Naive Bayes Algorithm

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

In [ ]:
```
# Import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
# load custom dataset
mausam = pd.read_csv('../../datasets/weather.csv')
mausam.head()
```

Out[ ]:

	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
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

In [ ]:
```
# Check missing values
mausam.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
# Drop rows with missing values
mausam.dropna(inplace=True)
```

In [ ]:
```
mausam.isnull().sum()
```

Out[ ]:
```
Formatted Date              0
Summary                     0
Precip Type                 0
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 [ ]:
```
# Number of rown and columns
mausam.shape
```

Out[ ]:
```
(95936, 12)
```

In [ ]:
```
# Data information
mausam.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 95936 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Formatted Date            95936 non-null  object 
 1   Summary                   95936 non-null  object 
 2   Precip Type               95936 non-null  object 
 3   Temperature (C)           95936 non-null  float64
 4   Apparent Temperature (C)  95936 non-null  float64
 5   Humidity                  95936 non-null  float64
 6   Wind Speed (km/h)         95936 non-null  float64
 7   Wind Bearing (degrees)    95936 non-null  float64
 8   Visibility (km)           95936 non-null  float64
 9   Loud Cover                95936 non-null  float64
 10  Pressure (millibars)      95936 non-null  float64
 11  Daily Summary             95936 non-null  object 
dtypes: float64(8), object(4)
memory usage: 9.5+ MB
```

In [ ]:
```
# Split data into X (all numeric data type) and y ('Precip Type')
X = mausam[['Temperature (C)', 'Apparent Temperature (C)', 'Humidity',
            'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
            'Loud Cover', 'Pressure (millibars)']]
y = mausam['Precip Type']
```

In [ ]:
```
X.head()
```

Out[ ]:

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
0	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13
1	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63
2	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94
3	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41
4	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51

In [ ]:
```
y
```

Out[ ]:
```
0        rain
1        rain
2        rain
3        rain
4        rain
         ... 
96448    rain
96449    rain
96450    rain
96451    rain
96452    rain
Name: Precip Type, Length: 95936, dtype: object
```

In [ ]:
```
# Lables frequence count
mausam['Precip Type'].value_counts()
```

Out[ ]:
```
rain    85224
snow    10712
Name: Precip Type, dtype: int64
```

In [ ]:
```
# Split data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[ ]:
```
((76748, 8), (19188, 8), (76748,), (19188,))
```

In [ ]:
```
# Train model
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB().fit(X_train, y_train)

# Predictions
y_pred = nb_model.predict(X_test)
y_pred
```

Out[ ]:
```
array(['snow', 'rain', 'rain', ..., 'rain', 'rain', 'rain'], dtype='<U4')
```

In [ ]:
```
# Check accuracy
from sklearn.metrics import accuracy_score

nb_acc = accuracy_score(y_test, y_pred)
print(f'Gaussian Naive Bayes model accuracy: {nb_acc*100:.2f}')
```

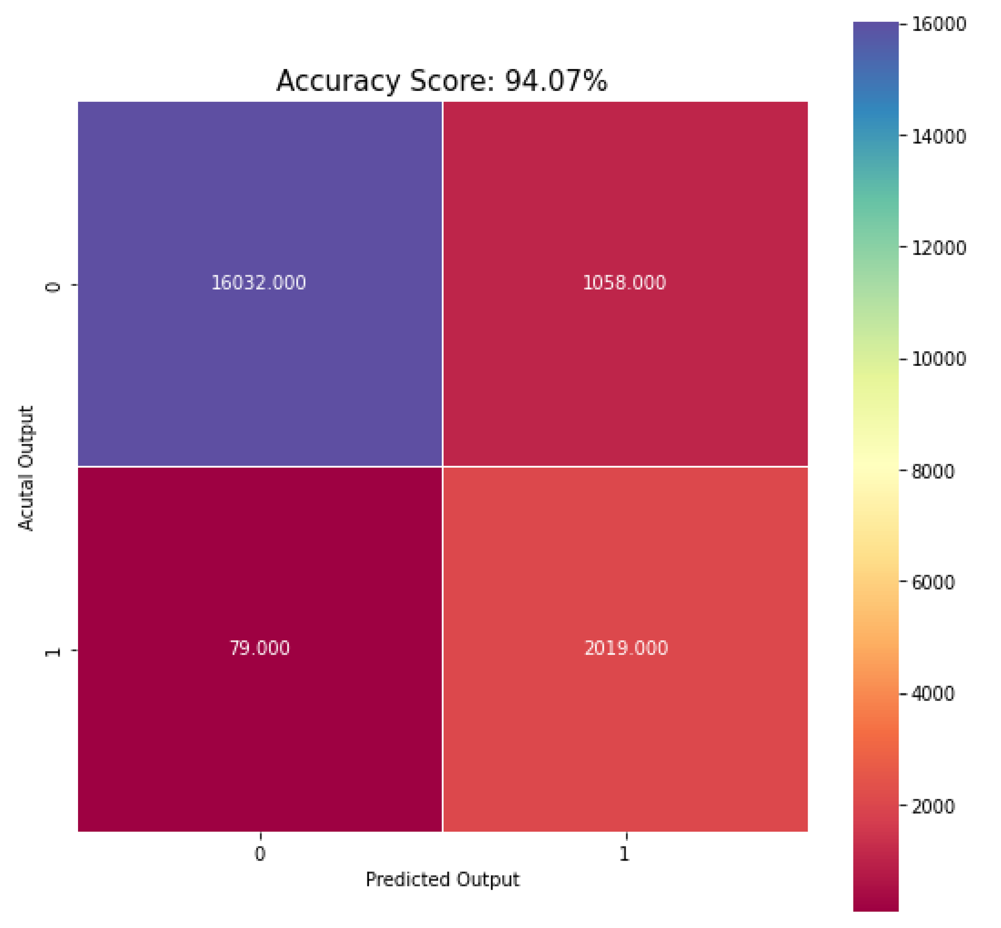
```
Gaussian Naive Bayes model accuracy: 94.07
```

In [ ]:
```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[ ]:
```
array([[16032,  1058],
       [   79,  2019]], dtype=int64)
```

In [ ]:
```
# Plot confusion matrix
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5,
            square=True, cmap='Spectral')

plt.xlabel('Predicted Output')
plt.ylabel('Acutal Output')
plt.title(f'Accuracy Score: {nb_acc*100:.2f}%', size=15);
```



Types of Naive Bayes

There are five types of Naive Bayes algorithms:

- **GaussianNB**: It is used in classification, specifically used when the features have continuous values. It assumes that features follow a gaussian distribution i.e, normal distribution.
- **MultinomialNB**: The multinomial Naive Bayes classifier is suitable for classification with discrete features. The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts may also work.
- **BernoulliNB**: This classifier is for multivariate models. Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.
- **ComplemntNB**: The ComplementNB classifier was designed to correct the "severe assumptions" made by the standard Multinomial Naive Bayes classifier. It is particularly suited for imbalanced datasets.
- **CategoricalNB**: It is suitable for classification with discrete features which assumes categorically distribution for each feature. The features should be encoded using label encoding techniques such that each category would be mapped to a unique number.