# Assignment 3

Name: Rohitha Sai Alla

No: 700734780

Video link:
https://drive.google.com/file/d/1p2RDNQiEMmu_XbQmDnjmL3UZ49Q6VNID/view?usp=sharing

Git link: https://github.com/RohithaSaiML/Assignment3

1.Reading the test and train data set and removing nulls

In [3]:
```
#1.reading the csv file
import pandas as pd

train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

#to drop null values in test and train
train.dropna(inplace = True)
train.isna().sum()

test.dropna(inplace = True)
test.isna().sum()
```

Out[3]:
```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

**1.1** Correlation between survived and sex:

In [2]:
```
#1.1 to find correlation between sex and survived
pd.crosstab(train.Sex, train.Survived)
```

Output:

Out[2]:

| Survived | 0 | 1 |
|---|---|---|
| Sex | | |
| female | 81 | 233 |
| male | 468 | 109 |

Than males more females have survived. This is one of the important information so we should have this feature.

**1.2** Do at least two visualizations to describe or show correlations.

We convert all non-numeric fields to numeric in order to do correlation.

```
# to convert sex from alphabetical value to numerical
from sklearn.preprocessing import LabelEncoder

le_Sex = LabelEncoder().fit(train.Sex.unique())
classes_Sex = le_Sex.classes_
train['Sex'] = le_Sex.transform(train.Sex.values)
```

```
#to convert cabin from alphabetical to numeric
le_cabin = LabelEncoder().fit(train.Cabin.unique())
classes_cabin = le_cabin.classes_
train['Cabin'] = le_cabin.transform(train.Cabin.values)
```

```
#to convert embarked from alphabetical to numeric
le_Embarked = LabelEncoder().fit(train.Embarked.unique())
classes_Embarked = le_Embarked.classes_
train['Embarked'] = le_Embarked.transform(train.Embarked.values)
```

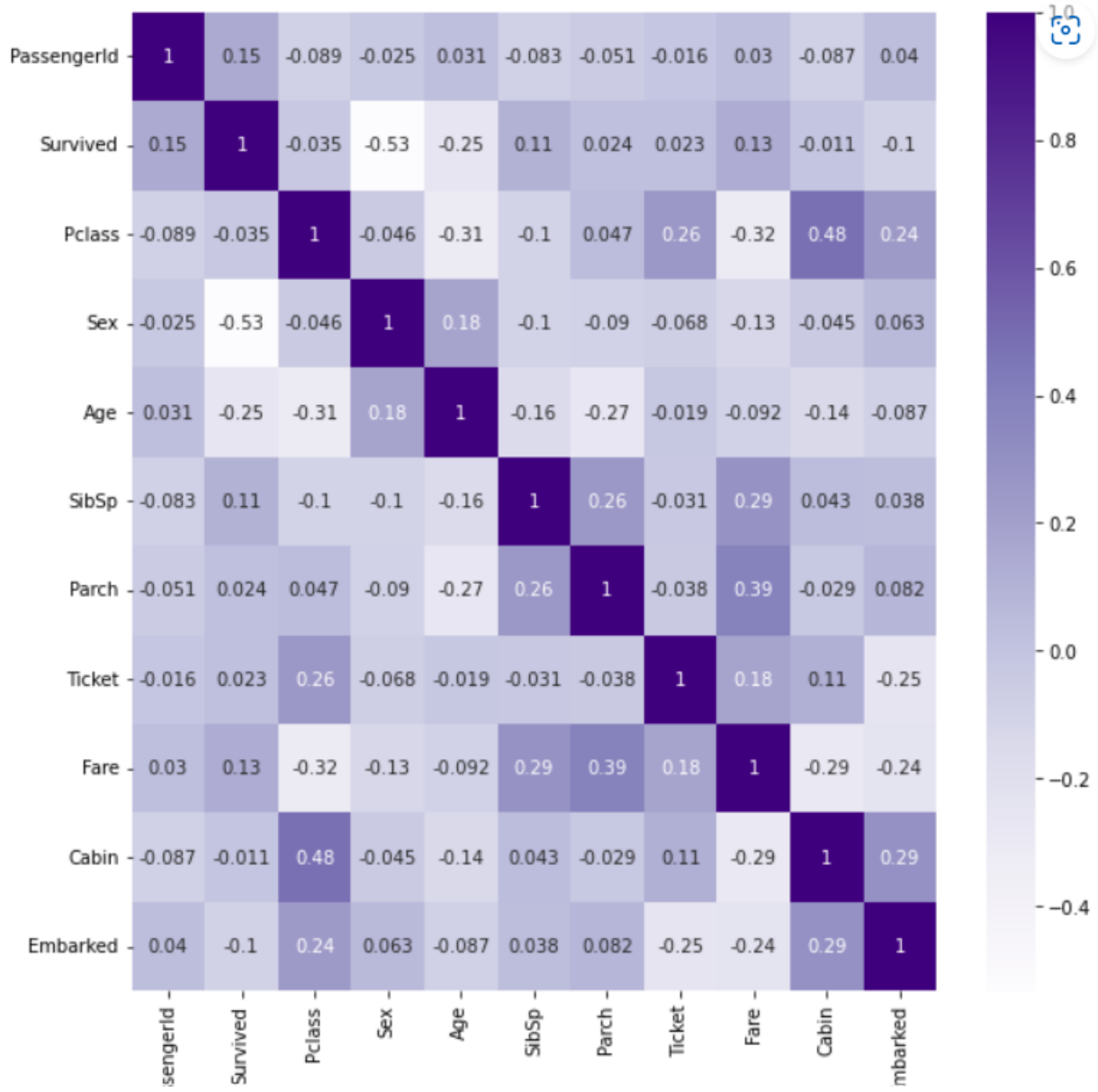
```
#to convert ticket value from alphabetical to numeric
le_Ticket = LabelEncoder().fit(train.Ticket.unique())
classes_Ticket = le_Ticket.classes_
train['Ticket'] = le_Ticket.transform(train.Ticket.values)
```

First visualization

