assignment3

September 20, 2023

[ ]:
```
#Avantika Mishra
#21BCE2102
```

1 1.import the necessary libraries

[75]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2 2.import the dataset

[76]:
```
dataset=pd.read_csv("tested.csv")
```

[77]:
```
dataset
```

[77]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
0	892	0	3	Kelly, Mr. James	male	34.5	0	0
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1
..	…	…	…					
413	1305	0	3					
414	1306	1	1					
415	1307	0	3					
416	1308	0	3					
417	1309	0	3					

```
..                                          ...    ...  ...   ...   ...
413                            Spector, Mr. Woolf    male  NaN     0     0
414                  Oliva y Ocana, Dona. Fermina  female 39.0     0     0
415                  Saether, Mr. Simon Sivertsen    male 38.5     0     0
416                           Ware, Mr. Frederick    male  NaN     0     0
417                      Peter, Master. Michael J    male  NaN     1     1

                Ticket      Fare Cabin Embarked
0               330911    7.8292   NaN        Q
1               363272    7.0000   NaN        S
2               240276    9.6875   NaN        Q
3               315154    8.6625   NaN        S
4              3101298   12.2875   NaN        S
..                 ...       ...   ...      ...
413         A.5. 3236    8.0500   NaN        S
414          PC 17758  108.9000  C105        C
415  SOTON/O.Q. 3101262    7.2500   NaN        S
416              359309    8.0500   NaN        S
417                2668   22.3583   NaN        C

[418 rows x 12 columns]
```

[78]:
```
dataset.head()
```

[78]:
```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                          Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

[79]:
```
dataset.tail()
```

[79]:
```
     PassengerId  Survived  Pclass                          Name     Sex  \
413         1305         0       3              Spector, Mr. Woolf    male
414         1306         1       1    Oliva y Ocana, Dona. Fermina  female
415         1307         0       3    Saether, Mr. Simon Sivertsen    male
416         1308         0       3             Ware, Mr. Frederick    male
417         1309         0       3        Peter, Master. Michael J    male

      Age  SibSp  Parch              Ticket      Fare Cabin Embarked
413   NaN      0      0           A.5. 3236    8.0500   NaN        S
414  39.0      0      0            PC 17758  108.9000  C105        C
415  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416   NaN      0      0              359309    8.0500   NaN        S
417   NaN      1      1                2668   22.3583   NaN        C
```

[80]:
```
dataset.shape
```

[80]:
```
(418, 12)
```

[81]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

[82]:
```
dataset.describe()
```

[82]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   418.000000  418.000000  418.000000  332.000000  418.000000
mean   1100.500000    0.363636    2.265550   30.272590    0.447368
std     120.810458    0.481622    0.841838   14.181209    0.896760
min     892.000000    0.000000    1.000000    0.170000    0.000000
```

```
25%     996.250000    0.000000    1.000000   21.000000    0.000000
50%    1100.500000    0.000000    3.000000   27.000000    0.000000
75%    1204.750000    1.000000    3.000000   39.000000    1.000000
max    1309.000000    1.000000    3.000000   76.000000    8.000000

            Parch        Fare
count  418.000000  417.000000
mean     0.392344   35.627188
std      0.981429   55.907576
min      0.000000    0.000000
25%      0.000000    7.895800
50%      0.000000   14.454200
75%      0.000000   31.500000
max      9.000000  512.329200
```

