In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
#read csv file
df = pd.read_csv('archive.zip')
df
```

Out[3]:

	Passengerid	Age	Fare	Sex	sibsp	zero	zero.1	zero.2	zero.3	zero.4	...	zero.12	zero.13	zero.14	Pclass	zero.15	zero.16	Emba
0	1	22.0	7.2500	0	1	0	0	0	0	0	...	0	0	0	3	0	0	
1	2	38.0	71.2833	1	1	0	0	0	0	0	...	0	0	0	1	0	0	
2	3	26.0	7.9250	1	0	0	0	0	0	0	...	0	0	0	3	0	0	
3	4	35.0	53.1000	1	1	0	0	0	0	0	...	0	0	0	1	0	0	
4	5	35.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1304	1305	28.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	
1305	1306	39.0	108.9000	1	0	0	0	0	0	0	...	0	0	0	1	0	0	
1306	1307	38.5	7.2500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	
1307	1308	28.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	
1308	1309	28.0	22.3583	0	1	0	0	0	0	0	...	0	0	0	3	0	0	

1309 rows × 28 columns

In [4]:
```
df.head()
```

Out[4]:

	Passengerid	Age	Fare	Sex	sibsp	zero	zero.1	zero.2	zero.3	zero.4	...	zero.12	zero.13	zero.14	Pclass	zero.15	zero.16	Embarked
0	1	22.0	7.2500	0	1	0	0	0	0	0	...	0	0	0	3	0	0	2.0
1	2	38.0	71.2833	1	1	0	0	0	0	0	...	0	0	0	1	0	0	0.0
2	3	26.0	7.9250	1	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
3	4	35.0	53.1000	1	1	0	0	0	0	0	...	0	0	0	1	0	0	2.0
4	5	35.0	8.0500	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0

5 rows × 28 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 28 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Passengerid  1309 non-null   int64
 1   Age          1309 non-null   float64
 2   Fare         1309 non-null   float64
 3   Sex          1309 non-null   int64
 4   sibsp        1309 non-null   int64
 5   zero         1309 non-null   int64
 6   zero.1       1309 non-null   int64
 7   zero.2       1309 non-null   int64
 8   zero.3       1309 non-null   int64
 9   zero.4       1309 non-null   int64
 10  zero.5       1309 non-null   int64
 11  zero.6       1309 non-null   int64
 12  Parch        1309 non-null   int64
 13  zero.7       1309 non-null   int64
 14  zero.8       1309 non-null   int64
 15  zero.9       1309 non-null   int64
 16  zero.10      1309 non-null   int64
 17  zero.11      1309 non-null   int64
 18  zero.12      1309 non-null   int64
 19  zero.13      1309 non-null   int64
 20  zero.14      1309 non-null   int64
 21  Pclass       1309 non-null   int64
```

```
 22  zero.15      1309 non-null   int64
 23  zero.16      1309 non-null   int64
 24  Embarked     1307 non-null   float64
 25  zero.17      1309 non-null   int64
 26  zero.18      1309 non-null   int64
 27  2urvived     1309 non-null   int64
dtypes: float64(3), int64(25)
memory usage: 286.5 KB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Passengerid    0
Age            0
Fare           0
Sex            0
sibsp          0
zero           0
zero.1         0
zero.2         0
zero.3         0
zero.4         0
zero.5         0
zero.6         0
Parch          0
zero.7         0
zero.8         0
zero.9         0
zero.10        0
zero.11        0
zero.12        0
zero.13        0
zero.14        0
Pclass         0
zero.15        0
zero.16        0
Embarked       2
zero.17        0
zero.18        0
2urvived       0
dtype: int64
```

In [9]:
```
df = df.drop(['Passengerid'],axis=1)
```

In [10]:
```
df
```

Out[10]:

	Age	Fare	Sex	sibsp	zero	zero.1	zero.2	zero.3	zero.4	zero.5	...	zero.12	zero.13	zero.14	Pclass	zero.15	zero.16	Embarked
0	22.0	7.2500	0	1	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
1	38.0	71.2833	1	1	0	0	0	0	0	0	...	0	0	0	1	0	0	0.0
2	26.0	7.9250	1	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
3	35.0	53.1000	1	1	0	0	0	0	0	0	...	0	0	0	1	0	0	2.0
4	35.0	8.0500	0	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1304	28.0	8.0500	0	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
1305	39.0	108.9000	1	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0.0
1306	38.5	7.2500	0	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
1307	28.0	8.0500	0	0	0	0	0	0	0	0	...	0	0	0	3	0	0	2.0
1308	28.0	22.3583	0	1	0	0	0	0	0	0	...	0	0	0	3	0	0	0.0

1309 rows × 27 columns

In [11]:
```
df = df.dropna()
```

In [12]:
```
df = df.drop(["zero","zero.1","zero.2","zero.3","zero.4","zero.5","zero.6","zero.7","zero.8","zero.9","zero.10",'
                  "zero.13","zero.14","zero.15","zero.16","zero.17","zero.18"],axis=1)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1307 entries, 0 to 1308
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Age       1307 non-null   float64
 1   Fare      1307 non-null   float64
 2   Sex       1307 non-null   int64  
 3   sibsp     1307 non-null   int64  
 4   Parch     1307 non-null   int64  
 5   Pclass    1307 non-null   int64  
 6   Embarked  1307 non-null   float64
 7   2urvived  1307 non-null   int64  
dtypes: float64(3), int64(5)
memory usage: 91.9 KB
```

In [14]:
```
x = df.drop(['2urvived'],axis=1)
y = df['2urvived']
```

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

Out[15]:

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
0	22.0	7.2500	0	1	0	3	2.0
1	38.0	71.2833	1	1	0	1	0.0
2	26.0	7.9250	1	0	0	3	2.0
3	35.0	53.1000	1	1	0	1	2.0
4	35.0	8.0500	0	0	0	3	2.0
...	...	...	...	...	...	...	...
1304	28.0	8.0500	0	0	0	3	2.0
1305	39.0	108.9000	1	0	0	1	0.0
1306	38.5	7.2500	0	0	0	3	2.0
1307	28.0	8.0500	0	0	0	3	2.0
1308	28.0	22.3583	0	1	1	3	0.0

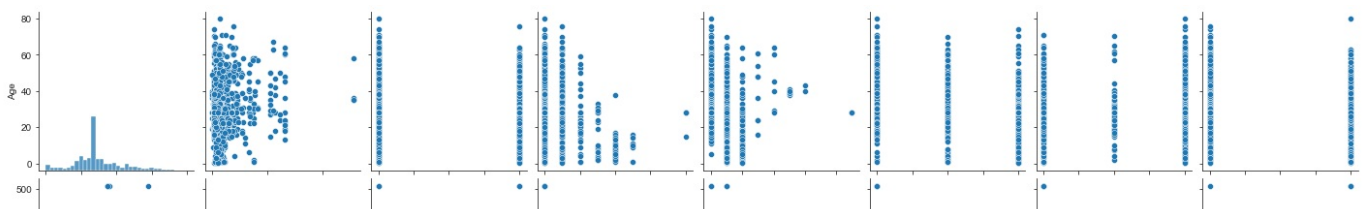
1307 rows × 7 columns

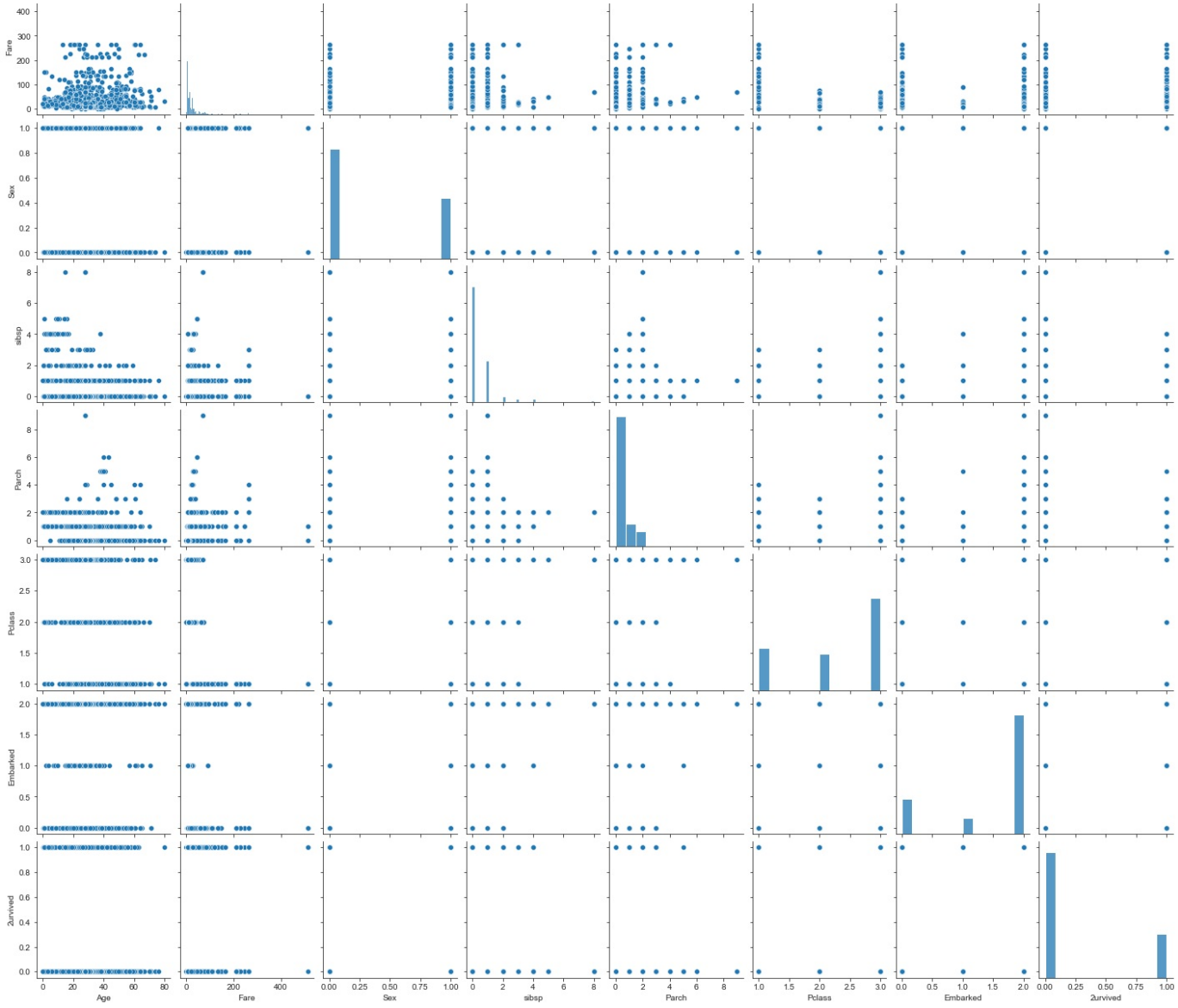
In [16]:
```
y
```

Out[16]:
```
0       0
1       1
2       1
3       1
4       0
       ..
1304    0
1305    0
1306    0
1307    0
1308    0
Name: 2urvived, Length: 1307, dtype: int64
```

To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function.

In [23]:
```
sns.pairplot(df,palette = "husl",diag_kind='hist')
sns.set_style('ticks')
plt.show()
```





Splitting the data

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [31]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1,random_state=2)
```

In [33]:
```
x_train[:3]
```

Out[33]:

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
1149	19.0	13.0000	1	0	0	2	2.0
533	28.0	22.3583	1	0	2	3	0.0
103	33.0	8.6542	0	0	0	3	2.0

Build the model

In [34]:
```
from sklearn.linear_model import LogisticRegression
```

In [35]:
```
model = LogisticRegression()
model
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
#Trainng: fit method
model.fit(x_train,y_train)
```

```
C:\Users\Nilesh koli\anaconda3\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarni
ng: lbfgs failed to 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[36]:
```
LogisticRegression()
```

In [37]:
```
model.intercept_
```

Out[37]:
```
array([0.92471061])
```

In [39]:
```
model.coef_
```

Out[39]:
```
array([[-3.41342023e-02, -2.09864842e-05,  1.82158231e+00,
        -1.84989412e-01, -5.31200351e-02, -7.71120704e-01,
         1.31400598e-02]])
```

In [41]:
```
x_test[:3]
```

Out[41]:

	Age	Fare	Sex	sibsp	Parch	Pclass	Embarked
789	46.0	79.2000	0	0	0	1	0.0
1207	57.0	146.5208	0	1	0	1	0.0
961	24.0	7.7500	1	0	0	3	1.0

In [58]:
```
#testing: predict
y_predicted = model.predict(x_test)
y_predicted
```

Out[58]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [53]:
```
#Accuracy of train and test data
model.score(x_train,y_train)*100
```

Out[53]:
```
78.06122448979592
```

In [54]:
```
model.score(x_test,y_test)*100
```

Out[54]:
```
82.44274809160305
```

In [59]:
```
model.score(x_test,y_predicted)*100
```

Out[59]:
```
100.0
```

In [60]:
```
from sklearn.metrics import accuracy_score
```

In [61]:
```
accuracy_score(y_test,y_predicted)*100
```

Out[61]: 82.44274809160305

In [ ]:
```

```