In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
url='https://www.kaggle.com/datasets/muthuj7/weather-dataset?select=weatherHistory.
data=pd.read_csv('weatherHistory.csv')
data
```

Out[2]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Be (deg
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	
...	...	...	...	...	...	...	...	
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Be (deg

96453 rows × 12 columns

In [3]:
```
data.shape
```

Out[3]: (96453, 12)

In [ ]:
```
```

In [14]:
```
num=data.select_dtypes(exclude='object')
cat=data.select_dtypes(include='object').drop('Formatted Date',axis=1)
cat
```

Out[14]:

	Summary	Precip Type	Daily Summary
0	Partly Cloudy	rain	Partly cloudy throughout the day.
1	Partly Cloudy	rain	Partly cloudy throughout the day.
2	Mostly Cloudy	rain	Partly cloudy throughout the day.
3	Partly Cloudy	rain	Partly cloudy throughout the day.
4	Mostly Cloudy	rain	Partly cloudy throughout the day.
...	...	...	...
96448	Partly Cloudy	rain	Partly cloudy starting in the morning.
96449	Partly Cloudy	rain	Partly cloudy starting in the morning.
96450	Partly Cloudy	rain	Partly cloudy starting in the morning.
96451	Partly Cloudy	rain	Partly cloudy starting in the morning.
96452	Partly Cloudy	rain	Partly cloudy starting in the morning.

96453 rows × 3 columns

In [5]:
```
num
```

Out[5]:

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressu (milliba
0	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.
1	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.
2	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.
3	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.
4	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.
...	...	...	...	...	...	...	...	
96448	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.
96449	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.
96450	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.
96451	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.
96452	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.

96453 rows × 8 columns

In [15]:
```
cat
```

Out[15]:

	Summary	Precip Type	Daily Summary
0	Partly Cloudy	rain	Partly cloudy throughout the day.
1	Partly Cloudy	rain	Partly cloudy throughout the day.
2	Mostly Cloudy	rain	Partly cloudy throughout the day.
3	Partly Cloudy	rain	Partly cloudy throughout the day.
4	Mostly Cloudy	rain	Partly cloudy throughout the day.
...	...	...	...
96448	Partly Cloudy	rain	Partly cloudy starting in the morning.
96449	Partly Cloudy	rain	Partly cloudy starting in the morning.
96450	Partly Cloudy	rain	Partly cloudy starting in the morning.
96451	Partly Cloudy	rain	Partly cloudy starting in the morning.
96452	Partly Cloudy	rain	Partly cloudy starting in the morning.

96453 rows × 3 columns

In [24]:
```
x=data['Summary'].unique()
print(len(x))
```

In [26]:
```
data.describe().T
```

Out[26]:

	count	mean	std	min	25%	50%	
Temperature (C)	96453.0	11.932678	9.551546	-21.822222	4.688889	12.0000	18.83
Apparent Temperature (C)	96453.0	10.855029	10.696847	-27.716667	2.311111	12.0000	18.83
Humidity	96453.0	0.734899	0.195473	0.000000	0.600000	0.7800	0.89
Wind Speed (km/h)	96453.0	10.810640	6.913571	0.000000	5.828200	9.9659	14.13
Wind Bearing (degrees)	96453.0	187.509232	107.383428	0.000000	116.000000	180.0000	290.00
Visibility (km)	96453.0	10.347325	4.192123	0.000000	8.339800	10.0464	14.81
Loud Cover	96453.0	0.000000	0.000000	0.000000	0.000000	0.0000	0.00
Pressure (millibars)	96453.0	1003.235956	116.969906	0.000000	1011.900000	1016.4500	1021.09

In [4]:
```
data.Summary.value_counts()
```

Out[4]:
```
Summary
Partly Cloudy                          31733
Mostly Cloudy                          28094
Overcast                               16597
Clear                                  10890
Foggy                                   7148
