Assignment_6_2_Raghuwanshi_Prashant_DSC550

October 9, 2021

Assignment: 6.2 Exercise: Unsupervised Learning Part 1: Collaborative Filtering and Frequent Pattern Mining

Name: Prashant Raghuwanshi

Date: 10/09/2021

Course: DSC550-T301 Data Mining (2221-1)

Case Study: Analyze data to predict who will Survive the Titanic

[1]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144:
FutureWarning: The sklearn.metrics.classification module is  deprecated in
version 0.22 and will be removed in version 0.24. The corresponding classes /
functions should instead be imported from sklearn.metrics. Anything that cannot
be imported from sklearn.metrics is now part of the private API.
  warnings.warn(message, FutureWarning)
```

[2]:

```
# 1.        Load the data from the "train.csv" file into a DataFrame.
addr1 = "C:/Users/dell/Documents/Machine_learning_assigments/week-6/train.csv"
df_train = pd.read_csv(addr1)
```

[3]:

```
# 2.        Display the dimensions of the file (so you’ll have a good idea the␣
 ↪amount of data you are working with.
print("The dimension of the table is: ", df_train.shape)
```

```
The dimension of the table is:  (891, 12)
```

[4]:

```
# 3.        Display the first 5 rows of data so you can see the column headings␣
 ↪and the type of data for each column.
print(df_train.head(5))
# a.        Notice that Survived is represented as a 1 or 0
# b.        Notice that missing data is represented as "NaN"
# c.        The Survived variable will be the "target" and the other variables␣
 ↪will be the "features"
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         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

   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
```

[5]:
```
#4.          Think about some questions that might help you predict who will␣
 ↪survive:
# a.          What do the variables look like? For example, are they numerical or␣
 ↪categorical data. If they are numerical, what are their distribution; if␣
 ↪they are categorical, how many are they in different categories?
# b.          Are the numerical variables correlated?
# c.          Are the distributions of numerical variables the same or different␣
 ↪among survived and not survived? Is the survival rate different for␣
 ↪different values? For example, were people more likely to survive if they␣
 ↪were younger?
# d.          Are there different survival rates in different categories? For␣
 ↪example, did more women survived than man?
```

[6]:
```
#5.          Look at summary information about your data (total, mean, min, max,␣
 ↪freq, unique, etc.)  Does this present any more questions for you?  Does it␣
 ↪lead you to a conclusion yet?
print("\nDescribe Data\n")
print(df_train.describe())
print("\nSummarized Data\n")
print(df_train.describe(include=['O']))
```

```
Describe Data

       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
```

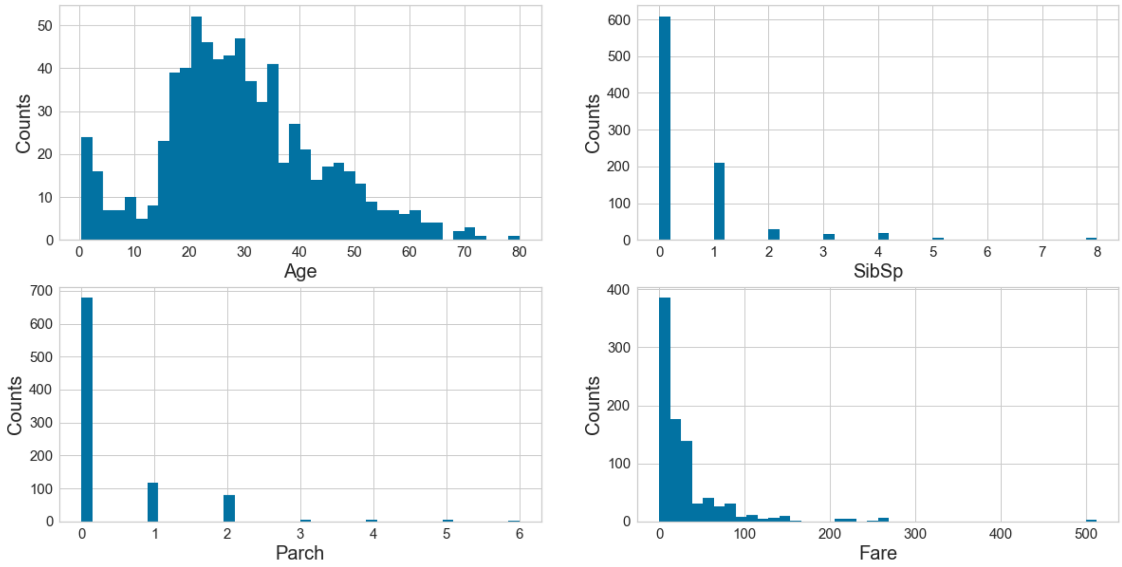
```
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200

Summarized Data

                           Name   Sex    Ticket Cabin Embarked
count                       891   891       891   204      889
unique                      891     2       681   147        3
top     Cribb, Mr. John Hatfield  male  CA. 2343    G6        S
freq                          1   577         7     4      644
```

[7]:

```
#6.         Make some histograms of your data ("A picture is worth a thousand␣
 ↪words!")
# Specify the features of interest
num_features = ['Age', 'SibSp', 'Parch', 'Fare']
xaxes = num_features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts']
```

[8]:

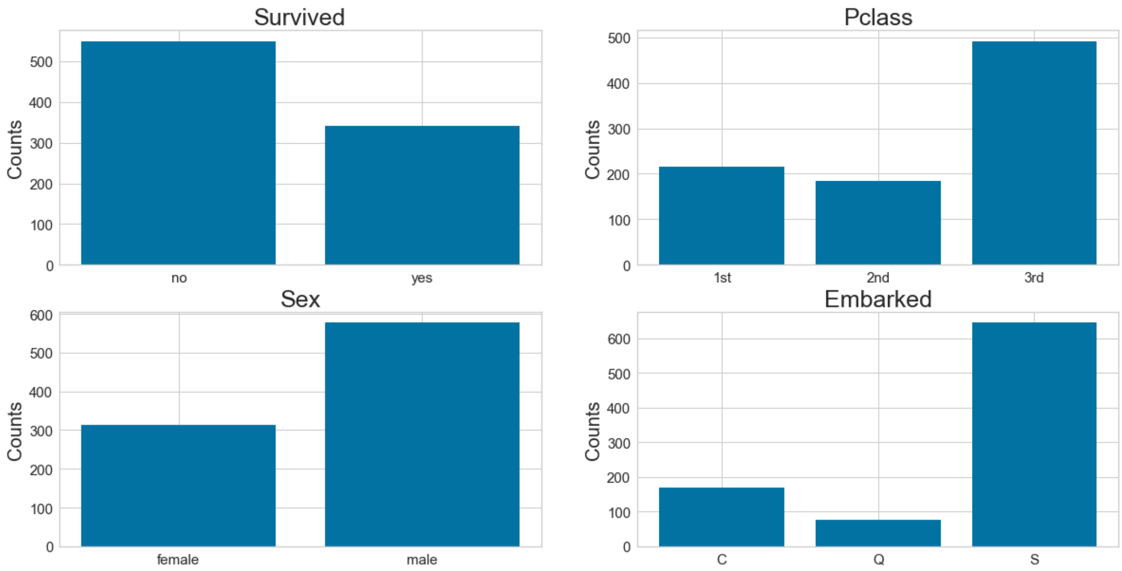
```
# set up the figure size
plt.rcParams['figure.figsize'] = (20, 10)
# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)
# draw histograms
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(df_train[num_features[idx]].dropna(), bins=40)
    ax.set_xlabel(xaxes[idx], fontsize=20)
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)
plt.show()
```



[9]:

```
#7.        Make some bar charts for variables with only a few options.
#%matplotlib inline
plt.rcParams['figure.figsize'] = (20, 10)
# make subplots
fig, axes = plt.subplots(nrows = 2, ncols = 2)
# make the data read to feed into the visulizer
X_Survived = df_train.replace({'Survived': {1: 'yes', 0: 'no'}}).
 ↪groupby('Survived').size().reset_index(name='Counts')['Survived']
Y_Survived = df_train.replace({'Survived': {1: 'yes', 0: 'no'}}).
 ↪groupby('Survived').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[0, 0].bar(X_Survived, Y_Survived)
axes[0, 0].set_title('Survived', fontsize=25)
axes[0, 0].set_ylabel('Counts', fontsize=20)
axes[0, 0].tick_params(axis='both', labelsize=15)
# make the data read to feed into the visulizer
X_Pclass = df_train.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).
 ↪groupby('Pclass').size().reset_index(name='Counts')['Pclass']
Y_Pclass = df_train.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).
 ↪groupby('Pclass').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[0, 1].bar(X_Pclass, Y_Pclass)
axes[0, 1].set_title('Pclass', fontsize=25)
axes[0, 1].set_ylabel('Counts', fontsize=20)
axes[0, 1].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
X_Sex = df_train.groupby('Sex').size().reset_index(name='Counts')['Sex']
```

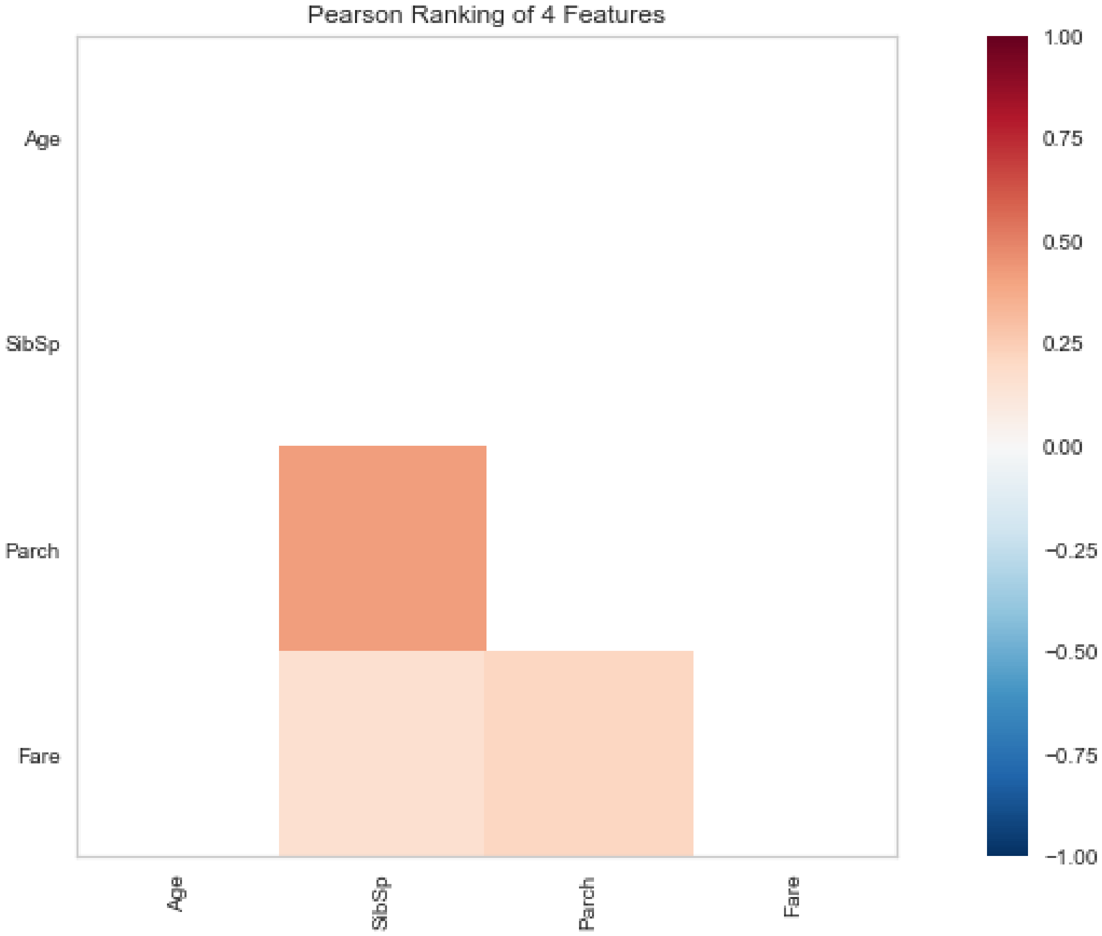
```
Y_Sex = df_train.groupby('Sex').size().reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 0].bar(X_Sex, Y_Sex)
axes[1, 0].set_title('Sex', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)

# make the data read to feed into the visulizer
X_Embarked = df_train.groupby('Embarked').size().
 ↪reset_index(name='Counts')['Embarked']
Y_Embarked = df_train.groupby('Embarked').size().
 ↪reset_index(name='Counts')['Counts']
# make the bar plot
axes[1, 1].bar(X_Embarked, Y_Embarked)
axes[1, 1].set_title('Embarked', fontsize=25)
axes[1, 1].set_ylabel('Counts', fontsize=20)
axes[1, 1].tick_params(axis='both', labelsize=15)
#plt.show()
```



[10]:

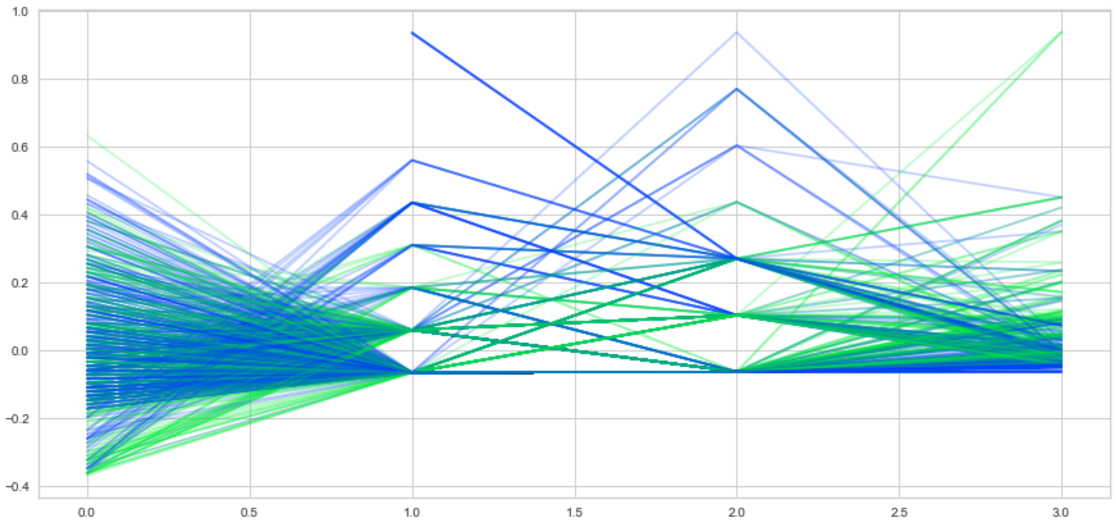
```
#8.         To see if the data is correlated, make some Pearson Ranking charts
#Step 8: Pearson Ranking
#set up the figure size
#%matplotlib inline
plt.rcParams['figure.figsize'] = (15, 7)

# import the package for visulization of the correlation
from yellowbrick.features import Rank2D
```

```
# extract the numpy arrays from the data frame
X = df_train[num_features].to_numpy()

# instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=num_features, algorithm='pearson')
visualizer.fit(X)                    # Fit the data to the visualizer
visualizer.transform(X)              # Transform the data
visualizer.poof(outpath="d://pcoords1.png") # Draw/show/poof the data
#plt.show()
```



[11]:
```
#9.         Use Parallel Coordinates visualization to compare the distributions␣
 ↪of numerical variables between passengers that survived and those that did␣
 ↪not survive.
# a.         That’s a cool chart, isn’t it?!  Passengers traveling with␣
 ↪siblings on the boat have a higher death rate and passengers who paid a␣
 ↪higher fare had a higher survival rate.
```

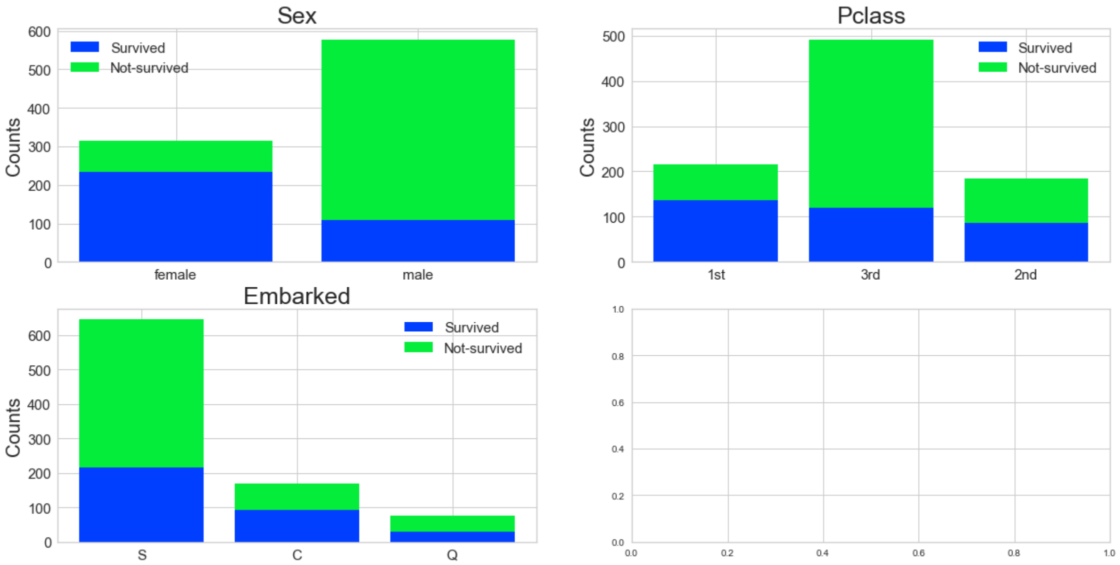
```
# Step 9:  Compare variables against Survived and Not Survived
#set up the figure size
#%matplotlib inline
plt.rcParams['figure.figsize'] = (15, 7)
plt.rcParams['font.size'] = 50

# setup the color for yellowbrick visulizer
from yellowbrick.style import set_palette
set_palette('sns_bright')

# import packages
from yellowbrick.features import ParallelCoordinates
# Specify the features of interest and the classes of the target
classes = ['Not-survived', 'Survived']
num_features = ['Age', 'SibSp', 'Parch', 'Fare']

# copy data to a new dataframe
data_norm = df_train.copy()
# normalize data to 0-1 range
for feature in num_features:
    data_norm[feature] = (df_train[feature] - df_train[feature].
 ↪mean(skipna=True)) / (df_train[feature].max(skipna=True) - df_train[feature].
 ↪min(skipna=True))

# Extract the numpy arrays from the data frame
X = data_norm[num_features].to_numpy()
y = df_train.Survived.to_numpy()

# Instantiate the visualizer
# Instantiate the visualizer
visualizer = ParallelCoordinates(classes=classes, features=num_features)


visualizer.fit(X, y)      # Fit the data to the visualizer
visualizer.transform(X)   # Transform the data
#visualizer.poof(outpath="d://pcoords2.png") # Draw/show/poof the data
plt.show();
```



```
[12]: #10.        Use Stack Bar Charts to compare passengers who survived to␣
       ↪passengers who didn’t survive based on the other variables.
      # a.        More females survived than men.  3rd Class Tickets had a lower␣
       ↪survival rate.  Also, Embarkation from Southampton port had a lower survival␣
       ↪rate.

