Logistic Regression

In [16]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [107]:

```
df1 = pd.read_csv('logic.csv')
df1
```

Out[107]:

	Gender	Hours	Pass
0	M	1	False
1	F	2	False
2	M	3	False
3	M	4	False
4	F	5	True
5	F	6	True
6	F	7	True
7	M	8	True

In [108]:

```
df1.head(1)
```

Out[108]:

	Gender	Hours	Pass
0	M	1	False

In [109]:

```
df1.drop('Gender', axis='columns')
# features = all inputs fields are called features. and label
```

Out[109]:

	Hours	Pass
0	1	False
1	2	False
2	3	False
3	4	False
4	5	True
5	6	True
6	7	True
7	8	True

In [110]:

```
# Drop the unsessary data field
df1.drop('Gender', axis=1)
```

Out[110]:

	Hours	Pass
0	1	False
1	2	False
2	3	False
3	4	False
4	5	True
5	6	True
6	7	True
7	8	True

In [112]:

```
# Data curation for data presentation / data make for useable

df1.replace({False:0, True:1}, inplace=True)
df1
```

Out[112]:

	Gender	Hours	Pass
0	M	1	0
1	F	2	0
2	M	3	0
3	M	4	0
4	F	5	1
5	F	6	1
6	F	7	1
7	M	8	1

In [113]:

```
df1.head(1)
```

Out[113]:

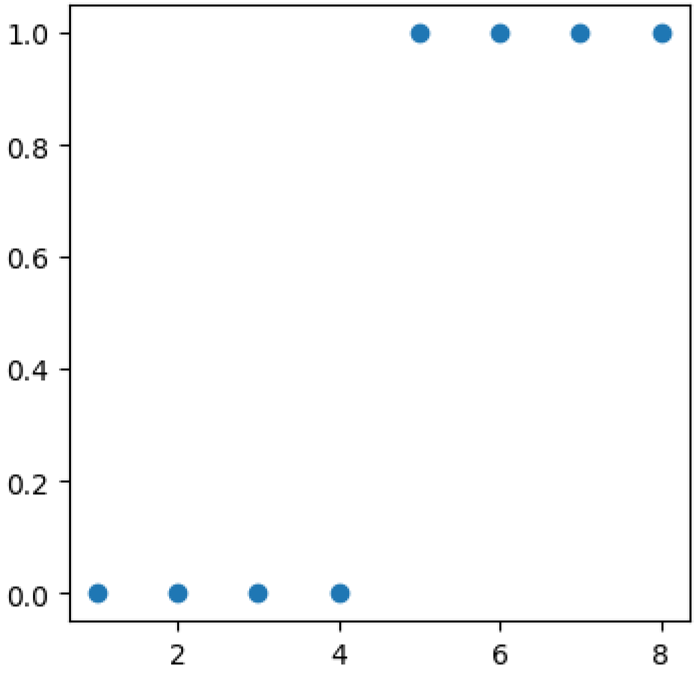
	Gender	Hours	Pass
0	M	1	0

In [ ]:

```
```

In [114]:

```
# scatter  sigmint function
plt.figure(figsize=(4,4))
plt.scatter(df1['Hours'], df1['Pass'])
plt.show()
```



In [24]:

```
import math
print(math.e)
print(math.pi)
```

```
2.718281828459045
3.141592653589793
```

In [25]:

```
# formula predict the value
#y = 1/(1+e^-x)
# e2.718
# 0.7 =1
# y =1/(1+2.7^-10)= 1.0(ans.)
# y = constant
# y =mx+c
# y=ax+bx+cx+..+c
```

In [115]:

```
from sklearn.model_selection import train_test_split
x =df1.drop('Pass', axis=1)
y =df1['Hours']# model.coef_ , Model
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

In [116]:

```
x_train
```

Out[116]:

	Gender	Hours
5	F	6
6	F	7
7	M	8
2	M	3
1	F	2
0	M	1

In [117]:

```
x_test
```

Out[117]:

	Gender	Hours
4	F	5
3	M	4

In [118]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model
```

Out[118]:

```
▾ LogisticRegression
LogisticRegression()
```

In [222]:

```
model.fit(x_train, y_train)
```

Out[222]:

```
▾ LogisticRegression
LogisticRegression()
```

In [226]:

```
y_pred = model.predict(x_test)
```

In [227]:

```
y_pred
```

Out[227]:

```
array([1, 1, 0, 0], dtype=int64)
```

In [228]:

```
x_test
```

Out[228]:

	age
10	45
12	48
3	28
7	27

In [229]:
```
# model.score(x_test, y_test)
```

In [230]:
```
elu =math.e
```

In [231]:
```
# sigmoid function
# Exponential Linear Units

def sig(x):
    elu =math.e

#     return 1/(1+math.exp(-x))
    return 1/(1+elu**(-0.5*x))
```

In [232]:
```
x = 3
sig(x)
```

Out[232]:
```
0.8175744761936437
```

In [233]:
```
import math
print(math.e)
print(math.pi)
```

```
2.718281828459045
3.141592653589793
```

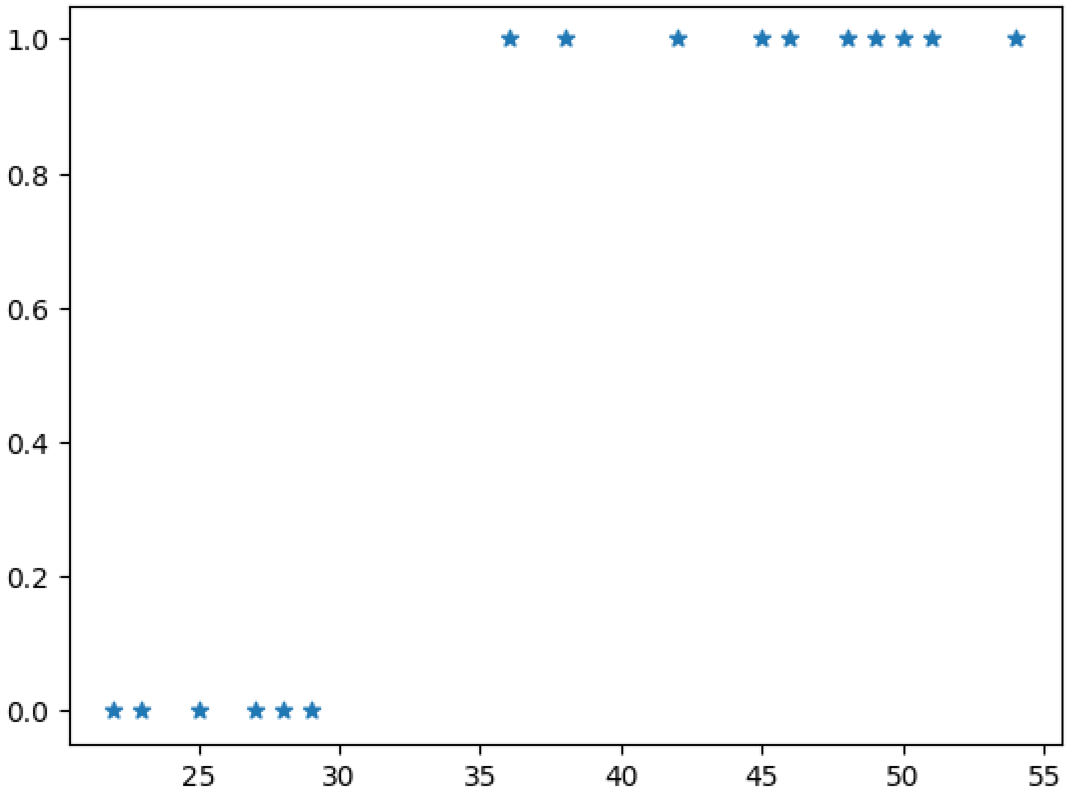
In [234]:
```
import pandas as pd
df=pd.read_csv('ins.csv')
df.head(1)
```

Out[234]:

	age	insurance
0	22	0

In [235]:
```
plt.scatter(df['age'], df['insurance'], marker='*')
```

Out[235]:
```
<matplotlib.collections.PathCollection at 0x18cb1cc95d0>
```



In [236]:
```
plt.show()
```

In [245]:
```
from sklearn.model_selection import train_test_split
```

In [250]:
```
x_train, x_test, y_train, y_test = train_test_split(df[['age']],df['insurance'
```

In [251]:
```
from sklearn.linear_model import LogisticRegression
```

In [252]:
```
reg =LogisticRegression()
```

In [253]:
```
reg.fit(X_train, Y_train)
```

Out[253]:
```
▾ LogisticRegression
LogisticRegression()
```

In [254]:
```
Y_test
```

Out[254]:
```
7     0
1     1
15    1
4     0
Name: insurance, dtype: int64
```

In [255]:

```
Y_pred = reg.predict(X_test)
Y_pred
```

Out[255]:

```
array([0, 1, 1, 0], dtype=int64)
```

In [256]:

```
reg.predict_proba(X_test)
```

Out[256]:

```
array([[9.85677615e-01, 1.43223851e-02],
       [1.14460337e-07, 9.99999886e-01],
       [1.08169846e-06, 9.99998918e-01],
       [9.99273288e-01, 7.26712428e-04]])
```

In [257]:

```
reg.coef_
```

Out[257]:

```
array([[0.74868678]])
```

In [258]:

```
reg.intercept_
```

Out[258]:

```
array([-24.44604862])
```

In [259]:

```
reg.score(X_test, Y_pred)
```

Out[259]:

```
1.0
```

In [260]:

```
# sigmoid
import math
eu =math.e
```

In [261]:

```
def sig(x):
    import math
    return 1/(1+(math.e **(-x)))
```

In [262]:

```
def pred_f(age):
    # y =mx+c (coef*x+intercept)
    z = 0.738* age+(-24.30)
    y =sig(z)
    return y
```

In [263]:

```
age = 36
round(pred_f(age))
```

Out[263]:

```
1
```

In [264]:

```
import pandas as pd
df =pd.read_csv('social.csv')
df.head(1)
```

Out[264]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0

In [265]:

```
df.drop(['User ID', 'Gender'], axis='columns', inplace=True)
df
```

Out[265]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

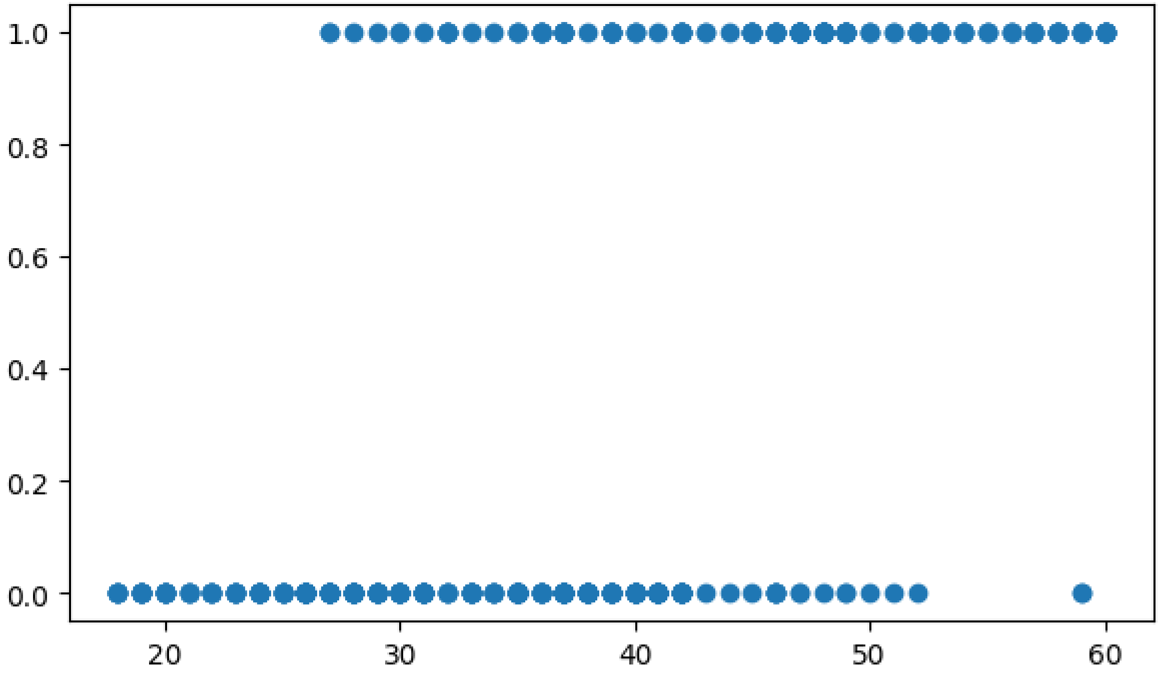
400 rows × 3 columns

In [266]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(7,4))

plt.scatter(df['Age'], df['Purchased'])
```

Out[266]:

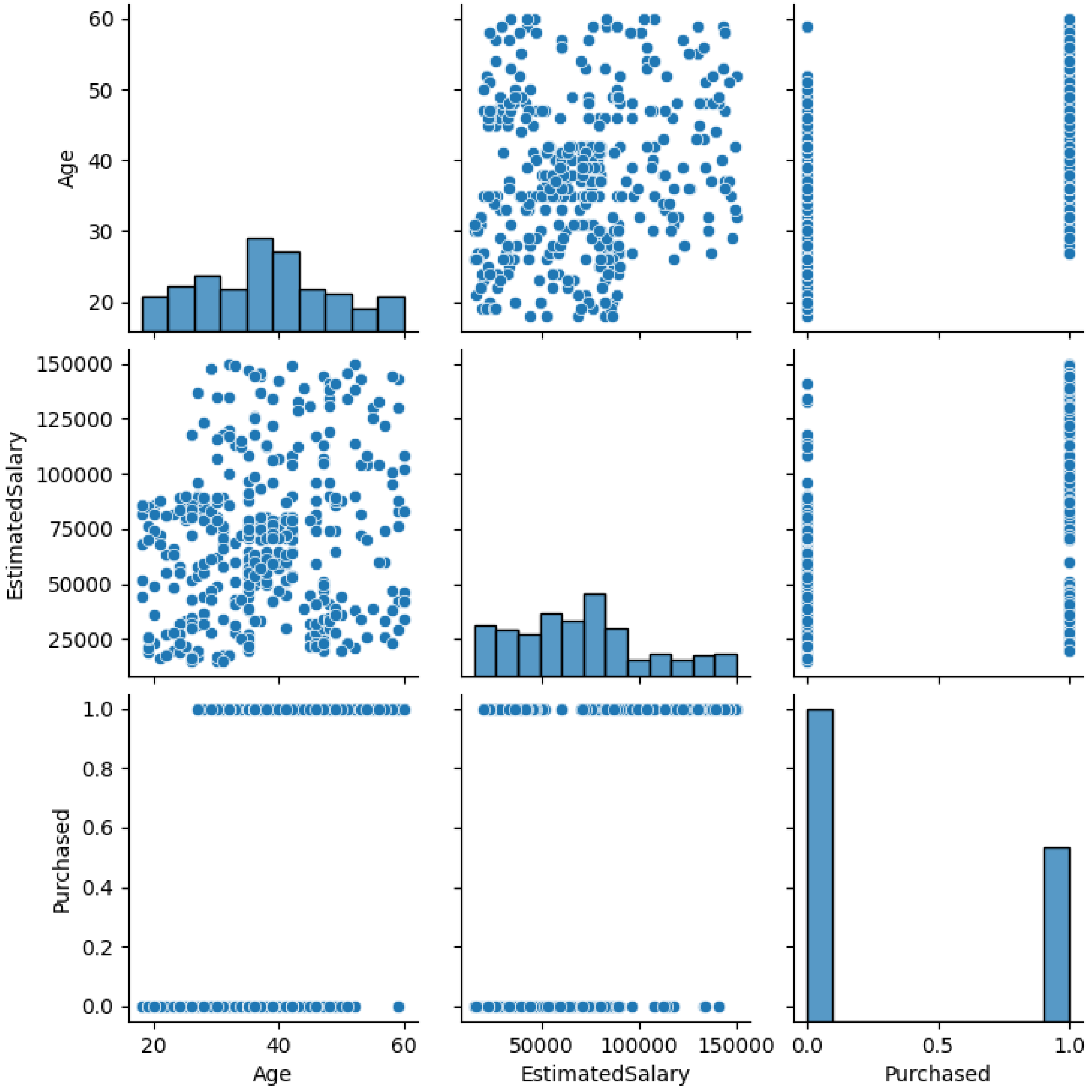
```
<matplotlib.collections.PathCollection at 0x18cb1190390>
```



In [267]:
```
import seaborn as sns
plt.figure(figsize=(4,4))
sns.pairplot(df)
```

Out[267]:
```
<seaborn.axisgrid.PairGrid at 0x18cb1db9490>
```

```
<Figure size 400x400 with 0 Axes>
```



In [268]:
```

from sklearn.model_selection import train_test_split
```

In [269]:
```
x= df.drop('Purchased', axis= 'columns')
```

In [270]:
```
y = df['Purchased']
```

In [271]:
```
x_train,x_test, y_train,y_test = train_test_split(x,y)
```

In [272]:
```
x_train.shape
```
Out[272]:
```
(300, 2)
```

In [273]:
```
x_test.shape
```
Out[273]:
```
(100, 2)
```

In [274]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [275]:
```
model.fit(x_train, y_train)
```
Out[275]:
```
▾ LogisticRegression
LogisticRegression()
```

In [279]:
```
y_pred= model.predict(x_test)
y_pred
```
Out[279]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [280]:
```
model.score(x_test, y_pred)
```
Out[280]:
```
1.0
```

In [281]:
```
model.score(x_test, y_test)*100
```
Out[281]:
```
70.0
```

In [282]:
```
# Logistic Regression -Iris Dataset
```

```
In [283]: import pandas as pd
```

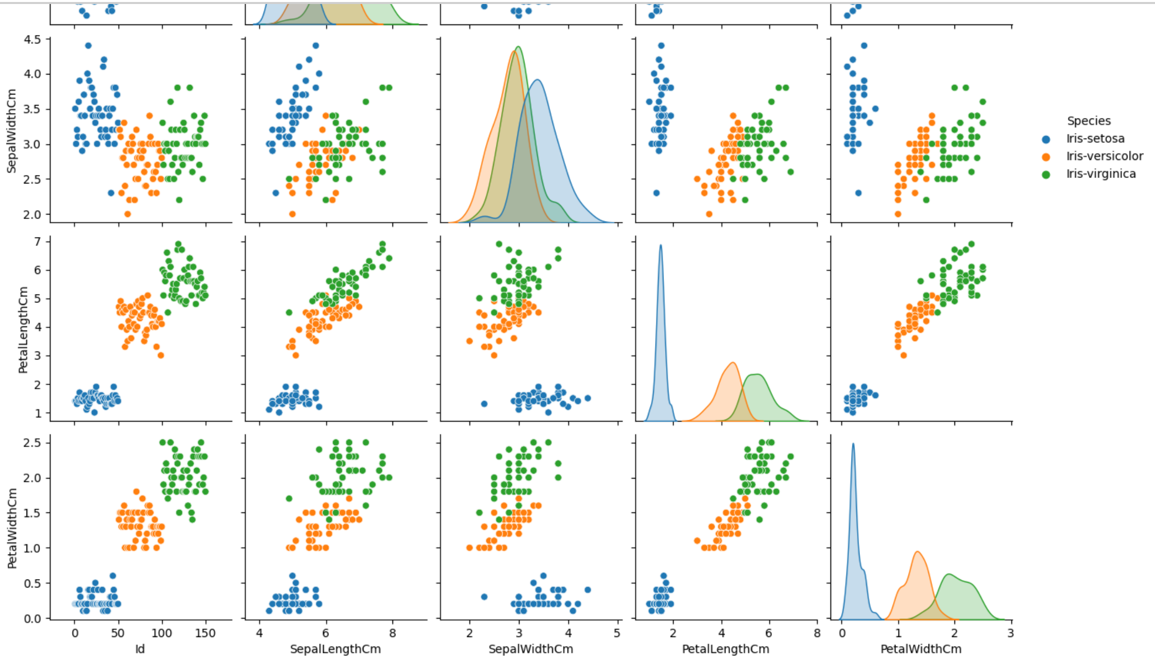
```
In [284]: df= pd.read_csv('iris.csv')
          df.head(1)
```

Out[284]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa

In [285]:

```
import seaborn as sns
plt.figure(figsize=(4,4))
sns.pairplot(df, hue='Species')
# Vsetosa-> sepal lenght(low), sepal width(low), petal length(low)
#petal width(low)
# Virginica-> sepal len petal width(high), sepal width(medium), petal len(high
```



In [286]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [287]:

```
df['Species'].unique()
```

Out[287]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [289]:

```
df['Species'].replace({'Iris-setosa': '1', 'Iris-versicolor':'2'}, inplace=True
```

In [290]:
```
df.sample(10)
```

Out[290]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
30	31	4.8	3.1	1.6	0.2	1
73	74	6.1	2.8	4.7	1.2	2
93	94	5.0	2.3	3.3	1.0	2
58	59	6.6	2.9	4.6	1.3	2
146	147	6.3	2.5	5.0	1.9	Iris-virginica
95	96	5.7	3.0	4.2	1.2	2
106	107	4.9	2.5	4.5	1.7	Iris-virginica
28	29	5.2	3.4	1.4	0.2	1
120	121	6.9	3.2	5.7	2.3	Iris-virginica
105	106	7.6	3.0	6.6	2.1	Iris-virginica

In [291]:
```
from sklearn.model_selection import train_test_split
```

In [292]:
```
x_train, x_test, y_train, y_test = train_test_split(df[['SepalLengthCm','SepalW
```

In [293]:
```
x_train.shape
```

Out[293]:
```
(112, 4)
```

In [294]:
```
x_test.shape
```

Out[294]:
```
(38, 4)
```

In [295]:
```
y_test.shape
```

Out[295]:
```
(38, 1)
```

In [296]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [297]:
```
model.fit(x_train, y_train)
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\utils\validation.py:1183: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n
_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed t
o converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[297]:
```
▾ LogisticRegression

LogisticRegression()
```

In [298]:
```
model.predict(x_test)
```

Out[298]:
```
array(['1', '1', '2', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', '1', '1', '2', 'Iris-virginica',
       'Iris-virginica', '1', 'Iris-virginica', '2', 'Iris-virginica',
       '2', '1', '2', '2', '2', '1', '1', 'Iris-virginica', '2', '1',
       'Iris-virginica', 'Iris-virginica', '1', '1', '2', '2', '1', '2',
       'Iris-virginica', '1', 'Iris-virginica', 'Iris-virginica', '1'],
      dtype=object)
```

In [299]:
```
model.score(x_test, y_test)
```

Out[299]:
```
0.9736842105263158
```

In [300]:
```
model
```

Out[300]:
```
▾ LogisticRegression

LogisticRegression()
```

In [301]:
```
model.coef_
```

Out[301]:
```
array([[-0.47658191,  0.80632333, -2.36826275, -0.97671983],
       [ 0.28087822, -0.23269826, -0.16694849, -0.77974292],
       [ 0.19570369, -0.57362508,  2.53521124,  1.75646275]])
```

In [302]:

```
model.intercept_
```

Out[302]:

```
array([ 10.12094814,   2.99462066, -13.11556879])
```

In [303]:

```
print('Accuracy Of My Model:',model.score(x_test, y_test)*100, '%')
```

```
Accuracy Of My Model: 97.36842105263158 %
```

In [304]:

```
def sig(x):
    import math
    return 1/(1+math.e **(x))
```

```
a-SepalLengthCm
b-SepalWidthCm
c-PetalLengthCm
d-PetalWidthCm
```

In [305]:

```
model.coef_
```

Out[305]:

```
array([[-0.47658191,  0.80632333, -2.36826275, -0.97671983],
       [ 0.28087822, -0.23269826, -0.16694849, -0.77974292],
       [ 0.19570369, -0.57362508,  2.53521124,  1.75646275]])
```

In [306]:

```
model.intercept_
```

Out[306]:

```
array([ 10.12094814,   2.99462066, -13.11556879])
```

In [307]:

```
def pred_f(a, b, c, d):
    z = (-0.33*a+0.95*b-2.31*c-0.97*d+8.36)+(0.50*a-0.34*b-0.12*c-0.83*d+1.81)
    y =sig(z)
    return y
```

In [308]:

```
import seaborn as sns
```

In [309]:
```
titanic = sns.load_dataset('Titanic')
titanic
```

Out[309]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_mal
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru
...	...	...	...	...	...	...	...	...	...	...	.
886	0	2	male	27.0	0	0	13.0000	S	Second	man	Tru
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fals
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	Fals
889	1	1	male	26.0	0	0	30.0000	C	First	man	Tru
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	Tru

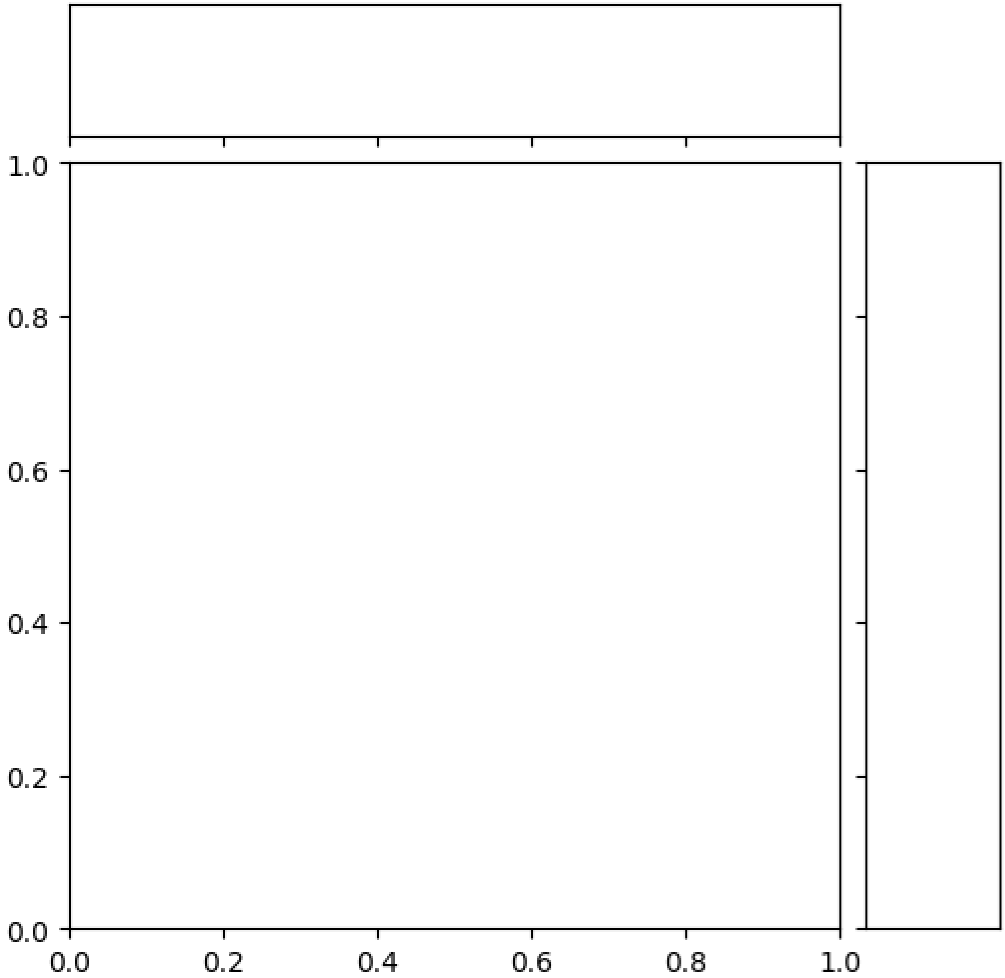
891 rows × 15 columns

In [351]:

```
sns.jointplot(x='sex', y='survived', hue='class')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[351], line 1
----> 1 sns.jointplot(x='sex', y='survived', hue='class')

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axis
grid.py:2241, in jointplot(data, x, y, hue, kind, height, ratio, space, dropn
a, xlim, ylim, color, palette, hue_order, hue_norm, marginal_ticks, joint_kw
s, marginal_kws, **kwargs)
   2238     dropna = True
   2240 # Initialize the JointGrid object
-> 2241 grid = JointGrid(
   2242     data=data, x=x, y=y, hue=hue,
   2243     palette=palette, hue_order=hue_order, hue_norm=hue_norm,
   2244     dropna=dropna, height=height, ratio=ratio, space=space,
   2245     xlim=xlim, ylim=ylim, marginal_ticks=marginal_ticks,
   2246 )
   2248 if grid.hue is not None:
   2249     marginal_kws.setdefault("legend", False)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axis
grid.py:1722, in JointGrid.__init__(self, data, x, y, hue, height, ratio, spa
ce, palette, hue_order, hue_norm, dropna, xlim, ylim, marginal_ticks)
   1719     ax_marg_y.xaxis.grid(False)
   1721 # Process the input variables
-> 1722 p = VectorPlotter(data=data, variables=dict(x=x, y=y, hue=hue))
   1723 plot_data = p.plot_data.loc[:, p.plot_data.notna().any()]
   1725 # Possibly drop NA

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_old
core.py:640, in VectorPlotter.__init__(self, data, variables)
    635 # var_ordered is relevant only for categorical axis variables, and ma
y
    636 # be better handled by an internal axis information object that track
s
    637 # such information and is set up by the scale_* methods. The analogou
s
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDic
t
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643 
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_old
core.py:701, in VectorPlotter.assign_variables(self, data, variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_old
```

```
core.py:938, in VectorPlotter._assign_variables_longform(self, data, **kwarg
s)
    933 elif isinstance(val, (str, bytes)):
    934 
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941 
    942     # Otherwise, assume the value is itself data
    943 
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Seri
es):

ValueError: Could not interpret value `sex` for parameter `x`
```



In [352]:
```
# scatter plot
iris= sns.load_dataset('iris')
iris.head(1)
```

Out[352]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

In [353]:

```
# ('iris')
```

In [354]:

```
tips  = sns.load_dataset('tips')
tips
```

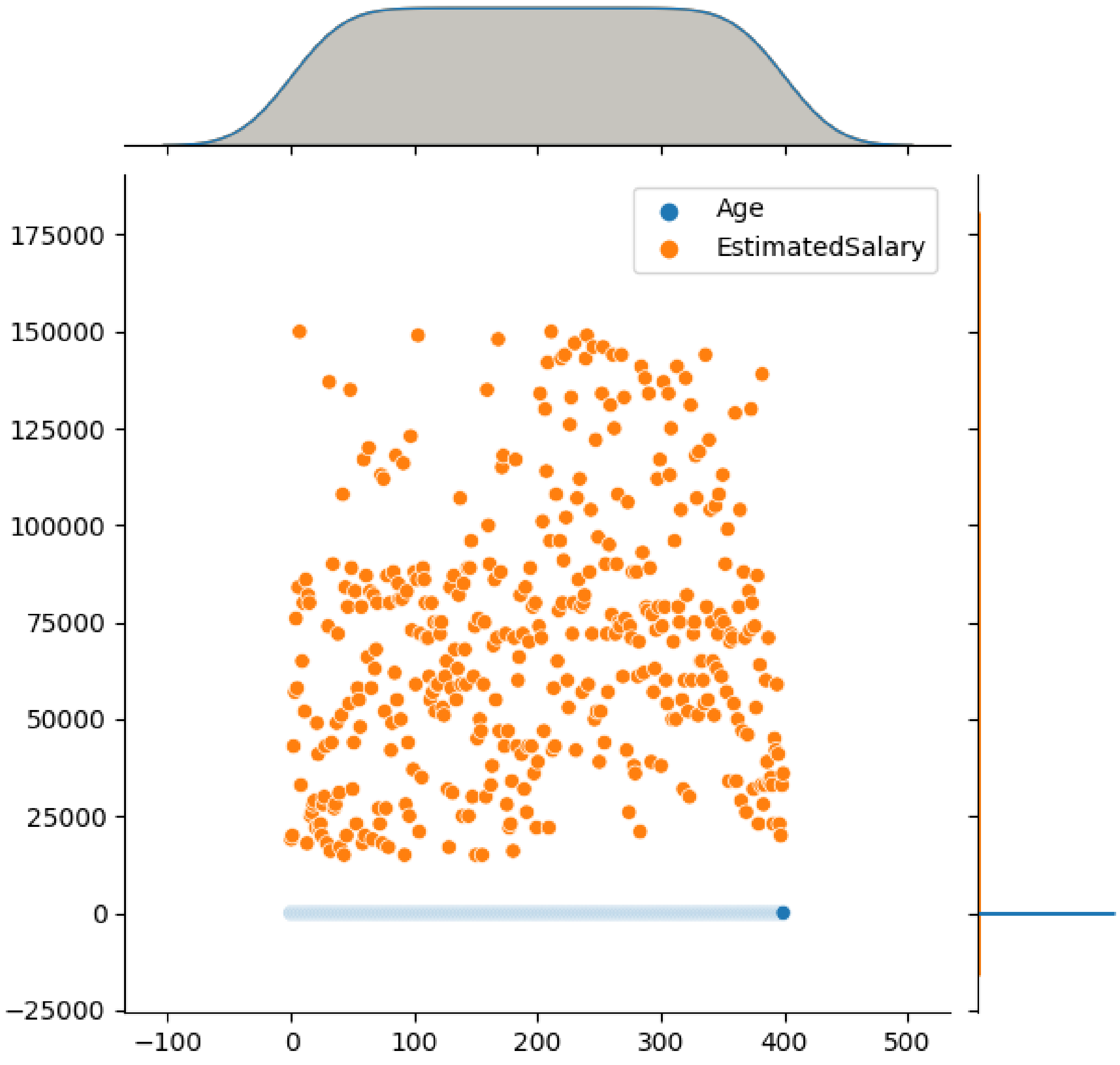
Out[354]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

```
In [355]: sns.jointplot(x)
```

```
Out[355]: <seaborn.axisgrid.JointGrid at 0x18cb8f51810>
```



In [366]:

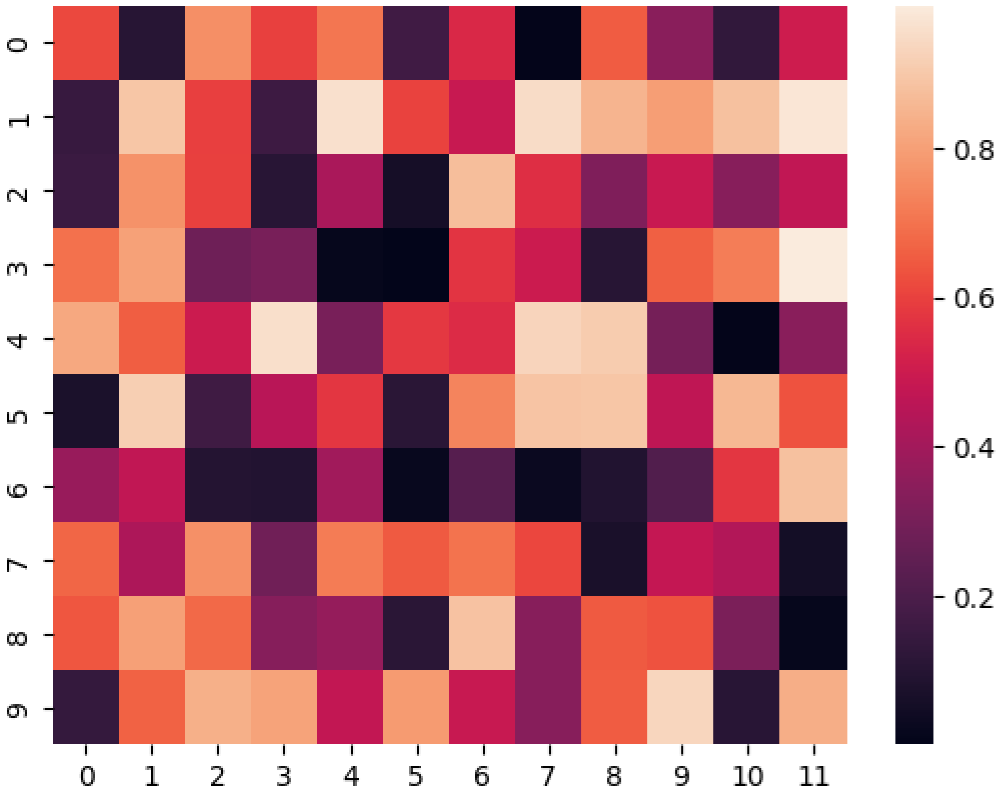
```
import numpy as np
a = np.random.rand(10, 12)
a
```

Out[366]:

```
array([[0.61403433, 0.10460578, 0.7625616 , 0.59787516, 0.70698845,
        0.17157899, 0.54176863, 0.00814857, 0.65129224, 0.34230904,
        0.13315401, 0.50088605],
       [0.14524483, 0.89011443, 0.59655651, 0.16287344, 0.96046712,
        0.60409859, 0.48536095, 0.94847499, 0.84789009, 0.79650537,
        0.88050038, 0.9786032 ],
       [0.15520051, 0.7675048 , 0.60041108, 0.10735425, 0.41850892,
        0.05643501, 0.87255858, 0.55601639, 0.32158732, 0.49045056,
        0.34096612, 0.46966974],
       [0.69590479, 0.80415423, 0.28009336, 0.30684191, 0.01712098,
        0.00201448, 0.57303677, 0.49957271, 0.10345631, 0.66047766,
        0.72071592, 0.99043891],
       [0.81987504, 0.65690939, 0.49633943, 0.95860488, 0.30653284,
        0.57734366, 0.54837524, 0.92990012, 0.90668612, 0.29777413,
        0.00709664, 0.34536544],
       [0.07332368, 0.91652665, 0.16458871, 0.45246868, 0.57589385,
        0.11128508, 0.73899858, 0.88710483, 0.89080114, 0.46560617,
        0.85713925, 0.63143983],
       [0.37707714, 0.47081403, 0.0957422 , 0.09363179, 0.39762973,
        0.02153215, 0.22211226, 0.03243622, 0.09037517, 0.21223435,
        0.57512189, 0.88176715],
       [0.67368659, 0.42358035, 0.76405142, 0.28457424, 0.71758855,
        0.64687459, 0.70012905, 0.60986886, 0.06874446, 0.47851213,
        0.43512756, 0.05283265],
       [0.64197858, 0.80072764, 0.67941894, 0.33762403, 0.36965593,
        0.1132895 , 0.8851748 , 0.33827889, 0.65044102, 0.63294083,
        0.31251355, 0.01631689],
       [0.13851685, 0.66338524, 0.83753192, 0.8088139 , 0.47561619,
        0.78596813, 0.48727138, 0.33849505, 0.65232061, 0.93569829,
        0.10897531, 0.8359316 ]])
```

In [367]:
```
sns.heatmap(a)
```

Out[367]:
```
<Axes: >
```



In [368]:
```
titanic = sns.load_dataset('Titanic')
titanic
```

Out[368]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_mal
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru
...	...	...	...	...	...	...	...	...	...	...	.
886	0	2	male	27.0	0	0	13.0000	S	Second	man	Tru
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fals
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	Fals
889	1	1	male	26.0	0	0	30.0000	C	First	man	Tru
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	Tru

