EDA_24Jun

January 31, 2018

1 Análisis Exploratorio de Datos

Objetivo del Análisis exploratorio de datos: * Detectar errores en los datos * Checar suposiciones * Seleccionar los modelos apropiados para describir la informacion * Encontrar relaciones entre los datos y variables * Dar una revision y realizar un diagnostico de las relaciones entre las variables que podrian explicar el fenomeno y su resultado.

```
In [1]: #Librerías para trabajar
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.mlab as mlab
        import math
```

```
In [2]: #Librerías para mostrar datos
        from IPython.display import Image
        %matplotlib inline
```

```
In [3]: #Cargamos los datos
        df = pd.read_csv("train.csv")
```

1.0.1 Primer Paso: Explorar el contenido de los datos, y determinar su naturaleza

```
In [4]: df.head(5)
```

```
Out[4]:    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

VARIABLE DESCRIPTIONS:
survival        Survival
                (0 = No; 1 = Yes)
pclass          Passenger Class
                (1 = 1st; 2 = 2nd; 3 = 3rd)
name            Name
sex             Sex
age             Age
sibsp           Number of Siblings/Spouses Aboard
parch           Number of Parents/Children Aboard
ticket          Ticket Number
fare            Passenger Fare
cabin           Cabin
embarked        Port of Embarkation
                (C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:
Pclass is a proxy for socio-economic status (SES)
 1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in Years; Fractional if Age less than One (1)
 If the Age is Estimated, it is in the form xx.5

With respect to the family relation variables (i.e. sibsp and parch)
some relations were ignored.  The following are the definitions used
for sibsp and parch.

Sibling:  Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic
Spouse:   Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)
Parent:   Mother or Father of Passenger Aboard Titanic
Child:    Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins,
nephews/nieces, aunts/uncles, and in-laws.  Some children travelled
only with a nanny, therefore parch=0 for them.  As well, some
travelled with very close friends or neighbors in a village, however,
the definitions do not support such relations.
```

```
In [5]: Image(url='http://figures.boundless.com/18394/full/penelement-fieldelemformat-gif.gif')
```

```
Out[5]: <IPython.core.display.Image object>
```

1.0.2 £Qué tipos de variables tenemos en nuestro dataset?

- PassengerId:
- Survived:
- Pclass:
- Name:
- Sex:
- Age:
- SibSp:
- Parch:
- Ticket:
- Fare:
- Cabin:
- Embarked:

1.0.3 Existen 2 tipos de análisis exploratorio

- Univariable: Encontrar el comportamiento de una sola variable en el dataset
- Multivariable: Encontrar el comportamiento de dos o más variables en el dataset.

Éstos análisis, pueden ser de dos tipos, numéricos y gráficos.

2 Análisis Univariable

Variables categoricas: El mejor análisis numérico es obtener el conteo de incidencias de una variable

In [6]:
```
df["Sex"].value_counts()
```

Out[6]:
```
male      577
female    314
Name: Sex, dtype: int64
```

Variables numéricas: La medición de la centralidad, desviación, la distancia entre cuartiles, nos pueden dar información importante para hacer evaluación de la distribución de la variable usando la muestra observada.

In [7]:
```
df.describe()
```

Out[7]:
```
       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
```

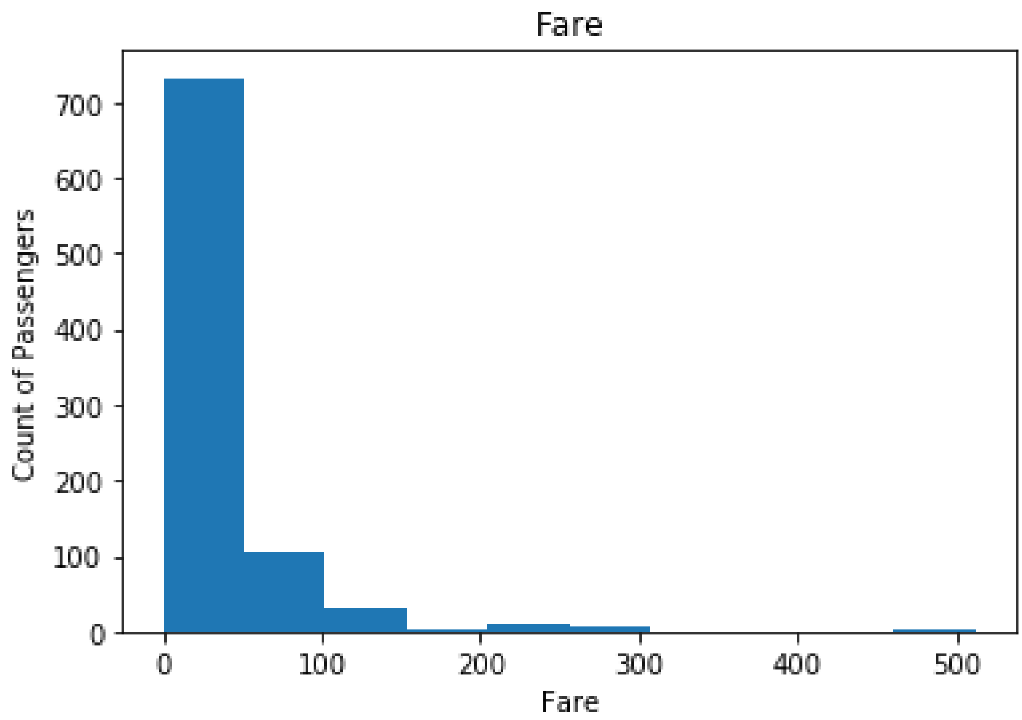
La curtosis y el sezgo son también medidas importantes para medir el comportamiento de una variable: * **Curtosis:** Que tan hacia la cola tiende la distribución * **Sezgo:** Que tanta falta de simetría tiene la distribución.

```
In [8]: Image(url="http://www.janzengroup.net/stats/images/skewkurt.JPG")
```

```
Out[8]: <IPython.core.display.Image object>
```

```
In [9]: print("skewness:", df["Fare"].skew())
        print("kurtosis:", df["Fare"].kurtosis())
        fig = plt.figure()
        ax = fig.add_subplot(111)
        x = np.linspace(df['Fare'].min(),df['Fare'].max(),1)
        ax.hist(df['Fare'], bins = 10, range = (df['Fare'].min(),df['Fare'].max()))
        plt.title('Fare')
        plt.xlabel('Fare')
        plt.ylabel('Count of Passengers')
        plt.show()
```

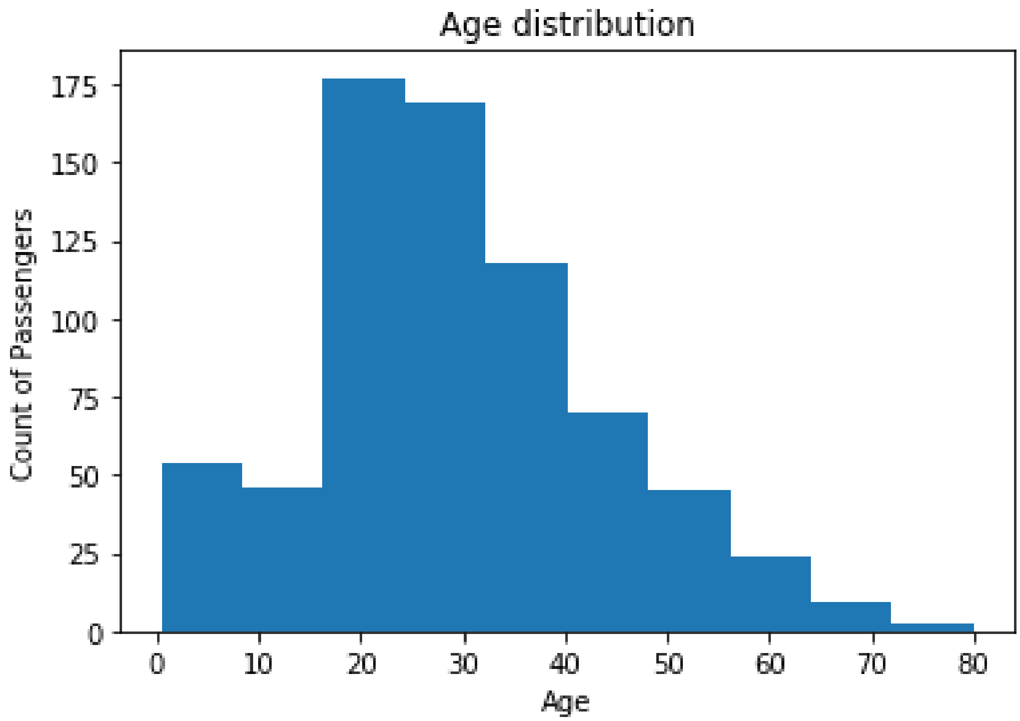
```
skewness: 4.78731651967
kurtosis: 33.3981408809
```




```
In [10]: print("skewness:", df["Age"].skew())
         print("kurtosis:", df["Age"].kurtosis())
         fig = plt.figure()
         ax = fig.add_subplot(111)
         x = np.linspace(df['Age'].min(),df['Age'].max(),1)
         ax.hist(df['Age'], bins = 10, range = (df['Age'].min(),df['Age'].max()))
         plt.title('Age distribution')
         plt.xlabel('Age')
         plt.ylabel('Count of Passengers')
         plt.show()
```

```
skewness: 0.389107782301
kurtosis: 0.178274153642
```

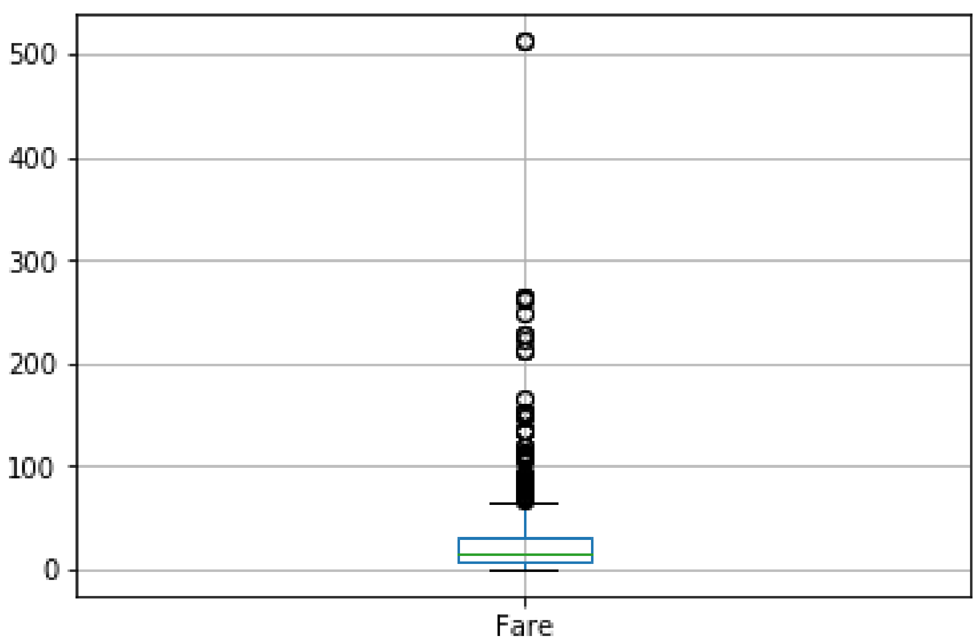
```
/Users/israel/anaconda3/lib/python3.6/site-packages/numpy/lib/function_base.py:583: RuntimeWarn
  keep = (tmp_a >= mn)
/Users/israel/anaconda3/lib/python3.6/site-packages/numpy/lib/function_base.py:584: RuntimeWarn
  keep &= (tmp_a <= mx)
```



En lo gráfico, también otras herramientas nos pueden servir para identificar patrones: Como los Boxplots

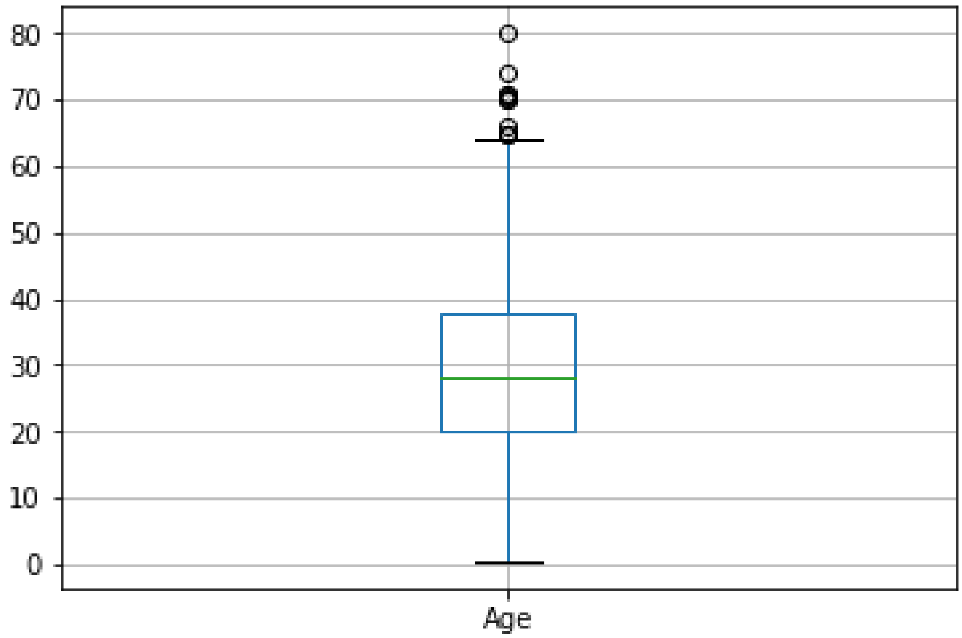
```
In [12]: df.boxplot(column='Fare')
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x11c3384e0>
```



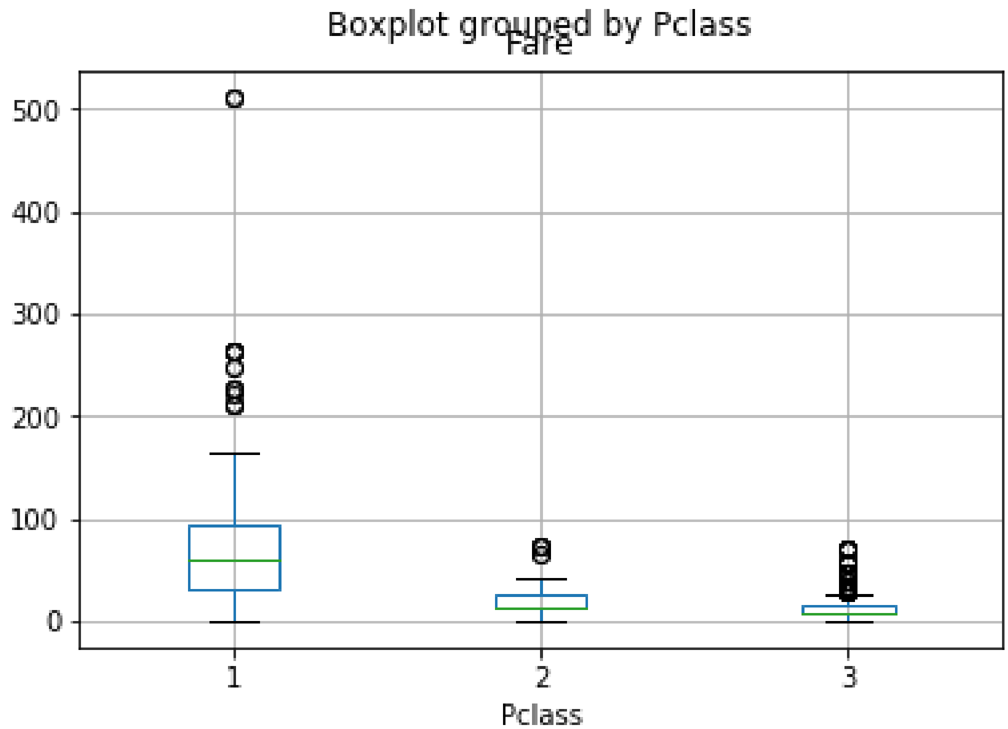
```
In [13]: df.boxplot(column='Age')
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x119cc0780>
```




```
In [14]: df.boxplot(column='Fare', by = 'Pclass')
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x11c6c5ba8>
```




```
In [17]: df.groupby('Pclass').Survived.sum()/df.groupby('Pclass').Survived.count()
```

```
Out[17]: Pclass
         1    0.629630
         2    0.472826
         3    0.242363
         Name: Survived, dtype: float64
```

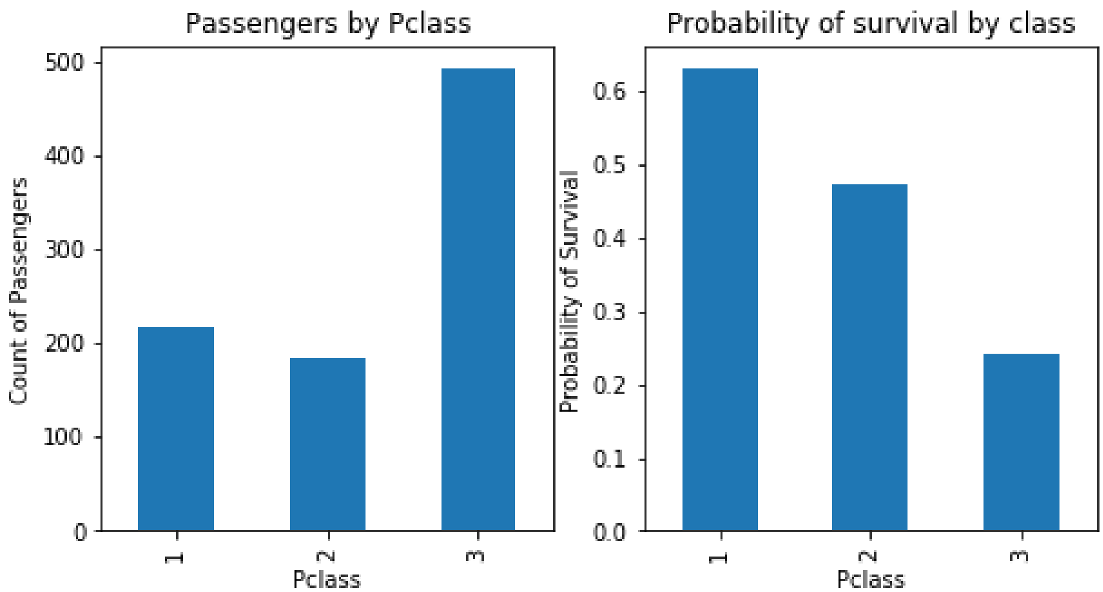
```
In [18]: #Agrupamos el conteo de sobrevivencia
         temp1 = df.groupby('Pclass').Survived.count()

         temp2 = df.groupby('Pclass').Survived.sum()/df.groupby('Pclass').Survived.count()
         fig = plt.figure(figsize=(8,4))
         #Agregamos el grafico usando matplotlib, desde pandas.
         ax1 = fig.add_subplot(121)
         ax1.set_xlabel('Pclass')
         ax1.set_ylabel('Count of Passengers')
         ax1.set_title("Passengers by Pclass")
         temp1.plot(kind='bar')


         ax2 = fig.add_subplot(122)
         temp2.plot(kind = 'bar')
```

```
ax2.set_xlabel('Pclass')
ax2.set_ylabel('Probability of Survival')
ax2.set_title("Probability of survival by class")
```

Out[18]: <matplotlib.text.Text at 0x11c9c04e0>



3 Análisis Multivariable

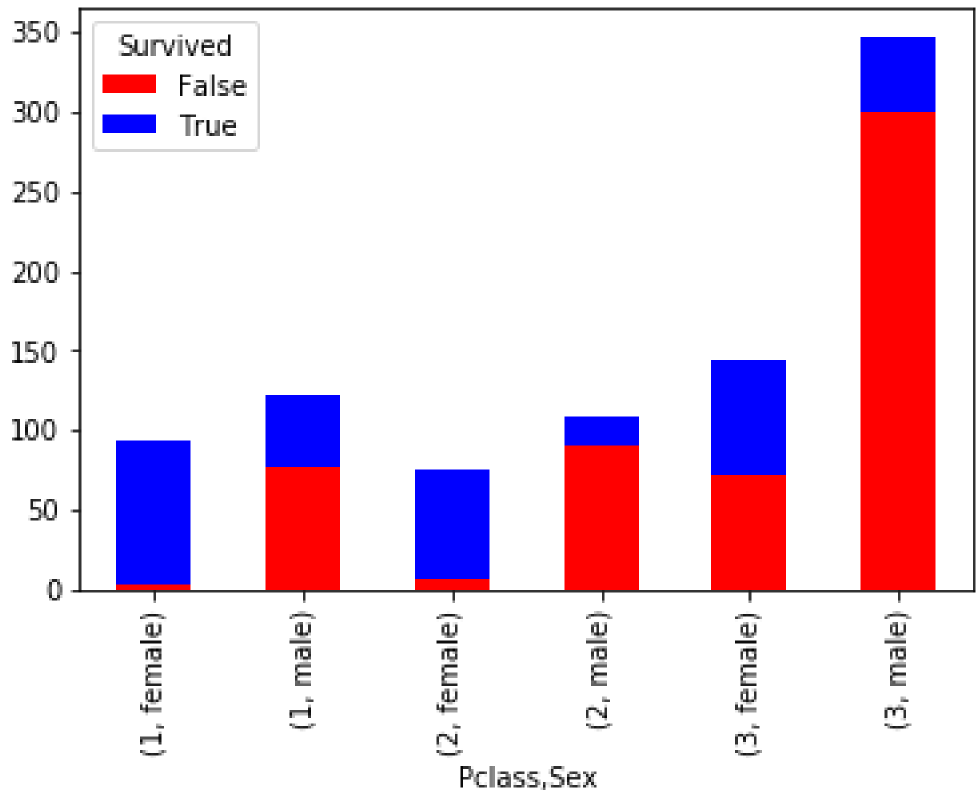
La Tabulación cruzada es la mejor herramienta no gráfica para explorar datos:

In [19]:
```
pd.crosstab([df.Pclass, df.Sex], df.Survived.astype(bool))
```

Out[19]:
```
Survived       False  True
Pclass Sex
1      female      3    91
       male       77    45
2      female      6    70
       male       91    17
3      female     72    72
       male      300    47
```

In [20]:
```
temp3 = pd.crosstab([df.Pclass, df.Sex], df.Survived.astype(bool))
temp3.plot(kind='bar', stacked=True, color=['red','blue'], grid=False)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x11c6edb00>



In [21]: df.corr()

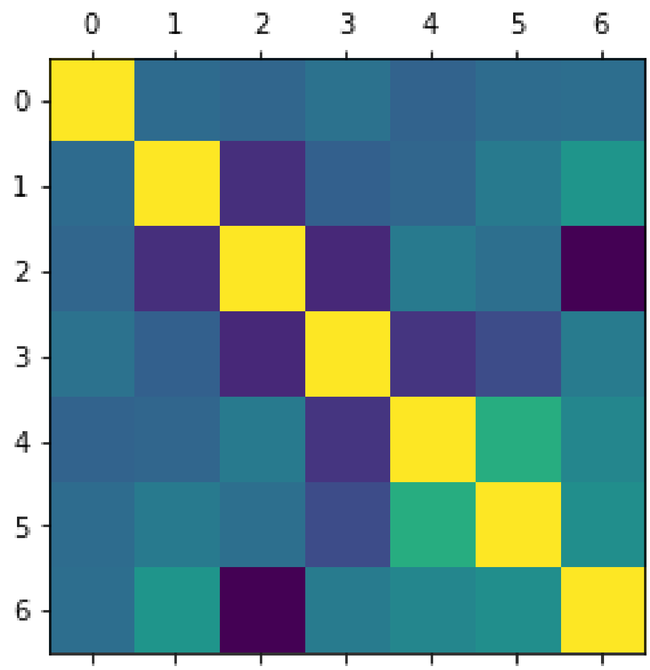
Out[21]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	\
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	

	Fare
PassengerId	0.012658
Survived	0.257307
Pclass	-0.549500
Age	0.096067
SibSp	0.159651
Parch	0.216225
Fare	1.000000

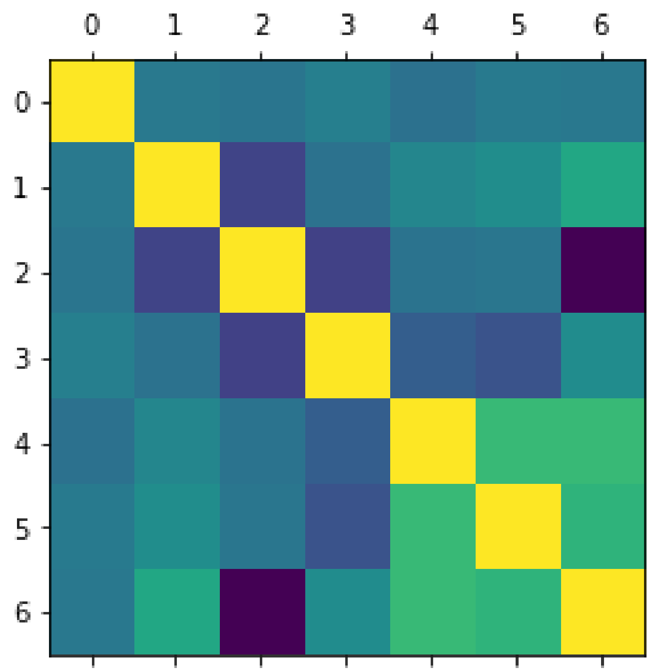
In [22]: plt.matshow(df.corr())

```
Out[22]: <matplotlib.image.AxesImage at 0x11cbd62b0>
```



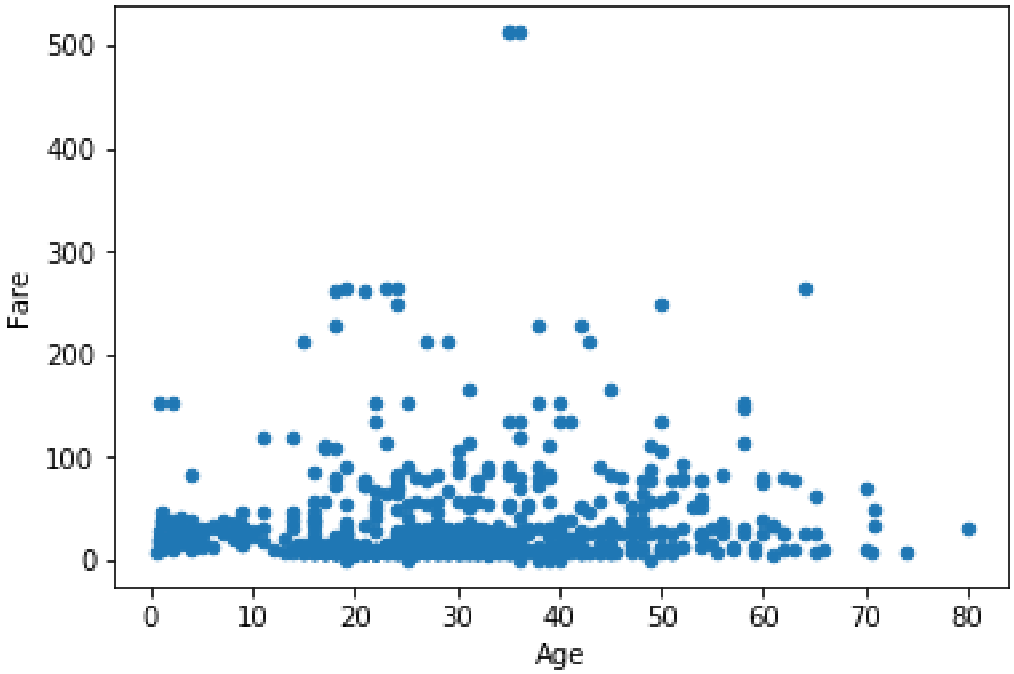

```
In [23]: plt.matshow(df.corr(method='spearman'))
```

```
Out[23]: <matplotlib.image.AxesImage at 0x11cf22400>
```



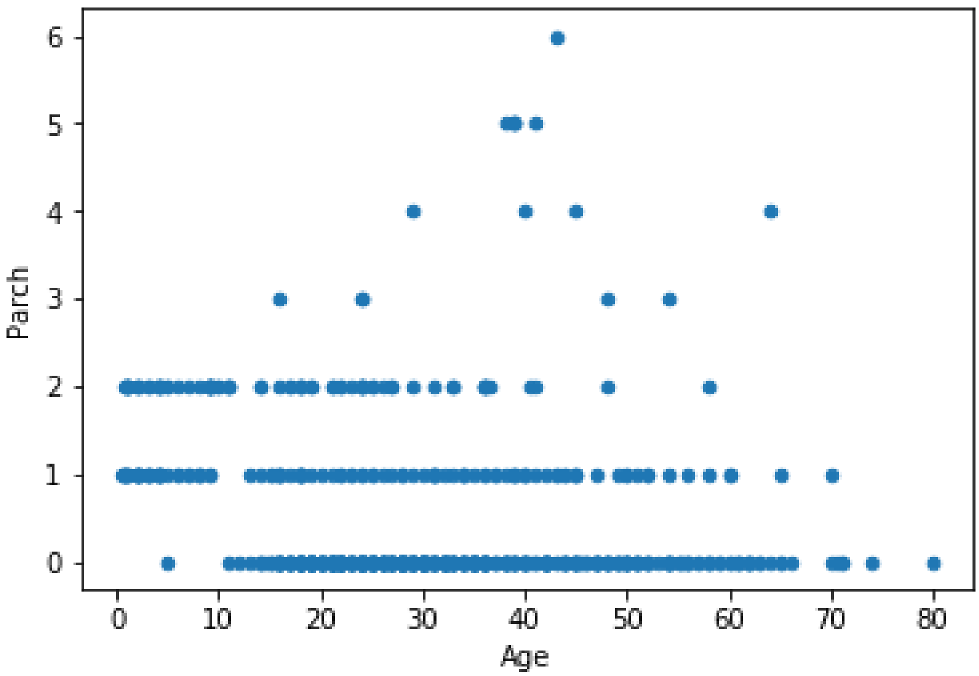
```
In [24]: df.plot(x="Age",y="Fare",kind="scatter")
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x11cbc6dd8>
```




```
In [25]: df.plot(x="Age",y="Parch",kind="scatter")
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x11d012208>
```



3.1 HETEROSKEDASTICITY

https://www.google.com.mx/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEv
espanol%2Fheteroskedasticity&usg=AFQjCNH3-uTaUolvQSKMEs2fYia_Kg3WGA