```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 4345352570553314311
 xla_global_id: -1]
```

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import numpy as np
import math
import pylab as plt
import sklearn
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import model_selection
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")




%matplotlib inline
```

```
Mounted at /content/drive
```

```
train_data = pd.read_csv('/content/drive/My Drive/kaggle-titanic/train.csv')
test_data = pd.read_csv('/content/drive/My Drive/kaggle-titanic/test.csv')
p_id = test_data['PassengerId']
data = pd.concat([train_data, test_data])
data.shape
```

```
(1309, 12)
```

```
print(train_data.isnull().any())
print()
```

```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
```

```
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

```
print(train_data.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
None
```

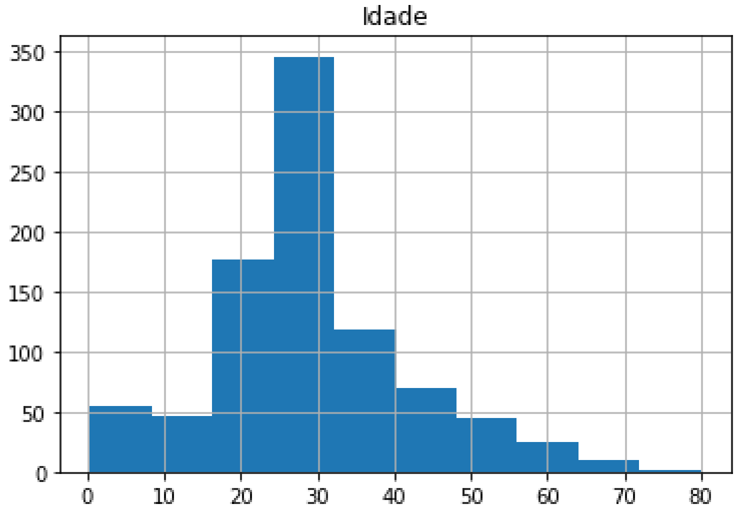
```
train_data['Cabin'].fillna('NotFClass', inplace=True)
train_data.info()
```

```
<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        891 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
media_idade = train_data['Age'].mean()
train_data['Age'].fillna(media_idade, inplace=True)
```

```
train_data['Age'].hist()
plt.title('Idade')
```

```
Text(0.5, 1.0, 'Idade')
```




```
train_data['Embarked'].fillna('S', inplace=True)
train_data.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          891 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        891 non-null    object 
 11  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
train_data.to_csv('lab2_train_no_nulls.csv', index=False)
```

```
train_data = pd.read_csv('lab2_train_no_nulls.csv')
```

```
train_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454

```
print(train_data.sort_values('Age', ascending=False).head(5)['Age'])
print(train_data.sort_values('Age', ascending=True).head(5)['Age'])
```

```
630    80.0
851    74.0
96     71.0
493    71.0
116    70.5
Name: Age, dtype: float64
803    0.42
755    0.67
644    0.75
469    0.75
831    0.83
Name: Age, dtype: float64
```

```
print(train_data.sort_values('Fare', ascending=False).head(5)['Fare'])
print(train_data.sort_values('Fare', ascending=True).head(5)['Fare'])
```

```
258    512.3292
737    512.3292
679    512.3292
88     263.0000
27     263.0000
Name: Fare, dtype: float64
271    0.0
597    0.0
302    0.0
633    0.0
277    0.0
Name: Fare, dtype: float64
```

```
train_data.to_csv('train_no_nulls_no_outliers.csv', index=False)
```

```
train_data = pd.read_csv('train_no_nulls_no_outliers.csv')
train_data.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250

```
novas_colunas = pd.get_dummies(train_data['Embarked'])
train_data = pd.concat([train_data,novas_colunas], axis=1) # axis = 1 concatena colunas. a
train_data.head(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
				Cumings,						

```
#train_data.drop('Embarked', axis=1, inplace=True)
```

```
novas_colunas_pclass = pd.get_dummies(train_data['Pclass'])
#novas_colunas_sex = pd.get_dummies(train_data['Sex'])

#train_data = pd.concat([train_data,novas_colunas_pclass, novas_colunas_sex], axis=1)
train_data = pd.concat([train_data,novas_colunas_pclass], axis=1)
#train_data.drop(['Pclass', 'Sex'], axis=1, inplace=True)
train_data.head(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
				Cumings,						

```
train_data.to_csv('train_no_nulls_no_outliers_ohe.csv', index=False)
```

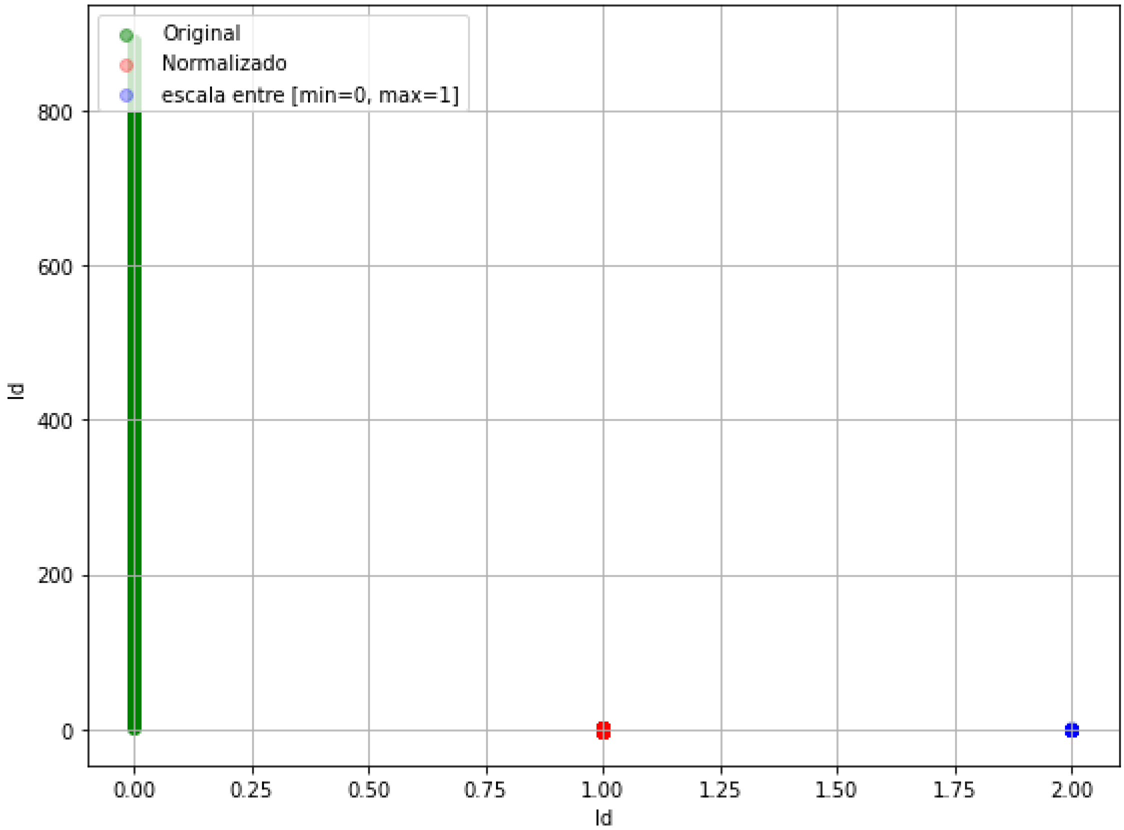
```
train_data.to_csv('train_no_nulls_no_outliers_feat_hash.csv', index=False)
```

```
train_data = pd.read_csv('train_no_nulls_no_outliers_feat_hash.csv')
train_data.head(2)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250