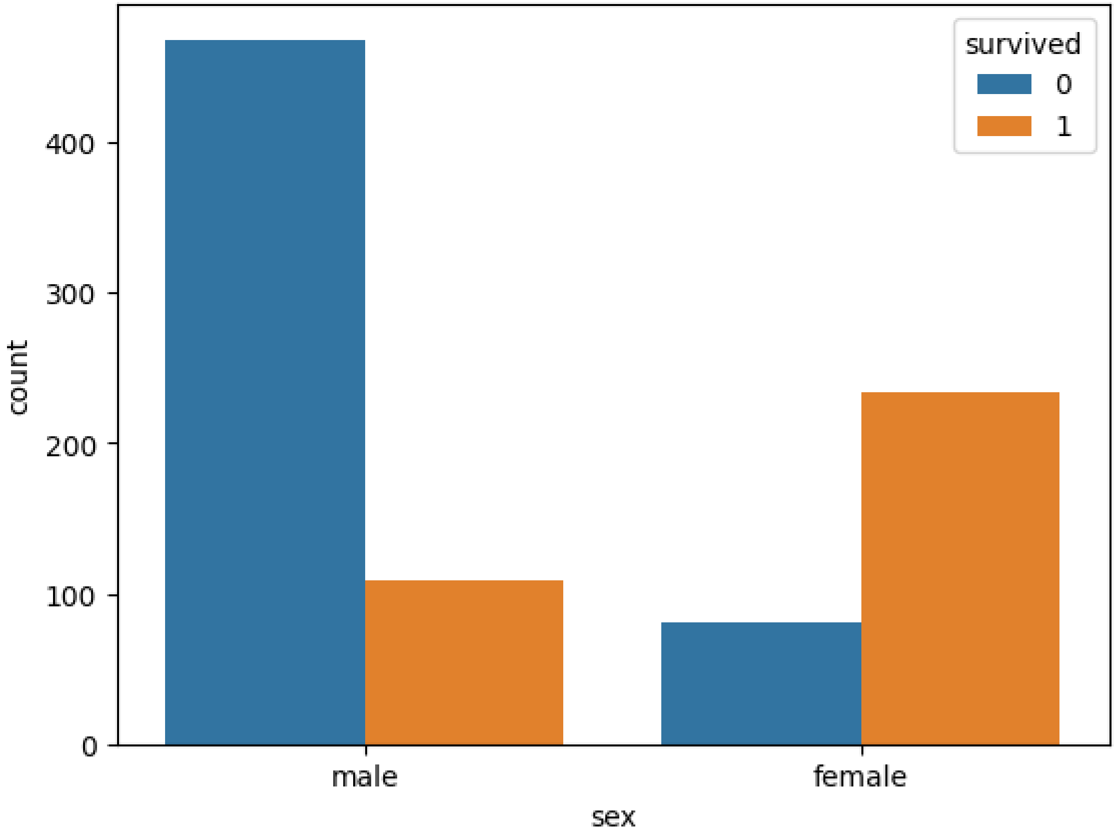
891 rows × 15 columns

In [386]:

```
# count plot
titanic.head(1)
sns.countplot(x ='sex', hue='survived', data=titanic)
```

Out[386]:

```
<Axes: xlabel='sex', ylabel='count'>
```



In [387]:

```
#violin plot
tips.head(1)
```

Out[387]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2

In [388]:

```
# pairplot
tips.head(1)
```

Out[388]:

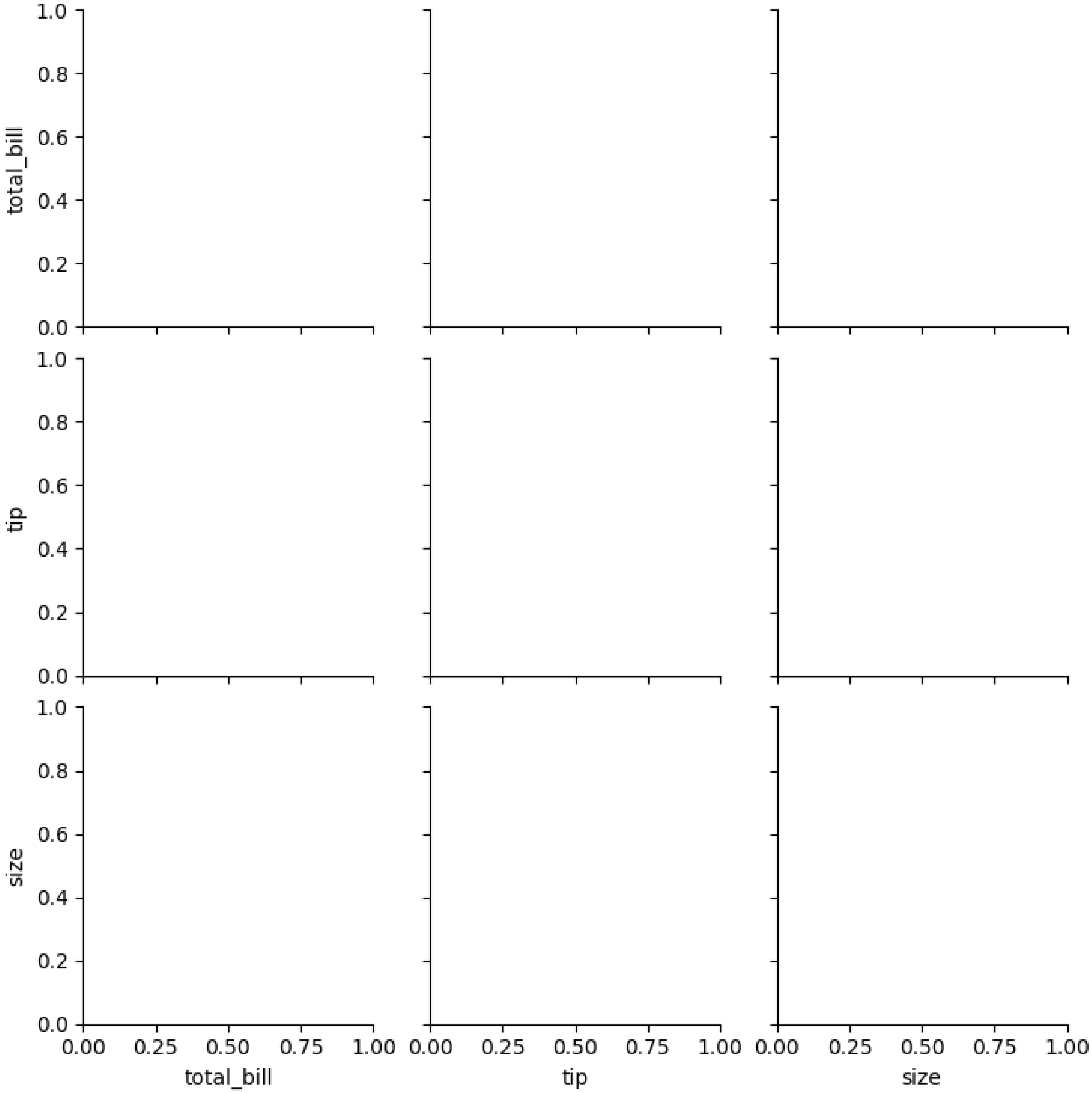
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2

In [389]:

```
#Pair Grid
sns.PairGrid(tips)
```

Out[389]:

```
<seaborn.axisgrid.PairGrid at 0x18cba649ad0>
```



In [390]:

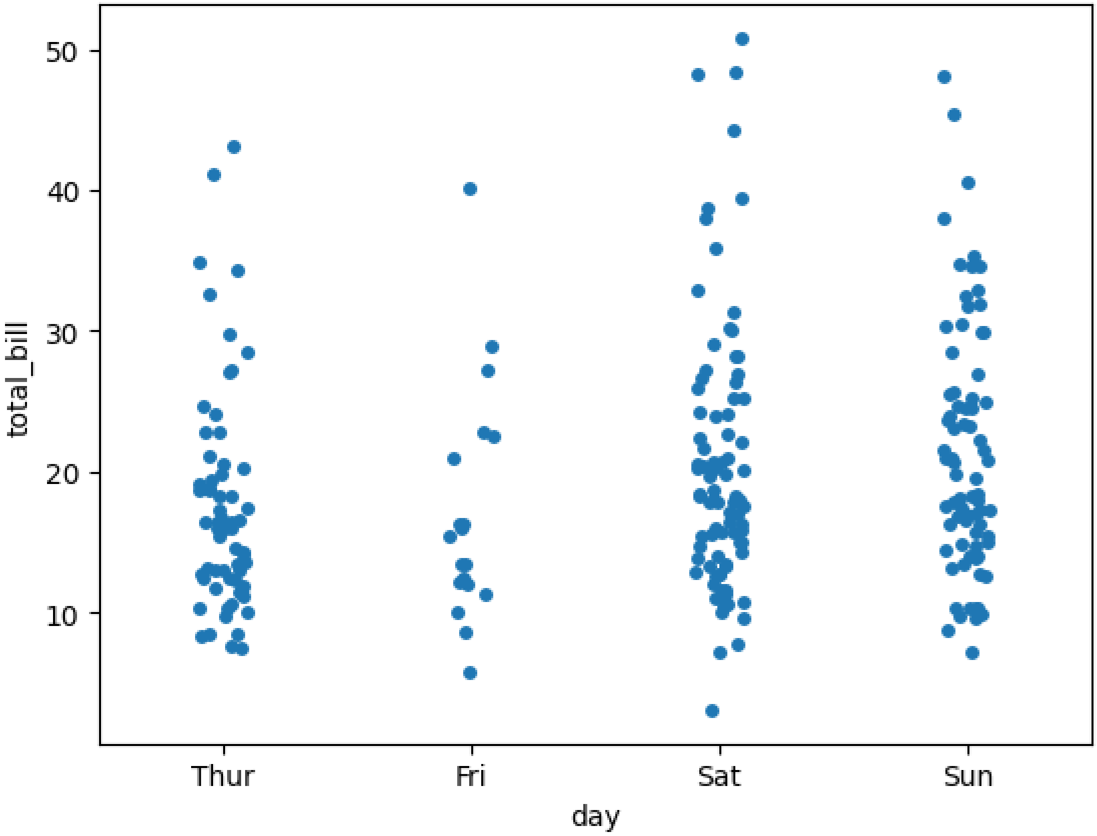
```
#Strip Plot
tips.head(1)
```

Out[390]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2

In [391]:
```
sns.stripplot(x='day', y='total_bill', data=tips)
```

Out[391]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



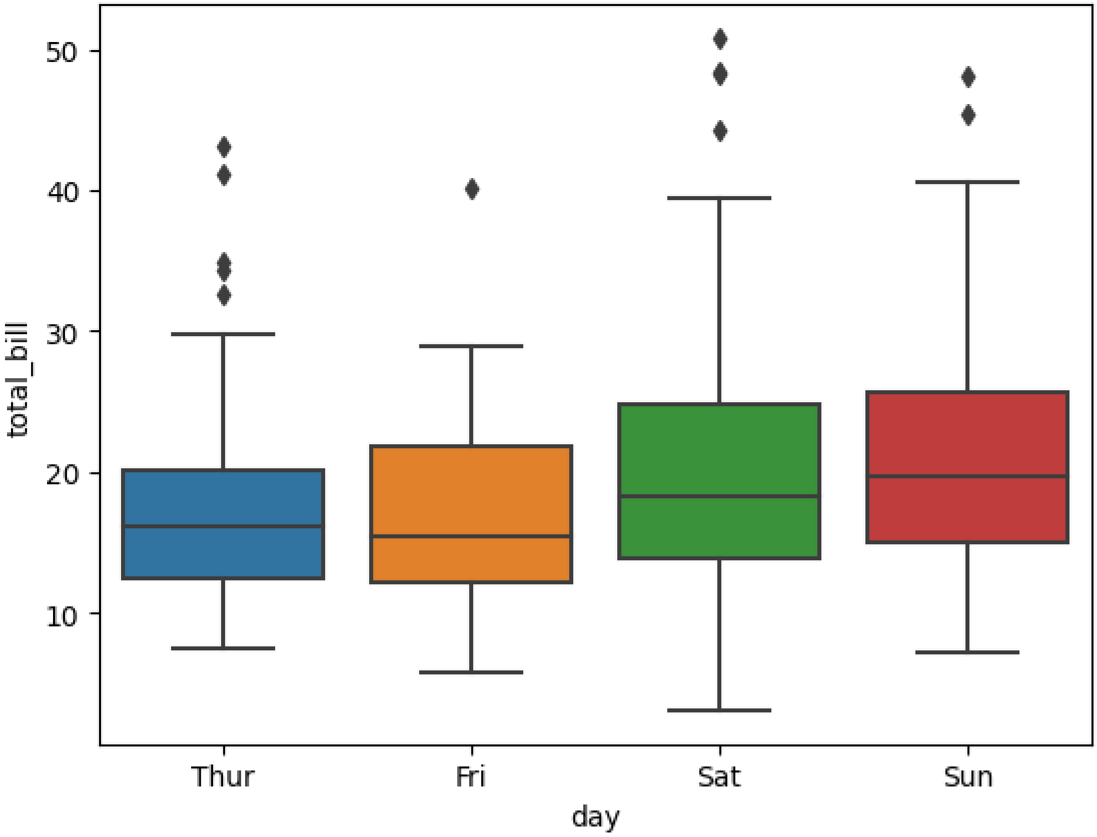
In [392]:
```
#Box Plot
tips.head(1)
```

Out[392]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2

In [393]:
```
sns.boxplot(x='day',y='total_bill', data=tips)
```

Out[393]:
```
<Axes: xlabel='day', ylabel='total_bill'>
```



In [394]:
```
# boxen plot
tips.head(1)
```

Out[394]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2