```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

df = pd.read_csv('/content/titanic_cleaned.csv')

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"Survived\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Pclass\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 3,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          3,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Sex\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 2,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          2,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Age\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 13.019696550973201,\n        \"min\":
0.42,\n        \"max\": 80.0,\n        \"num_unique_values\": 88,\n
\"samples\": [\n          0.75,\n          22.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
df.tail()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"Survived\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Pclass\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 3,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          2,\n          1\n        ],\n        \"semantic_type\":
```

\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Sex\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 1,\n
\"max\": 2,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   2,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Age\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4.722287581247038,\n   \"min\":
19.0,\n   \"max\": 32.0,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   19.0,\n   32.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    int64
 3   Age       891 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 28.0 KB
```

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"Survived\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 314.8713661874558,\n
\"min\": 0.0,\n   \"max\": 891.0,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   0.3838383838383838,\n
1.0,\n   0.4865924542648575\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
314.2523437079694,\n   \"min\": 0.836071240977049,\n
\"max\": 891.0,\n   \"num_unique_values\": 6,\n
\"samples\": [\n   891.0,\n   2.308641975308642,\n
3.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Sex\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 314.57050384040605,\n   \"min\": 0.4779900708960981,\n
\"max\": 891.0,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   1.3524130190796857,\n   2.0,\n
0.4779900708960981\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n

\"std\": 305.4002666458502,\n   \"min\": 0.42,\n   \"max\": 891.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n 29.36158249158249,\n   28.0,\n   891.0\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.shape
```

```
(891, 4)
```

```
from sklearn.model_selection import train_test_split
X = df.drop('Survived',axis=1)
y = df['Survived']
```

```
X
```

{"summary":"{\n \"name\": \"X\",\n \"rows\": 891,\n \"fields\": [\n {\n   \"column\": \"Pclass\",\n   \"properties\": {\n \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   3,\n   1,\n   2\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n   \"properties\": {\n \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n \"max\": 2,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   2,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 13.019696550973201,\n   \"min\": 0.42,\n   \"max\": 80.0,\n   \"num_unique_values\": 88,\n \"samples\": [\n   0.75,\n   22.0\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"X"}

```
y
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=42)
```

```
X_train.shape
```

```
(712, 3)
```

```
X_test.shape
```

```
(179, 3)
```

```
y_train.shape
```

```
(712,)
```

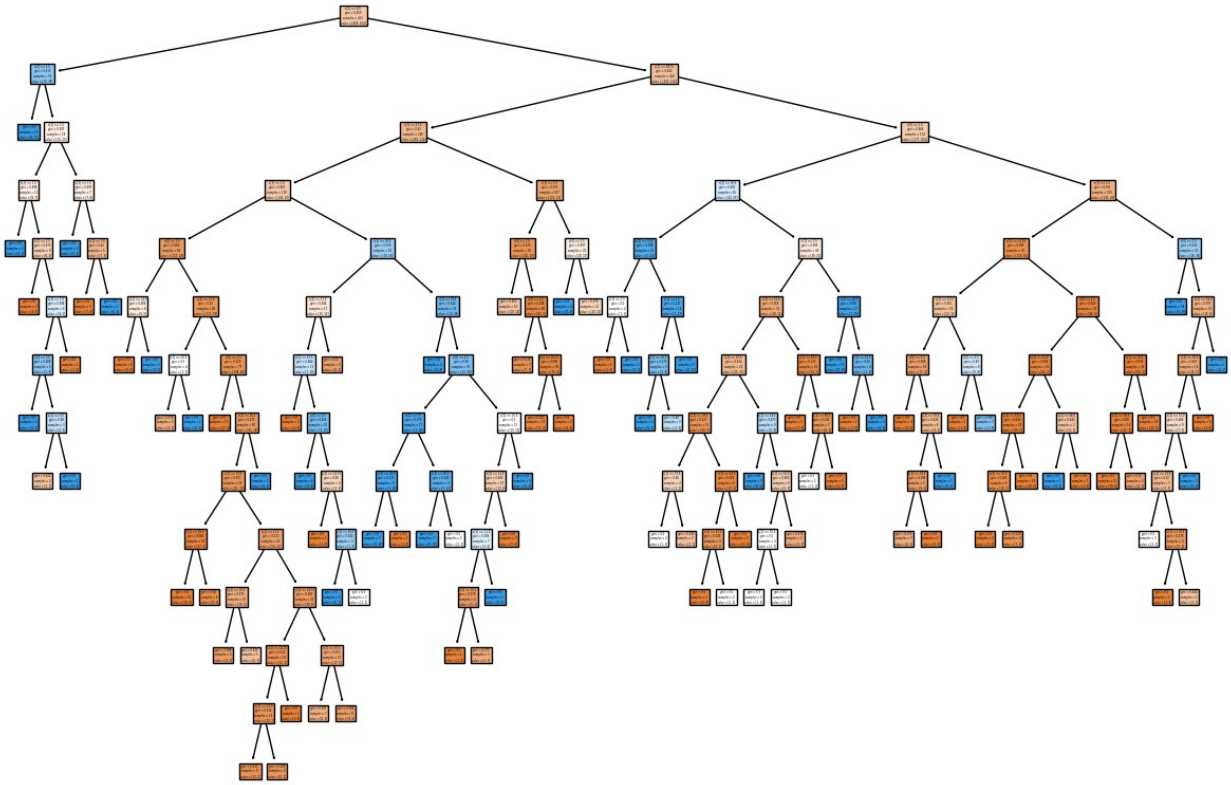
```
y_test.shape
```

```
(179,)
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
plt.figure(figsize=(15, 10))
plot_tree(rf.estimators_[0], filled=True)
plt.show()
```



```
y_pred = rf.predict(X_test)
y_pred
```

```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
0,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0,
       0, 1, 1])
```

```
print(classification_report(y_test,y_pred))
```

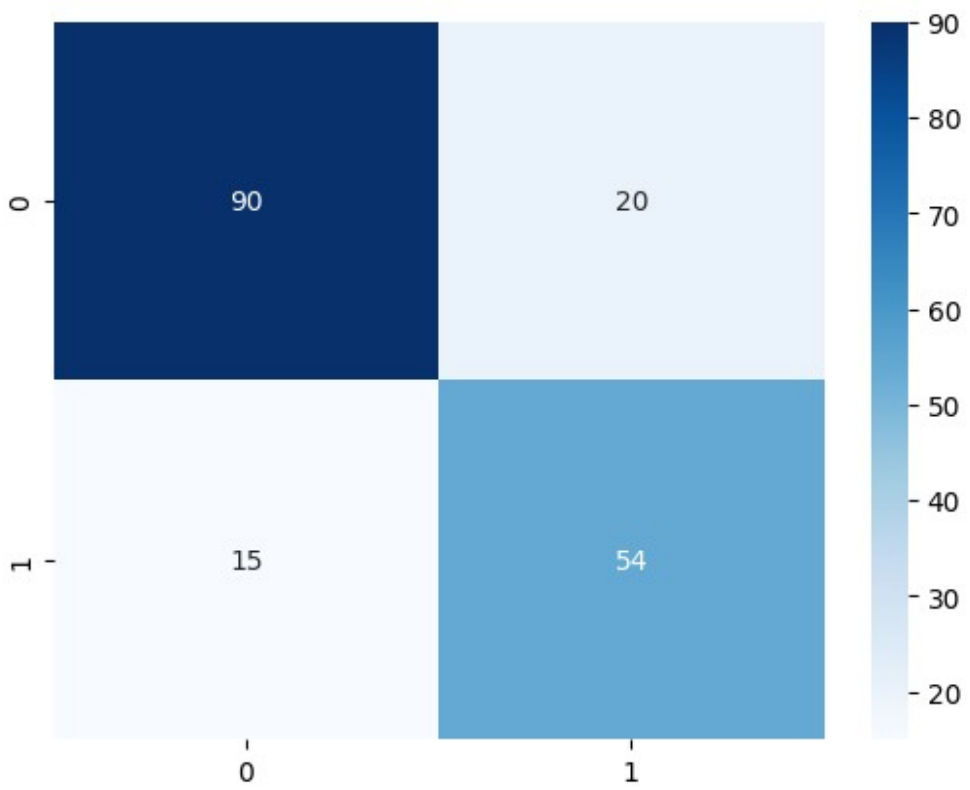
```
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       105
           1       0.78      0.73      0.76        74

    accuracy                           0.80       179
   macro avg       0.80      0.79      0.80       179
weighted avg       0.80      0.80      0.80       179
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_pred,y_test)
sns.heatmap(cm,annot=True,cmap='Blues')
```

```
<Axes: >
```




```
parameter = {
'n_estimators': [50, 100, 200],
'max_depth': [10,20,30],
'max_features': ['int','auto', 'sqrt', 'log2'],
'bootstrap': [True, False]   #randomness of tree
 }

from sklearn.model_selection import GridSearchCV
rfmodel = RandomForestClassifier(max_depth=2)
grid_search =
GridSearchCV(estimator=rfmodel,param_grid=parameter,cv=5,scoring='accu
racy')
grid_search.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/
_validation.py:425: FitFailedWarning:
180 fits failed out of a total of 360.
The score on these train-test partitions for these parameters will be
set to nan.
If these failures are not expected, you can try to debug them by
setting error_score='raise'.

Below are more details about the failures:
----------------------------------------------------------------------
----------
```

```
90 fits failed with the following error:
Traceback (most recent call last):
  File
"/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_vali
dation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line
1145, in wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line
638, in _validate_params
    validate_parameter_constraints(
  File
"/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validati
on.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The
'max_features' parameter of RandomForestClassifier must be an int in
the range [1, inf), a float in the range (0.0, 1.0], a str among
{'sqrt', 'log2'} or None. Got 'int' instead.

--------------------------------------------------------------------
----------
90 fits failed with the following error:
Traceback (most recent call last):
  File
"/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_vali
dation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line
1145, in wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line
638, in _validate_params
    validate_parameter_constraints(
  File
"/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validati
on.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The
'max_features' parameter of RandomForestClassifier must be an int in
the range [1, inf), a float in the range (0.0, 1.0], a str among
{'sqrt', 'log2'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_searc
h.py:979: UserWarning: One or more of the test scores are non-finite:
[       nan        nan        nan        nan        nan        nan
 0.80331922 0.80473752 0.80052201 0.80333891 0.80053186 0.80053186
```

```
        nan         nan         nan         nan         nan         nan
 0.7893135  0.79069241 0.79769526 0.79212056 0.79911356 0.79771496
        nan         nan         nan         nan         nan         nan
 0.79068256 0.80193046 0.80053186 0.79772481 0.79771496 0.79772481
        nan         nan         nan         nan         nan         nan
 0.80475721 0.80475721 0.80615582 0.80334876 0.80333891 0.80756427
        nan         nan         nan         nan         nan         nan
 0.79771496 0.79771496 0.79911356 0.79911356 0.79771496 0.79911356
        nan         nan         nan         nan         nan         nan
 0.79771496 0.79911356 0.79911356 0.79911356 0.79771496 0.79911356]
  warnings.warn(
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(max_depth=2),
             param_grid={'bootstrap': [True, False], 'max_depth': [10,
20, 30],
                         'max_features': ['int', 'auto', 'sqrt',
'log2'],
                         'n_estimators': [50, 100, 200]},
             scoring='accuracy')
```

```
grid_search.best_params_
```

```
{'bootstrap': False,
 'max_depth': 10,
 'max_features': 'log2',
 'n_estimators': 200}
```

```
y_pred2 = grid_search.predict(X_test)
```

```
print(classification_report(y_test,y_pred2))
```

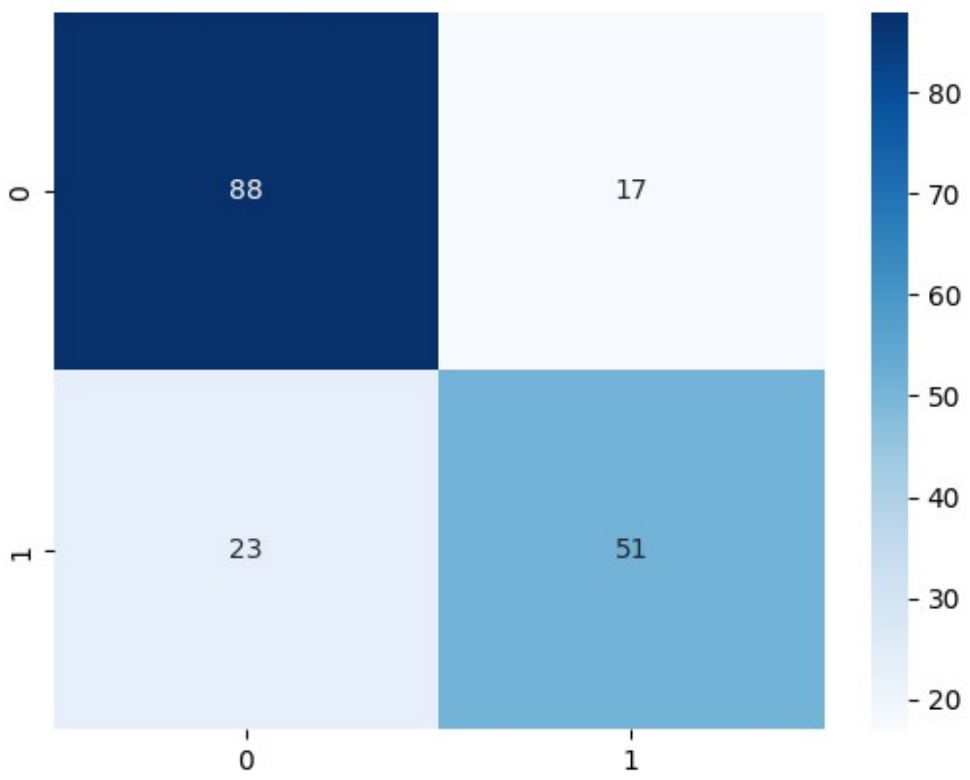
```
              precision    recall  f1-score   support

           0       0.79      0.84      0.81       105
           1       0.75      0.69      0.72        74

    accuracy                           0.78       179
   macro avg       0.77      0.76      0.77       179
weighted avg       0.78      0.78      0.77       179
```

```
cm = confusion_matrix(y_test,y_pred2)
sns.heatmap(cm,annot=True,cmap='Blues')
```

```
<Axes: >
```


88
17
23
51
0
1
0
1
80
70
60
50
40
30
20