```
from sklearn import preprocessing

## dados originais
ID_original = train_data['PassengerId'].values.reshape(-1, 1)
## Normaliza os dados
ID_standard = preprocessing.StandardScaler().fit_transform(train_data['PassengerId'].value
## Muda a escala dos dados para valores entre 0 e 1 (valores padrão, que poderiam ser pers
ID_minmax = preprocessing.MinMaxScaler().fit_transform(train_data['PassengerId'].values.re
```

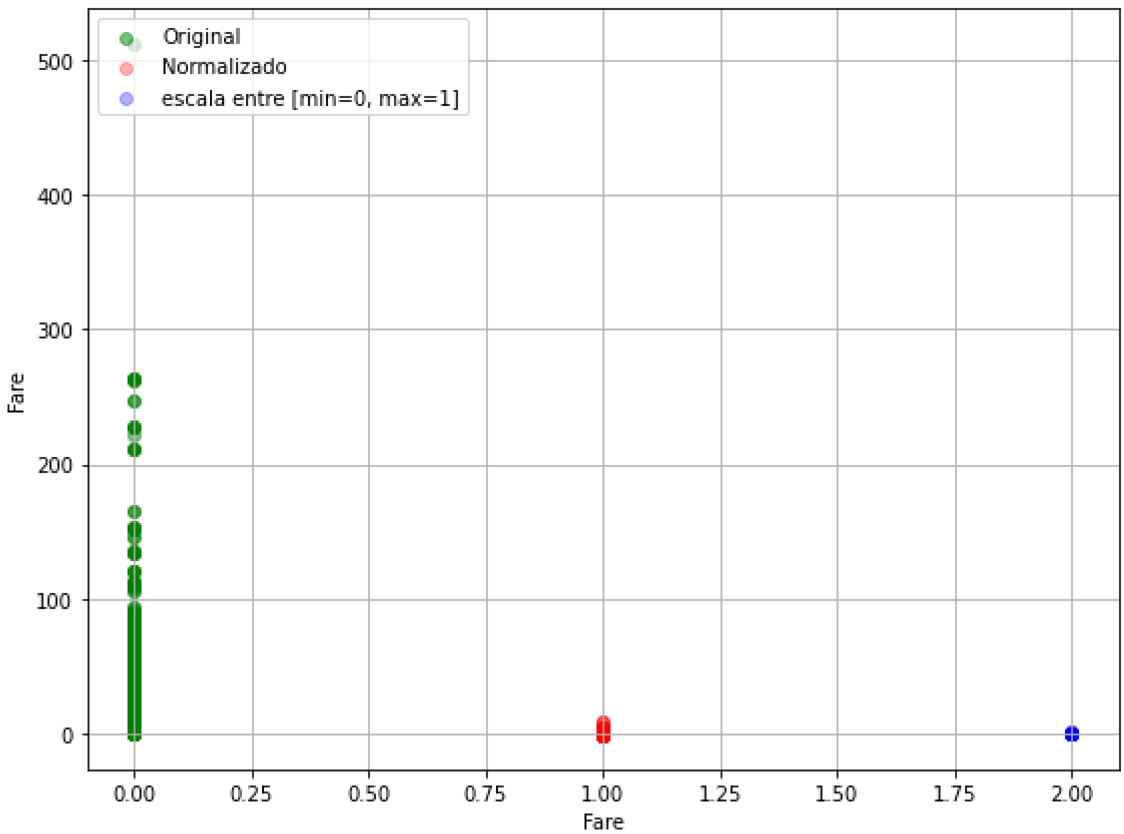
```
from matplotlib import pyplot as plt

def plot():
    plt.figure(figsize=(8,6))

    plt.scatter([0]*len(ID_original), ID_original,
            color='green', label='Original', alpha=0.5)

    plt.scatter([1]*len(ID_standard), ID_standard, color='red',
            label='Normalizado', alpha=0.3)

    plt.scatter([2]*len(ID_minmax), ID_minmax,
            color='blue', label='escala entre [min=0, max=1]', alpha=0.3)

    plt.xlabel('Id')
    plt.ylabel('Id')
    plt.legend(loc='upper left')
    plt.grid()

    plt.tight_layout()

plot()
plt.show()
```




```
## dados originais
Fare_original = train_data['Fare'].values.reshape(-1, 1)
## Normaliza os dados
Fare_standard = preprocessing.StandardScaler().fit_transform(train_data['Fare'].values.res
```

```
## Muda a escala dos dados para valores entre 0 e 1 (valores padrão, que poderiam ser pers
Fare_minmax = preprocessing.MinMaxScaler().fit_transform(train_data['Fare'].values.reshape
```

```
from matplotlib import pyplot as plt

def plot():
    plt.figure(figsize=(8,6))

    plt.scatter([0]*len(Fare_original), Fare_original,
            color='green', label='Original', alpha=0.5)

    plt.scatter([1]*len(Fare_original), Fare_standard, color='red',
            label='Normalizado', alpha=0.3)

    plt.scatter([2]*len(Fare_original), Fare_minmax,
            color='blue', label='escala entre [min=0, max=1]', alpha=0.3)

    plt.xlabel('Fare')
    plt.ylabel('Fare')
    plt.legend(loc='upper left')
    plt.grid()

    plt.tight_layout()

plot()
plt.show()
```



Pequeno tratamento

```
data1 = train_data.copy(deep = True)
```

```
data_cleaner = [data1, test_data]

for dataset in data_cleaner:
    #complete missing age with median
    dataset['Age'].fillna(dataset['Age'].median(), inplace = True)

    #complete embarked with mode
    dataset['Embarked'].fillna(dataset['Embarked'].mode()[0], inplace = True)

    #complete missing fare with median
    dataset['Fare'].fillna(dataset['Fare'].median(), inplace = True)

#delete the cabin feature/column and others previously stated to exclude in train dataset
drop_column = ['PassengerId','Cabin', 'Ticket']
data1.drop(drop_column, axis=1, inplace = True)

print(data1.isnull().sum())
print("-"*10)
print(test_data.isnull().sum())
```

```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
C           0
Q           0
S           0
1           0
2           0
3           0
dtype: int64
----------
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