[83]:
```
corr=dataset.corr()
corr
```

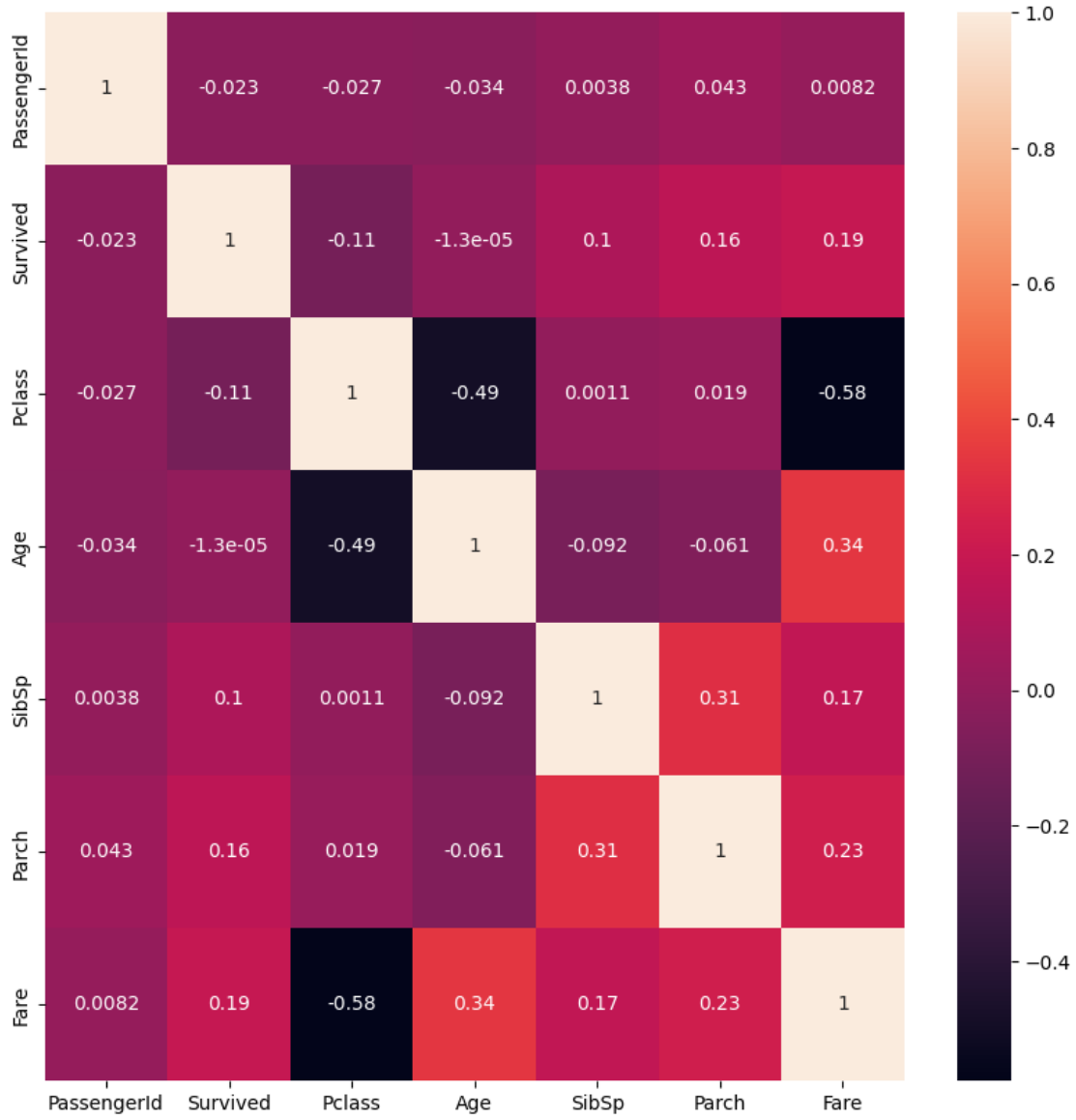
```
<ipython-input-83-f22ca9e9dc13>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr=dataset.corr()
```

[83]:
```
             PassengerId  Survived    Pclass       Age     SibSp     Parch  \
PassengerId     1.000000 -0.023245 -0.026751 -0.034102  0.003818  0.043080
Survived       -0.023245  1.000000 -0.108615 -0.000013  0.099943  0.159120
Pclass         -0.026751 -0.108615  1.000000 -0.492143  0.001087  0.018721
Age            -0.034102 -0.000013 -0.492143  1.000000 -0.091587 -0.061249
SibSp           0.003818  0.099943  0.001087 -0.091587  1.000000  0.306895
Parch           0.043080  0.159120  0.018721 -0.061249  0.306895  1.000000
Fare            0.008211  0.191514 -0.577147  0.337932  0.171539  0.230046

                 Fare
PassengerId  0.008211
Survived     0.191514
Pclass      -0.577147
Age          0.337932
SibSp        0.171539
Parch        0.230046
Fare         1.000000
```

[84]:
```
plt.subplots(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

[84]:
```
<Axes: >
```



[85]:
```
dataset.Survived.value_counts()
```

[85]:
```
0    266
1    152
Name: Survived, dtype: int64
```

[86]:
```
dataset.Sex.value_counts()
```

[86]:
```
male      266
female    152
Name: Sex, dtype: int64
```

[87]:
```
dataset.Pclass.value_counts()
```

[87]:
```
3    218
1    107
2     93
Name: Pclass, dtype: int64
```

3 3.Handling null values

[88]:
```
dataset.isnull().any()
```

[88]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare            True
Cabin           True
Embarked       False
dtype: bool
```

[89]:
```
dataset.isnull().sum()
```

[89]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

[90]:
```
dataset ["Fare"].fillna(dataset ["Fare"] .mean (), inplace=True)
```

[91]:
```
dataset ["Age"].fillna(dataset ["Age"] .mean (), inplace=True)
```

[92]:
```
dataset.isnull().any()
```

[92]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age            False
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked       False
dtype: bool
```

[93]:
```
#Since 80% of the values of "Cabin" are null, (327/418)...we will drop the column
dataset.drop(["Cabin"],axis=1)
```

[93]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	Kelly, Mr. James	male	34.50000	0	0	330911	7.8292	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.00000	1	0	363272	7.0000	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.00000	0				
3	895	0	3	Wirz, Mr. Albert	male	27.00000	0				
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.00000	1				
..	…	…	…	…	…	…	…				
413	1305	0	3	Spector, Mr. Woolf	male	30.27259	0				
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.00000	0				
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.50000	0				
416	1308	0	3	Ware, Mr. Frederick	male	30.27259	0				
417	1309	0	3	Peter, Master. Michael J	male	30.27259	1				

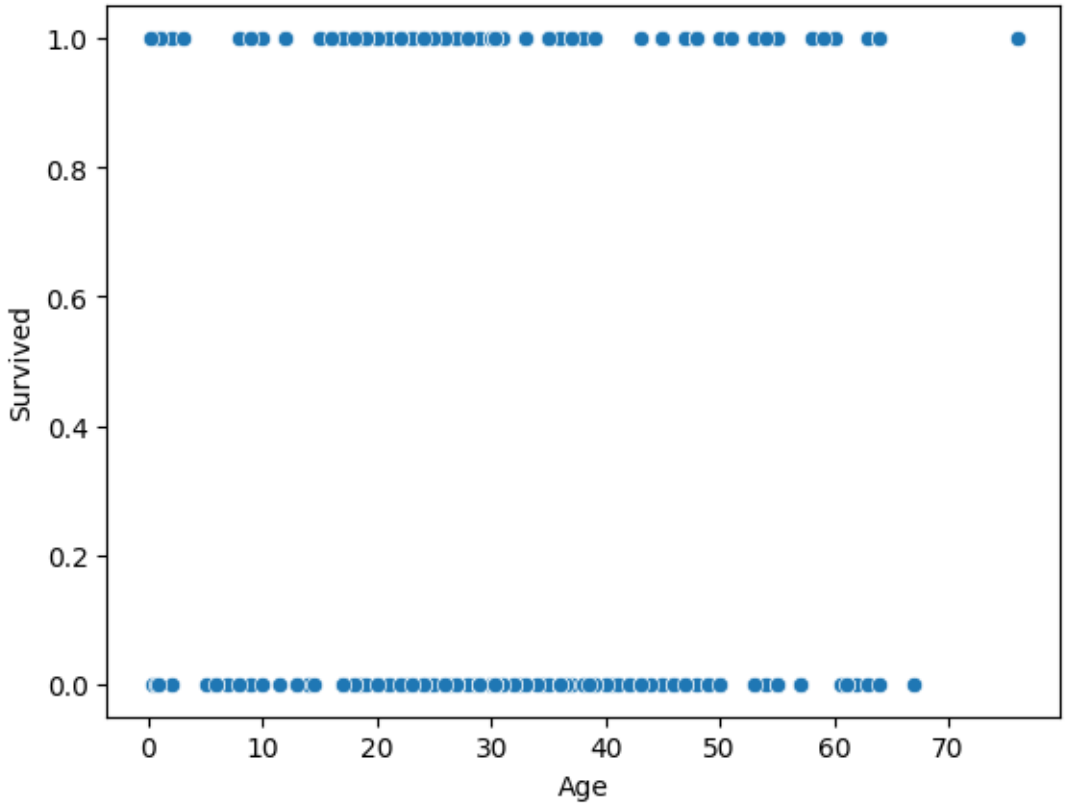
```
2         0              240276    9.6875        Q
3         0              315154    8.6625        S
4         1             3101298   12.2875        S
..      …                   …       …       …
413       0           A.5. 3236    8.0500        S
414       0            PC 17758  108.9000        C
415       0  SOTON/O.Q. 3101262    7.2500        S
416       0              359309    8.0500        S
417       1                2668   22.3583        C

[418 rows x 11 columns]
```

4 4.Data Visualisation

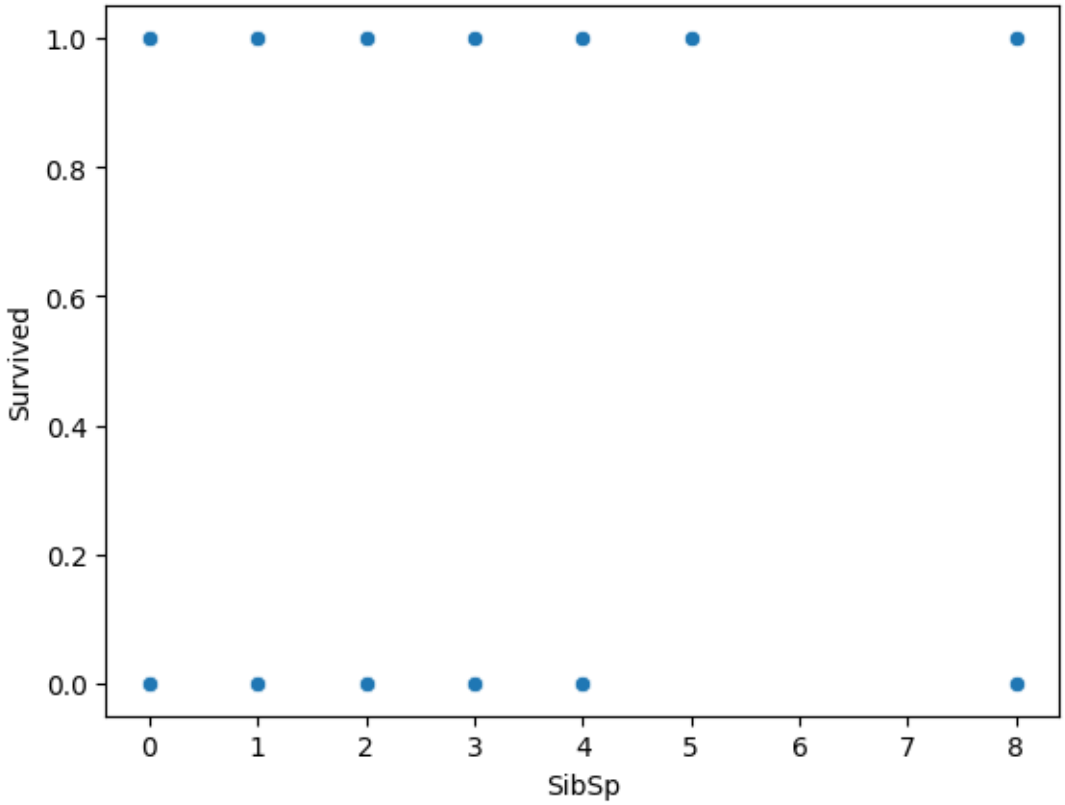
[94]:
```
sns.scatterplot(x="Age"         ,y="Survived",data=dataset)
```

[94]:
```
<Axes: xlabel='Age', ylabel='Survived'>
```



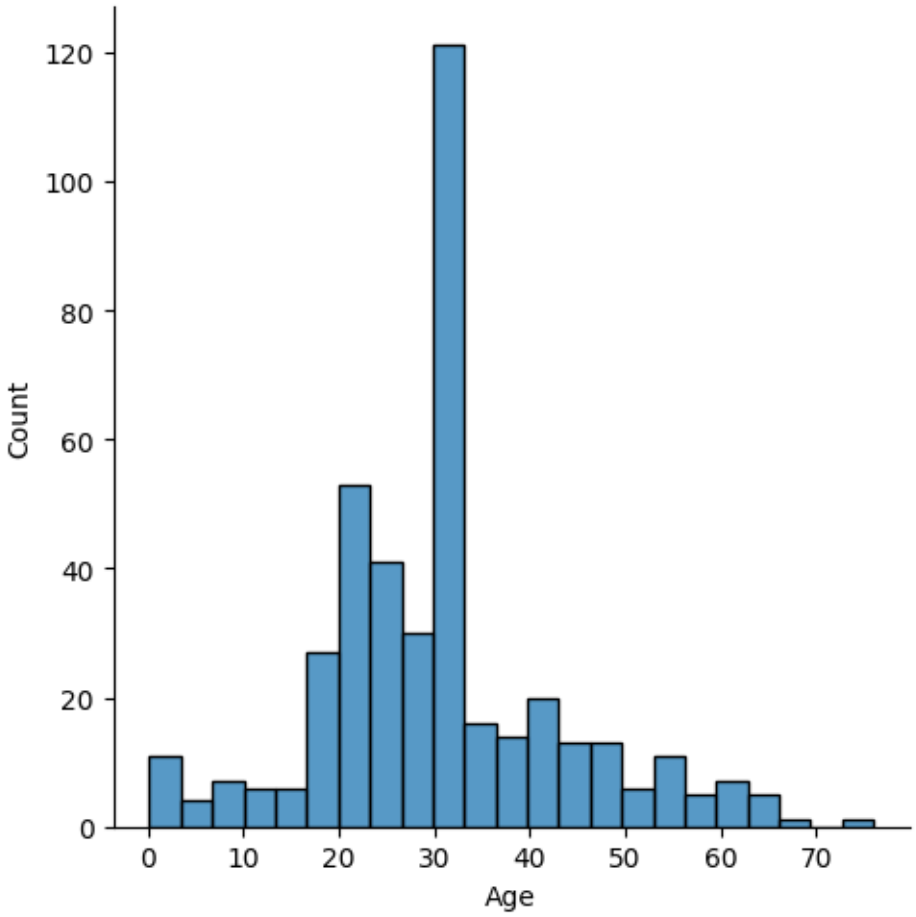
[95]:
```
sns.scatterplot(x="SibSp"         ,y="Survived",data=dataset)
```

[95]: <Axes: xlabel='SibSp', ylabel='Survived'>



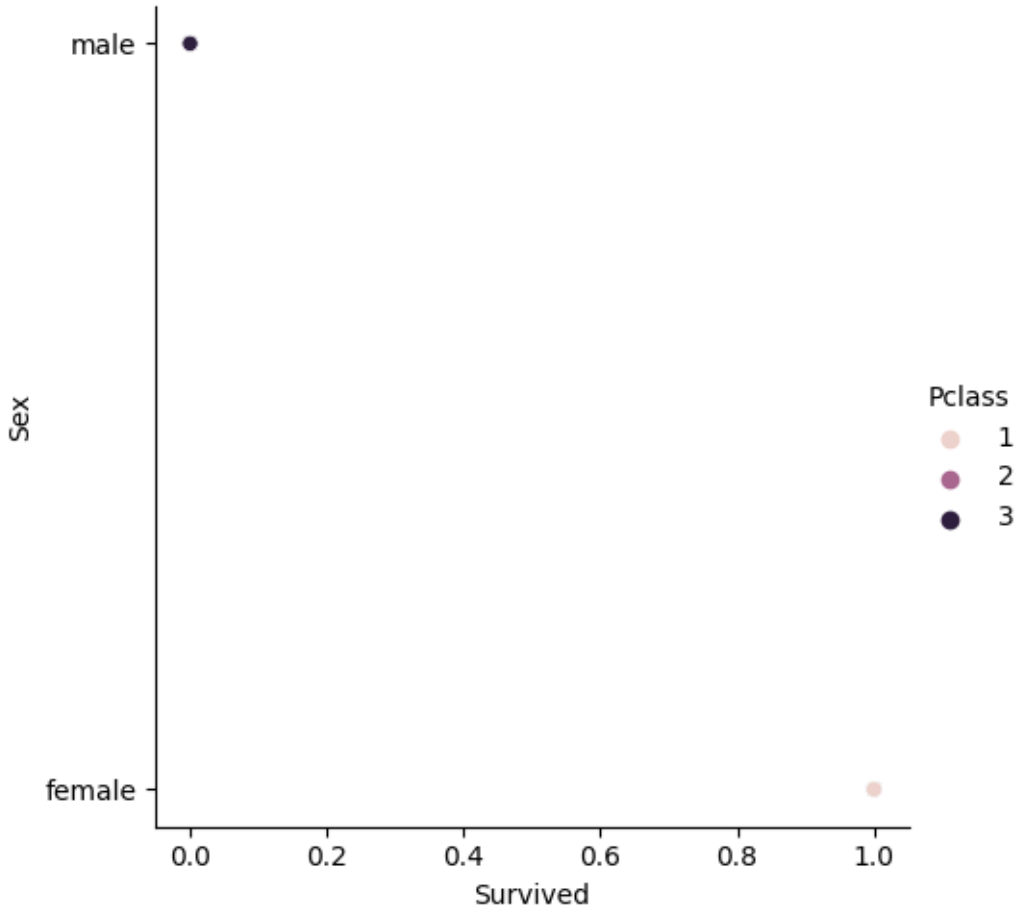
[96]:
```
sns.displot(dataset["Age"])
```

[96]: <seaborn.axisgrid.FacetGrid at 0x7b1b1aaf5b40>



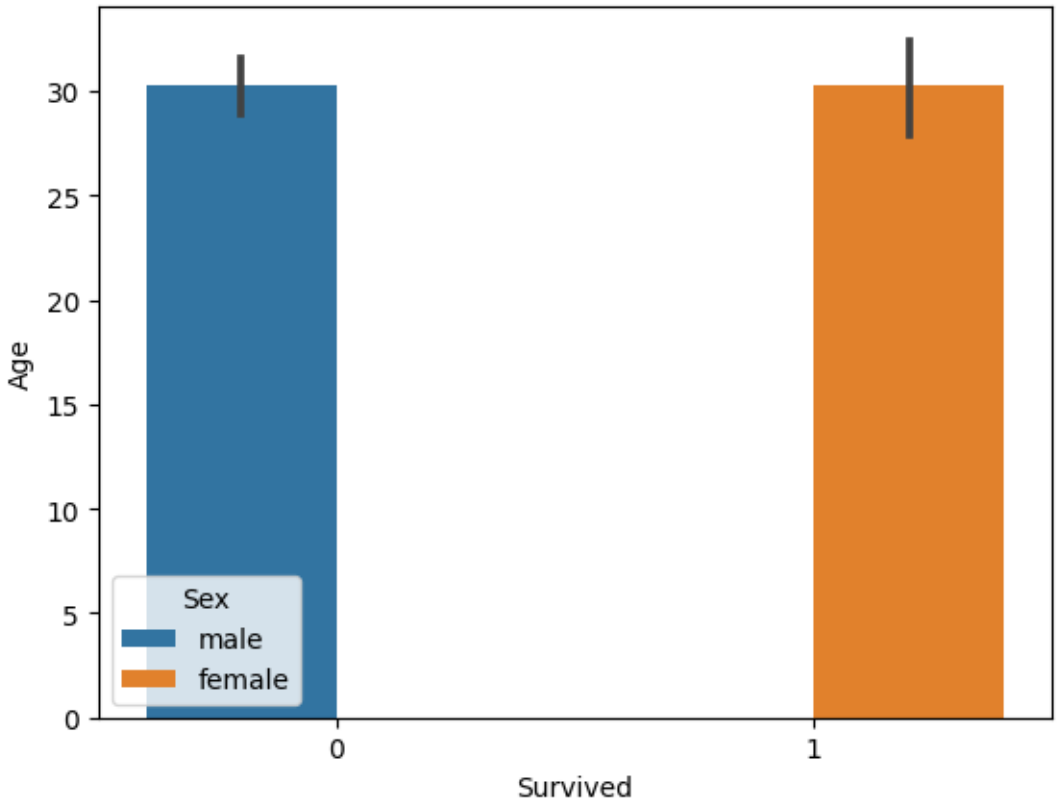
[97]:
```
sns.relplot(x="Survived",y="Sex",data=dataset,hue="Pclass")
```

[97]:
```
<seaborn.axisgrid.FacetGrid at 0x7b1b1a612ef0>
```



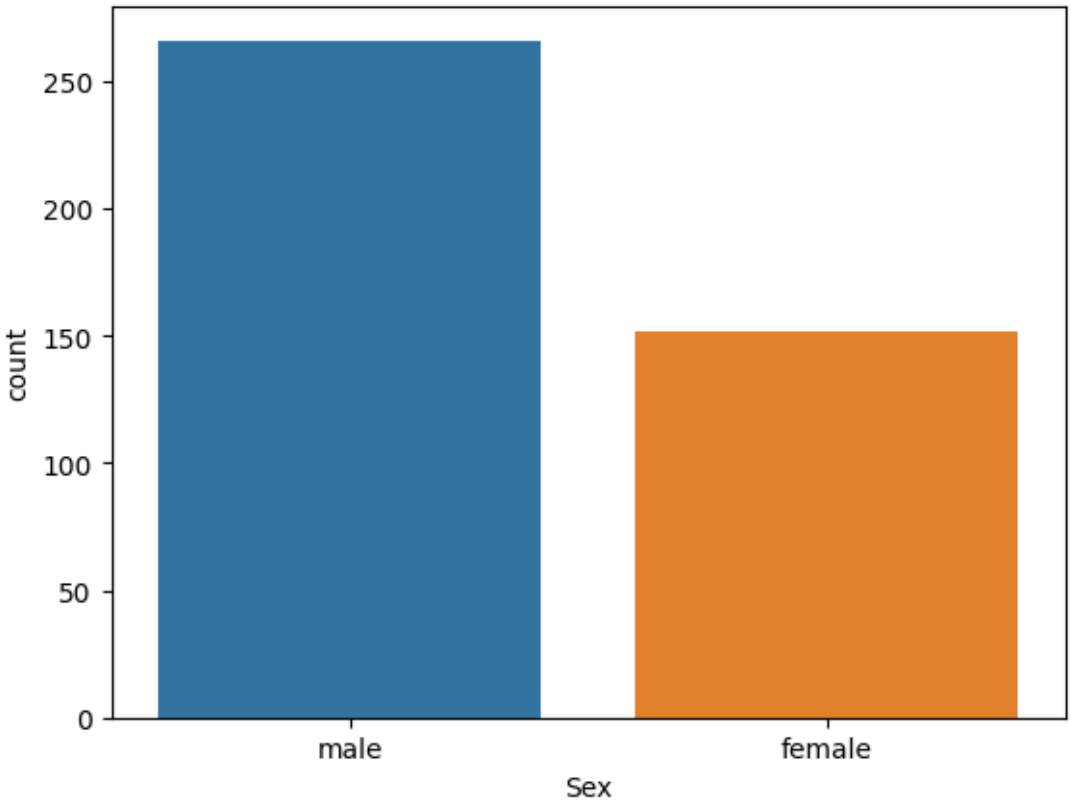
[98]:
```
sns.barplot(data=dataset,x="Survived",y="Age",hue="Sex")
```

[98]:
```
<Axes: xlabel='Survived', ylabel='Age'>
```



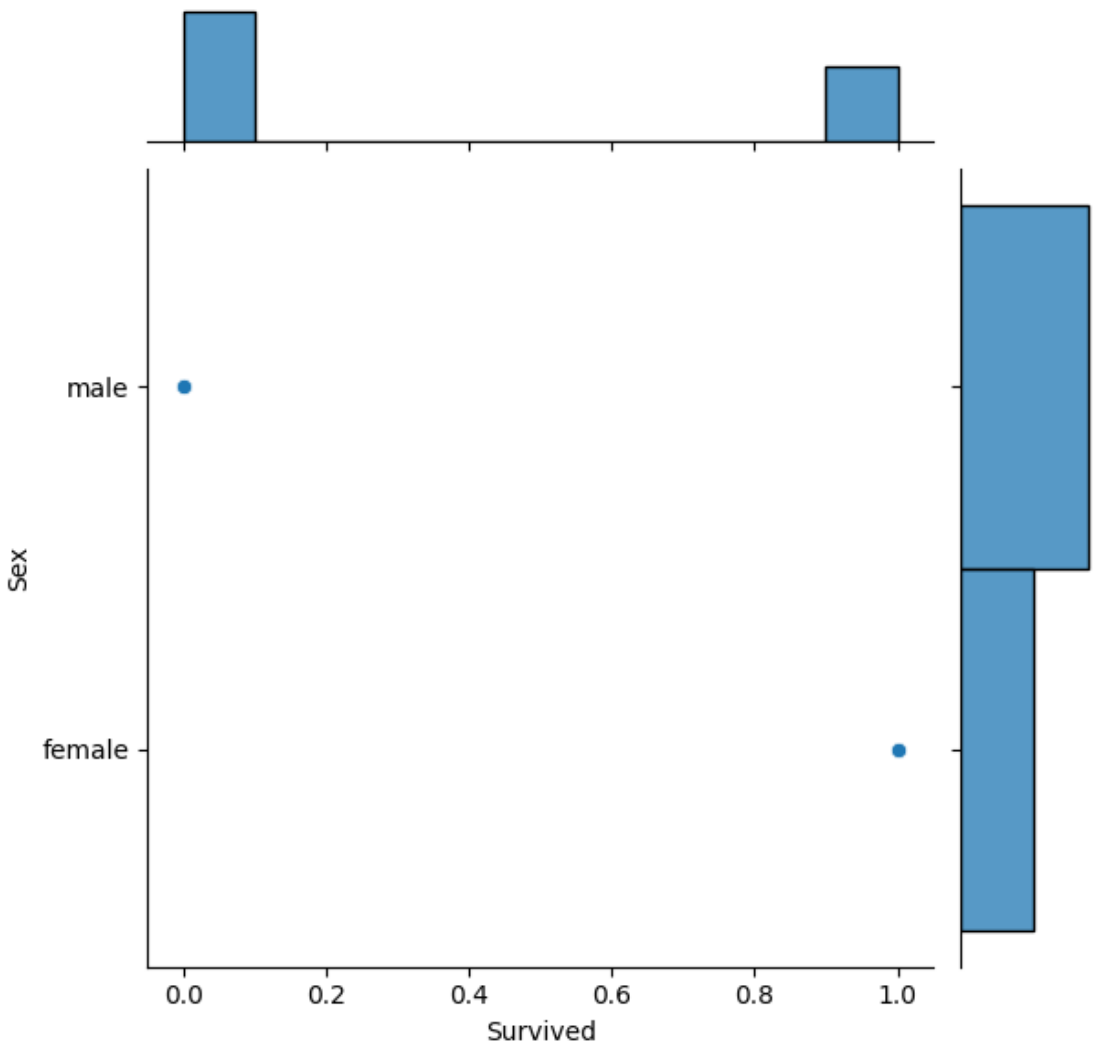
[99]:
```
sns.countplot(x='Sex',data=dataset)
```

[99]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



[100]:
```
sns.jointplot(x="Survived",y='Sex',data=dataset)
```

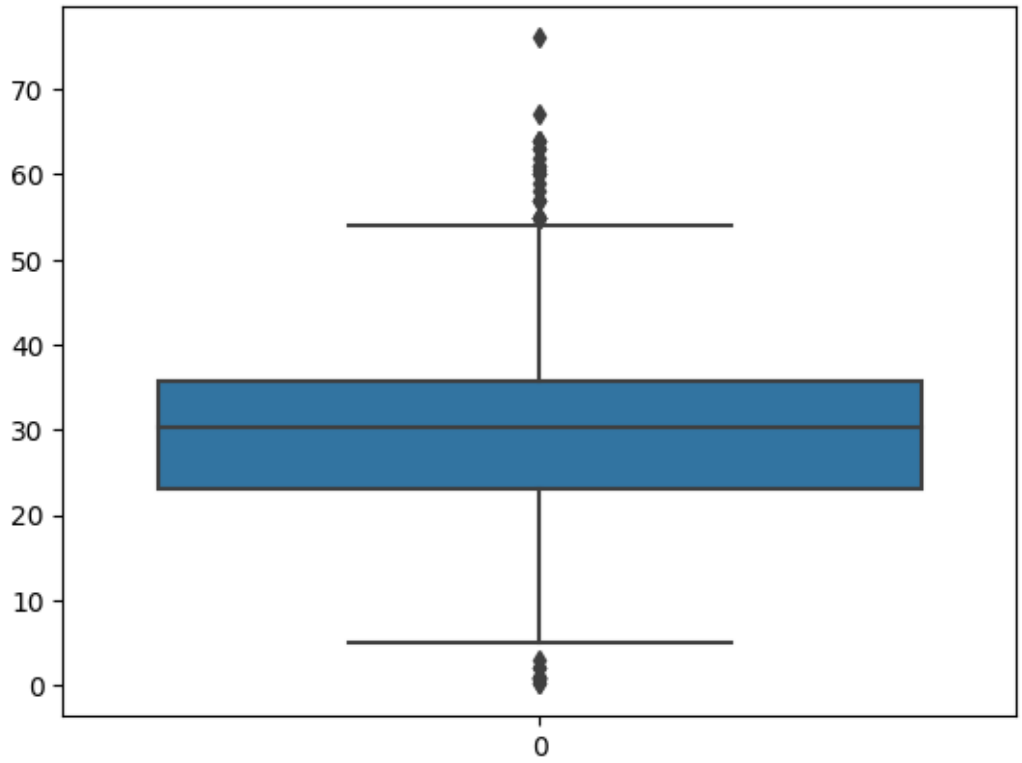
[100]:
```
<seaborn.axisgrid.JointGrid at 0x7b1b1b3ed090>
```



5 5.Outliers

[101]:
```
sns.boxplot(dataset.Age)
```

[101]: <Axes: >



6 6.Separating Dependent and Independent Varriables

[102]:
```
dependent_variable = dataset['Survived']
dependent_variable.head()
```

[102]:
```
0    0
1    1
2    0
3    0
4    1
Name: Survived, dtype: int64
```

[103]:
```
independent_variables = dataset[['PassengerId','Name','Pclass', 'Sex', 'Age',␣
↪'SibSp', 'Parch', 'Fare', 'Embarked']]
```

[104]:
```
independent_variables.head()
```

[104]:

	PassengerId	Name	Pclass	Sex	\
0	892	Kelly, Mr. James	3	male	
1	893	Wilkes, Mrs. James (Ellen Needs)	3	female	
2	894	Myles, Mr. Thomas Francis	2	male	

```
3          895                              Wirz, Mr. Albert       3    male
4          896  Hirvonen, Mrs. Alexander (Helga E Lindqvist)       3  female

    Age  SibSp  Parch     Fare Embarked
0  34.5      0      0   7.8292        Q
1  47.0      1      0   7.0000        S
2  62.0      0      0   9.6875        Q
3  27.0      0      0   8.6625        S
4  22.0      1      1  12.2875        S
```

[105]:
```
dependent_variable.shape
```

[105]:
```
(418,)
```

[106]:
```
independent_variables.shape
```

[106]:
```
(418, 9)
```

7 7.Encoding

[107]:
```
#Label encoding on Gender column
```

[108]:
```
from sklearn.preprocessing import LabelEncoder
```

[109]:
```
le=LabelEncoder()
```

[110]:
```
independent_variables["Sex"] = le.fit_transform(independent_variables["Sex"])
```

```
<ipython-input-110-c1630205b919>:1: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  independent_variables["Sex"] = le.fit_transform(independent_variables["Sex"])
```

8 8.splitting into training and testing set

[111]:
```
from sklearn.model_selection import train_test_split
independent_variables_train, independent_variables_test,␣
↪dependent_variable_train,dependent_variable_test=train_test_split(independent_variables,dep
↪3,random_state=0)
```

[112]:
```
independent_variables_train.shape, independent_variables_test.shape,␣
↪dependent_variable_train.shape,dependent_variable_test.shape
```

[112]: ((292, 9), (126, 9), (292,), (126,))

9 8.Feature scaling

[113]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler ()
```

[114]:
```
independent_variables = independent_variables.drop(columns=["Name"])
```

[115]:
```
independent_variables.head()
```

[115]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	892	3	1	34.5	0	0	7.8292	Q
1	893	3	0	47.0	1	0	7.0000	S
2	894	2	1	62.0	0	0	9.6875	Q
3	895	3	1	27.0	0	0	8.6625	S
4	896	3	0	22.0	1	1	12.2875	S

[118]:
```
independent_variables_train=sc.fit_transform(independent_variables_train)
independent_variables_test=sc.fit_transform(independent_variables_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-118-9b4fc125d233> in <cell line: 1>()
----> 1 independent_variables_train=sc.fit_transform(independent_variables_trai )
      2 independent_variables_test=sc.fit_transform(independent_variables_test)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_set_output.py in␣
 ↪wrapped(self, X, *args, **kwargs)
    138     @wraps(f)
    139     def wrapped(self, X, *args, **kwargs):
--> 140         data_to_wrap = f(self, X, *args, **kwargs)
    141         if isinstance(data_to_wrap, tuple):
    142             # only wrap the first output for cross decomposition

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in fit_transform(self,␣
 ↪X, y, **fit_params)
    876         if y is None:
    877             # fit method of arity 1 (unsupervised transformation)
--> 878             return self.fit(X, **fit_params).transform(X)
    879         else:
    880             # fit method of arity 2 (supervised transformation)

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
 ↪fit(self, X, y, sample_weight)
    822         # Reset internal state before fitting
```

```
    823         self._reset()
--> 824         return self.partial_fit(X, y, sample_weight)
    825
    826     def partial_fit(self, X, y=None, sample_weight=None):

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
↪partial_fit(self, X, y, sample_weight)
    859
    860         first_call = not hasattr(self, "n_samples_seen_")
--> 861         X = self._validate_data(
    862             X,
    863             accept_sparse=("csr", "csc"),

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,␣
↪X, y, reset, validate_separately, **check_params)
    563             raise ValueError("Validation should be done on X, y or both
↪")
    564         elif not no_val_X and no_val_y:
--> 565             X = check_array(X, input_name="X", **check_params)
    566             out = X
    567         elif no_val_X and not no_val_y:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
↪ensure_min_features, estimator, input_name)
    877                     array = xp.astype(array, dtype, copy=False)
    878                 else:
--> 879                     array = _asarray_with_order(array, order=order,␣
↪dtype=dtype, xp=xp)
    880             except ComplexWarning as complex_warning:
    881                 raise ValueError(

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_array_api.py in␣
↪_asarray_with_order(array, dtype, order, copy, xp)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in␣
↪__array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.
↪ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
```

```
   2072     def __array_wrap__(

ValueError: could not convert string to float: 'Cavendish, Mrs. Tyrell William␣
 ↪(Julia Florence Siegel)'
```

[119]:
```
dependent_variable_train=sc.fit_transform(dependent_variable_train)
dependent_variable_test=sc.fit_transform(dependent_variable_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-119-f14a8acc2321> in <cell line: 1>()
----> 1 dependent_variable_train=sc.fit_transform(dependent_variable_train)
      2 dependent_variable_test=sc.fit_transform(dependent_variable_test)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_set_output.py in␣
 ↪wrapped(self, X, *args, **kwargs)
    138     @wraps(f)
    139     def wrapped(self, X, *args, **kwargs):
--> 140         data_to_wrap = f(self, X, *args, **kwargs)
    141         if isinstance(data_to_wrap, tuple):
    142             # only wrap the first output for cross decomposition

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in fit_transform(self,␣
 ↪X, y, **fit_params)
    876         if y is None:
    877             # fit method of arity 1 (unsupervised transformation)
--> 878             return self.fit(X, **fit_params).transform(X)
    879         else:
    880             # fit method of arity 2 (supervised transformation)

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
 ↪fit(self, X, y, sample_weight)
    822         # Reset internal state before fitting
    823         self._reset()
--> 824         return self.partial_fit(X, y, sample_weight)
    825
    826     def partial_fit(self, X, y=None, sample_weight=None):

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
 ↪partial_fit(self, X, y, sample_weight)
    859
    860         first_call = not hasattr(self, "n_samples_seen_")
--> 861         X = self._validate_data(
    862             X,
    863             accept_sparse=("csr", "csc"),
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,
 ↪X, y, reset, validate_separately, **check_params)
    563             raise ValueError("Validation should be done on X, y or both
 ↪")
    564         elif not no_val_X and no_val_y:
--> 565             X = check_array(X, input_name="X", **check_params)
    566             out = X
    567         elif no_val_X and not no_val_y:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
 ↪ensure_min_features, estimator, input_name)
    900             # If input is 1D raise error
    901             if array.ndim == 1:
--> 902                 raise ValueError(
    903                     "Expected 2D array, got 1D array instead:\narray={}
 ↪\n"
    904                     "Reshape your data either using array.reshape(-1, 1
 ↪if "

ValueError: Expected 2D array, got 1D array instead:
array=[1. 0. 0. 0. 0. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 1. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0.
 0. 0. 1. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1.
 1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1.
 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1.
 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 0. 0.].
Reshape your data either using array.reshape(-1, 1) if your data has a single␣
 ↪feature or array.reshape(1, -1) if it contains a single sample.
```

[ ]: