Titanic Tutorial

Part 1 : Graphics Analysis

Part 2 : Feature Reduction (Extraction/Selection)

Part 3 : Filling in Missing Values

Part 1 : Graphics Analysis

In [3]:
```
import pandas as pd
import yellowbrick

import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
#Step 1:  Load data into a dataframe
addr = "Data/train.csv"
data = pd.read_csv(addr)
```

In [5]:
```
# Step 2:  check the dimension of the table
print("The dimension of the table is: ", data.shape)
```

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

In [6]:
```
#Step 3:  Look at the data
print(data.head(5))
```

```
   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
```

In [7]:
```
#Step 5:  what type of variables are in the table
print("Describe Data")
print(data.describe())
```

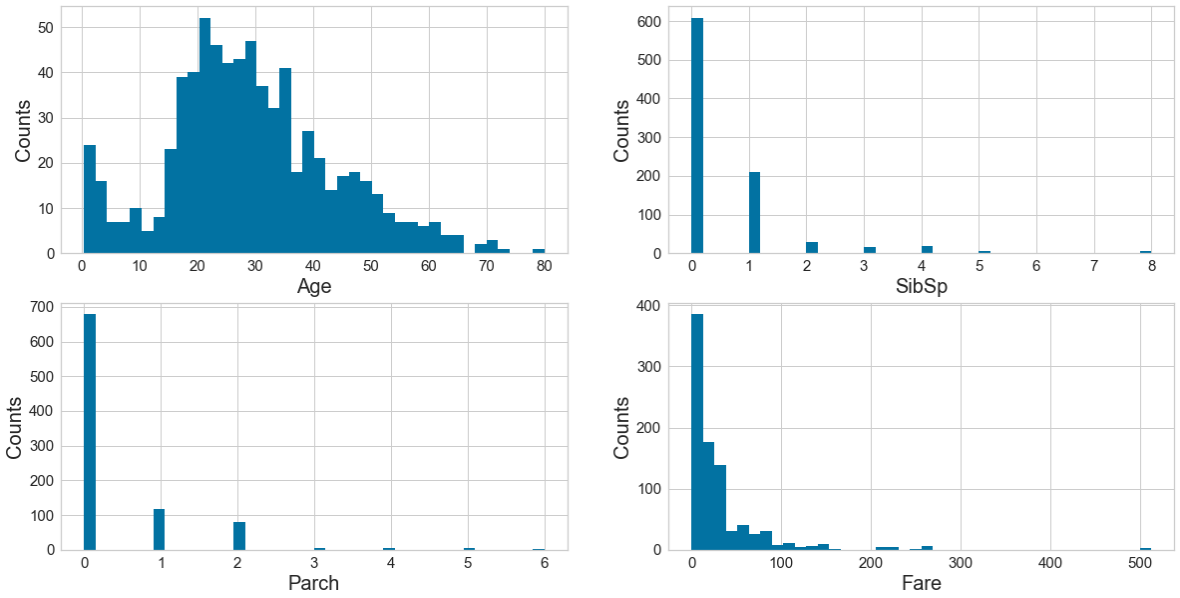
```
Describe Data
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    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
```

In [8]:
```
print("Summarized Data")
print(data.describe(include=['O']))
```

```
Summarized Data
                                          Name    Sex Ticket        Cabin Embar
ked
count                                      891    891    891          204
889
unique                                     891      2    681          147
3
top     Brown, Mr. Thomas William Solomon  male   1601  C23 C25 C27
S
freq                                         1    577      7            4
644
```

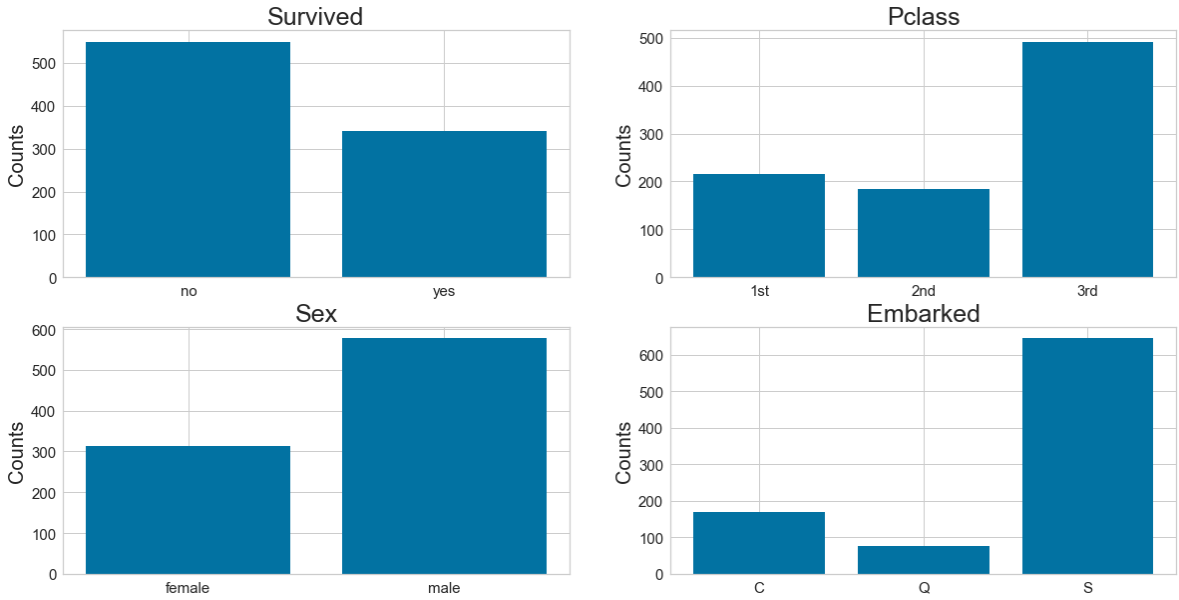
In [9]:
```
#Step 6: import visulization packages
import matplotlib.pyplot as plt
```

In [10]:
```
# set up the figure size
plt.rcParams['figure.figsize'] = (20, 10)

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

# Specify the features of interest
num_features = ['Age', 'SibSp', 'Parch', 'Fare']
xaxes = num_features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts']

# draw histograms
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(data[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()
```



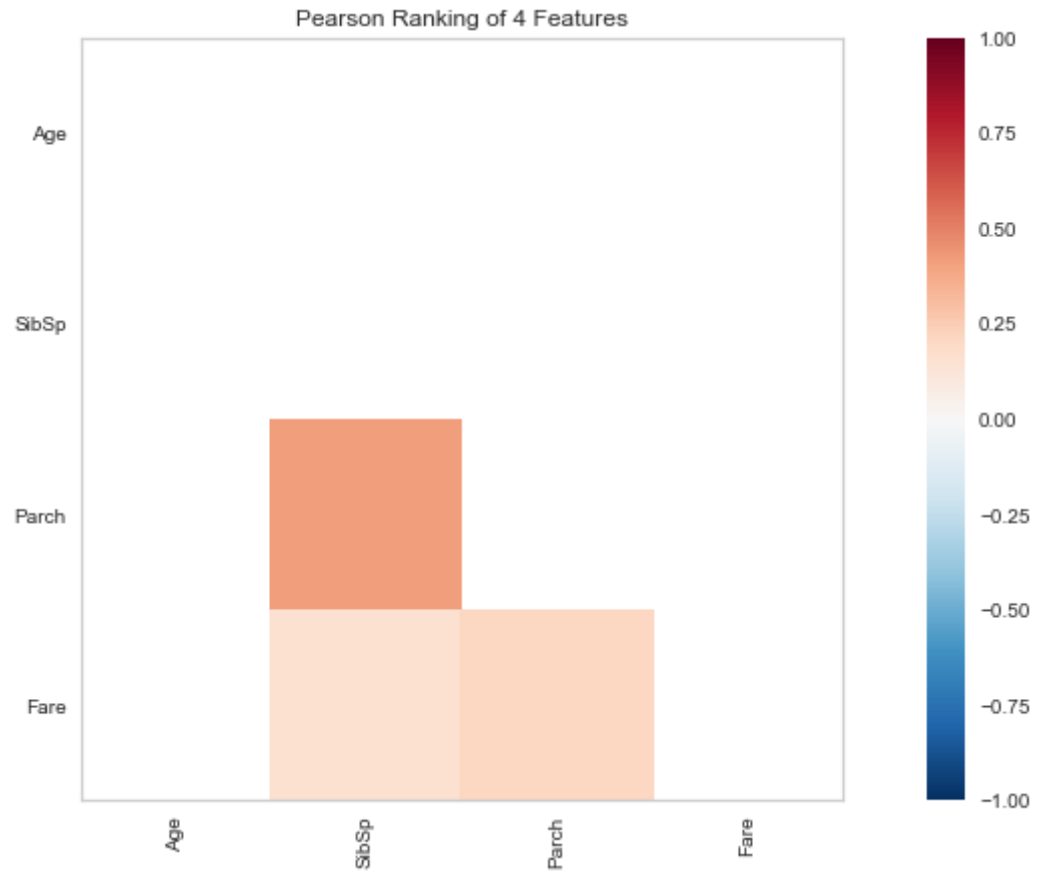
In [11]:

```
#7:  Barcharts: 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
X_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Sur
vived').size().reset_index(name='Counts')['Survived']
Y_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Sur
vived').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 = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).group
by('Pclass').size().reset_index(name='Counts')['Pclass']
Y_Pclass = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).group
by('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 = data.groupby('Sex').size().reset_index(name='Counts')['Sex']
Y_Sex = data.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 = data.groupby('Embarked').size().reset_index(name='Counts')[
'Embarked']
Y_Embarked = data.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()
```


Survived
Counts
0
100
200
300
400
500
no
yes
Pclass
Counts
0
100
200
300
400
500
1st
2nd
3rd
Sex
Counts
0
100
200
300
400
500
600
female
male
Embarked
Counts
0
100
200
300
400
500
600
C
Q
S

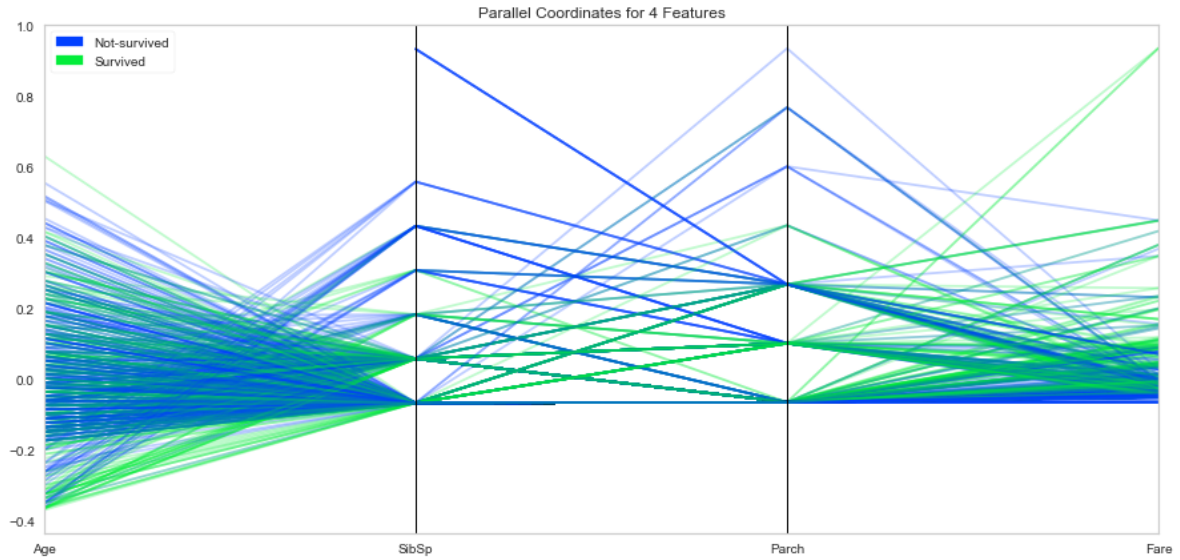
In [12]:

```
#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 = data[num_features].values

# 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="images/pcoords1.png") # Draw/show/poof the data
plt.show()
```



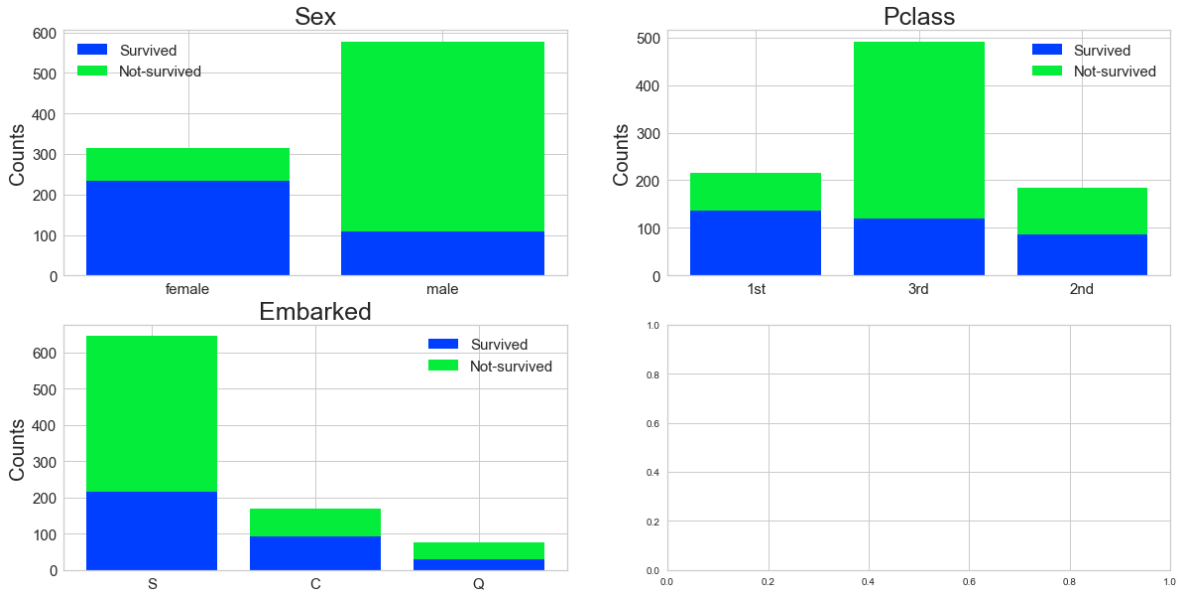
In [13]:

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

# Extract the numpy arrays from the data frame
X = data_norm[num_features].values
y = data.Survived.values

# 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="images/pcoords2.png") # Draw/show/poof the data
plt.show();
```



In [14]:
```
# 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 = data.replace({'Survived': {1: 'Survived', 0: 'Not-survive
d'}})[data['Survived']==1]['Sex'].value_counts()
Sex_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-surv
ived'}})[data['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, bott
om=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 = data.replace({'Survived': {1: 'Survived', 0: 'Not-survi
ved'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})[data['Survive
d']==1]['Pclass'].value_counts()
Pclass_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-s
urvived'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})[data['Sur
vived']==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 visulizer
Embarked_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-sur
vived'}})[data['Survived']==1]['Embarked'].value_counts()
Embarked_not_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not
-survived'}})[data['Survived']==0]['Embarked'].value_counts()
Embarked_not_survived = Embarked_not_survived.reindex(index = Embarked_su
rvived.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.va
lues, 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)

# Nothing to show in [1,1]
plt.show()
```



Part 2 : Feature Reduction (Extraction/Selection)

In [15]:
```
# Load Libraries and data
import pandas as pd
import yellowbrick

import warnings
warnings.filterwarnings("ignore")

#Step 1:  Load data into a dataframe
addr = "Data/train.csv"
data = pd.read_csv(addr)
data.shape
```

Out[15]:
```
(891, 12)
```

In [16]:
```
print(data['Age'].describe())
```

```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

In [17]:
```
data['Age'].median()
```

Out[17]:
```
28.0
```

In [18]:
```
# How many Nan's in Age Column?
data.isnull()['Age'].sum()
```

Out[18]:
```
177
```

In [19]:
```
#Show it.
data['Age']
```

Out[19]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [20]:
```
data['Age'][data['Age'] == 28].count()
```

Out[20]:
```
25
```

In [21]:
```
# Step 11 - fill in missing values and eliminate features
#fill the missing age data with median value
def fill_na_median(data, inplace):
    """
        This function calculate the median of the input data which comes
 in as a pandas series.
    """
    return data.fillna(round(data.median()), inplace=inplace)
```

In [22]:
```
# Apply the nfunction
fill_na_median(data['Age'],inplace=True)
print(data['Age'])
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

In [23]:
```
# check the result
print(data['Age'].describe())
#replacing NaNs with mean decreased the mean from 29.699 to 29.361
```

```
count    891.000000
mean      29.361582
std       13.019697
min        0.420000
25%       22.000000
50%       28.000000
75%       35.000000
max       80.000000
Name: Age, dtype: float64
```

In [24]:
```
# How many Nan's in Age Column?
if (data.isnull()['Age'].sum() == 0):
    print("All NaN's are replace with mean value.")
else:
    print("Looks line removing NaN's didn't work!")
```

```
All NaN's are replace with mean value.
```

In [25]:
```
# After replacing NaN's, total number of people age 28 changed from 25 to
202(25 + 177 NaNs)
data['Age'][data['Age'] == 28].count()
```

Out[25]:
```
202
```

In [26]:
```
# fill with the most represented value
def fill_na_most(data, inplace=True):
    """
        This function Replaces NaN's with letter 'S'.
    """
    return data.fillna('S', inplace=inplace)
```

In [27]:
```
print(data['Embarked'])
```

```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: Embarked, Length: 891, dtype: object
```

In [28]:
```
fill_na_most(data['Embarked'])

# check the result
print(data['Embarked'].describe())
```

```
count     891
unique      3
top         S
freq      646
Name: Embarked, dtype: object
```

In [29]:
```
# import package
import numpy as np

# log-transformation
def log_transformation(data):
    return data.apply(np.log1p)
```

In [30]:
```
print(data['Fare'])
```

```
0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
        ...
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
Name: Fare, Length: 891, dtype: float64
```

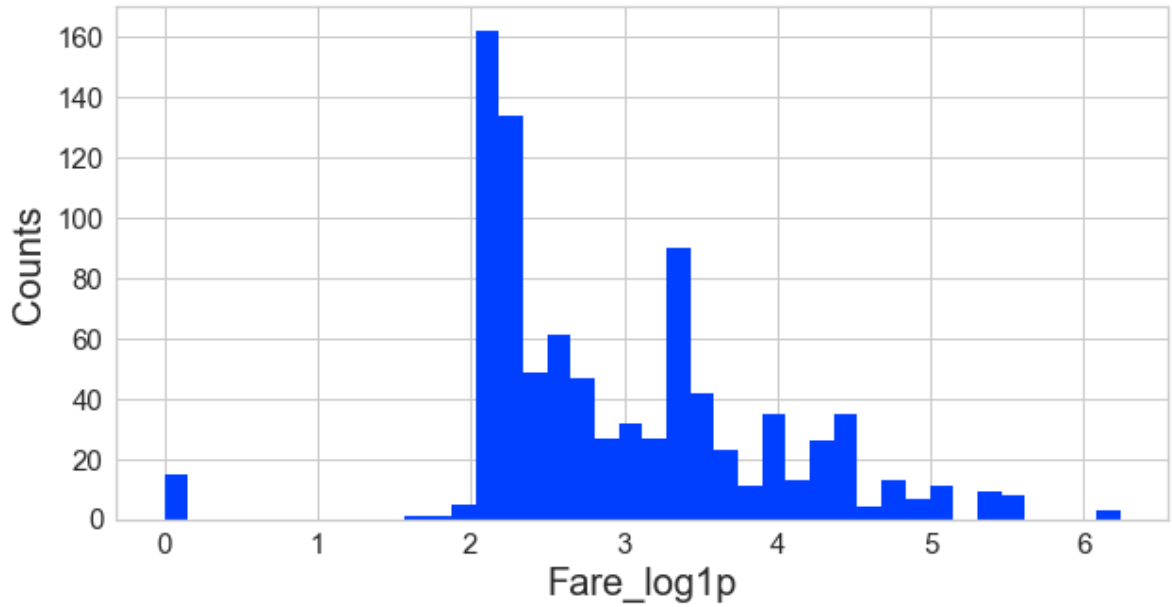
In [31]:
```
# The new column is
data['Fare_log1p'] = log_transformation(data['Fare'])

# check the data
print(data.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.361582    0.523008
std     257.353842    0.486592    0.836071   13.019697    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   35.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare  Fare_log1p
count  891.000000  891.000000  891.000000
mean     0.381594   32.204208    2.962246
std      0.806057   49.693429    0.969048
min      0.000000    0.000000    0.000000
25%      0.000000    7.910400    2.187218
50%      0.000000   14.454200    2.737881
75%      0.000000   31.000000    3.465736
max      6.000000  512.329200    6.240917
```

In [32]:
```
print(data[['Fare','Fare_log1p']])
```

```
        Fare  Fare_log1p
0     7.2500    2.110213
1    71.2833    4.280593
2     7.9250    2.188856
3    53.1000    3.990834
4     8.0500    2.202765
..       ...         ...
886  13.0000    2.639057
887  30.0000    3.433987
888  23.4500    3.196630
889  30.0000    3.433987
890   7.7500    2.169054

[891 rows x 2 columns]
```

In [33]:
```
#Step 12 - adjust skewed data (fare)
#check the distribution using histogram
# set up the figure size
%matplotlib inline
plt.rcParams['figure.figsize'] = (10, 5)

plt.hist(data['Fare_log1p'], bins=40)
plt.xlabel('Fare_log1p', fontsize=20)
plt.ylabel('Counts', fontsize=20)
plt.tick_params(axis='both', labelsize=15)
plt.show()
```



In [34]:
```
#Step 13 - convert categorical data to numbers
#get the categorical data
cat_features = ['Pclass', 'Sex', "Embarked"]
data_cat = data[cat_features]
data_cat = data_cat.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})
# One Hot Encoding
data_cat_dummies = pd.get_dummies(data_cat)
# check the data
print(data_cat_dummies.head(8))
```

```
   Pclass_1st  Pclass_2nd  Pclass_3rd  Sex_female  Sex_male  Embarked_C
\
0           0           0           1           0         1           0
1           1           0           0           1         0           1
2           0           0           1           1         0           0
3           1           0           0           1         0           0
4           0           0           1           0         1           0
5           0           0           1           0         1           0
6           1           0           0           0         1           0
7           0           0           1           0         1           0

   Embarked_Q  Embarked_S
0           0           1
1           0           0
2           0           1
3           0           1
4           0           1
5           1           0
6           0           1
7           0           1
```