Titanic dataset

In [285]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [286]:

```
df=pd.read_csv('titanic_train.csv')
```

In [287]:

```
df.head()
```

Out[287]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [288]:

```
df.shape
```

Out[288]:

```
(891, 12)
```

In [289]:

```
df.columns
```

Out[289]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [290]:

```
df.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
```

In [291]:

```
df.describe()
```

Out[291]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [292]:

```
df.dtypes
```

Out[292]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Observations; 1. great variance in the mean and median of fare column 2.great variance in 75% percentile and max of columns: Age, Sipsp, Parch and fare this indicates outlinersdf.Survived.unique()

In [293]:

```
df.Survived.unique()
```

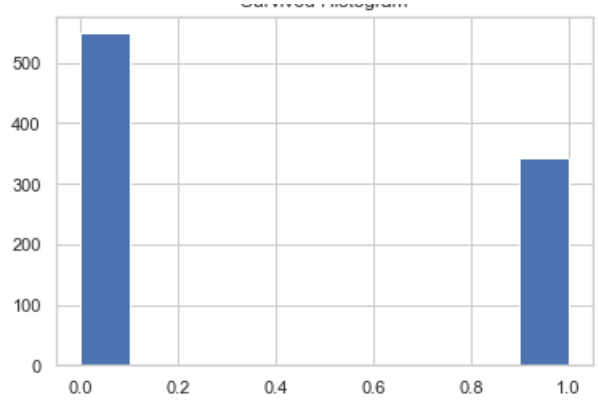
Out[293]:

```
array([0, 1], dtype=int64)
```

Data visualization

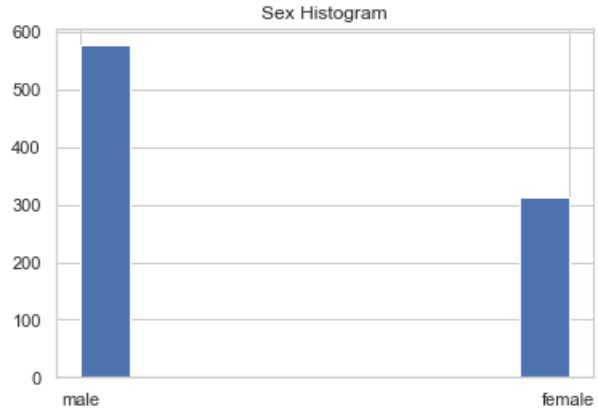
In [294]:

```
plt.hist(df['Survived'],bins=10)
plt.title('Survived Histogram')
plt.show()
```



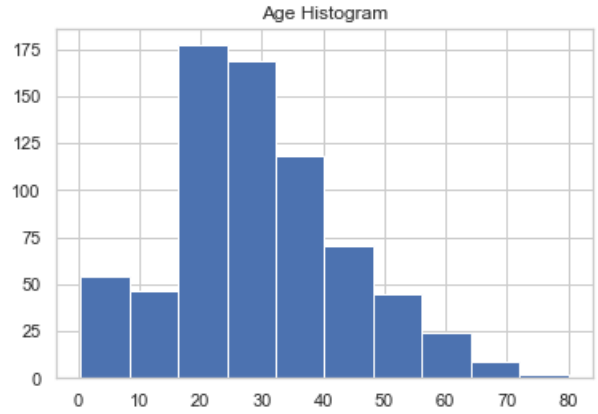
In [295]:

```
plt.hist(df['Sex'],bins=10)
plt.title('Sex Histogram')
plt.show()
```



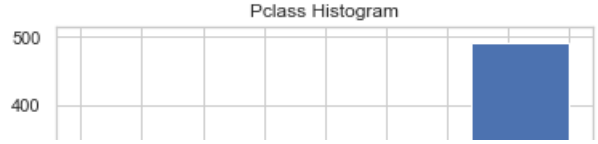
In [296]:

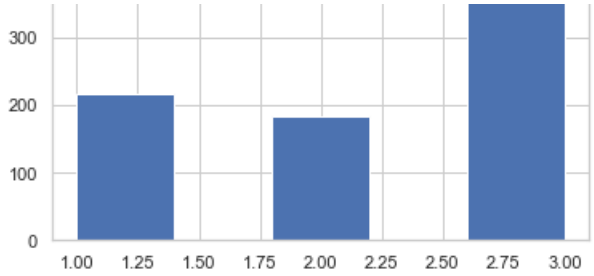
```
plt.hist(df['Age'],bins=10)
plt.title('Age Histogram')
plt.show()
```



In [297]:

```
plt.hist(df['Pclass'],bins=5)
plt.title('Pclass Histogram')
plt.show()
```



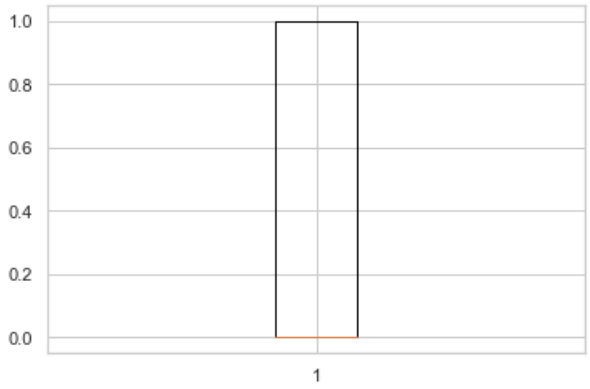


In [298]:

```
plt.boxplot(df['Survived'])
```

Out[298]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x28f8d7144c8>,
  <matplotlib.lines.Line2D at 0x28f8d70ef48>],
 'caps': [<matplotlib.lines.Line2D at 0x28f8d714f48>,
  <matplotlib.lines.Line2D at 0x28f8d714b88>],
 'boxes': [<matplotlib.lines.Line2D at 0x28f8d70ecc8>],
 'medians': [<matplotlib.lines.Line2D at 0x28f8d71aa48>],
 'fliers': [<matplotlib.lines.Line2D at 0x28f8d71af88>],
 'means': []}
```

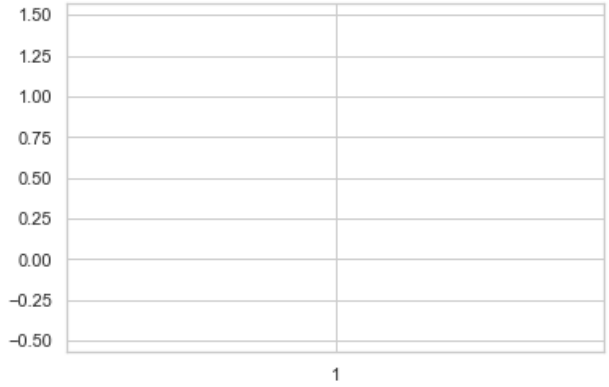


In [299]:

```
plt.boxplot(df['Age'])
```

Out[299]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x28f8d771508>,
  <matplotlib.lines.Line2D at 0x28f8d771bc8>],
 'caps': [<matplotlib.lines.Line2D at 0x28f8d771ec8>,
  <matplotlib.lines.Line2D at 0x28f8d771f48>],
 'boxes': [<matplotlib.lines.Line2D at 0x28f8d76ce88>],
 'medians': [<matplotlib.lines.Line2D at 0x28f8d777f88>],
 'fliers': [<matplotlib.lines.Line2D at 0x28f8d777c08>],
 'means': []}
```

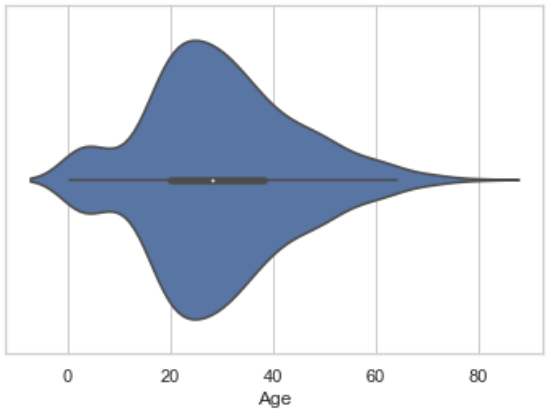


In [300]:

```
sns.violinplot(x='Age',data=df)
```

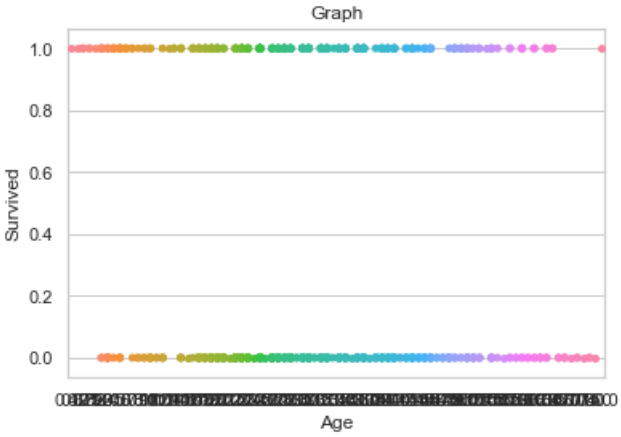
Out[300]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8d6d76c8>
```



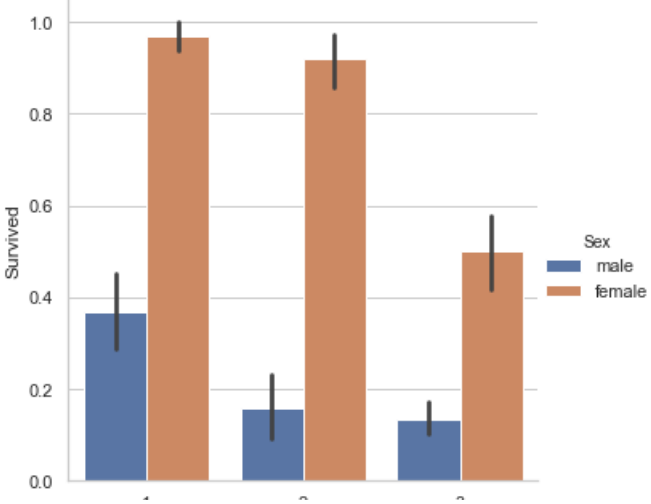
In [301]:

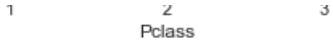
```
sns.set(style="whitegrid")
ax=sns.stripplot(x='Age',y='Survived',data=df);
plt.title('Graph')
plt.show()
```



In [302]:

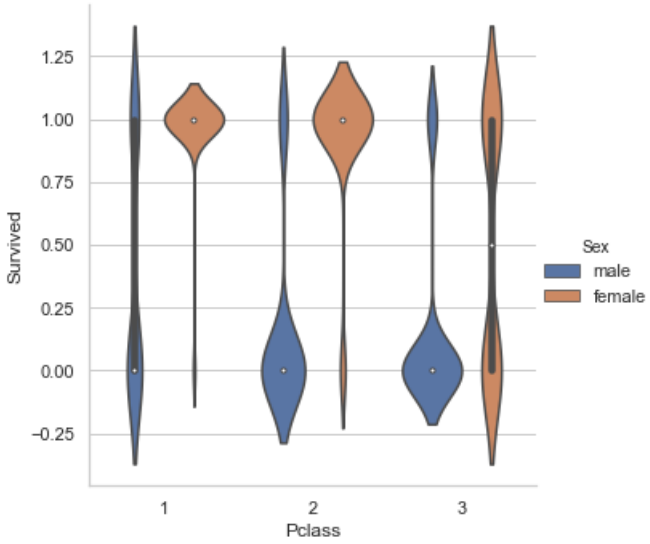
```
z=sns.catplot(x='Pclass',y='Survived',hue='Sex',data=df, kind='bar')
plt.show()
```





In [303]:

```
sns.catplot(x='Pclass',y='Survived',hue='Sex',data=df, kind='violin')
plt.show()
```



Checking correlation

In [304]:

```
dfcor=df.corr()
dfcor
```

Out[304]:

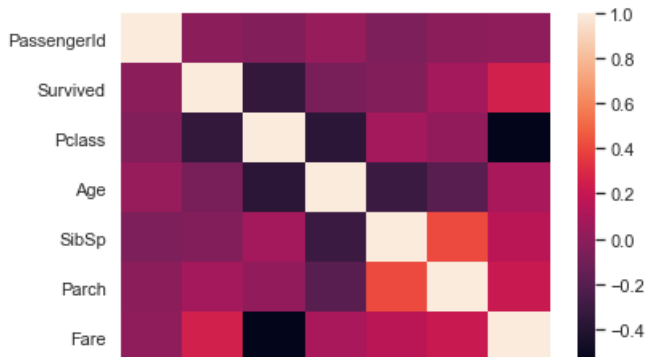
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221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

In [305]:

```
sns.heatmap(dfcor)
```

Out[305]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8db2b2c8>
```

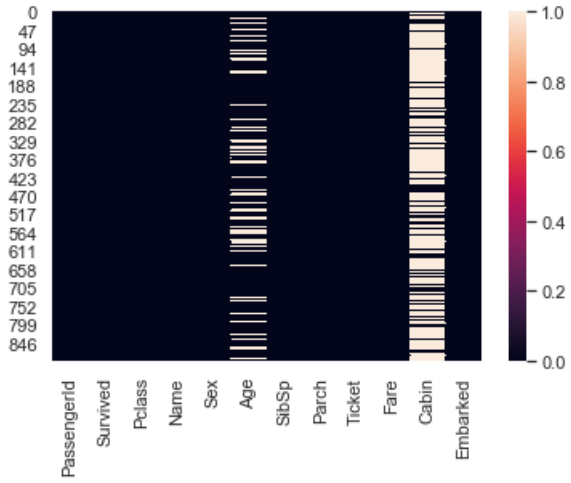


In [306]:

```
sns.heatmap(df.isnull())
```

Out[306]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8dbcb888>
```



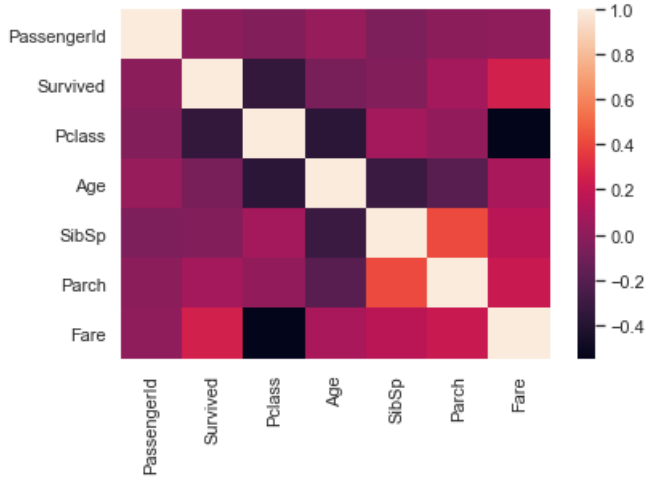
Observation; missing values are present in the dataset

In [307]:

```
sns.heatmap(dfcor)
```

Out[307]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8dc791c8>
```

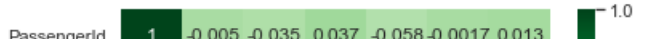


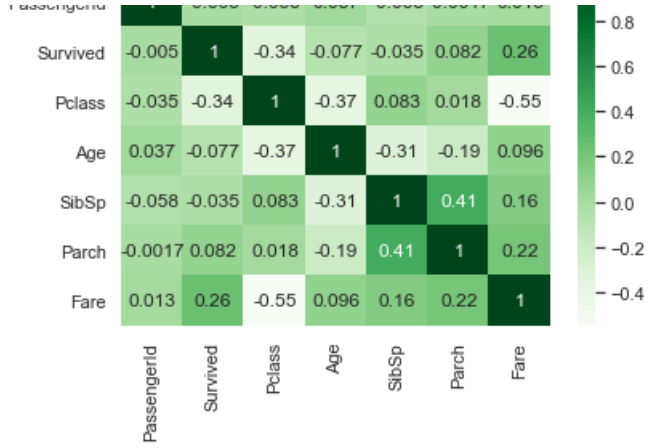
In [308]:

```
plt.figure(figsize=(6,4))
sns.heatmap(dfcor,cmap='Greens',annot=True)
```

Out[308]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f8ed2cc08>
```





In [309]:

```
df.head()
```

Out[309]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

ploting outliners

In [310]:

```
collist=df.columns.values
ncol=12
nrows=891
```

In [311]:

```
df.head(3)
```

Out[311]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [312]:

```
#drop Name,Ticket and Embarked columns
df.drop(['Name','Ticket','Embarked'],axis=1,inplace=True)
```

In [313]:

```
df.head(3)
```

Out[313]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin
0	1	0	3	male	22.0	1	0	7.2500	NaN
1	2	1	1	female	38.0	1	0	71.2833	C85
2	3	1	3	female	26.0	0	0	7.9250	NaN

In [314]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label=le.fit_transform(df['Sex'])
le.classes_
```

Out[314]:

```
array(['female', 'male'], dtype=object)
```

In [315]:

```
data=df.drop('Sex',axis='columns')
data
```

Out[315]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Cabin
0	1	0	3	22.0	1	0	7.2500	NaN
1	2	1	1	38.0	1	0	71.2833	C85
2	3	1	3	26.0	0	0	7.9250	NaN
3	4	1	1	35.0	1	0	53.1000	C123
4	5	0	3	35.0	0	0	8.0500	NaN
...	...	...	...	...	...	...	...	...
886	887	0	2	27.0	0	0	13.0000	NaN
887	888	1	1	19.0	0	0	30.0000	B42
888	889	0	3	NaN	1	2	23.4500	NaN
889	890	1	1	26.0	0	0	30.0000	C148
890	891	0	3	32.0	0	0	7.7500	NaN

891 rows × 8 columns

In [316]:

```
data['Sex']=label
data
```

Out[316]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Cabin	Sex
0	1	0	3	22.0	1	0	7.2500	NaN	1
1	2	1	1	38.0	1	0	71.2833	C85	0
2	3	1	3	26.0	0	0	7.9250	NaN	0
3	4	1	1	35.0	1	0	53.1000	C123	0
4	5	0	3	35.0	0	0	8.0500	NaN	1
...	...	...	...	...	...	...	...	...	...
886	887	0	2	27.0	0	0	13.0000	NaN	1
887	888	1	1	19.0	0	0	30.0000	B42	0
888	889	0	3	NaN	1	2	23.4500	NaN	0
889	890	1	1	26.0	0	0	30.0000	C148	1
890	891	0	3	32.0	0	0	7.7500	NaN	1

891 rows × 9 columns

In [317]:

```
# suppose we drop a new column from this data, let it be age
data.drop('Age',axis=1,inplace=True)
# data
```

In [318]:

```
#  now after deleting "age", if we run this cell, it would not show age
data.columns
```

Out[318]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'SibSp', 'Parch', 'Fare', 'Cabin',
       'Sex'],
      dtype='object')
```

In [319]:

```
# let me see those variables
collist
data.head()
```

Out[319]:

	PassengerId	Survived	Pclass	SibSp	Parch	Fare	Cabin	Sex
0	1	0	3	1	0	7.2500	NaN	1
1	2	1	1	1	0	71.2833	C85	0
2	3	1	3	0	0	7.9250	NaN	0
3	4	1	1	1	0	53.1000	C123	0
4	5	0	3	0	0	8.0500	NaN	1

In [320]:

```
collist = data.columns
collist
```

Out[320]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'SibSp', 'Parch', 'Fare', 'Cabin',
       'Sex'],
      dtype='object')
```

In [321]:

```
plt.figure(figsize=(16,16))
for i in range(0,len(collist)):
    plt.subplot(nrows,ncol,i+1)
    sns.distplot(data[collist[i]])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in kdensityfft(X, kernel, bw,
weights, gridsize, adjust, clip, cut, retgrid)
    450     try:
--> 451         bw = float(bw)
    452     except:

ValueError: could not convert string to float: 'scott'

During handling of the above exception, another exception occurred:

RuntimeError                              Traceback (most recent call last)
<ipython-input-321-8d81b8b80786> in <module>
```

```
      2 for i in range(0,len(collist)):
      3     plt.subplot(nrows,ncol,i+1)
----> 4     sns.distplot(data[collist[i]])

~\Anaconda3\lib\site-packages\seaborn\distributions.py in distplot(a, bins, hist, kde, rug, fit,
hist_kws, kde_kws, rug_kws, fit_kws, color, vertical, norm_hist, axlabel, label, ax)
    231     if kde:
    232         kde_color = kde_kws.pop("color", color)
--> 233         kdeplot(a, vertical=vertical, ax=ax, color=kde_color, **kde_kws)
    234         if kde_color != color:
    235             kde_kws["color"] = kde_color

~\Anaconda3\lib\site-packages\seaborn\distributions.py in kdeplot(data, data2, shade, vertical, ke
rnel, bw, gridsize, cut, clip, legend, cumulative, shade_lowest, cbar, cbar_ax, cbar_kws, ax, **kw
args)
    703         ax = _univariate_kdeplot(data, shade, vertical, kernel, bw,
    704                                  gridsize, cut, clip, legend, ax,
--> 705                                  cumulative=cumulative, **kwargs)
    706
    707     return ax

~\Anaconda3\lib\site-packages\seaborn\distributions.py in _univariate_kdeplot(data, shade,
vertical, kernel, bw, gridsize, cut, clip, legend, ax, cumulative, **kwargs)
    293         x, y = _statsmodels_univariate_kde(data, kernel, bw,
    294                                            gridsize, cut, clip,
--> 295                                            cumulative=cumulative)
    296     else:
    297         # Fall back to scipy if missing statsmodels

~\Anaconda3\lib\site-packages\seaborn\distributions.py in _statsmodels_univariate_kde(data,
kernel, bw, gridsize, cut, clip, cumulative)
    365     fft = kernel == "gau"
    366     kde = smnp.KDEUnivariate(data)
--> 367     kde.fit(kernel, bw, fft, gridsize=gridsize, cut=cut, clip=clip)
    368     if cumulative:
    369         grid, y = kde.support, kde.cdf

~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in fit(self, kernel, bw, fft,
weights, gridsize, adjust, cut, clip)
    138             density, grid, bw = kdensityfft(endog, kernel=kernel, bw=bw,
    139                     adjust=adjust, weights=weights, gridsize=gridsize,
--> 140                     clip=clip, cut=cut)
    141         else:
    142             density, grid, bw = kdensity(endog, kernel=kernel, bw=bw,

~\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py in kdensityfft(X, kernel, bw,
weights, gridsize, adjust, clip, cut, retgrid)
    451         bw = float(bw)
    452     except:
--> 453         bw = bandwidths.select_bandwidth(X, bw, kern) # will cross-val fit this pattern?
    454     bw *= adjust
    455

~\Anaconda3\lib\site-packages\statsmodels\nonparametric\bandwidths.py in select_bandwidth(x, bw, k
ernel)
    172         # eventually this can fall back on another selection criterion.
    173         err = "Selected KDE bandwidth is 0. Cannot estiamte density."
--> 174         raise RuntimeError(err)
    175     else:
    176         return bandwidth

RuntimeError: Selected KDE bandwidth is 0. Cannot estiamte density.
```

In [ ]:

```
```

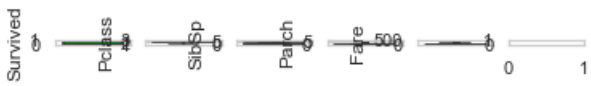
In [322]:

```
plt.figure(figsize=(ncol,5*ncol))
for i in range(1, len(collist)):
```

```
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(data[collist[i]],color='green',orient='v')
    plt.tight_layout()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-322-730e4a7141b8> in <module>
      2 for i in range(1, len(collist)):
      3     plt.subplot(nrows,ncol,i+1)
----> 4     sns.boxplot(data[collist[i]],color='green',orient='v')
      5     plt.tight_layout()

~\Anaconda3\lib\site-packages\seaborn\categorical.py in boxplot(x, y, hue, data, order, hue_order,
orient, color, palette, saturation, width, dodge, fliersize, linewidth, whis, ax, **kwargs)
   2245     kwargs.update(dict(whis=whis))
   2246
-> 2247     plotter.plot(ax, kwargs)
   2248     return ax
   2249

~\Anaconda3\lib\site-packages\seaborn\categorical.py in plot(self, ax, boxplot_kws)
    544     def plot(self, ax, boxplot_kws):
    545         """Make the plot."""
--> 546         self.draw_boxplot(ax, boxplot_kws)
    547         self.annotate_axes(ax)
    548         if self.orient == "h":

~\Anaconda3\lib\site-packages\seaborn\categorical.py in draw_boxplot(self, ax, kws)
    481                                             positions=[i],
    482                                             widths=self.width,
--> 483                                             **kws)
    484                 color = self.colors[i]
    485                 self.restyle_boxplot(artist_dict, color, props)

~\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*args, **kwargs)
    305                 f"for the old name will be dropped %(removal)s.")
    306             kwargs[new] = kwargs.pop(old)
--> 307         return func(*args, **kwargs)
    308
    309     # wrapper() must keep the same documented signature as func(): if we

~\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args, **kwargs)
   1597     def inner(ax, *args, data=None, **kwargs):
   1598         if data is None:
-> 1599             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1600
   1601         bound = new_sig.bind(ax, *args, **kwargs)

~\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in boxplot(self, x, notch, sym, vert, whis,
positions, widths, patch_artist, bootstrap, usermedians, conf_intervals, meanline, showmeans,
showcaps, showbox, showfliers, boxprops, labels, flierprops, medianprops, meanprops, capprops, whi
skerprops, manage_ticks, autorange, zorder)
   3667
   3668         bxpstats = cbook.boxplot_stats(x, whis=whis, bootstrap=bootstrap,
-> 3669                                        labels=labels, autorange=autorange)
   3670         if notch is None:
   3671             notch = rcParams['boxplot.notch']

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in boxplot_stats(X, whis, bootstrap,
labels, autorange)
   1283
   1284         # arithmetic mean
-> 1285         stats['mean'] = np.mean(x)
   1286
   1287         # medians and quartiles

<__array_function__ internals> in mean(*args, **kwargs)

~\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py in mean(a, axis, dtype, out, keepdims)
   3333
   3334     return _methods._mean(a, axis=axis, dtype=dtype,
-> 3335                           out=out, **kwargs)
   3336
   3337

~\Anaconda3\lib\site-packages\numpy\core\_methods.py in _mean(a, axis, dtype, out, keepdims)
```

```
    161                 ret = ret.dtype.type(ret / rcount)
    162         else:
--> 163             ret = ret / rcount
    164
    165         return ret

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

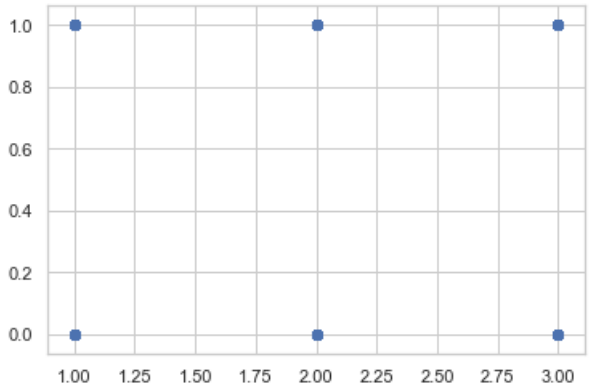


In [323]:

```
plt.scatter(df['Pclass'],df['Survived'])
```

Out[323]:

```
<matplotlib.collections.PathCollection at 0x28f8a4b4a08>
```

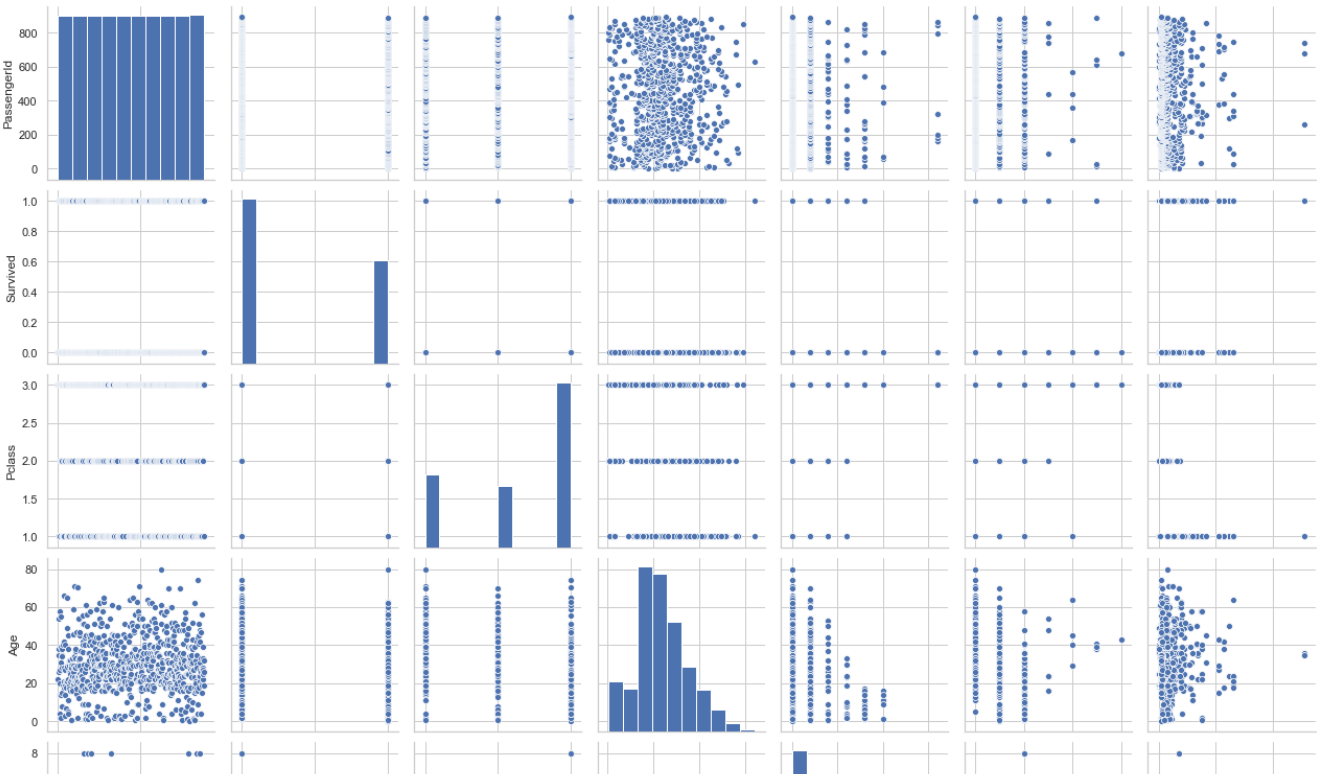


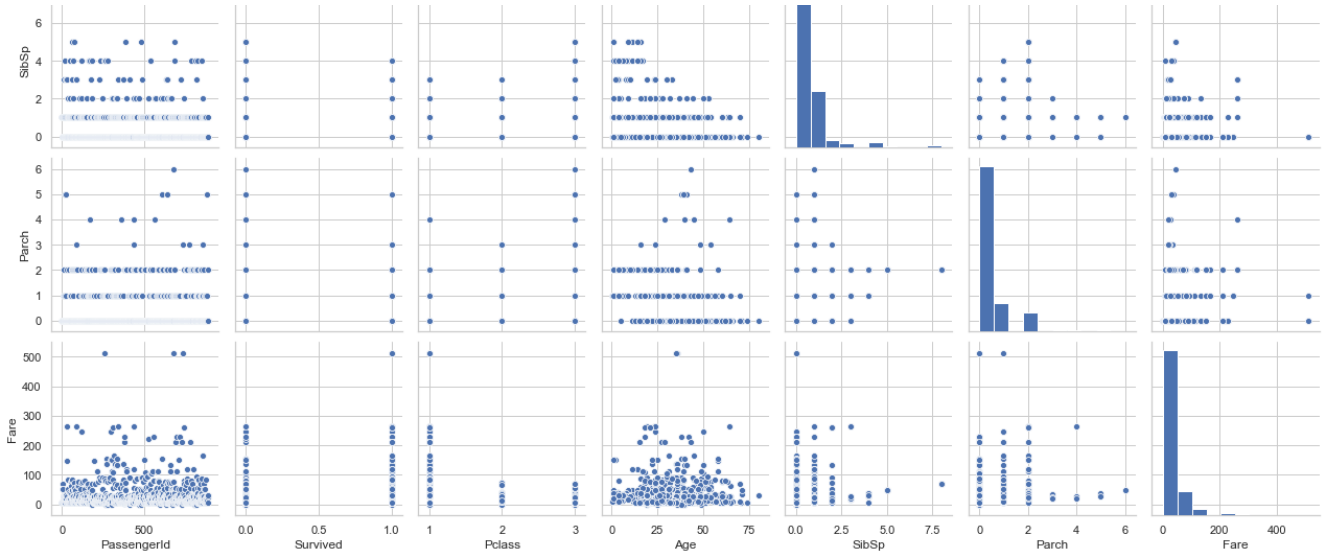
In [324]:

```
sns.pairplot(df)
```

Out[324]:

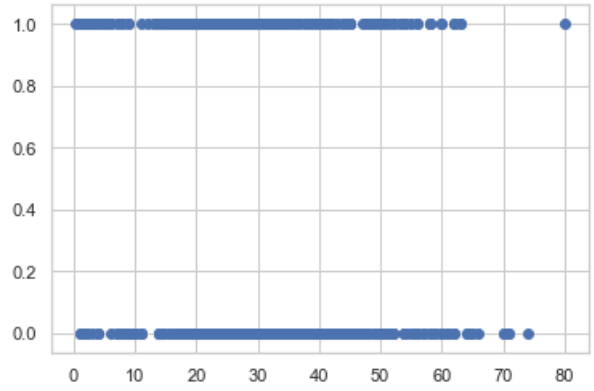
```
<seaborn.axisgrid.PairGrid at 0x28f8a740188>
```





In [325]:

```
plt.scatter(df['Age'],df['Survived'])
plt.show()
```



In [326]:

```
data.drop('Cabin',axis=1,inplace=True)
```

In [327]:

```
data.head()
```

Out[327]:

	PassengerId	Survived	Pclass	SibSp	Parch	Fare	Sex
0	1	0	3	1	0	7.2500	1
1	2	1	1	1	0	71.2833	0
2	3	1	3	0	0	7.9250	0
3	4	1	1	1	0	53.1000	0
4	5	0	3	0	0	8.0500	1

In [328]:

```
data.shape
```

Out[328]:

```
(891, 7)
```

Removing outliners

In [329]:

```
data.dtypes
```

Out[329]:

```
PassengerId      int64
Survived         int64
Pclass           int64
SibSp            int64
Parch            int64
Fare           float64
Sex              int32
dtype: object
```

In [330]:

```
from scipy.stats import zscore
z=np.abs(zscore(data))
z
```

Out[330]:

```
array([[1.73010796, 0.78927234, 0.82737724, ..., 0.47367361, 0.50244517,
        0.73769513],
       [1.72622007, 1.2669898 , 1.56610693, ..., 0.47367361, 0.78684529,
        1.35557354],
       [1.72233219, 1.2669898 , 0.82737724, ..., 0.47367361, 0.48885426,
        1.35557354],
       ...,
       [1.72233219, 0.78927234, 0.82737724, ..., 2.00893337, 0.17626324,
        1.35557354],
       [1.72622007, 1.2669898 , 1.56610693, ..., 0.47367361, 0.04438104,
        0.73769513],
       [1.73010796, 0.78927234, 0.82737724, ..., 0.47367361, 0.49237783,
        0.73769513]])
```

In [331]:

```
data=[(z<3).all(axis=1)] #removing outliners
```

In [332]:

```
data=pd.DataFrame(data=df)
print(data)
```

```
     PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare Cabin
0              1         0       3    male  22.0      1      0   7.2500   NaN
1              2         1       1  female  38.0      1      0  71.2833   C85
2              3         1       3  female  26.0      0      0   7.9250   NaN
3              4         1       1  female  35.0      1      0  53.1000  C123
4              5         0       3    male  35.0      0      0   8.0500   NaN
..           ...       ...     ...     ...   ...    ...    ...      ...   ...
886          887         0       2    male  27.0      0      0  13.0000   NaN
887          888         1       1  female  19.0      0      0  30.0000   B42
888          889         0       3  female   NaN      1      2  23.4500   NaN
889          890         1       1    male  26.0      0      0  30.0000  C148
890          891         0       3    male  32.0      0      0   7.7500   NaN

[891 rows x 9 columns]
```

In [333]:

```
data.shape
```

Out[333]:

```
(891, 9)
```

In [334]:

```
data.dropna(how='all')
```

Out[334]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin
0	1	0	3	male	22.0	1	0	7.2500	NaN
1	2	1	1	female	38.0	1	0	71.2833	C85
2	3	1	3	female	26.0	0	0	7.9250	NaN
3	4	1	1	female	35.0	1	0	53.1000	C123
4	5	0	3	male	35.0	0	0	8.0500	NaN
...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.0	0	0	13.0000	NaN
887	888	1	1	female	19.0	0	0	30.0000	B42
888	889	0	3	female	NaN	1	2	23.4500	NaN
889	890	1	1	male	26.0	0	0	30.0000	C148
890	891	0	3	male	32.0	0	0	7.7500	NaN

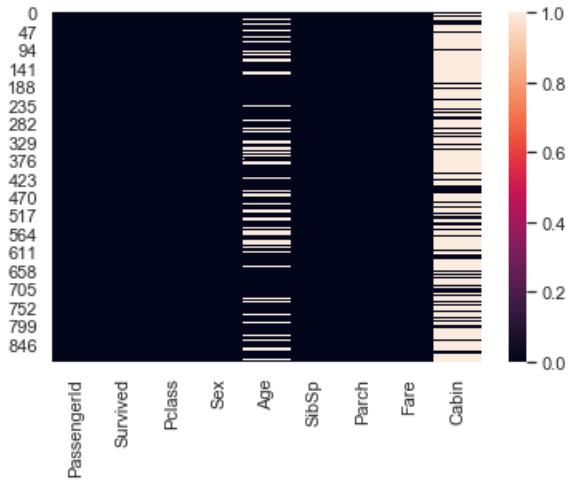
891 rows × 9 columns

In [335]:

```
sns.heatmap(data.isnull())
```

Out[335]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f9084ad08>
```



In [336]:

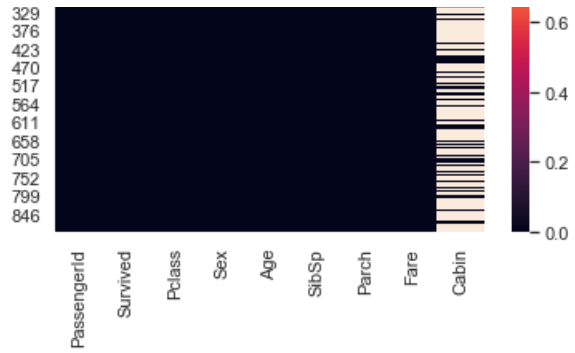
```
data['Age'].fillna(28,inplace=True)
```

In [337]:

```
sns.heatmap(data.isnull())
```

Out[337]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28f908a30c8>
```



In [338]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label=le.fit_transform(data['Sex'])
le.classes_
```

Out[338]:

```
array(['female', 'male'], dtype=object)
```

In [339]:

```
data=df.drop('Sex',axis='columns')
data
```

Out[339]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Cabin
0	1	0	3	22.0	1	0	7.2500	NaN
1	2	1	1	38.0	1	0	71.2833	C85
2	3	1	3	26.0	0	0	7.9250	NaN
3	4	1	1	35.0	1	0	53.1000	C123
4	5	0	3	35.0	0	0	8.0500	NaN
...	...	...	...	...	...	...	...	...
886	887	0	2	27.0	0	0	13.0000	NaN
887	888	1	1	19.0	0	0	30.0000	B42
888	889	0	3	28.0	1	2	23.4500	NaN
889	890	1	1	26.0	0	0	30.0000	C148
890	891	0	3	32.0	0	0	7.7500	NaN

891 rows × 8 columns

In [373]:

```
data['Sex']=label
data
```

Out[373]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	1	22.0	1	0	7.2500
1	2	1	1	0	38.0	1	0	71.2833
2	3	1	3	0	26.0	0	0	7.9250
3	4	1	1	0	35.0	1	0	53.1000
4	5	0	3	1	35.0	0	0	8.0500
...	...	...	...	...	...	...	...	...
886	887	0	2	1	27.0	0	0	13.0000

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
887	888	1	1	0	19.0	0	0	30.0000
888	889	0	3	0	28.0	1	2	23.4500
889	890	1	1	1	26.0	0	0	30.0000
890	891	0	3	1	32.0	0	0	7.7500

891 rows × 8 columns

In [374]:

```
data=df.drop('Cabin',axis='columns')
```

In [376]:

```
data.head()
```

Out[376]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	male	22.0	1	0	7.2500
1	2	1	1	female	38.0	1	0	71.2833
2	3	1	3	female	26.0	0	0	7.9250
3	4	1	1	female	35.0	1	0	53.1000
4	5	0	3	male	35.0	0	0	8.0500

In [377]:

```
data['Sex']=label
data
```

Out[377]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	1	22.0	1	0	7.2500
1	2	1	1	0	38.0	1	0	71.2833
2	3	1	3	0	26.0	0	0	7.9250
3	4	1	1	0	35.0	1	0	53.1000
4	5	0	3	1	35.0	0	0	8.0500
...	...	...	...	...	...	...	...	...
886	887	0	2	1	27.0	0	0	13.0000
887	888	1	1	0	19.0	0	0	30.0000
888	889	0	3	0	28.0	1	2	23.4500
889	890	1	1	1	26.0	0	0	30.0000
890	891	0	3	1	32.0	0	0	7.7500

891 rows × 8 columns

In [378]:

```
x=data.drop('Survived',axis=1)
y=data['Survived']
```

In [379]:

```
x.shape
```

Out[379]:

```
(891, 7)
```

In [380]:

```
y.shape
```

Out[380]:

(891,)

In [381]:

```
y=y.values.reshape(-1,1)
```

In [382]:

```
y.shape
```

Out[382]:

(891, 1)

In [383]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [384]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=43)
```

In [385]:

```
x_train.shape
```

Out[385]:

(694, 7)

In [386]:

```
x_test.shape
```

Out[386]:

(197, 7)

In [387]:

```
y_train.shape
```

Out[387]:

(694, 1)

In [388]:

```
lg=LogisticRegression()
```

In [389]:

```
lg.fit(x_train,y_train)
```

Out[389]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [390]:

```
pred=lg.predict(x_test)
print(pred)
```

```
[0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0
 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 1
 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0
 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1]
```

In [391]:

```
print('accuracy_score:',accuracy_score(y_test,pred))
```

```
accuracy_score: 0.7208121827411168
```

In [392]:

```
print(confusion_matrix(y_test,pred))
```

```
[[101  18]
 [ 37  41]]
```

In [393]:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.85      0.79       119
           1       0.69      0.53      0.60        78

    accuracy                           0.72       197
   macro avg       0.71      0.69      0.69       197
weighted avg       0.72      0.72      0.71       197
```

In [396]:

```
data.head(3)
```

Out[396]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	1	22.0	1	0	7.2500
1	2	1	1	0	38.0	1	0	71.2833
2	3	1	3	0	26.0	0	0	7.9250

In [397]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [398]:

```
#all models at once
model=[DecisionTreeClassifier(),SVC(),KNeighborsClassifier(),MultinomialNB()]
for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    predm=m.predict(x_test)
    print('Accuracy score of',m, 'is:')
    print(accuracy_score(y_test,predm))
    print(confusion_matrix(y_test,predm))
    print(classification_report(y_test,predm))
    print('\n')
```

```
Accuracy score of DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best') is:
0.700507614213198
[[93 26]
 [33 45]]
              precision    recall  f1-score   support

           0       0.74      0.78      0.76       119
           1       0.63      0.58      0.60        78

    accuracy                           0.70       197
   macro avg       0.69      0.68      0.68       197
weighted avg       0.70      0.70      0.70       197



Accuracy score of SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False) is:
0.6395939086294417
[[117   2]
 [ 69   9]]
              precision    recall  f1-score   support

           0       0.63      0.98      0.77       119
           1       0.82      0.12      0.20        78

    accuracy                           0.64       197
   macro avg       0.72      0.55      0.48       197
weighted avg       0.70      0.64      0.54       197



Accuracy score of KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform') is:
0.6192893401015228
[[96 23]
 [52 26]]
              precision    recall  f1-score   support

           0       0.65      0.81      0.72       119
           1       0.53      0.33      0.41        78

    accuracy                           0.62       197
   macro avg       0.59      0.57      0.56       197
weighted avg       0.60      0.62      0.60       197



Accuracy score of MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True) is:
0.649746192893401
[[99 20]
 [49 29]]
              precision    recall  f1-score   support
```

```
              0       0.67      0.83      0.74       119
              1       0.59      0.37      0.46        78

    accuracy                           0.65       197
   macro avg       0.63      0.60      0.60       197
weighted avg       0.64      0.65      0.63       197
```

In [399]:

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
```

In [400]:

```
mnb=MultinomialNB()

score=cross_val_score(mnb,x,y,cv=5)
print(score)
print(score.mean())
print(score.std())
```

```
[0.56424581 0.63483146 0.66292135 0.73595506 0.67977528]
0.6555457912246563
0.05634161331460007
```

In [401]:

```
sv=SVC()
score=cross_val_score(sv,x,y,cv=8)
print(score)
print(score.mean())
print(score.std())
```

```
[0.61607143 0.625      0.66964286 0.66666667 0.64864865 0.63063063
 0.71171171 0.63963964]
0.6510014478764479
0.028992273626627672
```

In [402]:

```
kNN=KNeighborsClassifier()
score=cross_val_score(kNN,x,y,cv=8)
print(score)
print(score.mean())
print(score.std())
```

```
[0.61607143 0.65178571 0.40178571 0.3963964  0.47747748 0.44144144
 0.54954955 0.66666667]
0.5251467985842986
0.10368319825140256
```

In [403]:

```
dtc=DecisionTreeClassifier()
score=cross_val_score(dtc,x,y,cv=8)
print(score)
print(score.mean())
print(score.std())
```

```
[0.75892857 0.69642857 0.54464286 0.64864865 0.63063063 0.53153153
 0.8018018  0.65765766]
0.6587837837837838
0.08804696048238081
```

In [405]:

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
parameters={'kernel':('linear','rbf'),'C':[1,10]}
svc=svm.SVC()
clf=GridSearchCV(svc,parameters)
clf.fit(x,y)

sorted(clf.cv_results_.keys())
```

Out[405]:

```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_C',
 'param_kernel',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

In [406]:

```
print(clf.best_params_)
```

```
{'C': 10, 'kernel': 'linear'}
```

In [407]:

```
sv=svm.SVC(kernel='linear',C=1)
sv.fit(x,y)
```

Out[407]:

```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [408]:

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=100,random_state=42)
#RandomForestClassifier(100)......Default
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.8121827411167513
[[110   9]
 [ 28  50]]
              precision    recall  f1-score   support

           0       0.80      0.92      0.86       119
           1       0.85      0.64      0.73        78

    accuracy                           0.81       197
   macro avg       0.82      0.78      0.79       197
weighted avg       0.82      0.81      0.81       197
```

In [411]:

```
from sklearn.ensemble import AdaBoostClassifier
#AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),n_estimators=50,learning_rate=1.0)

ad=AdaBoostClassifier()
ad.fit(x_train,y_train)

ad_pred=ad.predict(x_test)
print(accuracy_score(y_test,ad_pred))
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

```
0.7360406091370558
[[98 21]
 [31 47]]
              precision    recall  f1-score   support

           0       0.76      0.82      0.79       119
           1       0.69      0.60      0.64        78

    accuracy                           0.74       197
   macro avg       0.73      0.71      0.72       197
weighted avg       0.73      0.74      0.73       197
```

In [412]:

```
from sklearn.svm import SVC
svc=SVC()

ad =AdaBoostClassifier(n_estimators=50,base_estimator=svc,algorithm='SAMME')
ad.fit(x_train,y_train)

ad_pred=ad.predict(x_test)
print(accuracy_score(y_test,ad_pred))
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

```
0.6040609137055838
[[119   0]
 [ 78   0]]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75       119
           1       0.00      0.00      0.00        78

    accuracy                           0.60       197
   macro avg       0.30      0.50      0.38       197
weighted avg       0.36      0.60      0.45       197
```

In [413]:

```
from sklearn.svm import SVC
svc=SVC(probability=True,kernel='linear')

#create adaboost classifier object
ad =AdaBoostClassifier(n_estimators=50,base_estimator=svc)
ad.fit(x_train,y_train)

ad_pred=ad.predict(x_test)
print(accuracy_score(y_test,ad_pred))
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

```
0.6243654822335025
[[117   2]
 [ 72   6]]
              precision    recall  f1-score   support

           0       0.62      0.98      0.76       119
```

```
              1       0.75      0.08      0.14        78

    accuracy                           0.62       197
   macro avg       0.68      0.53      0.45       197
weighted avg       0.67      0.62      0.51       197
```

In [414]:

```
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
gb.fit(x_train,y_train)

gb_pred=gb.predict(x_test)
print(accuracy_score(y_test,gb_pred))
print(confusion_matrix(y_test,gb_pred))
print(classification_report(y_test,gb_pred))
```

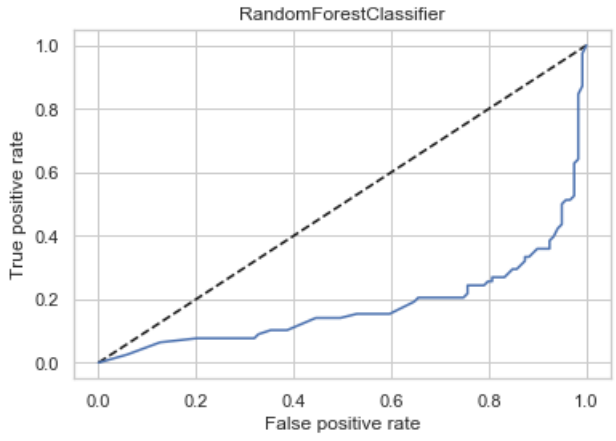
```
0.766497461928934
[[108  11]
 [ 35  43]]
              precision    recall  f1-score   support

           0       0.76      0.91      0.82       119
           1       0.80      0.55      0.65        78

    accuracy                           0.77       197
   macro avg       0.78      0.73      0.74       197
weighted avg       0.77      0.77      0.76       197
```

In [415]:

```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

from sklearn.metrics import roc_auc_score
```

In [418]:

```
y_pred_prob=rf.predict_proba(x_test)[:,0]
fpr,tpr,threshold=roc_curve(y_test,y_pred_prob)
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='RandomForestClassifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RandomForestClassifier')
plt.show()
auc_score=roc_auc_score(y_test,rf.predict(x_test))
auc_score
```



Out[418]:

```
0.7826976944624003
```

In [420]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
p=rf.predict(x_test)
print(accuracy_score(y_test,p))
```

```
0.7918781725888325
```

In [422]:

```
auc_score=roc_auc_score(y_test,rf.predict(x_test))
```

In [423]:

```
print(auc_score)
```

```
0.7570566688213748
```

In [424]:

```
#save the best model
from sklearn.externals import joblib
filename='svrftitanicfile.obj'
```

In [425]:

```
joblib.dump(sv,'svrftitanicfile.obj')
```

Out[425]:

```
['svrftitanicfile.obj']
```

In [ ]:

```

```