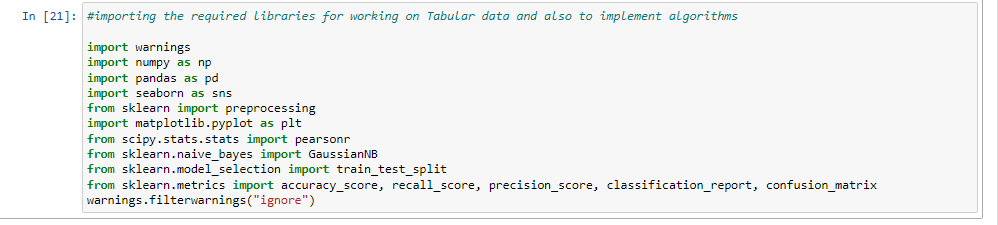
**Mahesh Kumar Uppu**

**700741747**



**1. Pandas**

1. Read the provided CSV file ‘data.csv’. <https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing>
2. Show the basic statistical description about the data.
3. Check if the data has null values. a. Replace the null values with the mean
4. Select at least two columns and aggregate the data using: min, max, count, mean.
5. Filter the dataframe to select the rows with calories values between 500 and 1000.
6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
7. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”.
8. Delete the “Maxpulse” column from the main df dataframe
9. Convert the datatype of Calories column to int datatype.
10. Using pandas create a scatter plot for the two columns (Duration and Calories)

**Source Code:**

**import warnings**

**import numpy as np**

**import pandas as pd**

**import seaborn as sns**

**from sklearn import preprocessing**

**import matplotlib.pyplot as plt**

**from scipy.stats.stats import pearsonr**

**from sklearn.naive\_bayes import GaussianNB**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, classification\_report, confusion\_matrix**

**warnings.filterwarnings("ignore")**

**1. Read the provided CSV file ‘data.csv’. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing**

**df = pd.read\_csv("/Users/varnanemulla/Documents/ML File/data.csv")**

**df.head()**

**#2. Show the basic statistical description about the data.**

**df.describe()**

**#3. Check if the data has null values.**

**df.isnull().any()**

**#Replace the null values with the mean**

**df.fillna(df.mean(), inplace=True)**

**df.isnull().any()**

**#4. Select at least two columns and aggregate the data using: min, max, count, mean.**

**df.agg({'Maxpulse':['min','max','count','mean'],'Calories':['min','max','count','mean']})**

**#5. Filter the dataframe to select the rows with calories values between 500 and 1000.**

**df.loc[(df['Calories']>500)&(df['Calories']<1000)]**

**#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.**

**df.loc[(df['Calories']>500)&(df['Pulse']<100)]**

**#7. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”.**

**df\_modified = df[['Duration','Pulse','Calories']]**

**df\_modified.head()**

**#8. Delete the “Maxpulse” column from the main df dataframe**

**del df['Maxpulse']**

**df.head()**

**df.dtypes**

**#9. Convert the datatype of Calories column to int datatype.**

**df['Calories'] = df['Calories'].astype(np.int64)**

**df.dtypes**

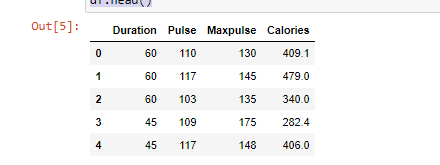
**#10. Using pandas create a scatter plot for the two columns (Duration and Calories).**

**df.plot.scatter(x='Duration',y='Calories',c='blue')**

#1. Read the provided CSV file ‘data.csv’. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing

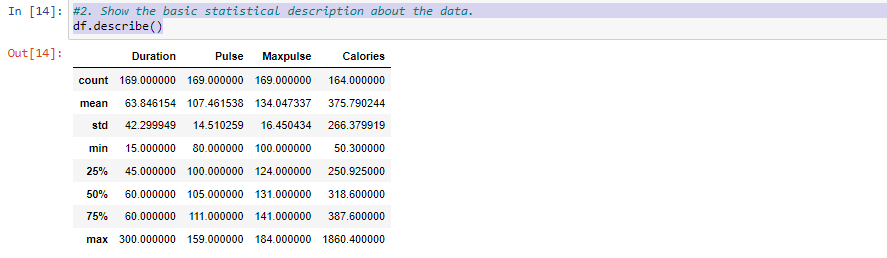
df = pd.read\_csv("data.csv")

df.head()



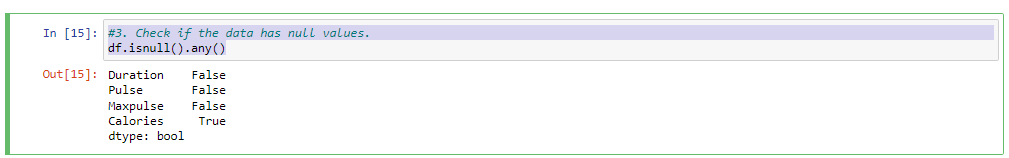
#2. Show the basic statistical description about the data.

df.describe()



#3. Check if the data has null values.

df.isnull().any()



#Replace the null values with the mean

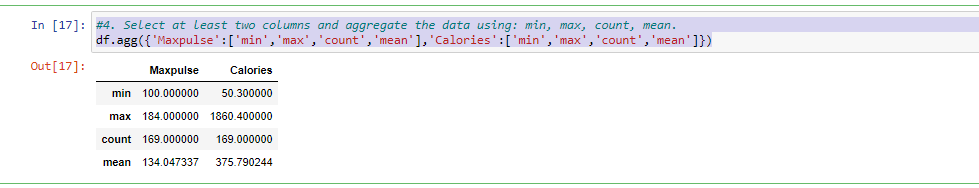
df.fillna(df.mean(), inplace=True)

df.isnull().any()



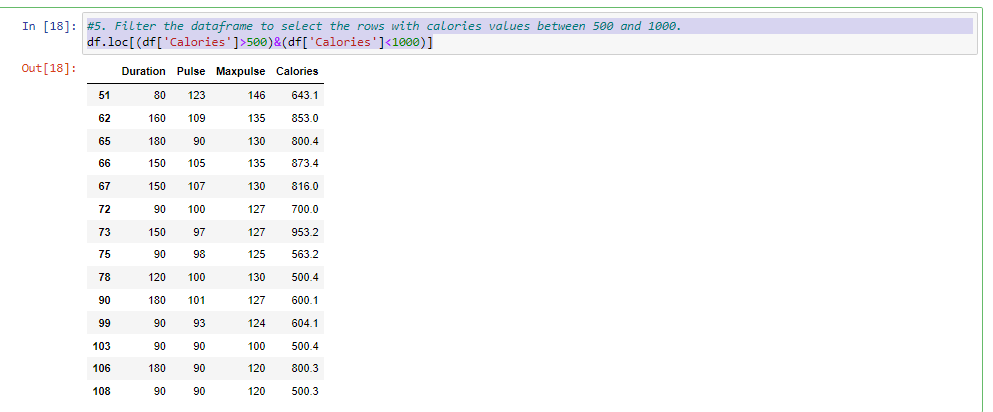
#4. Select at least two columns and aggregate the data using: min, max, count, mean.

df.agg({'Maxpulse':['min','max','count','mean'],'Calories':['min','max','count','mean']})



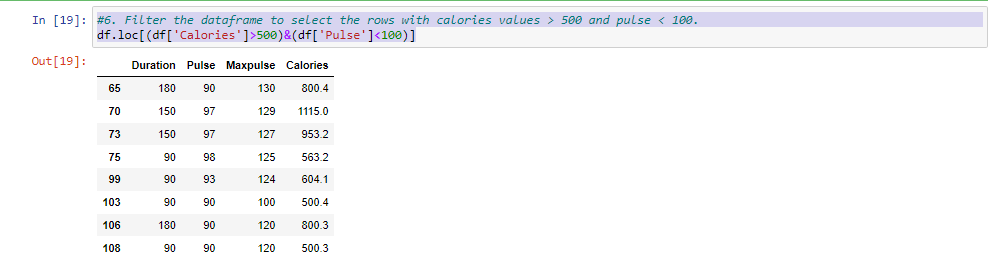
#5. Filter the dataframe to select the rows with calories values between 500 and 1000.

df.loc[(df['Calories']>500)&(df['Calories']<1000)]



