Name: Gaurav Kumar Singh

Registration Number: 19BCE2119

Course: Information Security Analysis and Audit (CSE3501) L21+22

Digital Assignment 2

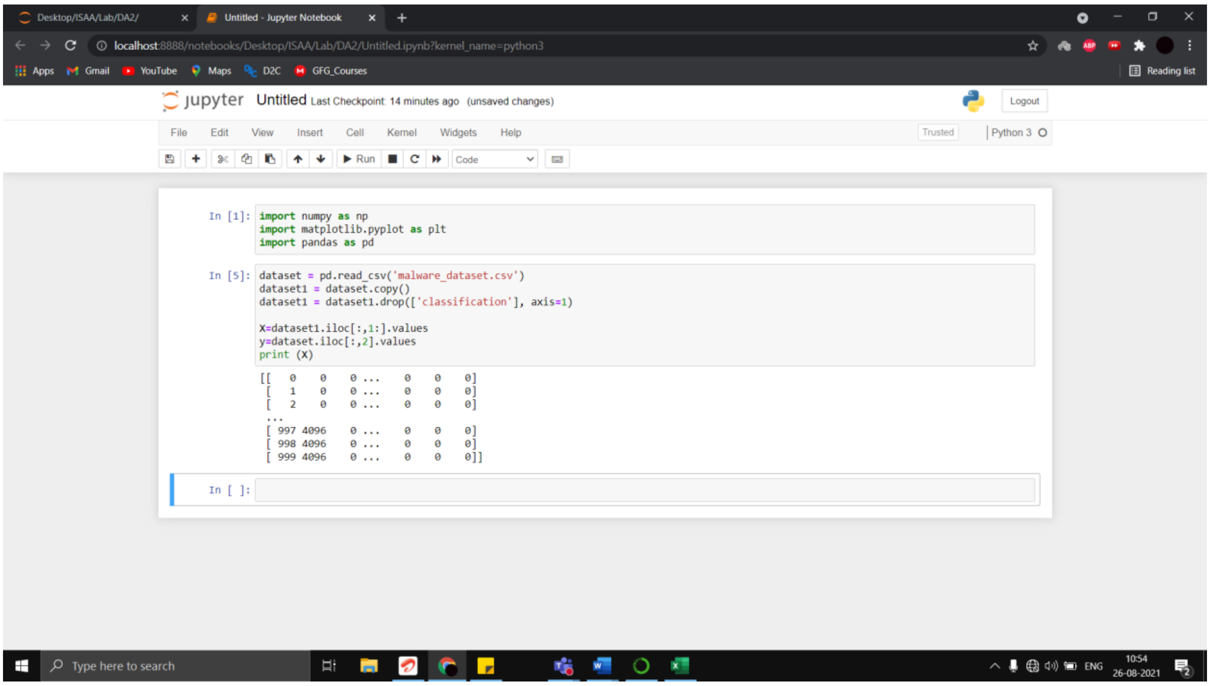
INTRODUCTION MALWARE DATASET

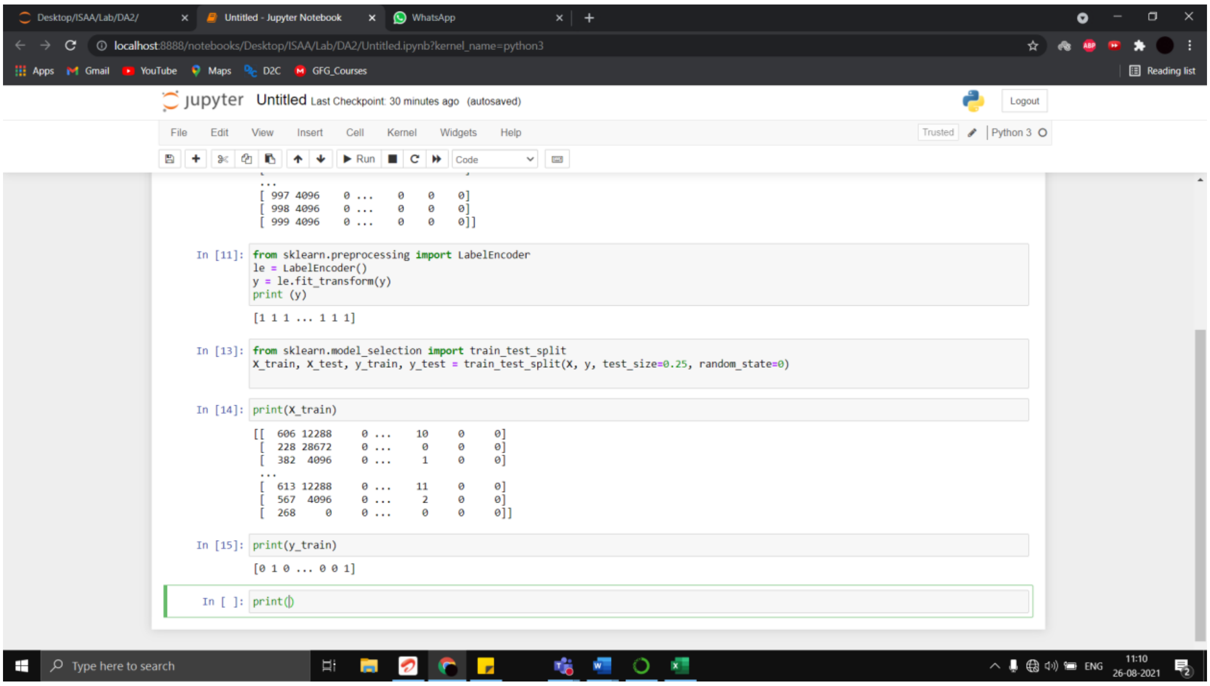
The dataset contains information about various benign and malware along with their characteristics like task priority, virtual memory statistics, etc. With the help of these statistics, we can make predictions about unknown software to predict the risk by classifying it as malware or benign. There are a lot of machine learning algorithms so we can train and test our machine learning algorithms to find out which algorithm works best for our use case.

LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm that is used to classify the data. It is used to classify data in only one of the two maximum possible ways in the case of Binomial Logistic Regression which we are going to use for our test because we have to classify the data as either malware or benign only. We must choose only meaningful variables to train our model and a large sample size is required to train the algorithm. It makes use of logistic functions, also known as sigmoid functions as part of its calculations for the classification of data.

MILESTONES WITH TIMESTAMPS





```
...
[ 997 4096    0 ...    0    0    0]
[ 998 4096    0 ...    0    0    0]
[ 999 4096    0 ...    0    0    0]]
```

In [11]:
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print (y)
```

```
[1 1 1 ... 1 1 1]
```

In [13]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

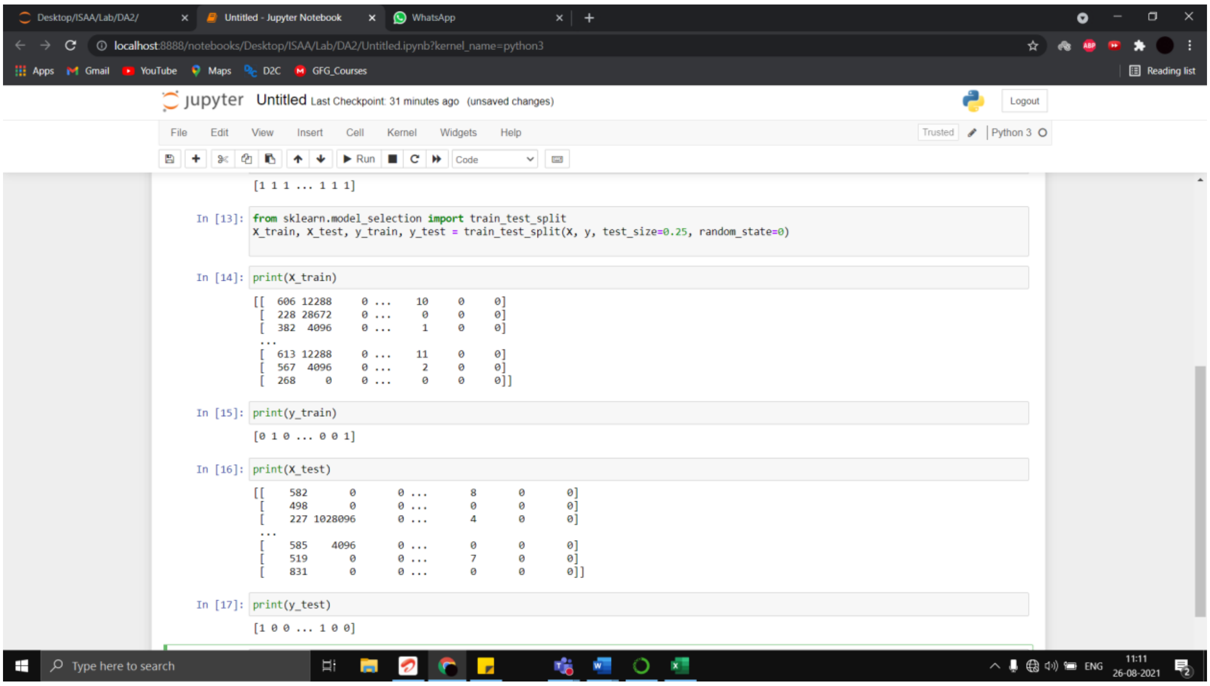
In [14]:
```python
print(X_train)
```

```
[[  606 12288     0 ...    10     0     0]
 [  228 28672     0 ...     0     0     0]
 [  382  4096     0 ...     1     0     0]
 ...
 [  613 12288     0 ...    11     0     0]
 [  567  4096     0 ...     2     0     0]
 [  268     0     0 ...     0     0     0]]
```

In [15]:
```python
print(y_train)
```

```
[0 1 0 ... 0 0 1]
```

In [ ]:
```python
print()
```



```
[1 1 1 ... 1 1 1]
```

In [13]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

In [14]:
```python
print(X_train)
```

```
[[  606 12288     0 ...    10     0     0]
 [  228 28672     0 ...     0     0     0]
 [  382  4096     0 ...     1     0     0]
 ...
 [  613 12288     0 ...    11     0     0]
 [  567  4096     0 ...     2     0     0]
 [  268     0     0 ...     0     0     0]]
```

In [15]:
```python
print(y_train)
```

```
[0 1 0 ... 0 0 1]
```

In [16]:
```python
print(X_test)
```

```
[[    582       0       0 ...       8       0       0]
 [    498       0       0 ...       0       0       0]
 [    227 1028096       0 ...       4       0       0]
 ...
 [    585    4096       0 ...       0       0       0]
 [    519       0       0 ...       7       0       0]
 [    831       0       0 ...       0       0       0]]
```

In [17]:
```python
print(y_test)
```

```
[1 0 0 ... 1 0 0]
```

jupyter Untitled Last Checkpoint: 34 minutes ago (autosaved)

```
 [    585   4096      0 ...      0      0      0]
 [    519      0      0 ...      7      0      0]
 [    831      0      0 ...      0      0      0]]
```

In [17]:
```
print(y_test)
```

```
[1 0 0 ... 1 0 0]
```

In [18]:
```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train= sc.fit_transform(X_train)
X_test= sc.transform(X_test)
print(X_train)
```

```
[[ 0.36731703 -0.15969785  0.         ...  2.5559669   0.
   0.        ]
 [-0.941068   -0.14179605  0.         ... -0.50896701  0.
   0.        ]
 [-0.40802225 -0.16864875  0.         ... -0.20247362  0.
   0.        ]
 ...
 [ 0.39154638 -0.15969785  0.         ...  2.8624603   0.
   0.        ]
 [ 0.23232492 -0.16864875  0.         ...  0.10401977  0.
   0.        ]
 [-0.80261455 -0.1731242   0.         ... -0.50896701  0.
   0.        ]]
```

In [ ]:

jupyter Untitled Last Checkpoint: 42 minutes ago (unsaved changes)

```
   0.        ]
 [-0.40802225 -0.16864875  0.         ... -0.20247362  0.
   0.        ]
 ...
 [ 0.39154638 -0.15969785  0.         ...  2.8624603   0.
   0.        ]
 [ 0.23232492 -0.16864875  0.         ...  0.10401977  0.
   0.        ]
 [-0.80261455 -0.1731242   0.         ... -0.50896701  0.
   0.        ]]
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train,y_train)
```

Out[20]: LogisticRegression(random_state=0)

In [21]:
```
y_pred=classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [0 0]
 [0 0]
 ...
 [1 1]
 [0 0]
 [0 0]]
```

In [ ]:

jupyter Untitled Last Checkpoint: an hour ago (unsaved changes)

In [22]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[11590   938]
 [  560 11912]]
```

Out[22]: 0.94008

In [23]:
```python
from sklearn.metrics import confusion_matrix, precision_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
precision_score(y_test, y_pred)
```

```
[[11590   938]
 [  560 11912]]
```

Out[23]: 0.9270038910505837

In [24]:
```python
from sklearn.metrics import confusion_matrix, recall_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
recall_score(y_test, y_pred)
```

```
[[11590   938]
 [  560 11912]]
```

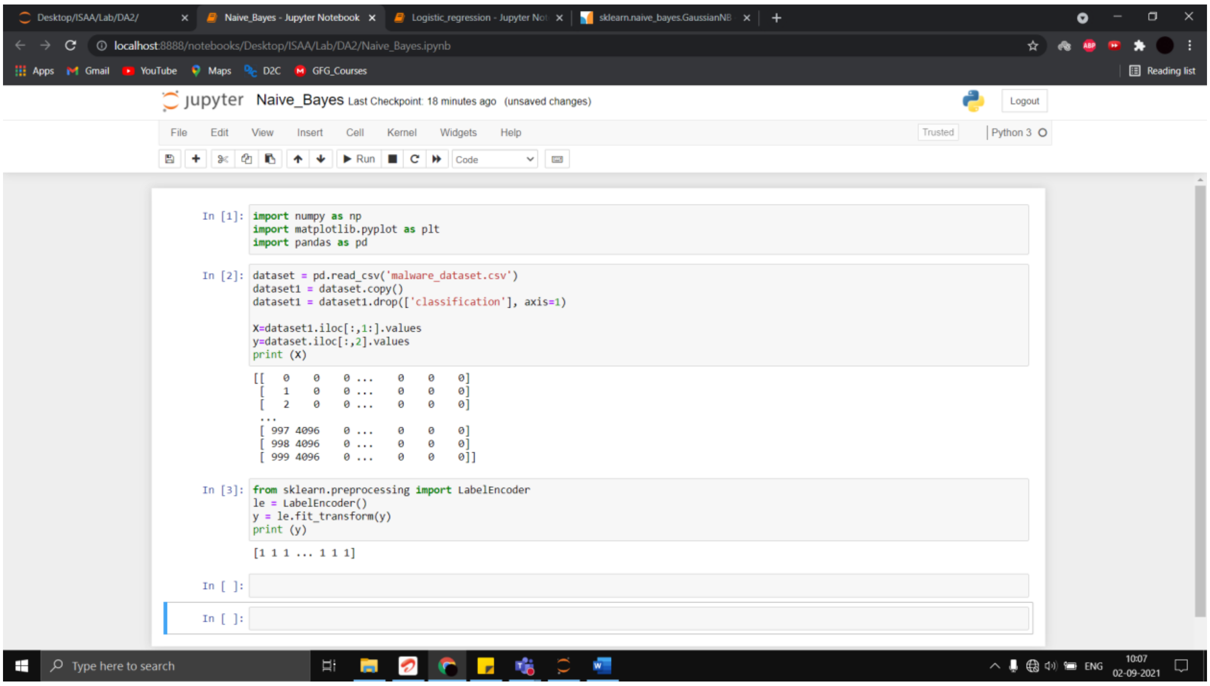
Out[24]: 0.9550994227068633

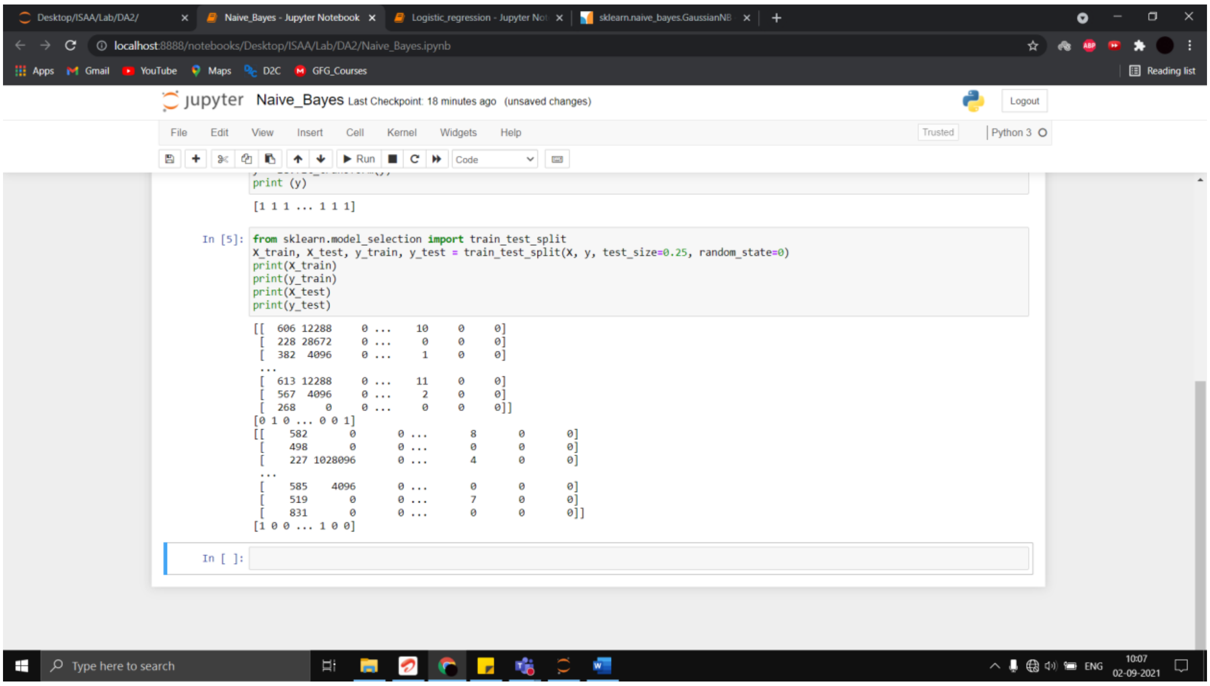
In [ ]:

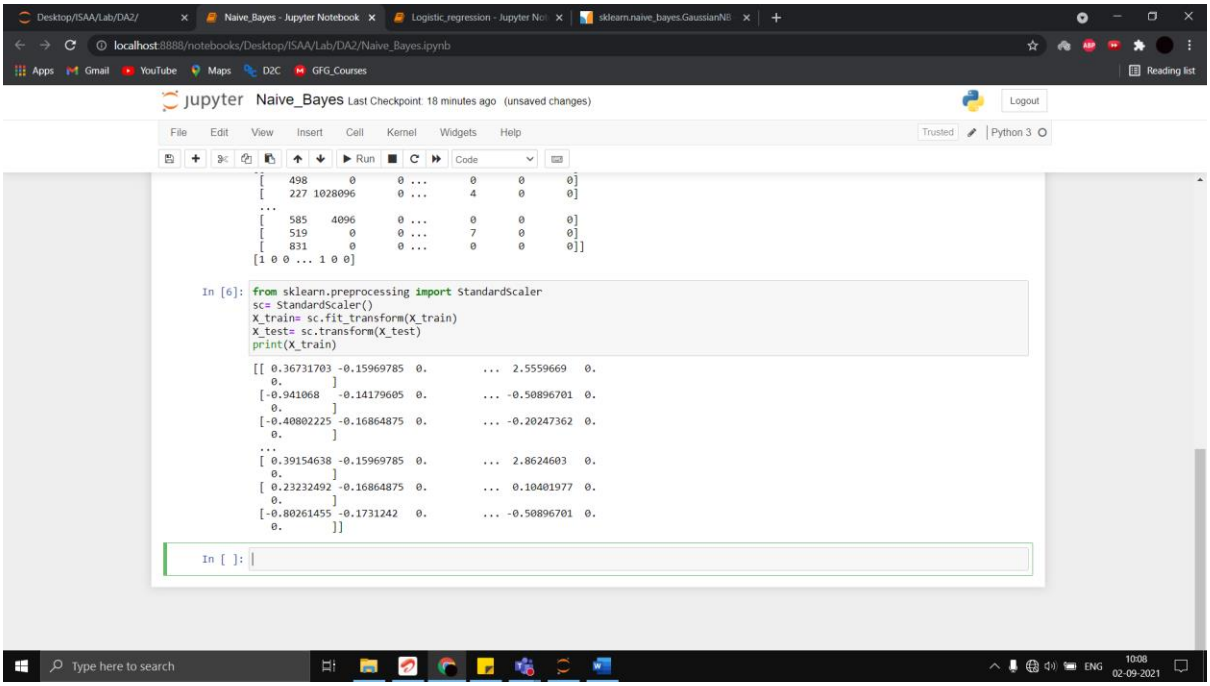
NAÏVE BAYES

Just like Logistic regression, Naïve Bayes is also a supervised learning algorithm based on Bayes Theorem for the classification of data. It is a probabilistic classifier, implying it uses the probability of event occurrence as part of its mathematical calculations. It has many variations in itself, a few of which are Gaussian Naïve Bayes, Multinomial Naïve Bayes etc. We are going to use Gaussian Naïve Bayes for our specific use case for this activity and calculate accuracy scores, recall scores, and precision scores to compare these respective scores to that of logistic regression to find which algorithm gives us the most accurate classification for any unknown data of application which need to be classified as either malware or benign.

MILESTONES WITH TIMESTAMPS




Naive_Bayes Last Checkpoint: 18 minutes ago (unsaved changes)
print (y)
[1 1 1 ... 1 1 1]
In [5]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)
[[606 12288 0 ... 10 0 0]
[228 28672 0 ... 0 0 0]
[382 4096 0 ... 1 0 0]
...
[613 12288 0 ... 11 0 0]
[567 4096 0 ... 2 0 0]
[268 0 0 ... 0 0 0]]
[0 1 0 ... 0 0 1]
[[582 0 0 ... 8 0 0]
[498 0 0 ... 0 0 0]
[227 1028096 0 ... 4 0 0]
...
[585 4096 0 ... 0 0 0]
[519 0 0 ... 7 0 0]
[831 0 0 ... 0 0 0]]
[1 0 0 ... 1 0 0]
In [ ]:


Naive_Bayes Last Checkpoint: 18 minutes ago (unsaved changes)
[498 0 0 ... 0 0 0]
[227 1028096 0 ... 4 0 0]
...
[585 4096 0 ... 0 0 0]
[519 0 0 ... 7 0 0]
[831 0 0 ... 0 0 0]]
[1 0 0 ... 1 0 0]
In [6]:
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train= sc.fit_transform(X_train)
X_test= sc.transform(X_test)
print(X_train)
[[0.36731703 -0.15969785 0. ... 2.5559669 0. 0.]
[-0.941068 -0.14179605 0. ... -0.50896701 0. 0.]
[-0.40802225 -0.16864875 0. ... -0.20247362 0. 0.]
...
[0.39154638 -0.15969785 0. ... 2.8624603 0. 0.]
[0.23232492 -0.16864875 0. ... 0.10401977 0. 0.]
[-0.80261455 -0.1731242 0. ... -0.50896701 0. 0.]]
In [ ]:

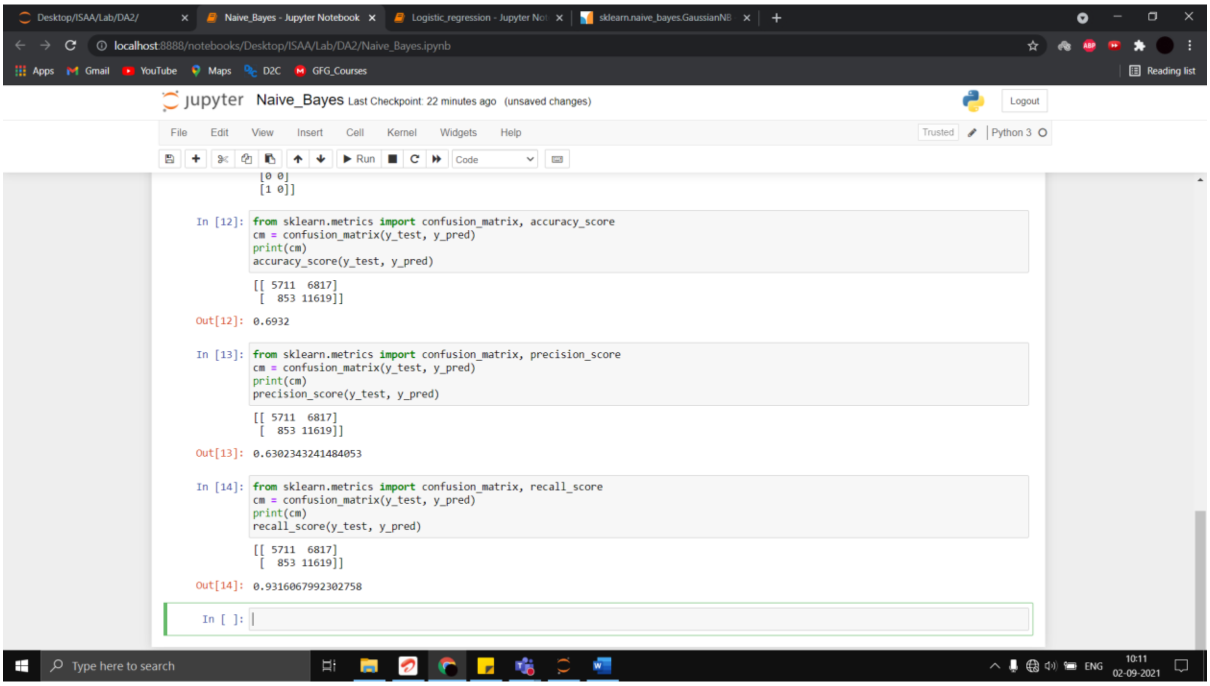
jupyter Naive_Bayes Last Checkpoint: 20 minutes ago (unsaved changes)

```
In [6]: from sklearn.preprocessing import StandardScaler
        sc= StandardScaler()
        X_train= sc.fit_transform(X_train)
        X_test= sc.transform(X_test)
        print(X_train)

[[ 0.36731703 -0.15969785  0.         ...  2.5559669   0.
   0.        ]
 [-0.941068   -0.14179605  0.         ... -0.50896701  0.
   0.        ]
 [-0.40802225 -0.16864875  0.         ... -0.20247362  0.
   0.        ]
 ...
 [ 0.39154638 -0.15969785  0.         ...  2.8624603   0.
   0.        ]
 [ 0.23232492 -0.16864875  0.         ...  0.10401977  0.
   0.        ]
 [-0.80261455 -0.1731242   0.         ... -0.50896701  0.
   0.        ]]

In [10]: from sklearn.naive_bayes import GaussianNB
         classifier = GaussianNB()
         classifier.fit(X_train,y_train)

Out[10]: GaussianNB()

In [ ]:

In [ ]:
```

jupyter Naive_Bayes Last Checkpoint: 21 minutes ago (unsaved changes)

```
   0.        ]
 [-0.40802225 -0.16864875  0.         ... -0.20247362  0.
   0.        ]
 ...
 [ 0.39154638 -0.15969785  0.         ...  2.8624603   0.
   0.        ]
 [ 0.23232492 -0.16864875  0.         ...  0.10401977  0.
   0.        ]
 [-0.80261455 -0.1731242   0.         ... -0.50896701  0.
   0.        ]]

In [10]: from sklearn.naive_bayes import GaussianNB
         classifier = GaussianNB()
         classifier.fit(X_train,y_train)

Out[10]: GaussianNB()
```


```
In [11]: y_pred=classifier.predict(X_test)
         print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

[[1 1]
 [1 0]
 [0 0]
 ...
 [1 1]
 [0 0]
 [1 0]]

In [ ]:
```



COMPARISON

<u>LOGISTIC REGRESSION</u>	<u>GAUSSIAN NAÏVE BAYES</u>
Accuracy Score: 0.94008	Accuracy Score: 0.6932
Precision Score: 0.9270038910505837	Precision Score: 0.6302343241484053
Recall Score: 0.9550994227068633	Recall Score: 0.9316067992302758
Accuracy and Precision Scores for logistic regression are very high when compared to Gaussian Naïve Bayes and Recall Score is also slightly higher.	Accuracy and Precision scores for Gaussian Naïve Bayes algorithm is not very good for this case when compared to Logistic Regression.

Logistic Regression is a better Machine Learning algorithm to train the malware dataset in our case when compared to the Gaussian Naïve Bayes algorithm because, in almost all positive performance metrics, Logistic Regression gave better scores.