Busca por Titulos

```
for dataset in data_cleaner:
    #Discrete variables
    dataset['FamilySize'] = dataset ['SibSp'] + dataset['Parch'] + 1

    dataset['IsAlone'] = 1 #initialize to yes/1 is alone
    dataset['IsAlone'].loc[dataset['FamilySize'] > 1] = 0 # now update to no/0 if family s

    #quick and dirty code split title from name: http://www.pythonforbeginners.com/diction
    dataset['Title'] = dataset['Name'].str.split(", ", expand=True)[1].str.split(".", expa


    #Continuous variable bins; qcut vs cut: https://stackoverflow.com/questions/30211923/w
    #Fare Bins/Buckets using qcut or frequency bins: https://pandas.pydata.org/pandas-docs
    dataset['FareBin'] = pd.qcut(dataset['Fare'], 4)

    #Age Bins/Buckets using cut or value bins: https://pandas.pydata.org/pandas-docs/stabl
    dataset['AgeBin'] = pd.cut(dataset['Age'].astype(int), 5)



#cleanup rare title names
#print(data1['Title'].value_counts())
stat_min = 10 #while small is arbitrary, we'll use the common minimum in statistics: http:
title_names = (data1['Title'].value_counts() < stat_min) #this will create a true false se

#apply and lambda functions are quick and dirty code to find and replace with fewer lines
data1['Title'] = data1['Title'].apply(lambda x: 'Misc' if title_names.loc[x] == True else
print(data1['Title'].value_counts())
print("-"*10)


#preview data again
data1.info()
test_data.info()
data1.sample(10)
```

```
 8   Embarked    891 non-null    object
 9   C           891 non-null    int64
 10  Q           891 non-null    int64
 11  S           891 non-null    int64
 12  1           891 non-null    int64
 13  2           891 non-null    int64
 14  3           891 non-null    int64
 15  FamilySize  891 non-null    int64
 16  IsAlone     891 non-null    int64
 17  Title       891 non-null    object
 18  FareBin     891 non-null    category
 19  AgeBin      891 non-null    category
dtypes: category(2), float64(2), int64(12), object(4)
memory usage: 127.6+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          418 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         418 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
 11  FamilySize   418 non-null    int64
 12  IsAlone      418 non-null    int64
 13  Title        418 non-null    object
 14  FareBin      418 non-null    category
 15  AgeBin       418 non-null    category
dtypes: category(2), float64(2), int64(6), object(6)
memory usage: 47.1+ KB
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	C
796	1	1	Leader, Dr. Alice (Farnham)	female	49.0	0	0	25.9292	S	0
344	0	2	Fox, Mr. Stanley Hubert	male	36.0	0	0	13.0000	S	0
751	1	3	Moor, Master. Meier	male	6.0	0	1	12.4750	S	0
233	1	3	Asplund, Miss. Lillian Gertrud	female	5.0	4	2	31.3875	S	0
550	1	1	Thayer, Mr. John Borland Jr	male	17.0	0	2	110.8833	C	1

Tratamento dos dados para facilitar analise

```
label = LabelEncoder()
for dataset in data_cleaner:
    dataset['Sex_Code'] = label.fit_transform(dataset['Sex'])
    dataset['Embarked_Code'] = label.fit_transform(dataset['Embarked'])
    dataset['Title_Code'] = label.fit_transform(dataset['Title'])
    dataset['AgeBin_Code'] = label.fit_transform(dataset['AgeBin'])
    dataset['FareBin_Code'] = label.fit_transform(dataset['FareBin'])


#define y variable aka target/outcome
Target = ['Survived']

#define x variables for original features aka feature selection
data1_x = ['Sex','Pclass', 'Embarked', 'Title','SibSp', 'Parch', 'Age', 'Fare', 'FamilySiz
data1_x_calc = ['Sex_Code','Pclass', 'Embarked_Code', 'Title_Code','SibSp', 'Parch', 'Age'
data1_xy =  Target + data1_x
print('Original X Y: ', data1_xy, '\n')


#define x variables for original w/bin features to remove continuous variables
data1_x_bin = ['Sex_Code','Pclass', 'Embarked_Code', 'Title_Code', 'FamilySize', 'AgeBin_C
data1_xy_bin = Target + data1_x_bin
print('Bin X Y: ', data1_xy_bin, '\n')


#define x and y variables for dummy features original
data1_dummy = pd.get_dummies(data1[data1_x])
data1_x_dummy = data1_dummy.columns.tolist()
data1_xy_dummy = Target + data1_x_dummy
print('Dummy X Y: ', data1_xy_dummy, '\n')



data1_dummy.head()
```

```
Original X Y:  ['Survived', 'Sex', 'Pclass', 'Embarked', 'Title', 'SibSp', 'Parch',

Bin X Y:  ['Survived', 'Sex_Code', 'Pclass', 'Embarked_Code', 'Title_Code', 'FamilyS:

Dummy X Y:  ['Survived', 'Pclass', 'SibSp', 'Parch', 'Age', 'Fare', 'FamilySize', 'Is
```

	Pclass	SibSp	Parch	Age	Fare	FamilySize	IsAlone	Sex_female	Sex_male	Emb
0	3	1	0	22.0	7.2500	2	0	0	1	
1	1	1	0	38.0	71.2833	2	0	1	0	
2	3	0	0	26.0	7.9250	1	1	1	0	
3	1	1	0	35.0	53.1000	2	0	1	0	
4	3	0	0	35.0	8.0500	1	1	0	1	

```
print('Train columns with null values: \n', data1.isnull().sum())
```

```
print("-"*10)
print (data1.info())
print("-"*10)

print('Test/Validation columns with null values: \n', test_data.isnull().sum())
print("-"*10)
print (test_data.info())
print("-"*10)

train_data.describe(include = 'all')
```

```
SibSp            0
Parch            0
Fare             0
Embarked         0
C                0
Q                0
S                0
1                0
2                0
3                0
FamilySize       0
IsAlone          0
Title            0
FareBin          0
AgeBin           0
Sex_Code         0
Embarked_Code    0
Title_Code       0
AgeBin_Code      0
FareBin_Code     0
dtype: int64
----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 25 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Survived       891 non-null    int64
 1   Pclass         891 non-null    int64
 2   Name           891 non-null    object
 3   Sex            891 non-null    object
 4   Age            891 non-null    float64
 5   SibSp          891 non-null    int64
 6   Parch          891 non-null    int64
 7   Fare           891 non-null    float64
 8   Embarked       891 non-null    object
 9   C              891 non-null    int64
 10  Q              891 non-null    int64
 11  S              891 non-null    int64
 12  1              891 non-null    int64
 13  2              891 non-null    int64
 14  3              891 non-null    int64
 15  FamilySize     891 non-null    int64
 16  IsAlone        891 non-null    int64
 17  Title          891 non-null    object
 18  FareBin        891 non-null    category
 19  AgeBin         891 non-null    category
 20  Sex_Code       891 non-null    int64
 21  Embarked_Code  891 non-null    int64
 22  Title_Code     891 non-null    int64
 23  AgeBin_Code    891 non-null    int64
 24  FareBin_Code   891 non-null    int64
dtypes: category(2), float64(2), int64(17), object(4)
memory usage: 162.4+ KB
None
----------
Test/Validation columns with null values:
 PassengerId       0
Pclass            0
Name              0
```

```
Sex                0
Age                0
SibSp              0
Parch              0
Ticket             0
Fare               0
Cabin            327
Embarked           0
FamilySize         0
IsAlone            0
Title              0
FareBin            0
AgeBin             0
Sex_Code           0
Embarked_Code      0
Title_Code         0
AgeBin_Code        0
FareBin_Code       0
dtype: int64
----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PassengerId    418 non-null    int64
 1   Pclass         418 non-null    int64
 2   Name           418 non-null    object
 3   Sex            418 non-null    object
 4   Age            418 non-null    float64
 5   SibSp          418 non-null    int64
```

Divisão dos conjuntos de dados

```
 9   Cabin          91 non-null     object
```

```
train1_x, test1_x, train1_y, test1_y = model_selection.train_test_split(data1[data1_x_calc
train1_x_bin, test1_x_bin, train1_y_bin, test1_y_bin = model_selection.train_test_split(da
train1_x_dummy, test1_x_dummy, train1_y_dummy, test1_y_dummy = model_selection.train_test_


print("Data1 Shape: {}".format(data1.shape))
print("Train1 Shape: {}".format(train1_x.shape))
print("Test1 Shape: {}".format(test1_x.shape))

train1_x_bin.head()
```

```
Data1 Shape: (891, 25)
Train1 Shape: (668, 8)
```

Padrões Interessantes

```
105      1      3           2          3          1          1
```

Alguns resultados

Correlaçao entre sobreviver e demais atributos

```
706      0      2           2          4          1          2
```

```
for x in data1_x:
    if data1[x].dtype != 'float64' :
        print('Survival Correlation by:', x)
        print(data1[[x, Target[0]]].groupby(x, as_index=False).mean())
        print('-'*10, '\n')


#using crosstabs: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.crosstab.h
print(pd.crosstab(data1['Title'],data1[Target[0]]))
```

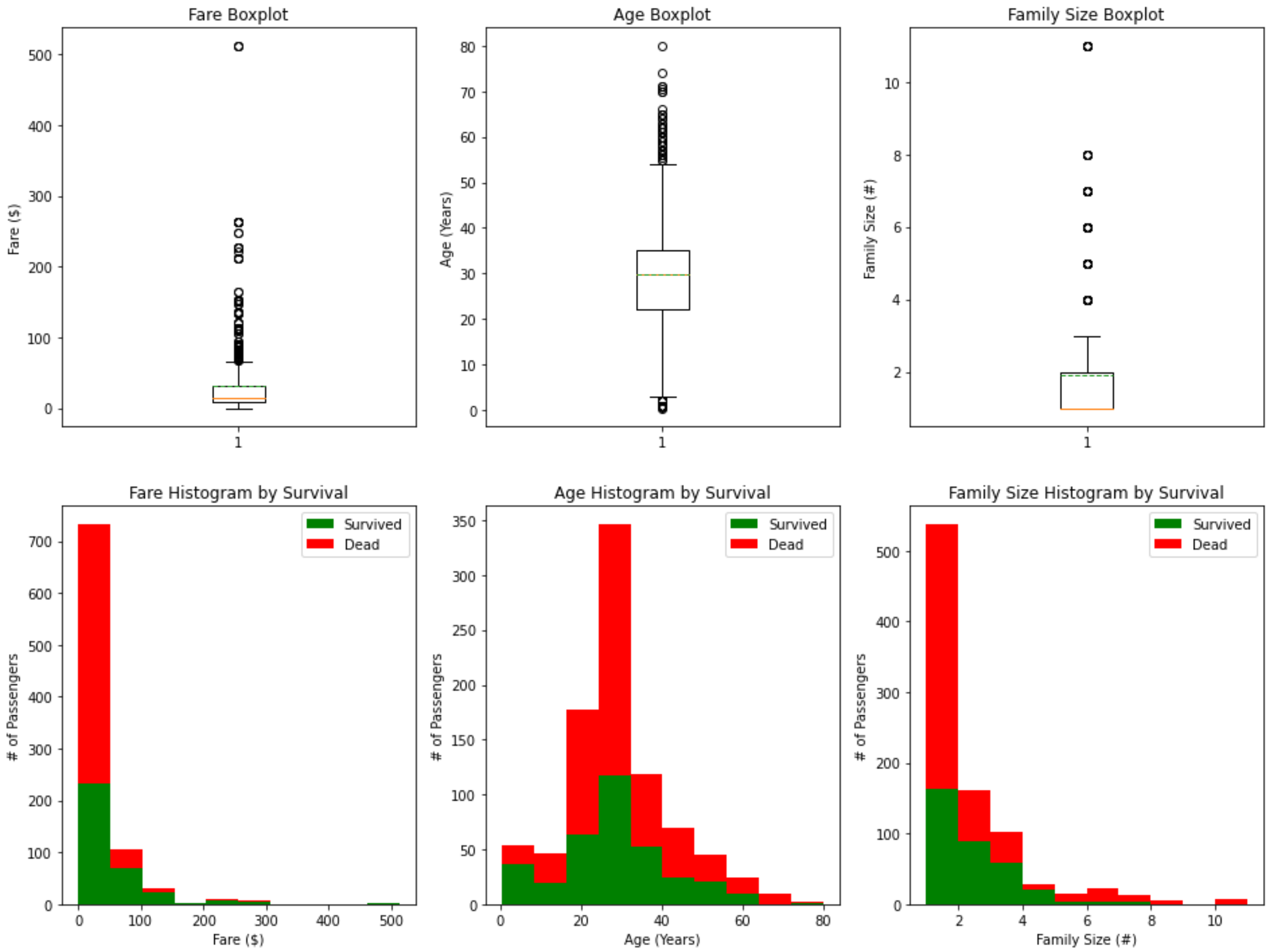
```
Survival Correlation by: Title
    Title  Survived
0  Master  0.575000
1    Misc  0.444444
2    Miss  0.697802
3      Mr  0.156673
4     Mrs  0.792000
----------

Survival Correlation by: SibSp
   SibSp  Survived
0      0  0.345395
1      1  0.535885
2      2  0.464286

3      3  0.250000
4      4  0.166667
5      5  0.000000
6      8  0.000000
----------

Survival Correlation by: Parch
   Parch  Survived
0      0  0.343658
1      1  0.550847
2      2  0.500000
3      3  0.600000
4      4  0.000000
5      5  0.200000
6      6  0.000000
----------

Survival Correlation by: FamilySize
   FamilySize  Survived
0           1  0.303538
```

```
0           1  0.303538
1           2  0.552795
2           3  0.578431
3           4  0.724138
4           5  0.200000
5           6  0.136364
6           7  0.333333
7           8  0.000000
8          11  0.000000
----------

Survival Correlation by: IsAlone
   IsAlone  Survived
0        0  0.505650
1        1  0.303538
----------

Survived    0    1
Title
Master     17   23
Misc       15   12
Miss       55  127
Mr        436   81
Mrs        26   99
```

```
plt.figure(figsize=[16,12])

plt.subplot(231)
plt.boxplot(x=data1['Fare'], showmeans = True, meanline = True)
plt.title('Fare Boxplot')
plt.ylabel('Fare ($)')

plt.subplot(232)
plt.boxplot(data1['Age'], showmeans = True, meanline = True)
plt.title('Age Boxplot')
plt.ylabel('Age (Years)')

plt.subplot(233)
plt.boxplot(data1['FamilySize'], showmeans = True, meanline = True)
plt.title('Family Size Boxplot')
plt.ylabel('Family Size (#)')

plt.subplot(234)
plt.hist(x = [data1[data1['Survived']==1]['Fare'], data1[data1['Survived']==0]['Fare']],
         stacked=True, color = ['g','r'],label = ['Survived','Dead'])
plt.title('Fare Histogram by Survival')
plt.xlabel('Fare ($)')
plt.ylabel('# of Passengers')
plt.legend()

plt.subplot(235)
plt.hist(x = [data1[data1['Survived']==1]['Age'], data1[data1['Survived']==0]['Age']],
         stacked=True, color = ['g','r'],label = ['Survived','Dead'])
plt.title('Age Histogram by Survival')
plt.xlabel('Age (Years)')
plt.ylabel('# of Passengers')
plt.legend()
```

