```python
In [1]:  import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn import metrics
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report
         from sklearn.metrics import f1_score , recall_score , precision_score
         from sklearn.datasets import make_classification
         from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score
         from sklearn.metrics import accuracy_score
```

```python
In [2]:  Data = pd.read_csv("P1_Data.csv") # Upload the data
```

```python
In [3]:  Data
```

Out[3]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F13	F14	F15	F16	F17	F18	F19	F20	F21	Class
0	1.6430	0	-4894.24	-13.0281	-4.793400	0	5.12700	-17.1100	-63.340	3.61690	...	5.783440	-11315.46	22912.53	-0.4	103811.34	5.4380	1747.920	-4879.68	-41.58	False
1	0.5310	0	-5085.44	-16.2210	-3.991776	0	4.62560	-4.5800	-10.314	3.64880	...	8.180000	-12852.96	25696.44	-0.4	103884.02	5.0960	1496.080	-4186.38	-45.96	True
2	0.2640	0	-7021.44	-11.7591	-6.161700	0	4.36280	-14.7118	-6.806	3.62830	...	5.760312	-11012.16	20232.84	-1.4	103987.08	2.3652	1523.412	-4067.28	NaN	False
3	0.3196	1	-4648.76	-11.8110	-4.217700	0	8.93800	-7.5360	-4.670	3.01503	...	6.437100	-10297.86	23592.84	-1.4	103842.08	4.4080	1506.810	1352.52	NaN	True
4	4.0800	0	-4877.20	-11.2635	-8.061000	1	6.28000	-14.5805	-45.920	3.60030	...	6.393200	-11527.38	24778.74	-1.4	103842.48	3.1334	1581.790	-5095.88	-45.93	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.6828	1	-5766.04	-11.8536	-4.866600	1	7.93800	-11.1720	-8.014	3.27040	...	7.489000	-11922.06	23569.44	-1.4	103930.52	2.8358	1551.190	-5559.68	NaN	False
996	1.2240	1	-4424.64	-16.6770	-8.313000	1	4.91960	-15.1500	-12.676	3.81610	...	7.069000	-11328.69	24196.14	-0.4	103922.40	4.0364	1519.940	-4089.48	NaN	False
997	0.9912	1	-5566.64	-10.9698	-8.364000	0	3.99842	-15.4260	-17.442	3.73230	...	5.875600	-12229.26	18449.64	-1.4	103872.12	3.8368	1509.360	-3772.28	NaN	True
998	0.6697	1	-4630.96	-11.4516	-4.970700	0	8.61600	-19.5220	-1.698	3.22630	...	5.761497	-10846.56	23102.40	-1.4	103837.70	4.4460	1549.200	-3947.68	-40.11	False
999	0.8394	1	-5488.44	-11.9115	-5.107200	0	7.04600	-12.8750	-13.292	3.30530	...	6.069000	-11860.29	23621.04	-0.4	103885.30	3.4320	1522.223	-4645.08	NaN	True

1000 rows × 22 columns

```python
In [4]:  del Data['F21'] # Delete the column F21
```

```python
In [5]:  X = Data.drop('Class', axis = 1) # Excluding target variable
```

```python
In [6]:  y = Data['Class'] # target variable
```

```python
In [8]:  from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report
         from sklearn.metrics import f1_score , recall_score , precision_score
         from sklearn.metrics import plot_confusion_matrix
```

```python
In [9]:  test = pd.read_csv("P1_Test.csv") # Upload the test data
         new_test = test.drop('Class', axis = 1) # Excluding target variable
```

```python
In [10]:  # define new input

          new_input = []

          for i in range(1000):
              new_input.append([ test["F1"][i],test["F2"][i],test["F3"][i],test["F4"][i],test["F5"][i],test["F6"][i],test["F7"][i],test["F8"][i],test["F9"][i...
```

```python
In [11]:  # Apply Gradient Boosting Classifier to predict class values
          from sklearn.ensemble import GradientBoostingClassifier
          GRA = GradientBoostingClassifier(n_estimators = 200, random_state = 1, learning_rate = 0.01)
          GRA.fit(X,y)
          y_predict = GRA.predict(new_input)
          rs = []
          for i in range(len(new_input)):
              rs.append(y_predict[i])

          print(rs)
```

[False, False, True, False, True, True, True, False, True, False, False, False, False, True, True, True, False, True, False, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, True, False, False, False, True, True, False, False, False, True, True, False, False, False, False, False, True, True, True, True, True, True, True, True, True, True, False, True, False, False, True, True, True, False, True, False, False, False, True, True, True, True, True, True, True, True, True, False, True, True, True, False, True, True, False, False, False, True, True, False, False, True, True, True, False, True, True, False, True, False, True, False, False, False, True, True, True, True, True, True, False, True, True, True, True, True, True, False, False, True, False, True, True, True, False, True, False, False, True, True, False, False, True, True, True, True, True, True, True, False, False, False, False, True, False, True, True, True, True, False, False, False, True, True, False, False, True, True, True, False, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, True, True, True, True, False, True, False, False, False, True, False, True, True, True, True, False, True, True, False, False, False, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, False, False, True, True, True, False, True, False, True, True, False, True, True, False, True, True, True, True, True, False, True, True, True, True, True, True, False, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, False, True, True, True, False, True, False, False, True, True, True, True, True, True, False, True, True, False, False, False, True, True, True, True, False, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, True, True, False, True, True, True, True, True, False, True, True, True, True, False, True, True, False, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, False, True, False, False, True, False, True, False, False, True, True, True, True, True, True, True, True, True, True, False, False, True, True, True, True, True, False, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, False, False, True, True, True, True, True, True, True, True, False, True, True, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, True, True, False, True, True, True, False, True, True, True, False, True, True, True, True, True, False, False, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, False, True, True, False, True, True, True, True, True, False, True, True, True, False, False, False, True, True, True, False, True, True, True, True, True, True, True, False, True, False, False, False, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, False, False, False, True, True, True, True, True, False, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, False, False, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, False, True, True, False, True, True, True, True, False, True, True, True, True, True, True, False, True, False, True, False, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, False, True, True, False, True, True, False, True, True, True, True, False, True, False, True, True, False, True, True, False, True, False, True, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, False, True, True, True, True, True, True, True, False, True, True, False, False, True, True, True, False, True, False, True, False, True, True, True, False, True, True, True, False, True, True, False, True, True, True, True, True, False, True, True, True, True, False, True, True, False, True, True, False, True, True, True, True, True, False, True, True, False, True, True, False, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, False, True, True, True, True, False, True, False, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, False, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, True, False, True, True, True, True, True, True, True, False, True, True, False, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, False, True, True, True, True, False, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, False, True, True, True, False, True, True, True, True, True, True, True, True, False, True, False, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, False, True, True, True, False, True, False, True, True, True, True, True, False, True, False, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, False, False, True, False, True, True, False, True, True, False, True, True, True, False, True, True, True, True, False, True, False, True, True, False, False, True, True, True, False, True, True, True, True, True, True, False, True, False, False, True, True, True, False, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, False, True, True, False, True, True, False, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, False, False, True, True, True, True, True, False, True, True, False, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, False, True, True, True, True, True, False, False, True, True, True, True, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, False, True, True, False, False, True, True, True, True, True, True, False, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, False, True, True, False, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, False, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, False, False, True, True, True, False, True, True, True, True, True, True, True, True, True, False, True, True, True, False, True, True, True, True, False, True, False, True, True, False, True, False, True, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]

```python
In [12]:  del new_test['F21'] # Delete the column "F21"
```

```python
In [13]:  new_test.insert(20, "Class", rs) # Insert target variable in the new_test data
```

```python
In [14]:  splitx = new_test.drop('Class', axis = 1)  # Excluding target variable from the new_test data
```

```python
In [15]:  splity = new_test['Class'] # target variable
```

```python
In [16]:  # Split the dataset into train and test subsets
          splitx_train, splitx_test, splity_train, splity_test = train_test_split(splitx, splity, test_size = 0.3, random_state = 0)
          # To train the model, we use splitx_train as the features and splity_train as true labels.
          # To test the model, we use splitx_test as the features and splity_test is used to validate the predicted labels
          # Test size is the percentage of data that should be used for testing/validation.
```

```python
In [17]:  from sklearn.metrics import confusion_matrix
          from sklearn.metrics import classification_report
          from sklearn.metrics import f1_score , recall_score , precision_score
          from sklearn.metrics import plot_confusion_matrix
```

```python
In [18]:  # Apply Gradient Boosting Classifier to determine accuracy of the test model
          from sklearn.ensemble import GradientBoostingClassifier
          GRA=GradientBoostingClassifier(n_estimators=200, random_state= 1, learning_rate=0.01)
          GRA.fit(splitx_train,splity_train)
          y_predict = GRA.predict(splitx_test)
          print(confusion_matrix(splity_test,y_predict))
          print(classification_report(splity_test,y_predict))
          print(accuracy_score(splity_test, y_predict))
          plot_confusion_matrix(GRA,splitx_test,splity_test, cmap = plt.cm.Blues)
          plt.show()
```

```
[[135   0]
 [  2 163]]
              precision    recall  f1-score   support

       False       0.99      1.00      0.99       135
        True       1.00      0.99      0.99       165

    accuracy                           0.99       300
   macro avg       0.99      0.99      0.99       300
weighted avg       0.99      0.99      0.99       300

0.9933333333333333
```