Breezy and Overcast                      528
Breezy and Mostly Cloudy                 516
Breezy and Partly Cloudy                 386
Dry and Partly Cloudy                     86
Windy and Partly Cloudy                   67
Light Rain                                63
Breezy                                    54
Windy and Overcast                        45
Humid and Mostly Cloudy                   40
Drizzle                                   39
Breezy and Foggy                          35
Windy and Mostly Cloudy                   35
Dry                                       34
Humid and Partly Cloudy                   17
Dry and Mostly Cloudy                     14
Rain                                      10
Windy                                      8
Humid and Overcast                         7
Windy and Foggy                            4
Windy and Dry                              1
Dangerously Windy and Partly Cloudy        1
Breezy and Dry                             1
Name: count, dtype: int64
```

In [5]:
```
data.isnull().sum()
```

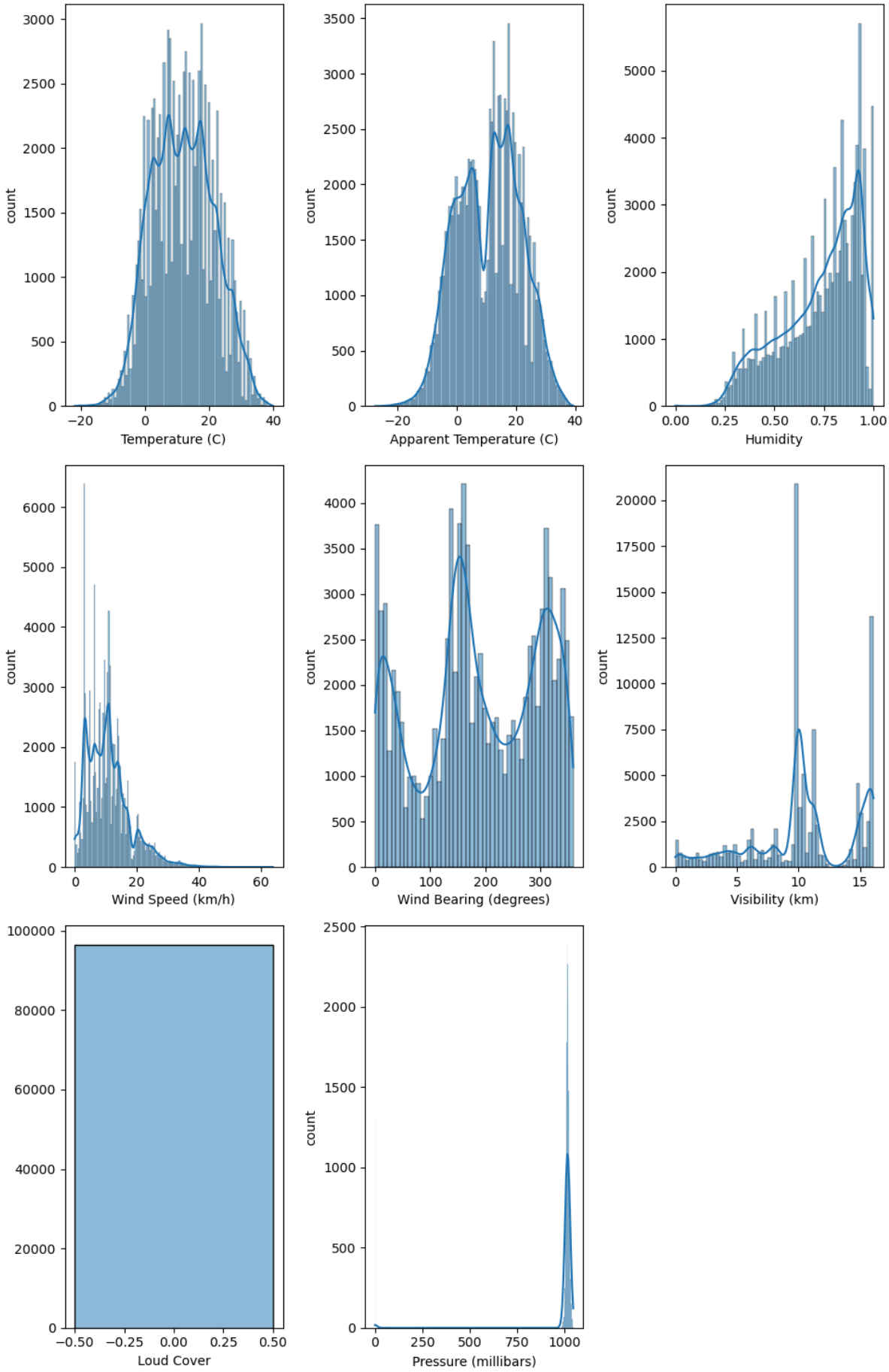
Out[5]:
```
Formatted Date               0
Summary                      0
Precip Type                517
Temperature (C)              0
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
```

In [6]:
```
data.dtypes
```

Out[6]:
```
Formatted Date               object
Summary                      object
Precip Type                  object
Temperature (C)             float64
Apparent Temperature (C)    float64
Humidity                    float64
Wind Speed (km/h)           float64
Wind Bearing (degrees)      float64
Visibility (km)             float64
Loud Cover                  float64
Pressure (millibars)        float64
Daily Summary                object
dtype: object
```

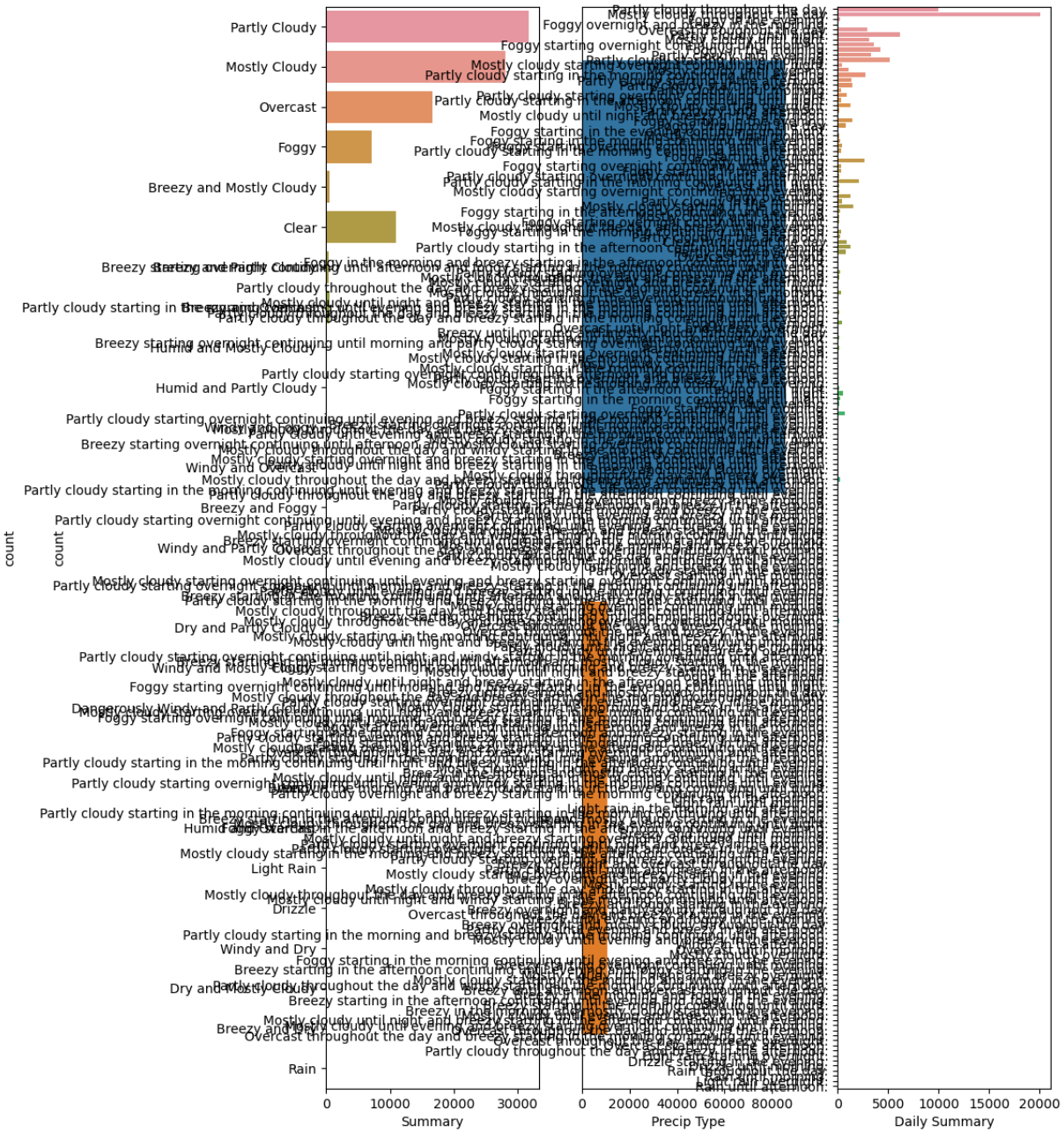
In [13]:
```
#EDA
import matplotlib.pyplot as plt
```

In [20]:
```
plt.figure(figsize=(10,15),facecolor='white')
plotnumber=1
for i in num:
    if plotnumber<=9:
        ax=plt.subplot(3,3,plotnumber)
        sns.histplot(x=num[i],kde=True)
        plt.xlabel(i)
        plt.ylabel('count')
    plotnumber+=1
plt.tight_layout()
```


count
Temperature (C)
Apparent Temperature (C)
Humidity
Wind Speed (km/h)
Wind Bearing (degrees)
Visibility (km)
Loud Cover
Pressure (millibars)

In [ ]:
```
data['Loud Cover']
cat
```

In [59]:
```
plt.figure(figsize=(10,15),facecolor='white')
plotnumber=1
for i in cat:
    if plotnumber<=9:
        ax=plt.subplot(1,3,plotnumber)
        sns.countplot(y=cat[i])
        plt.xlabel(i)
        plt.ylabel('count')
    plotnumber+=1
plt.tight_layout()
```

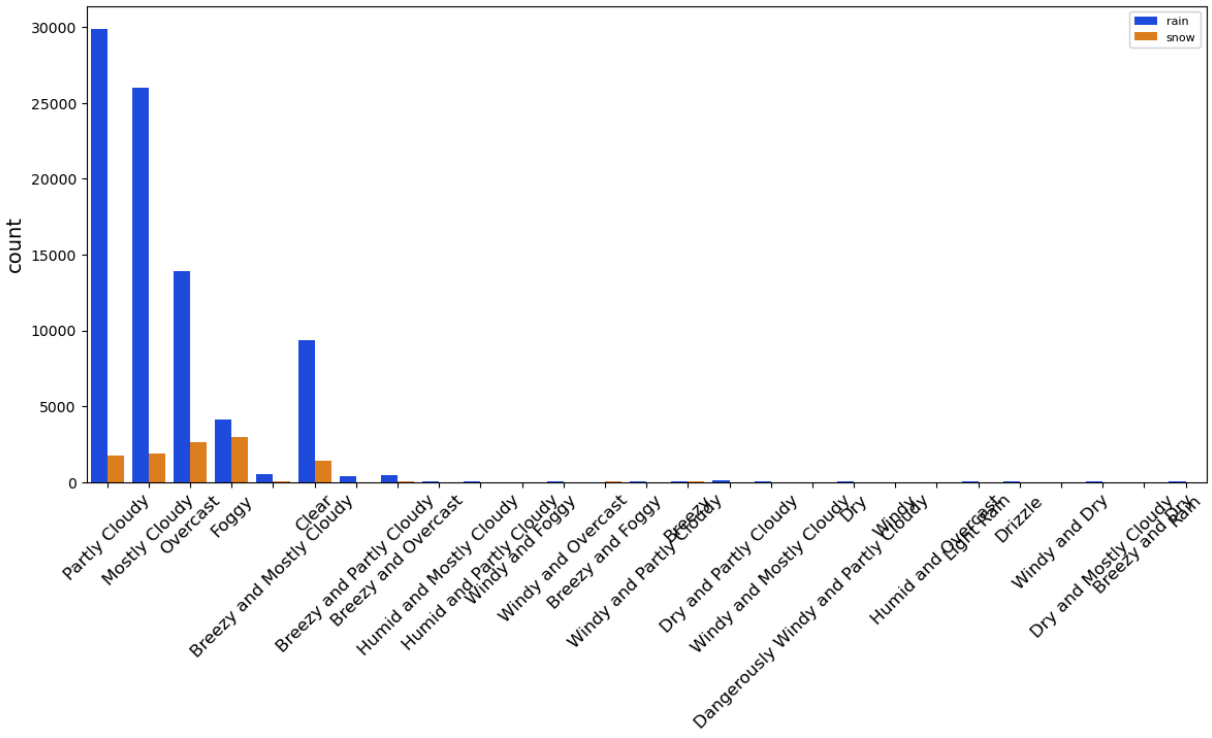


In [18]:
```
cat.colum
```

Out[18]:
```
Index(['Summary', 'Precip Type', 'Daily Summary'], dtype='object')
```

In [66]:
```
plt.figure(figsize=(14,6))
sns.countplot(data=data, x='Summary',hue='Precip Type',palette='bright')
plt.xlabel("" , fontsize=14)
plt.ylabel("count" , fontsize=14)
plt.xticks(fontsize=12, rotation=45)  # Rotate x-axis labels by 45 degrees and incr
plt.legend(fontsize=8, loc='upper right')
plt.show
```

Out[66]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [67]:
```
data.isnull().sum()
```

Out[67]:
```
Formatted Date               0
Summary                      0
Precip Type                517
Temperature (C)              0
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
```

In [72]:
```
data.loc[data['Precip Type'].isnull()==True,'Precip Type'] ='rain'
```

In [73]:
```
data.loc[data['Precip Type'].isnull()==True,'Precip Type']
```

Out[73]:
```
Series([], Name: Precip Type, dtype: object)
```

In [74]:
```
data.columns
```

Out[74]:
```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
       'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
       'Pressure (millibars)', 'Daily Summary'],