```
plt.subplot(236)
plt.hist(x = [data1[data1['Survived']==1]['FamilySize'], data1[data1['Survived']==0]['Fami
         stacked=True, color = ['g','r'],label = ['Survived','Dead'])
plt.title('Family Size Histogram by Survival')
plt.xlabel('Family Size (#)')
plt.ylabel('# of Passengers')
plt.legend()
```

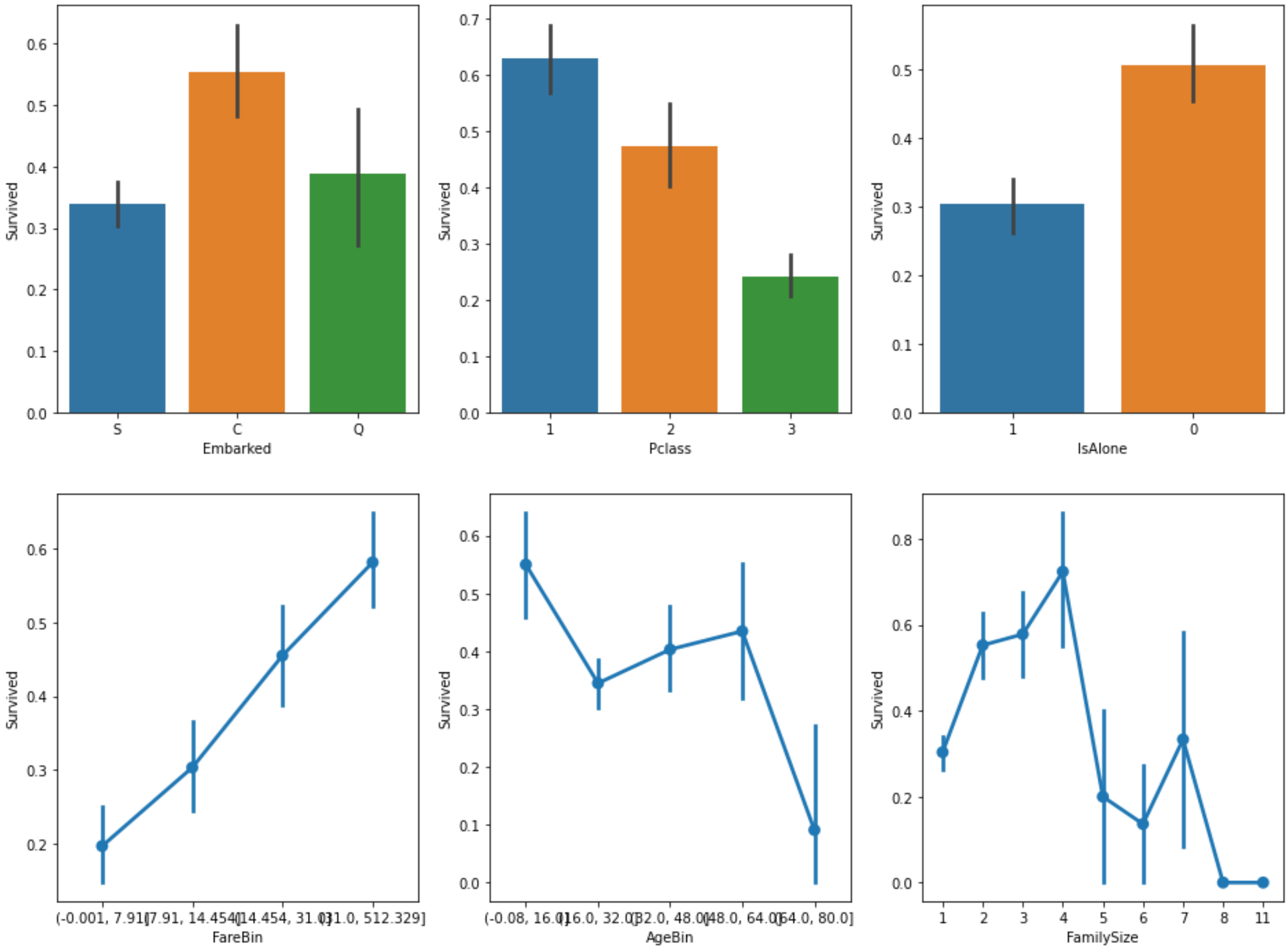
```
<matplotlib.legend.Legend at 0x7f6b3533ba90>
```




```
#we will use seaborn graphics for multi-variable comparison: https://seaborn.pydata.org/ap

#graph individual features by survival
fig, saxis = plt.subplots(2, 3,figsize=(16,12))

sns.barplot(x = 'Embarked', y = 'Survived', data=data1, ax = saxis[0,0])
```

```
sns.barplot(x = 'Pclass', y = 'Survived', order=[1,2,3], data=data1, ax = saxis[0,1])
sns.barplot(x = 'IsAlone', y = 'Survived', order=[1,0], data=data1, ax = saxis[0,2])

sns.pointplot(x = 'FareBin', y = 'Survived',  data=data1, ax = saxis[1,0])
sns.pointplot(x = 'AgeBin', y = 'Survived',  data=data1, ax = saxis[1,1])
sns.pointplot(x = 'FamilySize', y = 'Survived', data=data1, ax = saxis[1,2])
```

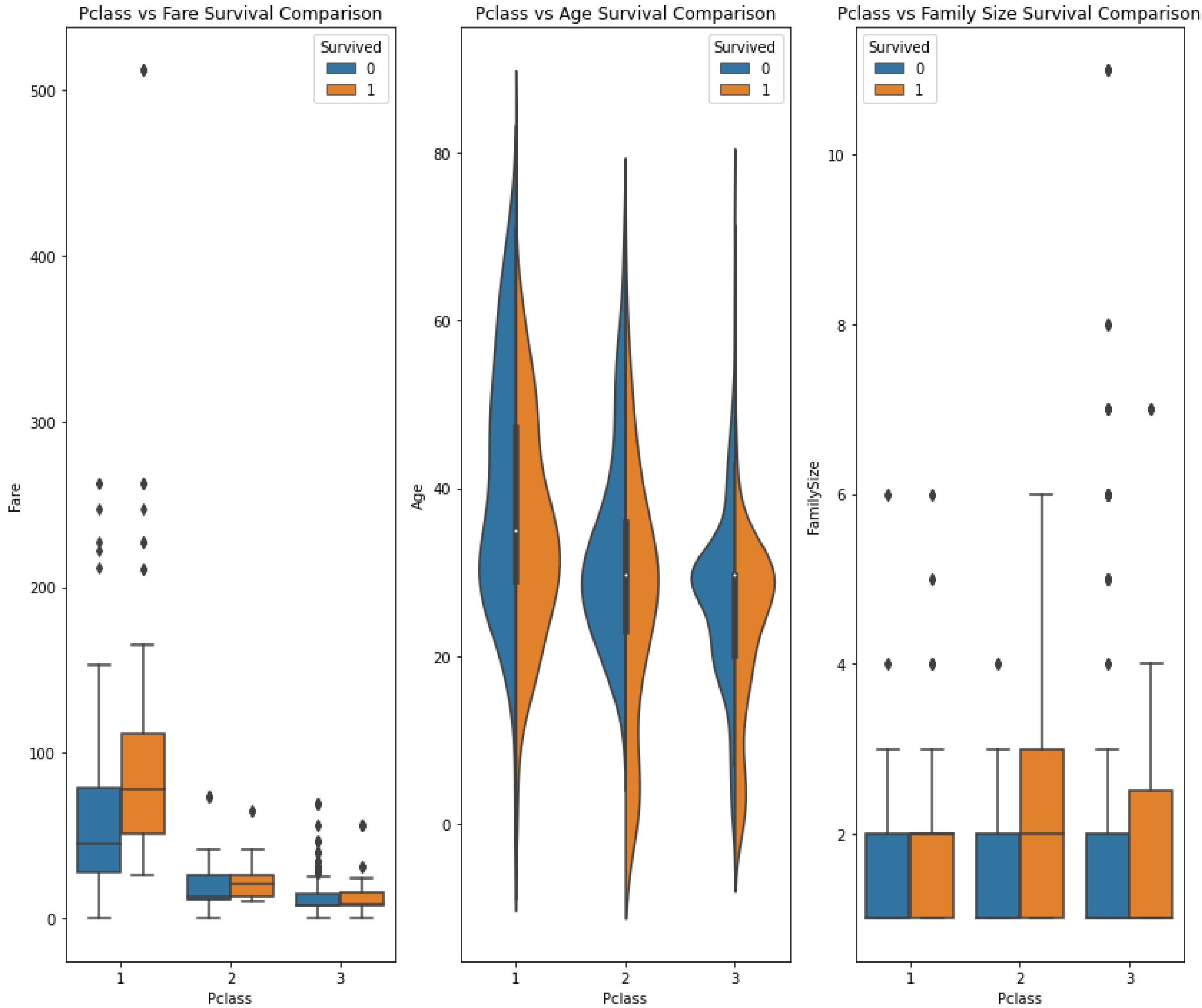
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6b3509af50>
```




```
#graph distribution of qualitative data: Pclass
#we know class mattered in survival, now let's compare class and a 2nd feature
fig, (axis1,axis2,axis3) = plt.subplots(1,3,figsize=(14,12))

sns.boxplot(x = 'Pclass', y = 'Fare', hue = 'Survived', data = data1, ax = axis1)
axis1.set_title('Pclass vs Fare Survival Comparison')

sns.violinplot(x = 'Pclass', y = 'Age', hue = 'Survived', data = data1, split = True, ax =
```

