In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
pd.set_option("display.max_columns",None)
pd.set_option("display.max_rows",None)
```

Generating dataframe of train dataset

In [3]:
```
df_train = pd.read_csv("D:\\titanic/train.csv")
df_train.head(2)
```

Out[3]:

	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

Generating dataframe of test dataset

In [4]:
```
df_test = pd.read_csv("D:\\titanic/test.csv")
df_test.head(2)
```

Out[4]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S

In [5]:
```
df = pd.concat([df_train,df_test],axis=0)
df
```

				Adolfina								
15	16	1.0	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.00	0	0	248706	16.0000	NaN	S
16	17	0.0	3	Rice, Master. Eugene	male	2.00	4	1	382652	29.1250	NaN	Q
17	18	1.0	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0.0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.00	1	0	345763	18.0000	NaN	S
19	20	1.0	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0.0	2	Fynney, Mr. Joseph J	male	35.00	0	0	239865	26.0000	NaN	S
21	22	1.0	2	Beesley, Mr. Lawrence	male	34.00	0	0	248698	13.0000	D56	S
22	23	1.0	3	McGowan, Miss. Anna "Annie"	female	15.00	0	0	330923	8.0292	NaN	Q
23	24	1.0	1	Sloper, Mr. William Thompson	male	28.00	0	0	113788	35.5000	A6	S
24	25	0.0	3	Palsson, Miss. Torborg Danira	female	8.00	3	1	349909	21.0750	NaN	S

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object 
 11  Embarked     1307 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 132.9+ KB
```

dropping those column which is not significant for the model preparation

In [7]:
```
df_train.drop(['PassengerId','Cabin'],axis = 1, inplace=True)
df_train.head(10)
```

Out[7]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
5	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Q
6	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S
7	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	S
8	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	S
9	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	C

spliting name by respictive Title

In [8]:
```
Name = df_train['Name']
df_train['Title'] = [ i.split('.')[0].split(',')[-1] for i in Name]
df_train.head(10)
```

Out[8]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S	Mr
5	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Q	Mr
6	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S	Mr
7	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	S	Master
8	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	S	Mrs
9	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	C	Mrs

In [9]:
```
df_train.drop("Name",axis=1,inplace=True)
df_train.head(10)
```

Out[9]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	0	0	373450	8.0500	S	Mr
5	0	3	male	NaN	0	0	330877	8.4583	Q	Mr
6	0	1	male	54.0	0	0	17463	51.8625	S	Mr
7	0	3	male	2.0	3	1	349909	21.0750	S	Master
8	1	3	female	27.0	0	2	347742	11.1333	S	Mrs
9	1	2	female	14.0	1	0	237736	30.0708	C	Mrs

define a class for data cleaning

In [10]:
```
def data_cleaning(data):
    d = df_train[data]
    for column in d.columns:
        if df_train[column].isnull().any():
            df_train[column].fillna(df_train[column].median(),inplace = True)
```

In [11]:
```
feature = ['Age','SibSp','Parch','Fare']
data_cleaning(feature)
```

In [12]:
```
df_train['Embarked'].fillna(method='ffill',inplace=True)
```

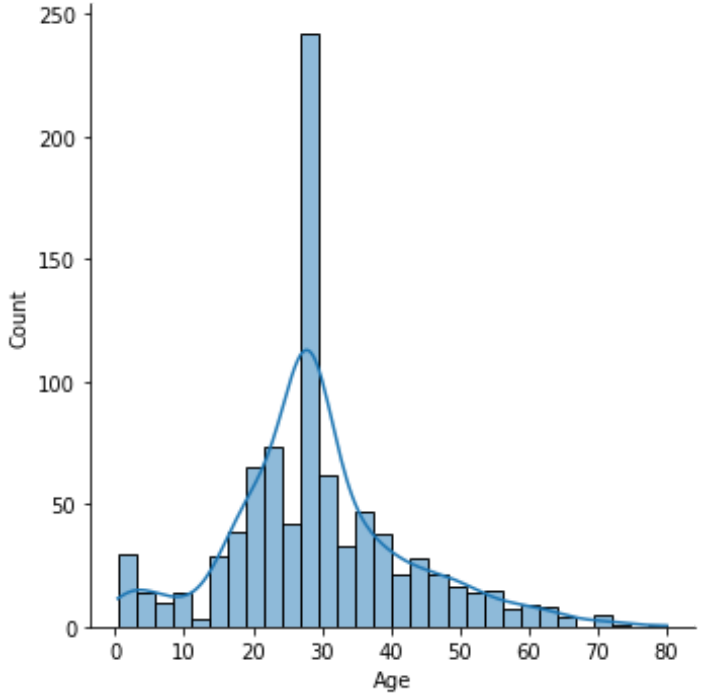
In [13]:
```
print(df_train.isnull().sum())
```

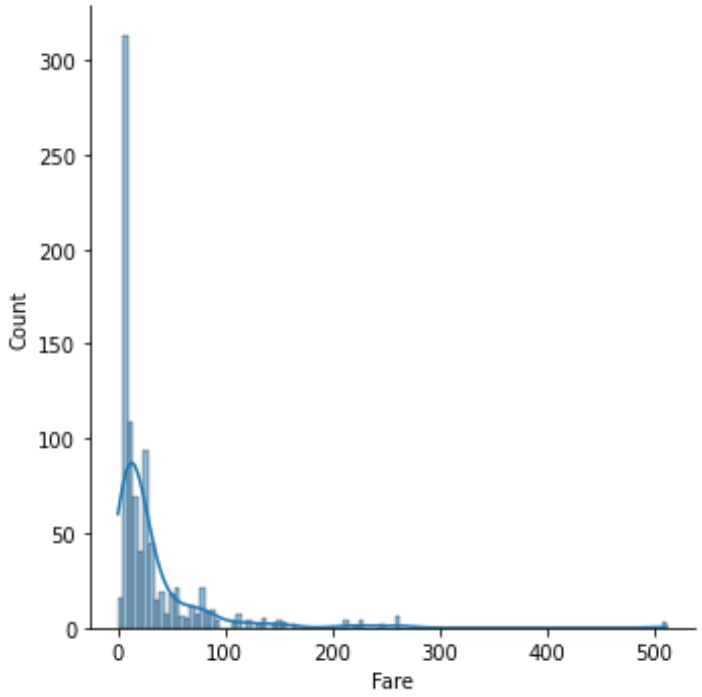
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    0
Title       0
dtype: int64
```

univariate analysis for feature having numerical datatype

In [14]:
```
def plot(feature):
    v = df_train[feature]
    for i in v.columns:
        sns.displot(df_train[i],kde=True)
        plt.show()
```

In [15]:
```
feature = ['Age','Fare']
plot(feature)
```

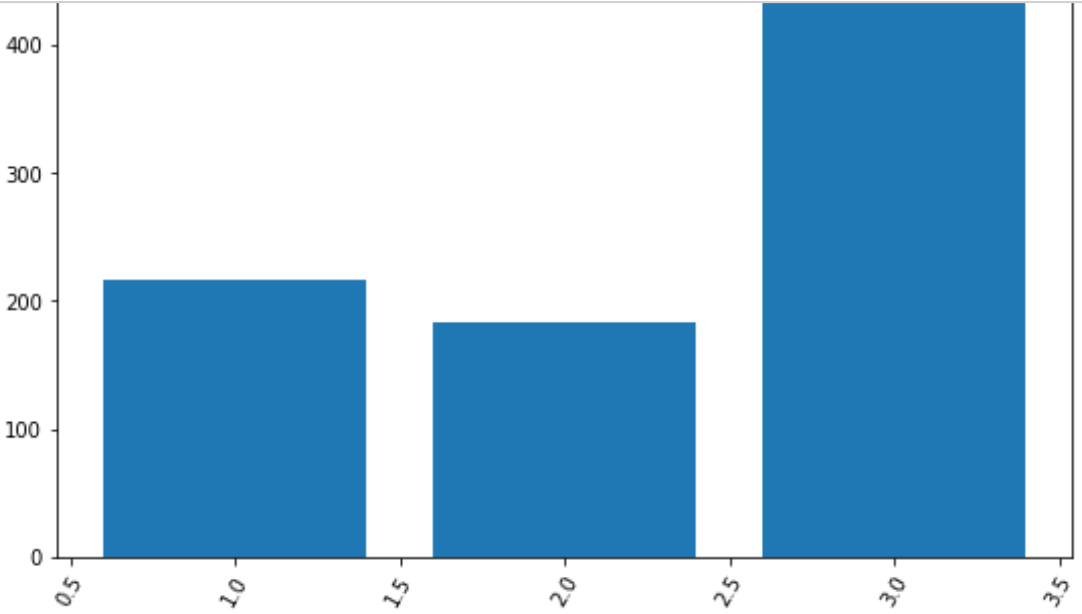




univariate analysis for feature having categorical datatypes

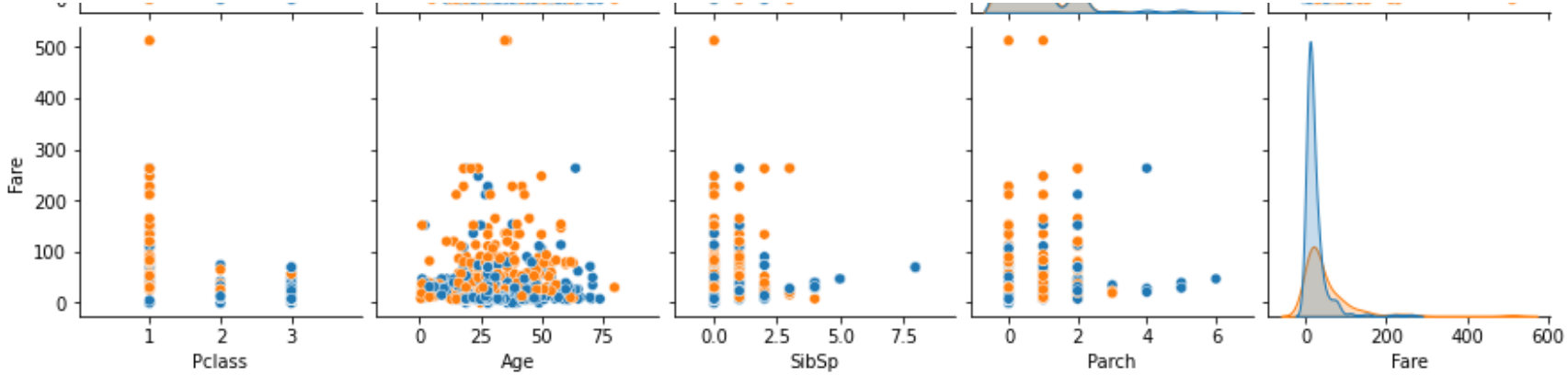
In [16]:
```
def plot(feature):
    v = df_train[feature]
    v_value = v.value_counts()
    plt.figure(figsize = (9,6))
    plt.xticks(rotation = 60)
    plt.bar(v_value.index,v_value)
    plt.show()
```

In [17]:
```
feature = ['Pclass','Sex','SibSp','Parch','Embarked','Title']
for i in feature:
    plot(i)
```



Multivariate Analysis

In [18]:
```
sns.pairplot(df_train,hue="Survived")
plt.show()
```



In [19]:
```
df_train.head(10)
```

Out[19]:

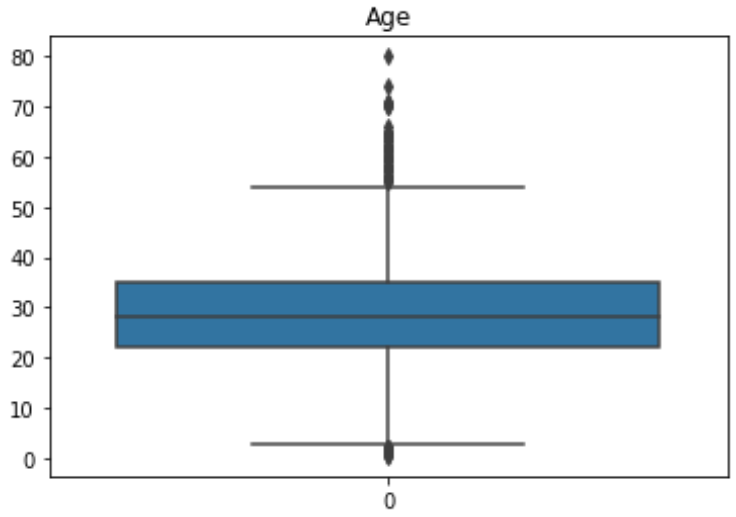
	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	0	0	373450	8.0500	S	Mr
5	0	3	male	28.0	0	0	330877	8.4583	Q	Mr
6	0	1	male	54.0	0	0	17463	51.8625	S	Mr
7	0	3	male	2.0	3	1	349909	21.0750	S	Master
8	1	3	female	27.0	0	2	347742	11.1333	S	Mrs
9	1	2	female	14.0	1	0	237736	30.0708	C	Mrs

Finding Outliers

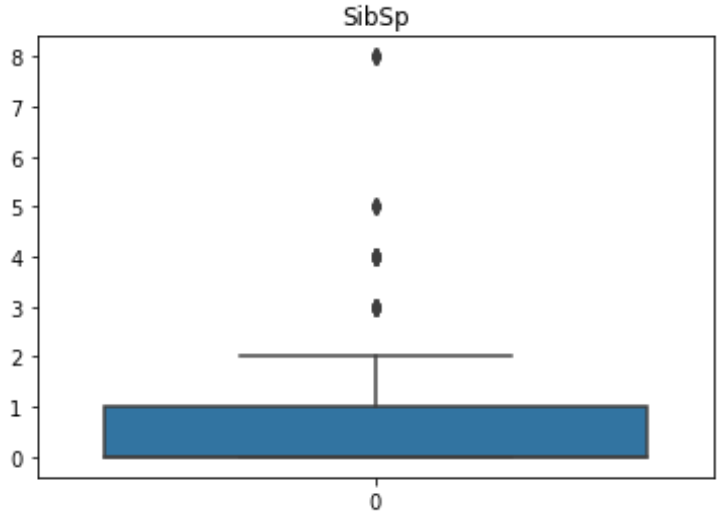
In [20]:

```
def detect_outliers(feature):
    threshold = 3
    outliers = []
    data = df_train[feature]
    mean = np.mean(data)
    std = np.std(data)
    for x in data:
        z_score = ( x - mean ) / std
        if z_score > threshold :
            outliers.append(x)
    plt.title(f"{i}")
    sns.boxplot(data=data)
    plt.show()
    print(f"outliers of {feature} column:- \n {outliers}")
```

In [21]:
```
features = ["Age","SibSp","Parch","Fare"]
for i in features :
    detect_outliers(i)
```

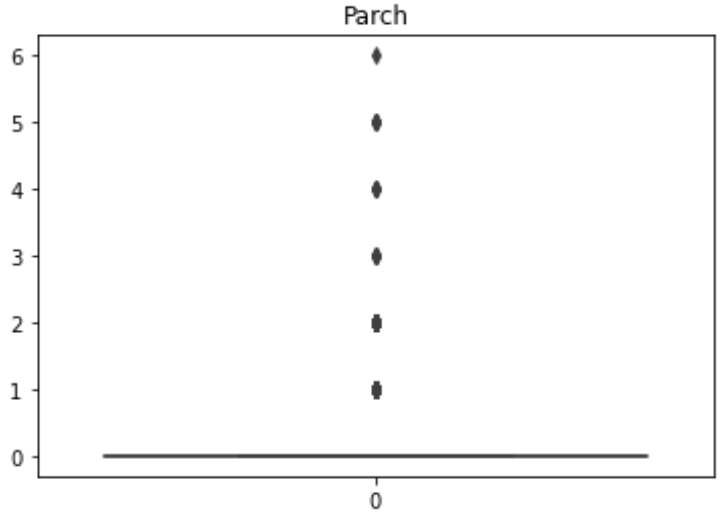



```
outliers of Age column:-
 [71.0, 70.5, 71.0, 80.0, 70.0, 70.0, 74.0]
```

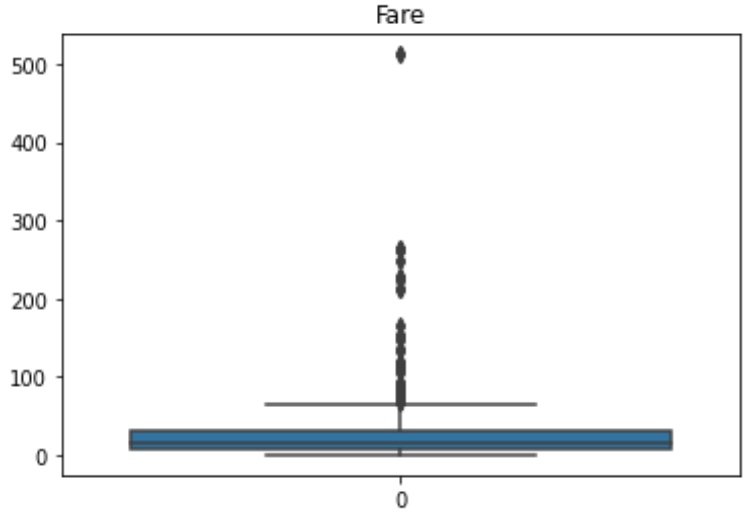



```
outliers of SibSp column:-
```

[4, 4, 5, 4, 5, 4, 8, 4, 4, 8, 4, 8, 4, 4, 4, 4, 8, 5, 5, 4, 4, 5, 4, 4, 8, 4, 4, 8, 4, 8]



outliers of Parch column:-
[5, 5, 3, 4, 4, 3, 4, 4, 5, 5, 6, 3, 3, 3, 5]



outliers of Fare column:-
[263.0, 263.0, 247.5208, 512.3292, 247.5208, 262.375, 263.0, 211.5, 227.525, 263.0, 221.7792, 227.525, 512.3292, 211.3375, 227.525, 227.525, 211.3375, 512.3292, 262.375, 211.3375]

In [22]:
```
df_train.head(10)
```

Out[22]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	0	0	373450	8.0500	S	Mr
5	0	3	male	28.0	0	0	330877	8.4583	Q	Mr
6	0	1	male	54.0	0	0	17463	51.8625	S	Mr
7	0	3	male	2.0	3	1	349909	21.0750	S	Master
8	1	3	female	27.0	0	2	347742	11.1333	S	Mrs
9	1	2	female	14.0	1	0	237736	30.0708	C	Mrs

In [23]:
```
#outliers of Age column:- [71.0, 70.5, 71.0, 80.0, 70.0, 70.0, 74.0]
o = [71.0, 70.5, 71.0, 80.0, 70.0, 70.0, 74.0]
a = df_train['Age']
for i in a:
    if i in o:
        df_train.Age.replace(df_train['Age'].median(),inplace=True)
```

In [24]:
```
# outliers of Parch column:- [5, 5, 3, 4, 4, 3, 4, 4, 5, 5, 6, 3, 3, 3, 5]
o = [5, 5, 3, 4, 4, 3, 4, 4, 5, 5, 6, 3, 3, 3, 5]
a = df_train['Parch']
for i in a:
    if i in o:
        df_train.Parch.replace(df_train['Parch'].median(),inplace=True)
```

In [25]:
```
# outliers of Fare column:- [263.0, 263.0, 247.5208, 512.3292, 247.5208, 262.375, 263.0, 211.5, 227.525, 263.0, 221.779
o = [263.0, 263.0, 247.5208, 512.3292, 247.5208, 262.375, 263.0, 211.5, 227.525, 263.0, 221.7792, 227.525, 512.3292, 21
a = df_train['Fare']
for i in a:
    if i in o:
        df_train.Fare.replace(df_train['Fare'].median(),inplace=True)
```

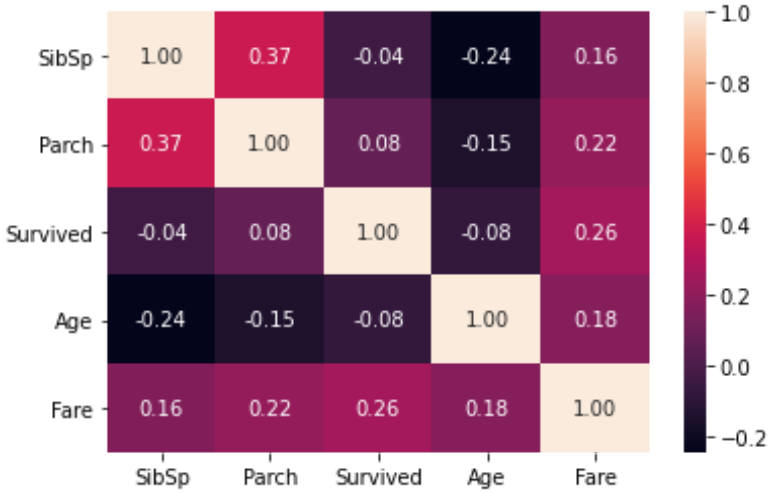
In [26]:
```
# outliers of SibSp column:- [4, 4, 5, 4, 5, 4, 8, 4, 4, 8, 4, 8, 4, 4, 4, 4, 8, 5, 5, 4, 4, 5, 4, 4, 8, 4, 4, 8, 4, 8]
o = [4, 4, 5, 4, 5, 4, 8, 4, 4, 8, 4, 8, 4, 4, 4, 4, 8, 5, 5, 4, 4, 5, 4, 4, 8, 4, 4, 8, 4, 8]
for i in a:
    if i in o:
        df_train.SibSp.replace(df_train['SibSp'].median(),inplace=True)
```

In [27]:
```
df_train.head(10)
```

Out[27]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	38.0	1	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	1	0	373450	8.0500	S	Mr
5	0	3	male	35.0	1	0	330877	8.4583	Q	Mr
6	0	1	male	54.0	1	0	17463	51.8625	S	Mr
7	0	3	male	2.0	3	0	349909	21.0750	S	Master
8	1	3	female	2.0	3	0	347742	11.1333	S	Mrs
9	1	2	female	14.0	1	0	237736	30.0708	C	Mrs

In [28]:
```
l = ["SibSp","Parch","Survived","Age","Fare"]
sns.heatmap(df[l].corr(),annot = True,fmt = ".2f")
plt.show()
```

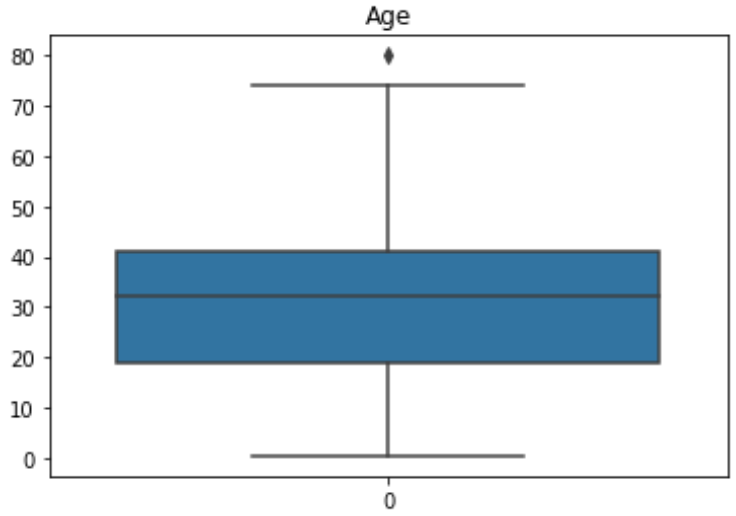


In [29]:
```
df_train.head()
```

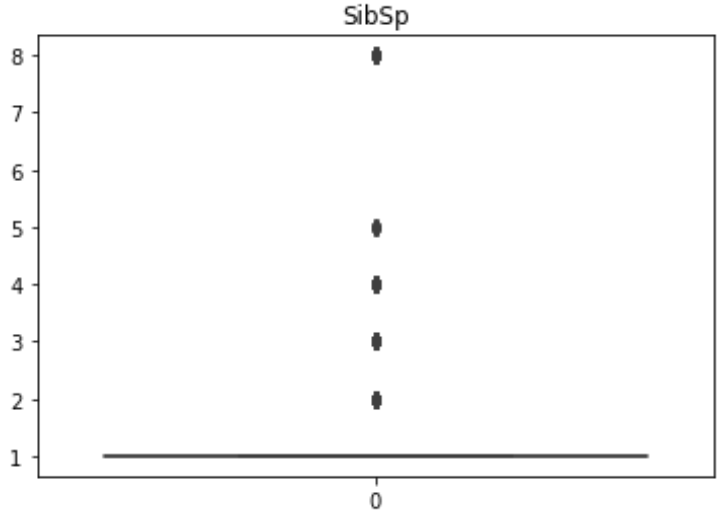
Out[29]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	38.0	1	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	1	0	373450	8.0500	S	Mr

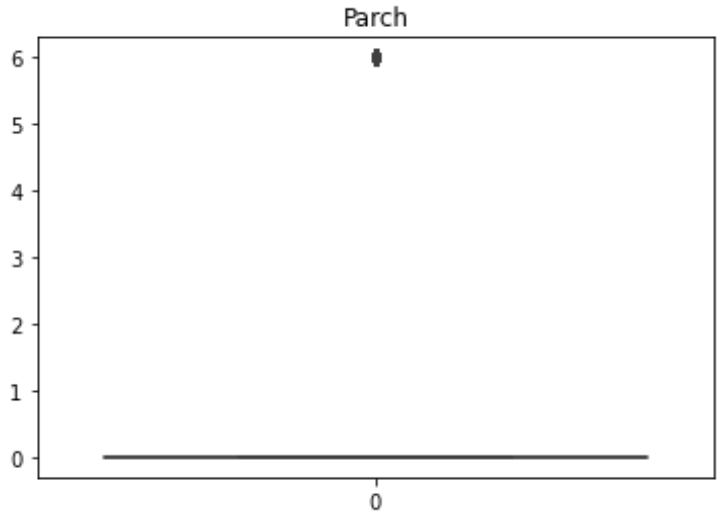
In [30]:
```
for i in features :
    detect_outliers(i)
```



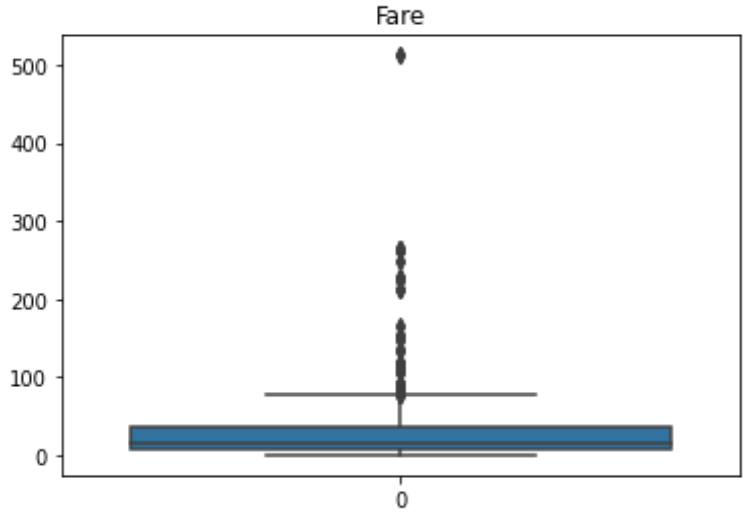
outliers of Age column:-
 [80.0]



```
outliers of SibSp column:-
 [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
```




```
outliers of Parch column:-
 []
```




```
outliers of Fare column:-
 [263.0, 263.0, 247.5208, 512.3292, 247.5208, 262.375, 263.0, 263.0, 263.0, 263.0, 263.0, 263.0, 211.5, 227.525, 263.
0, 221.7792, 227.525, 512.3292, 211.3375, 227.525, 227.525, 211.3375, 512.3292, 262.375, 211.3375]
```

In [32]:
```
df_train.head(10)
```

Out[32]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title
0	0	3	male	22.0	1	0	A/5 21171	7.2500	S	Mr
1	1	1	female	38.0	1	0	PC 17599	71.2833	C	Mrs
2	1	3	female	38.0	1	0	STON/O2. 3101282	7.9250	S	Miss
3	1	1	female	35.0	1	0	113803	53.1000	S	Mrs
4	0	3	male	35.0	1	0	373450	8.0500	S	Mr
5	0	3	male	35.0	1	0	330877	8.4583	Q	Mr
6	0	1	male	54.0	1	0	17463	51.8625	S	Mr
7	0	3	male	2.0	3	0	349909	21.0750	S	Master
8	1	3	female	2.0	3	0	347742	11.1333	S	Mrs
9	1	2	female	14.0	1	0	237736	30.0708	C	Mrs

In [33]:
```
df_train=pd.get_dummies(df_train,columns=["Title"])
df_train = pd.get_dummies(df_train, columns=["Embarked"])
df_train = pd.get_dummies(df_train, columns= ["Ticket"], prefix = "T")
df_train["Sex"] = df_train["Sex"].astype("category")
df_train = pd.get_dummies(df_train, columns=["Sex"])
df_train.head(2)
```

Out[33]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Title_ Capt	Title_ Col	Title_ Don	Title_ Dr	Title_ Jonkheer	Title_ Lady	Title_ Major	Title_ Master	Title_ Miss	Title_ Mlle	Title_ Mme	Title_ Mr	Title_ Mrs
0	0	3	22.0	1	0	7.2500	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	38.0	1	0	71.2833	0	0	0	0	0	0	0	0	0	0	0	0	1

In [34]:
```
df_train.head(10)
```

Out[34]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Title_ Capt	Title_ Col	Title_ Don	Title_ Dr	Title_ Jonkheer	Title_ Lady	Title_ Major	Title_ Master	Title_ Miss	Title_ Mlle	Title_ Mme	Title_ Mr	Title_ Mrs
0	0	3	22.0	1	0	7.2500	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	38.0	1	0	71.2833	0	0	0	0	0	0	0	0	0	0	0	0	1
2	1	3	38.0	1	0	7.9250	0	0	0	0	0	0	0	0	1	0	0	0	0
3	1	1	35.0	1	0	53.1000	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	3	35.0	1	0	8.0500	0	0	0	0	0	0	0	0	0	0	0	1	0
5	0	3	35.0	1	0	8.4583	0	0	0	0	0	0	0	0	0	0	0	1	0
6	0	1	54.0	1	0	51.8625	0	0	0	0	0	0	0	0	0	0	0	1	0
7	0	3	2.0	3	0	21.0750	0	0	0	0	0	0	0	1	0	0	0	0	0
8	1	3	2.0	3	0	11.1333	0	0	0	0	0	0	0	0	0	0	0	0	1
9	1	2	14.0	1	0	30.0708	0	0	0	0	0	0	0	0	0	0	0	0	1

In [50]:
```
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, VotingClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [51]:
```
x = df_train.drop("Survived", axis = 1)
y = df_train["Survived"]
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.33, random_state = 42)
```

In [52]:
```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\linear_model\_logistic.py:765: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.
html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mod
ules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[52]:
```
LogisticRegression()
```

In [53]:
```
y_predicted=model.predict(x_test)
```

In [54]:
```
model.score(x_test,y_test)
```

Out[54]:
```
0.823728813559322
```

In [55]:
```
from sklearn.metrics import r2_score,mean_squared_error
print(f'R^2 : {r2_score(y_test,y_predicted)}')
print(f'MSE : {mean_squared_error(y_test,y_predicted)}')
print(f'RMSE: {np.sqrt(mean_squared_error(y_test, y_predicted))}')
```

```
R^2 : 0.2695238095238095
MSE : 0.17627118644067796
RMSE: 0.41984662251907895
```

In [56]:

```
random_state = 42
classifier = [DecisionTreeClassifier(random_state = random_state),
              SVC(random_state = random_state),
              RandomForestClassifier(random_state = random_state),
              LogisticRegression(random_state = random_state),
              KNeighborsClassifier()]

dt_grid_parameter = {"min_samples_split" : range(10,500,20),
                "max_depth": range(1,20,2)}

svc_grid_parameter = {"kernel" : ["rbf"],
                 "gamma": [0.001, 0.01, 0.1, 1],
                 "C": [1,10,50,100,200,300,1000]}

rf_grid_parameter = {"max_features": [1,3,10],
                "min_samples_split":[2,3,10],
                "min_samples_leaf":[1,3,10],
                "bootstrap":[False],
                "n_estimators":[100,300],
                "criterion":["gini"]}

lr_grid_parameter = {"C":np.logspace(-3,3,7),
                     "penalty": ["l1","l2"]}

knn_grid_parameter = {"n_neighbors": np.linspace(1,19,10, dtype = int).tolist(),
                 "weights": ["uniform","distance"],
                 "metric":["euclidean","manhattan"]}
classifier_param = [dt_grid_parameter,
                    svc_grid_parameter,
                    rf_grid_parameter,
                    lr_grid_parameter,
                    knn_grid_parameter]
```

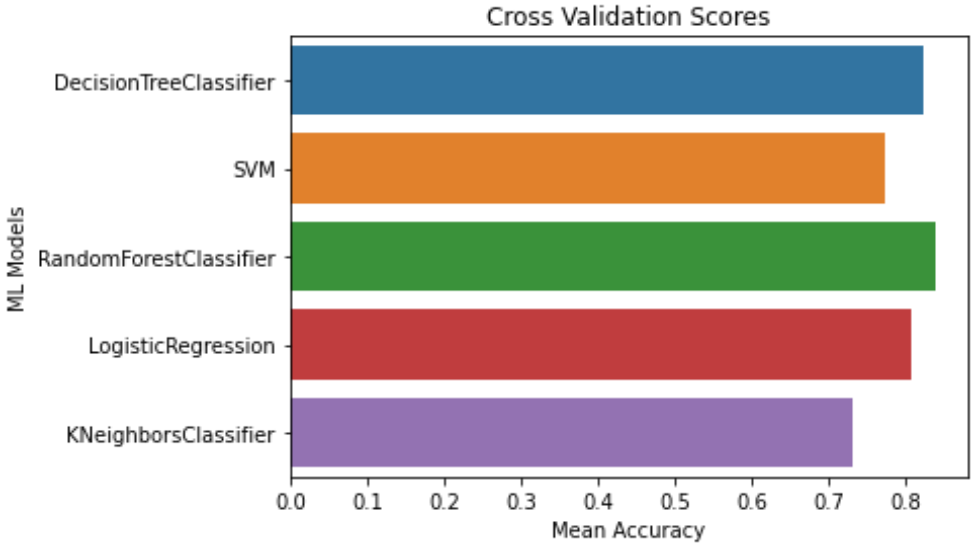
In [58]:
```
cv_result = []
best_estimators = []
for i in range(len(classifier)):
    clf = GridSearchCV(classifier[i], param_grid=classifier_param[i], cv = StratifiedKFold(n_splits = 10), scoring = "a
    clf.fit(x_train,y_train)
    cv_result.append(clf.best_score_)
    best_estimators.append(clf.best_estimator_)
    print(cv_result[i])
```

```
Fitting 10 folds for each of 250 candidates, totalling 2500 fits
0.8238983050847457
Fitting 10 folds for each of 28 candidates, totalling 280 fits
0.7734745762711865
Fitting 10 folds for each of 54 candidates, totalling 540 fits
0.8389830508474576
Fitting 10 folds for each of 14 candidates, totalling 140 fits

C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\model_selection\_search.py:925: UserWa
rning: One or more of the test scores are non-finite: [       nan 0.68446328        nan 0.78163842        nan 0.791977
4
        nan 0.80706215        nan 0.78183616        nan 0.78019774
        nan 0.8070339 ]
  category=UserWarning
C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\linear_model\_logistic.py:765: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.
html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mod
ules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

0.8070621468926553
Fitting 10 folds for each of 40 candidates, totalling 400 fits
0.7315536723163841
```

In [59]:
```
cv_results = pd.DataFrame({"Cross Validation Means":cv_result, "ML Models":["DecisionTreeClassifier", "SVM","RandomFore
             "LogisticRegression",
             "KNeighborsClassifier"]})

g = sns.barplot("Cross Validation Means", "ML Models", data = cv_results)
g.set_xlabel("Mean Accuracy")
g.set_title("Cross Validation Scores")
```

```
C:\Users\kants\AppData\Local\Programs\Python\Python37\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass
the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, a
nd passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[59]:
```
Text(0.5, 1.0, 'Cross Validation Scores')
```



In [63]:
```
poll = VotingClassifier(estimators = [("dt",best_estimators[0]),
                                      ("rfc",best_estimators[2]),
                                      ("lr",best_estimators[3])],
                                      voting = "soft", n_jobs = -1)
poll = votingC.fit(x_train, y_train)
print(accuracy_score(votingC.predict(x_test),y_test))
```

```
0.8305084745762712
```

In [ ]: