jupyter SONAR Rock vs Mine Prediction Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTree
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
C:\Users\Lenovo\Desktop\Machine Learning Projects\mach%20ML\Rock_vs_Mine_Prediction\pyg of sonar data - Copy of sonar data.
```

In [2]:
```
df = pd.read_csv('C:\Users\Machine Learning Projects\mach%20ML\Rock_vs_Mine_Prediction\pyg of sonar data - Copy of sonar data.
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	0.0907	0.2857	0.3748	0.5744	0.6572	0.6747	0.6915	0.6004	0.5060	0.482
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2913	0.3964	0.3888	0.3671	0.4274	0.5929	0.6884	0.7690	0.7274	0.7232	0.6448	0.5652	0.5351	0.5002
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5904	0.5616	0.6552	0.7677	0.8056	0.8929	0.8096	0.7950	0.7432	0.6465	0.5772	0.4861
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.1084	0.2290	0.2929	0.2492	0.2351	0.3950	0.5548	0.6426	0.6064	0.4709	0.4239	0.4078	0.4797	0.5783	0.5071
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1884	0.3039	0.2988	0.4250	0.6343	0.8073	0.9270	0.9772	0.9616	0.9589	0.9274	0.8384	0.7482	0.548
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
203	0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	0.3179	0.4419	0.5883	0.7775	0.8633	0.9173	0.9975	0.9054	0.8078	0.7624
204	0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	0.3085	0.4227	0.5832	0.6481	0.6372	0.6224	0.6019	0.6062	0.5092	0.348
205	0.0522	0.0437	0.0369	0.0482	0.0548	0.0921	0.1073	0.2274	0.3929	0.5190	0.6318	0.6655	0.6807	0.6364	0.6843	0.8670	0.9097	0.8004	0.6307	0.4372
206	0.0544	0.0320	0.0419	0.0365	0.0491	0.0787	0.1743	0.2339	0.3884	0.5715	0.6748	0.7267	0.7432	0.8673	0.9463	0.9544	0.8884	0.8253	0.6879	0.5929
207	0.0395	0.0369	0.0415	0.0314	0.0375	0.0744	0.1913	0.2811	0.3625	0.4488	0.5351	0.6239	0.7507	0.8929	0.9381	0.8592	0.6884	0.5690	0.5267	0.4251

208 rows × 61 columns

In [3]: df.describe()

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	50	51	52
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	...	208.000000	208.000000	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208811	...	0.016069	0.013420	0.010709
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	...	0.012008	0.009634	0.007060
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005300	0.007300	0.011300	...	0.000900	0.000500	0.001000
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	...	0.008425	0.007275	0.005075
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.107250	0.112100	0.152250	0.182400	...	0.013200	0.010400	0.007500
75%	0.035550	0.047950	0.057500	0.064500	0.100275	0.134125	0.154400	0.169600	0.233425	0.268700	...	0.020425	0.016725	0.012925
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382000	0.372900	0.459000	0.682800	0.710600	...	0.100400	0.070800	0.039000

8 rows × 60 columns

In [4]: df.info()

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208 entries, 0 to 207
Data columns (total 61 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       208 non-null    float64
 1   1       208 non-null    float64
 2   2       208 non-null    float64
 3   3       208 non-null    float64
 4   4       208 non-null    float64
 5   5       208 non-null    float64
 6   6       208 non-null    float64
 7   7       208 non-null    float64
 8   8       208 non-null    float64
 9   9       208 non-null    float64
 10  10      208 non-null    float64
 11  11      208 non-null    float64
 12  12      208 non-null    float64
 13  13      208 non-null    float64
 14  14      208 non-null    float64
 15  15      208 non-null    float64
 16  16      208 non-null    float64
 17  17      208 non-null    float64
 18  18      208 non-null    float64
 19  19      208 non-null    float64
 20  20      208 non-null    float64
 21  21      208 non-null    float64
 22  22      208 non-null    float64
 23  23      208 non-null    float64
 24  24      208 non-null    float64
 25  25      208 non-null    float64
 26  26      208 non-null    float64
 27  27      208 non-null    float64
 28  28      208 non-null    float64
 29  29      208 non-null    float64
 30  30      208 non-null    float64
 31  31      208 non-null    float64
 32  32      208 non-null    float64
 33  33      208 non-null    float64
 34  34      208 non-null    float64
 35  35      208 non-null    float64
 36  36      208 non-null    float64
 37  37      208 non-null    float64
 38  38      208 non-null    float64
 39  39      208 non-null    float64
 40  40      208 non-null    float64
 41  41      208 non-null    float64
 42  42      208 non-null    float64
 43  43      208 non-null    float64
 44  44      208 non-null    float64
 45  45      208 non-null    float64
 46  46      208 non-null    float64
 47  47      208 non-null    float64
 48  48      208 non-null    float64
 49  49      208 non-null    float64
 50  50      208 non-null    float64
 51  51      208 non-null    float64
 52  52      208 non-null    float64
 53  53      208 non-null    float64
 54  54      208 non-null    float64
 55  55      208 non-null    float64
 56  56      208 non-null    float64
 57  57      208 non-null    float64
 58  58      208 non-null    float64
 59  59      208 non-null    float64
 60  60      208 non-null    object
dtypes: float64(60), object(1)
memory usage: 99.3 KB
```

In [5]: df[60].value_counts()

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

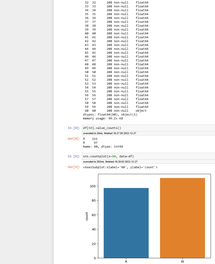
Out[5]:

```
M    111
R    97
Name: 60, dtype: int64
```

In [6]: sns.countplot(x=60, data=df)

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

Out[6]:



In [7]: df.groupby(60).mean()

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	50	51	52
60														
R	0.034989	0.045568	0.050192	0.059240	0.080715	0.107091	0.112021	0.101967	0.123063	0.160591	...	0.012532	0.009474	0.007543
M	0.022498	0.032465	0.038661	0.046447	0.062828	0.096224	0.114180	0.117836	0.157302	0.192293	...	0.012129	0.010402	0.006981

2 rows × 60 columns

In [8]: X = df.drop(columns= -1, axis = 1)
y = df[-1]

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

In [9]: X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size = 0.2 , shuffle = True , random_state = 1 , stratify= y)

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

In [10]:
```
train_data_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(train_data_prediction , y_train)
```

In [11]: print(training_data_accuracy * 100) - 0

Out[11]: 84.37499999999999

In [12]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print("Accuracy Score: {:.4f%}".format(accuracy_score(y_train, pred) * 100 ,PCT))
        print("_______________________________________________")
        print("CLASSIFICATION REPORT:\n{clf_report}".format(clf_report=clf_report))
        print("_______________________________________________")
        print("Confusion Matrix: \n {confusion_matrix(y_train, pred)}".)

    elif train:=False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print("Accuracy Score: {:.4f%}".format(accuracy_score(y_test, pred) * 100 ,PCT))
        print("_______________________________________________")
        print("CLASSIFICATION REPORT:\n{clf_report}".format(clf_report=clf_report))
        print("_______________________________________________")
        print("Confusion Matrix: \n {confusion_matrix(y_test, pred)}".)
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

In [13]:
```
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support
           R       1.0     1.0     1.0     1.0     1.0
           M       1.0     1.0     1.0     1.0     1.0
    accuracy       1.0     1.0     1.0     1.0     1.0
   macro avg       1.0     1.0     1.0     1.0     166.0
weighted avg       1.0     1.0     1.0     1.0     166.0
_______________________________________________
Confusion Matrix:
 [[ 94   0]
 [  0 72]]

Test Result:
================================================
Accuracy Score: 69.05%
_______________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support
           R       0.76666 0.68000 0.69333 0.72000 0.69333
           M       0.59000 0.69000 0.69076 0.65000 0.69076
    accuracy       0.69000 0.69000 0.69000 0.69000 0.69000
   macro avg       0.68000 0.69000 0.69000 0.68500 42.00000
weighted avg       0.69000 0.69000 0.69000 0.69200 42.00000
_______________________________________________
Confusion Matrix:
 [[17 8]
 [18 39]]
```

In [14]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

params = {
    "criterion":("gini", "entropy"),
    "splitter":("best", "random"),
    "max_depth":(list(range(1, 3))),
    "min_samples_split":(1, 5, 10),
    "min_samples_leaf":(list(range(1, 20)),
}

tree_clf = DecisionTreeClassifier(random_state=42)
tree_cv = GridSearchCV(tree_clf, params, scoring="accuracy", n_jobs=-1, verbose=1, cv=3)
tree_cv.fit(X_train, y_train)
best_params = tree_cv.best_params_
print("Best parameters: {best_params}")

tree_clf = DecisionTreeClassifier(**best_params)
tree_clf.fit(X_train, y_train)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

```
Fitting 3 folds for each of 612 candidates, totalling 1836 fits
Best parameters: {'criterion': 'entropy', 'max_depth': 1, 'min_samples_leaf': 4, 'min_samples_split': 1, 'splitter': 'rando
Train Result:
================================================
Accuracy Score: 86.75%
_______________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support
           R       0.83396 0.91000 0.87017 0.91000 0.87017
           M       0.91304 0.79130 0.85767 0.85130 0.85767
    accuracy       0.86750 0.86750 0.86750 0.86750 0.86750
   macro avg       0.86900 0.85065 0.85767 0.88065 166.00000
weighted avg       0.86900 0.86750 0.85767 0.88750 166.00000
_______________________________________________
Confusion Matrix:
 [[ 91  3]
 [ 19 53]]

Test Result:
================================================
Accuracy Score: 71.43%
_______________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support
           R       0.72727 0.72000 0.72364 0.73000 0.72364
           M       0.70370 0.71429 0.70894 0.71000 0.70894
    accuracy       0.71429 0.71429 0.71429 0.71429 0.71429
   macro avg       0.71548 0.71714 0.71629 0.72000 42.00000
weighted avg       0.71429 0.71429 0.71429 0.71429 42.00000
_______________________________________________
Confusion Matrix:
 [[18 7]
 [12 15]]
```

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
rf_clf = RandomForestClassifier(random_state=42)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
              precision    R    M  f1-score  macro avg  weighted avg
           R        1.0  1.0  1.0       1.0        1.0           1.0
           M        1.0  1.0  1.0       1.0        1.0           1.0
    accuracy        1.0  1.0  1.0       1.0        1.0           1.0
   macro avg        1.0  1.0  1.0       1.0        1.0           1.0
weighted avg        1.0  1.0  1.0       1.0        1.0           1.0
_______________________________________________
Confusion Matrix:
 [[ 94  0]
 [  0 72]]

Test Result:
================================================
Accuracy Score: 76.57%
_______________________________________________
CLASSIFICATION REPORT:
              precision    recall  f1-score   support
           R       0.76000 0.82222 0.78724 0.79000 0.79024
           M       0.77551 0.70370 0.73786 0.76000 0.74880
    accuracy       0.76571 0.76571 0.76571 0.76571 0.76571
   macro avg       0.76776 0.76296 0.76255 0.77500 42.00000
weighted avg       0.76714 0.76571 0.76571 0.77357 42.00000
_______________________________________________
Confusion Matrix:
 [[19 4]
 [12 27]]
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

n_estimators = [int(x) for x in np.linspace(start=200, stop=2000, num=10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num=11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {'n_estimators': n_estimators, 'max_features': max_features,
               'max_depth': max_depth, 'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf, 'bootstrap': bootstrap}

rf_clf = RandomForestClassifier(random_state=42)

rf_cv = GridSearchCV(rf_clf, param_grid, scoring="f1", cv=3, verbose=1,n_jobs= -1)

rf_cv.fit(X_train, y_train)
rf_best_params = rf_cv.best_params_
print("Best parameters: {rf_best_params}")

rf_clf = RandomForestClassifier(**rf_best_params)
rf_clf.fit(X_train, y_train)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
C:\Users\Lenovo\Desktop\Rock-Mine-Prediction-Machine-Learning-Project-main
```

```
Fitting 3 folds for each of 540 candidates, totalling 1620 fits

D:\Programs\Lib\site-packages\sklearn\model_selection\_search.py:305: UserWarning: One or more of the test scores are no
n-finite: [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 
```