Data Mining Assignment.

Submitted By: Amrutha Dinesh CS A Roll no: 17

```
# Data Mining Assignment.
# Aim: To perform data preprocessing and do a classification.
#About: Data preprocessing using median is done for Titanic dataset. Random Forest classifier is used for classification
# Submitted By : Amrutha Dinesh CS A Roll no:17

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
import seaborn as sns
import os
import matplotlib.pyplot as plt

for dirname, _, filenames in os.walk('/kaggle/input'):
   for filename in filenames:
       print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

[6]:

```
train_data = pd.read_csv("/kaggle/input/titanic/train.csv")
train_data.head()
```

[6]:

	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

[7]:

```
test_data = pd.read_csv("/kaggle/input/titanic/test.csv")
test_data.head()
```

[7]:

	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
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Data Exploration

:

```
train_data.info()
```

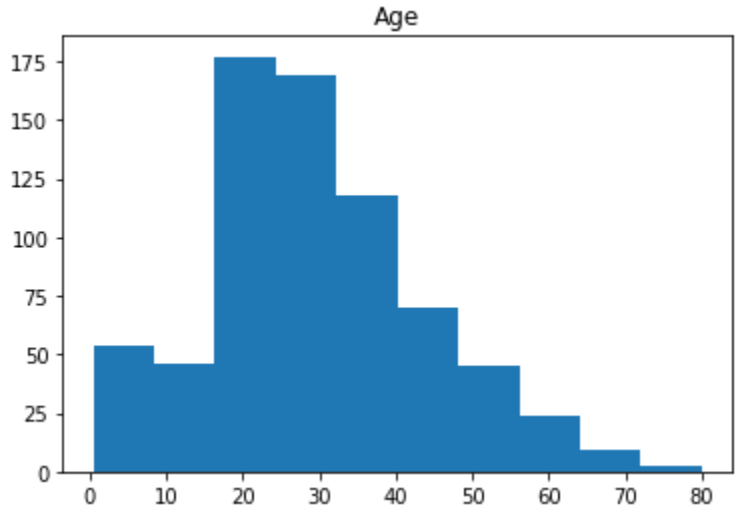
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    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        891 non-null    object 
 11  Embarked     889 non-null    object
```

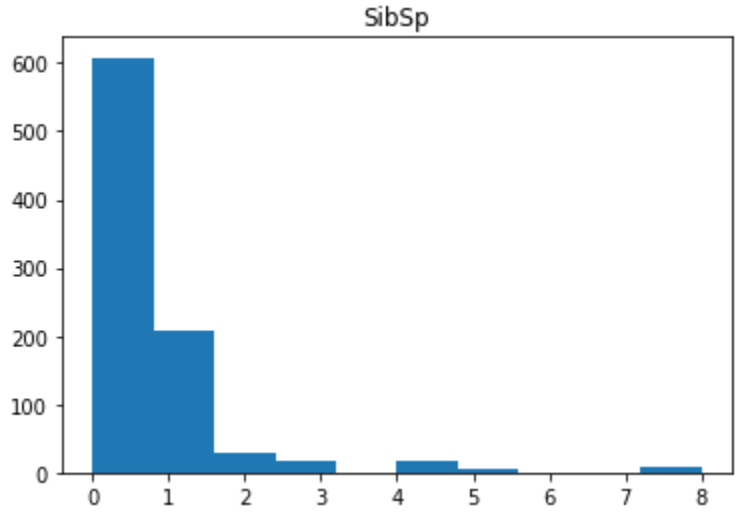
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB

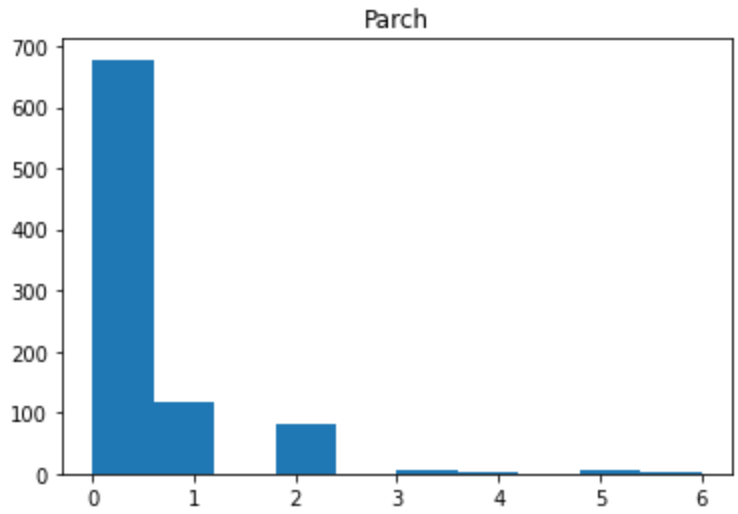
Here I'am separating the attributes to those having numerical value and those that does'nt

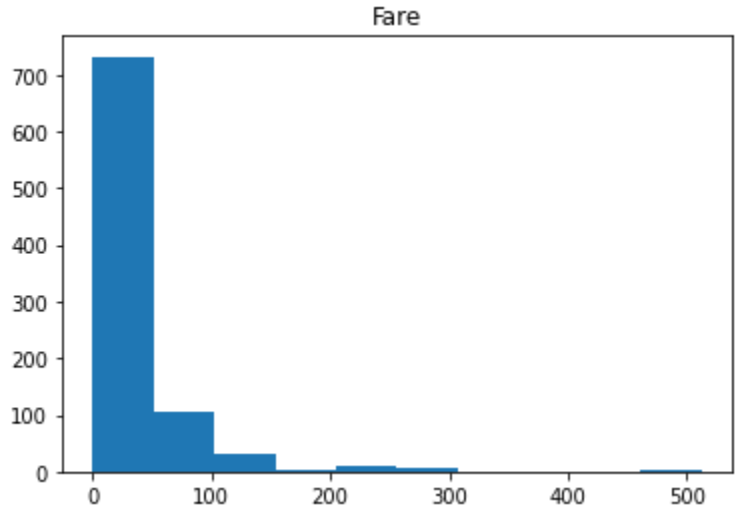
[31]:

```
#distributions for all numeric variables
for i in df_num.columns:
   plt.hist(df_num[i])
   plt.title(i)
   plt.show()
```









We find that only Age is normalized and the others Sibsp, Fare and Parch needs normalization.

```
#Finding out how the different attributes relate to survival
# compare survival rate across Age, SibSp, Parch, and Fare
pd.pivot_table(train_data, index = 'Survived', values =
['Age','SibSp','Parch','Fare'])
```

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

Data Preprocessing

[42]:

```
#Data set is concatenated for data preprocessing
all_data = pd.concat([train_data,test_data])

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

#The age value is substituted with its median, the fare value is also
substituted with median
all_data.Age = all_data.Age.fillna(training.Age.median())

all_data.Fare = all_data.Fare.fillna(training.Fare.median())
```

```
train_data.describe()
```

	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

```
print(train_data.isnull().sum())
```

```
PassengerId     0
Survived        0
Pclass          0
Name            0
Sex             0
Age           177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin         687
Embarked        2
dtype: int64
```

```
#We found that 687 of the Cabin values are null so we can replace it with a value 'unknown'
train_data.Cabin = train_data.Cabin.fillna('unknown')
print(train_data.isnull().sum())
```

```
PassengerId     0
Survived        0
Pclass          0
Name            0
```

```
Sex          0
Age        177
SibSp        0
Parch        0
Ticket       0
Fare         0
Cabin        0
Embarked     2
dtype: int64
```

Classification

```
#Used RandomForest Classifier
from sklearn.ensemble import RandomForestClassifier

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived':
predictions})
output.to_csv('submission.csv', index=False)
print("The predicted outputs are displayed below")
```

The predicted outputs are displayed below

[20]:

```
output.head(10)
```

[20]:

	PassengerId	Survived
0	892	0

1	893	1
2	894	0
3	895	0
4	896	1
5	897	0
6	898	1
7	899	0
8	900	1
9	901	0