#6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

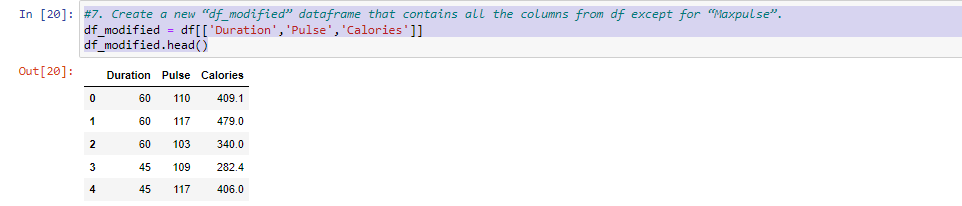
df.loc[(df['Calories']>500)&(df['Pulse']<100)]



#7. Create a new “df\_modified” dataframe that contains all the columns from df except for “Maxpulse”.

df\_modified = df[['Duration','Pulse','Calories']]

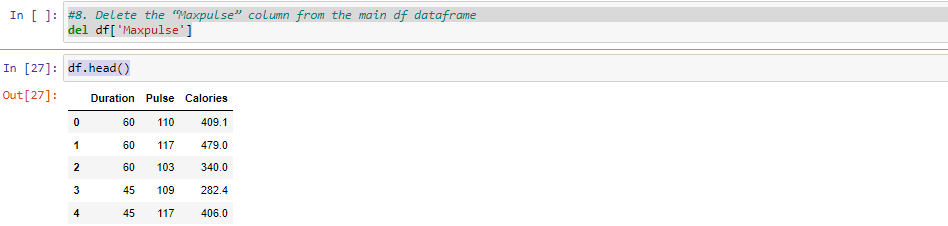
df\_modified.head()



#8. Delete the “Maxpulse” column from the main df dataframe

del df['Maxpulse']

df.head()



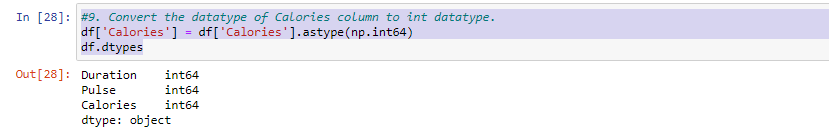
df.dtypes



#9. Convert the datatype of Calories column to int datatype.

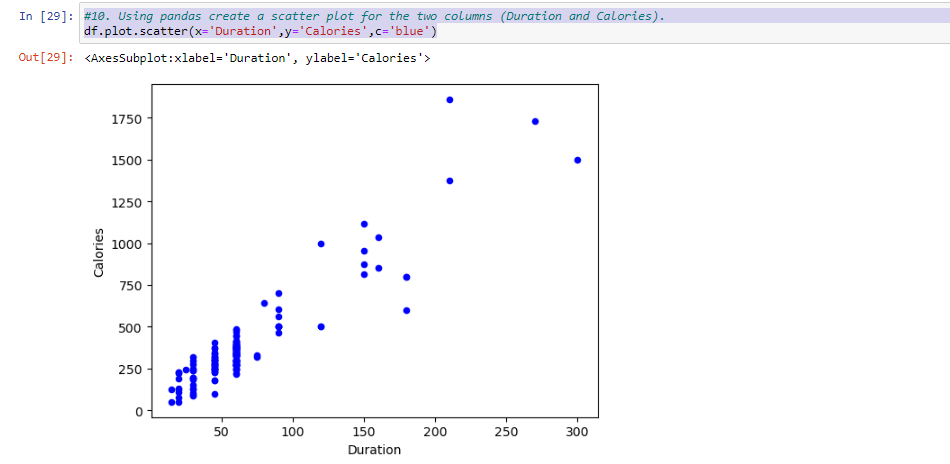
df['Calories'] = df['Calories'].astype(np.int64)

df.dtypes



#10. Using pandas create a scatter plot for the two columns (Duration and Calories).

df.plot.scatter(x='Duration',y='Calories',c='blue')



**1. (Titanic Dataset)**

1. Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case inclass. 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.