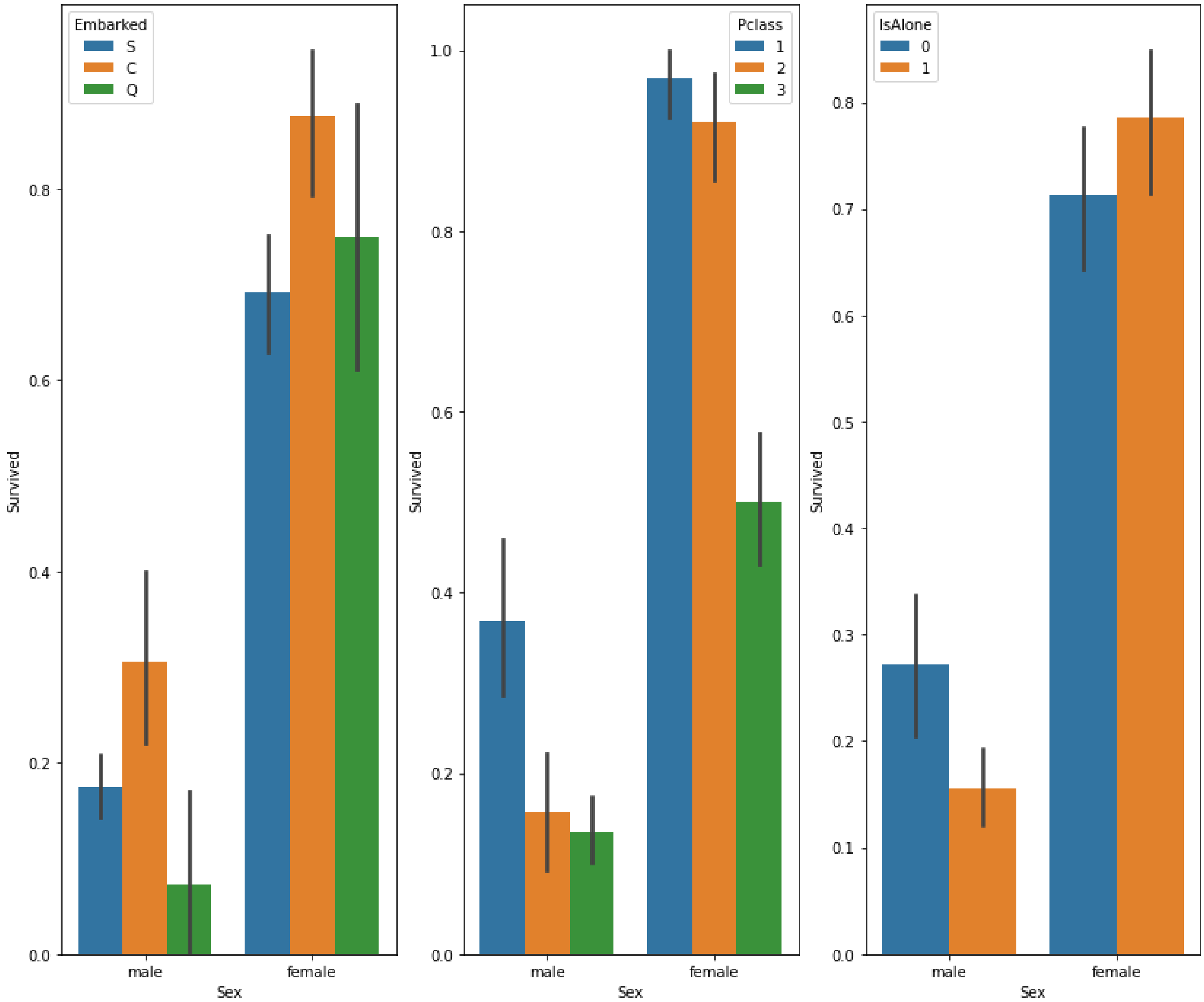
```
axis2.set_title('Pclass vs Age Survival Comparison')

sns.boxplot(x = 'Pclass', y ='FamilySize', hue = 'Survived', data = data1, ax = axis3)
axis3.set_title('Pclass vs Family Size Survival Comparison')
```

```
Text(0.5, 1.0, 'Pclass vs Family Size Survival Comparison')
```



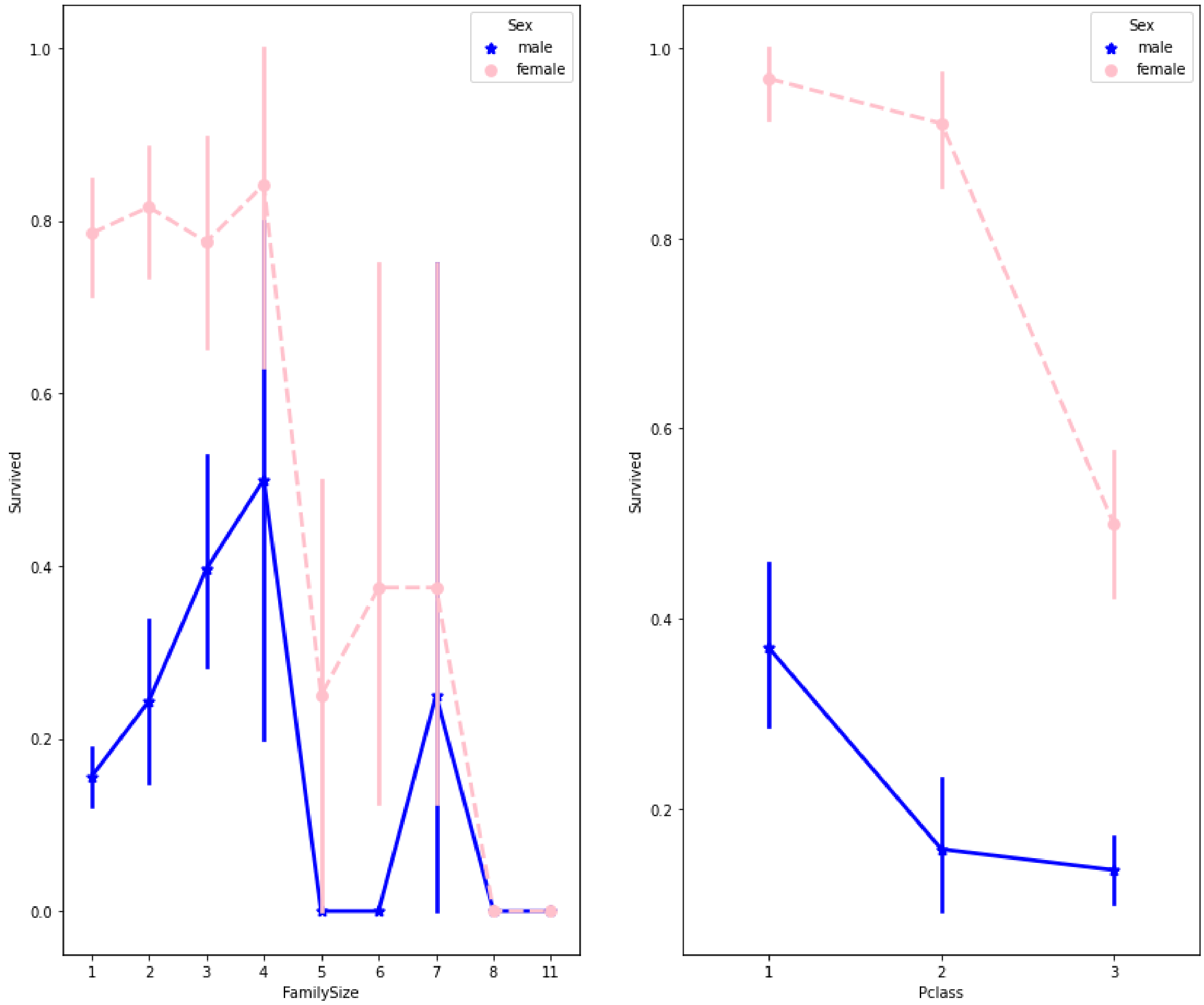

```
fig, qaxis = plt.subplots(1,3,figsize=(14,12))

sns.barplot(x = 'Sex', y = 'Survived', hue = 'Embarked', data=data1, ax = qaxis[0])
axis1.set_title('Sex vs Embarked Survival Comparison')

sns.barplot(x = 'Sex', y = 'Survived', hue = 'Pclass', data=data1, ax  = qaxis[1])
axis1.set_title('Sex vs Pclass Survival Comparison')

sns.barplot(x = 'Sex', y = 'Survived', hue = 'IsAlone', data=data1, ax  = qaxis[2])
axis1.set_title('Sex vs IsAlone Survival Comparison')
```

```
Text(0.5, 1.0, 'Sex vs IsAlone Survival Comparison')
```



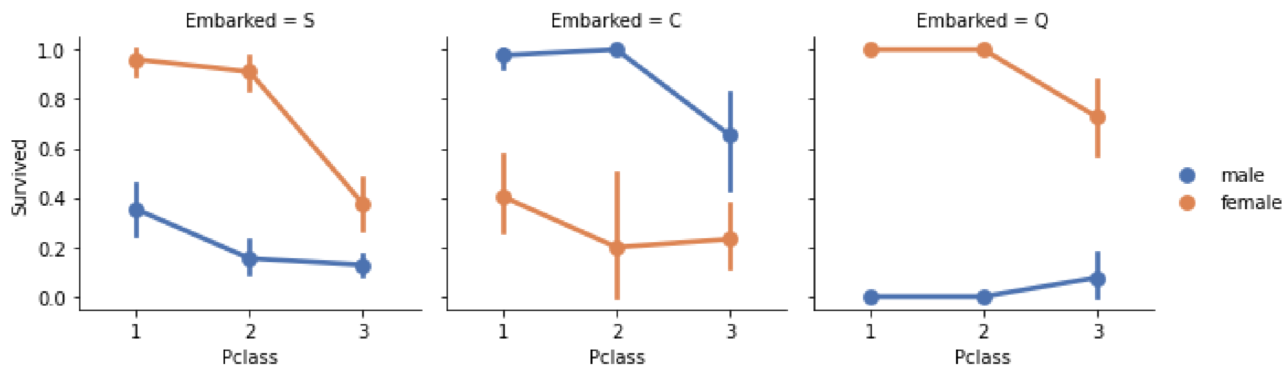

```
fig, (maxis1, maxis2) = plt.subplots(1, 2,figsize=(14,12))

#how does family size factor with sex & survival compare
sns.pointplot(x="FamilySize", y="Survived", hue="Sex", data=data1,
              palette={"male": "blue", "female": "pink"},
              markers=["*", "o"], linestyles=["-", "--"], ax = maxis1)

#how does class factor with sex & survival compare
sns.pointplot(x="Pclass", y="Survived", hue="Sex", data=data1,
              palette={"male": "blue", "female": "pink"},
              markers=["*", "o"], linestyles=["-", "--"], ax = maxis2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6b34c76710>
```



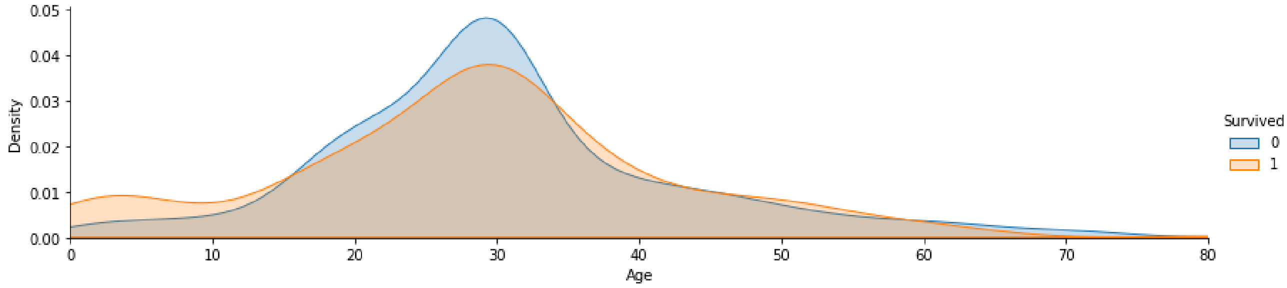

```
#how does embark port factor with class, sex, and survival compare
#facetgrid: https://seaborn.pydata.org/generated/seaborn.FacetGrid.html
e = sns.FacetGrid(data1, col = 'Embarked')
e.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', ci=95.0, palette = 'deep')
e.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6b34b12950>
```



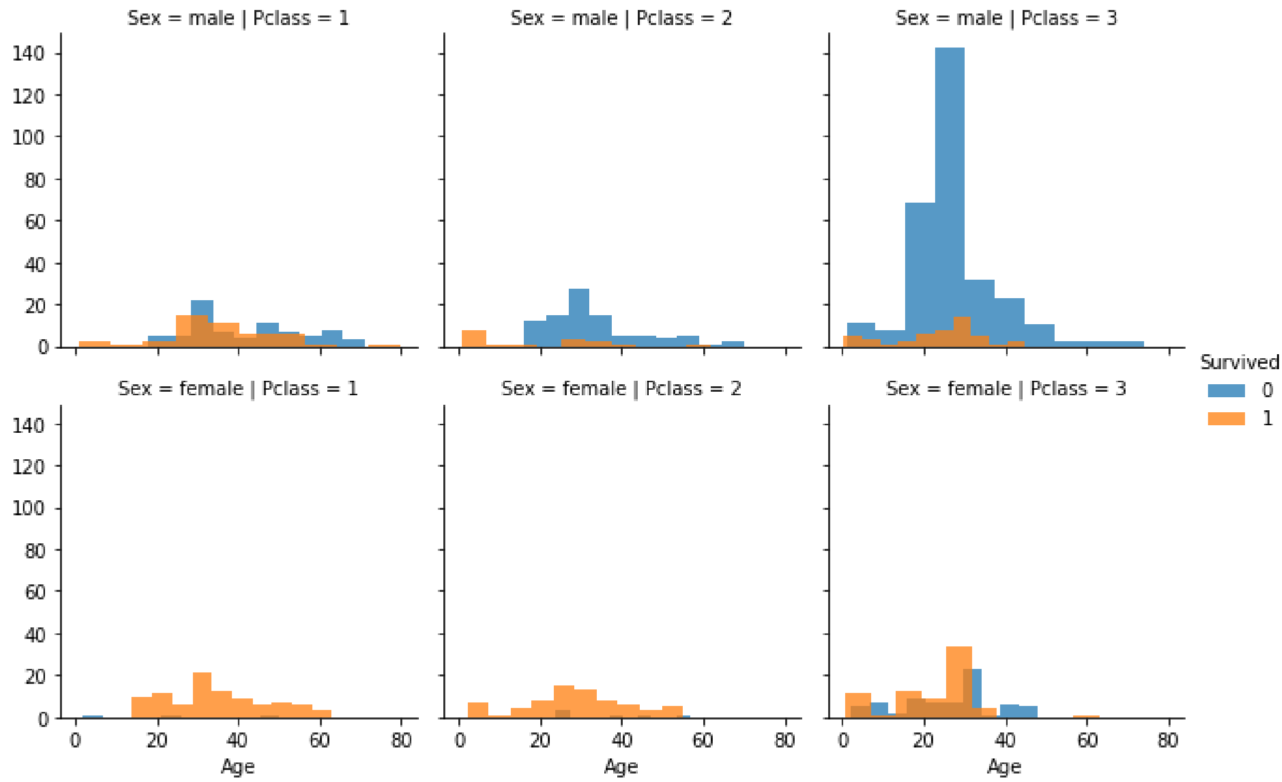
```
a = sns.FacetGrid( data1, hue = 'Survived', aspect=4 )
a.map(sns.kdeplot, 'Age', shade= True )
a.set(xlim=(0 , data1['Age'].max()))
a.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6b34b12990>
```




```
#histogram comparison of sex, class, and age by survival
h = sns.FacetGrid(data1, row = 'Sex', col = 'Pclass', hue = 'Survived')
h.map(plt.hist, 'Age', alpha = .75)
h.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6b34936510>
```



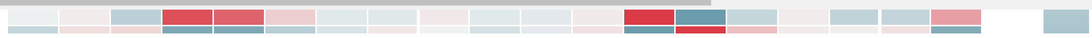
Mapa de calor da Correlação

```
def correlation_heatmap(df):
```

```
    _ , ax = plt.subplots(figsize =(14, 12))
    colormap = sns.diverging_palette(220, 10, as_cmap = True)

    _ = sns.heatmap(
        df.corr(),
        cmap = colormap,
        square=True,
        cbar_kws={'shrink':.9 },
        ax=ax,
        annot=True,
        linewidths=0.1,vmax=1.0, linecolor='white',
        annot_kws={'fontsize':12 }
    )

    plt.title('Pearson Correlation of Features', y=1.05, size=15)

correlation_heatmap(data1)
```


```
oht = TransactionEncoder()
oht_ary = oht.fit(dataset).transform(dataset)
df = pd.DataFrame(oht_ary, columns=oht.columns_)
train_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs	female	38.0	1	0	PC 17599	71.



Apriori


```
nominal_cols = ['Embarked','Pclass','Age', 'Survived', 'Sex']
cat_cols = ['Embarked','Pclass','Age', 'Survived', 'Title']
train_data['Title'] = train_data.Name.str.extract('\, ([A-Z][^ ]*\.)',expand=False)
train_data['Title'].fillna('Title_UK', inplace=True)
train_data['Embarked'].fillna('Unknown',inplace=True)
train_data['Age'].fillna(0, inplace=True)
# Replacing Binary with String
rep = {0: "Dead", 1: "Survived"}
train_data.replace({'Survived' : rep}, inplace=True)


def binning(col, cut_points, labels=None):
  minval = col.min()
  maxval = col.max()
  break_points = [minval] + cut_points + [maxval]
  if not labels:
    labels = range(len(cut_points)+1)
  colBin = pd.cut(col,bins=break_points,labels=labels,include_lowest=True)
  return colBin

cut_points = [1, 20, 50 ]
labels = ["Unknown", "Young", "Adult", "Old"]
train_data['Age'] = binning(train_data['Age'], cut_points, labels)
in_titanic = train_data[nominal_cols]
cat_titanic = train_data[cat_cols]


in_titanic.head()
```

	Embarked	Pclass	Age	Survived	Sex
0	S	3	Adult	Dead	male
1	C	1	Adult	Survived	female
2	S	3	Adult	Survived	female

```
cat_titanic.head()
```

	Embarked	Pclass	Age	Survived	Title
0	S	3	Adult	Dead	Mr.
1	C	1	Adult	Survived	Mrs.
2	S	3	Adult	Survived	Miss.
3	S	1	Adult	Survived	Mrs.
4	S	3	Adult	Dead	Mr.

```
dataset = []
for i in range(0, in_titanic.shape[0]-1):
    dataset.append([str(in_titanic.values[i,j]) for j in range(0, in_titanic.shape[1])])
# dataset = in_titanic.to_xarray()

oht = TransactionEncoder()
oht_ary = oht.fit(dataset).transform(dataset)
df = pd.DataFrame(oht_ary, columns=oht.columns_)
df.head()
```

	1	2	3	Adult	C	Dead	Old	Q	S	Survived	Unknown	Youn
0	False	False	True	True	False	True	False	False	True	False	False	Fals
1	True	False	False	True	True	False	False	False	False	True	False	Fals
2	False	False	True	True	False	False	False	False	True	True	False	Fals
3	True	False	False	True	False	False	False	False	True	True	False	Fals
4	False	False	True	True	False	True	False	False	True	False	False	Fals

```
oht.columns_
```

```
['1',
 '2',
 '3',
 'Adult',
 'C',
 'Dead',
 'Old',
 'Q',
 'S',
 'Survived',
```

```
        'Unknown',
        'Young',
        'female',
        'male']

output = apriori(df, min_support=0.2, use_colnames=oht.columns_)
output.head()
```

	support	itemsets
0	0.242697	(1)
1	0.206742	(2)
2	0.550562	(3)
3	0.726966	(Adult)
4	0.615730	(Dead)

```
config = [
    ('antecedent support', 0.7),
    ('support', 0.5),
    ('confidence', 0.8),
    ('conviction', 3)
]

for metric_type, th in config:
    rules = association_rules(output, metric=metric_type, min_threshold=th)
    if rules.empty:
        print ('Empty Data Frame For Metric Type : ',metric_type,' on Threshold : ',th)
        continue
    print (rules.columns.values)
    print ('-------------------------------------')
    print ('Configuration : ', metric_type, ' : ', th)
    print ('-------------------------------------')
    print (rules)

    #support=rules.to_numpy(columns=['support'])
    #confidence=rules.to_numpy(columns=['confidence'])

    support=rules['support'].to_numpy()
    confidence=rules['confidence'].to_numpy()

    plt.scatter(support, confidence, edgecolors='red')
    plt.xlabel('support')
    plt.ylabel('confidence')
    plt.title(metric_type+' : '+str(th))
    plt.show()
```