Tatanic Test Train Project

- 1) Understand the shape of the data (Histograms, box plots, etc.)
- 2) Data Cleaning
- 3) Data Exploration
- 4) Feature Engineering
- 5) Data Preprocessing for Model
- 6) Basic Model Building

Project Planning

When starting any project, I like to outline the steps that I plan to take. Below is the rough outline that I created for this project using commented cells.

Understand nature of the data .info() .describe()

Histograms and boxplots

Value counts

Missing data

Correlation between the metrics

Explore interesting themes

```
# Wealthy survive?
# By location
# Age scatterplot with ticket price
# Young and wealthy Variable?
# Total spent?
```

Feature engineering

preprocess data together or use a transformer?

```
    # use label for train and test
```

Scaling?

Model Baseline

In [1]:
```
## import Path of the file
import os
os.getcwd()
```

Out[1]:
```
'C:\\Users\\ap983\\Desktop\\Pandey  IMp\\COMPLETE PYTHON PROJECT\\7.TITANIC T
RAIN TEST  FULL PROJECT'
```

In [2]:
```
# ignore the All Warnings in the Projets
import warnings

# Filter warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
## import Manipulated libraries
import numpy as np
import pandas as pd

## import visualisation libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
## import the dataset
# titanic test data
# titanic teain data
test = pd.read_csv('titanic_test.csv')
train = pd.read_csv('titanic_train.csv')
```

In [5]:
```
## observe the top five rows in the dataset
test.head()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [6]:
```
## observe the top five rows in the dataset

train.head()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N

In [7]:
```
test.tail()
```

Out[7]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	

In [8]:
```
train.tail()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN

In [9]:
```
test.shape
```

Out[9]:
```
(418, 11)
```

In [10]:
```
train.shape
```

Out[10]:
```
(891, 12)
```

In [11]:
```
test.columns
```

Out[11]:
```
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [12]:
```
train.columns
```

Out[12]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [13]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [14]:
```
train['train_test'] = 1
test['train_test'] = 0
test['Survived'] = np.NaN
all_data = pd.concat([train,test])
all_data.columns
```

Out[14]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'train_test'],
      dtype='object')
```

In [15]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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 
 12  train_test   891 non-null    int64  
dtypes: float64(2), int64(6), object(5)
memory usage: 90.6+ KB
```

In [16]:
```
train.describe()
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	trai
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 [17]:
```
def_num = train[['Age','SibSp','Parch','Fare']]
def_num
```

Out[17]:

	Age	SibSp	Parch	Fare
0	22.0	1	0	7.2500
1	38.0	1	0	71.2833
2	26.0	0	0	7.9250
3	35.0	1	0	53.1000
4	35.0	0	0	8.0500
...	...	...	...	...
886	27.0	0	0	13.0000
887	19.0	0	0	30.0000
888	NaN	1	2	23.4500
889	26.0	0	0	30.0000
890	32.0	0	0	7.7500

891 rows × 4 columns

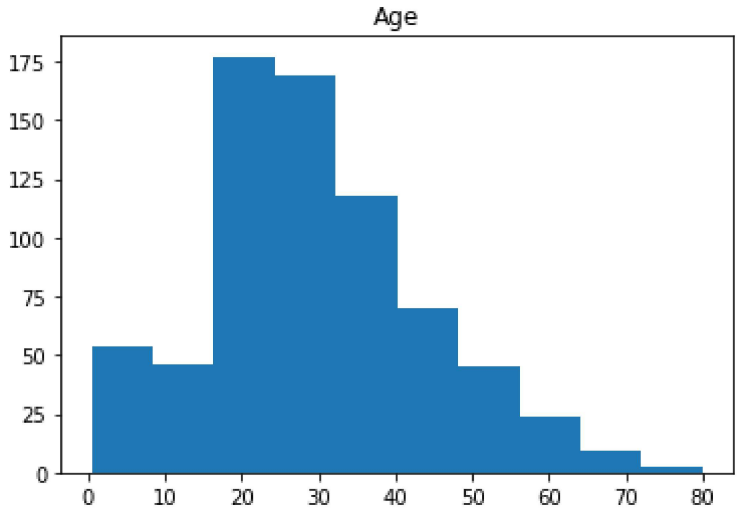
In [18]:
```
def_cat = train[['Survived','Pclass','Sex','Ticket','Cabin','Embarked']]
def_cat
```

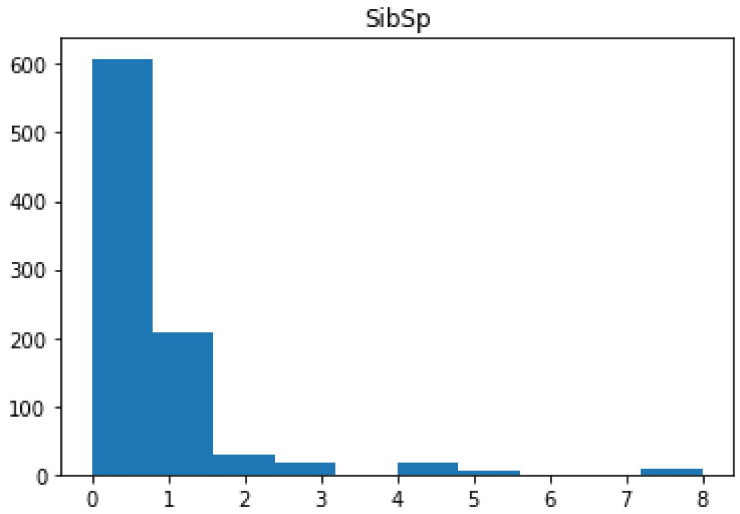
Out[18]:

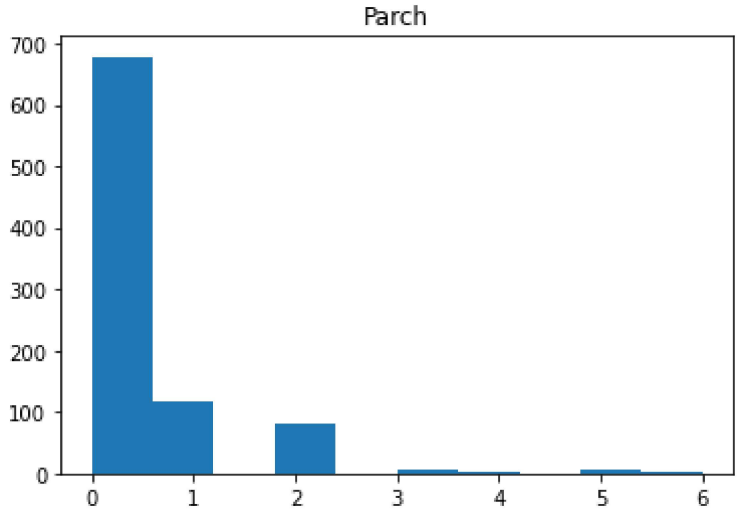
	Survived	Pclass	Sex	Ticket	Cabin	Embarked
0	0	3	male	A/5 21171	NaN	S
1	1	1	female	PC 17599	C85	C
2	1	3	female	STON/O2. 3101282	NaN	S
3	1	1	female	113803	C123	S
4	0	3	male	373450	NaN	S
...	...	...	...	...	...	...
886	0	2	male	211536	NaN	S
887	1	1	female	112053	B42	S
888	0	3	female	W./C. 6607	NaN	S
889	1	1	male	111369	C148	C
890	0	3	male	370376	NaN	Q

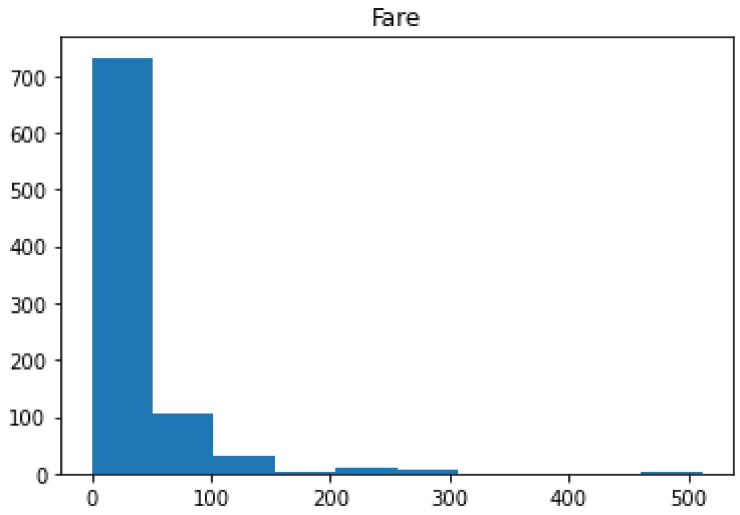
891 rows × 6 columns

In [19]:
```
for i in def_num.columns:
    plt.hist(def_num[i])
    plt.title(i)
    plt.show()
```



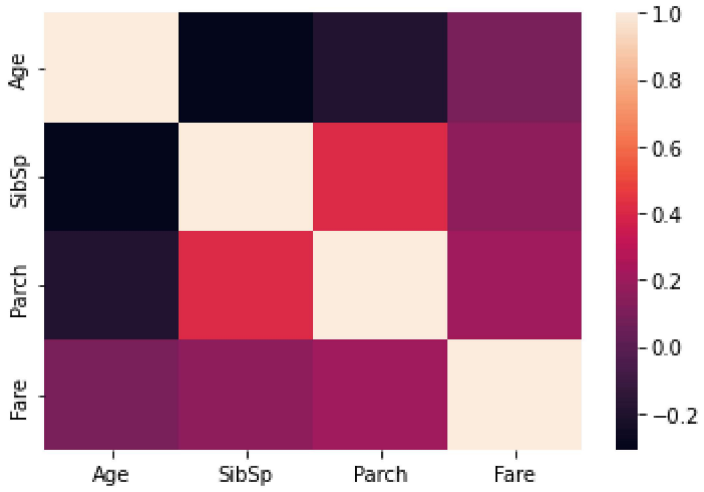






In [20]:
```
sns.heatmap(def_num.corr())
```

Out[20]: <AxesSubplot:>

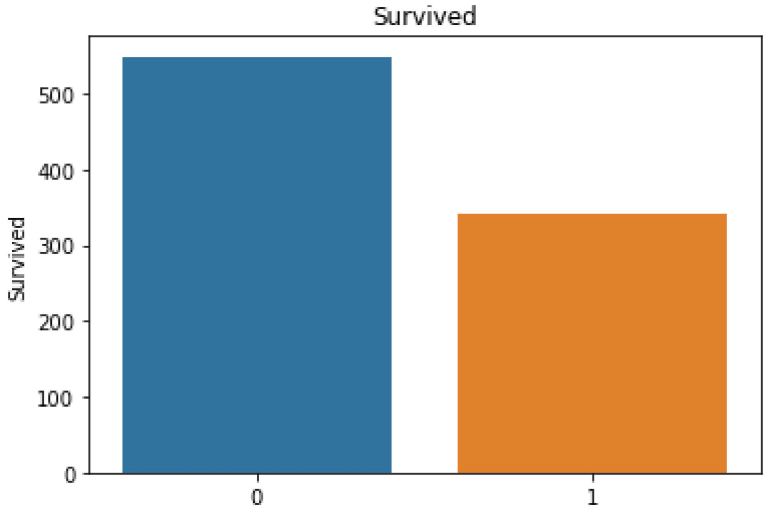


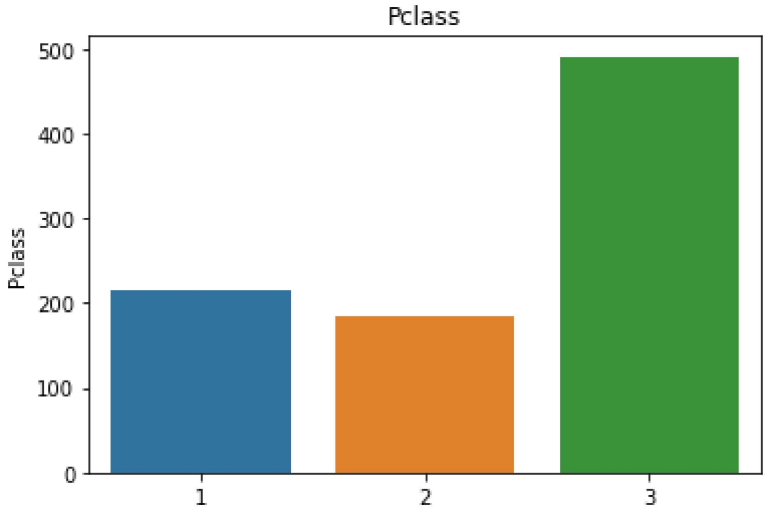
In [21]:
```
pd.pivot_table(train, index = 'Survived', values =['Age','SibSp','Parch','Fare'
```

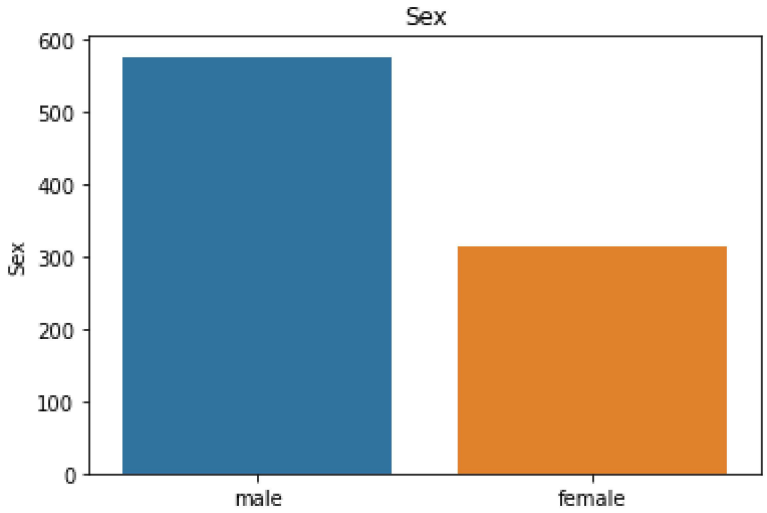
Out[21]:

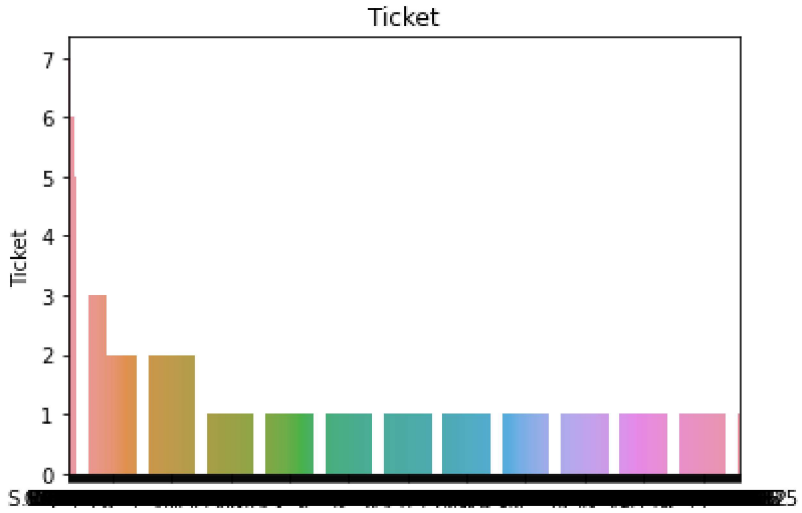
Survived	Age	Fare	Parch	SibSp
0	30.626179	22.117887	0.329690	0.553734
1	28.343690	48.395408	0.464912	0.473684

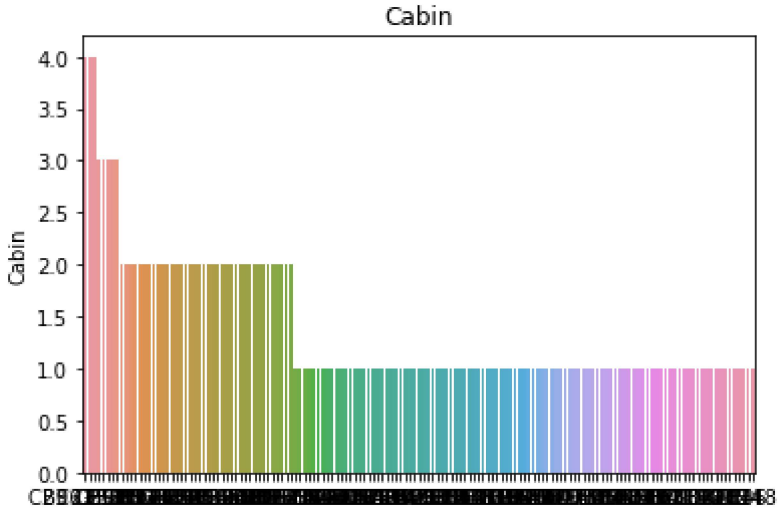
In [22]:
```
for i in def_cat.columns:
    sns.barplot(def_cat[i].value_counts().index,def_cat[i].value_counts()).set_
    plt.show()
```

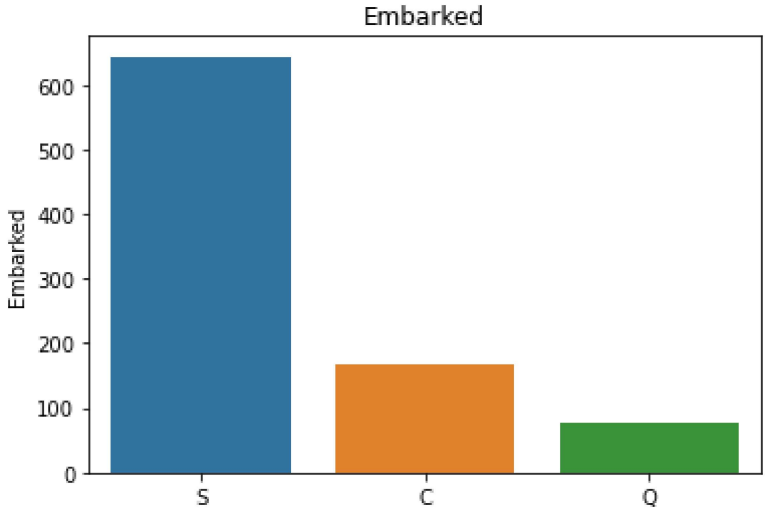












Using Feature Engineering Method

In [23]:
```
def_cat.Cabin
train['cabin_multiple'] = train.Cabin.apply(lambda x: 0 if pd.isna(x) else len(
train['cabin_multiple'].value_counts()
```

Out[23]:
```
0    687
1    180
2     16
3      6
4      2
Name: cabin_multiple, dtype: int64
```

In [24]:
```
## Check the Pivot
pd.pivot_table(train, index = 'Survived', columns ='cabin_multiple',values = '
```

Out[24]:

cabin_multiple	0	1	2	3	4
Survived					
0	481.0	58.0	7.0	3.0	NaN
1	206.0	122.0	9.0	3.0	2.0

In [25]:
```
# n stands for null
# in this case we will treat null values like it's own category
train['cabin_adv'] = train.Cabin.apply(lambda x:str(x)[0])
```

In [26]:
```
#comparing survival rates by cabin
print(train.cabin_adv.value_counts())
pd.pivot_table(train,index='Survived',columns='cabin_adv',values = 'Name', aggf
```

```
n    687
C     59
B     47
D     33
E     32
A     15
F     13
G      4
T      1
Name: cabin_adv, dtype: int64
```

Out[26]:

cabin_adv	A	B	C	D	E	F	G	T	n
Survived									
0	8.0	12.0	24.0	8.0	8.0	5.0	2.0	1.0	481.0
1	7.0	35.0	35.0	25.0	24.0	8.0	2.0	NaN	206.0

In [27]:
```
train['numeric_ticket'] = train.Ticket.apply(lambda x: 1 if x.isnumeric() else
train['ticket_letters'] = train.Ticket.apply(lambda x:''.join(x.split(' ')[:-1]
```

In [28]:
```
train['numeric_ticket'].value_counts()
```

Out[28]:
```
1    661
0    230
Name: numeric_ticket, dtype: int64
```

In [29]:
```
train['ticket_letters'].value_counts()
```

Out[29]:
```
0            665
pc            60
ca            41
a5            21
stono2        18
sotonoq       15
scparis       11
wc            10
a4             7
soc            6
fcc            5
c              5
sopp           3
pp             3
wep            3
ppp            2
scah           2
sotono2        2
swpp           2
fc             1
scahbasle      1
as             1
sp             1
sc             1
scow           1
fa             1
sop            1
sca4           1
casoton        1
Name: ticket_letters, dtype: int64
```

In [30]:
```
#difference in numeric vs non-numeric tickets in survival rate
pd.pivot_table(train,index='Survived',columns='numeric_ticket', values = 'Ticke
```

Out[30]:

numeric_ticket	0	1
Survived		
0	142	407
1	88	254

In [31]:
```
#survival rate across different tyicket types
pd.pivot_table(train,index='Survived',columns='ticket_letters', values = 'Ticke
```

Out[31]:

ticket_letters	0	a4	a5	as	c	ca	casoton	fa	fc	fcc	...	soc	sop	sopp	s
Survived															
0	410.0	7.0	19.0	1.0	3.0	27.0	1.0	1.0	1.0	1.0	...	5.0	1.0	3.0	
1	255.0	NaN	2.0	NaN	2.0	14.0	NaN	NaN	NaN	4.0	...	1.0	NaN	NaN	

2 rows × 29 columns

In [32]:
```
#feature engineering on person's title
train.Name.head(50)
train['name_title'] = train.Name.apply(lambda x:x.split(',')[1].split('.')[0].s
train['name_title'].value_counts()
```

Out[32]:
```
Mr              517
Miss            182
Mrs             125
Master           40
Dr                7
Rev               6
Mlle              2
Major             2
Col               2
the Countess      1
Capt              1
Ms                1
Sir               1
Lady              1
Mme               1
Don               1
Jonkheer          1
Name: name_title, dtype: int64
```

Step 5: Data preprocessing for model

In this segment, we make our data, model-ready. The objectives we have to fulfill are listed below:

- Drop the null values from the Embarked column
- Include only relevant data
- Categorically transform all of the data, using something called atransformer.
- Impute data with the central tendencies for age and fare.
- Normalize the fare column to have a more normal distribution.
- using standard scaler scale data 0-1

In [33]:
```
#create all categorical variables that we did above for both training and test
all_data['cabin_multiple'] = all_data.Cabin.apply(lambda x: 0 if pd.isna(x) els
all_data['cabin_adv'] = all_data.Cabin.apply(lambda x: str(x)[0])
all_data['numeric_ticket'] = all_data.Ticket.apply(lambda x: 1 if x.isnumeric()
all_data['ticket_letters'] = all_data.Ticket.apply(lambda x: ''.join(x.split('
all_data['name_title'] = all_data.Name.apply(lambda x: x.split(',')[1].split('.
```

In [34]:
```
#impute nulls for continuous data
#all_data.Age = all_data.Age.fillna(training.Age.mean())
all_data.Age = all_data.Age.fillna(train.Age.median())
all_data.Age
```

Out[34]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
413    28.0
414    39.0
415    38.5
416    28.0
417    28.0
Name: Age, Length: 1309, dtype: float64
```

In [35]:
```
#all_data.Fare = all_data.Fare.fillna(training.Fare.mean())
all_data.Fare = all_data.Fare.fillna(train.Fare.median())
all_data.Fare
```

Out[35]:
```
0        7.2500
1       71.2833
2        7.9250
3       53.1000
4        8.0500
         ...
413      8.0500
414    108.9000
415      7.2500
416      8.0500
417     22.3583
Name: Fare, Length: 1309, dtype: float64
```

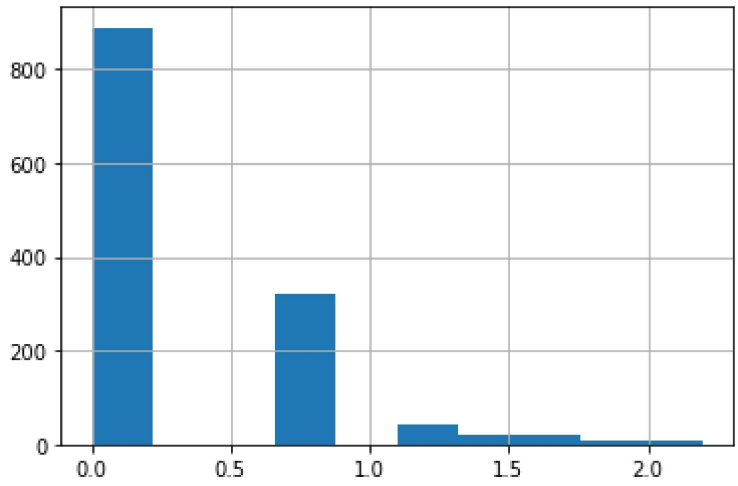
In [36]:
```
#drop null 'embarked' rows. Only 2 instances of this in training and 0 in test
all_data.dropna(subset=['Embarked'],inplace = True)
all_data.head()
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C1
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N

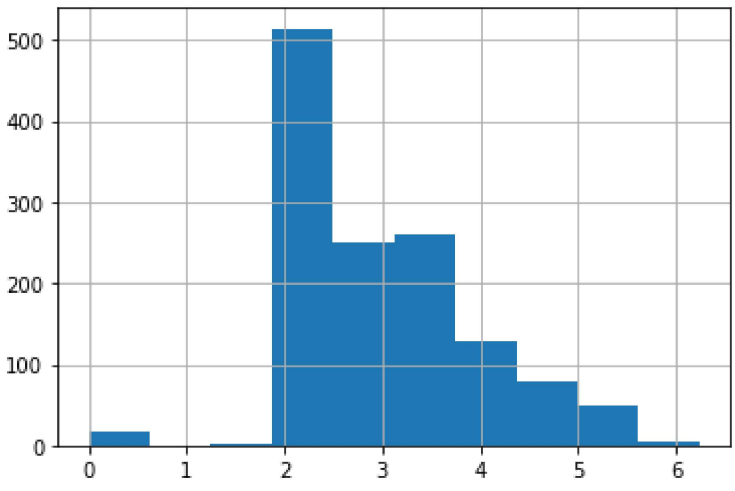
In [37]:
```
#tried log norm of sibsp (not used)
all_data['norm_sibsp'] = np.log(all_data.SibSp+1)
all_data['norm_sibsp'].hist()
```

Out[37]: <AxesSubplot:>



In [38]:
```
# Log norm of fare (used)
all_data['norm_fare'] = np.log(all_data.Fare+1)
all_data['norm_fare'].hist()
```

Out[38]: <AxesSubplot:>



In [39]:
```
# converted fare to category for pd.get_dummies()
all_data.Pclass = all_data.Pclass.astype(str)
all_data
```

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050
...	...	...	...	...	...	...	...	...	...	.
413	1305	NaN	3	Spector, Mr. Woolf	male	28.0	0	0	A.5. 3236	8.050
414	1306	NaN	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.900
415	1307	NaN	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.250
416	1308	NaN	3	Ware, Mr. Frederick	male	28.0	0	0	359309	8.050
417	1309	NaN	3	Peter, Master. Michael J	male	28.0	1	1	2668	22.358

1307 rows × 20 columns

In [40]:
```
#created dummy variables from categories (also can use OneHotEncoder)
all_dummies = pd.get_dummies(all_data[['Pclass','Sex','Age','SibSp','Parch','nc
all_dummies
```

Out[40]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	train_test	Pclass_1	Pclass_2
0	22.0	1	0	2.110213	0	0	1	0	0
1	38.0	1	0	4.280593	1	0	1	1	0
2	26.0	0	0	2.188856	0	0	1	0	0
3	35.0	1	0	3.990834	1	1	1	1	0
4	35.0	0	0	2.202765	0	1	1	0	0
...	...	...	...	...	...	...	...	...	...
413	28.0	0	0	2.202765	0	0	0	0	0
414	39.0	0	0	4.699571	1	0	0	1	0
415	38.5	0	0	2.110213	0	0	0	0	0
416	28.0	0	0	2.202765	0	1	0	0	0
417	28.0	1	1	3.150952	0	1	0	0	0

1307 rows × 42 columns

In [41]:
```
#Split to train test again
X_train = all_dummies[all_dummies.train_test == 1].drop(['train_test'], axis =1
X_train
```

Out[41]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	Pclass_1	Pclass_2	Pclass_3
0	22.0	1	0	2.110213	0	0	0	0	1
1	38.0	1	0	4.280593	1	0	1	0	0
2	26.0	0	0	2.188856	0	0	0	0	1
3	35.0	1	0	3.990834	1	1	1	0	0
4	35.0	0	0	2.202765	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...
886	27.0	0	0	2.639057	0	1	0	1	0
887	19.0	0	0	3.433987	1	1	1	0	0
888	28.0	1	2	3.196630	0	0	0	0	1
889	26.0	0	0	3.433987	1	1	1	0	0
890	32.0	0	0	2.169054	0	1	0	0	1

889 rows × 41 columns

In [42]:
```
X_test = all_dummies[all_dummies.train_test == 0].drop(['train_test'], axis =1)
X_test
```

Out[42]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	Pclass_1	Pclass_2	Pclass_3
0	34.5	0	0	2.178064	0	1	0	0	1
1	47.0	1	0	2.079442	0	1	0	0	1
2	62.0	0	0	2.369075	0	1	0	1	0
3	27.0	0	0	2.268252	0	1	0	0	1
4	22.0	1	1	2.586824	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...
413	28.0	0	0	2.202765	0	0	0	0	1
414	39.0	0	0	4.699571	1	0	1	0	0
415	38.5	0	0	2.110213	0	0	0	0	1
416	28.0	0	0	2.202765	0	1	0	0	1
417	28.0	1	1	3.150952	0	1	0	0	1

418 rows × 41 columns

In [43]:
```
y_train = all_data[all_data.train_test==1].Survived
y_train.shape
```

Out[43]: (889,)

In [44]:
```
# Scale data
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
all_dummies_scaled = all_dummies.copy()
all_dummies_scaled[['Age','SibSp','Parch','norm_fare']]= scale.fit_transform(al
all_dummies_scaled
```

Out[44]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	train_test	Pclass
0	-0.580261	0.480272	-0.445407	-0.896331	0	0	1	
1	0.662297	0.480272	-0.445407	1.347870	1	0	1	
2	-0.269621	-0.479537	-0.445407	-0.815013	0	0	1	
3	0.429318	0.480272	-0.445407	1.048255	1	1	1	
4	0.429318	-0.479537	-0.445407	-0.800632	0	1	1	
...	...	...	...	...	...	...	...	
413	-0.114301	-0.479537	-0.445407	-0.800632	0	0	0	
414	0.739957	-0.479537	-0.445407	1.781098	1	0	0	
415	0.701127	-0.479537	-0.445407	-0.896331	0	0	0	
416	-0.114301	-0.479537	-0.445407	-0.800632	0	1	0	
417	-0.114301	0.480272	0.709647	0.179806	0	1	0	

1307 rows × 42 columns

In [45]:
```
X_train_scaled = all_dummies_scaled[all_dummies_scaled.train_test == 1].drop(['
X_train_scaled
```

Out[45]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	Pclass_1	Pclass_
0	-0.580261	0.480272	-0.445407	-0.896331	0	0	0	
1	0.662297	0.480272	-0.445407	1.347870	1	0	1	
2	-0.269621	-0.479537	-0.445407	-0.815013	0	0	0	
3	0.429318	0.480272	-0.445407	1.048255	1	1	1	
4	0.429318	-0.479537	-0.445407	-0.800632	0	1	0	
...	...	...	...	...	...	...	...	
886	-0.191961	-0.479537	-0.445407	-0.349500	0	1	0	
887	-0.813240	-0.479537	-0.445407	0.472468	1	1	1	
888	-0.114301	0.480272	1.864701	0.227038	0	0	0	
889	-0.269621	-0.479537	-0.445407	0.472468	1	1	1	
890	0.196338	-0.479537	-0.445407	-0.835490	0	1	0	

889 rows × 41 columns

In [46]:
```
X_test_scaled = all_dummies_scaled[all_dummies_scaled.train_test == 0].drop(['t
X_test_scaled
```

Out[46]:

	Age	SibSp	Parch	norm_fare	cabin_multiple	numeric_ticket	Pclass_1	Pclass_
0	0.390488	-0.479537	-0.445407	-0.826172	0	1	0	
1	1.361236	0.480272	-0.445407	-0.928150	0	1	0	
2	2.526134	-0.479537	-0.445407	-0.628665	0	1	0	
3	-0.191961	-0.479537	-0.445407	-0.732917	0	1	0	
4	-0.580261	0.480272	0.709647	-0.403510	0	1	0	
...	...	...	...	...	...	...	...	
413	-0.114301	-0.479537	-0.445407	-0.800632	0	0	0	
414	0.739957	-0.479537	-0.445407	1.781098	1	0	1	
415	0.701127	-0.479537	-0.445407	-0.896331	0	0	0	
416	-0.114301	-0.479537	-0.445407	-0.800632	0	1	0	
417	-0.114301	0.480272	0.709647	0.179806	0	1	0	

418 rows × 41 columns

In [47]:
```
y_train = all_data[all_data.train_test==1].Survived
y_train
```

Out[47]:
```
0      0.0
1      1.0
2      1.0
3      1.0
4      0.0
      ...
886    0.0
887    1.0
888    0.0
889    1.0
890    0.0
Name: Survived, Length: 889, dtype: float64
```

Step 6: Model Deployment

Here we will simply deploy the various models with default parameters and see which one yields the best result. The models can further be tuned for better performance but are not in the scope of this one article. The models we will run are:

- Logistic regression
- K Nearest Neighbour
- Support Vector classifier

First, we import the necessary models

In [48]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

1) Logistic Regression

In [49]:
```
lr = LogisticRegression(max_iter = 2000)
cv = cross_val_score(lr,X_train_scaled,y_train,cv=5)
cv
```

Out[49]:
```
array([0.8258427 , 0.80898876, 0.80337079, 0.82022472, 0.85310734])
```

In [50]:
```
cv.mean()
```

Out[50]:
```
0.8223068621849807
```

Logistic regression: 82.2%

2) K Nearest Neighbour

In [51]:
```
knn = KNeighborsClassifier()
cv = cross_val_score(knn,X_train_scaled,y_train,cv=5)
cv
```

Out[51]:
```
array([0.79775281, 0.79213483, 0.83146067, 0.79775281, 0.85310734])
```

In [52]:
```
cv.mean()
```

Out[52]:
```
0.8144416936456548
```

K Nearest Neighbour: 81.4%

3) Support Vector Classifier

In [53]:
```
svc = SVC(probability = True)
cv = cross_val_score(svc,X_train_scaled,y_train,cv=5)
cv
```

Out[53]:
```
array([0.85393258, 0.82022472, 0.8258427 , 0.80337079, 0.86440678])
```

In [54]:
```
cv.mean()
```

Out[54]:
```
0.8335555132355742
```

SVC: 83.3%

Therefore the accuracy of the models are:

As you can see we get decent accuracy with all our models, but the best one is SVC. And voila, just like that you've completed your first data science project

Thank you