titanic-assignment-2

April 10, 2024

[10]:
```
import os
import pandas as pnd
import numpy as nupy
import seaborn as sns
import matplotlib.pyplot as plot
```

[11]:
```
url1 = 'https://raw.githubusercontent.com/Shourya0712/titanicDT/main/
  ↪titanic_train.csv'
url2 = 'https://raw.githubusercontent.com/Shourya0712/titanicDT/main/test.csv'
train_set = pnd.read_csv(url1)
test_set = pnd.read_csv(url2)
```

[12]:
```
train_set.head()
```

[12]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[14]:
```
train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

[15]:
```
train_set.isnull().sum()
```

[15]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[16]:
```
train_set['Age'] = train_set['Age'].fillna(train_set['Age'].mean())
```

[18]:
```
train_set.isnull().sum()
```

[18]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
```

```
Ticket            0
Fare              0
Cabin           687
Embarked          2
dtype: int64
```

```
[19]: train_set['Embarked'] = train_set['Embarked'].fillna(train_set['Embarked'].
        ↪mode()[0])
```

```
[20]: train_set.isnull().sum()
```

```
[20]: PassengerId       0
      Survived          0
      Pclass            0
      Name              0
      Sex               0
      Age               0
      SibSp             0
      Parch             0
      Ticket            0
      Fare              0
      Cabin           687
      Embarked          0
      dtype: int64
```

```
[21]: embarked_mode = train_set['Embarked'].mode()[0]
      print(embarked_mode)
```

```
S
```

```
[22]: train_set['Embarked'] = train_set['Embarked'].fillna(embarked_mode)
```

```
[23]: train_set.isnull().sum()
```

```
[23]: PassengerId       0
      Survived          0
      Pclass            0
      Name              0
      Sex               0
      Age               0
      SibSp             0
      Parch             0
      Ticket            0
      Fare              0
      Cabin           687
      Embarked          0
      dtype: int64
```

```
[24]: train_set['Cabin_Letter'] = train_set['Cabin'].apply(lambda x: str(x)[0])
```

```
[25]: train_set['Cabin_Letter'].value_counts()
```

```
[25]: Cabin_Letter
      n    687
      C     59
      B     47
      D     33
      E     32
      A     15
      F     13
      G      4
      T      1
      Name: count, dtype: int64
```

```
[26]: X = train_set[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',␣
       ↪'Cabin_Letter', 'Embarked']]
      y = train_set['Survived']
```

```
[27]: from sklearn.preprocessing import LabelEncoder
      from sklearn.model_selection import train_test_split
      from sklearn.tree import DecisionTreeClassifier
      from sklearn import tree
```

```
[28]: le = LabelEncoder()
      train_set['Sex'] = le.fit_transform(train_set['Sex'])
```

```
[29]: X = train_set[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',␣
       ↪'Cabin_Letter', 'Embarked']]
      y = train_set['Survived']
```

```
[30]: x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
       ↪random_state=75)
```

```
[31]: train_set['Embarked'] = le.fit_transform(train_set['Embarked'])
```

```
[32]: train_set['Cabin_Letter'] = le.fit_transform(train_set['Cabin_Letter'])
```

```
[33]: train_set.head(25)
```

```
[33]:     PassengerId  Survived  Pclass  \
      0             1         0       3
      1             2         1       1
      2             3         1       3
      3             4         1       1
      4             5         0       3
```

```
5             6         0       3
6             7         0       1
7             8         0       3
8             9         1       3
9            10         1       2
10           11         1       3
11           12         1       1
12           13         0       3
13           14         0       3
14           15         0       3
15           16         1       2
16           17         0       3
17           18         1       2
18           19         0       3
19           20         1       3
20           21         0       2
21           22         1       2
22           23         1       3
23           24         1       1
24           25         0       3

                                                 Name  Sex        Age  SibSp  \
0                             Braund, Mr. Owen Harris    1  22.000000      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…    0  38.000000      1
2                              Heikkinen, Miss. Laina    0  26.000000      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)    0  35.000000      1
4                            Allen, Mr. William Henry    1  35.000000      0
5                                    Moran, Mr. James    1  29.699118      0
6                             McCarthy, Mr. Timothy J    1  54.000000      0
7                      Palsson, Master. Gosta Leonard    1   2.000000      3
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)    0  27.000000      0
9                 Nasser, Mrs. Nicholas (Adele Achem)    0  14.000000      1
10                    Sandstrom, Miss. Marguerite Rut    0   4.000000      1
11                           Bonnell, Miss. Elizabeth    0  58.000000      0
12                     Saundercock, Mr. William Henry    1  20.000000      0
13                        Andersson, Mr. Anders Johan    1  39.000000      1
14               Vestrom, Miss. Hulda Amanda Adolfina    0  14.000000      0
15                   Hewlett, Mrs. (Mary D Kingcome)     0  55.000000      0
16                               Rice, Master. Eugene    1   2.000000      4
17                       Williams, Mr. Charles Eugene    1  29.699118      0
18  Vander Planke, Mrs. Julius (Emelia Maria Vande…    0  31.000000      1
19                            Masselmani, Mrs. Fatima    0  29.699118      0
20                               Fynney, Mr. Joseph J    1  35.000000      0
21                              Beesley, Mr. Lawrence    1  34.000000      0
22                        McGowan, Miss. Anna "Annie"    0  15.000000      0
23                       Sloper, Mr. William Thompson    1  28.000000      0
24                      Palsson, Miss. Torborg Danira    0   8.000000      3
```

```
    Parch            Ticket     Fare Cabin  Embarked  Cabin_Letter
0       0         A/5 21171   7.2500   NaN         2             8
1       0          PC 17599  71.2833   C85         0             2
2       0  STON/O2. 3101282   7.9250   NaN         2             8
3       0            113803  53.1000  C123         2             2
4       0            373450   8.0500   NaN         2             8
5       0            330877   8.4583   NaN         1             8
6       0             17463  51.8625   E46         2             4
7       1            349909  21.0750   NaN         2             8
8       2            347742  11.1333   NaN         2             8
9       0            237736  30.0708   NaN         0             8
10      1           PP 9549  16.7000    G6         2             6
11      0            113783  26.5500  C103         2             2
12      0         A/5. 2151   8.0500   NaN         2             8
13      5            347082  31.2750   NaN         2             8
14      0            350406   7.8542   NaN         2             8
15      0            248706  16.0000   NaN         2             8
16      1            382652  29.1250   NaN         1             8
17      0            244373  13.0000   NaN         2             8
18      0            345763  18.0000   NaN         2             8
19      0              2649   7.2250   NaN         0             8
20      0            239865  26.0000   NaN         2             8
21      0            248698  13.0000   D56         2             3
22      0            330923   8.0292   NaN         1             8
23      0            113788  35.5000    A6         2             0
24      1            349909  21.0750   NaN         2             8
```

[34]:
```
train_set.corr(numeric_only=1)
```

[34]:
```
              PassengerId  Survived    Pclass       Sex       Age     SibSp  \
PassengerId      1.000000 -0.005007 -0.035144  0.042939  0.033207 -0.057527
Survived        -0.005007  1.000000 -0.338481 -0.543351 -0.069809 -0.035322
Pclass          -0.035144 -0.338481  1.000000  0.131900 -0.331339  0.083081
Sex              0.042939 -0.543351  0.131900  1.000000  0.084153 -0.114631
Age              0.033207 -0.069809 -0.331339  0.084153  1.000000 -0.232625
SibSp           -0.057527 -0.035322  0.083081 -0.114631 -0.232625  1.000000
Parch           -0.001652  0.081629  0.018443 -0.245489 -0.179191  0.414838
Fare             0.012658  0.257307 -0.549500 -0.182333  0.091566  0.159651
Embarked         0.013128 -0.167675  0.162098  0.108262 -0.026749  0.068230
Cabin_Letter    -0.030939 -0.301116  0.746616  0.123076 -0.249134  0.041540

                 Parch      Fare  Embarked  Cabin_Letter
PassengerId  -0.001652  0.012658  0.013128     -0.030939
Survived      0.081629  0.257307 -0.167675     -0.301116
Pclass        0.018443 -0.549500  0.162098      0.746616
Sex          -0.245489 -0.182333  0.108262      0.123076
```

```
Age          -0.179191  0.091566 -0.026749     -0.249134
SibSp         0.414838  0.159651  0.068230      0.041540
Parch         1.000000  0.216225  0.039798     -0.032548
Fare          0.216225  1.000000 -0.224719     -0.523013
Embarked      0.039798 -0.224719  1.000000      0.194255
Cabin_Letter -0.032548 -0.523013  0.194255      1.000000
```

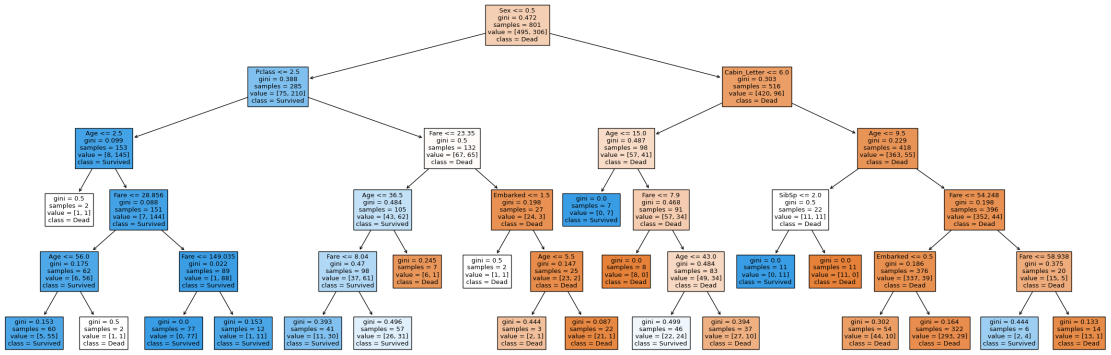
[35]:
```
X = train_set[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',␣
 ↪'Cabin_Letter', 'Embarked']]
# X = train_set[['Pclass', 'Sex', 'Age',  'SibSp', 'Parch', 'Fare', 'Embarked']]
y = train_set['Survived']
```

[36]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
 ↪random_state=65)
```

[37]:
```
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
param_grid = {
    'max_depth': [3, 5, 7, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
dt = DecisionTreeClassifier()
gs = GridSearchCV(estimator=dt, param_grid=param_grid, scoring='accuracy', cv=8)
gs.fit(x_train, y_train)

best_clf = gs.best_estimator_
y_pred = best_clf.predict(x_test)
yt_pred = best_clf.predict(x_train)
test_accuracy = accuracy_score(y_test, y_pred)
train_accuracy = accuracy_score(y_train, yt_pred)

# Print the best parameters found
print("Best Parameters:", gs.best_params_)
print("Test Accuracy:", test_accuracy)
print("Train Accuracy:", train_accuracy)

#plot tree
plot.figure(figsize=(30, 10))
tree.plot_tree(best_clf, feature_names=list(X.columns), class_names=['Dead',␣
 ↪'Survived'], filled=True)
plot.show()
```

```
Best Parameters: {'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split':
10}
Test Accuracy: 0.8222222222222222
```

```
Train Accuracy: 0.846441947565543
```


Sex <= 0.5 gini = 0.472 samples = 801 value = [495, 306] class = Dead
Pclass <= 2.5 gini = 0.388 samples = 285 value = [75, 210] class = Survived
Cabin_Letter <= 6.0 gini = 0.303 samples = 516 value = [420, 96] class = Dead
Age <= 2.5 gini = 0.099 samples = 153 value = [8, 145] class = Survived
Fare <= 23.35 gini = 0.5 samples = 132 value = [67, 65] class = Dead
Age <= 15.0 gini = 0.487 samples = 98 value = [57, 41] class = Dead
Age <= 9.5 gini = 0.229 samples = 418 value = [363, 55] class = Dead
gini = 0.5 samples = 2 value = [1, 1] class = Dead
Fare <= 28.856 gini = 0.088 samples = 151 value = [7, 144] class = Survived
Age <= 36.5 gini = 0.484 samples = 105 value = [43, 62] class = Survived
Embarked <= 1.5 gini = 0.198 samples = 27 value = [24, 3] class = Dead
gini = 0.0 samples = 7 value = [0, 7] class = Survived
Fare <= 7.9 gini = 0.468 samples = 91 value = [57, 34] class = Dead
SibSp <= 2.0 gini = 0.5 samples = 22 value = [11, 11] class = Dead
Fare <= 54.248 gini = 0.198 samples = 396 value = [352, 44] class = Dead
Age <= 56.0 gini = 0.175 samples = 62 value = [6, 56] class = Survived
Fare <= 149.035 gini = 0.022 samples = 89 value = [1, 88] class = Survived
Fare <= 8.04 gini = 0.47 samples = 98 value = [37, 61] class = Survived
gini = 0.245 samples = 7 value = [6, 1] class = Dead
gini = 0.5 samples = 2 value = [1, 1] class = Dead
Age <= 5.5 gini = 0.147 samples = 25 value = [23, 2] class = Dead
gini = 0.0 samples = 8 value = [8, 0] class = Dead
Age <= 43.0 gini = 0.484 samples = 83 value = [49, 34] class = Dead
gini = 0.0 samples = 11 value = [0, 11] class = Survived
gini = 0.0 samples = 11 value = [11, 0] class = Dead
Embarked <= 0.5 gini = 0.186 samples = 376 value = [337, 39] class = Dead
Fare <= 58.938 gini = 0.375 samples = 20 value = [15, 5] class = Dead
gini = 0.153 samples = 60 value = [5, 55] class = Survived
gini = 0.5 samples = 2 value = [1, 1] class = Dead
gini = 0.0 samples = 77 value = [0, 77] class = Survived
gini = 0.153 samples = 12 value = [1, 11] class = Survived
gini = 0.393 samples = 41 value = [11, 30] class = Survived
gini = 0.496 samples = 57 value = [26, 31] class = Survived
gini = 0.444 samples = 3 value = [2, 1] class = Dead
gini = 0.087 samples = 22 value = [21, 1] class = Dead
gini = 0.499 samples = 46 value = [22, 24] class = Survived
gini = 0.394 samples = 37 value = [27, 10] class = Dead
gini = 0.302 samples = 54 value = [44, 10] class = Dead
gini = 0.164 samples = 322 value = [293, 29] class = Dead
gini = 0.444 samples = 6 value = [2, 4] class = Survived
gini = 0.133 samples = 14 value = [13, 1] class = Dead

[38]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_features='sqrt', oob_score=True,␣
 ↪random_state=65, n_jobs=-1)
param_grid = { "criterion" : ["gini", "entropy"], "min_samples_leaf" : [1, 5,␣
 ↪10], "min_samples_split" : [2, 4, 10, 12, 16], "n_estimators": [50, 100,␣
 ↪400, 700, 1000]}
gs = GridSearchCV(estimator=rf, param_grid=param_grid, scoring='accuracy',␣
 ↪cv=3, n_jobs=-1)
gs = gs.fit(x_train, y_train)
print(gs.best_score_)
print(gs.best_params_)
print(gs.cv_results_)
```

```
0.832709113607990
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 10,
'n_estimators': 100}
{'mean_fit_time': array([0.34907699, 0.47951063, 2.31736398, 3.98057111,
6.09895651,
       0.36027638, 0.52474229, 2.27914437, 4.25119646, 4.66219838,
       0.21627617, 0.74940578, 2.28568403, 3.95806193, 5.10880907,
       0.34962797, 0.6259199 , 2.32556979, 3.81450311, 5.21995211,
       0.24948502, 0.472001  , 2.41081119, 3.69699502, 5.40499751,
       0.27779945, 0.54392282, 2.37712789, 4.43397546, 5.43011244,
       0.29737163, 0.6869212 , 2.19706726, 4.30621847, 5.26877371,
       0.31758889, 0.69795338, 2.27575501, 4.13023901, 5.31245263,
       0.31889478, 0.57440702, 2.45663484, 4.21012338, 5.39353808,
       0.2785267 , 0.48835429, 2.52662849, 4.1319441 , 5.78748178,
       0.41892974, 0.61710167, 2.29580863, 4.37137198, 5.01174005,
       0.27848403, 0.57237601, 1.91613801, 3.80944037, 5.09366051,
       0.27876981, 0.51791573, 2.13594246, 3.92401052, 5.43244489,
       0.29948306, 0.56669935, 2.12246052, 3.53022687, 4.82387249,
       0.21106871, 0.49973361, 2.12068876, 3.9803412 , 4.99692233,
       0.20691291, 0.4576896 , 2.16682291, 4.28788638, 5.75789404,
       0.23433709, 0.54917614, 2.75081452, 5.85532451, 8.07800992,
       0.37149151, 1.08914002, 3.49035962, 5.00185378, 5.90075755,
       0.30375075, 1.19137732, 2.47537692, 4.99299844, 6.15784907,
       0.54188983, 0.80590494, 3.35590744, 5.73874728, 7.42789753,
       1.16030089, 1.14225062, 3.25916338, 4.57489808, 5.17702039,
       0.32697988, 0.62200348, 2.40692266, 4.46010733, 5.71907465,
       0.28451816, 0.64166792, 2.53603514, 4.04102238, 5.74017016,
       0.35059841, 0.57539344, 2.56810355, 4.61839366, 5.50336703,
       0.27912792, 0.53297599, 2.88186971, 4.55324674, 4.98643994,
       0.2385366 , 0.54889448, 2.82010976, 3.67296338, 6.03951351,
       0.28554749, 0.55582635, 2.49368016, 4.24646179, 5.99370782,
       0.42883348, 0.52140601, 2.39965375, 4.64167245, 5.42444428,
       0.33522932, 0.69128927, 2.22550305, 4.40603908, 5.02422643,
       0.230136  , 0.48122263, 2.72996728, 3.47905207, 4.39550789]),
'std_fit_time': array([1.12464268e-01, 5.57594174e-02, 1.20206967e-01,
1.68776778e-01,
       1.61000071e-01, 1.53760859e-01, 3.89951740e-02, 1.69150551e-01,
       5.29695813e-01, 8.73800242e-02, 2.38324974e-03, 5.91352206e-02,
       8.16989046e-02, 2.75212310e-01, 1.11918563e-01, 3.01406800e-02,
       9.84429180e-02, 1.84025516e-01, 4.55329912e-01, 1.61673722e-01,
       1.76552052e-02, 3.47172241e-02, 4.70250318e-02, 3.50975611e-01,
       7.22511842e-02, 1.49320038e-02, 4.00566317e-03, 8.11429450e-02,
       1.36483417e-01, 2.55617171e-01, 4.17998863e-02, 6.58981518e-02,
       2.23041070e-02, 2.90343761e-01, 7.64214381e-02, 3.08187327e-02,
       4.54385258e-02, 1.81882714e-01, 2.53567868e-01, 3.56658238e-02,
       1.41179536e-02, 6.87518107e-02, 1.36370270e-01, 3.51896326e-01,
       4.60227047e-02, 2.35189784e-02, 3.41532555e-02, 1.60184603e-01,
       4.91200332e-01, 2.19703036e-01, 8.92604672e-02, 4.32002054e-02,
       7.28289104e-02, 2.21642966e-01, 1.05660330e-01, 2.06188822e-02,
       4.67626801e-02, 8.59685624e-02, 3.30203013e-01, 2.10595123e-02,
       6.55774442e-02, 1.88987697e-04, 1.57890856e-01, 3.96620130e-01,
       3.34399912e-01, 2.46771061e-02, 7.20176572e-02, 1.55199267e-01,
       1.24976451e-01, 3.74775959e-02, 2.19984419e-02, 6.15921178e-02,
       4.94933579e-02, 2.07145510e-01, 9.25125556e-02, 1.25637769e-02,
       5.73974246e-03, 1.65024439e-01, 2.90836797e-01, 4.15379203e-02,
       2.86660186e-02, 8.54558385e-02, 1.93403722e-01, 2.18034673e-01,
       3.99167500e-01, 5.82273063e-02, 2.01000079e-01, 2.40141249e-01,
       4.66884526e-02, 6.43366914e-02, 1.94006160e-02, 1.41796343e-01,
       7.04611781e-02, 7.89087695e-02, 2.06671311e-01, 1.12931124e-01,
       1.30573756e-01, 2.88508474e-01, 1.50900073e-01, 6.34725883e-01,
       2.41328112e-02, 1.78824720e-01, 3.29695547e-01, 1.96986635e-01,
       3.05597187e-01, 4.61556821e-02, 5.66517471e-02, 7.18861920e-02,
       2.45235555e-01, 1.02329925e-01, 1.66576941e-02, 5.73928423e-02,
       1.51583493e-01, 2.31795028e-01, 2.92754105e-01, 6.68537362e-02,
       2.20297964e-02, 1.28707520e-01, 1.13555728e-01, 4.15893300e-02,
       3.38400664e-02, 8.12329833e-02, 2.98516949e-01, 1.02462208e-01,
       5.88649860e-02, 9.88760229e-03, 3.96751115e-02, 7.11480814e-02,
       2.54782797e-01, 1.42674259e-01, 1.53792155e-02, 6.85390376e-02,
       3.44972507e-01, 3.04562958e-01, 1.95407675e-01, 4.40913281e-02,
       1.52272865e-02, 1.22376578e-01, 2.67365086e-01, 1.60440915e-01,
       4.67156520e-02, 7.35830989e-02, 1.50267513e-01, 1.43029233e-01,
       5.49922954e-02, 2.17485606e-02, 6.12020502e-02, 1.49538646e-01,
       7.69389358e-02, 1.29425880e-01]), 'mean_score_time': array([0.07711196,
0.08096687, 0.32944274, 0.29442318, 0.43775996,
       0.08379698, 0.11320901, 0.2487576 , 0.27256298, 0.35280506,
       0.05271618, 0.17230829, 0.18314902, 0.3291719 , 0.39053027,
       0.12490447, 0.09097028, 0.16408269, 0.28727889, 0.399671  ,
       0.05670389, 0.05697219, 0.15479819, 0.29518096, 0.44616628,
       0.08205342, 0.09562834, 0.19699216, 0.35467148, 0.41472419,
       0.06919797, 0.11790951, 0.26417971, 0.34450841, 0.40684613,
       0.08987848, 0.20189246, 0.24750662, 0.32168913, 0.46569705,
       0.11429588, 0.13125475, 0.23503025, 0.31150985, 0.45176776,
       0.05597377, 0.08360664, 0.24218313, 0.35426116, 0.47621258,
       0.1165568 , 0.1625975 , 0.18177072, 0.26157125, 0.36082975,
       0.08935189, 0.08418417, 0.15565189, 0.31823826, 0.40727679,
       0.06103611, 0.07836707, 0.18871578, 0.30734682, 0.3452247 ,
       0.0604225 , 0.10544745, 0.14067586, 0.2788736 , 0.35378941,
       0.04665033, 0.05083148, 0.15398073, 0.26300907, 0.38637503,
       0.04271221, 0.07267698, 0.21537844, 0.33825739, 0.55572422,
       0.05761425, 0.10964076, 0.28049207, 0.86732491, 0.43971801,
       0.18800759, 0.66448776, 0.24887482, 0.57451781, 0.48641094,
       0.21989115, 0.27687534, 0.47354674, 0.64961616, 0.68787487,
       0.21455288, 0.25737238, 0.52310205, 0.8204987 , 0.6312685 ,
       0.48763021, 0.43758742, 0.46468584, 0.37977648, 0.43555037,
       0.1445988 , 0.16519125, 0.18836737, 0.36724345, 0.41921465,
       0.10754943, 0.12155541, 0.24816537, 0.38215685, 0.4416879 ,
       0.08359226, 0.14964962, 0.21288347, 0.52272979, 0.42648943,
       0.06958071, 0.198409  , 0.19633802, 0.38641079, 0.37993765,
       0.08408618, 0.06604942, 0.17933544, 0.31487139, 0.39437453,
       0.05242268, 0.09232728, 0.21564444, 0.32590151, 0.42640599,
       0.23722919, 0.07157405, 0.18787479, 0.33597199, 0.40464449,
       0.13028892, 0.1198624 , 0.20513717, 0.31397923, 0.4381841 ,
       0.04061214, 0.08443387, 0.18439611, 0.26216332, 0.28720371]),
'std_score_time': array([2.19927663e-02, 1.58139964e-02, 1.97713860e-01,
9.87385236e-03,
       2.82018201e-02, 1.97571034e-02, 5.36739618e-02, 2.49870308e-02,
       2.68125191e-02, 5.80711862e-03, 7.95396681e-03, 7.96676603e-02,
       3.89848372e-02, 4.21151136e-02, 8.55290254e-03, 2.48702787e-02,
       2.02194032e-02, 6.49068034e-03, 1.46745211e-02, 2.23158553e-02,
       7.09034852e-03, 8.50757906e-03, 1.32636633e-02, 2.50112815e-02,
       5.34269568e-03, 1.72290643e-02, 1.10408299e-02, 1.19894405e-02,
       1.57893290e-02, 2.16898410e-02, 1.55755438e-02, 2.56058298e-02,
       2.40378650e-02, 2.08309932e-02, 4.44569837e-02, 2.56919739e-02,
       5.33135703e-02, 4.33365734e-02, 2.18267827e-02, 2.30406342e-02,
       3.37985991e-02, 4.95329771e-02, 2.99436461e-02, 3.14909807e-02,
       2.51847262e-02, 7.36726912e-03, 4.85699533e-03, 2.64297797e-02,
       4.44409254e-02, 7.88623580e-02, 3.86775562e-02, 2.60799813e-02,
       1.19067719e-02, 7.27470875e-03, 3.39550767e-02, 3.73847609e-02,
       2.96382549e-02, 1.27994130e-02, 7.46115682e-02, 6.30948893e-02,
       1.05391442e-02, 2.31152198e-02, 3.43078358e-02, 1.68741924e-02,
       4.79482404e-03, 7.30741234e-03, 2.05274024e-02, 1.26092106e-02,
       3.57998030e-02, 5.41304628e-03, 2.67030322e-04, 8.78843037e-03,
       9.30805876e-03, 1.40993733e-02, 2.21704439e-02, 8.19571020e-03,
       1.52250808e-02, 3.63830615e-02, 3.46463078e-02, 1.14844987e-01,
       7.58161971e-03, 1.27662734e-02, 5.20677175e-02, 4.58443529e-02,
       2.31286199e-02, 1.04006218e-01, 1.03565203e-01, 5.58034113e-02,
       1.84338951e-01, 1.27646994e-02, 1.01791175e-01, 9.56059527e-02,
       2.63626162e-01, 1.40178777e-01, 1.49566583e-01, 7.83353083e-02,
       8.39756393e-02, 1.25109709e-01, 2.68824656e-01, 7.86175690e-02,
       1.64321010e-01, 8.85265529e-02, 8.47155974e-02, 4.23160356e-02,
       4.43960291e-02, 6.36678443e-03, 6.96982122e-02, 2.36022352e-06,
       5.12405446e-02, 9.21195735e-03, 1.48048166e-02, 2.71938884e-02,
       8.55341813e-02, 8.63331643e-02, 4.90917606e-02, 4.07892563e-02,
       6.76998601e-02, 2.99933264e-02, 1.56346567e-01, 3.81433478e-02,
       2.76622846e-02, 6.26029469e-02, 2.89544728e-02, 1.54555378e-01,
       4.21794623e-03, 2.88835448e-02, 8.85023566e-03, 1.23256105e-02,
       9.76763894e-03, 2.43717476e-02, 7.60827940e-03, 1.41773479e-02,
       5.10028410e-02, 3.47707091e-02, 1.16985244e-02, 9.07849177e-02,
       1.44355512e-02, 2.36202363e-02, 3.25458526e-02, 1.50185974e-02,
       7.79409869e-03, 2.93305056e-02, 1.93729122e-02, 1.66387053e-02,
       1.65790552e-02, 4.72336921e-03, 6.94164161e-03, 1.05235467e-02,
       1.24236942e-02, 3.44467268e-03]), 'param_criterion':
masked_array(data=['gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
                   'gini', 'gini', 'gini', 'gini', 'gini', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
                   'entropy', 'entropy', 'entropy', 'entropy'],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_min_samples_leaf': masked_array(data=[1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                   1, 1, 1, 1, 1, 1, 1, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 1, 1, 1, 1, 1, 1, 1, 1,
                   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_min_samples_split': masked_array(data=[2, 2,
2, 2, 2, 4, 4, 4, 4, 4, 10, 10, 10, 10, 10, 12,
                   12, 12, 12, 12, 16, 16, 16, 16, 16, 2, 2, 2, 2, 2, 4,
                   4, 4, 4, 4, 10, 10, 10, 10, 10, 12, 12, 12, 12, 12, 16,
                   16, 16, 16, 16, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 10, 10,
                   10, 10, 10, 12, 12, 12, 12, 12, 16, 16, 16, 16, 16, 2,
                   2, 2, 2, 2, 4, 4, 4, 4, 4, 10, 10, 10, 10, 10, 12, 12,
                   12, 12, 12, 16, 16, 16, 16, 16, 2, 2, 2, 2, 2, 4, 4, 4,
                   4, 4, 10, 10, 10, 10, 10, 12, 12, 12, 12, 12, 16, 16,
                   16, 16, 16, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 10, 10, 10,
                   10, 10, 12, 12, 12, 12, 12, 16, 16, 16, 16, 16],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_n_estimators': masked_array(data=[50, 100,
400, 700, 1000, 50, 100, 400, 700, 1000, 50,
                   100, 400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100,
                   400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100, 400,
                   700, 1000, 50, 100, 400, 700, 1000, 50, 100, 400, 700,
                   1000, 50, 100, 400, 700, 1000, 50, 100, 400, 700, 1000,
                   50, 100, 400, 700, 1000, 50, 100, 400, 700, 1000, 50,
                   100, 400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100,
                   400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100, 400,
                   700, 1000, 50, 100, 400, 700, 1000, 50, 100, 400, 700,
                   1000, 50, 100, 400, 700, 1000, 50, 100, 400, 700, 1000,
                   50, 100, 400, 700, 1000, 50, 100, 400, 700, 1000, 50,
                   100, 400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100,
                   400, 700, 1000, 50, 100, 400, 700, 1000, 50, 100, 400,
                   700, 1000, 50, 100, 400, 700, 1000],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'params': [{'criterion': 'gini', 'min_samples_leaf':
1, 'min_samples_split': 2, 'n_estimators': 50}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 400}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 700}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 1000},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 4,
'n_estimators': 50}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 4, 'n_estimators': 100}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 400},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 4,
'n_estimators': 700}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 4, 'n_estimators': 1000}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 10,
'n_estimators': 100}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 10, 'n_estimators': 400}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 700},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 10,
'n_estimators': 1000}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 12, 'n_estimators': 50}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 12, 'n_estimators': 100},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 12,
'n_estimators': 400}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 12, 'n_estimators': 700}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 12, 'n_estimators': 1000},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 16,
'n_estimators': 50}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 16, 'n_estimators': 100}, {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 16, 'n_estimators': 400},
{'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 16,
'n_estimators': 700}, {'criterion': 'gini', 'min_samples_leaf': 1,
'min_samples_split': 16, 'n_estimators': 1000}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 50},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 2,
'n_estimators': 100}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 2, 'n_estimators': 400}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 700},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 2,
'n_estimators': 1000}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 4, 'n_estimators': 50}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 4, 'n_estimators': 100},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 4,
'n_estimators': 400}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 4, 'n_estimators': 700}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 4, 'n_estimators': 1000},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 10,
'n_estimators': 50}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 10, 'n_estimators': 100}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 10, 'n_estimators': 400},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 10,
'n_estimators': 700}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 10, 'n_estimators': 1000}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 50},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 12,
'n_estimators': 100}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 12, 'n_estimators': 400}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 700},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 12,
'n_estimators': 1000}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 16, 'n_estimators': 50}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 16, 'n_estimators': 100},
{'criterion': 'gini', 'min_samples_leaf': 5, 'min_samples_split': 16,
'n_estimators': 400}, {'criterion': 'gini', 'min_samples_leaf': 5,
'min_samples_split': 16, 'n_estimators': 700}, {'criterion': 'gini',
'min_samples_leaf': 5, 'min_samples_split': 16, 'n_estimators': 1000},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 2,
'n_estimators': 50}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 2, 'n_estimators': 100}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 2, 'n_estimators': 400},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 2,
'n_estimators': 700}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 2, 'n_estimators': 1000}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 4, 'n_estimators': 50},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 4,
'n_estimators': 100}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 4, 'n_estimators': 400}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 4, 'n_estimators': 700},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 4,
'n_estimators': 1000}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 10, 'n_estimators': 50}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 100},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 10,
'n_estimators': 400}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 10, 'n_estimators': 700}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 1000},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 12,
'n_estimators': 50}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 12, 'n_estimators': 100}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 12, 'n_estimators': 400},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 12,
'n_estimators': 700}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 12, 'n_estimators': 1000}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 16, 'n_estimators': 50},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 16,
'n_estimators': 100}, {'criterion': 'gini', 'min_samples_leaf': 10,
'min_samples_split': 16, 'n_estimators': 400}, {'criterion': 'gini',
'min_samples_leaf': 10, 'min_samples_split': 16, 'n_estimators': 700},
{'criterion': 'gini', 'min_samples_leaf': 10, 'min_samples_split': 16,
'n_estimators': 1000}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 50}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 400}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 700}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 1000},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 4,
'n_estimators': 50}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 4, 'n_estimators': 100}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 400},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 4,
'n_estimators': 700}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 4, 'n_estimators': 1000}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 10,
'n_estimators': 100}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 10, 'n_estimators': 400}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 700},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 10,
'n_estimators': 1000}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 12, 'n_estimators': 50}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 12, 'n_estimators': 100},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 12,
'n_estimators': 400}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 12, 'n_estimators': 700}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 12, 'n_estimators': 1000},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 16,
'n_estimators': 50}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 16, 'n_estimators': 100}, {'criterion': 'entropy',
'min_samples_leaf': 1, 'min_samples_split': 16, 'n_estimators': 400},
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 16,
'n_estimators': 700}, {'criterion': 'entropy', 'min_samples_leaf': 1,
'min_samples_split': 16, 'n_estimators': 1000}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 50},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 2,
'n_estimators': 100}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 2, 'n_estimators': 400}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 700},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 2,
'n_estimators': 1000}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 4, 'n_estimators': 50}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 4, 'n_estimators': 100},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 4,
'n_estimators': 400}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 4, 'n_estimators': 700}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 4, 'n_estimators': 1000},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 10,
'n_estimators': 50}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 10, 'n_estimators': 100}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 10, 'n_estimators': 400},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 10,
'n_estimators': 700}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 10, 'n_estimators': 1000}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 50},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 12,
'n_estimators': 100}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 12, 'n_estimators': 400}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 700},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 12,
'n_estimators': 1000}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 16, 'n_estimators': 50}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 16, 'n_estimators': 100},
{'criterion': 'entropy', 'min_samples_leaf': 5, 'min_samples_split': 16,
'n_estimators': 400}, {'criterion': 'entropy', 'min_samples_leaf': 5,
'min_samples_split': 16, 'n_estimators': 700}, {'criterion': 'entropy',
'min_samples_leaf': 5, 'min_samples_split': 16, 'n_estimators': 1000},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 2,
'n_estimators': 50}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 2, 'n_estimators': 100}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 2, 'n_estimators': 400},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 2,
'n_estimators': 700}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 2, 'n_estimators': 1000}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 4, 'n_estimators': 50},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 4,
'n_estimators': 100}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 4, 'n_estimators': 400}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 4, 'n_estimators': 700},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 4,
'n_estimators': 1000}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 10, 'n_estimators': 50}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 100},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 10,
'n_estimators': 400}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 10, 'n_estimators': 700}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 1000},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 12,
'n_estimators': 50}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 12, 'n_estimators': 100}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 12, 'n_estimators': 400},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 12,
'n_estimators': 700}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 12, 'n_estimators': 1000}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 16, 'n_estimators': 50},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 16,
'n_estimators': 100}, {'criterion': 'entropy', 'min_samples_leaf': 10,
'min_samples_split': 16, 'n_estimators': 400}, {'criterion': 'entropy',
'min_samples_leaf': 10, 'min_samples_split': 16, 'n_estimators': 700},
{'criterion': 'entropy', 'min_samples_leaf': 10, 'min_samples_split': 16,
'n_estimators': 1000}], 'split0_test_score': array([0.8164794 , 0.82022472,
0.81273408, 0.82022472, 0.82022472,
       0.80898876, 0.81273408, 0.8164794 , 0.8164794 , 0.8164794 ,
       0.81273408, 0.80898876, 0.8164794 , 0.82022472, 0.82022472,
       0.80149813, 0.80898876, 0.82022472, 0.82397004, 0.82397004,
       0.83146067, 0.82397004, 0.82022472, 0.8164794 , 0.8164794 ,
       0.82022472, 0.81273408, 0.82397004, 0.82397004, 0.81273408,
       0.82022472, 0.81273408, 0.82397004, 0.82397004, 0.81273408,
       0.82022472, 0.81273408, 0.82397004, 0.82397004, 0.81273408,
       0.80898876, 0.81273408, 0.82022472, 0.82022472, 0.8164794 ,
       0.82397004, 0.8164794 , 0.82022472, 0.82022472, 0.82022472,
       0.83146067, 0.82771536, 0.82771536, 0.82771536, 0.82397004,
       0.83146067, 0.82771536, 0.82771536, 0.82771536, 0.82397004,
       0.83146067, 0.82771536, 0.82771536, 0.82771536, 0.82397004,
       0.83146067, 0.82771536, 0.82771536, 0.82771536, 0.82397004,
       0.83146067, 0.82771536, 0.82771536, 0.82771536, 0.82397004,
       0.82022472, 0.8164794 , 0.83146067, 0.82397004, 0.82771536,
       0.80898876, 0.82022472, 0.82397004, 0.82022472, 0.82022472,
       0.80898876, 0.8164794 , 0.82022472, 0.82022472, 0.81273408,
       0.80524345, 0.81273408, 0.82022472, 0.81273408, 0.8164794 ,
       0.79775281, 0.80149813, 0.8164794 , 0.81273408, 0.81273408,
       0.82397004, 0.82022472, 0.8164794 , 0.8164794 , 0.8164794 ,
       0.82397004, 0.82022472, 0.8164794 , 0.8164794 , 0.8164794 ,
       0.82397004, 0.82022472, 0.8164794 , 0.8164794 , 0.8164794 ,
       0.82022472, 0.82771536, 0.82397004, 0.82022472, 0.8164794 ,
       0.82022472, 0.82022472, 0.82022472, 0.82397004, 0.82022472,
       0.82022472, 0.82022472, 0.82771536, 0.83520599, 0.82397004,
       0.82022472, 0.82022472, 0.82771536, 0.83520599, 0.82397004,
       0.82022472, 0.82022472, 0.82771536, 0.83520599, 0.82397004,
       0.82022472, 0.82022472, 0.82771536, 0.83520599, 0.82397004,
       0.82022472, 0.82022472, 0.82771536, 0.83520599, 0.82397004]),
'split1_test_score': array([0.83895131, 0.83520599, 0.84269663, 0.84269663,
0.83895131,
       0.84644195, 0.84269663, 0.84644195, 0.84269663, 0.84644195,
       0.86516854, 0.86142322, 0.85018727, 0.85018727, 0.85018727,
       0.86516854, 0.8576779 , 0.85018727, 0.85018727, 0.85018727,
       0.83895131, 0.84269663, 0.85018727, 0.84644195, 0.84644195,
       0.84644195, 0.84644195, 0.84644195, 0.84269663, 0.84644195,
       0.84644195, 0.84644195, 0.84644195, 0.84269663, 0.84644195,
       0.84644195, 0.84644195, 0.84644195, 0.84269663, 0.84644195,
       0.85393258, 0.84269663, 0.85018727, 0.84269663, 0.84644195,
       0.84644195, 0.85018727, 0.84269663, 0.83895131, 0.84269663,
       0.82771536, 0.83895131, 0.82022472, 0.82397004, 0.82397004,
       0.82771536, 0.83895131, 0.82022472, 0.82397004, 0.82397004,
       0.82771536, 0.83895131, 0.82022472, 0.82397004, 0.82397004,
       0.82771536, 0.83895131, 0.82022472, 0.82397004, 0.82397004,
       0.82771536, 0.83895131, 0.82022472, 0.82397004, 0.82397004,
       0.83895131, 0.85018727, 0.84269663, 0.84269663, 0.84644195,
       0.84644195, 0.84644195, 0.84644195, 0.85393258, 0.85393258,
       0.86142322, 0.8576779 , 0.85018727, 0.85018727, 0.8576779 ,
       0.8576779 , 0.85393258, 0.86142322, 0.8576779 , 0.8576779 ,
       0.84644195, 0.85393258, 0.85018727, 0.84644195, 0.84644195,
       0.86142322, 0.85393258, 0.84644195, 0.84644195, 0.84644195,
       0.86142322, 0.85393258, 0.84644195, 0.84644195, 0.84644195,
       0.86142322, 0.85393258, 0.84644195, 0.84644195, 0.84644195,
       0.84269663, 0.84644195, 0.84269663, 0.84269663, 0.84269663,
       0.83520599, 0.83895131, 0.83895131, 0.83895131, 0.83895131,
       0.83520599, 0.83146067, 0.82771536, 0.8164794 , 0.82022472,
       0.83520599, 0.83146067, 0.82771536, 0.8164794 , 0.82022472,
       0.83520599, 0.83146067, 0.82771536, 0.8164794 , 0.82022472,
       0.83520599, 0.83146067, 0.82771536, 0.8164794 , 0.82022472,
       0.83520599, 0.83146067, 0.82771536, 0.8164794 , 0.82022472]),
'split2_test_score': array([0.79026217, 0.78277154, 0.79400749, 0.79026217,
0.78651685,
       0.7752809 , 0.79400749, 0.80524345, 0.80524345, 0.80149813,
       0.80898876, 0.8164794 , 0.80898876, 0.80149813, 0.80149813,
       0.80149813, 0.80524345, 0.79400749, 0.80149813, 0.80149813,
       0.79026217, 0.79775281, 0.80149813, 0.80524345, 0.80524345,
       0.80898876, 0.80898876, 0.79775281, 0.80898876, 0.80524345,
       0.80898876, 0.80898876, 0.79775281, 0.80898876, 0.80524345,
       0.80898876, 0.80898876, 0.79775281, 0.80898876, 0.80524345,
       0.79775281, 0.80898876, 0.80524345, 0.80524345, 0.80524345,
       0.80898876, 0.81273408, 0.81273408, 0.80524345, 0.80524345,
       0.79775281, 0.80149813, 0.80524345, 0.80149813, 0.79400749,
       0.79775281, 0.80149813, 0.80524345, 0.80149813, 0.79400749,
       0.79775281, 0.80149813, 0.80524345, 0.80149813, 0.79400749,
       0.79775281, 0.80149813, 0.80524345, 0.80149813, 0.79400749,
       0.79775281, 0.80149813, 0.80524345, 0.80149813, 0.79400749,
       0.79400749, 0.79400749, 0.80524345, 0.79026217, 0.79026217,
       0.80524345, 0.80524345, 0.81273408, 0.80524345, 0.80149813,
       0.82397004, 0.82397004, 0.80524345, 0.80149813, 0.79775281,
       0.78277154, 0.79775281, 0.80524345, 0.80524345, 0.79775281,
       0.79026217, 0.79775281, 0.80898876, 0.80898876, 0.80524345,
       0.79775281, 0.80149813, 0.79775281, 0.80524345, 0.80524345,
       0.79775281, 0.80149813, 0.79775281, 0.80524345, 0.80524345,
       0.79775281, 0.80149813, 0.79775281, 0.80524345, 0.80524345,
       0.79775281, 0.80149813, 0.80898876, 0.81273408, 0.80524345,
       0.80524345, 0.80524345, 0.80149813, 0.80524345, 0.80524345,
       0.80149813, 0.79775281, 0.80149813, 0.79775281, 0.79400749,
       0.80149813, 0.79775281, 0.80149813, 0.79775281, 0.79400749,
       0.80149813, 0.79775281, 0.80149813, 0.79775281, 0.79400749,
       0.80149813, 0.79775281, 0.80149813, 0.79775281, 0.79400749,
       0.80149813, 0.79775281, 0.80149813, 0.79775281, 0.79400749]),
'mean_test_score': array([0.81523096, 0.81273408, 0.8164794 , 0.81772784,
0.81523096,
       0.8102372 , 0.8164794 , 0.8227216 , 0.82147316, 0.82147316,
       0.8289638 , 0.8289638 , 0.82521848, 0.82397004, 0.82397004,
       0.8227216 , 0.82397004, 0.82147316, 0.82521848, 0.82521848,
       0.82022472, 0.82147316, 0.82397004, 0.8227216 , 0.8227216 ,
       0.82521848, 0.8227216 , 0.8227216 , 0.82521848, 0.82147316,
       0.82521848, 0.8227216 , 0.8227216 , 0.82521848, 0.82147316,
       0.82521848, 0.8227216 , 0.8227216 , 0.82521848, 0.82147316,
       0.82022472, 0.82147316, 0.82521848, 0.8227216 , 0.8227216 ,
       0.82646692, 0.82646692, 0.82521848, 0.82147316, 0.8227216 ,
       0.81897628, 0.8227216 , 0.81772784, 0.81772784, 0.81398252,
       0.81897628, 0.8227216 , 0.81772784, 0.81772784, 0.81398252,
       0.81897628, 0.8227216 , 0.81772784, 0.81772784, 0.81398252,
       0.81897628, 0.8227216 , 0.81772784, 0.81772784, 0.81398252,
       0.81897628, 0.8227216 , 0.81772784, 0.81772784, 0.81398252,
       0.81772784, 0.82022472, 0.82646692, 0.81897628, 0.82147316,
       0.82022472, 0.82397004, 0.82771536, 0.82646692, 0.82521848,
       0.83146067, 0.83270911, 0.82521848, 0.82397004, 0.8227216 ,
       0.81523096, 0.82147316, 0.8289638 , 0.82521848, 0.82397004,
       0.81148564, 0.81772784, 0.82521848, 0.8227216 , 0.82147316,
       0.82771536, 0.82521848, 0.82022472, 0.8227216 , 0.8227216 ,
       0.82771536, 0.82521848, 0.82022472, 0.8227216 , 0.8227216 ,
       0.82771536, 0.82521848, 0.82022472, 0.8227216 , 0.8227216 ,
       0.82022472, 0.82521848, 0.82521848, 0.82521848, 0.82147316,
       0.82022472, 0.82147316, 0.82022472, 0.8227216 , 0.82147316,
       0.81897628, 0.8164794 , 0.81897628, 0.8164794 , 0.81273408,
       0.81897628, 0.8164794 , 0.81897628, 0.8164794 , 0.81273408,
       0.81897628, 0.8164794 , 0.81897628, 0.8164794 , 0.81273408,
       0.81897628, 0.8164794 , 0.81897628, 0.8164794 , 0.81273408,
       0.81897628, 0.8164794 , 0.81897628, 0.8164794 , 0.81273408]),
'std_test_score': array([0.01989685, 0.02205184, 0.02005291, 0.02147896,
0.02169556,
       0.02906479, 0.02005291, 0.01738875, 0.01569264, 0.01868493,
       0.02564624, 0.0231551 , 0.01791848, 0.02005291, 0.02005291,
       0.03001452, 0.02388405, 0.02295228, 0.01989685, 0.01989685,
       0.02140628, 0.01843299, 0.02005291, 0.01738875, 0.01738875,
       0.01569264, 0.01684237, 0.01989685, 0.01378946, 0.01791848,
       0.01569264, 0.01684237, 0.01989685, 0.01378946, 0.01791848,
       0.01569264, 0.01684237, 0.01989685, 0.01378946, 0.01791848,
       0.02427244, 0.01508495, 0.01868493, 0.0153918 , 0.01738875,
       0.0153918 , 0.01684237, 0.01273163, 0.01378946, 0.0153918 ,
       0.01508495, 0.01569264, 0.00934247, 0.01157755, 0.01412448,
       0.01508495, 0.01569264, 0.00934247, 0.01157755, 0.01412448,
       0.01508495, 0.01569264, 0.00934247, 0.01157755, 0.01412448,
       0.01508495, 0.01569264, 0.00934247, 0.01157755, 0.01412448,
       0.01508495, 0.01569264, 0.00934247, 0.01157755, 0.01412448,
       0.01843299, 0.02308769, 0.01569264, 0.02169556, 0.02335616,
       0.01860133, 0.01702644, 0.0140137 , 0.02036143, 0.02169556,
       0.02205184, 0.01791848, 0.01868493, 0.02005291, 0.02546327,
       0.03138528, 0.02375318, 0.02375318, 0.0231551 , 0.02503113,
       0.02490629, 0.02564624, 0.01791848, 0.01684237, 0.01791848,
       0.0261279 , 0.02169556, 0.02005291, 0.01738875, 0.01738875,
       0.0261279 , 0.02169556, 0.02005291, 0.01738875, 0.01738875,
       0.0261279 , 0.02169556, 0.02005291, 0.01738875, 0.01738875,
       0.01834824, 0.01843299, 0.01378946, 0.01273163, 0.01569264,
       0.01223216, 0.01378946, 0.0152902 , 0.01378946, 0.01378946,
       0.01378946, 0.0140137 , 0.01235892, 0.0152902 , 0.01332969,
       0.01378946, 0.0140137 , 0.01235892, 0.0152902 , 0.01332969,
       0.01378946, 0.0140137 , 0.01235892, 0.0152902 , 0.01332969,
       0.01378946, 0.0140137 , 0.01235892, 0.0152902 , 0.01332969,
       0.01378946, 0.0140137 , 0.01235892, 0.0152902 , 0.01332969]),
'rank_test_score': array([135, 143, 123, 110, 135, 150, 123,  42,  75,  69,   3,
3,  19,
        35,  35,  42,  35,  69,  19,  19,  84,  75,  35,  42,  42,  19,
        42,  42,  19,  75,  19,  42,  42,  19,  75,  19,  42,  42,  19,
        75,  84,  69,  14,  42,  42,  10,  11,  14,  69,  42,  94,  42,
       110, 110, 138,  94,  42, 110, 110, 138,  94,  42, 110, 110, 138,
        94,  42, 110, 110, 138,  94,  42, 110, 110, 138, 110,  84,  11,
        94,  75,  84,  35,   6,  11,  19,   2,   1,  14,  35,  42, 135,
        75,   3,  14,  35, 149, 110,  19,  42,  75,   6,  19,  84,  42,
        42,   6,  19,  84,  42,  42,   6,  19,  84,  42,  42,  84,  19,
        19,  14,  75,  84,  69,  84,  42,  69,  94, 123,  94, 123, 143,
        94, 123,  94, 123, 143,  94, 123,  94, 123, 143,  94, 123,  94,
       123, 143,  94, 123,  94, 123, 143])}
```

[39]:
```
gs = gs.fit(x_test, y_test)
print(gs.best_score_)
```

```
0.8222222222222223
```

1 Predicting on test data

[52]:
```
test_set['Age'] = test_set['Age'].fillna(test_set['Age'].mean())
test_set['Fare'] = test_set['Fare'].fillna(test_set['Fare'].mean())
test_set['Embarked'] = test_set['Embarked'].fillna(test_set['Embarked'].
 ↪mode()[0])
test_set['Embarked'] = le.fit_transform(test_set['Embarked'])
test_set['Cabin_Letter'] = test_set['Cabin'].apply(lambda x: str(x)[0])
test_set['Cabin_Letter'] = le.fit_transform(test_set['Cabin_Letter'])
test_set['Sex'] = le.fit_transform(test_set['Sex'])
```

[53]:
```
test_set.head(25)
```

[53]:
```
    PassengerId  Pclass                                           Name  \
0           892       3                               Kelly, Mr. James
1           893       3               Wilkes, Mrs. James (Ellen Needs)
2           894       2                      Myles, Mr. Thomas Francis
3           895       3                               Wirz, Mr. Albert
4           896       3   Hirvonen, Mrs. Alexander (Helga E Lindqvist)
5           897       3                     Svensson, Mr. Johan Cervin
6           898       3                           Connolly, Miss. Kate
7           899       2                   Caldwell, Mr. Albert Francis
8           900       3      Abrahim, Mrs. Joseph (Sophie Halaut Easu)
9           901       3                        Davies, Mr. John Samuel
10          902       3                               Ilieff, Mr. Ylio
11          903       1                     Jones, Mr. Charles Cresson
12          904       1    Snyder, Mrs. John Pillsbury (Nelle Stevenson)
13          905       2                           Howard, Mr. Benjamin
14          906       1  Chaffee, Mrs. Herbert Fuller (Carrie Constance…
15          907       2    del Carlo, Mrs. Sebastiano (Argenia Genovesi)
```

```
16          908       2                                   Keane, Mr. Daniel
17          909       3                                   Assaf, Mr. Gerios
18          910       3                          Ilmakangas, Miss. Ida Livija
19          911       3               Assaf Khalil, Mrs. Mariana (Miriam")"
20          912       1                               Rothschild, Mr. Martin
21          913       3                              Olsen, Master. Artur Karl
22          914       1                Flegenheim, Mrs. Alfred (Antoinette)
23          915       1                          Williams, Mr. Richard Norris II
24          916       1    Ryerson, Mrs. Arthur Larned (Emily Maria Borie)

    Sex       Age  SibSp  Parch             Ticket      Fare            Cabin  \
0     1  34.50000      0      0             330911    7.8292              NaN
1     0  47.00000      1      0             363272    7.0000              NaN
2     1  62.00000      0      0             240276    9.6875              NaN
3     1  27.00000      0      0             315154    8.6625              NaN
4     0  22.00000      1      1            3101298   12.2875              NaN
5     1  14.00000      0      0               7538    9.2250              NaN
6     0  30.00000      0      0             330972    7.6292              NaN
7     1  26.00000      1      1             248738   29.0000              NaN
8     0  18.00000      0      0               2657    7.2292              NaN
9     1  21.00000      2      0          A/4 48871   24.1500              NaN
10    1  30.27259      0      0             349220    7.8958              NaN
11    1  46.00000      0      0                694   26.0000              NaN
12    0  23.00000      1      0              21228   82.2667              B45
13    1  63.00000      1      0              24065   26.0000              NaN
14    0  47.00000      1      0         W.E.P. 5734   61.1750              E31
15    0  24.00000      1      0      SC/PARIS 2167   27.7208              NaN
16    1  35.00000      0      0             233734   12.3500              NaN
17    1  21.00000      0      0               2692    7.2250              NaN
18    0  27.00000      1      0  STON/O2. 3101270    7.9250              NaN
19    0  45.00000      0      0               2696    7.2250              NaN
20    1  55.00000      1      0           PC 17603   59.4000              NaN
21    1   9.00000      0      1            C 17368    3.1708              NaN
22    0  30.27259      0      0           PC 17598   31.6833              NaN
23    1  21.00000      0      1           PC 17597   61.3792              NaN
24    0  48.00000      1      3           PC 17608  262.3750  B57 B59 B63 B66

    Embarked  Cabin_Letter
0          1             7
1          2             7
2          1             7
3          2             7
4          2             7
5          2             7
6          1             7
7          2             7
8          0             7
```

```
9           2             7
10          2             7
11          2             7
12          2             1
13          2             7
14          2             4
15          0             7
16          1             7
17          0             7
18          2             7
19          0             7
20          0             7
21          2             7
22          2             7
23          0             7
24          0             1
```

[54]:
```
test_set.isnull().sum()
```

[54]:
```
PassengerId       0
Pclass            0
Name              0
Sex               0
Age               0
SibSp             0
Parch             0
Ticket            0
Fare              0
Cabin           327
Embarked          0
Cabin_Letter      0
dtype: int64
```

[58]:
```
test_X = test_set[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch',␣
 ↪'Fare','Cabin_Letter', 'Embarked']]
# Make predictions on the test data using the trained Decision Tree classifier
test_predictions_dt = best_clf.predict(test_X)
# Make predictions on the test data using the trained Random Forest classifier
test_predictions_rf = gs.predict(test_X)
# Output predictions
output_dt = pnd.DataFrame({'PassengerId': test_set.PassengerId, 'Survived':
 ↪test_predictions_dt})
output_rf = pnd.DataFrame({'PassengerId': test_set.PassengerId, 'Survived':
 ↪test_predictions_rf})
# Save the predictions to a CSV file
output_dt.to_csv('predictions_decision_tree.csv', index=False)
output_rf.to_csv('predictions_random_forest.csv', index=False)
```

[60]:
```python
compareDtAndRf = accuracy_score(test_predictions_dt, test_predictions_rf)
print("Comparing results predicted by Decision Tree and Random Forest",␣
 ↪compareDtAndRf)
```

```
Comparing results predicted by Decision Tree and Random Forest
0.8444976076555024
```

moreover ipynb file and predicted data set ca be found on following github repo: https://github.com/Shourya0712/titanicDT