**Source Code:**

**#Loading the data file into te program**

**df=pd.read\_csv("/Users/varnanemulla/Documents/ML File/train.csv")**

**df.head()**

**#converted categorical data to numerical values for correlation calculation**

**label\_encoder = preprocessing.LabelEncoder()**

**df['Sex'] = label\_encoder.fit\_transform(df.Sex.values)**

**#Calculation of correlation for 'Survived' and 'Sex' in data**

**correlation\_Value= df['Survived'].corr(df['Sex'])**

**print(correlation\_Value)**

**#print correlation matrix**

**matrix = df.corr()**

**print(matrix)**

**# One way of visualizing correlation matrix in form of spread chart**

**df.corr().style.background\_gradient(cmap="Reds")**

**#Second form of visuaizing correlation matriX using heatmap() from seaborn**

**sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')**

**plt.show()**

**#Loaded data files test and train and merged files**

**train\_raw = pd.read\_csv('/Users/varnanemulla/Documents/ML File/train.csv')**

**test\_raw = pd.read\_csv('/Users/varnanemulla/Documents/ML File/test.csv')**

**train\_raw['train'] = 1**

**test\_raw['train'] = 0**

**df = train\_raw.append(test\_raw, sort=False)**

**features = ['Age', 'Embarked', 'Fare', 'Parch', 'Pclass', 'Sex', 'SibSp']**

**target = 'Survived'**

**df = df[features + [target] + ['train']]**

**df['Sex'] = df['Sex'].replace(["female", "male"], [0, 1])**

**df['Embarked'] = df['Embarked'].replace(['S', 'C', 'Q'], [1, 2, 3])**

**train = df.query('train == 1')**

**test = df.query('train == 0')**

**# Drop missing values from the train set.**

**train.dropna(axis=0, inplace=True)**

**labels = train[target].values**

**train.drop(['train', target, 'Pclass'], axis=1, inplace=True)**

**test.drop(['train', target, 'Pclass'], axis=1, inplace=True)**

**#Test and train split**

**X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(train, labels, test\_size=0.2, random\_state=1)**

**classifier = GaussianNB()**

**classifier.fit(X\_train, Y\_train)**

**y\_pred = classifier.predict(X\_val)**

**# Summary of the predictions made by the classifier**

**print(classification\_report(Y\_val, y\_pred))**

**print(confusion\_matrix(Y\_val, y\_pred))**

**# Accuracy score**

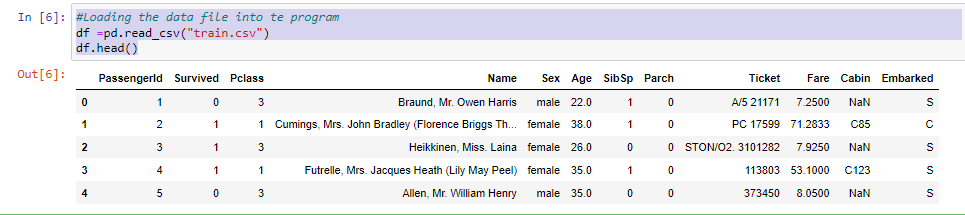
**from sklearn.metrics import accuracy\_score**

**print('accuracy is',accuracy\_score(Y\_val, y\_pred))**

#Loading the data file into te program

df =pd.read\_csv("train.csv")

df.head()



#converted categorical data to numerical values for correlation calculation

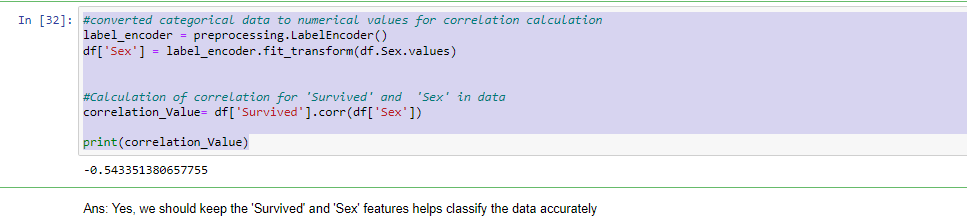
label\_encoder = preprocessing.LabelEncoder()

df['Sex'] = label\_encoder.fit\_transform(df.Sex.values)

#Calculation of correlation for 'Survived' and 'Sex' in data

correlation\_Value= df['Survived'].corr(df['Sex'])

print(correlation\_Value)

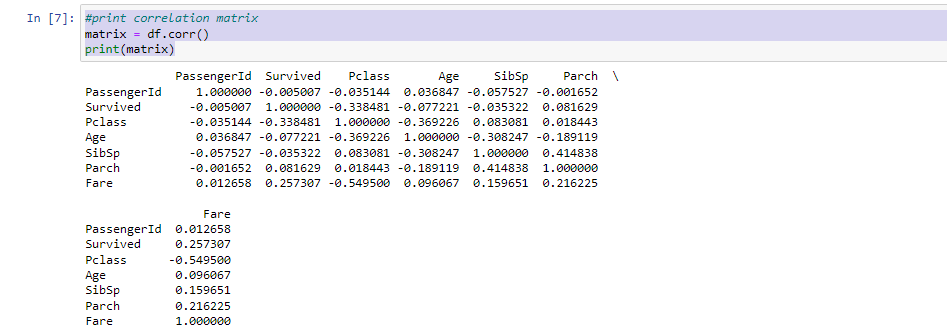


Ans: Yes, we should keep the 'Survived' and 'Sex' features helps classify the data accurately

#print correlation matrix

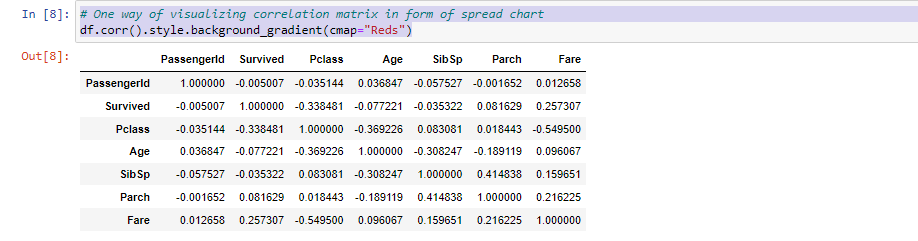
matrix = df.corr()

print(matrix)



# One way of visualizing correlation matrix in form of spread chart

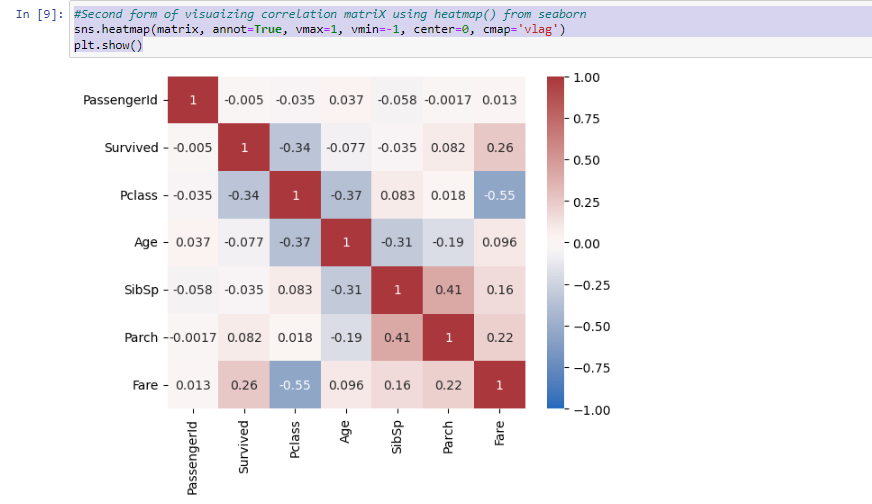
df.corr().style.background\_gradient(cmap="Reds")



#Second form of visuaizing correlation matriX using heatmap() from seaborn

sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')

plt.show()



#Loaded data files test and train and merged files

train\_raw = pd.read\_csv('train.csv')

test\_raw = pd.read\_csv('test.csv')

train\_raw['train'] = 1

test\_raw['train'] = 0

df = train\_raw.append(test\_raw, sort=False)

features = ['Age', 'Embarked', 'Fare', 'Parch', 'Pclass', 'Sex', 'SibSp']

target = 'Survived'

df = df[features + [target] + ['train']]

df['Sex'] = df['Sex'].replace(["female", "male"], [0, 1])

df['Embarked'] = df['Embarked'].replace(['S', 'C', 'Q'], [1, 2, 3])

train = df.query('train == 1')

test = df.query('train == 0')

# Drop missing values from the train set.

train.dropna(axis=0, inplace=True)

labels = train[target].values

train.drop(['train', target, 'Pclass'], axis=1, inplace=True)

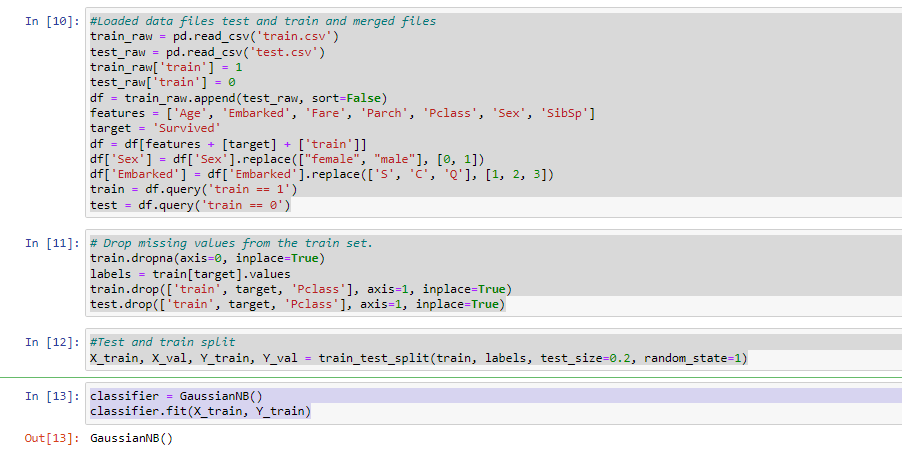
test.drop(['train', target, 'Pclass'], axis=1, inplace=True)

#Test and train split

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(train, labels, test\_size=0.2, random\_state=1)

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)



y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

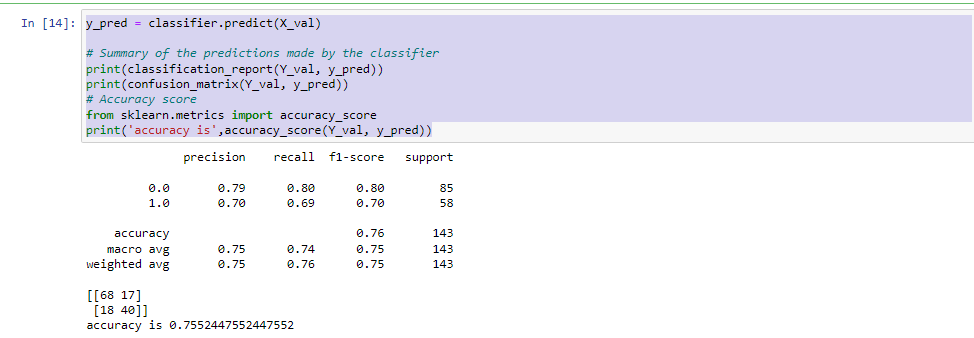
print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))



**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)
3. Implement linear SVM method using scikit 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.
4. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred) Do at least two visualizations to describe or show correlations in the Glass Dataset. Which algorithm you got better accuracy? Can you justify why?

**Source Code:**

**glass=pd.read\_csv("/Users/varnanemulla/Documents/ML File/glass.csv")**

**glass.head()**

**glass.corr().style.background\_gradient(cmap="Reds")**

**sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')**

**plt.show()**

**features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']**

**target = 'Type'**

**X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(glass[::-1], glass['Type'],test\_size=0.2, random\_state=1)**

**classifier = GaussianNB()**

**classifier.fit(X\_train, Y\_train)**

**y\_pred = classifier.predict(X\_val)**

**# Summary of the predictions made by the classifier**

**print(classification\_report(Y\_val, y\_pred))**

**print(confusion\_matrix(Y\_val, y\_pred))**

**# Accuracy score**

**print('accuracy is',accuracy\_score(Y\_val, y\_pred))**

**from sklearn.svm import SVC, LinearSVC**

**classifier = LinearSVC()**

**classifier.fit(X\_train, Y\_train)**

**y\_pred = classifier.predict(X\_val)**

**# Summary of the predictions made by the classifier**

**print(classification\_report(Y\_val, y\_pred))**

**print(confusion\_matrix(Y\_val, y\_pred))**

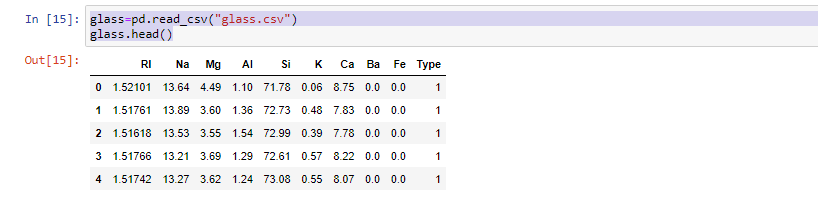
**# Accuracy score**

**from sklearn.metrics import accuracy\_score**

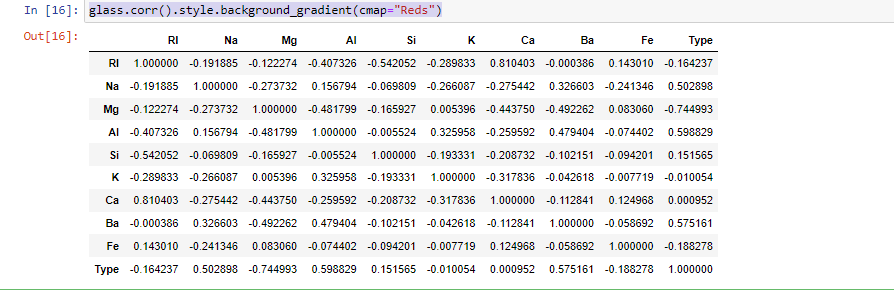
**print('accuracy is',accuracy\_score(Y\_val, y\_pred))**

glass=pd.read\_csv("glass.csv")

glass.head()

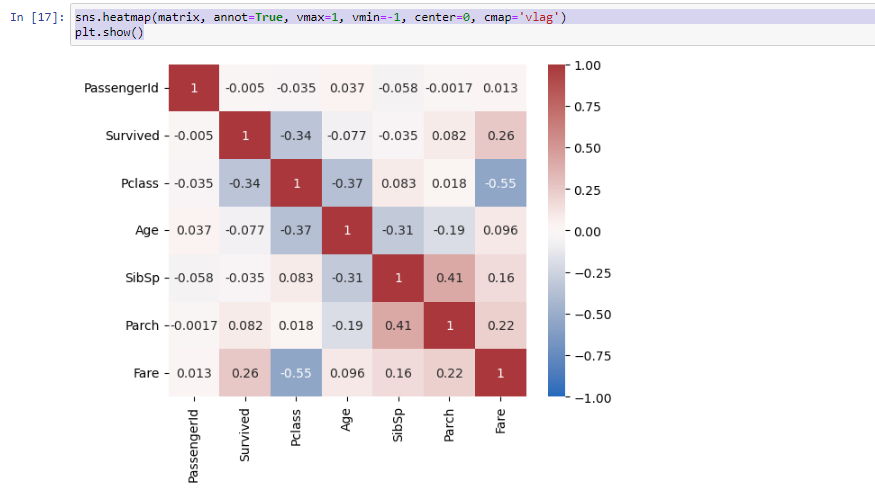


glass.corr().style.background\_gradient(cmap="Reds")



sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')

plt.show()



features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']

target = 'Type'

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(glass[::-1], glass['Type'],test\_size=0.2, random\_state=1)

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

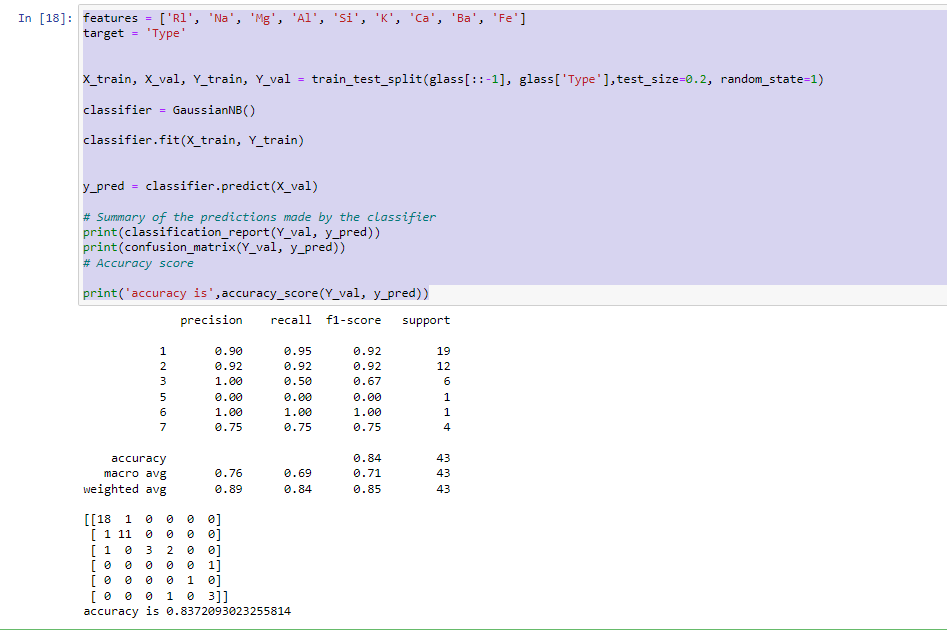
# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))



features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']

target = 'Type'

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(glass[::-1], glass['Type'],test\_size=0.2, random\_state=1)

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

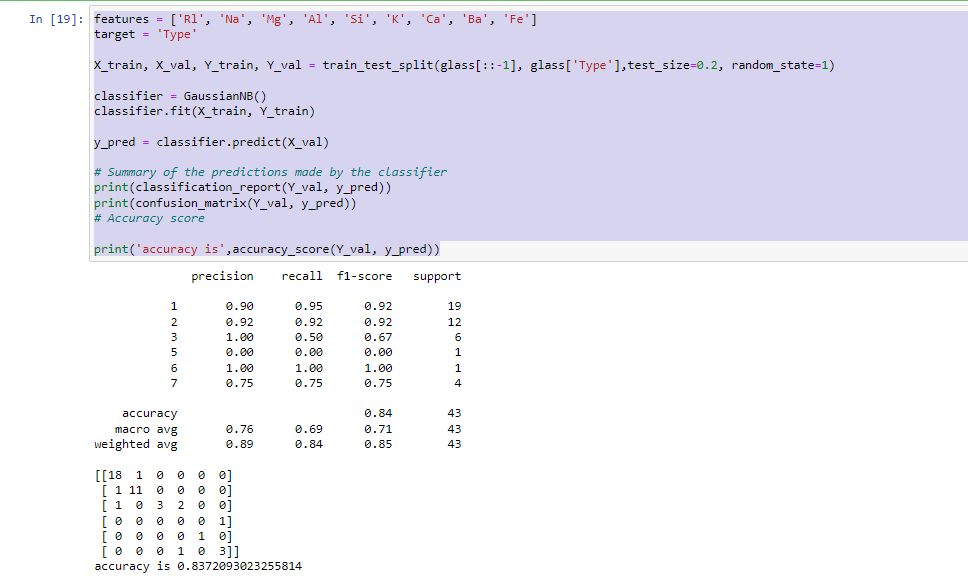
# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))



from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

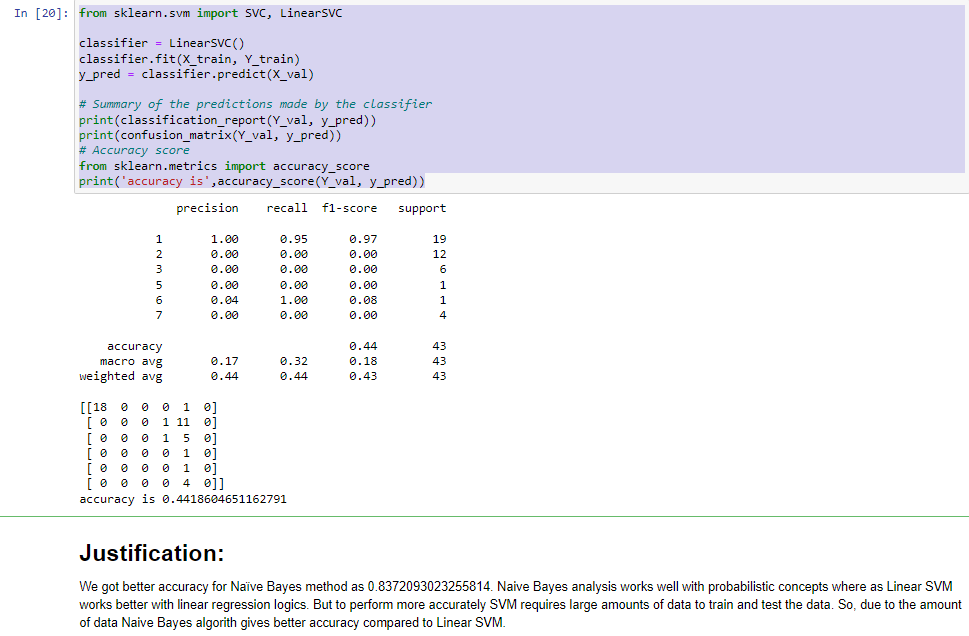
print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))



# Justification:

We got better accuracy for Naïve Bayes method as 0.8372093023255814. Naive Bayes analysis works well with probabilistic concepts where as Linear SVM works better with linear regression logics. But to perform more accurately SVM requires large amounts of data to train and test the data. So, due to the amount of data Naive Bayes algorith gives better accuracy compared to Linear SVM.