      dtype='object')
```

In [75]:
```
data.drop(['Formatted Date','Daily Summary','Loud Cover'], axis=1, inplace=True)
```

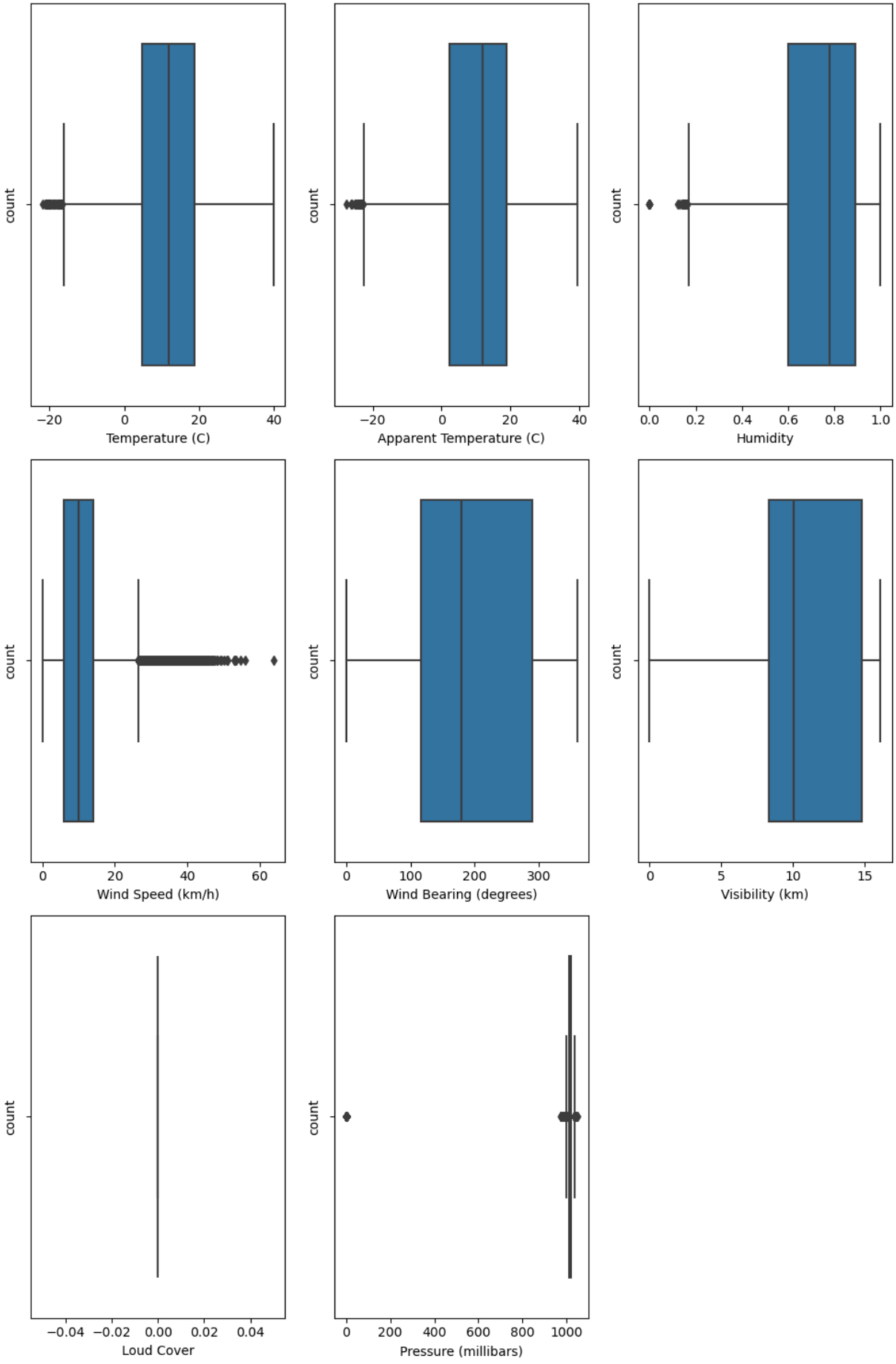
In [79]:
```
data.rename(columns={"target" : "Summary"},inplace=True)
```

In [81]:
```
data.rename(columns={"Precip Type" : "target"},inplace=True)
```

In [82]:
```
data.target.unique()
```

Out[82]:
```
array(['rain', 'snow'], dtype=object)
```

In [84]:
```
# outlier handling
plt.figure(figsize=(10,15),facecolor='white')
plotnumber=1
for i in num:
    if plotnumber<=9:
        ax=plt.subplot(3,3,plotnumber)
        sns.boxplot(x=num[i])
        plt.xlabel(i)
        plt.ylabel('count')
    plotnumber+=1
plt.tight_layout()
```


count
−20
0
20
40
Temperature (C)
count
−20
0
20
40
Apparent Temperature (C)
count
0.0
0.2
0.4
0.6
0.8
1.0
Humidity
count
0
20
40
60
Wind Speed (km/h)
count
0
100
200
300
Wind Bearing (degrees)
count
0
5
10
15
Visibility (km)
count
−0.04
−0.02
0.00
0.02
0.04
Loud Cover
count
0
200
400
600
800
1000
Pressure (millibars)

In [86]:
```
data.isnull().sum()
```

Out[86]:
```
Summary                     0
target                      0
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Pressure (millibars)        0
dtype: int64
```

In [88]:
```
# temperature
q1=data["Temperature (C)"].quantile(0.25)
q3=data["Temperature (C)"].quantile(0.75)
```

In [90]:
```
iqr=q3-q1
iqr
```

Out[90]:
```
14.150000000000002
```

In [91]:
```
low=q1-1.5*iqr
high=q3+1.5*iqr
```

In [103…
```
out=(data["Temperature (C)"]<low) | (data["Temperature (C)"]>high)
per=out.sum()/len(data["Temperature (C)"])
per*100
```

Out[103…
```
0.045618073051123344
```

In [95]:
```
Q1=num.quantile(0.25)
Q3=num.quantile(0.75)
IQR=Q3-Q1
lower=Q1-1.5*IQR
upper=Q3+1.5*IQR
outliers_count=((num<lower)|(num>upper)).sum()
outliers_percentage=(outliers_count/len(num))*100
print(outliers_percentage)
print(outliers_count)
```

