

In [1]:

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 import seaborn as sns
6 from IPython import get_ipython
7 import warnings
8 warnings.filterwarnings("ignore")

```

In [2]:

```
1 data = pd.read_csv('rainfall_aus.csv')
```

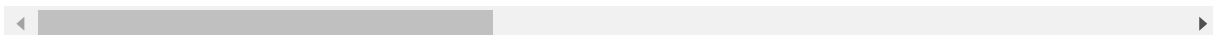
In [3]:

```
1 data.head()
```

Out[3]:

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

5 rows × 23 columns



In [4]:

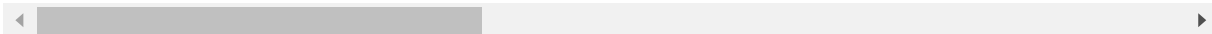


```
1 data.tail()
```

Out[4]:

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

5 rows × 23 columns



In [5]:



```
1 data.shape
```

Out[5]:

```
(145460, 23)
```

In [6]:



```
1 data.columns
```

Out[6]:

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporatio  
n',  
      'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3  
pm',  
      'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',  
      'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',  
      'Temp3pm', 'RainToday', 'RainTomorrow'],  
      dtype='object')
```

In [7]:



```
1 data.duplicated().sum()
```

Out[7]:

0

In [8]:



```
1 data.isnull().sum()
```

Out[8]:

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 [9]:



```
1 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 [10]:



```
1 data.describe()
```

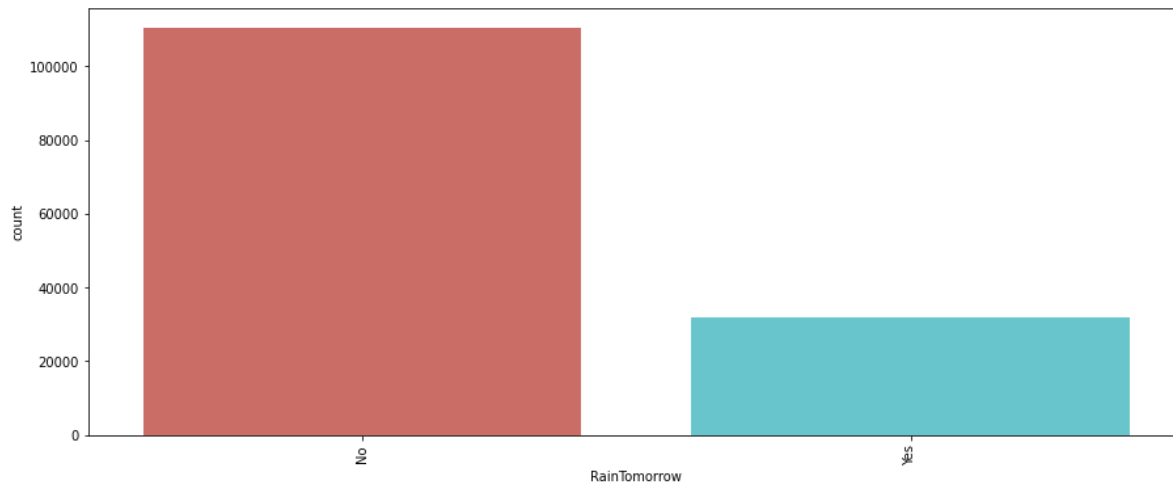
Out[10]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSp
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 [11]:



```
1 plt.figure(figsize=(15,6))
2 sns.countplot(x= data["RainTomorrow"], data = data, palette='hls')
3 plt.xticks(rotation = 90)
4 plt.show()
```



In [13]:



```
1 corrmat = data.corr()
2 corrmat
```

Out[13]:

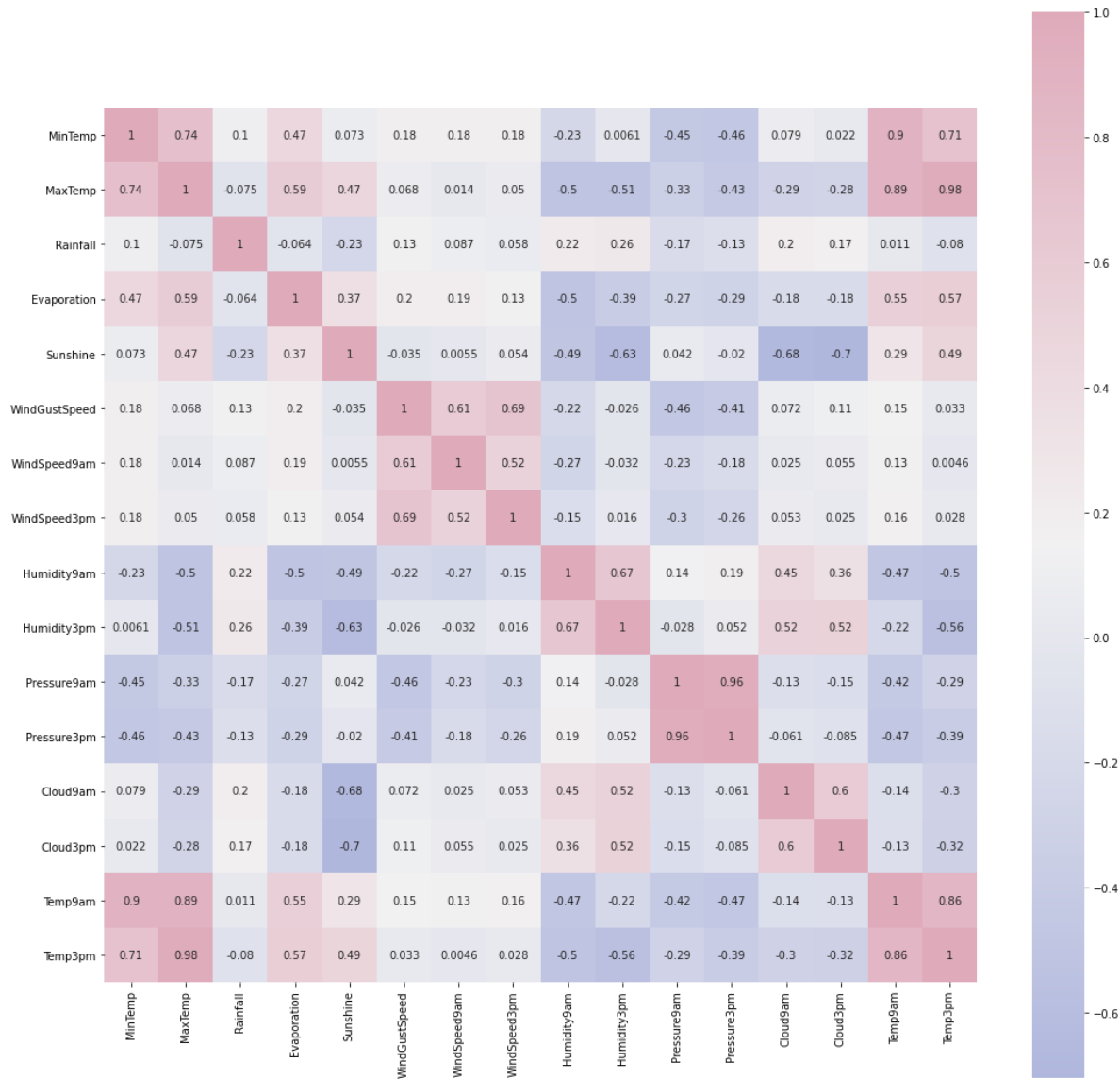
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSp
MinTemp	1.000000	0.736555	0.103938	0.466993	0.072586	0.177415	
MaxTemp	0.736555	1.000000	-0.074992	0.587932	0.470156	0.067615	
Rainfall	0.103938	-0.074992	1.000000	-0.064351	-0.227549	0.133659	
Evaporation	0.466993	0.587932	-0.064351	1.000000	0.365602	0.203021	
Sunshine	0.072586	0.470156	-0.227549	0.365602	1.000000	-0.034750	
WindGustSpeed	0.177415	0.067615	0.133659	0.203021	-0.034750	1.000000	
WindSpeed9am	0.175064	0.014450	0.087338	0.193084	0.005499	0.605303	
WindSpeed3pm	0.175173	0.050300	0.057887	0.129400	0.053834	0.686307	
Humidity9am	-0.232899	-0.504110	0.224405	-0.504092	-0.490819	-0.215070	-
Humidity3pm	0.006089	-0.508855	0.255755	-0.390243	-0.629130	-0.026327	-
Pressure9am	-0.450970	-0.332061	-0.168154	-0.270362	0.041970	-0.458744	-
Pressure3pm	-0.461292	-0.427167	-0.126534	-0.293581	-0.019719	-0.413749	-
Cloud9am	0.078754	-0.289370	0.198528	-0.183793	-0.675323	0.071736	
Cloud3pm	0.021605	-0.277921	0.172403	-0.182618	-0.703930	0.109168	
Temp9am	0.901821	0.887210	0.011192	0.545115	0.291188	0.150150	
Temp3pm	0.708906	0.984503	-0.079657	0.572893	0.490501	0.032748	

In [14]:

```
1 cmap = sns.diverging_palette(260,-10,s=50, l=75, n=6, as_cmap=True)
2 plt.subplots(figsize=(18,18))
3 sns.heatmap(corrmat,cmap= cmap,annot=True, square=True)
```

Out[14]:

<AxesSubplot:>



In [15]:



```
1 lengths = data["Date"].str.len()
2 lengths.value_counts()
```

Out[15]:

```
10    145460
Name: Date, dtype: int64
```

In [16]:

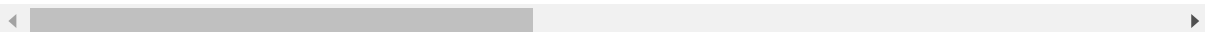


```
1 data['Date'] = pd.to_datetime(data["Date"])
2 data['year'] = data.Date.dt.year
3
4 def encode(data, col, max_val):
5     data[col + '_sin'] = np.sin(2 * np.pi * data[col]/max_val)
6     data[col + '_cos'] = np.cos(2 * np.pi * data[col]/max_val)
7     return data
8
9 data['month'] = data.Date.dt.month
10 data = encode(data, 'month', 12)
11
12 data['day'] = data.Date.dt.day
13 data = encode(data, 'day', 31)
14
15 data.head()
```

Out[16]:

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

5 rows × 30 columns



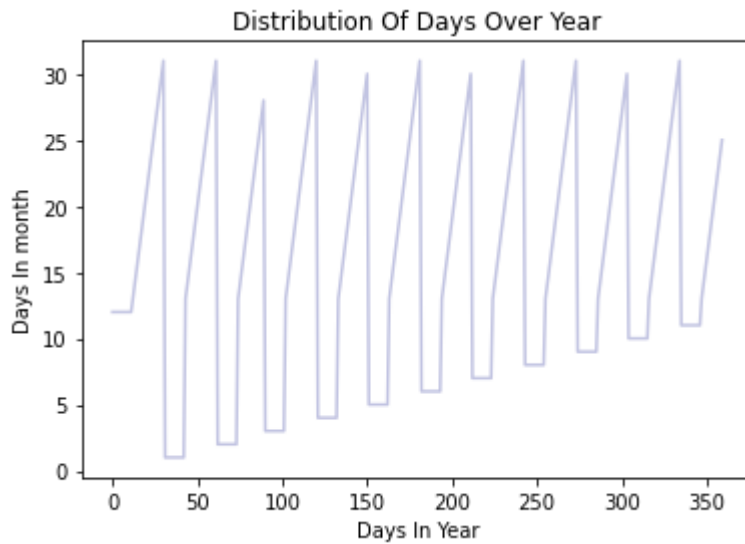
In [17]:



```
1 section = data[:360]
2 tm = section["day"].plot(color="#C2C4E2")
3 tm.set_title("Distribution Of Days Over Year")
4 tm.set_ylabel("Days In month")
5 tm.set_xlabel("Days In Year")
```

Out[17]:

Text(0.5, 0, 'Days In Year')

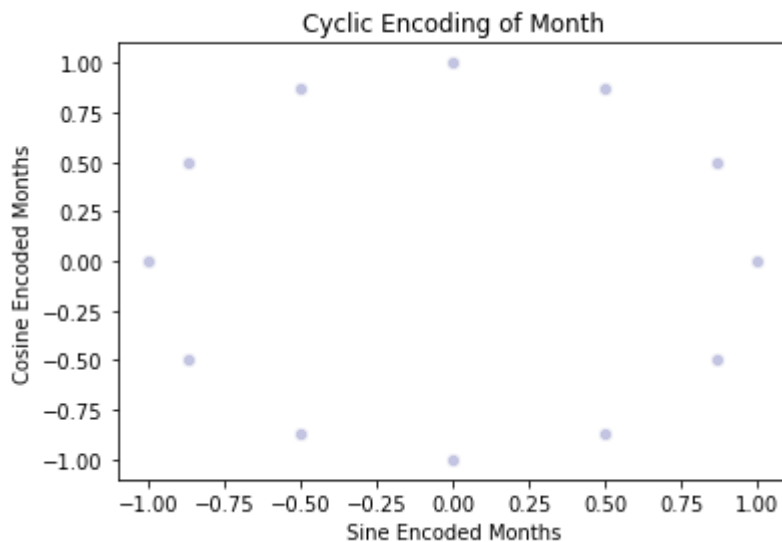


In [18]:

```
1 cyclic_month = sns.scatterplot(x="month_sin",y="month_cos",  
2                               data=data, color="#C2C4E2")  
3 cyclic_month.set_title("Cyclic Encoding of Month")  
4 cyclic_month.set_ylabel("Cosine Encoded Months")  
5 cyclic_month.set_xlabel("Sine Encoded Months")
```

Out[18]:

Text(0.5, 0, 'Sine Encoded Months')

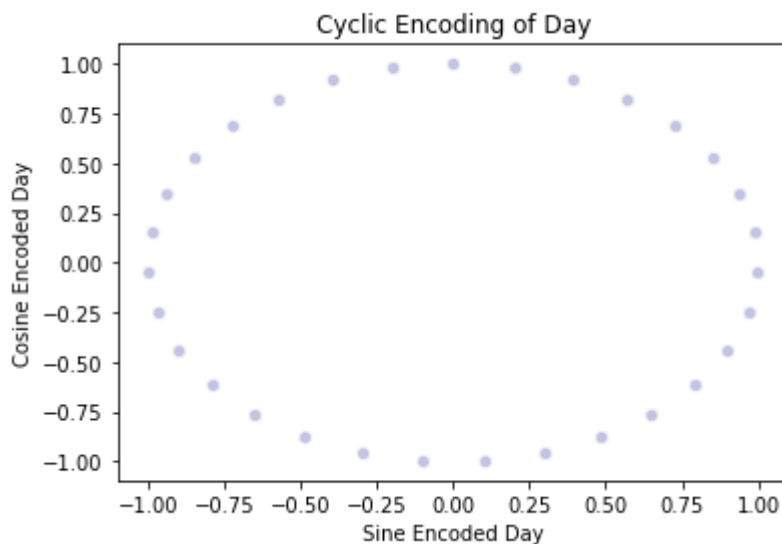


In [19]:

```
1 cyclic_day = sns.scatterplot(x='day_sin',y='day_cos',data=data,  
2                              color="#C2C4E2")  
3 cyclic_day.set_title("Cyclic Encoding of Day")  
4 cyclic_day.set_ylabel("Cosine Encoded Day")  
5 cyclic_day.set_xlabel("Sine Encoded Day")
```

Out[19]:

Text(0.5, 0, 'Sine Encoded Day')



In [20]:



```
1 s = (data.dtypes == "object")
2 object_cols = list(s[s].index)
3
4 print("Categorical variables:")
5 print(object_cols)
```

Categorical variables:

```
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'Rain
Tomorrow']
```

In [21]:



```
1 for i in object_cols:
2     print(i, data[i].isnull().sum())
```

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

In [22]:



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

In [23]:



```
1 t = (data.dtypes == "float64")
2 num_cols = list(t[t].index)
3
4 print("Neumeric variables:")
5 print(num_cols)
```

Neumeric variables:

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpe
ed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressu
re9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'mont
h_sin', 'month_cos', 'day_sin', 'day_cos']
```

In [24]:



```
1 for i in num_cols:  
2     print(i, data[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  
month_sin 0  
month_cos 0  
day_sin 0  
day_cos 0
```

In [25]:



```

1 for i in num_cols:
2     data[i].fillna(data[i].median(), inplace=True)
3
4 data.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 30 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Date                  145460 non-null  datetime64[ns]
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
23  year                  145460 non-null  int64
24  month                 145460 non-null  int64
25  month_sin             145460 non-null  float64
26  month_cos             145460 non-null  float64
27  day                   145460 non-null  int64
28  day_sin               145460 non-null  float64
29  day_cos               145460 non-null  float64
dtypes: datetime64[ns](1), float64(20), int64(3), object(6)
memory usage: 33.3+ MB

```

In [27]:



```
1 import datetime
2 from sklearn.preprocessing import LabelEncoder
3 from sklearn import preprocessing
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.model_selection import train_test_split
6 from keras.layers import Dense, BatchNormalization, Dropout, LSTM
7 from keras.models import Sequential
8 from keras.utils import to_categorical
9 from keras.optimizers import Adam
10 from tensorflow.keras import regularizers
11 from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report
12 from keras import callbacks
```

In [28]:



```

1 label_encoder = LabelEncoder()
2 for i in object_cols:
3     data[i] = label_encoder.fit_transform(data[i])
4
5 data.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 30 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Date                  145460 non-null  datetime64[ns]
1   Location              145460 non-null  int32
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  int32
8   WindGustSpeed         145460 non-null  float64
9   WindDir9am            145460 non-null  int32
10  WindDir3pm            145460 non-null  int32
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  int32
22  RainTomorrow          145460 non-null  int32
23  year                  145460 non-null  int64
24  month                 145460 non-null  int64
25  month_sin             145460 non-null  float64
26  month_cos             145460 non-null  float64
27  day                   145460 non-null  int64
28  day_sin               145460 non-null  float64
29  day_cos               145460 non-null  float64
dtypes: datetime64[ns](1), float64(20), int32(6), int64(3)
memory usage: 30.0 MB

```

In [29]:



```

1 features = data.drop(['RainTomorrow', 'Date', 'day', 'month'], axis=1) # dropping ta
2
3 target = data['RainTomorrow']
4
5 col_names = list(features.columns)
6 s_scaler = preprocessing.StandardScaler()
7 features = s_scaler.fit_transform(features)
8 features = pd.DataFrame(features, columns=col_names)
9
10 features.describe().T

```

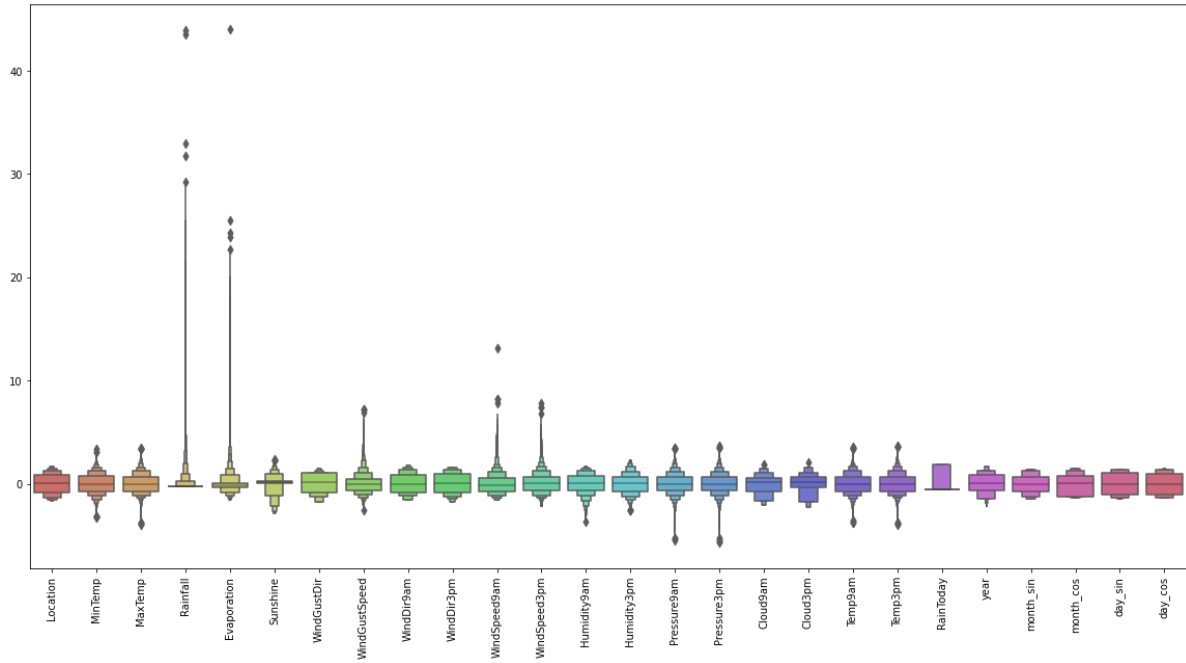
Out[29]:

	count	mean	std	min	25%	50%	75%
Location	145460.0	7.815677e-18	1.000003	-1.672228	-0.899139	0.014511	0.857881
MinTemp	145460.0	-4.501830e-16	1.000003	-3.250525	-0.705659	-0.030170	0.723865
MaxTemp	145460.0	3.001220e-16	1.000003	-3.952405	-0.735852	-0.086898	0.703133
Rainfall	145460.0	7.815677e-18	1.000003	-0.275097	-0.275097	-0.275097	-0.203581
Evaporation	145460.0	-3.282584e-17	1.000003	-1.629472	-0.371139	-0.119472	0.006361
Sunshine	145460.0	-5.424080e-16	1.000003	-2.897217	0.076188	0.148710	0.257494
WindGustDir	145460.0	6.252542e-18	1.000003	-1.724209	-0.872075	0.193094	1.045228
WindGustSpeed	145460.0	1.824961e-16	1.000003	-2.588407	-0.683048	-0.073333	0.460168
WindDir9am	145460.0	7.190423e-17	1.000003	-1.550000	-0.885669	0.000105	0.885879
WindDir3pm	145460.0	8.284618e-17	1.000003	-1.718521	-0.837098	0.044324	0.925747
WindSpeed9am	145460.0	5.627287e-17	1.000003	-1.583291	-0.793380	-0.116314	0.560752
WindSpeed3pm	145460.0	6.565169e-17	1.000003	-2.141841	-0.650449	0.037886	0.611499
Humidity9am	145460.0	2.250915e-16	1.000003	-3.654212	-0.631189	0.058273	0.747734
Humidity3pm	145460.0	-8.440931e-17	1.000003	-2.518329	-0.710918	0.021816	0.656852
Pressure9am	145460.0	-4.314254e-16	1.000003	-5.520544	-0.616005	-0.006653	0.617561
Pressure3pm	145460.0	5.027043e-15	1.000003	-5.724832	-0.622769	-0.007520	0.622735
Cloud9am	145460.0	-1.016038e-16	1.000003	-2.042425	-0.727490	0.149133	0.587445

	count	mean	std	min	25%	50%	75%
Cloud3pm	145460.0	7.346736e-17	1.000003	-2.235619	-0.336969	0.137693	0.612356
Temp9am	145460.0	7.503050e-17	1.000003	-3.750358	-0.726764	-0.044517	0.699753
Temp3pm	145460.0	-6.877796e-17	1.000003	-3.951301	-0.725322	-0.083046	0.661411
RainToday	145460.0	-8.988029e-18	1.000003	-0.529795	-0.529795	-0.529795	-0.529795
year	145460.0	2.080221e-14	1.000003	-2.273637	-0.697391	0.090732	0.878855
month_sin	145460.0	-8.011069e-18	1.000003	-1.424382	-0.715973	-0.007564	0.700845
month_cos	145460.0	9.735403e-17	1.000003	-1.397692	-1.208546	0.014112	0.720014
day_sin	145460.0	-2.165187e-17	1.000003	-1.403031	-1.018734	-0.001896	1.014942
day_cos	145460.0	6.375883e-17	1.000003	-1.402828	-1.066146	-0.056419	0.998236

In [30]:

```
1 plt.figure(figsize=(20,10))
2 sns.boxenplot(data = features, palette = 'hls')
3 plt.xticks(rotation=90)
4 plt.show()
```



In [31]:

```

1 features["RainTomorrow"] = target
2
3 features = features[(features["MinTemp"]<2.3)&(features["MinTemp"]>-2.3)]
4 features = features[(features["MaxTemp"]<2.3)&(features["MaxTemp"]>-2)]
5 features = features[(features["Rainfall"]<4.5)]
6 features = features[(features["Evaporation"]<2.8)]
7 features = features[(features["Sunshine"]<2.1)]
8 features = features[(features["WindGustSpeed"]<4)&(features["WindGustSpeed"]>-4)]
9 features = features[(features["WindSpeed9am"]<4)]
10 features = features[(features["WindSpeed3pm"]<2.5)]
11 features = features[(features["Humidity9am"]>-3)]
12 features = features[(features["Humidity3pm"]>-2.2)]
13 features = features[(features["Pressure9am"]< 2)&(features["Pressure9am"]>-2.7)]
14 features = features[(features["Pressure3pm"]< 2)&(features["Pressure3pm"]>-2.7)]
15 features = features[(features["Cloud9am"]<1.8)]
16 features = features[(features["Cloud3pm"]<2)]
17 features = features[(features["Temp9am"]<2.3)&(features["Temp9am"]>-2)]
18 features = features[(features["Temp3pm"]<2.3)&(features["Temp3pm"]>-2)]
19
20 features.shape

```

Out[31]:

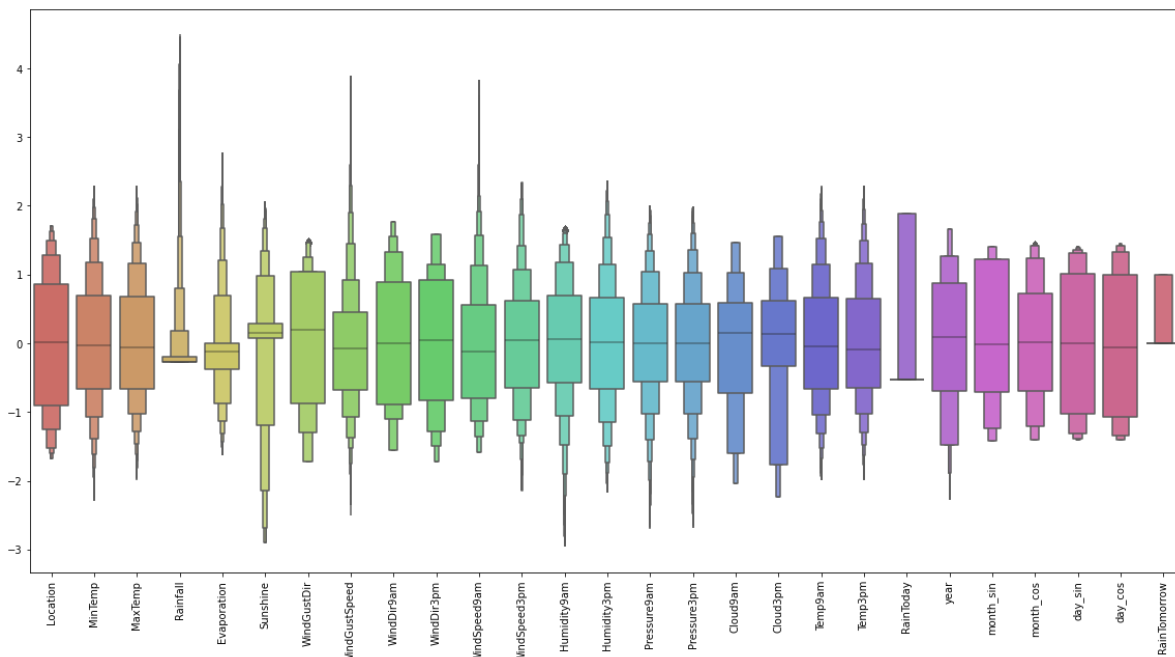
(127536, 27)

In [32]:

```

1 plt.figure(figsize=(20,10))
2 sns.boxenplot(data = features,palette = 'hls')
3 plt.xticks(rotation=90)
4 plt.show()

```



In [33]:



```
1 X = features.drop(["RainTomorrow"], axis=1)
2 y = features["RainTomorrow"]
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y,
5                                                    test_size = 0.2,
6                                                    random_state = 42)
7
8 X.shape
```

Out[33]:

(127536, 26)

In [34]:



```

1 early_stopping = callbacks.EarlyStopping(
2     min_delta=0.001,
3     patience=20,
4     restore_best_weights=True,
5 )
6
7 model = Sequential()
8
9 model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu', in
10 model.add(Dense(units = 32, kernel_initializer = 'uniform', activation = 'relu'))
11 model.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu'))
12 model.add(Dropout(0.25))
13 model.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
14 model.add(Dropout(0.5))
15 model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
16
17 opt = Adam(learning_rate=0.00009)
18 model.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])
19
20 history = model.fit(X_train, y_train, batch_size = 32,
21                     epochs = 20, callbacks=[early_stopping],
22                     validation_split=0.2)

```

Epoch 1/20

2551/2551 [=====] - 5s 2ms/step - loss: 0.4700 - accuracy: 0.7841 - val_loss: 0.3908 - val_accuracy: 0.7860

Epoch 2/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.4101 - accuracy: 0.8106 - val_loss: 0.3823 - val_accuracy: 0.8413

Epoch 3/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.4037 - accuracy: 0.8120 - val_loss: 0.3765 - val_accuracy: 0.8424

Epoch 4/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.4009 - accuracy: 0.8131 - val_loss: 0.3721 - val_accuracy: 0.8437

Epoch 5/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3981 - accuracy: 0.8131 - val_loss: 0.3697 - val_accuracy: 0.8450

Epoch 6/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3964 - accuracy: 0.8148 - val_loss: 0.3685 - val_accuracy: 0.8450

Epoch 7/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3942 - accuracy: 0.8163 - val_loss: 0.3668 - val_accuracy: 0.8451

Epoch 8/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3950 - accuracy: 0.8140 - val_loss: 0.3666 - val_accuracy: 0.8448

Epoch 9/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3927 - accuracy: 0.8157 - val_loss: 0.3652 - val_accuracy: 0.8456

Epoch 10/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3927 - accuracy: 0.8145 - val_loss: 0.3648 - val_accuracy: 0.8463

Epoch 11/20

2551/2551 [=====] - 4s 2ms/step - loss: 0.3912 - accuracy: 0.8177 - val_loss: 0.3639 - val_accuracy: 0.8464

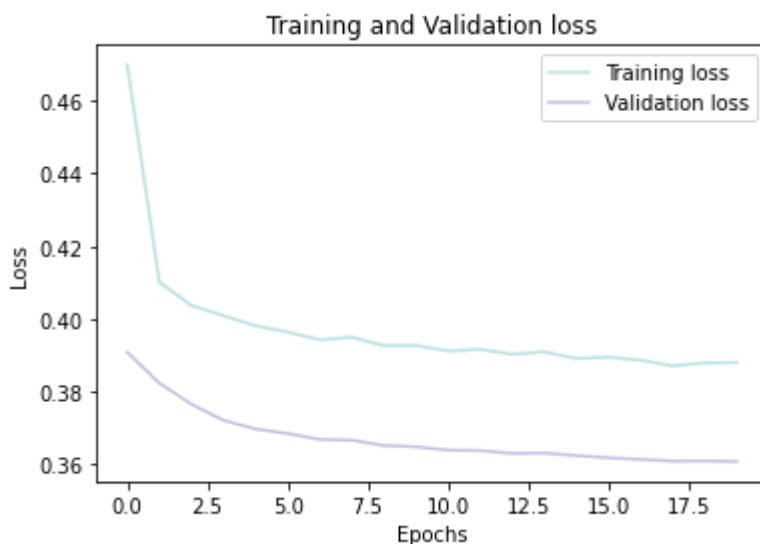
Epoch 12/20

```
2551/2551 [=====] - 4s 2ms/step - loss: 0.3916 - accuracy: 0.8163 - val_loss: 0.3638 - val_accuracy: 0.8455
Epoch 13/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3903 - accuracy: 0.8170 - val_loss: 0.3630 - val_accuracy: 0.8459
Epoch 14/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3910 - accuracy: 0.8156 - val_loss: 0.3631 - val_accuracy: 0.8452
Epoch 15/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3891 - accuracy: 0.8163 - val_loss: 0.3624 - val_accuracy: 0.8449
Epoch 16/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3895 - accuracy: 0.8164 - val_loss: 0.3618 - val_accuracy: 0.8455
Epoch 17/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3886 - accuracy: 0.8169 - val_loss: 0.3613 - val_accuracy: 0.8458
Epoch 18/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3871 - accuracy: 0.8182 - val_loss: 0.3609 - val_accuracy: 0.8450
Epoch 19/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3879 - accuracy: 0.8172 - val_loss: 0.3609 - val_accuracy: 0.8450
Epoch 20/20
2551/2551 [=====] - 4s 2ms/step - loss: 0.3880 - accuracy: 0.8170 - val_loss: 0.3608 - val_accuracy: 0.8444
```

In [35]:



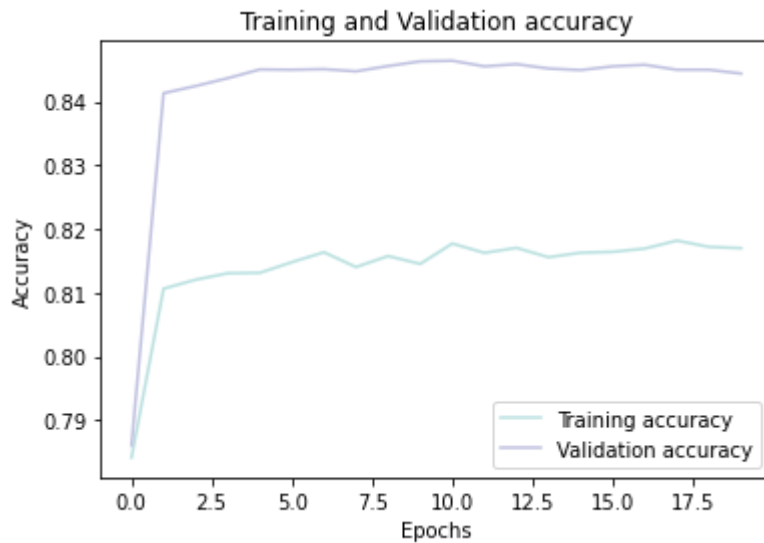
```
1 history_df = pd.DataFrame(history.history)
2
3 plt.plot(history_df.loc[:, ['loss']], "#BDE2E2", label='Training loss')
4 plt.plot(history_df.loc[:, ['val_loss']], "#C2C4E2", label='Validation loss')
5 plt.title('Training and Validation loss')
6 plt.xlabel('Epochs')
7 plt.ylabel('Loss')
8 plt.legend(loc="best")
9
10 plt.show()
```



In [36]:



```
1 history_df = pd.DataFrame(history.history)
2
3 plt.plot(history_df.loc[:, ['accuracy']], "#BDE2E2", label='Training accuracy')
4 plt.plot(history_df.loc[:, ['val_accuracy']], "#C2C4E2", label='Validation accuracy')
5
6 plt.title('Training and Validation accuracy')
7 plt.xlabel('Epochs')
8 plt.ylabel('Accuracy')
9 plt.legend()
10 plt.show()
```



In [37]:



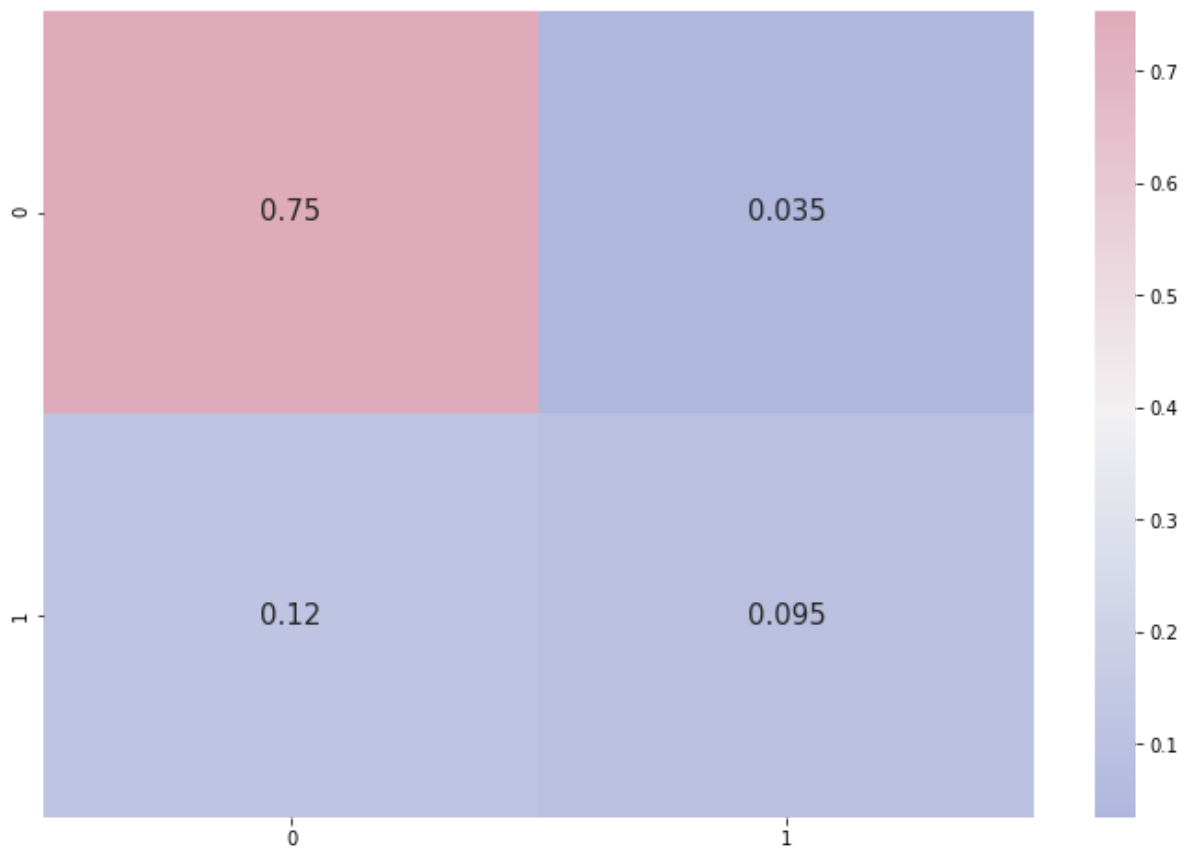
```
1 y_pred = model.predict(X_test)
2 y_pred = (y_pred > 0.5)
```

In [38]:

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

Out[38]:

<AxesSubplot:>



In [39]:



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

	precision	recall	f1-score	support
0	0.87	0.96	0.91	20110
1	0.73	0.45	0.56	5398
accuracy			0.85	25508
macro avg	0.80	0.70	0.73	25508
weighted avg	0.84	0.85	0.83	25508