In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from xgboost import XGBClassifier
from sklearn.metrics import classification_report as cr

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("C:\\Users\\HP\\Downloads\\titanic_train.csv")
df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [3]:
```
df.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
```

In [4]:
```
avgAge = df.Age.mean()
df.Age = df.Age.fillna(value=avgAge)
```

In [5]:
```
df.isna().sum()
```

Out[5]:
```
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
```

In [6]:
```
df.drop('Cabin',axis=1, inplace=True)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

In [8]:
```
df.dropna(inplace=True)
```

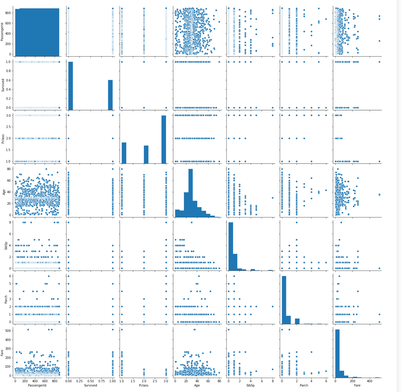
In [9]:
```
df.head()
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

In [10]:
```
sns.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x28aa36706a0>
```



In [11]:
```
df.drop(['PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)
```

In [12]:
```
df.head()
```

Out[12]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [13]:
```
x = df.iloc[:, 1:]
y = df.iloc[:, 0]
```

In [14]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), ['Sex', 'Embarked'])], r
emainder='passthrough')

x = np.array(ct.fit_transform(x))
```

In [15]:
```
x
```

Out[15]:
```
array([[ 0.    ,  1.    ,  0.    , ...,  1.    ,  0.    ,  7.25  ],
       [ 1.    ,  0.    ,  1.    , ...,  1.    ,  0.    , 71.2833],
       [ 1.    ,  0.    ,  0.    , ...,  0.    ,  0.    ,  7.925 ],
       ...,
       [ 1.    ,  0.    ,  0.    , ...,  1.    ,  2.    , 23.45  ],
       [ 0.    ,  1.    ,  0.    , ...,  0.    ,  0.    , 30.    ],
       [ 0.    ,  1.    ,  0.    , ...,  0.    ,  0.    ,  7.75  ]])
```

In [16]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.25, random_state=1)
```

In [17]:
```
xgb=XGBClassifier(eval_metric='error')
```

In [18]:
```
xgb.fit(xtrain,ytrain)
```

Out[18]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='error',
              gamma=0, gpu_id=-1, importance_type='gain',
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=1, subsample=1, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

In [19]:
```
ypred=xgb.predict(xtest)
```

In [20]:
```
print( f"Classification Report -:\n {cr(ytest, ypred)}" )
```

```
Classification Report -:
               precision    recall  f1-score   support

           0       0.87      0.85      0.86       138
           1       0.76      0.79      0.77        85

    accuracy                           0.83       223
   macro avg       0.81      0.82      0.82       223
weighted avg       0.83      0.83      0.83       223
```

In [21]:
```
cvs = cross_val_score(xgb, xtrain, ytrain, cv=10, scoring='accuracy')
print(f'Accuracy: {cvs.mean()*100} \n Standard Deviation: {cvs.std()*100}\n\n')
```

```
Accuracy: 79.12256897331524
 Standard Deviation: 3.7485561617116714
```

In [22]:
```
from sklearn.model_selection import GridSearchCV
params={
    'max_depth':range(3,10,1),
    'min_child_weight':range(1,6,1),
    'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05],
    'n_estimators':[100, 200, 300, 400, 500]
}
searchi = GridSearchCV(estimator = xgb,param_grid = params, cv=5,n_jobs=5,verbose=True)
```

In [23]:
```
best_model = searchi.fit(x,y)
```

```
Fitting 5 folds for each of 875 candidates, totalling 4375 fits
```

```
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done  40 tasks      | elapsed:    7.1s
[Parallel(n_jobs=5)]: Done 190 tasks      | elapsed:   23.2s
[Parallel(n_jobs=5)]: Done 440 tasks      | elapsed:   49.9s
[Parallel(n_jobs=5)]: Done 790 tasks      | elapsed:  1.5min
[Parallel(n_jobs=5)]: Done 1240 tasks     | elapsed:  2.5min
[Parallel(n_jobs=5)]: Done 1790 tasks     | elapsed:  3.9min
[Parallel(n_jobs=5)]: Done 2440 tasks     | elapsed:  5.7min
[Parallel(n_jobs=5)]: Done 3190 tasks     | elapsed:  8.0min
[Parallel(n_jobs=5)]: Done 4040 tasks     | elapsed: 11.1min
[Parallel(n_jobs=5)]: Done 4375 out of 4375 | elapsed: 12.2min finished
```

In [24]:
```
ypred = best_model.predict(xtest)
```

In [25]:
```
print( f"Classification Report -:\n {cr(ytest, ypred)}" )
```

```
Classification Report -:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97       138
           1       0.95      0.95      0.95        85

    accuracy                           0.96       223
   macro avg       0.96      0.96      0.96       223
weighted avg       0.96      0.96      0.96       223
```

In [ ]:
```

```