```
Temperature (C)             0.045618
Apparent Temperature (C)    0.022809
Humidity                    0.047692
Wind Speed (km/h)           3.139353
Wind Bearing (degrees)      0.000000
Visibility (km)             0.000000
Loud Cover                  0.000000
Pressure (millibars)        4.561807
dtype: float64
Temperature (C)               44
Apparent Temperature (C)      22
Humidity                      46
Wind Speed (km/h)           3028
Wind Bearing (degrees)         0
Visibility (km)                0
Loud Cover                     0
Pressure (millibars)        4400
dtype: int64
```

In [108…
```
q1=data["Wind Speed (km/h)"].quantile(0.25)
q3=data["Wind Speed (km/h)"].quantile(0.75)
iqr=q3-q1
min=q1-1.5*iqr
max=q3+1.5*iqr
```

Out[108…
```
26.597199999999997
```

In [109…
```
data.loc[(data["Wind Speed (km/h)"]<min) | (data["Wind Speed (km/h)"]>high),"Wind S
```

In [110…
```
data.loc[(data["Wind Speed (km/h)"]<min) | (data["Wind Speed (km/h)"]>high),"Wind S
```

Out[110…
```
Series([], Name: Wind Speed (km/h), dtype: float64)
```

In [166…
```
q1=data["Pressure (millibars)"].quantile(0.25)
q3=data["Pressure (millibars)"].quantile(0.75)
iqr=q3-q1
min=q1-1.5*iqr
max=q3+1.5*iqr
```

In [167…
```
#data.drop("Pressure (millibars", axis=1,inplace=True)
data.loc[(data["Pressure (millibars)"]<min) | (data["Pressure (millibars)"]>max),"P
```

Out[167…
```
858      989.83909
874      989.83909
924      989.83909
945      989.83909
1074     989.83909
           ...
93147    989.83909
93149    989.83909
93150    989.83909
93152    989.83909
93153    989.83909
Name: Pressure (millibars), Length: 1288, dtype: float64
```

In [168…
```
data.loc[(data["Pressure (millibars)"]<min) | (data["Pressure (millibars)"]>max),"P
```

In [169…
```
data.loc[(data["Pressure (millibars)"]<min) | (data["Pressure (millibars)"]>max),"P
```

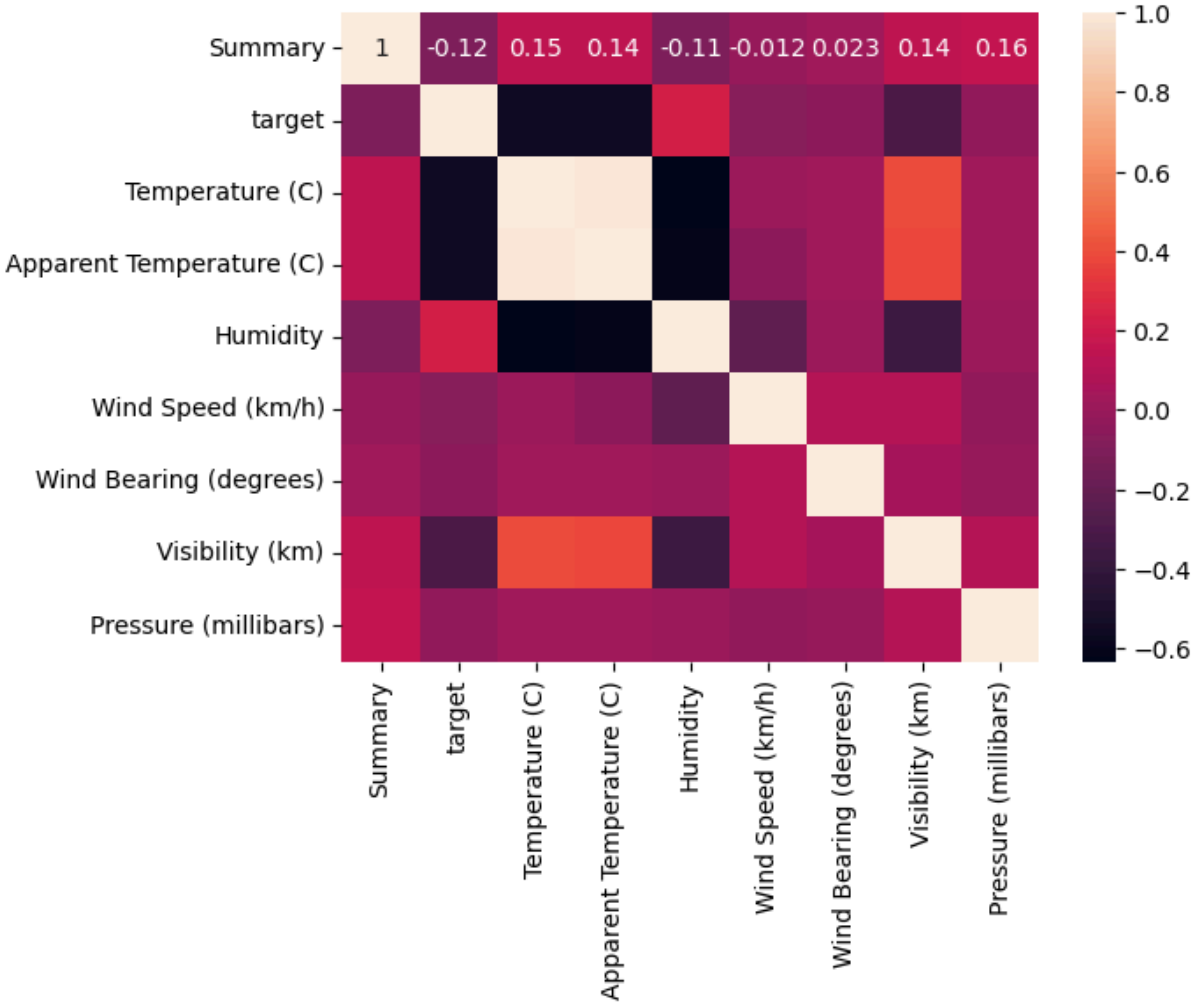
Out[169…
```
858      989.83909
874      989.83909
924      989.83909
945      989.83909
1074     989.83909
           ...
93147    989.83909
93149    989.83909
93150    989.83909
93152    989.83909
93153    989.83909
Name: Pressure (millibars), Length: 1288, dtype: float64
```

In [170…
```
data["Pressure (millibars)"].mean()
```

Out[170…
```
989.8390899076212
```

In [171…
```
corr=data.select_dtypes(exclude="object").corr()
sns.heatmap(corr,annot=True)
```

Out[171…
```
<Axes: >
```



In [172…

```
#label encoding
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
for i in data.select_dtypes(include="object"):
    data[i]=l.fit_transform(data[i])
```

In [173…

```
data
```

Out[173...

	Summary	target	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibili (k
0	19	0	9.472222	7.388889	0.89	14.1197	251.0	15.82
1	19	0	9.355556	7.227778	0.86	14.2646	259.0	15.82
2	17	0	9.377778	9.377778	0.89	3.9284	204.0	14.95
3	19	0	8.288889	5.944444	0.83	14.1036	269.0	15.82
4	17	0	8.755556	6.977778	0.83	11.0446	259.0	15.82
...	...	...	...	...	...	...	...	
96448	19	0	26.016667	26.016667	0.43	10.9963	31.0	16.10
96449	19	0	24.583333	24.583333	0.48	10.0947	20.0	15.55
96450	19	0	22.038889	22.038889	0.56	8.9838	30.0	16.10
96451	19	0	21.522222	21.522222	0.60	10.5294	20.0	16.10
96452	19	0	20.438889	20.438889	0.61	5.8765	39.0	15.52

96453 rows × 9 columns

In [174...
```
X=data.drop("target" ,axis=1)
y=data.target
```

In [153...
```
y
```

Out[153...
```
0        0
1        0
2        0
3        0
4        0
        ..
96448    0
96449    0
96450    0
96451    0
96452    0
Name: target, Length: 96453, dtype: int32
```

In [175...
```
from sklearn.model_selection import train_test_split
x_train,x_test, y_train,y_test=train_test_split(X,y, random_state=13, test_size=0.2
```

In [176...
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(77162, 8)
(19291, 8)
(77162,)
(19291,)
```

In [177…
```
from sklearn.preprocessing import MinMaxScaler
s=MinMaxScaler()
x_train.columns
```

Out[177…
```
Index(['Summary', 'Temperature (C)', 'Apparent Temperature (C)', 'Humidity',
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Pressure (millibars)'],
      dtype='object')
```

In [178…
```
x_train[['Summary', 'Temperature (C)', 'Apparent Temperature (C)', 'Humidity',
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Pressure (millibars)']]=s.fit_transform(x_train[['Summary', 'Temperature (C
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Pressure (millibars)']])
x_train
```

Out[178…

	Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	P (m
24705	0.653846	0.518228	0.513462	0.79	0.576367	0.440111	0.620	
2565	0.653846	0.404203	0.409494	0.85	0.276929	0.807799	0.920	
11716	0.730769	0.423847	0.429542	0.95	0.288585	0.554318	0.264	
15577	0.692308	0.368844	0.369729	0.85	0.285370	0.955432	0.328	
72779	0.653846	0.478757	0.491591	0.79	0.270498	0.554318	0.601	
…	…	…	…	…	…	…	…	…
25324	0.653846	0.474737	0.481733	0.85	0.323151	0.442897	0.413	
65689	0.730769	0.714207	0.735482	0.79	0.274920	0.080780	0.983	
87796	0.730769	0.566377	0.601441	0.73	0.427653	0.779944	1.000	
33634	0.730769	0.723527	0.743932	0.46	0.577974	0.501393	0.620	
47280	0.461538	0.233257	0.299395	0.96	0.076768	0.139276	0.010	

77162 rows × 8 columns

In [179…
```
x_test[['Summary', 'Temperature (C)', 'Apparent Temperature (C)', 'Humidity',
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Pressure (millibars)']]=s.transform(x_test[['Summary', 'Temperature (C)', '
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Pressure (millibars)']])
x_test
```

Out[179…

	Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	P (m
40927	0.730769	0.641480	0.669539	0.70	0.185289	0.523677	0.643	
12715	0.346154	0.955413	0.931820	0.22	0.202572	0.473538	0.620	
27978	0.461538	0.313202	0.323834	0.92	0.200965	0.974930	0.190	
55997	0.461538	0.322339	0.380167	0.88	0.000000	0.000000	0.190	
70049	0.730769	0.768661	0.784856	0.31	0.091238	0.406685	0.966	
...	...	...	...	...	...	...	...	
1012	0.230769	0.704157	0.726369	0.88	0.158762	0.389972	0.983	
87953	0.730769	0.525537	0.564411	0.48	0.219051	0.064067	0.772	
52022	0.730769	0.887529	0.874907	0.30	0.092444	0.426184	0.643	
20066	0.730769	0.343901	0.343302	0.72	0.279341	0.506964	0.383	
31479	0.653846	0.493467	0.514788	0.71	0.209405	0.766017	0.732	

19291 rows × 8 columns

In [180…

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
```

Out[180…

▾ LogisticRegression

LogisticRegression()

In [187…

```
y_pred=model.predict(x_test)
y_pred
```

Out[187…

```
40927    0
12715    0
27978    1
55997    1
70049    0
        ..
1012     0
87953    0
52022    0
20066    1
31479    0
Name: target, Length: 19291, dtype: int32
```

In [199…

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score,roc_auc_score
```

In [185…
```
accuracy_score(y_test,y_pred)
```

Out[185…
```
0.9910839251464414
```

In [186…
```
confusion_matrix(y_test,y_pred)
```

Out[186…
```
array([[17117,    77],
       [   95,  2002]], dtype=int64)
```

In [188…
```
pd.crosstab(y_test,y_pred)
```

Out[188…

col_0	0	1
target		
0	17117	77
1	95	2002

In [189…
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     17194
           1       0.96      0.95      0.96      2097

    accuracy                           0.99     19291
   macro avg       0.98      0.98      0.98     19291
weighted avg       0.99      0.99      0.99     19291
```

In [190…
```
f1_score(y_test,y_pred)
```

Out[190…
```
0.9588122605363985
```

In [191…
```
#SVC classifier
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
```

Out[191…
```
▾ SVC
SVC()
```

In [192…
```
y_pred=model.predict(x_test)
y_pred
```

Out[192…
```
array([0, 0, 1, ..., 0, 1, 0])
```

In [193…
```
accuracy_score(y_test,y_pred)
```

Out[193…
```
0.9919133274584003
```

In [194…
```
confusion_matrix(y_test,y_pred)
```

Out[194…
```
array([[17096,    98],
       [   58,  2039]], dtype=int64)
```

In [196…
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     17194
           1       0.95      0.97      0.96      2097

    accuracy                           0.99     19291
   macro avg       0.98      0.98      0.98     19291
weighted avg       0.99      0.99      0.99     19291
```

In [197…
```
f1_score(y_test,y_pred)
```

Out[197…
```
0.9631554085970713
```

In [200…
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
roc_auc = roc_auc_score(y_test, y_pred)
```

In [201…
```
conf_matrix = confusion_matrix(y_test, y_pred)

# Classification Report
class_report = classification_report(y_test, y_pred)
```

In [202…
```
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC AUC Score:", roc_auc)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
Accuracy: 0.9919133274584003
Precision: 0.9920014048930544
Recall: 0.9919133274584003
F1 Score: 0.9919468175234536
ROC AUC Score: 0.9833208887397867
Confusion Matrix:
 [[17096    98]
 [   58  2039]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.99      1.00     17194
           1       0.95      0.97      0.96      2097

    accuracy                           0.99     19291
   macro avg       0.98      0.98      0.98     19291
weighted avg       0.99      0.99      0.99     19291
```

In [ ]: