titanic

September 9, 2020

1 Titanic: Machine Learning from Disaster

[1]:
```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib import style
```

1.1 Parsing and Viewing Data

1.1.1 Training Set

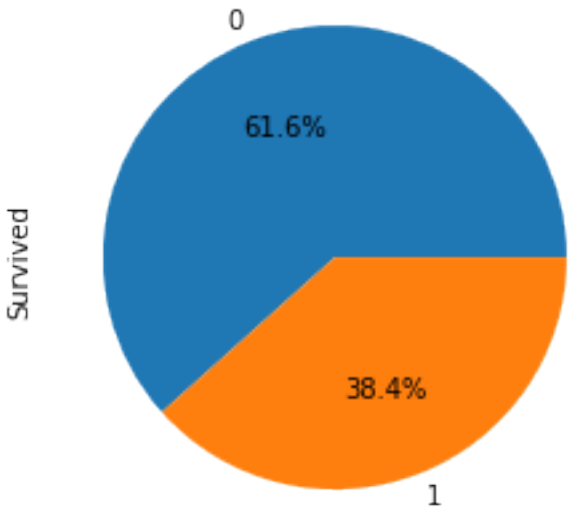
[2]:
```
train_data = pd.read_csv('data/train.csv')
train_data
```

[2]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
```

```
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                               Allen, Mr. William Henry    male  35.0      0
..                                                   …     …   …    …
886                                Montvila, Rev. Juozas    male  27.0      0
887                         Graham, Miss. Margaret Edith  female  19.0      0
888             Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                                Behr, Mr. Karl Howell    male  26.0      0
890                                  Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …      …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[3]:
```
# Find the survival percentage
train_data['Survived'].value_counts().plot.pie(autopct = '%1.1f%%')
```

[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac72bfbe0>
```



1.1.2 Testing Set

[4]:
```
test_data = pd.read_csv('data/test.csv')
test_data
```

[4]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …     …                                               …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                    Peter, Master. Michael J

        Sex   Age  SibSp  Parch              Ticket      Fare Cabin Embarked
0      male  34.5      0      0              330911    7.8292   NaN        Q
1    female  47.0      1      0              363272    7.0000   NaN        S
2      male  62.0      0      0              240276    9.6875   NaN        Q
3      male  27.0      0      0              315154    8.6625   NaN        S
4    female  22.0      1      1             3101298   12.2875   NaN        S
..      …   …    …    …                 …       …   …      …
413    male   NaN      0      0           A.5. 3236    8.0500   NaN        S
414  female  39.0      0      0            PC 17758  108.9000  C105        C
415    male  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416    male   NaN      0      0              359309    8.0500   NaN        S
417    male   NaN      1      1                2668   22.3583   NaN        C

[418 rows x 11 columns]
```

1.1.3 Information of Datasets

[5]:
```
train_data.info()
print('-' * 40)
test_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
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 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          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

1.2 Filling Missing Values for Training Set

From `train_data.info()`, we note that columns **Age**, **Cabin**, and **Embarked** have missing values.

1.2.1 Data Cleaning for Cabin Feature

There are too many missing values in Cabin feature, we can consider to drop this feature. I suppose the Cabin feature does not have enough relationship to the survival.

[6]:
```
# Drop Cabin feature
del train_data['Cabin']
train_data
```

[6]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
```

```
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                             Montvila, Rev. Juozas    male  27.0      0
887                      Graham, Miss. Margaret Edith  female  19.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                             Behr, Mr. Karl Howell    male  26.0      0
890                               Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     …                …       …      …
886      0            211536  13.0000        S
887      0            112053  30.0000        S
888      2        W./C. 6607  23.4500        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[891 rows x 11 columns]
```

1.2.2 Filling Values for Embarked Feature

Since the Embarked feature has only 2 missing values, we will just fill these with the most common one, the mode.

[7]:
```
# Find the mode of Embarked feature
mode = train_data['Embarked'].dropna().mode()
mode
```

[7]:
```
0    S
dtype: object
```

[8]:
```
# Fill the missing values with the mode
train_data['Embarked'] = train_data['Embarked'].fillna('S')
train_data
```

[8]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                            Montvila, Rev. Juozas    male  27.0      0
887                     Graham, Miss. Margaret Edith  female  19.0      0
888         Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                            Behr, Mr. Karl Howell    male  26.0      0
890                              Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     …               …      …      …
886      0            211536  13.0000        S
887      0            112053  30.0000        S
888      2        W./C. 6607  23.4500        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[891 rows x 11 columns]
```

[9]:
```
train_data.loc[train_data['Embarked'] == 'S']
```

[9]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
2              3         1       3
3              4         1       1
4              5         0       3
6              7         0       1
..           …       …     …
883          884         0       2
884          885         0       3
886          887         0       2
887          888         1       1
888          889         0       3

                                                 Name     Sex   Age  SibSp  Parch  \
0                             Braund, Mr. Owen Harris    male  22.0      1      0
2                              Heikkinen, Miss. Laina  female  26.0      0      0
3    Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0
4                           Allen, Mr. William Henry    male  35.0      0      0
6                            McCarthy, Mr. Timothy J    male  54.0      0      0
..                                                 …     …   …    …    …
883                Banfield, Mr. Frederick James    male  28.0      0      0
884                       Sutehall, Mr. Henry Jr    male  25.0      0      0
886                         Montvila, Rev. Juozas    male  27.0      0      0
887                  Graham, Miss. Margaret Edith  female  19.0      0      0
888      Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1      2

               Ticket     Fare Embarked
0           A/5 21171   7.2500        S
2    STON/O2. 3101282   7.9250        S
3              113803  53.1000        S
4              373450   8.0500        S
6               17463  51.8625        S
..                …      …      …
883  C.A./SOTON 34068  10.5000        S
884   SOTON/OQ 392076   7.0500        S
886            211536  13.0000        S
887            112053  30.0000        S
888        W./C. 6607  23.4500        S

[646 rows x 11 columns]
```

1.2.3 Filling Values for Age Feature

In this case, if we replace the missing values by one single value, such as median and mode, the data will not be balanced and even more biased. So, I create an array that contains random numbers,

which are computed based on the mean age value in regards to the standard deviation and the number of missing values.

[10]:
```
# Find the mean and standard deviation of the training set
mean = train_data['Age'].mean()
std = train_data['Age'].std()

# Count the number of missing values
num_null = train_data['Age'].isnull().sum()

# Fill missing values in Age column with random values generated
rand_age = np.random.randint(mean - std, mean + std, size = num_null)
age_slice = train_data['Age'].copy()
age_slice[np.isnan(age_slice)] = rand_age
train_data['Age'] = age_slice
train_data['Age'] = train_data['Age'].astype(int)
train_data['Age'].isnull().sum() # It gets 0. Check the null values

train_data
```

[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1				
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38	1				
2	3	1	3	Heikkinen, Miss. Laina	female	26	0				
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1				
4	5	0	3	Allen, Mr. William Henry	male	35	0				
..	…	…	…	…	…	…	…				
886	887	0	2	Montvila, Rev. Juozas	male	27	0				
887	888	1	1	Graham, Miss. Margaret Edith	female	19	0				
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	32	1				
889	890	1	1	Behr, Mr. Karl Howell	male	26	0				
890	891	0	3	Dooley, Mr. Patrick	male	32	0				

```
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     …               …       …       …
886      0            211536  13.0000        S
887      0            112053  30.0000        S
888      2        W./C. 6607  23.4500        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[891 rows x 11 columns]
```

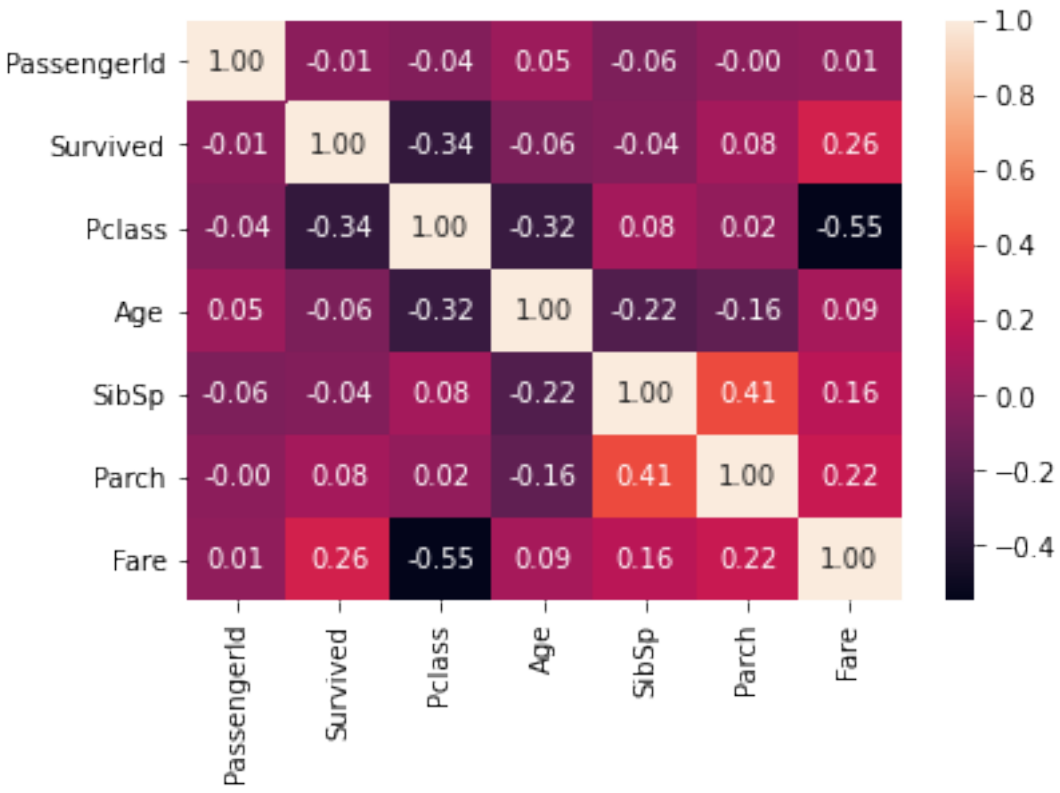
[11]:
```
train_data.info() # Now, all the features have no null values except the Cabin
 ↪feature
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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    int64
 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  Embarked     891 non-null    object
dtypes: float64(1), int64(6), object(4)
memory usage: 76.7+ KB
```

1.3 Find Relations among Different Features and Survival

[12]:
```
sns.heatmap(train_data.corr(), annot = True, fmt = ".2f")
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac726ddd8>
```



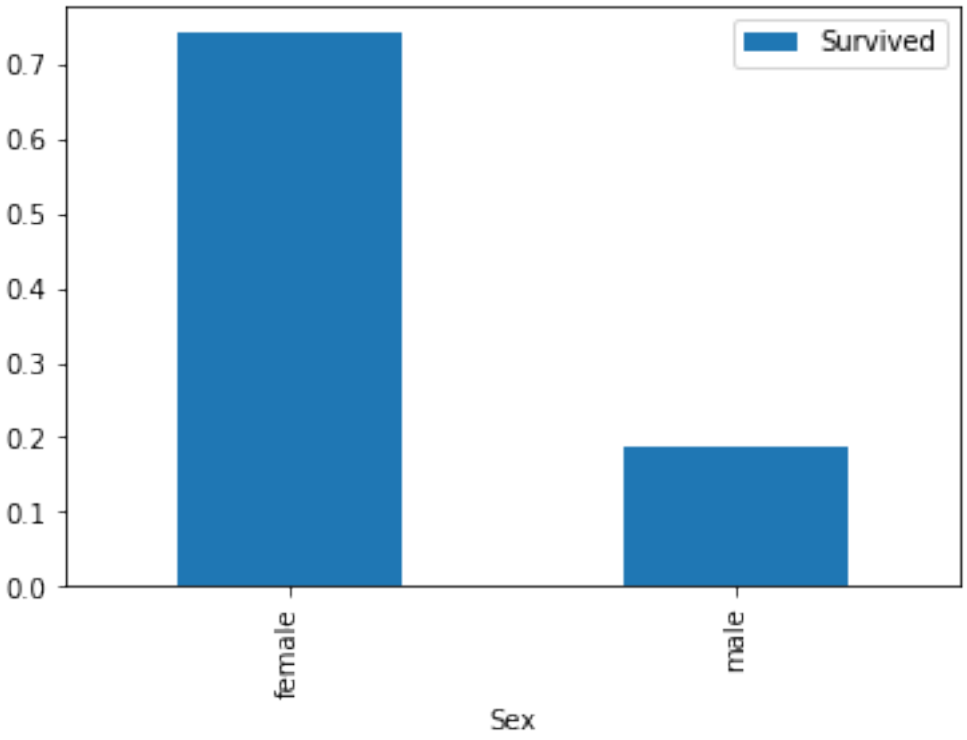
1.3.1 Is There a Relationship between Sex and Survival?

[13]:
```
train_data.groupby(['Sex', 'Survived'])['Survived'].count()
```

[13]:
```
Sex     Survived
female  0            81
        1           233
male    0           468
        1           109
Name: Survived, dtype: int64
```

[14]:
```
train_data[['Sex', 'Survived']].groupby(['Sex']).mean().plot.bar()
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac7956b70>
```



The bar chart gives that the survival rate of female is greater than that of male. "Ladies first" is a widely kept tradition at that time.

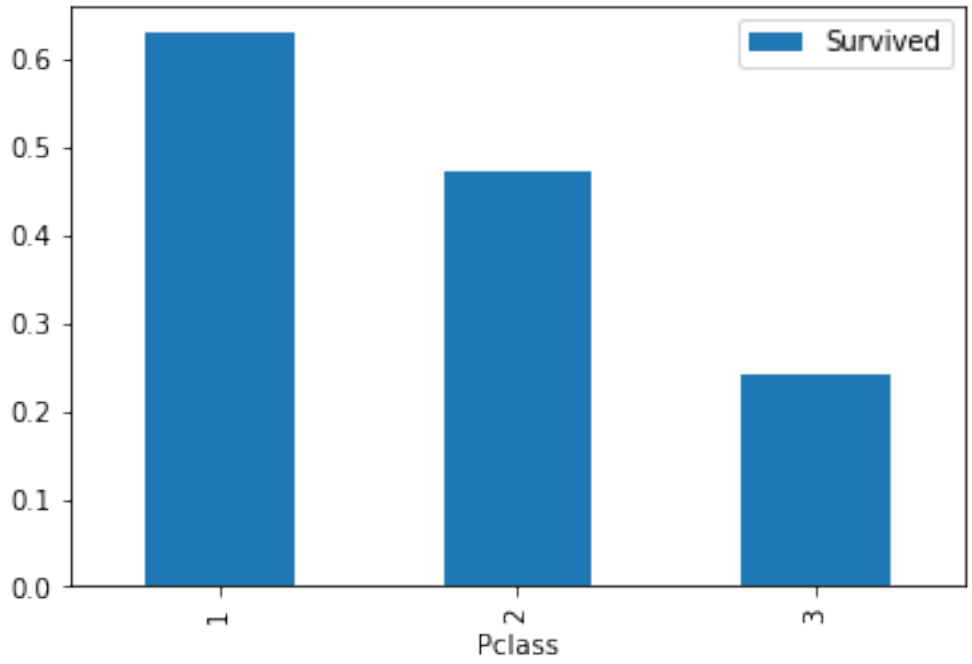
1.3.2 Is There a Relationship between Passenger Class and Survival?

[15]:
```
train_data.groupby(['Pclass', 'Survived'])['Pclass'].count()
```

[15]:
```
Pclass  Survived
1       0            80
        1           136
2       0            97
        1            87
3       0           372
        1           119
Name: Pclass, dtype: int64
```

[16]:
```
train_data[['Pclass', 'Survived']].groupby(['Pclass']).mean().plot.bar()
```

[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac7a30b00>
```



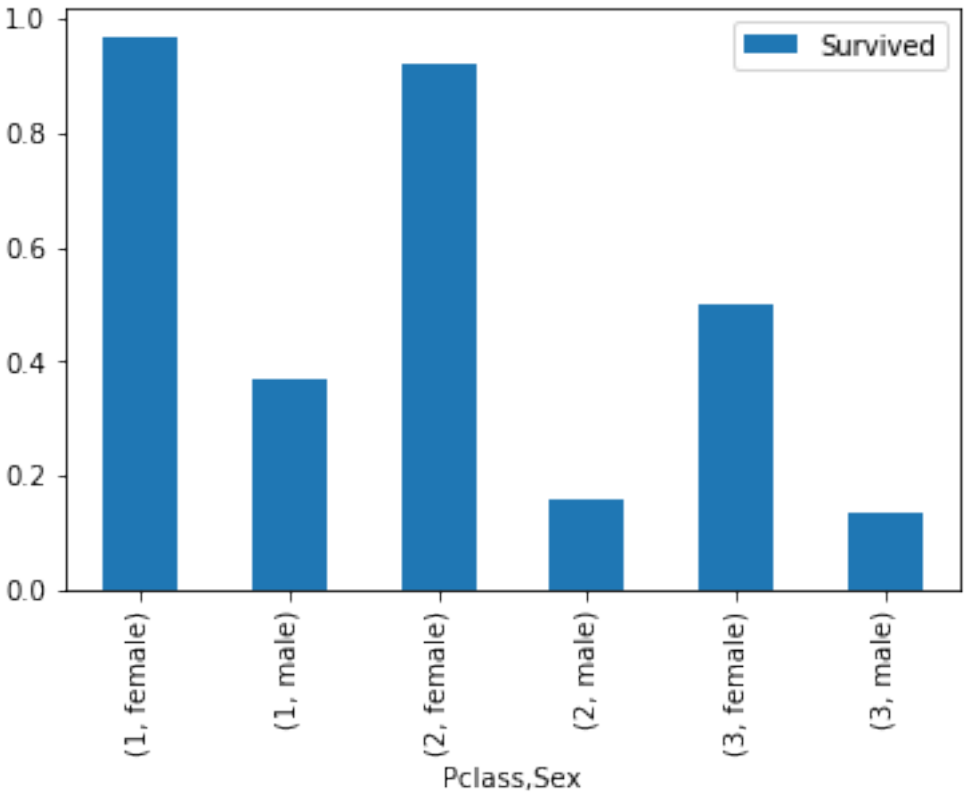
We generally note that the upper passenger class has the higher survival rate. With consideration of sex feature, we get the same result that females have a higher survival rate.

[17]:
```
train_data.groupby(['Sex', 'Pclass', 'Survived'])['Survived'].count()
```

[17]:
```
Sex     Pclass  Survived
female  1       0             3
                1            91
        2       0             6
                1            70
        3       0            72
                1            72
male    1       0            77
                1            45
        2       0            91
                1            17
        3       0           300
                1            47
Name: Survived, dtype: int64
```

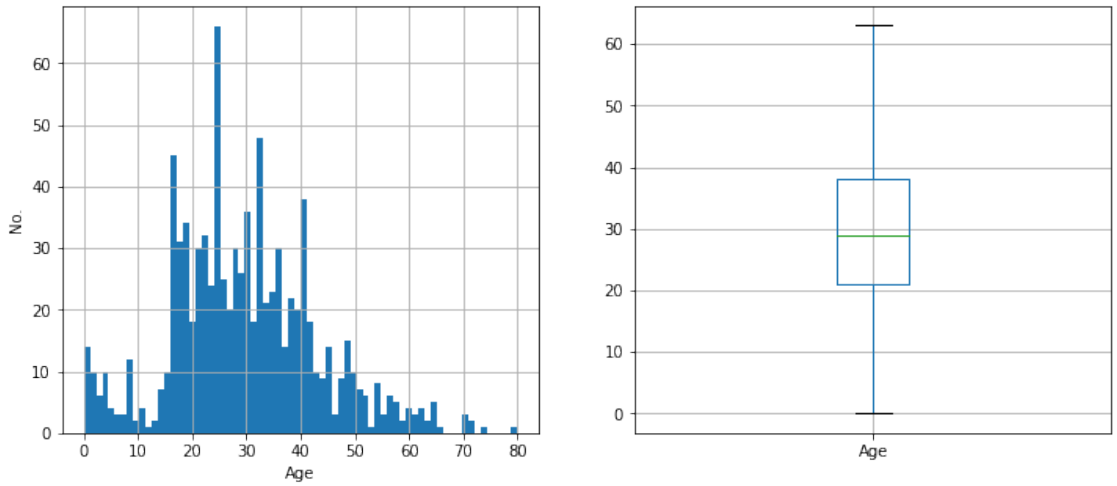
[18]:
```
train_data[['Sex', 'Pclass', 'Survived']].groupby(['Pclass', 'Sex']).mean().
↪plot.bar()
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac7a97b38>
```



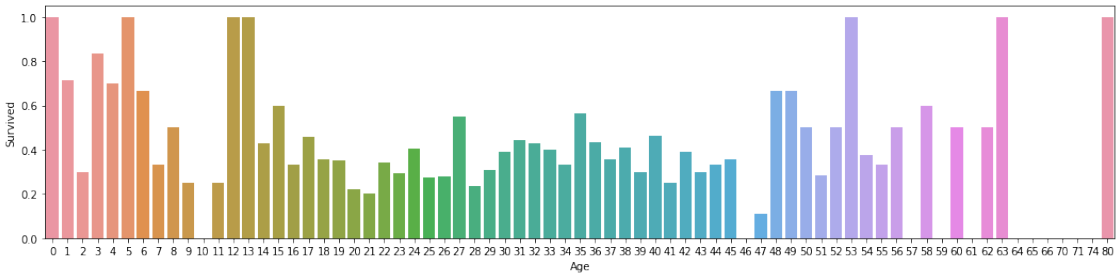
1.3.3 Is There a Relationship between Age and Survival?

[19]:
```
plt.figure(figsize = (12, 5))
plt.subplot(121)
train_data['Age'].hist(bins = 70)
plt.xlabel('Age')
plt.ylabel('No.')

plt.subplot(122)
train_data.boxplot(column = 'Age', showfliers = False)
plt.show()
```



[20]:
```
fig, axis1 = plt.subplots(1, 1, figsize = (18, 4))
train_data['Age'] = train_data['Age'].astype(int)
average_age_per = train_data[['Age', 'Survived']].groupby(['Age'], as_index =␣
↪False).mean()
sns.barplot(x = 'Age', y = 'Survived', data = average_age_per)
```

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7feac7e7f6a0>



[21]:
```
train_data['Age'].describe()
```

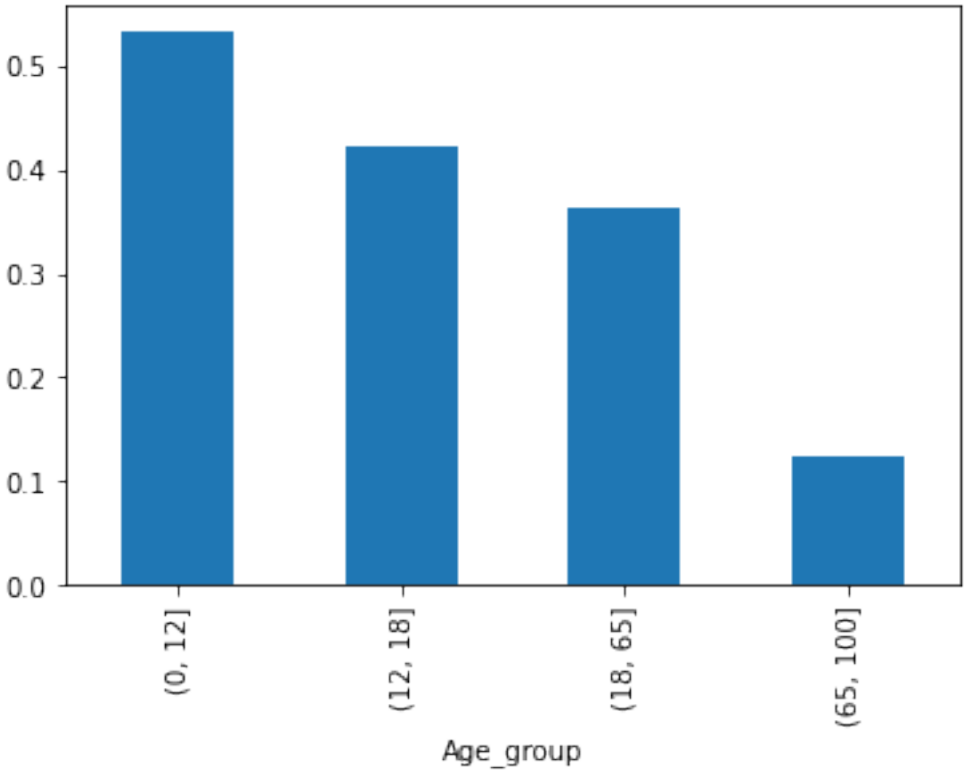
[21]:
```
count    891.000000
mean      29.634119
std       13.558642
min        0.000000
25%       21.000000
50%       29.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

[22]:
```
bins = [0, 12, 18, 65, 100]
train_data['Age_group'] = pd.cut(train_data['Age'], bins)
by_age = train_data.groupby('Age_group')['Survived'].mean()
by_age
```

[22]:
```
Age_group
(0, 12]      0.532258
(12, 18]     0.421053
(18, 65]     0.363004
(65, 100]    0.125000
Name: Survived, dtype: float64
```

[23]:
```
by_age.plot(kind = 'bar')
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac807cac8>
```



1.3.4 Is There a Relationship between Name and Survival?

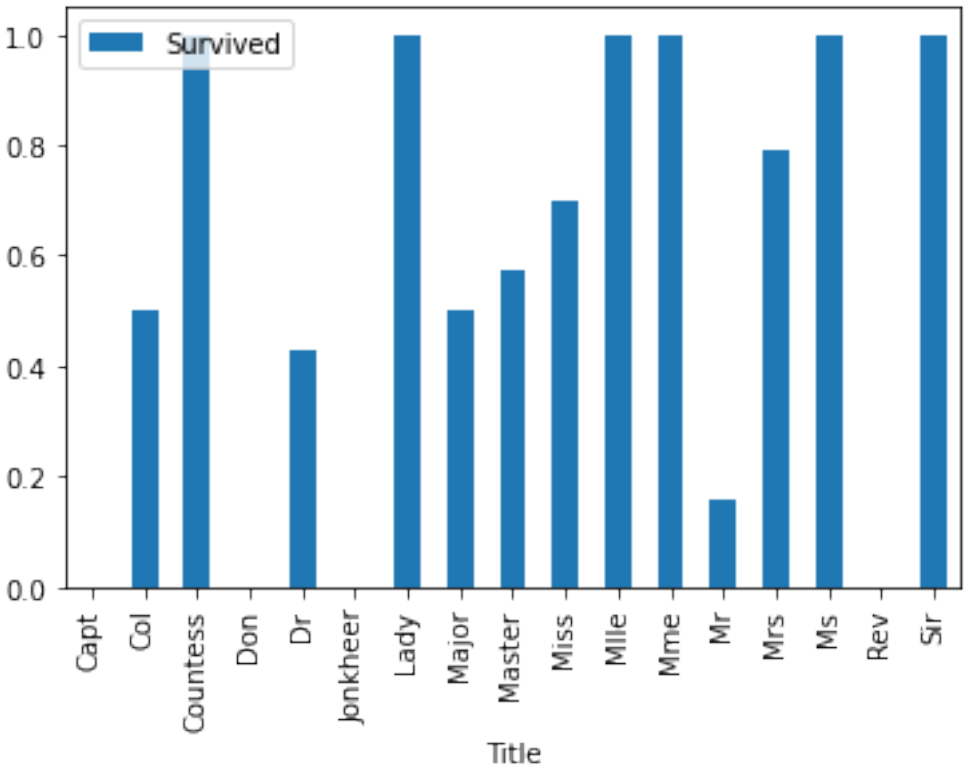
[24]:
```
train_data['Title'] = train_data['Name'].str.extract(' ([A-Za-z]+)\.', expand =␣
↪False)
pd.crosstab(train_data['Title'], train_data['Sex'])
```

[24]:

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

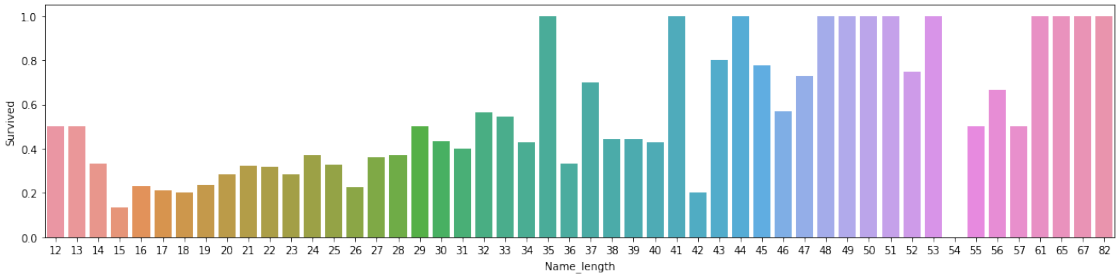
[25]:
```
train_data[['Title','Survived']].groupby(['Title']).mean().plot.bar()
```

[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac82534a8>
```



[26]:
```
fig, axis1 = plt.subplots(1, 1, figsize = (18, 4))
train_data['Name_length'] = train_data['Name'].apply(len)
name_length = train_data[['Name_length', 'Survived']].
 ↪groupby(['Name_length'],as_index = False).mean()
sns.barplot(x = 'Name_length', y = 'Survived', data = name_length)
```

[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac82f1c88>
```



It is hard to conclude the relationship between name and survival.

1.3.5 Is There a Relationship between Siblings and Survival?

[27]:
```
sibsp_df = train_data[train_data['SibSp'] != 0]
no_sibsp_df = train_data[train_data['SibSp'] == 0]
```

[28]:
```
sibsp_df
```

[28]:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
3	4	1	1	
7	8	0	3	
9	10	1	2	
..	…	…	…	
866	867	1	2	
869	870	1	3	
871	872	1	1	
874	875	1	2	
888	889	0	3	

	Name	Sex	Age	SibSp	\
0	Braund, Mr. Owen Harris	male	22	1	
1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38	1	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	
7	Palsson, Master. Gosta Leonard	male	2	3	

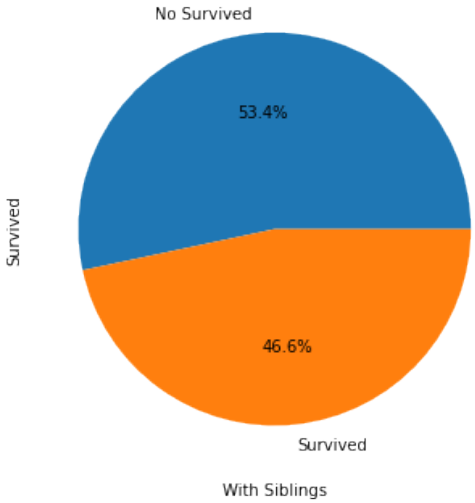
```
9                 Nasser, Mrs. Nicholas (Adele Achem)  female   14      1
..                                                 …     …  …    …
866                     Duran y More, Miss. Asuncion  female   27      1
869                  Johnson, Master. Harold Theodor    male    4      1
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female   47      1
874             Abelson, Mrs. Samuel (Hannah Wizosky)  female   28      1
888          Johnston, Miss. Catherine Helen "Carrie"  female   32      1

     Parch         Ticket     Fare Embarked Age_group   Title  Name_length
0        0      A/5 21171   7.2500        S  (18, 65]      Mr           23
1        0       PC 17599  71.2833        C  (18, 65]     Mrs           51
3        0         113803  53.1000        S  (18, 65]     Mrs           44
7        1         349909  21.0750        S   (0, 12]  Master           30
9        0         237736  30.0708        C  (12, 18]     Mrs           35
..     …            …      …      …       …     …          …
866      0  SC/PARIS 2149  13.8583        C  (18, 65]    Miss           28
869      1         347742  11.1333        S   (0, 12]  Master           31
871      1          11751  52.5542        S  (18, 65]     Mrs           48
874      0      P/PP 3381  24.0000        C  (18, 65]     Mrs           37
888      2     W./C. 6607  23.4500        S  (18, 65]    Miss           40

[283 rows x 14 columns]
```

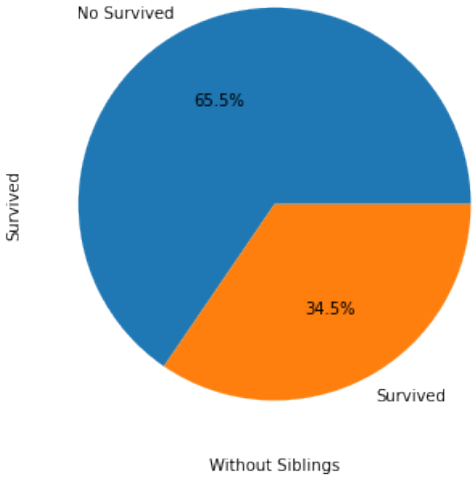
[29]:
```
no_sibsp_df
```

[29]:
```
     PassengerId  Survived  Pclass  \
2              3         1       3
4              5         0       3
5              6         0       3
6              7         0       1
8              9         1       3
..           …       …     …
885          886         0       3
886          887         0       2
887          888         1       1
889          890         1       1
890          891         0       3

                                                 Name     Sex  Age  SibSp  \
2                             Heikkinen, Miss. Laina  female   26      0
4                           Allen, Mr. William Henry    male   35      0
5                                   Moran, Mr. James    male   29      0
6                            McCarthy, Mr. Timothy J    male   54      0
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female   27      0
..                                                 …     …  …    …
885              Rice, Mrs. William (Margaret Norton)  female   39      0
886                             Montvila, Rev. Juozas    male   27      0
```

```
887                      Graham, Miss. Margaret Edith  female   19      0
889                             Behr, Mr. Karl Howell    male   26      0
890                               Dooley, Mr. Patrick    male   32      0

     Parch            Ticket     Fare Embarked Age_group Title  Name_length
2        0  STON/O2. 3101282   7.9250        S  (18, 65]  Miss           22
4        0            373450   8.0500        S  (18, 65]    Mr           24
5        0            330877   8.4583        Q  (18, 65]    Mr           16
6        0             17463  51.8625        S  (18, 65]    Mr           23
8        2            347742  11.1333        S  (18, 65]   Mrs           49
..     …               …       …       …       …   …          …
885      5            382652  29.1250        Q  (18, 65]   Mrs           36
886      0            211536  13.0000        S  (18, 65]   Rev           21
887      0            112053  30.0000        S  (18, 65]  Miss           28
889      0            111369  30.0000        C  (18, 65]    Mr           21
890      0            370376   7.7500        Q  (18, 65]    Mr           19

[608 rows x 14 columns]
```

[30]:
```
plt.figure(figsize = (12, 6))

plt.subplot(121)
sibsp_df['Survived'].value_counts().plot.pie(labels = ['No Survived',␣
↪'Survived'], autopct = '%1.1f%%')
plt.xlabel('With Siblings')

plt.subplot(122)
no_sibsp_df['Survived'].value_counts().plot.pie(labels = ['No Survived',␣
↪'Survived'], autopct = '%1.1f%%')
plt.xlabel('Without Siblings')

plt.show()
```





1.3.6 Is There a Relationship between Parents/Children and Survival?

[31]:
```
parch_df = train_data[train_data['Parch'] != 0]
no_parch_df = train_data[train_data['Parch'] == 0]
```

[32]:
```
parch_df
```

[32]:

	PassengerId	Survived	Pclass	\
7	8	0	3	
8	9	1	3	
10	11	1	3	
13	14	0	3	
16	17	0	3	
..	…	…	…	
871	872	1	1	
879	880	1	1	
880	881	1	2	
885	886	0	3	
888	889	0	3	

	Name	Sex	Age	SibSp	\
7	Palsson, Master. Gosta Leonard	male	2	3	
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	
10	Sandstrom, Miss. Marguerite Rut	female	4	1	
13	Andersson, Mr. Anders Johan	male	39	1	
16	Rice, Master. Eugene	male	2	4	
..	…	…	…	…	
871	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47	1	
879	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56	0	

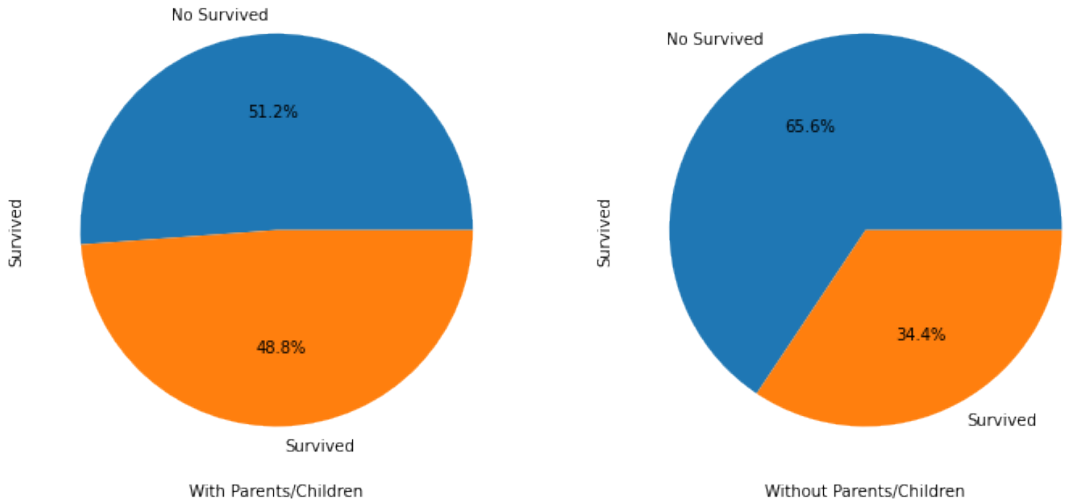
```
880       Shelley, Mrs. William (Imanita Parrish Hall)  female   25      0
885                Rice, Mrs. William (Margaret Norton)  female   39      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   32      1

     Parch      Ticket     Fare Embarked Age_group   Title  Name_length
7        1      349909  21.0750        S   (0, 12]  Master           30
8        2      347742  11.1333        S  (18, 65]     Mrs           49
10       1     PP 9549  16.7000        S   (0, 12]    Miss           31
13       5      347082  31.2750        S  (18, 65]      Mr           27
16       1      382652  29.1250        Q   (0, 12]  Master           20
..     …          …       …      …       …     …          …
871      1       11751  52.5542        S  (18, 65]     Mrs           48
879      1       11767  83.1583        C  (18, 65]     Mrs           45
880      1      230433  26.0000        S  (18, 65]     Mrs           44
885      5      382652  29.1250        Q  (18, 65]     Mrs           36
888      2  W./C. 6607  23.4500        S  (18, 65]    Miss           40

[213 rows x 14 columns]
```

[33]:
```
no_parch_df
```

[33]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
884          885         0       3
886          887         0       2
887          888         1       1
889          890         1       1
890          891         0       3

                                                 Name     Sex  Age  SibSp  \
0                             Braund, Mr. Owen Harris    male   22      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female   38      1
2                                Heikkinen, Miss. Laina  female   26      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35      1
4                               Allen, Mr. William Henry    male   35      0
..                                                   …     …  …    …
884                           Sutehall, Mr. Henry Jr    male   25      0
886                            Montvila, Rev. Juozas    male   27      0
887                     Graham, Miss. Margaret Edith  female   19      0
889                            Behr, Mr. Karl Howell    male   26      0
890                              Dooley, Mr. Patrick    male   32      0
```

```
     Parch            Ticket     Fare Embarked Age_group Title  Name_length
0        0         A/5 21171   7.2500        S  (18, 65]    Mr           23
1        0          PC 17599  71.2833        C  (18, 65]   Mrs           51
2        0  STON/O2. 3101282   7.9250        S  (18, 65]  Miss           22
3        0            113803  53.1000        S  (18, 65]   Mrs           44
4        0            373450   8.0500        S  (18, 65]    Mr           24
..     ...               ...      ...      ...       ...   ...          ...
884      0   SOTON/OQ 392076   7.0500        S  (18, 65]    Mr           22
886      0            211536  13.0000        S  (18, 65]   Rev           21
887      0            112053  30.0000        S  (18, 65]  Miss           28
889      0            111369  30.0000        C  (18, 65]    Mr           21
890      0            370376   7.7500        Q  (18, 65]    Mr           19

[678 rows x 14 columns]
```

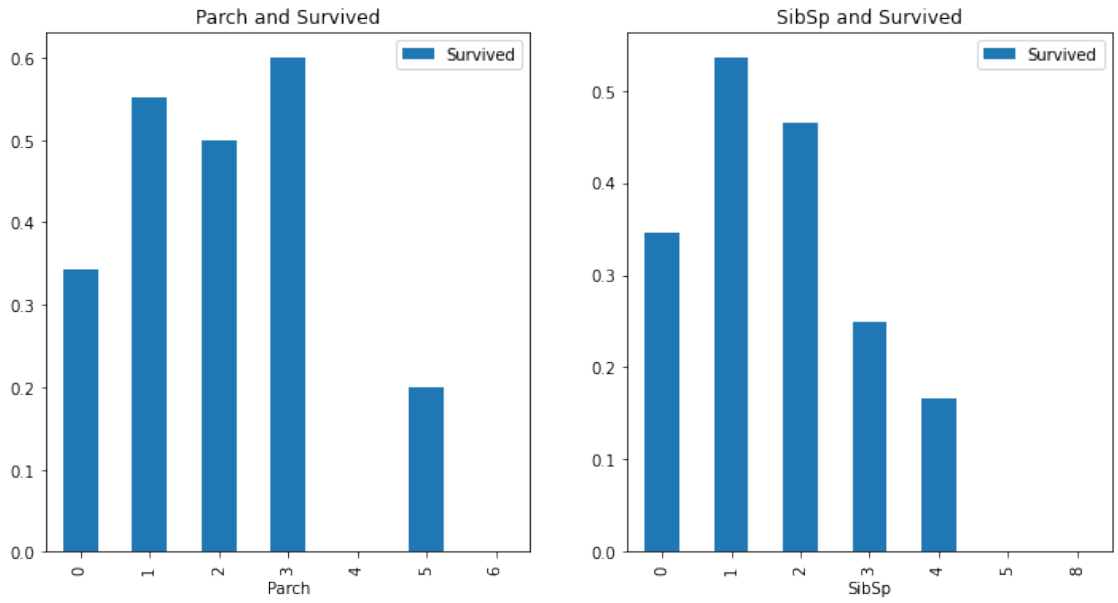
[34]:
```
plt.figure(figsize = (12, 6))
plt.subplot(121)
parch_df['Survived'].value_counts().plot.pie(labels = ['No Survived',␣
↪'Survived'], autopct = '%1.1f%%')
plt.xlabel('With Parents/Children')

plt.subplot(122)
no_parch_df['Survived'].value_counts().plot.pie(labels = ['No Survived',␣
↪'Survived'], autopct = '%1.1f%%')
plt.xlabel('Without Parents/Children')

plt.show()
```



1.3.7 Is There a Relationship between Family and Survival?

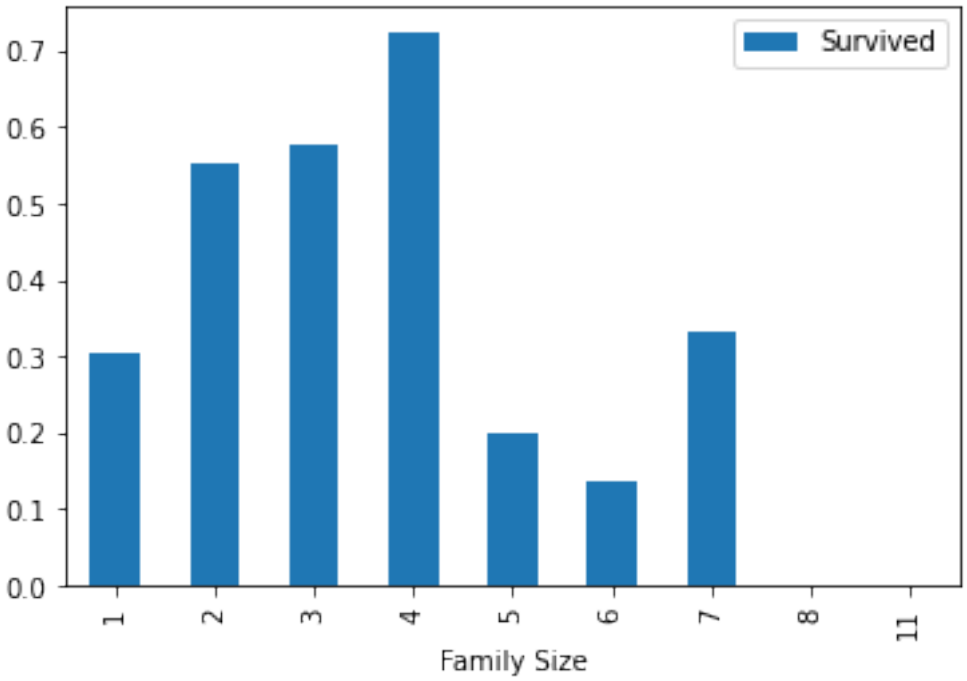
[35]:
```
fig, ax = plt.subplots(1, 2, figsize = (12, 6))

train_data[['Parch', 'Survived']].groupby(['Parch']).mean().plot.bar(ax = ax[0])
ax[0].set_title('Parch and Survived')

train_data[['SibSp', 'Survived']].groupby(['SibSp']).mean().plot.bar(ax = ax[1])
ax[1].set_title('SibSp and Survived')
```

[35]:
```
Text(0.5, 1.0, 'SibSp and Survived')
```



[36]:
```
train_data['Family Size'] = train_data['Parch'] + train_data['SibSp'] + 1
train_data[['Family Size', 'Survived']].groupby(['Family Size']).mean().plot.
↪bar()
```

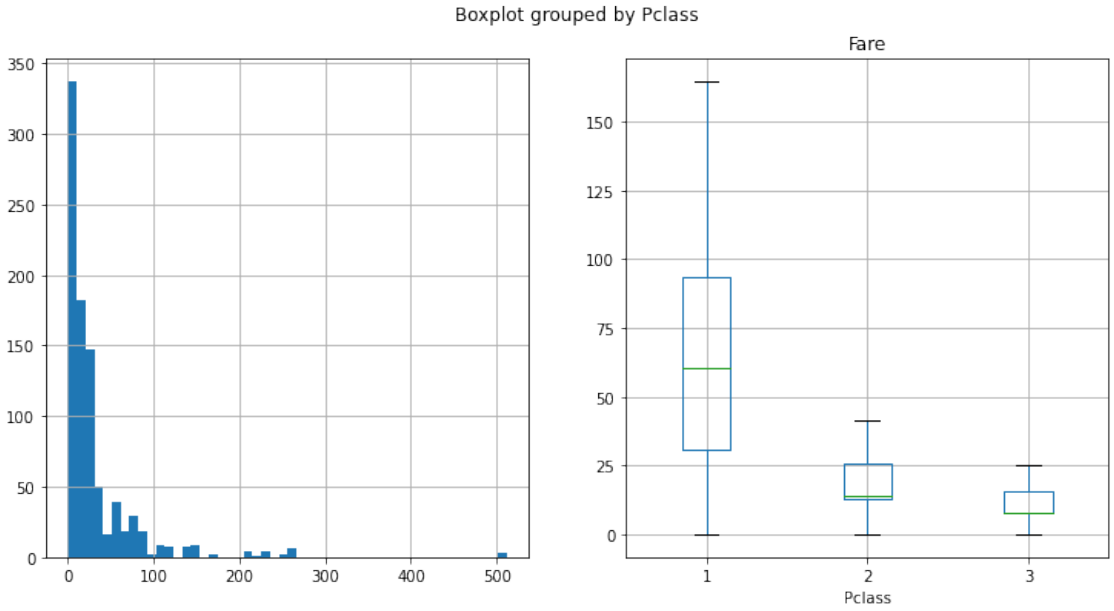
[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feac808eb38>
```



The plot shows that the survival rates for people alone and larger family size are low, but for the average size family (like 2 to 4 members), their survival rates tend to be higher.

1.3.8 Is There a Relationship between Fare and Survival?

[37]:
```
fig, ax = plt.subplots(1, 2, figsize = (12, 6))
train_data['Fare'].hist(bins = 50, ax = ax[0])
train_data.boxplot(column = 'Fare', by = 'Pclass', ax = ax[1], showfliers =␣
↪False)
plt.show()
```



[38]:
```
train_data['Fare'].describe()
```

[38]:
```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

[39]:
```
fare_not_survived = train_data['Fare'][train_data['Survived'] == 0]
fare_survived = train_data['Fare'][train_data['Survived'] == 1]
```

[40]:
```
fare_not_survived
```

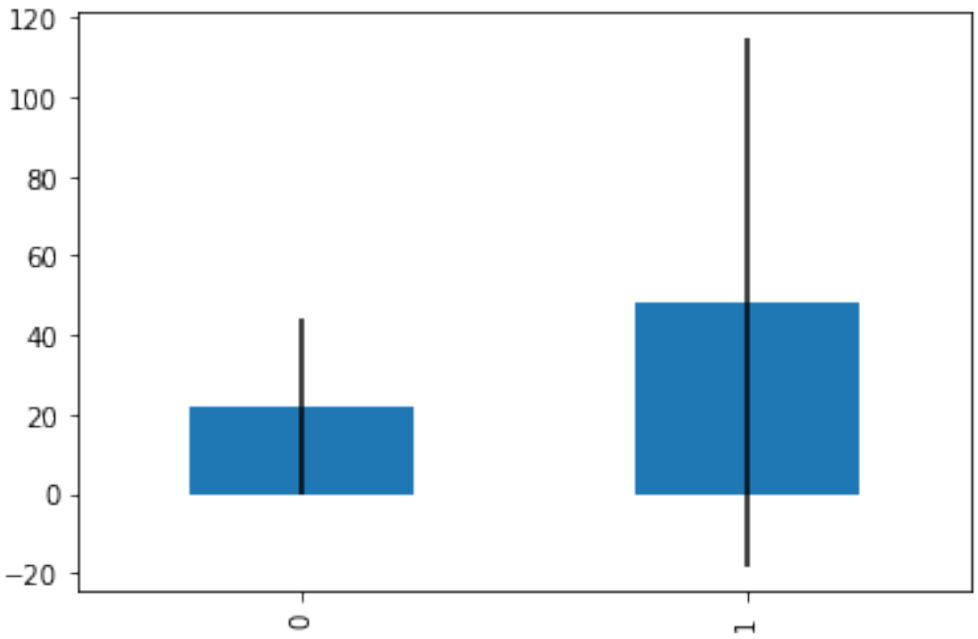
[40]:
```
0       7.2500
4       8.0500
5       8.4583
6      51.8625
7      21.0750
        ...
884     7.0500
885    29.1250
886    13.0000
888    23.4500
```

```
890     7.7500
Name: Fare, Length: 549, dtype: float64
```

[41]:
```
fare_survived
```

[41]:
```
1      71.2833
2       7.9250
3      53.1000
8      11.1333
9      30.0708
        ...
875     7.2250
879    83.1583
880    26.0000
887    30.0000
889    30.0000
Name: Fare, Length: 342, dtype: float64
```

[42]:
```
average_fare = pd.DataFrame([fare_not_survived.mean(), fare_survived.mean()])
std_fare = pd.DataFrame([fare_not_survived.mean(), fare_survived.std()])
average_fare.plot(yerr = std_fare, kind = 'bar', legend = False)

plt.show()
```

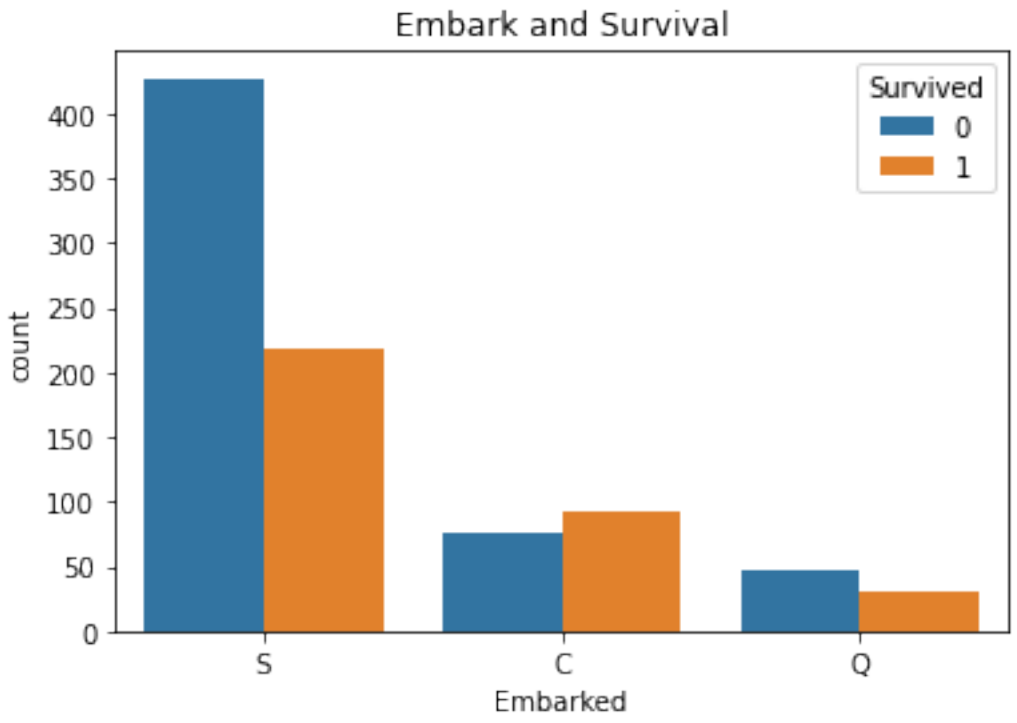


The plot above gives that the fares survivors give tend to be greater than those others give.

1.3.9 Is There a Relationship between Embark and Survival?

[43]:
```
sns.countplot('Embarked', hue = 'Survived', data = train_data)
plt.title('Embark and Survival')
```

[43]:
```
Text(0.5, 1.0, 'Embark and Survival')
```



[44]:
```
train_data[['Embarked', 'Survived']].groupby(['Embarked'], as_index = False).
↪mean().sort_values(by = 'Survived', ascending = False)
```

[44]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

The Titanic departed from Southampton Port in the United Kingdom and passed through Cherbourg, France and Queenstown, Ireland. Those who boarded the ship before Queenstown may disembark at Cherbourg or Queenstown. These people would not meet To shipwreck.

Also, the survival rates vary from port to port. The people who embarked at Cherbourg have the highest possibility to survive, while the people who embarked at Southampton Port have the lowest possibility to survive. Also, applying the Central Limit Theorem, if these samples are randomly selected, the result we get is representative.

[45]:
```
corr_matrix = train_data.corr()
corr_matrix['Survived'].sort_values(ascending = False)
```

[45]:
```
Survived       1.000000
Name_length    0.332350
Fare           0.257307
Parch          0.081629
Family Size    0.016639
PassengerId   -0.005007
SibSp         -0.035322
Age           -0.061288
Pclass        -0.338481
Name: Survived, dtype: float64
```

1.4 Filling Missing Values for Testing Set

From `test_data.info()`, we note that columns **Age**, **Fare**, and **Cabin** have missing values.

1.4.1 Data Cleaning for Cabin Feature

[46]:
```
del test_data['Cabin']
test_data
```

[46]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	S
..	…	…	…	…	…	…	…	…	…	…
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	S
417	1309	3	Peter, Master. Michael J							

```
417    male   NaN      1      1                2668   22.3583        C

[418 rows x 10 columns]
```

1.4.2 Filling Values for Age Feature

[47]:
```
# Find the mean and standard deviation of the training set
mean = test_data['Age'].mean()
std = test_data['Age'].std()

# Count the number of missing values
num_null = test_data['Age'].isnull().sum()

# Fill missing values in Age column with random values generated
rand_age = np.random.randint(mean - std, mean + std, size = num_null)
age_slice = test_data['Age'].copy()
age_slice[np.isnan(age_slice)] = rand_age
test_data['Age'] = age_slice
test_data['Age'] = test_data['Age'].astype(int)
test_data['Age'].isnull().sum() # It gets 0. Check the null values

test_data
```

[47]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …     …                                               …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex  Age  SibSp  Parch              Ticket      Fare Embarked
0      male   34      0      0              330911    7.8292        Q
1    female   47      1      0              363272    7.0000        S
2      male   62      0      0              240276    9.6875        Q
3      male   27      0      0              315154    8.6625        S
4    female   22      1      1             3101298   12.2875        S
..      …  …    …    …                 …       …      …
413    male   16      0      0           A.5. 3236    8.0500        S
414  female   39      0      0            PC 17758  108.9000        C
415    male   38      0      0  SOTON/O.Q. 3101262    7.2500        S
416    male   18      0      0              359309    8.0500        S
```

```
417    male   30      1      1                2668   22.3583        C

[418 rows x 10 columns]
```

[48]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 10 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    int64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Embarked     418 non-null    object
dtypes: float64(1), int64(5), object(4)
memory usage: 32.8+ KB
```

[49]:
```
test_data['Age'].describe()
```

[49]:
```
count    418.000000
mean      30.306220
std       13.198794
min        0.000000
25%       21.250000
50%       28.500000
75%       38.750000
max       76.000000
Name: Age, dtype: float64
```

1.4.3 Filling Values for Fare Feature

In the **Fare** column, it has only one missing value, so the mean fare is a great option to fill the missing one.

[50]:
```
test_data['Fare'].fillna(test_data['Fare'].mean(), inplace = True)
test_data
```

[50]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
```

```
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …     …                                              …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex  Age  SibSp  Parch              Ticket      Fare Embarked
0      male   34      0      0              330911    7.8292        Q
1    female   47      1      0              363272    7.0000        S
2      male   62      0      0              240276    9.6875        Q
3      male   27      0      0              315154    8.6625        S
4    female   22      1      1             3101298   12.2875        S
..      …  …    …    …                 …       …      …
413    male   16      0      0           A.5. 3236    8.0500        S
414  female   39      0      0            PC 17758  108.9000        C
415    male   38      0      0  SOTON/O.Q. 3101262    7.2500        S
416    male   18      0      0              359309    8.0500        S
417    male   30      1      1                2668   22.3583        C

[418 rows x 10 columns]
```

1.5 Feature Selection

By previous parts, we note that **PassengerId**, **Name**, **Cabin**, and **Ticket** are hard to use as a classifier, so we give up on these features.

[51]:
```
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
train_features = train_data[features]
train_labels = train_data['Survived']
test_features = test_data[features]
```

[52]:
```
train_features
```

[52]:
```
     Pclass     Sex  Age  SibSp  Parch     Fare Embarked
0         3    male   22      1      0   7.2500        S
1         1  female   38      1      0  71.2833        C
2         3  female   26      0      0   7.9250        S
3         1  female   35      1      0  53.1000        S
4         3    male   35      0      0   8.0500        S
..      …     …  …    …    …      …       …
886       2    male   27      0      0  13.0000        S
887       1  female   19      0      0  30.0000        S
888       3  female   32      1      2  23.4500        S
```

```
889       1    male   26      0      0  30.0000        C
890       3    male   32      0      0   7.7500        Q

[891 rows x 7 columns]
```

[53]:
```
train_labels
```

[53]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

[54]:
```
test_features
```

[54]:
```
     Pclass     Sex  Age  SibSp  Parch      Fare Embarked
0         3    male   34      0      0    7.8292        Q
1         3  female   47      1      0    7.0000        S
2         2    male   62      0      0    9.6875        Q
3         3    male   27      0      0    8.6625        S
4         3  female   22      1      1   12.2875        S
..      …     …  …    …    …       …       …
413       3    male   16      0      0    8.0500        S
414       1  female   39      0      0  108.9000        C
415       3    male   38      0      0    7.2500        S
416       3    male   18      0      0    8.0500        S
417       3    male   30      1      1   22.3583        C

[418 rows x 7 columns]
```

In the features **Embarked** and **Sex**, which are categorical, we transform these features to numerical, represented by 0 and 1. For the feature **Sex**, we have either male or female. In visualization, we transform it to `Sex = female` and `Sex = male` with possible values 0 and 1. We do the similar things for **Embarked** by transforming variables to `Embarked = C`, `Embarked = Q`, and `Embarked = S`. We use the module `DictVectorizer` to achieve what we want.

[55]:
```
from sklearn.feature_extraction import DictVectorizer
dv = DictVectorizer(sparse = False)
train_features = dv.fit_transform(train_features.to_dict(orient = 'record'))
```

[56]:
```
print(dv.feature_names_)
```

```
['Age', 'Embarked=C', 'Embarked=Q', 'Embarked=S', 'Fare', 'Parch', 'Pclass',
'Sex=female', 'Sex=male', 'SibSp']
```

[57]:
```
train_features
```

[57]:
```
array([[22.,  0.,  0., …,  0.,  1.,  1.],
       [38.,  1.,  0., …,  1.,  0.,  1.],
       [26.,  0.,  0., …,  1.,  0.,  0.],
       …,
       [32.,  0.,  0., …,  1.,  0.,  1.],
       [26.,  1.,  0., …,  0.,  1.,  0.],
       [32.,  0.,  1., …,  0.,  1.,  0.]])
```

1.6 Decision Tree and Further Prediction

I use the ID3 (Iterative Dichotomiser 3) algorithm to construct a decision tree via creating a `DecisionTreeClassifier` with the criterion entropy.

[58]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion = 'entropy')
dtc.fit(train_features, train_labels) # train the decision tree
```

[58]:
```
DecisionTreeClassifier(criterion='entropy')
```

[59]:
```
dt_accuracy = round(dtc.score(train_features, train_labels), 6)
dt_accuracy
```

[59]:
```
0.987654
```

The accuracy of this decision tree is great, since it is close to 1. However, we are measuring for the training set. For the test set, we use **k-fold cross validation**.

[60]:
```
test_features = dv.transform(test_features.to_dict(orient = 'record'))
pred_labels = dtc.predict(test_features)
```

[61]:
```
pred_labels
```

[61]:
```
array([1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
```

```
0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0])
```

[62]:
```
from sklearn.model_selection import cross_val_score
np.mean(cross_val_score(dtc, train_features, train_labels, cv = 10))
```

[62]:
```
0.78458177278402
```

It seems the difference between training and testing sets is huge. The decision tree is not the best model in this case.

1.7 Submission

[63]:
```
submission = pd.read_csv("data/test.csv")
submission['Survived'] = pred_labels
submission.drop(submission.columns.difference(['PassengerId', 'Survived']),␣
↪axis = 1, inplace = True) # Select the necessary columns
submission
```

[63]:
```
     PassengerId  Survived
0            892         1
1            893         0
2            894         1
3            895         1
4            896         1
..           …       …
413         1305         0
414         1306         1
415         1307         0
416         1308         0
417         1309         0

[418 rows x 2 columns]
```

[64]:
```
submission.to_csv('submission.csv', index = False)
```

1.8 Visualizing Decision Tree

[65]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn import tree

plt.figure(figsize = (25, 15))
tree.plot_tree(dtc, feature_names = dv.feature_names_, filled = True, rounded =␣
↪True)
```

[65]:
```
mples = 2\nvalue = [1, 1]'),
 Text(718.7977099236641, 352.1045454545455, 'entropy = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(740.0954198473282, 463.2954545454545, 'Fare <= 29.85\nentropy =
0.945\nsamples = 80\nvalue = [51, 29]'),
 Text(729.4465648854962, 426.2318181818182, 'entropy = 0.0\nsamples = 9\nvalue =
[9, 0]'),
 Text(750.7442748091603, 426.2318181818182, 'Age <= 24.5\nentropy =
0.976\nsamples = 71\nvalue = [42, 29]'),
 Text(740.0954198473282, 389.1681818181818, 'entropy = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(761.3931297709923, 389.1681818181818, 'Age <= 27.5\nentropy =
0.987\nsamples = 67\nvalue = [38, 29]'),
 Text(740.0954198473282, 352.1045454545455, 'Parch <= 1.0\nentropy =
0.764\nsamples = 9\nvalue = [2, 7]'),
 Text(729.4465648854962, 315.04090909090905, 'entropy = 0.0\nsamples = 7\nvalue
= [0, 7]'),
 Text(750.7442748091603, 315.04090909090905, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(782.6908396946565, 352.1045454545455, 'Fare <= 369.927\nentropy =
0.958\nsamples = 58\nvalue = [36, 22]'),
 Text(772.0419847328244, 315.04090909090905, 'Fare <= 30.598\nentropy =
0.94\nsamples = 56\nvalue = [36, 20]'),
 Text(736.7676526717557, 277.97727272727275, 'Age <= 33.5\nentropy =
0.918\nsamples = 6\nvalue = [2, 4]'),
 Text(726.1187977099237, 240.91363636363633, 'Fare <= 30.285\nentropy =
0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(715.4699427480916, 203.85000000000002, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(736.7676526717557, 203.85000000000002, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(747.4165076335878, 240.91363636363633, 'entropy = 0.0\nsamples = 3\nvalue
= [0, 3]'),
 Text(807.3163167938932, 277.97727272727275, 'Fare <= 52.277\nentropy =
0.904\nsamples = 50\nvalue = [34, 16]'),
 Text(768.7142175572519, 240.91363636363633, 'Age <= 47.5\nentropy =
0.629\nsamples = 19\nvalue = [16, 3]'),
```

```
 Text(758.0653625954199, 203.85000000000002, 'Fare <= 36.252\nentropy =
0.503\nsamples = 18\nvalue = [16, 2]'),
 Text(747.4165076335878, 166.7863636363636, 'Age <= 42.5\nentropy =
0.764\nsamples = 9\nvalue = [7, 2]'),
 Text(736.7676526717557, 129.7227272727273, 'Fare <= 35.25\nentropy =
0.863\nsamples = 7\nvalue = [5, 2]'),
 Text(726.1187977099237, 92.65909090909088, 'Embarked=C <= 0.5\nentropy =
0.65\nsamples = 6\nvalue = [5, 1]'),
 Text(715.4699427480916, 55.59545454545457, 'entropy = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(736.7676526717557, 55.59545454545457, 'Fare <= 30.848\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(726.1187977099237, 18.531818181818153, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(747.4165076335878, 18.531818181818153, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(747.4165076335878, 92.65909090909088, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(758.0653625954199, 129.7227272727273, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(768.7142175572519, 166.7863636363636, 'entropy = 0.0\nsamples = 9\nvalue =
[9, 0]'),
 Text(779.363072519084, 203.85000000000002, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(845.9184160305343, 240.91363636363633, 'Fare <= 59.087\nentropy =
0.981\nsamples = 31\nvalue = [18, 13]'),
 Text(800.6607824427481, 203.85000000000002, 'Age <= 29.5\nentropy =
0.918\nsamples = 12\nvalue = [4, 8]'),
 Text(790.011927480916, 166.7863636363636, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(811.3096374045801, 166.7863636363636, 'Age <= 49.5\nentropy =
0.845\nsamples = 11\nvalue = [3, 8]'),
 Text(800.6607824427481, 129.7227272727273, 'Age <= 39.5\nentropy =
0.722\nsamples = 10\nvalue = [2, 8]'),
 Text(790.011927480916, 92.65909090909088, 'Age <= 34.5\nentropy =
0.918\nsamples = 6\nvalue = [2, 4]'),
 Text(779.363072519084, 55.59545454545457, 'entropy = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(800.6607824427481, 55.59545454545457, 'Parch <= 0.5\nentropy =
0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(790.011927480916, 18.531818181818153, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(811.3096374045801, 18.531818181818153, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(811.3096374045801, 92.65909090909088, 'entropy = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(821.9584923664122, 129.7227272727273, 'entropy = 0.0\nsamples = 1\nvalue =
```

```
[1, 0]'),
 Text(891.1760496183206, 203.85000000000002, 'Fare <= 86.29\nentropy =
0.831\nsamples = 19\nvalue = [14, 5]'),
 Text(853.9050572519084, 166.7863636363636, 'Age <= 47.0\nentropy =
0.469\nsamples = 10\nvalue = [9, 1]'),
 Text(843.2562022900763, 129.7227272727273, 'entropy = 0.0\nsamples = 7\nvalue =
[7, 0]'),
 Text(864.5539122137404, 129.7227272727273, 'Parch <= 0.5\nentropy =
0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(853.9050572519084, 92.65909090909088, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(875.2027671755725, 92.65909090909088, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(928.4470419847328, 166.7863636363636, 'Embarked=S <= 0.5\nentropy =
0.991\nsamples = 9\nvalue = [5, 4]'),
 Text(907.1493320610687, 129.7227272727273, 'Fare <= 89.552\nentropy =
0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(896.5004770992366, 92.65909090909088, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(917.7981870229007, 92.65909090909088, 'entropy = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(949.744751908397, 129.7227272727273, 'SibSp <= 0.5\nentropy =
0.811\nsamples = 4\nvalue = [1, 3]'),
 Text(939.095896946565, 92.65909090909088, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(960.3936068702291, 92.65909090909088, 'entropy = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(793.3396946564885, 315.04090909090905, 'entropy = 0.0\nsamples = 2\nvalue
= [0, 2]'),
 Text(857.232824427481, 537.4227272727272, 'Age <= 75.5\nentropy =
0.575\nsamples = 22\nvalue = [19, 3]'),
 Text(846.5839694656488, 500.3590909090909, 'Embarked=S <= 0.5\nentropy =
0.454\nsamples = 21\nvalue = [19, 2]'),
 Text(835.9351145038167, 463.2954545454545, 'SibSp <= 0.5\nentropy =
0.811\nsamples = 8\nvalue = [6, 2]'),
 Text(825.2862595419847, 426.2318181818182, 'Age <= 57.0\nentropy =
0.592\nsamples = 7\nvalue = [6, 1]'),
 Text(814.6374045801526, 389.1681818181818, 'Fare <= 33.098\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(803.9885496183206, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(825.2862595419847, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(835.9351145038167, 389.1681818181818, 'entropy = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(846.5839694656488, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
```

```
 Text(857.232824427481, 463.2954545454545, 'entropy = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(867.881679389313, 500.3590909090909, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(757.3998091603054, 685.6772727272727, 'Age <= 3.5\nentropy =
0.258\nsamples = 23\nvalue = [22, 1]'),
 Text(746.7509541984733, 648.6136363636364, 'Age <= 2.5\nentropy =
0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(736.1020992366413, 611.55, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(757.3998091603054, 611.55, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(768.0486641221374, 648.6136363636364, 'entropy = 0.0\nsamples = 18\nvalue
= [18, 0]'),
 Text(1127.7802958015268, 759.8045454545454, 'Pclass <= 2.5\nentropy =
0.824\nsamples = 314\nvalue = [81, 233]'),
 Text(969.0458015267176, 722.7409090909091, 'Fare <= 28.856\nentropy =
0.299\nsamples = 170\nvalue = [9, 161]'),
 Text(947.7480916030535, 685.6772727272727, 'Fare <= 28.231\nentropy =
0.469\nsamples = 70\nvalue = [7, 63]'),
 Text(937.0992366412214, 648.6136363636364, 'Age <= 23.5\nentropy =
0.426\nsamples = 69\nvalue = [6, 63]'),
 Text(926.4503816793894, 611.55, 'entropy = 0.0\nsamples = 15\nvalue = [0,
15]'),
 Text(947.7480916030535, 611.55, 'Age <= 27.5\nentropy = 0.503\nsamples =
54\nvalue = [6, 48]'),
 Text(921.1259541984733, 574.4863636363636, 'Age <= 25.5\nentropy =
0.845\nsamples = 11\nvalue = [3, 8]'),
 Text(899.8282442748092, 537.4227272727272, 'Fare <= 13.75\nentropy =
0.592\nsamples = 7\nvalue = [1, 6]'),
 Text(889.1793893129772, 500.3590909090909, 'entropy = 1.0\nsamples = 2\nvalue =
[1, 1]'),
 Text(910.4770992366413, 500.3590909090909, 'entropy = 0.0\nsamples = 5\nvalue =
[0, 5]'),
 Text(942.4236641221374, 537.4227272727272, 'Fare <= 17.429\nentropy =
1.0\nsamples = 4\nvalue = [2, 2]'),
 Text(931.7748091603054, 500.3590909090909, 'entropy = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(953.0725190839695, 500.3590909090909, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(974.3702290076336, 574.4863636363636, 'Age <= 37.5\nentropy =
0.365\nsamples = 43\nvalue = [3, 40]'),
 Text(963.7213740458016, 537.4227272727272, 'entropy = 0.0\nsamples = 24\nvalue
= [0, 24]'),
 Text(985.0190839694657, 537.4227272727272, 'Age <= 39.0\nentropy =
0.629\nsamples = 19\nvalue = [3, 16]'),
 Text(974.3702290076336, 500.3590909090909, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(995.6679389312977, 500.3590909090909, 'Age <= 56.0\nentropy =
```

```
0.503\nsamples = 18\nvalue = [2, 16]'),
 Text(974.3702290076336, 463.2954545454545, 'SibSp <= 0.5\nentropy =
0.337\nsamples = 16\nvalue = [1, 15]'),
 Text(963.7213740458016, 426.2318181818182, 'entropy = 0.0\nsamples = 12\nvalue
= [0, 12]'),
 Text(985.0190839694657, 426.2318181818182, 'Parch <= 0.5\nentropy =
0.811\nsamples = 4\nvalue = [1, 3]'),
 Text(974.3702290076336, 389.1681818181818, 'Age <= 43.0\nentropy = 1.0\nsamples
= 2\nvalue = [1, 1]'),
 Text(963.7213740458016, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(985.0190839694657, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(995.6679389312977, 389.1681818181818, 'entropy = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(1016.9656488549618, 463.2954545454545, 'Age <= 57.5\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(1006.3167938931298, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1027.614503816794, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(958.3969465648855, 648.6136363636364, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(990.3435114503817, 685.6772727272727, 'Age <= 2.5\nentropy =
0.141\nsamples = 100\nvalue = [2, 98]'),
 Text(979.6946564885496, 648.6136363636364, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(1000.9923664122138, 648.6136363636364, 'Parch <= 1.5\nentropy =
0.081\nsamples = 99\nvalue = [1, 98]'),
 Text(990.3435114503817, 611.55, 'entropy = 0.0\nsamples = 84\nvalue = [0,
84]'),
 Text(1011.6412213740458, 611.55, 'Age <= 24.5\nentropy = 0.353\nsamples =
15\nvalue = [1, 14]'),
 Text(1000.9923664122138, 574.4863636363636, 'entropy = 0.0\nsamples = 10\nvalue
= [0, 10]'),
 Text(1022.2900763358779, 574.4863636363636, 'Age <= 28.0\nentropy =
0.722\nsamples = 5\nvalue = [1, 4]'),
 Text(1011.6412213740458, 537.4227272727272, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1032.93893129771, 537.4227272727272, 'entropy = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(1286.5147900763359, 722.7409090909091, 'Fare <= 23.35\nentropy =
1.0\nsamples = 144\nvalue = [72, 72]'),
 Text(1220.625, 685.6772727272727, 'Embarked=S <= 0.5\nentropy = 0.977\nsamples
= 117\nvalue = [48, 69]'),
 Text(1131.4408396946565, 648.6136363636364, 'Fare <= 15.621\nentropy =
0.877\nsamples = 54\nvalue = [16, 38]'),
```

```
 Text(1120.7919847328244, 611.55, 'Fare <= 13.935\nentropy = 0.932\nsamples =
46\nvalue = [16, 30]'),
 Text(1064.8854961832062, 574.4863636363636, 'Age <= 17.5\nentropy =
0.758\nsamples = 32\nvalue = [7, 25]'),
 Text(1054.2366412213742, 537.4227272727272, 'entropy = 0.0\nsamples = 10\nvalue
= [0, 10]'),
 Text(1075.5343511450383, 537.4227272727272, 'Age <= 38.5\nentropy =
0.902\nsamples = 22\nvalue = [7, 15]'),
 Text(1064.8854961832062, 500.3590909090909, 'Fare <= 6.987\nentropy =
0.949\nsamples = 19\nvalue = [7, 12]'),
 Text(1054.2366412213742, 463.2954545454545, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1075.5343511450383, 463.2954545454545, 'Parch <= 1.0\nentropy =
0.918\nsamples = 18\nvalue = [6, 12]'),
 Text(1064.8854961832062, 426.2318181818182, 'Fare <= 8.008\nentropy =
0.874\nsamples = 17\nvalue = [5, 12]'),
 Text(1054.2366412213742, 389.1681818181818, 'Age <= 31.0\nentropy =
0.811\nsamples = 16\nvalue = [4, 12]'),
 Text(1043.587786259542, 352.1045454545455, 'Age <= 23.5\nentropy =
0.971\nsamples = 10\nvalue = [4, 6]'),
 Text(1032.93893129771, 315.04090909090905, 'Fare <= 7.683\nentropy =
0.811\nsamples = 8\nvalue = [2, 6]'),
 Text(1022.2900763358779, 277.97727272727275, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1043.587786259542, 277.97727272727275, 'Age <= 20.5\nentropy =
0.592\nsamples = 7\nvalue = [1, 6]'),
 Text(1032.93893129771, 240.91363636363633, 'entropy = 0.0\nsamples = 5\nvalue =
[0, 5]'),
 Text(1054.2366412213742, 240.91363636363633, 'Age <= 21.5\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(1043.587786259542, 203.85000000000002, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1064.8854961832062, 203.85000000000002, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1054.2366412213742, 315.04090909090905, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(1064.8854961832062, 352.1045454545455, 'entropy = 0.0\nsamples = 6\nvalue
= [0, 6]'),
 Text(1075.5343511450383, 389.1681818181818, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1086.1832061068703, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1086.1832061068703, 500.3590909090909, 'entropy = 0.0\nsamples = 3\nvalue
= [0, 3]'),
 Text(1176.6984732824428, 574.4863636363636, 'Embarked=Q <= 0.5\nentropy =
0.94\nsamples = 14\nvalue = [9, 5]'),
 Text(1150.0763358778627, 537.4227272727272, 'Parch <= 1.5\nentropy =
```

```
0.722\nsamples = 10\nvalue = [8, 2]'),
 Text(1128.7786259541986, 500.3590909090909, 'Age <= 16.0\nentropy =
0.544\nsamples = 8\nvalue = [7, 1]'),
 Text(1118.1297709923665, 463.2954545454545, 'Age <= 14.5\nentropy =
0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(1107.4809160305344, 426.2318181818182, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(1128.7786259541986, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1139.4274809160306, 463.2954545454545, 'entropy = 0.0\nsamples = 5\nvalue
= [5, 0]'),
 Text(1171.3740458015268, 500.3590909090909, 'Age <= 27.5\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(1160.7251908396947, 463.2954545454545, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1182.0229007633588, 463.2954545454545, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1203.320610687023, 537.4227272727272, 'Parch <= 0.5\nentropy =
0.811\nsamples = 4\nvalue = [1, 3]'),
 Text(1192.671755725191, 500.3590909090909, 'entropy = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(1213.969465648855, 500.3590909090909, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(1142.0896946564885, 611.55, 'entropy = 0.0\nsamples = 8\nvalue = [0, 8]'),
 Text(1309.8091603053435, 648.6136363636364, 'Age <= 36.5\nentropy =
1.0\nsamples = 63\nvalue = [32, 31]'),
 Text(1288.5114503816794, 611.55, 'Age <= 32.5\nentropy = 0.998\nsamples =
57\nvalue = [27, 30]'),
 Text(1277.8625954198474, 574.4863636363636, 'Fare <= 7.763\nentropy =
1.0\nsamples = 54\nvalue = [27, 27]'),
 Text(1245.9160305343512, 537.4227272727272, 'Age <= 23.5\nentropy =
0.65\nsamples = 6\nvalue = [1, 5]'),
 Text(1235.2671755725191, 500.3590909090909, 'entropy = 0.0\nsamples = 5\nvalue
= [0, 5]'),
 Text(1256.5648854961833, 500.3590909090909, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1309.8091603053435, 537.4227272727272, 'Fare <= 10.825\nentropy =
0.995\nsamples = 48\nvalue = [26, 22]'),
 Text(1277.8625954198474, 500.3590909090909, 'Fare <= 10.152\nentropy =
0.918\nsamples = 27\nvalue = [18, 9]'),
 Text(1267.2137404580153, 463.2954545454545, 'Parch <= 0.5\nentropy =
0.954\nsamples = 24\nvalue = [15, 9]'),
 Text(1256.5648854961833, 426.2318181818182, 'Fare <= 9.84\nentropy =
0.902\nsamples = 22\nvalue = [15, 7]'),
 Text(1245.9160305343512, 389.1681818181818, 'Fare <= 8.767\nentropy =
0.863\nsamples = 21\nvalue = [15, 6]'),
 Text(1235.2671755725191, 352.1045454545455, 'Fare <= 8.673\nentropy =
```

```
0.937\nsamples = 17\nvalue = [11, 6]'),
 Text(1224.618320610687, 315.04090909090905, 'Fare <= 7.988\nentropy =
0.896\nsamples = 16\nvalue = [11, 5]'),
 Text(1213.969465648855, 277.97727272727275, 'Age <= 18.5\nentropy =
0.98\nsamples = 12\nvalue = [7, 5]'),
 Text(1203.320610687023, 240.91363636363633, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(1224.618320610687, 240.91363636363633, 'Age <= 22.5\nentropy =
1.0\nsamples = 10\nvalue = [5, 5]'),
 Text(1213.969465648855, 203.85000000000002, 'entropy = 0.0\nsamples = 2\nvalue
= [0, 2]'),
 Text(1235.2671755725191, 203.85000000000002, 'Age <= 25.5\nentropy =
0.954\nsamples = 8\nvalue = [5, 3]'),
 Text(1224.618320610687, 166.7863636363636, 'entropy = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(1245.9160305343512, 166.7863636363636, 'Age <= 27.5\nentropy =
0.971\nsamples = 5\nvalue = [2, 3]'),
 Text(1235.2671755725191, 129.7227272727273, 'entropy = 0.0\nsamples = 3\nvalue
= [0, 3]'),
 Text(1256.5648854961833, 129.7227272727273, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(1235.2671755725191, 277.97727272727275, 'entropy = 0.0\nsamples = 4\nvalue
= [4, 0]'),
 Text(1245.9160305343512, 315.04090909090905, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1256.5648854961833, 352.1045454545455, 'entropy = 0.0\nsamples = 4\nvalue
= [4, 0]'),
 Text(1267.2137404580153, 389.1681818181818, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1277.8625954198474, 426.2318181818182, 'entropy = 0.0\nsamples = 2\nvalue
= [0, 2]'),
 Text(1288.5114503816794, 463.2954545454545, 'entropy = 0.0\nsamples = 3\nvalue
= [3, 0]'),
 Text(1341.7557251908397, 500.3590909090909, 'Fare <= 17.25\nentropy =
0.959\nsamples = 21\nvalue = [8, 13]'),
 Text(1309.8091603053435, 463.2954545454545, 'Age <= 25.0\nentropy =
0.619\nsamples = 13\nvalue = [2, 11]'),
 Text(1299.1603053435115, 426.2318181818182, 'entropy = 0.0\nsamples = 8\nvalue
= [0, 8]'),
 Text(1320.4580152671756, 426.2318181818182, 'SibSp <= 0.5\nentropy =
0.971\nsamples = 5\nvalue = [2, 3]'),
 Text(1309.8091603053435, 389.1681818181818, 'entropy = 0.0\nsamples = 3\nvalue
= [0, 3]'),
 Text(1331.1068702290077, 389.1681818181818, 'entropy = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(1373.7022900763359, 463.2954545454545, 'Fare <= 21.55\nentropy =
0.811\nsamples = 8\nvalue = [6, 2]'),
```

```
 Text(1363.0534351145038, 426.2318181818182, 'Age <= 30.0\nentropy =
0.592\nsamples = 7\nvalue = [6, 1]'),
 Text(1352.4045801526718, 389.1681818181818, 'entropy = 0.0\nsamples = 5\nvalue
= [5, 0]'),
 Text(1373.7022900763359, 389.1681818181818, 'Parch <= 0.5\nentropy =
1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(1363.0534351145038, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(1384.351145038168, 352.1045454545455, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(1384.351145038168, 426.2318181818182, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(1299.1603053435115, 574.4863636363636, 'entropy = 0.0\nsamples = 3\nvalue
= [0, 3]'),
 Text(1331.1068702290077, 611.55, 'Age <= 55.0\nentropy = 0.65\nsamples =
6\nvalue = [5, 1]'),
 Text(1320.4580152671756, 574.4863636363636, 'entropy = 0.0\nsamples = 5\nvalue
= [5, 0]'),
 Text(1341.7557251908397, 574.4863636363636, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1352.4045801526718, 685.6772727272727, 'Parch <= 0.5\nentropy =
0.503\nsamples = 27\nvalue = [24, 3]'),
 Text(1341.7557251908397, 648.6136363636364, 'entropy = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(1363.0534351145038, 648.6136363636364, 'Fare <= 31.331\nentropy =
0.391\nsamples = 26\nvalue = [24, 2]'),
 Text(1352.4045801526718, 611.55, 'entropy = 0.0\nsamples = 15\nvalue = [15,
0]'),
 Text(1373.7022900763359, 611.55, 'Fare <= 32.881\nentropy = 0.684\nsamples =
11\nvalue = [9, 2]'),
 Text(1363.0534351145038, 574.4863636363636, 'entropy = 0.0\nsamples = 2\nvalue
= [0, 2]'),
 Text(1384.351145038168, 574.4863636363636, 'entropy = 0.0\nsamples = 9\nvalue =
[9, 0]')]
```