NAME: MANSI RATURI
REG. NO: 19BCE0488

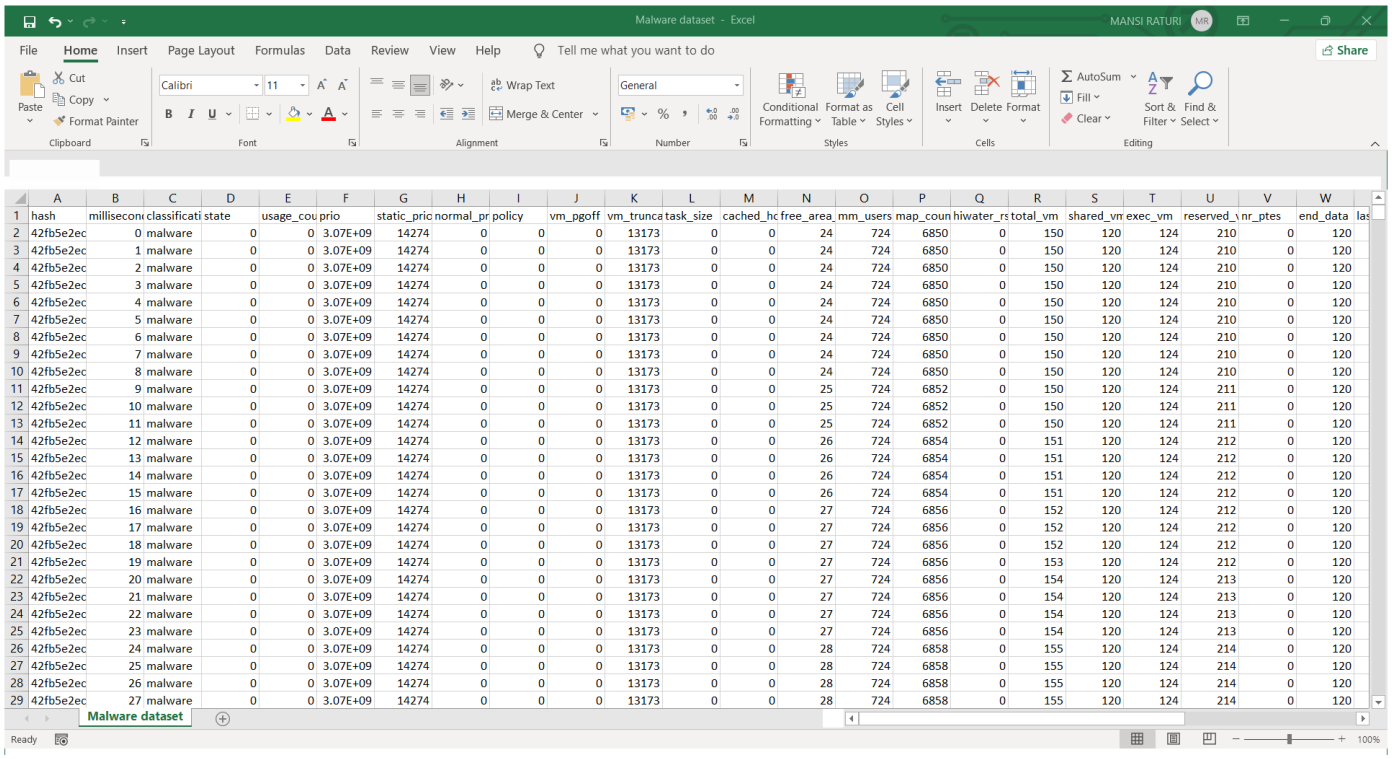
COURSE CODE: CSE3501
SLOT: L5+L6

DIGITAL ASSIGNMENT 5

MACHINE LEARNING BASED MALWARE DETECTION

Language: python
Platform: Google Colab

DATASET:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	hash	millisecon	classificati	state	usage_cou	prio	static_prio	normal_pr	policy	vm_pgoff	vm_runca	task_size	cached_hc	free_area_	mm_users	map_coun	hiwater_rs	total_vm	shared_vm	exec_vm	reserved_	nr_ptes	end_data
2	42fb5e2ec	0	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
3	42fb5e2ec	1	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
4	42fb5e2ec	2	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
5	42fb5e2ec	3	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
6	42fb5e2ec	4	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
7	42fb5e2ec	5	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
8	42fb5e2ec	6	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
9	42fb5e2ec	7	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
10	42fb5e2ec	8	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850	0	150	120	124	210	0	120
11	42fb5e2ec	9	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852	0	150	120	124	211	0	120
12	42fb5e2ec	10	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852	0	150	120	124	211	0	120
13	42fb5e2ec	11	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852	0	150	120	124	211	0	120
14	42fb5e2ec	12	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854	0	151	120	124	212	0	120
15	42fb5e2ec	13	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854	0	151	120	124	212	0	120
16	42fb5e2ec	14	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854	0	151	120	124	212	0	120
17	42fb5e2ec	15	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854	0	151	120	124	212	0	120
18	42fb5e2ec	16	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	152	120	124	212	0	120
19	42fb5e2ec	17	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	152	120	124	212	0	120
20	42fb5e2ec	18	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	152	120	124	212	0	120
21	42fb5e2ec	19	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	153	120	124	212	0	120
22	42fb5e2ec	20	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	154	120	124	213	0	120
23	42fb5e2ec	21	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	154	120	124	213	0	120
24	42fb5e2ec	22	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	154	120	124	213	0	120
25	42fb5e2ec	23	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856	0	154	120	124	213	0	120
26	42fb5e2ec	24	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	28	724	6858	0	155	120	124	214	0	120
27	42fb5e2ec	25	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	28	724	6858	0	155	120	124	214	0	120
28	42fb5e2ec	26	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	28	724	6858	0	155	120	124	214	0	120
29	42fb5e2ec	27	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	28	724	6858	0	155	120	124	214	0	120

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

#reading the dataset 'Malware detection'
dataset = pd.read_csv('Malware dataset.csv')
dataset2 = dataset.copy()
dataset2 = dataset.drop(['classification'], axis=1)
X = dataset2.iloc[:,1:].values
y = dataset.iloc[:, 2].values
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
#converting the labels into a numeric form so as to convert them into the machine-readable form
labEnc = LabelEncoder()
y = labEnc.fit_transform(y)
```

Data Splitting

```
from sklearn.model_selection import train_test_split
```

```
#Splitting the dataset into 75% training set and 25% for testing
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)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
#normalize the range of independent variables or features of data(data normalizatio
n): preprocessing of data
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)
```

#1.Training Model-Linear Regression

```
from sklearn.linear_model import LogisticRegression
#training the dataset by applying logical regression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1))
,1))

from sklearn.metrics import confusion_matrix, accuracy_score , precision_score , re
call_score , f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
as1=accuracy_score(y_test, y_pred)
ps1=precision_score(y_test, y_pred)
r1=recall_score(y_test, y_pred)
f11=f1_score(y_test, y_pred)
print("Accuracy Score :",accuracy_score(y_test, y_pred))
print("Precision Score :",precision_score(y_test, y_pred))
print("Recall Score :",recall_score(y_test, y_pred))
print("f1 Score :",f1_score(y_test, y_pred))
```

Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=['benign', 'malware']))
from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred, target_names=['benign', 'malware
'], output_dict=True)
cleaned_report={}
cleaned_report['benign']=clf_report['benign']
cleaned_report['malware']=clf_report['malware']
sns.heatmap(pd.DataFrame(cleaned_report).iloc[:-1,:],annot=True)
```

#2.Training Model-Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
#training the dataset by applying decision tree algorithm
dtree = DecisionTreeClassifier(random_state = 0)
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1))
,1))
from sklearn.metrics import confusion_matrix, accuracy_score , precision_score , re
call_score , f1_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
as2=accuracy_score(y_test, y_pred)
ps2=precision_score(y_test, y_pred)
r2=recall_score(y_test, y_pred)
f12=f1_score(y_test, y_pred)
print("Accuracy Score :",accuracy_score(y_test, y_pred))
print("Precision Score :",precision_score(y_test, y_pred))
print("Recall Score :",recall_score(y_test, y_pred))
print("f1 Score :",f1_score(y_test, y_pred))
```

Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=['benign', 'malware']))

from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred, target_names=['benign', 'malware
'], output_dict=True)
cleaned_report={}
cleaned_report['benign']=clf_report['benign']
cleaned_report['malware']=clf_report['malware']
sns.heatmap(pd.DataFrame(cleaned_report).iloc[:-1,:],annot=True)
```

Comparison

```
comp_graph_data=[['Logistic Regression',as1,ps1,r1,f11],['Decision Tree',as2,ps2,r2
,f12]]
df=pd.DataFrame(comp_graph_data, columns=['Algorithm', 'Accuracy Score', 'Precision
 Score', 'Recall Score', 'F1 Score'])
df.set_index('Algorithm', inplace=True)
df
df.plot.bar()
plt.show()
```

SCREENSHOTS

All the required libraries and dataset are imported

Then the labels are converted into numeric form by label encoding

The data is split into 75% training set and 25% testing set.

Data normalisation is done through feature scaling

19BCE0488MANSIRATURI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ 831   0   0 ...   0   0   0]]
```

[8]
```
print(y_test)
```

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

Feature Scaling

[9]
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
#normalize the range of independent variables or features of data(data normalization): preprocessing of data
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[10]
```
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.        ]]
```

1. Logical Regression

Performing dataset training and predicting the test set results.

Apps Online Courses - A... User Login | VIT eG... User defined functi... Must Solve C Progr... Adobe Illustrator C... Simple/Basic C Pro... GUVI | Course ca

19BCE0488MANSIRATURI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
   0.        ]]
```

[11]
```
print(X_test)
```

```
[[ 0.28424496 -0.1731242   0.         ...  1.94298012  0.
   0.        ]
 [-0.00650727 -0.1731242   0.         ... -0.50896701  0.
   0.        ]
 [-0.94452933  0.95021383  0.         ...  0.71700655  0.
   0.        ]
 ...
 [ 0.29462897 -0.16864875  0.         ... -0.50896701  0.
   0.        ]
 [ 0.06618079 -0.1731242   0.         ...  1.63648673  0.
   0.        ]
 [ 1.14611763 -0.1731242   0.         ... -0.50896701  0.
   0.        ]]
```

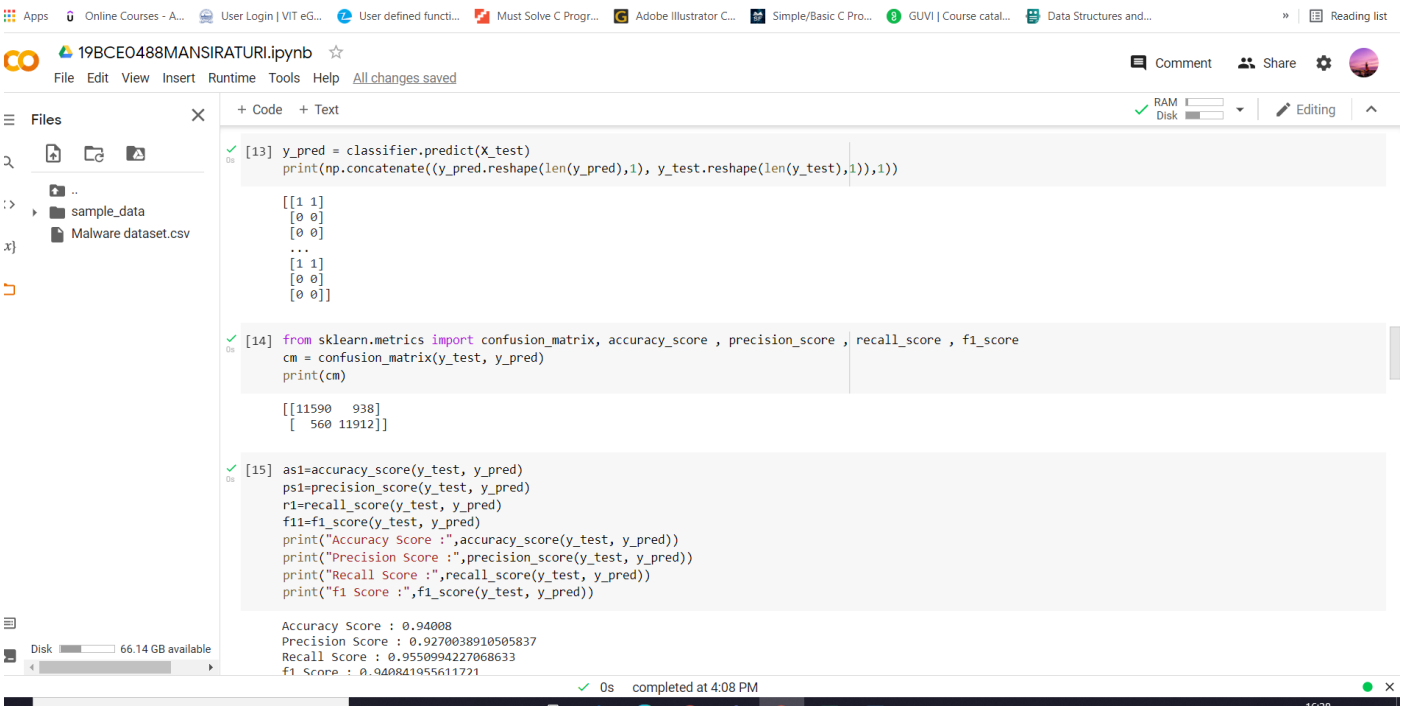
Training model-Logical Reression

[12]
```
from sklearn.linear_model import LogisticRegression
#training the dataset by applying logical regression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

0s completed at 4:08 PM

Type here to search



The model has an accuracy of 94% with 0.92 precision, 0.95 recall and 0.94 F1 score values.

Classification Report

Heat Map

19BCE0488MANSIRATURI.ipynb

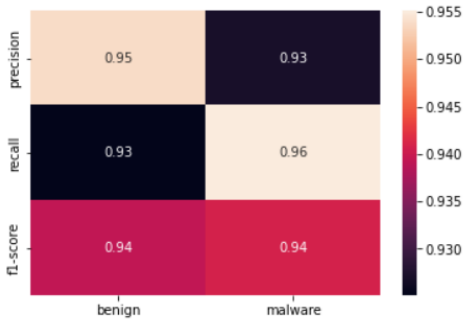
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

sample_data

Malware dataset.csv

[17] <matplotlib.axes._subplots.AxesSubplot at 0x7fbecc072550>



2. Decision Tree

Performing dataset training and predicting the test set results.

19BCE0488MANSIRATURI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Training model-Decision Tree

```
[18] from sklearn.tree import DecisionTreeClassifier
     #training the dataset by applying decision tree algorithm
     dtree = DecisionTreeClassifier(random_state = 0)
     dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

```
[19] y_pred = dtree.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]]
```

```
[20] from sklearn.metrics import confusion_matrix, accuracy_score , precision_score , recall_score , f1_score
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     as2=accuracy_score(y_test, y_pred)
     ps2=precision_score(y_test, y_pred)
     r2=recall_score(y_test, y_pred)
```

0s completed at 4:08 PM

Classification report

19BCE0488MANSIRATURI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[20] from sklearn.metrics import confusion_matrix, accuracy_score , precision_score , recall_score , f1_score
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     as2=accuracy_score(y_test, y_pred)
     ps2=precision_score(y_test, y_pred)
     r2=recall_score(y_test, y_pred)
     f12=f1_score(y_test, y_pred)
     print("Accuracy Score :",accuracy_score(y_test, y_pred))
     print("Precision Score :",precision_score(y_test, y_pred))
     print("Recall Score :",recall_score(y_test, y_pred))
     print("f1 Score :",f1_score(y_test, y_pred))
     from sklearn.metrics import classification_report
     print(classification_report(y_test, y_pred, target_names=['benign', 'malware']))
```

```
[[12528     0]
 [    1 12471]]
Accuracy Score : 0.99996
Precision Score : 1.0
Recall Score : 0.9999198203976908
f1 Score : 0.9999599085915888
              precision    recall  f1-score   support

      benign       1.00      1.00      1.00     12528
     malware       1.00      1.00      1.00     12472

    accuracy                           1.00     25000
   macro avg       1.00      1.00      1.00     25000
weighted avg       1.00      1.00      1.00     25000
```

The model has an accuracy of 99% with 1.00 precision, 0.99 recall and 0.99 F1 score values.

Heat Map

19BCE0488MANSIRATURI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[21] from sklearn.metrics import classification_report
     clf_report = classification_report(y_test, y_pred, target_names=['benign', 'malware'], output_dict=True)
     cleaned_report={}
     cleaned_report['benign']=clf_report['benign']
     cleaned_report['malware']=clf_report['malware']
     sns.heatmap(pd.DataFrame(cleaned_report).iloc[:-1,:],annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbecaa392d0>
```


```
[22] comp_graph_data=[['Logistic Regression',as1,ps1,r1,f11],['Decision Tree',as2,ps2,r2,f12]]
     df=pd.DataFrame(comp_graph_data, columns=['Algorithm', 'Accuracy Score', 'Precision Score', 'Recall Score', 'F1 Score'])
     df.set_index('Algorithm', inplace=True)
     df
```

Comparison

The two models Logistic Regression Model and Decision Tree Model are compared for the given data set on the basis of Accuracy, Precision, Recall and F1 Score.

Algorithm	Accuracy Score	Precision Score	Recall Score	F1 Score
Logistic Regression	0.94008	0.927004	0.955099	0.940842
Decision Tree	0.99996	1.000000	0.999920	0.999960

RESULT: Logistic Regression model has an accuracy of 94.008 percentage whereas Decision Tree has an accuracy of 99.996 percentage. The reason of such high accuracy of Decision Tree accuracy may be over-fitting of data