```
#to find all the correlations
corr = train.corr()

#1.2 First visualization
#to plot the correlations
plt.figure(figsize = (10, 10))
sns.heatmap(corr, annot = True, cmap = 'Purples')
plt.show()
```

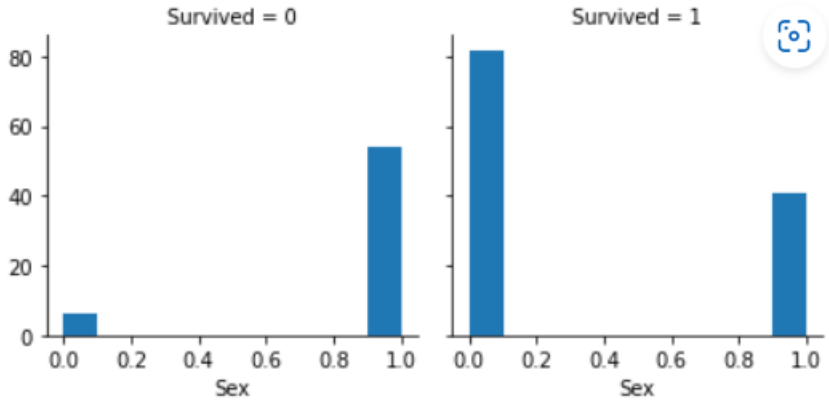
Output:




Second visualization:

```
In [64]: #1.2 Second visualization
         g = sns.FacetGrid(train, col='Survived')
         g.map(plt.hist, 'Sex', bins=10)
```

Output:




**1.3** Naïve Bayes

To split data into feature and target

```
# split into X and Y
X = train.drop(columns = ['Survived','Name']).values
y = train.Survived.values

print(X.shape)
print(y.shape)
```

To split data using train_test_split

```
#to split data into train and test dataset(75-25)
from sklearn.model_selection import train_test_split as tts

X_train, X_test, y_train, y_test = tts(X, y, test_size = .25, random_state = 0, stratify = y)
```

Implementing Naïve Bayes

```
In [74]: #1.3 Naive Bayes
         from sklearn.naive_bayes import GaussianNB
         nb = GaussianNB()
         nb.fit(X_train, y_train)
         nb_pred = nb.predict(X_test)
```

From sklearn.metrics we can import confusion matrix and classification report

```
# evaluating the classification reports and confusion matrices of each classifier
from sklearn.metrics import classification_report, confusion_matrix


# Naive Bayes
print("Naive Bayes Confusion Matrix: \n", confusion_matrix(y_test, nb_pred))
print("Classification Report:\n", classification_report(y_test, nb_pred))
print("Naive bayes is 72% accurate")
```

Naïve bayes is 72% accurate.

Output:

```
Naive Bayes Confusion Matrix:
 [[10  5]
 [ 8 23]]
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.67      0.61        15
           1       0.82      0.74      0.78        31

    accuracy                           0.72        46
   macro avg       0.69      0.70      0.69        46
weighted avg       0.73      0.72      0.72        46

Naive bayes is 72% accurate
```

**2.Using glass dataset**

2.1 To implement Naïve Bayes method using scikit learn

Using train_test_split method to get training and testing data.

In [101]:
```
# to split data into 75 train and 25 test
from sklearn.model_selection import train_test_split as tts

X_train, X_test, y_train, y_test = tts(X, y, test_size = .25, random_state = 0, stratify = y)
```

In [117]:
```
#2.1 Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)
```

2.2 Evaluating the model on testing part using score and classification_report(y_true, y_pred)

In [112]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Naive Bayes
print("Naive Bayes Confusion Matrix:\n", confusion_matrix(y_test, nb_pred))

#2.2 Classification report on testing part using score on Naive bayes
print("Classification Report for Naive Bayes:\n", classification_report(y_test, nb_pred))
print()
```

Output:

```
Naive Bayes Confusion Matrix:
 [[8 4 5 0 0 0]
 [5 9 0 3 0 0]
 [2 1 1 0 0 0]
 [0 1 0 1 0 0]
 [0 0 0 0 2 0]
 [0 0 0 0 0 7]]
Classification Report for Naive Bayes:
              precision    recall  f1-score   support

           1       0.53      0.47      0.50        17
           2       0.60      0.53      0.56        17
           3       0.17      0.25      0.20         4
           5       0.25      0.50      0.33         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         7

    accuracy                           0.57        49
   macro avg       0.59      0.62      0.60        49
weighted avg       0.60      0.57      0.58        49
```

2.1 To implement SVM using scikit learn

```
#2.1 Support Vector Machine
from sklearn.svm import SVC
svm = SVC(kernel = 'rbf')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
```

2.2 Evaluating the model on testing part using score and classification_report(y_true, y_pred)

```
# Support Vector Machine
print("Support Vector Machine Confusion Matrix:\n", confusion_matrix(y_test, svm_pred))

#2.2 Classification report on testing part using score on SVM
print("Classification Report for SVM:\n", classification_report(y_test, svm_pred))
print()
```

Output:

```
Support Vector Machine Confusion Matrix:
 [[13  4  0  0  0  0]
 [ 4 13  0  0  0  0]
 [ 2  2  0  0  0  0]
 [ 0  1  0  1  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  7]]
Classification Report for SVM:
              precision    recall  f1-score   support

           1       0.68      0.76      0.72        17
           2       0.65      0.76      0.70        17
           3       0.00      0.00      0.00         4
           5       1.00      0.50      0.67         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         7

    accuracy                           0.73        49
   macro avg       0.72      0.67      0.68        49
weighted avg       0.69      0.73      0.71        49
```

Accuracy for SVM:

```
In [118]: #Accuracy is better in SVM than Naive bayes
          svm = accuracy_score(y_test,svm_pred)
          print('Accuracy for svm: ',svm)
```

Output:

```
Accuracy for svm:  0.7346938775510204
```

Accuracy for Naïve Bayes:

```
In [119]: nb = accuracy_score(y_test,nb_pred)
          print('Accuracy in Naive Bayes: ',nb)
```

Output:

```
Accuracy in Naive Bayes:  0.5714285714285714
```

**SVM's accuracy is best than Naïve Bayes.**

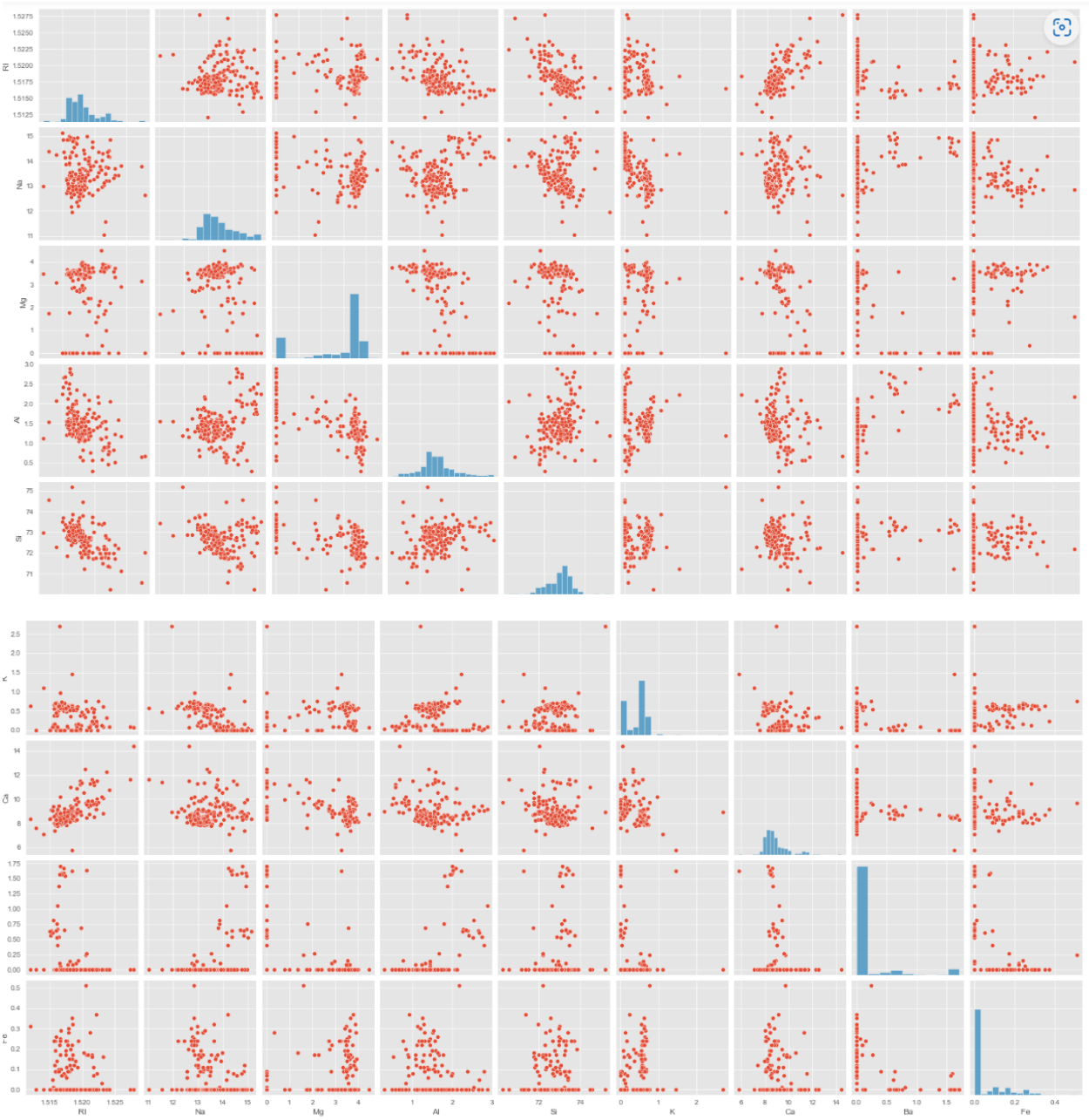
SVM performs better because it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes.

Visualizations to describe or show correlations in glass dataset.

First visualization for glass dataset

```
In [128]: #First visualization to show correlation in glass dataset
          plt.figure(figsize=(15,7))
          sn.pairplot(df[features])
```

Output:




Ri and Ca have the most positive relationship.

Now we will check the correlation between each item.

Second visualization

```
In [129]: corr  = df[features].corr()
          plt.figure(figsize=(15,8))
          sn.heatmap(corr,annot=True,vmax=0.9, square=True)
```

Output:


With a cut off 0.9 there is no multi collinearity.