Titanic Data Analysis -- Salman AlMaskati

```
#imports
from tabulate import tabulate
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Data dictionary

```
table = [['Variable', 'Definition','Key'],
         ['Survival', 'Survived or not','0= NO 1=YES'],
         ['Pclass','Ticket Class', '1 = 1st, 2 = 2nd, 3 = 3rd'],
         ['sibSp','# of siblings / spouses aboard the Titanic'],
         ['Parch','# of parents / children aboard the Titanic '],
         ['Fare','Passenger fare'],
         ['Cabin','Cabin number'],
         ['Embarked','Port of Embarkation', 'C = Cherbourg, Q = Queenstown, S = Southampton']
        ]

print(tabulate(table, headers='firstrow', tablefmt='fancy_grid'))
```

Variable	Definition	Key
Survival	Survived or not	0= NO 1=YES
Pclass	Ticket Class	1 = 1st, 2 = 2nd, 3 = 3rd
sibSp	# of siblings / spouses aboard the Titanic	
Parch	# of parents / children aboard the Titanic	
Fare	Passenger fare	
Cabin	Cabin number	
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Exploratory Data Analysis

```
#read datasets
url="https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv" #raw URL
df= pd.read_csv(url)
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
```

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803

```
#basic stats

countF = (df['Sex'] == 'female').sum()
countM = (df['Sex'] == 'male').sum()
countSurv = (df['Survived'] == 1).sum()
countClass1= (df['Pclass'] == 1).sum()
countClass2= (df['Pclass'] == 2).sum()
countClass3= (df['Pclass'] == 3).sum()

print(f'Number of women on board {countF}')
print(f'Number of men on board {countM}\n')
print(f'Number of people Survived {countSurv}')
print(f'Number of people died {len(df)-countSurv}\n')
print(f'Number of people in 1st class {countClass1}')
print(f'Number of people in 2nd class {countClass2}')
print(f'Number of people in 3rd class {countClass3}')
```

```
Number of women on board 314
Number of men on board 577

Number of people Survived 342
Number of people died 549

Number of people in 1st class 216
Number of people in 2nd class 184
Number of people in 3rd class 491
```

```
#Visualizing NA's
countSurv = (df['Survived'] == 1).sum()
sns.heatmap(df.isnull(),yticklabels=False,cbar=False)
plt.title("Visualizing NA's")
```

```
Text(0.5, 1.0, "Visualizing NA's")
```

```
numrows=len(df)
cabinNAs=df['Cabin'].isna().sum()
ageNAs= df['Age'].isna().sum()
print(f"The number of NA's in the Cabin coloumn is {cabinNAs} out of {numrows}")
print(f"The number of NA's in the Age coloumn is {ageNAs} out of {numrows}")
```

```
The number of NA's in the Cabin coloumn is 687 out of 891
The number of NA's in the Age coloumn is 177 out of 891
```

Since most of the cabin information is missing we can go ahead and remove it from the df. Additonally, we can also drop the name and ticket column since it adds no value to the df

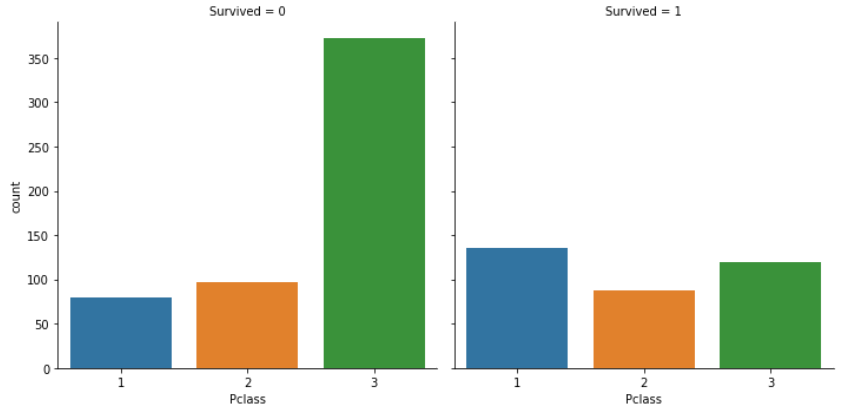
```
df1=df.drop(['Ticket','Cabin'   ,'Name'],axis=1)
df1
```

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

891 rows × 9 columns

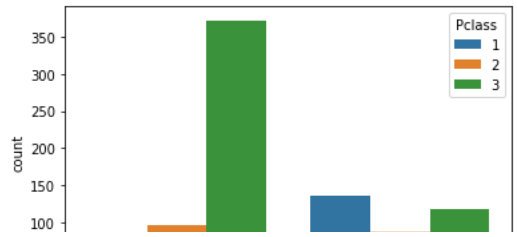
```
sns.factorplot(x="Pclass", col="Survived", kind='count',data=df1)
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3717: UserWarning: T
  warnings.warn(msg)
<seaborn.axisgrid.FacetGrid at 0x7f08db602250>
```




```
sns.countplot(x='Survived',hue='Pclass',data=df1)
```

```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

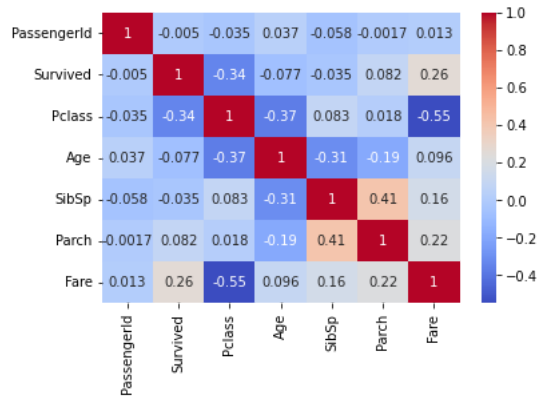



```
df1.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000

```
sns.heatmap(df1.corr(), annot=True, cmap='coolwarm')
```

```
<AxesSubplot:>
```




```
#get dummies
dummy_sex= pd.get_dummies(df1['Sex'])
dummy_embarked= pd.get_dummies(df1['Embarked'])
df1.drop(['Sex', 'Embarked'],axis=1,inplace=True)
df1= pd.concat([df1,dummy_sex,dummy_embarked],axis=1)
```

```
df1
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	female	male	C	Q
0	1	0	3	22.0	1	0	7.2500	0	1	0	0
2	3	1	3	26.0	0	0	7.9250	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...

One hot encoded the non numeric columns (Emabrked, Sex) using pandas function pd.get_dummies

To fill in the age coloumn, i have taken the mean age of every class and replaced it with the NA's in the Age column

```
age_pclass1=df.loc[df['Pclass'] == 1, 'Age'].mean().astype('int')
age_pclass2=df.loc[df['Pclass'] == 2, 'Age'].mean().astype('int')
age_pclass3=df.loc[df['Pclass'] == 3, 'Age'].mean().astype('int')

print(f'Mean age for 1st class is {age_pclass1}')
print(f'Mean age for 2st class is {age_pclass2}')
print(f'Mean age for 3st class is {age_pclass3}')
```

```
Mean age for 1st class is 38
Mean age for 2st class is 29
Mean age for 3st class is 25
```

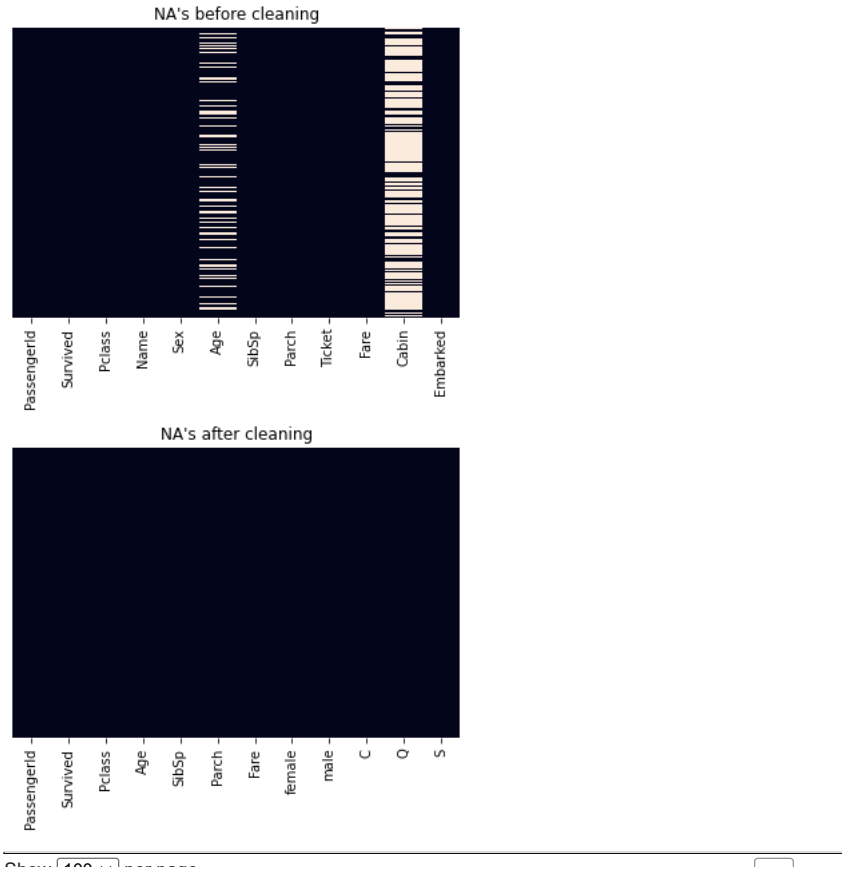
```
df2 = df1.copy()
df2.loc[(df2['Pclass'] == 1) & (df2['Age'].isna()), 'Age'] = age_pclass1
df2.loc[(df2['Pclass'] == 2) & (df2['Age'].isna()), 'Age'] = age_pclass2
df2.loc[(df2['Pclass'] == 3) & (df2['Age'].isna()), 'Age'] = age_pclass3
df2
```

1 to 100 of 153 entries (filtered from 891 total entries) Filter

index	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	female	male	C	Q	S
1	2	1	1	38.0	1	0	71.2833	1	0	1	0	0
2	3	1	3	26.0	0	0	7.925	1	0	0	0	1
3	4	1	1	35.0	1	0	53.1	1	0	0	0	1
4	5	0	3	35.0	0	0	8.05	0	1	0	0	1
5	6	0	3	25.0	0	0	8.4583	0	1	0	1	0
6	7	0	1	54.0	0	0	51.8625	0	1	0	0	1
7	8	0	3	2.0	3	1	21.075	0	1	0	0	1
8	9	1	3	27.0	0	2	11.1333	1	0	0	0	1
9	10	1	2	14.0	1	0	30.0708	1	0	1	0	0
10	11	1	3	4.0	1	1	16.7	1	0	0	0	1
11	12	1	1	58.0	0	0	26.55	1	0	0	0	1
12	13	0	3	20.0	0	0	8.05	0	1	0	0	1
13	14	0	3	39.0	1	5	31.275	0	1	0	0	1
14	15	0	3	14.0	0	0	7.8542	1	0	0	0	1
15	16	1	2	55.0	0	0	16.0	1	0	0	0	1
16	17	0	3	2.0	4	1	29.125	0	1	0	1	0
17	18	1	2	29.0	0	0	13.0	0	1	0	0	1
18	19	0	3	31.0	1	0	18.0	1	0	0	0	1
19	20	1	3	25.0	0	0	7.225	1	0	1	0	0
20	21	0	2	35.0	0	0	26.0	0	1	0	0	1
21	22	1	2	34.0	0	0	13.0	0	1	0	0	1
22	23	1	3	15.0	0	0	8.0292	1	0	0	1	0
23	24	1	1	28.0	0	0	35.5	0	1	0	0	1
24	25	0	3	8.0	3	1	21.075	1	0	0	0	1
25	26	1	3	38.0	1	5	31.3875	1	0	0	0	1
26	27	0	3	25.0	0	0	7.225	0	1	1	0	0
27	28	0	1	19.0	3	2	263.0	0	1	0	0	1
28	29	1	3	25.0	0	0	7.8792	1	0	0	1	0
29	30	0	3	25.0	0	0	7.8958	0	1	0	0	1
30	31	0	1	40.0	0	0	27.7208	0	1	1	0	0
31	32	1	1	38.0	1	0	146.5208	1	0	1	0	0
32	33	1	3	25.0	0	0	7.75	1	0	0	1	0
33	34	0	2	66.0	0	0	10.5	0	1	0	0	1
34	35	0	1	28.0	1	0	82.1708	0	1	1	0	0
35	36	0	1	42.0	1	0	52.0	0	1	0	0	1
36	37	1	3	25.0	0	0	7.2292	0	1	1	0	0
37	38	0	3	21.0	0	0	8.05	0	1	0	0	1
38	39	0	3	18.0	2	0	18.0	1	0	0	0	1
39	40	1	3	14.0	1	0	11.2417	1	0	1	0	0
40	41	0	3	40.0	1	0	9.475	1	0	0	0	1
41	42	0	2	27.0	1	0	21.0	1	0	0	0	1
42	43	0	3	25.0	0	0	7.8958	0	1	1	0	0
43	44	1	2	3.0	1	2	41.5792	1	0	1	0	0
44	45	1	3	19.0	0	0	7.8792	1	0	0	1	0
45	46	0	3	25.0	0	0	8.05	0	1	0	0	1
46	47	0	3	25.0	1	0	15.5	0	1	0	1	0
47	48	1	3	25.0	0	0	7.75	1	0	0	1	0
48	49	0	3	25.0	2	0	21.6792	0	1	1	0	0
49	50	0	3	18.0	1	0	17.8	1	0	0	0	1
50	51	0	3	7.0	4	1	39.6875	0	1	0	0	1
51	52	0	3	21.0	0	0	7.8	0	1	0	0	1
52	53	1	1	49.0	1	0	76.7292	1	0	1	0	0
53	54	1	2	29.0	1	0	26.0	1	0	0	0	1
54	55	0	1	65.0	0	1	61.9792	0	1	1	0	0
55	56	1	1	38.0	0	0	35.5	0	1	0	0	1
56	57	1	2	21.0	0	0	10.5	1	0	0	0	1
57	58	0	3	28.5	0	0	7.2292	0	1	1	0	0
58	59	1	2	5.0	1	2	27.75	1	0	0	0	1
59	60	0	3	11.0	5	2	46.9	0	1	0	0	1
60	61	0	3	22.0	0	0	7.2292	0	1	1	0	0
61	62	1	1	38.0	0	0	80.0	1	0	0	0	0
62	63	0	1	45.0	1	0	83.475	0	1	0	0	1

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False)
plt.title("NA's before cleaning")
plt.show()
```

```
sns.heatmap(df2.isnull(),yticklabels=False,cbar=False)
plt.title("NA's after cleaning")
plt.show()
```



Double-click (or enter) to edit

Predicitve Analytics

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
feat=['PassengerId','Pclass','Age','Parch','Fare','female','male','C','Q','S']
X=df2[feat]
y=df2['Survived']
y.head()
```

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=10)
```

```
y_test.to_frame()
```

	Survived
590	0
131	0
628	0
195	1
230	1
...	...
456	0
191	0
603	0
94	0

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)

svm_acc_train=svm.score(X_train, y_train).round(2)
print(f'Accuracy of SVM classifier on training set: {svm_acc_train}')

svm_acc_test=svm.score(X_test, y_test).round(2)
print(f'Accuracy of SVM classifier on test set: {svm_acc_test}')
```

```
Accuracy of SVM classifier on training set: 0.63
Accuracy of SVM classifier on test set: 0.69
```

```
#K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
knn_acc_train= knn.score(X_train, y_train).round(2)
knn_acc_test= knn.score(X_test, y_test).round(2)

print(f'Accuracy of K-NN classifier k=5 on train set: {knn_acc_train}')
print(f'Accuracy of K-NN classifier k=5 on test set: {knn_acc_test}')
```

```
Accuracy of K-NN classifier k=5 on train set: 0.73
Accuracy of K-NN classifier k=5 on test set: 0.71
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.tree import export_text


clf = DecisionTreeClassifier().fit(X_train,y_train)

dt_acc_train= clf.score(X_train,y_train).round(2)
print(f'Accuracy of Decision Tree classifier on training set: {dt_acc_train}')

dt_acc_test= clf.score(X_test,y_test).round(2)
print(f'Accuracy of Decision Tree classifier on test set: {dt_acc_test}')


r = export_text(clf, feature_names=list(X_train.columns))
#print(r)
```

```
Accuracy of Decision Tree classifier on training set: 1.0
Accuracy of Decision Tree classifier on test set: 0.83
```

```
table = [['Model', 'Training Score','Test Score'],
         ['KNN', knn_acc_train ,knn_acc_test],
         ['Decison Tree',dt_acc_train, dt_acc_test],
         ['SVM',svm_acc_train,svm_acc_test],

        ]
```

```
print(tabulate(table, headers='firstrow', tablefmt='fancy_grid'))
```

```
╒═══════════════╤══════════════════╤════════════╕
│ Model         │   Training Score │ Test Score │
╞═══════════════╪══════════════════╪════════════╡
│ KNN           │             0.73 │       0.71 │
├───────────────┼──────────────────┼────────────┤
│ Decison Tree  │             1    │       0.83 │
├───────────────┼──────────────────┼────────────┤
│ SVM           │             0.63 │       0.69 │
╘═══════════════╧══════════════════╧════════════╛
```

```
y_pred = clf.predict(X_test)
```

```
len(y_pred)
```

```
179
```

```
X_test.shape
```

```
(179, 10)
```

```
final_df= pd.DataFrame(X_test['PassengerId'])
final_df['Survived Predictions'] = y_pred
final_df
```

	PassengerId	Survived Predictions
590	591	0
131	132	0
628	629	0
195	196	1
230	231	1
...	...	...
456	457	0
191	192	1
603	604	0
94	95	0
766	767	0

179 rows × 2 columns

Colab paid products - Cancel contracts here

✓ 0s completed at 4:22 AM