=float32)
```

In [41]:
```
ypred=np.where(ypred>0.5,1,0)
ypred
```

Out[41]:
```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [0],
       [0]])
```

Checking Accuracy

In [42]:
```
from sklearn.metrics import classification_report
```

In [43]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.95      0.91     21043
           1       0.75      0.54      0.63      5875

    accuracy                           0.86     26918
   macro avg       0.82      0.74      0.77     26918
weighted avg       0.85      0.86      0.85     26918
```

In [ ]:
```

```