In [12]:
```
import pandas as pd
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.model_selection import GridSearchCV, cross_validate, train_test_s
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
pd.set_option('display.max_columns',None)
pd.set_option('display.width',500)
import warnings
import joblib
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export_text
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
df=sns.load_dataset('titanic')
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[12], line 5
      3 from sklearn.model_selection import GridSearchCV, cross_validate, tra
in_test_split, validation_curve
      4 from sklearn.neighbors import KNeighborsClassifier
----> 5 from sklearn.tree import RandomTreeClassifier
      6 from sklearn.preprocessing import StandardScaler
      7 pd.set_option('display.max_columns',None)

ImportError: cannot import name 'RandomTreeClassifier' from 'sklearn.tree'
(C:\ProgramData\anaconda3\Lib\site-packages\sklearn\tree\__init__.py)
```

In [8]:
```
p=['pclass','age','sibsp','fare']
X = df[p]
y = df['survived']
updated_df = X
updated_df['age']=updated_df['age'].fillna(updated_df['age'].mean())
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_12340\2227514206.py:5: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  updated_df['age']=updated_df['age'].fillna(updated_df['age'].mean())
```

In [14]:
```
cart_model = DecisionTreeClassifier().fit(X,y)
```

In [15]:
```
y_pred = cart_model.predict(X)
```

In [16]:
```
print(classification_report(y,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       549
           1       0.98      0.91      0.94       342

    accuracy                           0.96       891
   macro avg       0.96      0.95      0.95       891
weighted avg       0.96      0.96      0.96       891
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30)
```

In [19]:
```
cart_model = DecisionTreeClassifier().fit(X_train, y_train)
```

In [20]:
```
y_pred = cart_model.predict(X_train)
```

In [21]:
```
print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       388
           1       0.98      0.91      0.95       235

    accuracy                           0.96       623
   macro avg       0.97      0.95      0.96       623
weighted avg       0.96      0.96      0.96       623
```

In [22]:
```
cv_results = cross_validate(cart_model, X,y, cv=5, scoring=["accuracy","f1","r
```

In [23]:
```
cv_results
```

Out[23]:
```
{'fit_time': array([0.00635815, 0.00713468, 0.00803518, 0.00931168, 0.
]),
 'score_time': array([0.01810265, 0.        , 0.00801206, 0.        , 0.
]),
 'test_accuracy': array([0.65921788, 0.61797753, 0.66853933, 0.65168539, 0.66
853933]),
 'test_f1': array([0.55474453, 0.49253731, 0.528     , 0.55072464, 0.5755395
7]),
 'test_roc_auc': array([0.66357049, 0.60066845, 0.65173797, 0.6493984 , 0.651
70855])}
```

In [24]:
```
cart_model.get_params()
```

Out[24]:
```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

In [25]:
```
cart_params = {'max_depth': range(1,11),
               'min_samples_split': range(2,20)}
```

In [26]:
```
cart_best_grid = GridSearchCV(cart_model, cart_params, cv=5, n_jobs=-1, verbos
```

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
```

In [27]:
```
cart_best_grid.best_params_
```

Out[27]:
```
{'max_depth': 4, 'min_samples_split': 14}
```

In [28]:
```
cart_best_grid.best_score_
```

Out[28]:
```
0.7105015378821167
```

In [ ]:
```
##5.Final Model
```

In [29]:
```
cart_final = DecisionTreeClassifier(**cart_best_grid.best_params_).fit(X,y)
```

In [30]:
```
cv_results = cross_validate(cart_final, X,y, cv=5, scoring=["accuracy","f1","r
```

In [31]:
```
cv_results['test_accuracy'].mean()
```

Out[31]:
```
0.7105015378821167
```

In [32]:
```
cv_results['test_f1'].mean()
```

Out[32]:
```
0.5444387722267016
```

In [33]:
```
cv_results['test_roc_auc'].mean()
```

Out[33]:
```
0.7262925288692552
```