MACHINE LEARNING ASSIGNMENT 4

Name: Rupa Mallempati

Student id:700740339

Video link : https://drive.google.com/file/d/17ZBduvqYTcVISb80-KjFwY3w0Hwu5WkV/view?usp=share_link

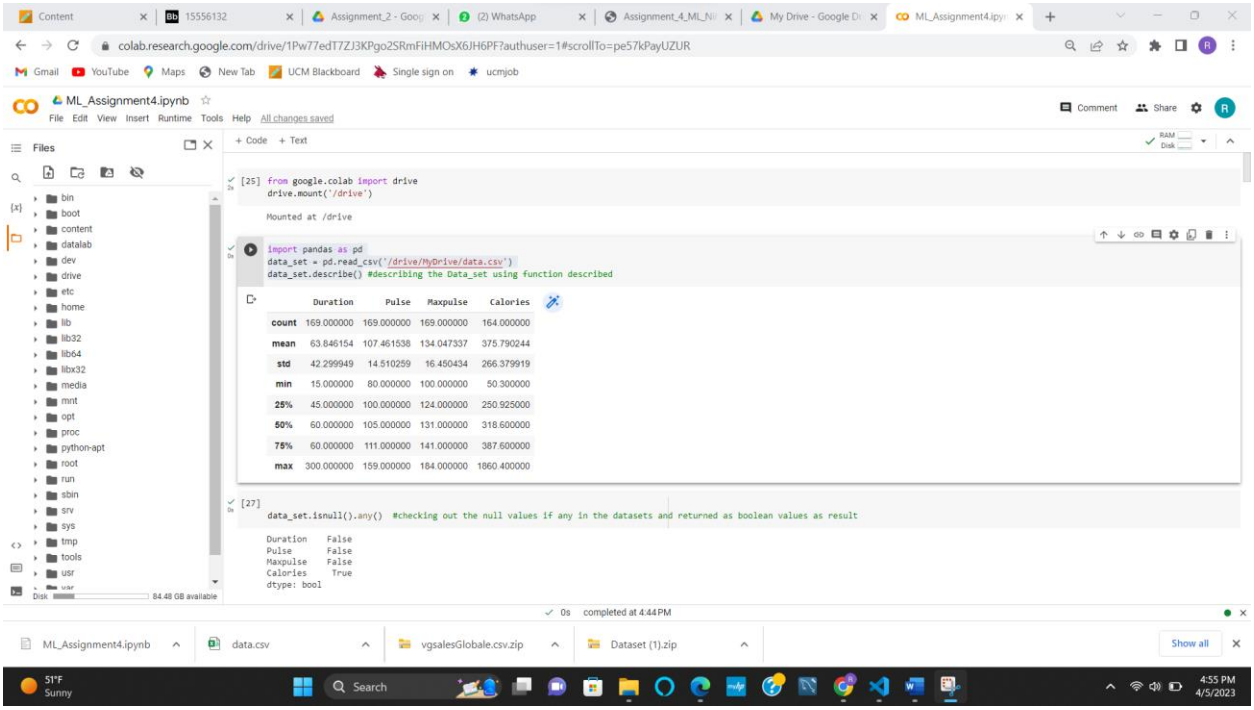
Github link : https://github.com/rupamallempati/ML_Assignment4.git

1. Pandas

1. Read the provided CSV file 'data.csv'.

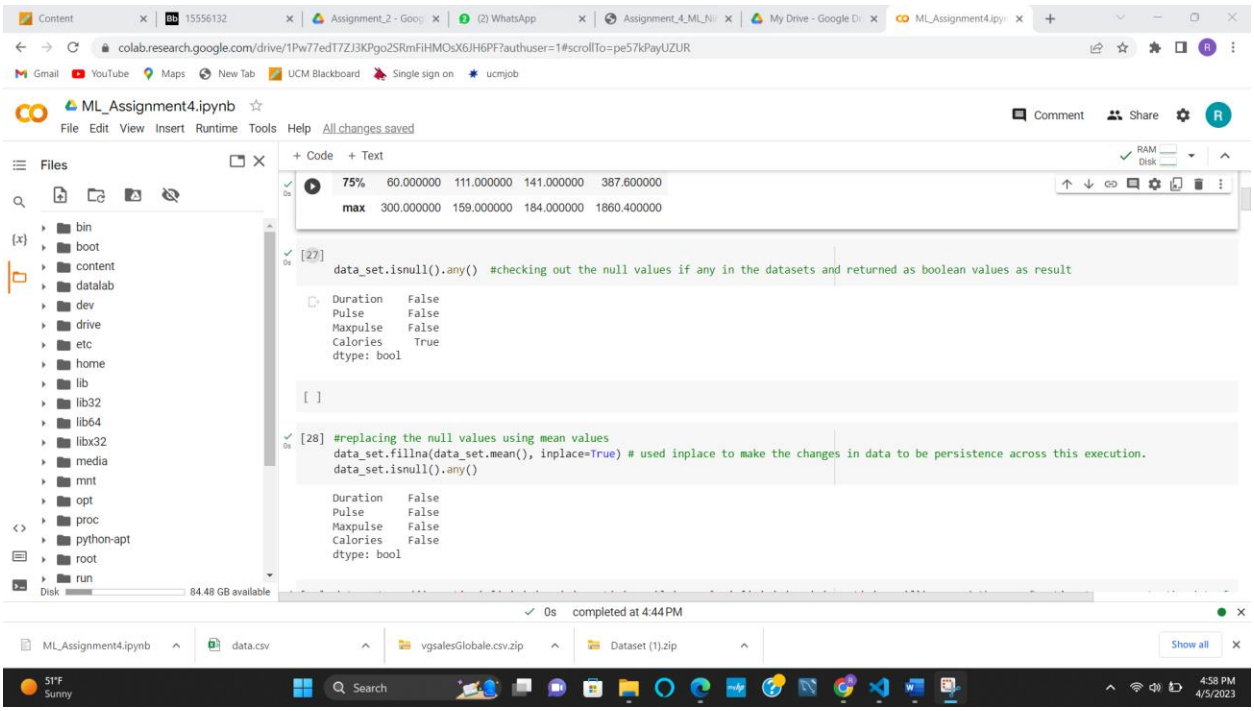
https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing

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

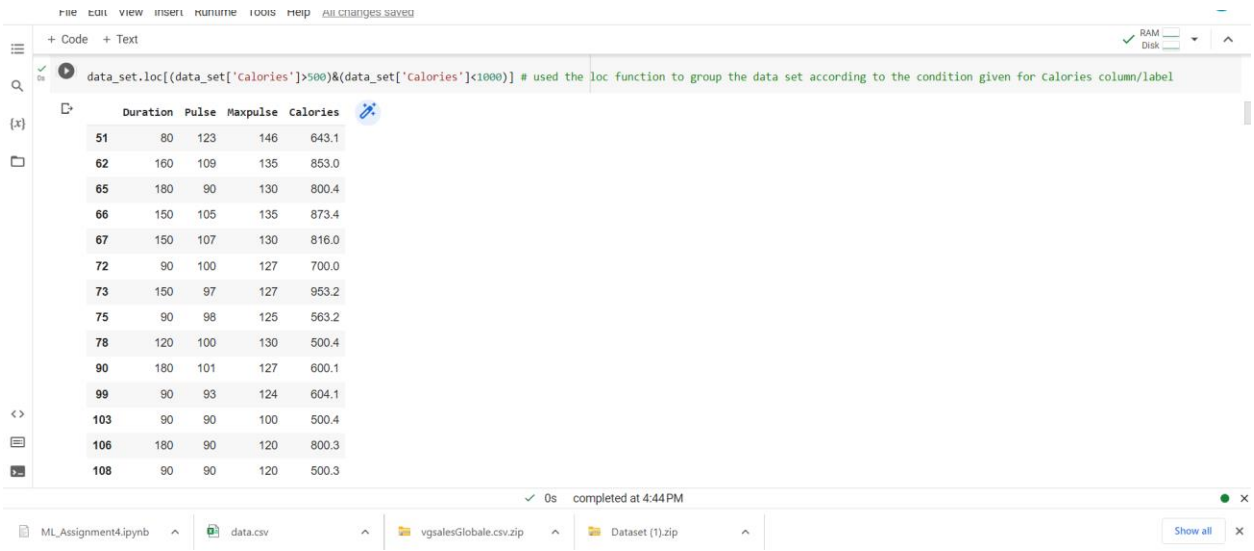


4. Select at least two columns and aggregate the data using: min, max, count, mean

```
data_set.agg({'Duration':['min','max','count','mean'],'Maxpulse':['min','max','count','mean']}) # used the agg function to aggregate the data for 2 columns
```

	Duration	Maxpulse
min	15.000000	100.000000
max	300.000000	184.000000
count	169.000000	169.000000
mean	63.846154	134.047337

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



```
data_set.loc[(data_set['Calories']>500)&(data_set['Calories']<1000)] # used the loc function to group the data set according to the condition given for Calories column/label
```

	Duration	Pulse	Maxpulse	Calories
51	80	123	146	643.1
62	160	109	135	853.0
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
72	90	100	127	700.0
73	150	97	127	953.2
75	90	98	125	563.2
78	120	100	130	500.4
90	180	101	127	600.1
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

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

```
data_set.loc[(data_set['Calories']>500)&(data_set['Pulse']<100)] # used the loc function to group the data set according to the condition given for Calories column/label
```

	Duration	Pulse	Maxpulse	Calories
65	180	90	130	800.4
70	150	97	129	1115.0
73	150	97	127	953.2
75	90	98	125	563.2
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

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

```
df_modified = data_set[['Duration','Pulse','Calories']]  # excluded the Maxpulse column from the dataset
df_modified.head() #printing the top overview of data set in df_modified
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0

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

```
del data_set['Maxpulse']    #deleting the label/column Maxpulse from the dataset
display (data_set)
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0
...	...	...	...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows × 3 columns

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

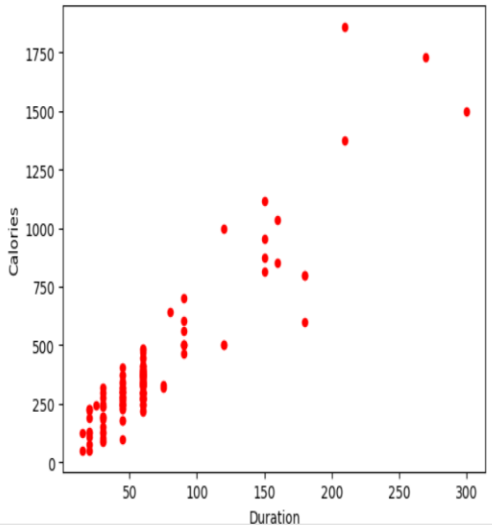
```
data_set.dtypes  #before the conversion of data type for the value in data set
data_set['Calories'] = data_set['Calories'].astype(int)  # applied conversion of data type for the data in label Calories
data_set.dtypes  #display after conversion
```

```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

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

```
data_set.plot.scatter(x='Duration',y='Calories',c='red') # displaying the pictorial plot for two column(Duration and calories)
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  scatter = ax.scatter(
<Axes: xlabel='Duration', ylabel='Calories'>
```



✓ 0s completed at 4:44PM

1. (Titanic Dataset)

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

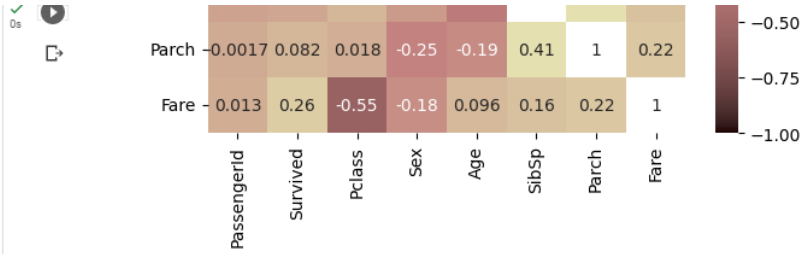
```
# importing libraries all needed

import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
df=pd.read_csv("/drive/MyDrive/train.csv")
# Below we are Finding the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case

le = preprocessing.LabelEncoder()
df['Sex'] = le.fit_transform(df.Sex.values)
df['Survived'].corr(df['Sex'])
```

a. Do you think we should keep this feature?

Yes, as By using correlation to demonstrate which variable has a high or low correlation with another variable, we can plot correlation matrices to show the strength of the relationship between the dependent and independent variables/columns.

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

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




```
[40] # Implementing Naïve Bayes methods ,,,,1.3.1

     train_raw = pd.read_csv('/drive/MyDrive/train.csv')
     test_raw = pd.read_csv('/drive/MyDrive/test.csv')

     # Join data to analyse and process the set as one.
     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']]
     # Categorical values need to be transformed into numeric.
     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')
```

Accuracy:

```
#1.3.2
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
display('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.84        43
   macro avg       0.76      0.69      0.71        43
weighted avg       0.89      0.84      0.85        43

[[18  1  0  0  0  0]
 [ 1 11  0  0  0  0]
 [ 1  0  3  2  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
'accuracy is'
0.8372093023255814
```

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.

```
[47] #question 2 starts # Reading file from glass csv data file from current directory
     glass=pd.read_csv("/drive/MyDrive/glass.csv")
```

```
#2.1.b
#2.2
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)
# Summarizing 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 #2.1.1,2
display('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
```

```
print(confusion_matrix(Y_val, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score #2.1.1,2
display('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.84        43
   macro avg       0.76      0.69      0.71        43
weighted avg       0.89      0.84      0.85        43

[[18  1  0  0  0  0]
 [ 1 11  0  0  0  0]
 [ 1  0  3  2  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
'accuracy is'
0.8372093023255814
```

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

1. 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.

2 . Evaluate the model on testing part using score and

```
#2nd part 2nd question
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
display('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      0.95      0.97        19
           2       0.85      0.92      0.88        12
           3       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         1
```

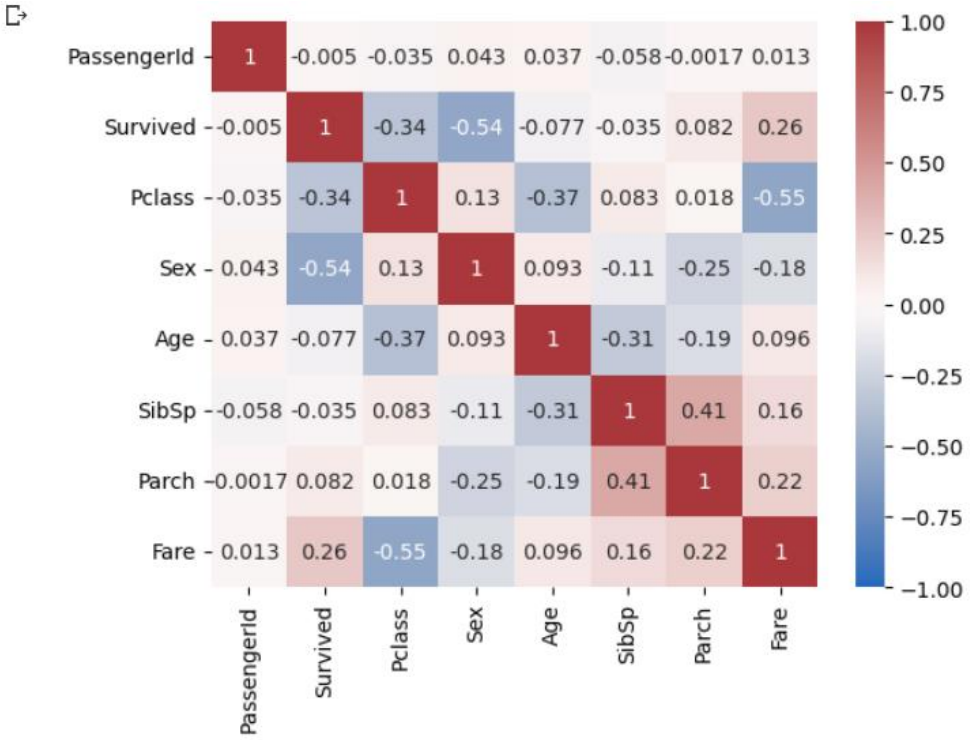
```
from sklearn.metrics import accuracy_score
display('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      0.95      0.97        19
           2       0.85      0.92      0.88        12
           3       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         1
           6       0.00      0.00      0.00         1
           7       0.44      1.00      0.62         4

    accuracy                           0.77        43
   macro avg       0.38      0.48      0.41        43
weighted avg       0.72      0.77      0.73        43

[[18  1  0  0  0  0]
 [ 0 11  0  0  1  0]
 [ 0  1  0  2  0  3]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  0  4]]
'accuracy is'
0.7674418604651163
```

Do at least two visualizations to describe or show correlations in the Glass Dataset.

```
#last ans in 2nd question #visualization1
glass.corr().style.background_gradient(cmap="Reds")
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000



```
matrix = glass.corr()  #corelations of glass data
display(matrix)
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000

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

ANS: Naïve Bayes

Justification: Logistic regression and SVM are discriminative models, but the Multinomial "Naive Bayes" in this case has higher accuracy in comparison to the SVM algorithm from the executions above. While Naive Bayes is a generative model, logistic regression and SVM are discriminative models. It is well known that a generative model can outperform a discriminative model when you have very little data, taking into account the parameter (Dataset) sensitivity and implementation pattern.