Machine Learning

Assignment 3

Laxma Reddy Nalla

700732071

Lxn20710

YouTube Video Link: https://youtu.be/Klh-Yg5_6w0

GitHub Link: CS-5710/Assigment-3 at dev · LaxmaReddy-Nalla/CS-5710 (github.com)

Question 1:

1. (Titanic Dataset)

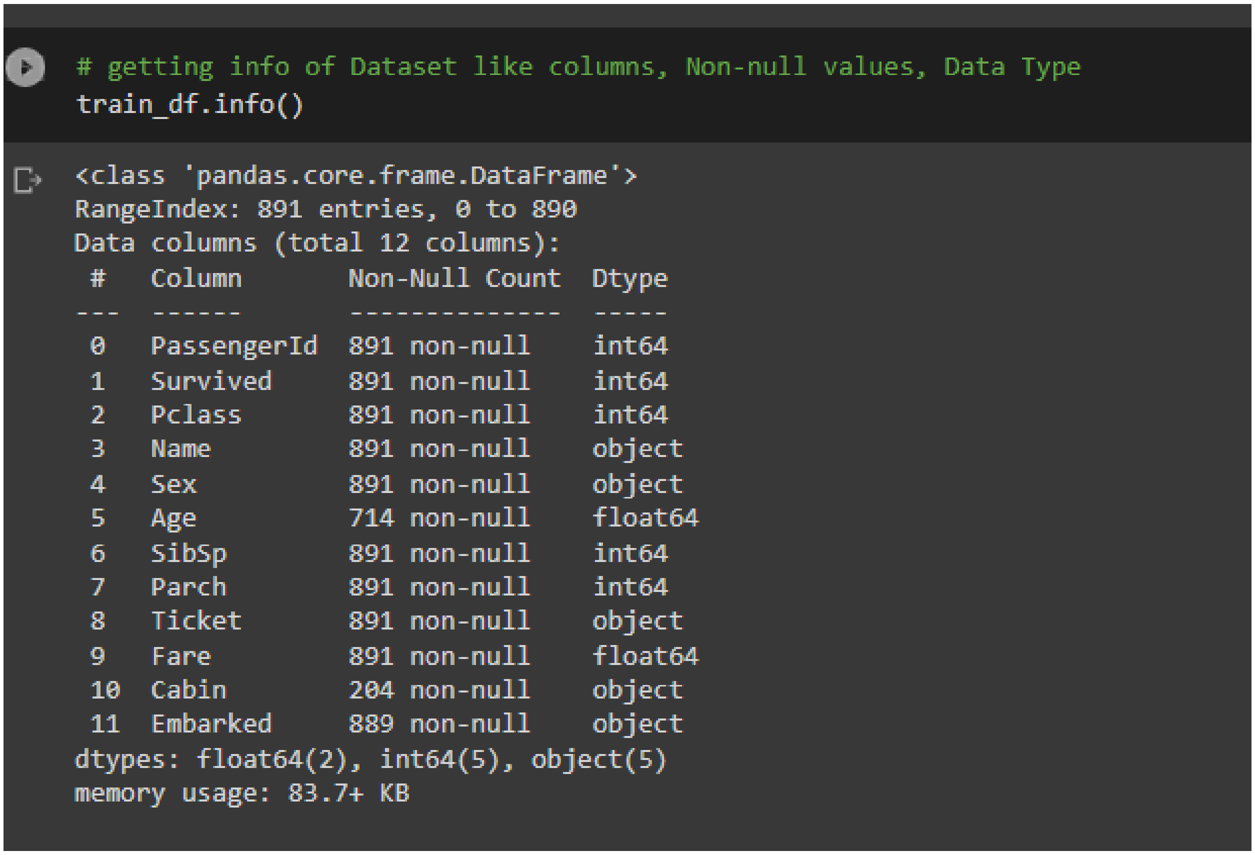
1. Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case in class.

   a. Do you think we should keep this feature?

2. Do at least two visualizations to describe or show correlations.

3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

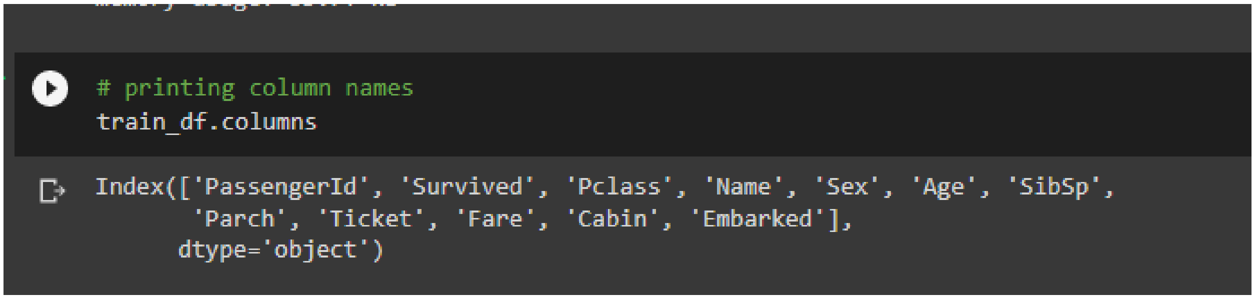
- Imported the titanic dataset and printed info about the dataset using info() method



```
# getting info of Dataset like columns, Non-null values, Data Type
train_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
```

Here by the result of info() method, we can get to know that there are total of 891 columns and Age, Embarked and Cabin columns has null values.

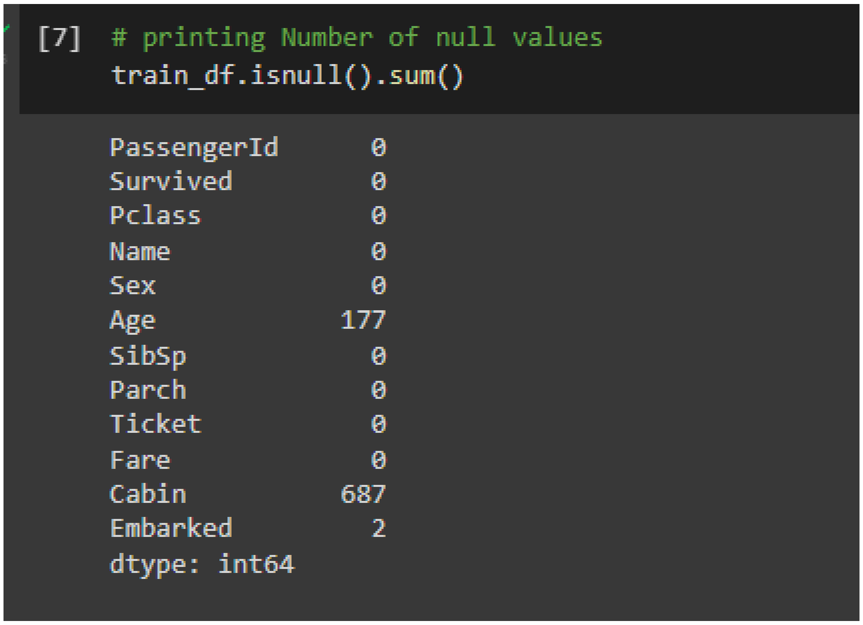


```
# printing column names
train_df.columns
```

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

These are the columns in the titanic dataset

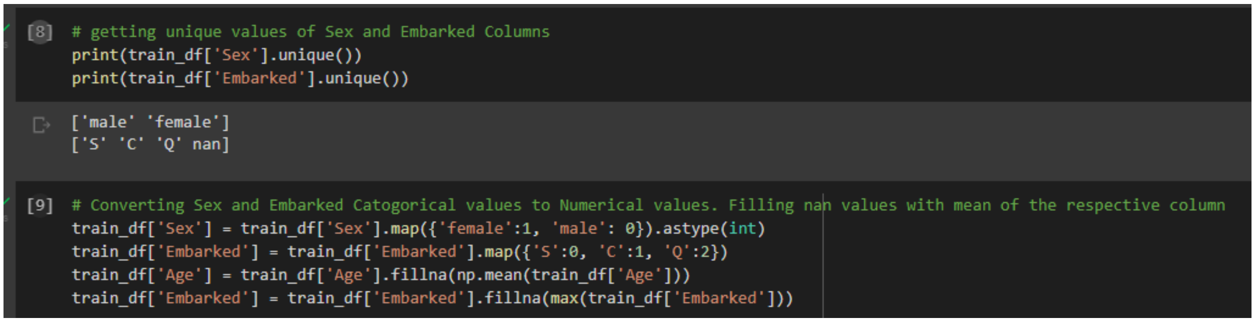
- Null values inside the dataset



```
[7]  # printing Number of null values
     train_df.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
```

- Transforming Categorical values of Sex and Embarked columns into Numerical values using map function.
- Filling null values with respective columns mean values.

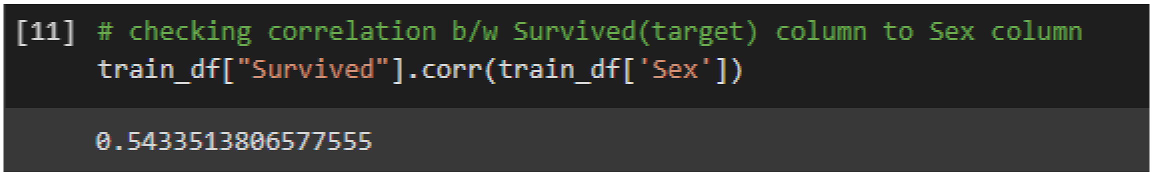


```
[8]  # getting unique values of Sex and Embarked Columns
     print(train_df['Sex'].unique())
     print(train_df['Embarked'].unique())
```

```
['male' 'female']
['S' 'C' 'Q' nan]
```

```
[9]  # Converting Sex and Embarked Catogorical values to Numerical values. Filling nan values with mean of the respective column
     train_df['Sex'] = train_df['Sex'].map({'female':1, 'male': 0}).astype(int)
     train_df['Embarked'] = train_df['Embarked'].map({'S':0, 'C':1, 'Q':2})
     train_df['Age'] = train_df['Age'].fillna(np.mean(train_df['Age']))
     train_df['Embarked'] = train_df['Embarked'].fillna(max(train_df['Embarked']))
```

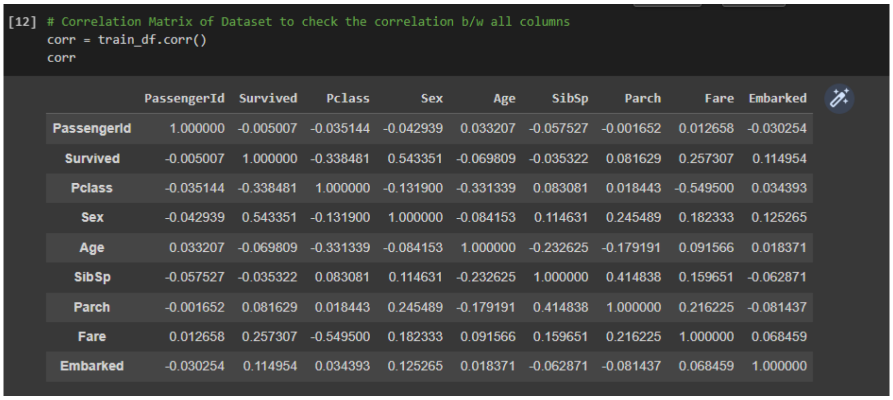
- Getting Correlation between the Survived and Sex columns.



```
[11] # checking correlation b/w Survived(target) column to Sex column
     train_df["Survived"].corr(train_df['Sex'])

     0.5433513806577555
```

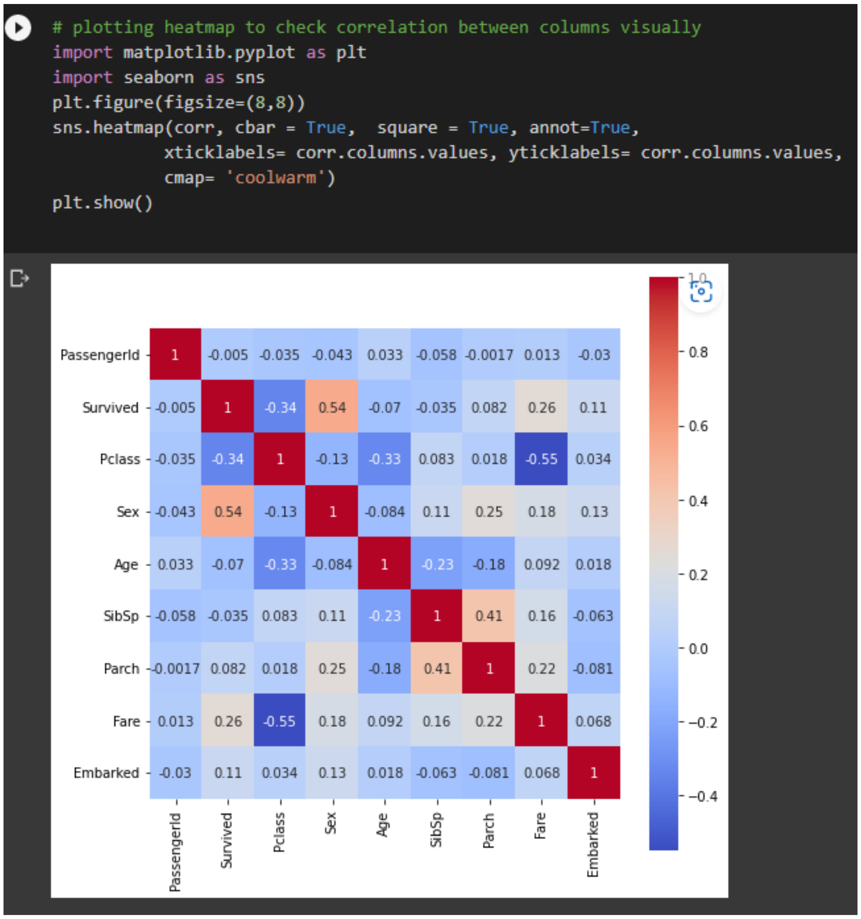
- Printing correlation matrix which is helpful to get more insight into relation between each column pair



```
[12] # Correlation Matrix of Dataset to check the correlation b/w all columns
     corr = train_df.corr()
     corr
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId	1.000000	-0.005007	-0.035144	-0.042939	0.033207	-0.057527	-0.001652	0.012658	-0.030254
Survived	-0.005007	1.000000	-0.338481	0.543351	-0.069809	-0.035322	0.081629	0.257307	0.114954
Pclass	-0.035144	-0.338481	1.000000	-0.131900	-0.331339	0.083081	0.018443	-0.549500	0.034393
Sex	-0.042939	0.543351	-0.131900	1.000000	-0.084153	0.114631	0.245489	0.182333	0.125265
Age	0.033207	-0.069809	-0.331339	-0.084153	1.000000	-0.232625	-0.179191	0.091566	0.018371
SibSp	-0.057527	-0.035322	0.083081	0.114631	-0.232625	1.000000	0.414838	0.159651	-0.062871
Parch	-0.001652	0.081629	0.018443	0.245489	-0.179191	0.414838	1.000000	0.216225	-0.081437
Fare	0.012658	0.257307	-0.549500	0.182333	0.091566	0.159651	0.216225	1.000000	0.068459
Embarked	-0.030254	0.114954	0.034393	0.125265	0.018371	-0.062871	-0.081437	0.068459	1.000000

- Plotting heatmap of correlation matrix which will help to visually seek into correlation of columns



```
# plotting heatmap to check correlation between columns visually
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,8))
sns.heatmap(corr, cbar = True,  square = True, annot=True,
            xticklabels= corr.columns.values, yticklabels= corr.columns.values,
            cmap= 'coolwarm')
plt.show()
```

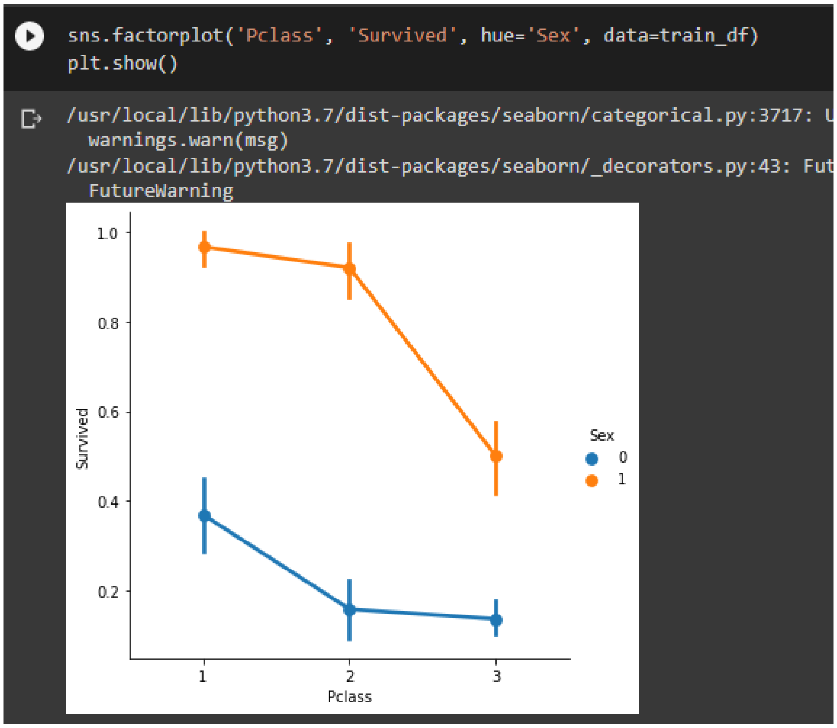
PassengerId 1 -0.005 -0.035 -0.043 0.033 -0.058 -0.0017 0.013 -0.03
Survived -0.005 1 -0.34 0.54 -0.07 -0.035 0.082 0.26 0.11
Pclass -0.035 -0.34 1 -0.13 -0.33 0.083 0.018 -0.55 0.034
Sex -0.043 0.54 -0.13 1 -0.084 0.11 0.25 0.18 0.13
Age 0.033 -0.07 -0.33 -0.084 1 -0.23 -0.18 0.092 0.018
SibSp -0.058 -0.035 0.083 0.11 -0.23 1 0.41 0.16 -0.063
Parch -0.0017 0.082 0.018 0.25 -0.18 0.41 1 0.22 -0.081
Fare 0.013 0.26 -0.55 0.18 0.092 0.16 0.22 1 0.068
Embarked -0.03 0.11 0.034 0.13 0.018 -0.063 -0.081 0.068 1
PassengerId
Survived
Pclass
Sex
Age
SibSp
Parch
Fare
Embarked
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

- Getting visual correlation between Survival, Plclass columns with respective to sex column.
- Factor plot is used to draw categorical plot

Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case in class.

Do you think we should keep this feature?

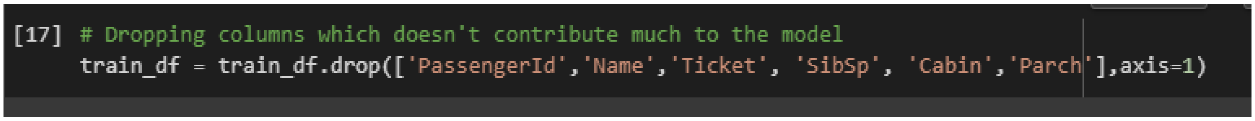
- I believe that column sex came be kept as is by looking into the heatmap and correlation matrix the column is not much correlated with other columns. And we are predicting discrete values weather the passenger survived or died since it is not a classification problem Pearson correlation doesn't contribute much for the model training and accuracy.



```
sns.factorplot('Pclass', 'Survived', hue='Sex', data=train_df)
plt.show()
```

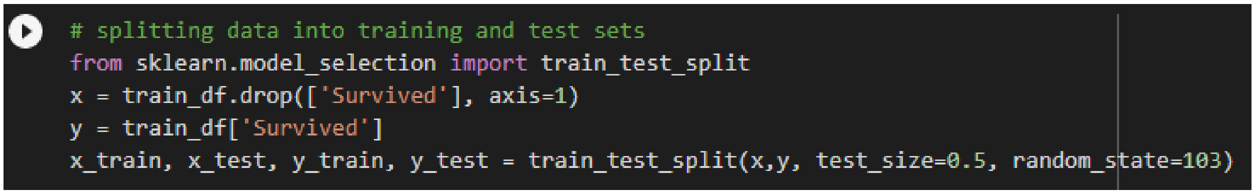
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:3717: U
  warnings.warn(msg)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: Fut
  FutureWarning
```

- Removed columns which are not much contribution to the model



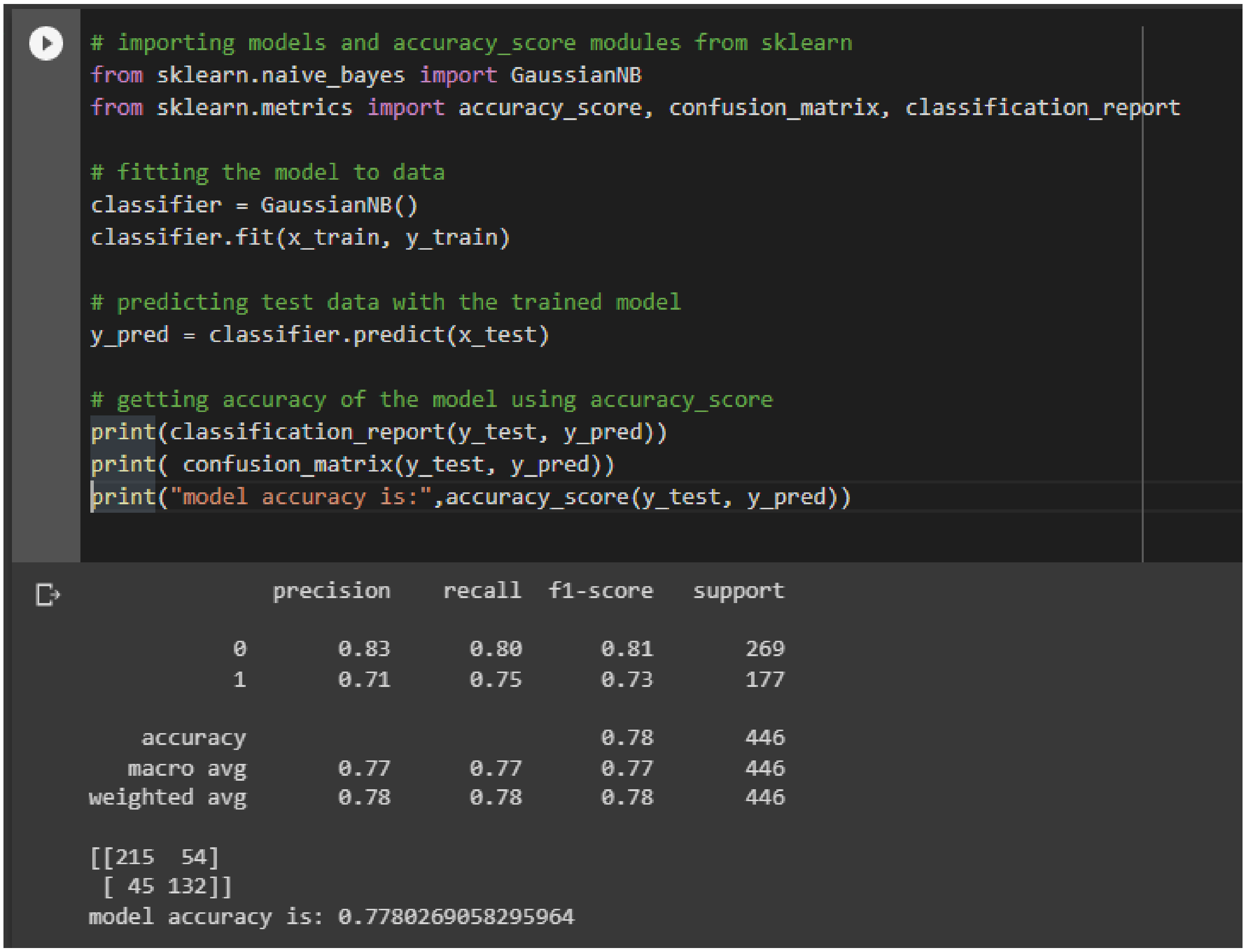
```
[17] # Dropping columns which doesn't contribute much to the model
     train_df = train_df.drop(['PassengerId','Name','Ticket', 'SibSp', 'Cabin','Parch'],axis=1)
```

- Splitting the Data into Train test Split datasets.



```
# splitting data into training and test sets
from sklearn.model_selection import train_test_split
x = train_df.drop(['Survived'], axis=1)
y = train_df['Survived']
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5, random_state=103)
```

- Fitting the Gaussian Naive Bayes model to the dataset
- Printing Classification report and accuracy of the model after predicting values using test dataset
- The model accuracy is 77% or 0.77



```
# importing models and accuracy_score modules from sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# fitting the model to data
classifier = GaussianNB()
classifier.fit(x_train, y_train)

# predicting test data with the trained model
y_pred = classifier.predict(x_test)

# getting accuracy of the model using accuracy_score
print(classification_report(y_test, y_pred))
print( confusion_matrix(y_test, y_pred))
print("model accuracy is:",accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.80      0.81       269
           1       0.71      0.75      0.73       177

    accuracy                           0.78       446
   macro avg       0.77      0.77      0.77       446
weighted avg       0.78      0.78      0.78       446

[[215  54]
 [ 45 132]]
model accuracy is: 0.7780269058295964
```

Question 2:

(Glass Dataset)

1. Implement Naïve Bayes method using scikit-learn library.

   a. Use the glass dataset available in Link also provided in your assignment.

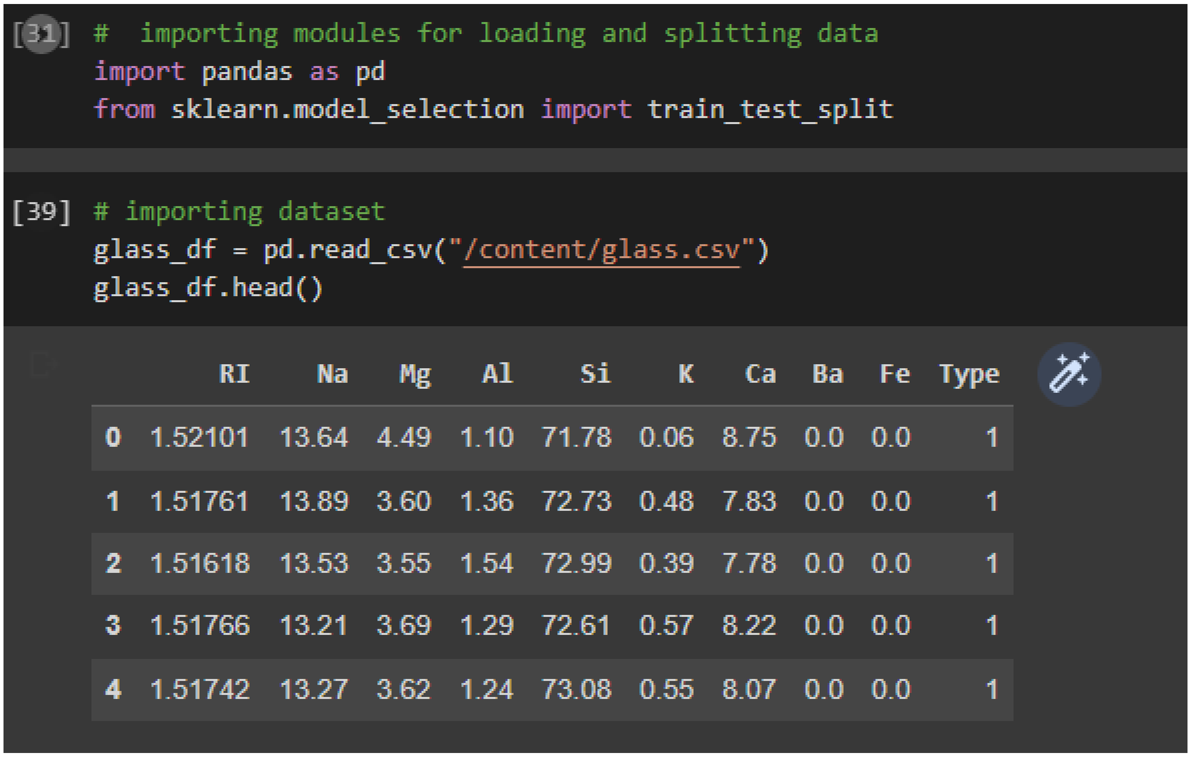
   b. Use train_test_split to create training and testing part.

2. Evaluate the model on testing part using score and classification_report(y_true, y_pred)

- Importing basic modules for data import

- import glass dataset

- print sample of dataset using head() method

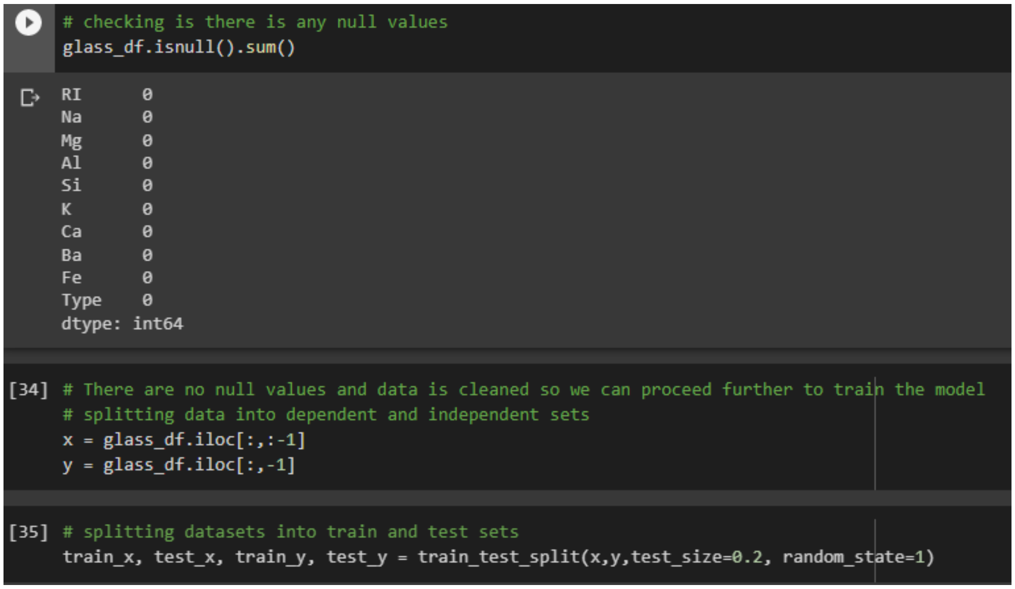


```
[31] #  importing modules for loading and splitting data
     import pandas as pd
     from sklearn.model_selection import train_test_split
```

```
[39] # importing dataset
     glass_df = pd.read_csv("/content/glass.csv")
     glass_df.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

- Checking null value count
- Partitioning data into independent and dependent variables
- Splitting data into train test datasets



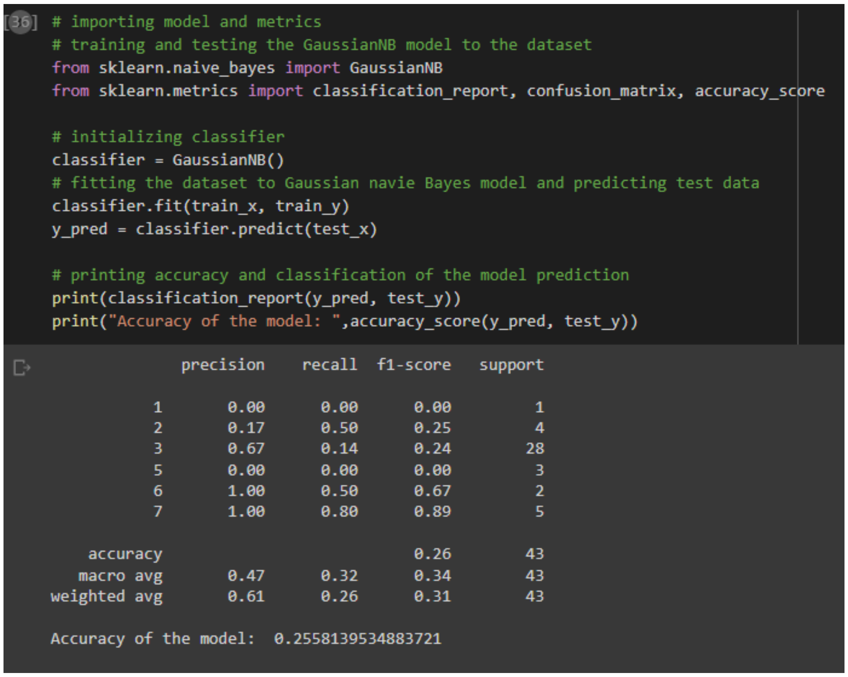
```
# checking is there is any null values
glass_df.isnull().sum()
```

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

```
[34] # There are no null values and data is cleaned so we can proceed further to train the model
     # splitting data into dependent and independent sets
     x = glass_df.iloc[:,:-1]
     y = glass_df.iloc[:,-1]
```

```
[35] # splitting datasets into train and test sets
     train_x, test_x, train_y, test_y = train_test_split(x,y,test_size=0.2, random_state=1)
```

- Training Gaussian Naïve Bayes model and reporting accuracy using accuracy_score and classification report
- Accuracy of the model: 25% or 0.25



```
[36] # importing model and metrics
     # training and testing the GaussianNB model to the dataset
     from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

     # initializing classifier
     classifier = GaussianNB()
     # fitting the dataset to Gaussian navie Bayes model and predicting test data
     classifier.fit(train_x, train_y)
     y_pred = classifier.predict(test_x)

     # printing accuracy and classification of the model prediction
     print(classification_report(y_pred, test_y))
     print("Accuracy of the model: ",accuracy_score(y_pred, test_y))
```

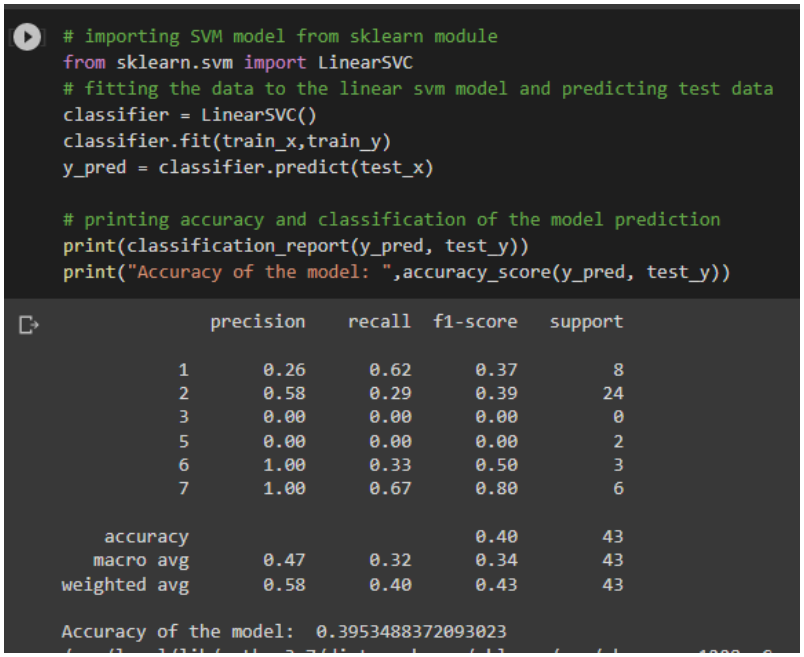
```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.17      0.50      0.25         4
           3       0.67      0.14      0.24        28
           5       0.00      0.00      0.00         3
           6       1.00      0.50      0.67         2
           7       1.00      0.80      0.89         5

    accuracy                           0.26        43
   macro avg       0.47      0.32      0.34        43
weighted avg       0.61      0.26      0.31        43

Accuracy of the model:  0.2558139534883721
```

- Training the Linear SVM model and reporting results using accuracy_score and classification_report
- Accuracy of linear SVM model: 39% or 0.39



```
# importing SVM model from sklearn module
from sklearn.svm import LinearSVC
# fitting the data to the linear svm model and predicting test data
classifier = LinearSVC()
classifier.fit(train_x,train_y)
y_pred = classifier.predict(test_x)

# printing accuracy and classification of the model prediction
print(classification_report(y_pred, test_y))
print("Accuracy of the model: ",accuracy_score(y_pred, test_y))
```

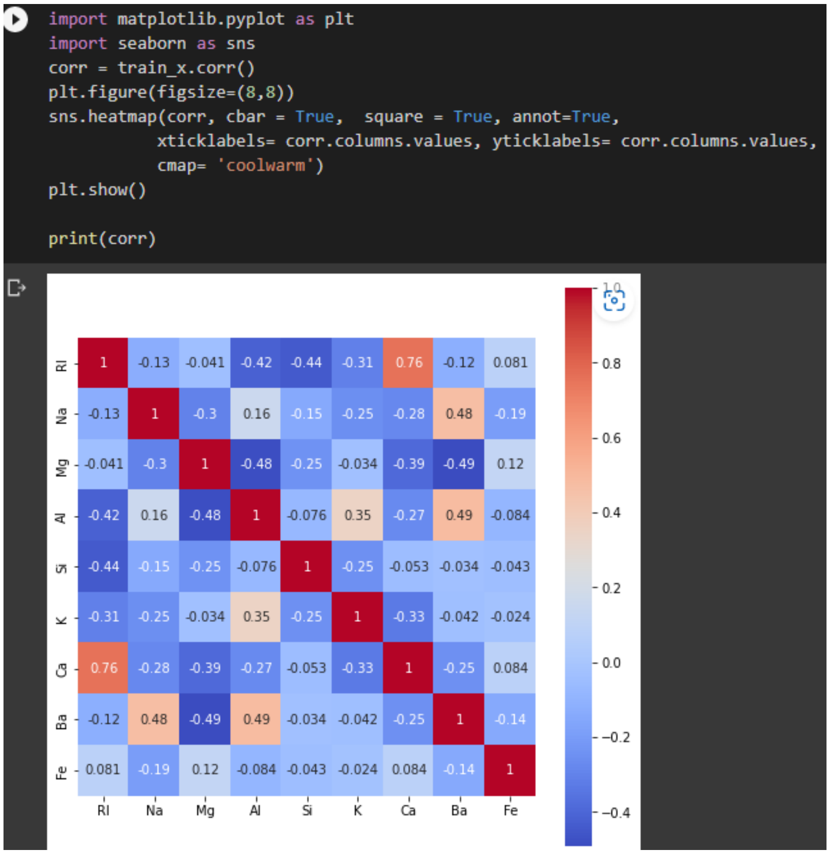
```
              precision    recall  f1-score   support

           1       0.26      0.62      0.37         8
           2       0.58      0.29      0.39        24
           3       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         2
           6       1.00      0.33      0.50         3
           7       1.00      0.67      0.80         6

    accuracy                           0.40        43
   macro avg       0.47      0.32      0.34        43
weighted avg       0.58      0.40      0.43        43

Accuracy of the model:  0.3953488372093023
```

- Visualizing correlation of dataset using heatmap

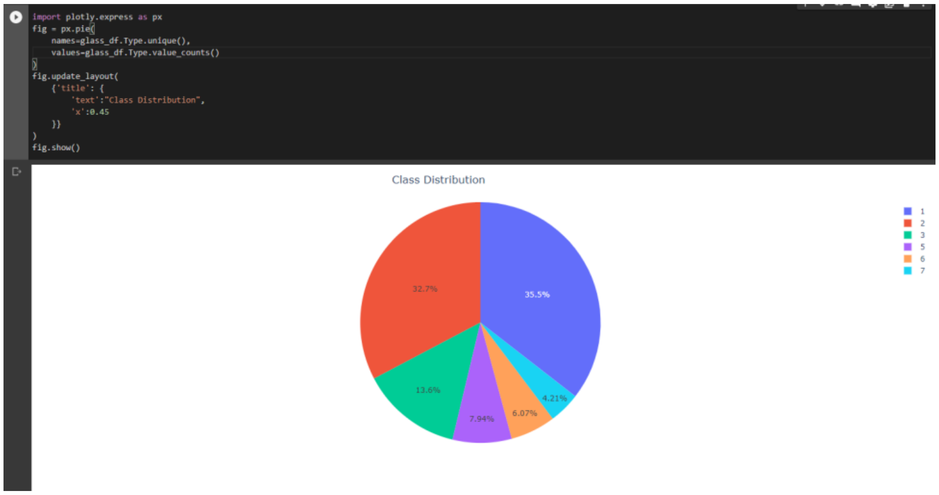


```
import matplotlib.pyplot as plt
import seaborn as sns
corr = train_x.corr()
plt.figure(figsize=(8,8))
sns.heatmap(corr, cbar = True,  square = True, annot=True,
            xticklabels= corr.columns.values, yticklabels= corr.columns.values,
            cmap= 'coolwarm')
plt.show()

print(corr)
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
RI	1	-0.13	-0.041	-0.42	-0.44	-0.31	0.76	-0.12	0.081
Na	-0.13	1	-0.3	0.16	-0.15	-0.25	-0.28	0.48	-0.19
Mg	-0.041	-0.3	1	-0.48	-0.25	-0.034	-0.39	-0.49	0.12
Al	-0.42	0.16	-0.48	1	-0.076	0.35	-0.27	0.49	-0.084
Si	-0.44	-0.15	-0.25	-0.076	1	-0.25	-0.053	-0.034	-0.043
K	-0.31	-0.25	-0.034	0.35	-0.25	1	-0.33	-0.042	-0.024
Ca	0.76	-0.28	-0.39	-0.27	-0.053	-0.33	1	-0.25	0.084
Ba	-0.12	0.48	-0.49	0.49	-0.034	-0.042	-0.25	1	-0.14
Fe	0.081	-0.19	0.12	-0.084	-0.043	-0.024	0.084	-0.14	1

- Visualizing glass types and percentage of each type



```
import plotly.express as px
fig = px.pie(
    names=glass_df.Type.unique(),
    values=glass_df.Type.value_counts()
)
fig.update_layout(
    {'title': {
        'text':"Class Distribution",
        'x':0.45
    }}
)
fig.show()
```

Which algorithm you got better accuracy? Can you justify why?

Linear SVM got the high accuracy.

The Linear SVM algorithm performs well on glass dataset because SVM is more powerful to handle classification task and it generalizes well in high dimensional data space. On the other hand Naïve Bayes works on the concept of conditional probability.

The biggest difference between these two models is Gaussian Naïve bayes each feature as independent one while SVM tries to build a linear relationship between the features.