      # Step 10 - stacked bar charts to compare survived/not survived
      #set up the figure size
      #%matplotlib inline
      plt.rcParams['figure.figsize'] = (20, 10)

      # make subplots
      fig, axes = plt.subplots(nrows = 2, ncols = 2)

      # make the data read to feed into the visulizer
      Sex_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
       ↪'Not-survived'}})[df_train['Survived']==1]['Sex'].value_counts()
      Sex_not_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
       ↪'Not-survived'}})[df_train['Survived']==0]['Sex'].value_counts()
      Sex_not_survived = Sex_not_survived.reindex(index = Sex_survived.index)
      # make the bar plot
      p1 = axes[0, 0].bar(Sex_survived.index, Sex_survived.values)
      p2 = axes[0, 0].bar(Sex_not_survived.index, Sex_not_survived.values,␣
       ↪bottom=Sex_survived.values)
      axes[0, 0].set_title('Sex', fontsize=25)
      axes[0, 0].set_ylabel('Counts', fontsize=20)
      axes[0, 0].tick_params(axis='both', labelsize=15)
      axes[0, 0].legend((p1[0], p2[0]), ('Survived', 'Not-survived'), fontsize = 15)
```

```
# make the data read to feed into the visualizer
Pclass_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
 ↪'Not-survived'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3:␣
 ↪'3rd'}})[df_train['Survived']==1]['Pclass'].value_counts()
Pclass_not_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
 ↪'Not-survived'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3:␣
 ↪'3rd'}})[df_train['Survived']==0]['Pclass'].value_counts()
Pclass_not_survived = Pclass_not_survived.reindex(index = Pclass_survived.index)
# make the bar plot
p3 = axes[0, 1].bar(Pclass_survived.index, Pclass_survived.values)
p4 = axes[0, 1].bar(Pclass_not_survived.index, Pclass_not_survived.values,␣
 ↪bottom=Pclass_survived.values)
axes[0, 1].set_title('Pclass', fontsize=25)
axes[0, 1].set_ylabel('Counts', fontsize=20)
axes[0, 1].tick_params(axis='both', labelsize=15)
axes[0, 1].legend((p3[0], p4[0]), ('Survived', 'Not-survived'), fontsize = 15)

# make the data read to feed into the visualizer
Embarked_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
 ↪'Not-survived'}})[df_train['Survived']==1]['Embarked'].value_counts()
Embarked_not_survived = df_train.replace({'Survived': {1: 'Survived', 0:␣
 ↪'Not-survived'}})[df_train['Survived']==0]['Embarked'].value_counts()
Embarked_not_survived = Embarked_not_survived.reindex(index = Embarked_survived.
 ↪index)
# make the bar plot
p5 = axes[1, 0].bar(Embarked_survived.index, Embarked_survived.values)
p6 = axes[1, 0].bar(Embarked_not_survived.index, Embarked_not_survived.values,␣
 ↪bottom=Embarked_survived.values)
axes[1, 0].set_title('Embarked', fontsize=25)
axes[1, 0].set_ylabel('Counts', fontsize=20)
axes[1, 0].tick_params(axis='both', labelsize=15)
axes[1, 0].legend((p5[0], p6[0]), ('Survived', 'Not-survived'), fontsize = 15)
#plt.show()
```

```
[12]: <matplotlib.legend.Legend at 0x1f0880f4340>
```


Sex
Survived
Not-survived
Counts
600
500
400
300
200
100
0
female
male
Pclass
Survived
Not-survived
Counts
500
400
300
200
100
0
1st
3rd
2nd
Embarked
Survived
Not-survived
Counts
600
500
400
300
200
100
0
S
C
Q
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0