Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Name – DIVYESH KKUNT **SAP – 60009210116**

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

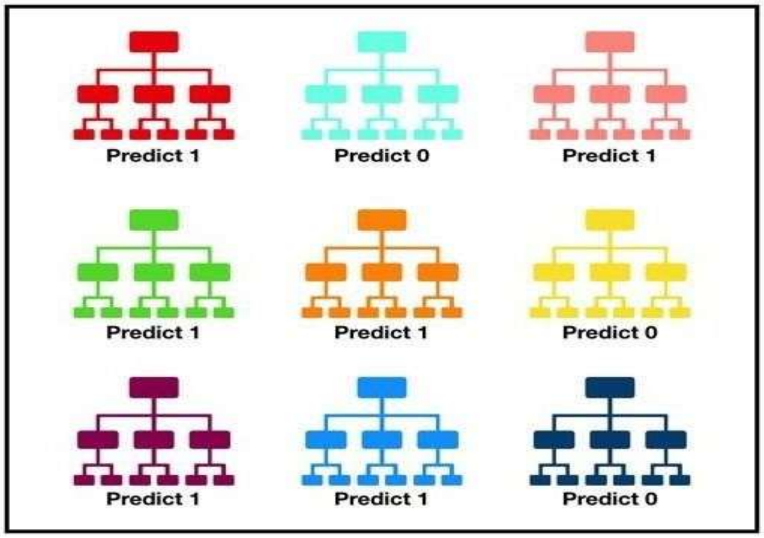
Experiment 6

(Random Forest)

Aim: Implement Random Forest algorithm on given datasets and compare the results with Decision Tree classifiers for the same datasets.

Theory:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).



Tally: Six 1s and Three 0s
Prediction: 1

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each**

other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

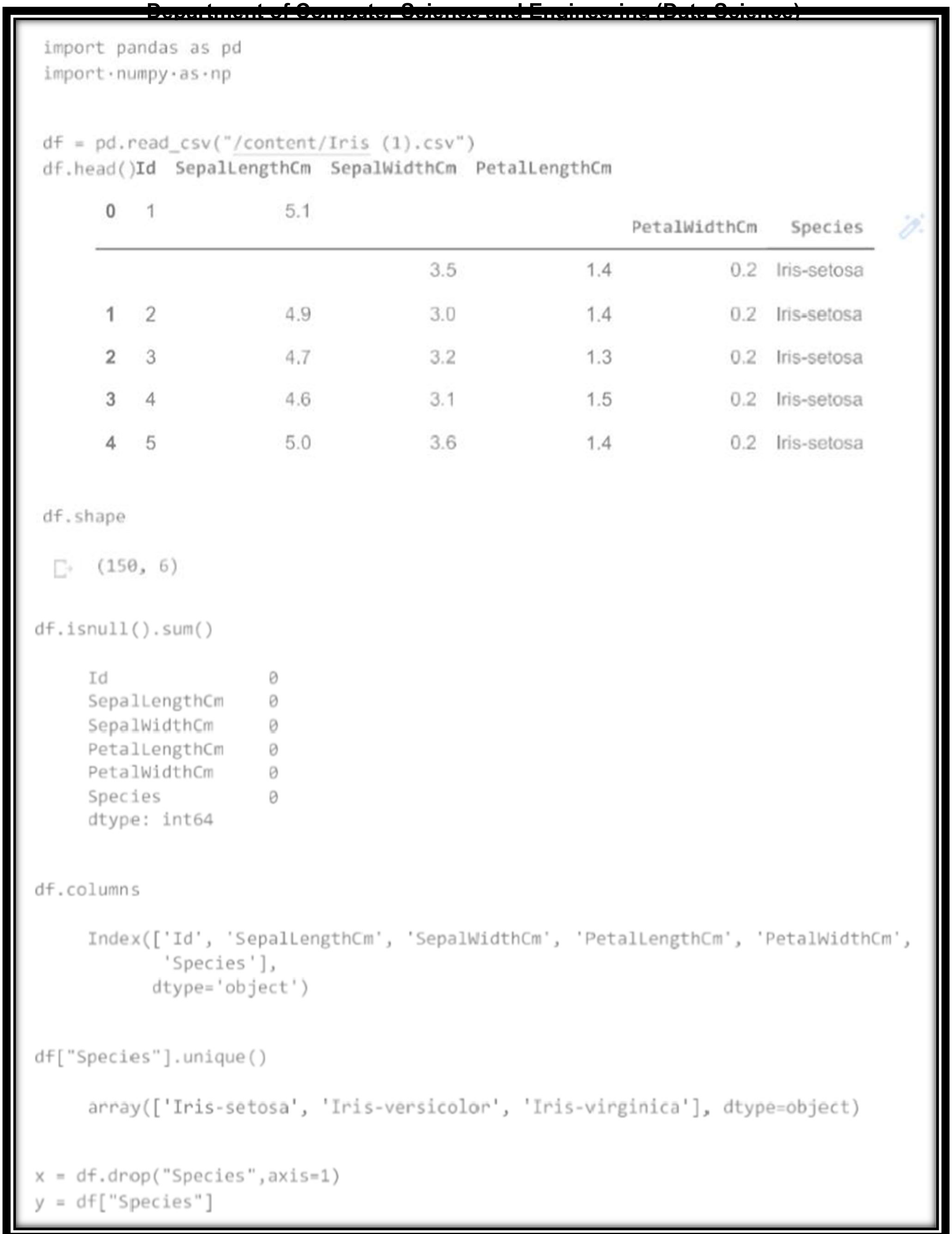
1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:
Dataset 1: IRIS.csv
Dataset 2: BehaviouralRskFactorSurvillanceSystem.csv (The objective of the BRFSS is to collect uniform, state-specific data on preventive health practices and risk behaviors that are linked to chronic diseases, injuries, and preventable infectious diseases in the adult population. Factors assessed by the BRFSS include tobacco use, health care coverage, HIV/AIDS knowledge or prevention, physical activity, and fruit and vegetable consumption. Data are collected from a random sample of adults (one per household) through a telephone survey. The Behavioral Risk Factor Surveillance System (BRFSS) is the nation's premier system of health-related telephone surveys that collect state data about U.S. residents regarding their health-related risk behaviors, chronic health conditions, and use of preventive services. Established in 1984 with 15 states, BRFSS now collects data in all 50 states as well as the District of Columbia and three U.S. territories. BRFSS completes more than 400,000 adult interviews each year, making it the largest continuously conducted health survey system in the world.)

1. Compare the results of decision tree and random forest classifier for dataset 1 and 2.
2. Compare the results of random forest with and without selecting important features only for buildingthe classifier on dataset 2 and 3.



```
import pandas as pd
import·numpy·as·np
```

```
df = pd.read_csv("/content/Iris (1).csv")
df.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm
0   1            5.1
                                                  PetalWidthCm   Species
                              3.5           1.4            0.2   Iris-setosa
1   2            4.9          3.0           1.4            0.2   Iris-setosa
2   3            4.7          3.2           1.3            0.2   Iris-setosa
3   4            4.6          3.1           1.5            0.2   Iris-setosa
4   5            5.0          3.6           1.4            0.2   Iris-setosa
```

```
df.shape
```

```
(150, 6)
```

```
df.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

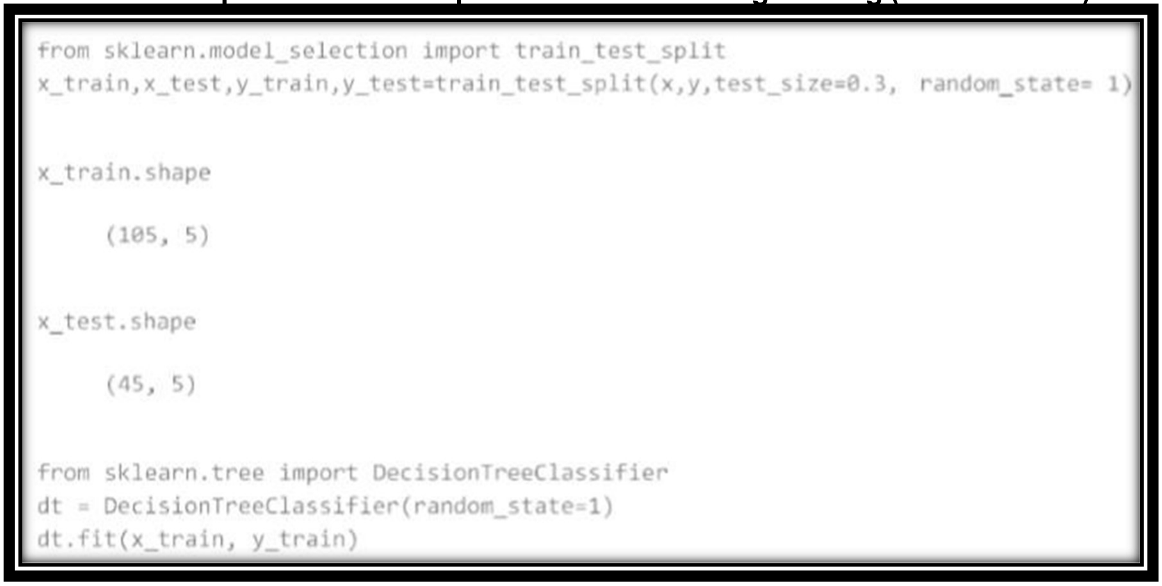
```
df.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

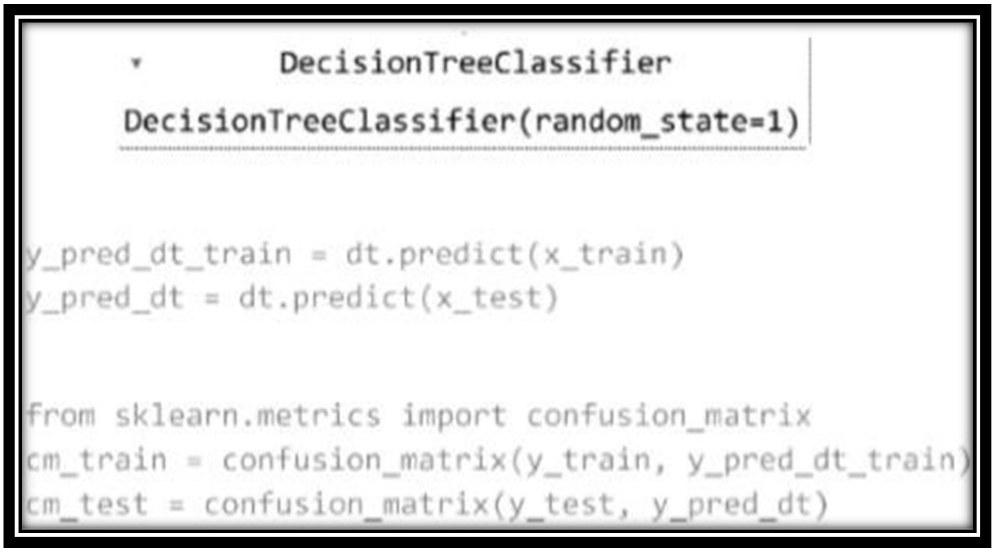
```
df["Species"].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

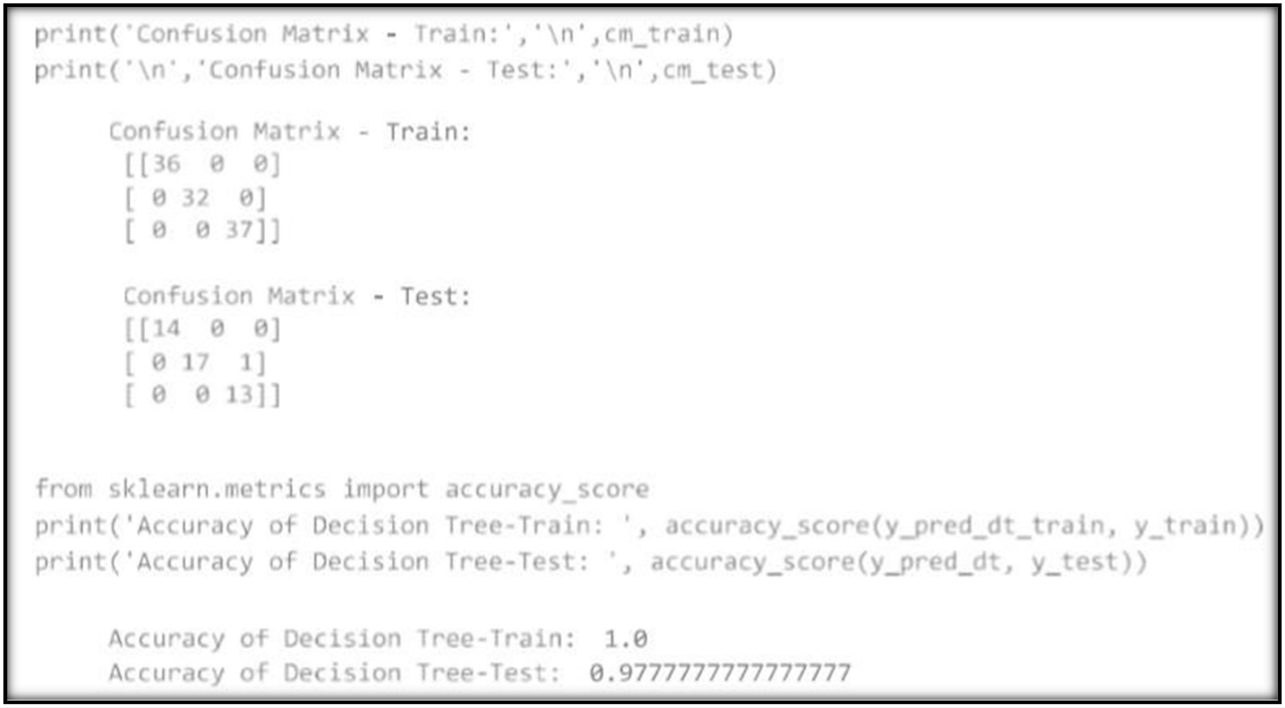
```
x = df.drop("Species",axis=1)
y = df["Species"]
```



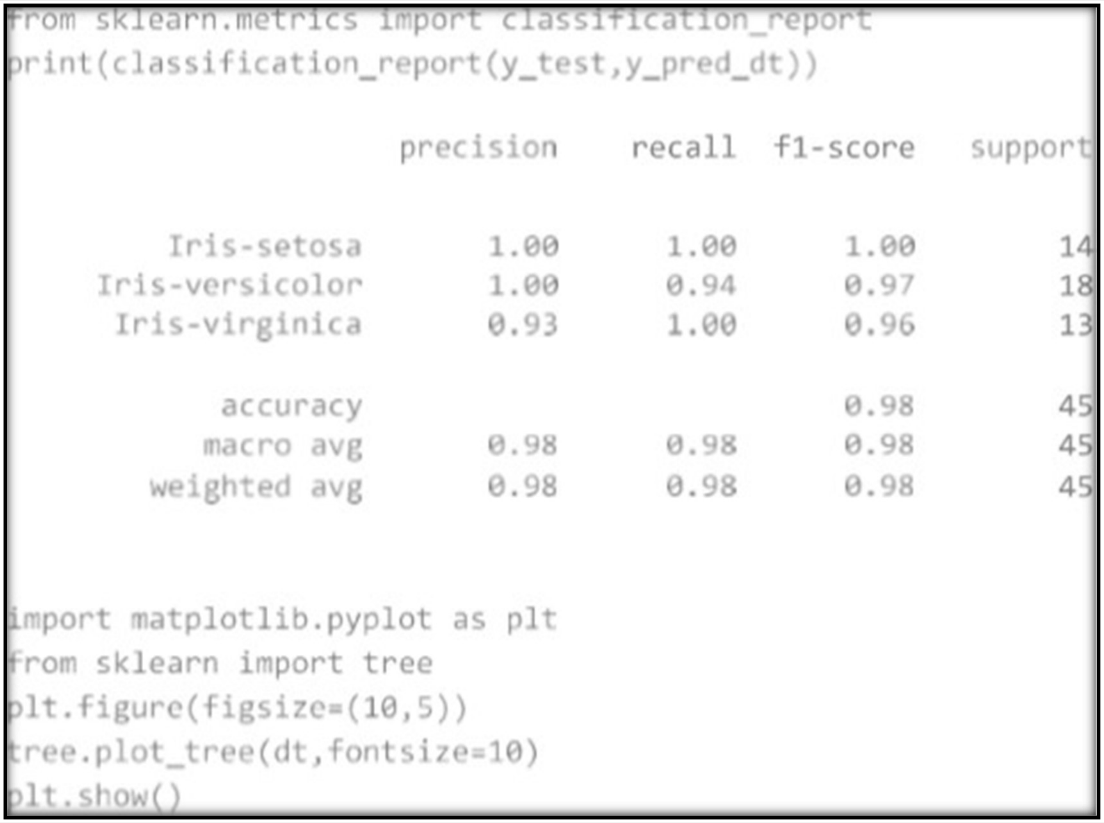
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,  random_state= 1)


x_train.shape

     (105, 5)


x_test.shape

     (45, 5)


from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=1)
dt.fit(x_train, y_train)
```



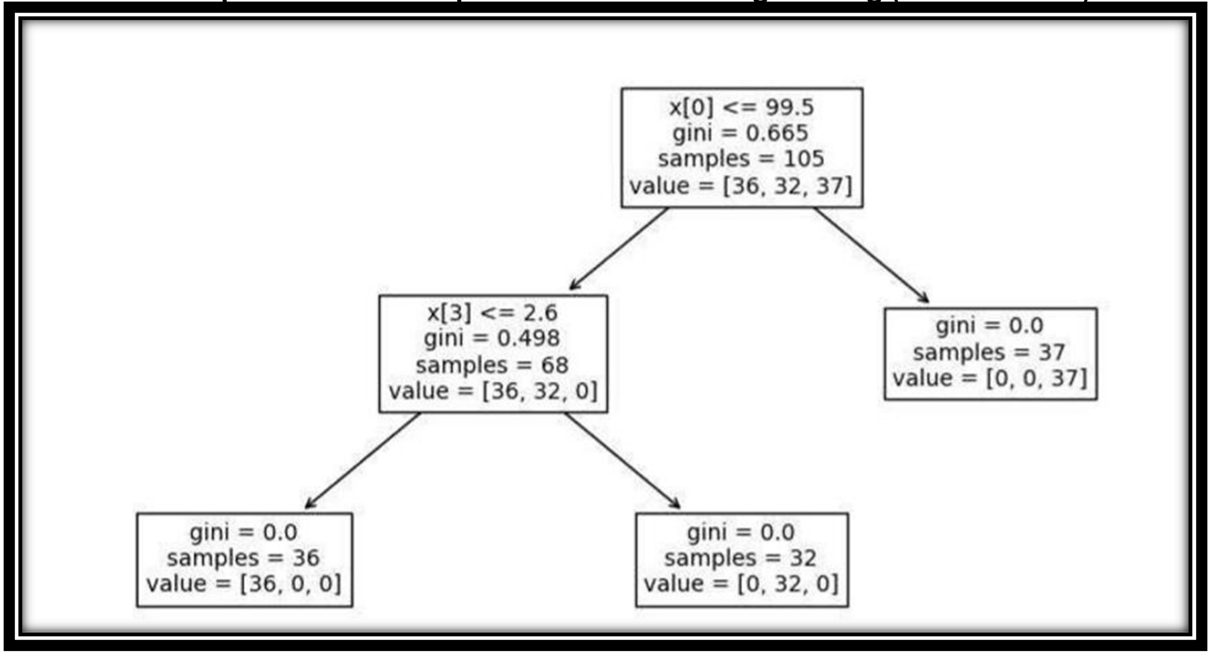
```
     ▾         DecisionTreeClassifier
     DecisionTreeClassifier(random_state=1)


y_pred_dt_train = dt.predict(x_train)
y_pred_dt = dt.predict(x_test)


from sklearn.metrics import confusion_matrix
cm_train = confusion_matrix(y_train, y_pred_dt_train)
cm_test = confusion_matrix(y_test, y_pred_dt)
```

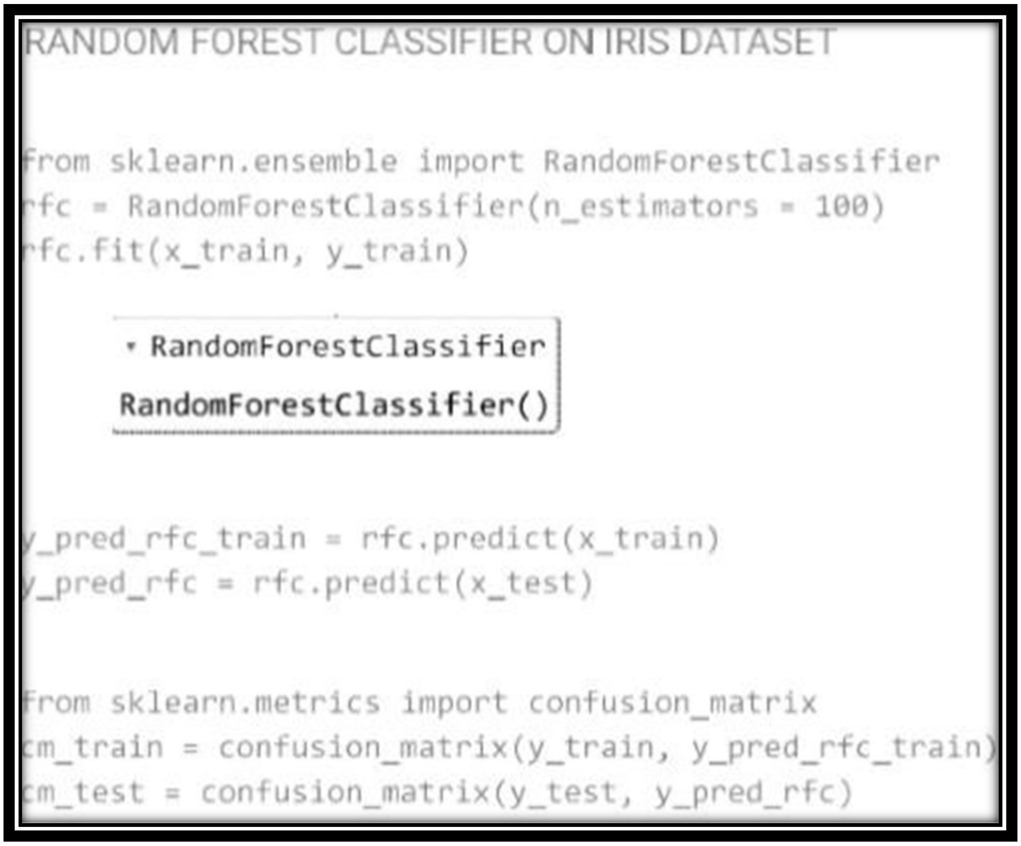


```
print('Confusion Matrix - Train:','\n',cm_train)
print('\n','Confusion Matrix - Test:','\n',cm_test)

    Confusion Matrix - Train:
     [[36  0  0]
     [ 0 32  0]
     [ 0  0 37]]

     Confusion Matrix - Test:
     [[14  0  0]
     [ 0 17  1]
     [ 0  0 13]]


from sklearn.metrics import accuracy_score
print('Accuracy of Decision Tree-Train: ', accuracy_score(y_pred_dt_train, y_train))
print('Accuracy of Decision Tree-Test: ', accuracy_score(y_pred_dt, y_test))

    Accuracy of Decision Tree-Train:  1.0
    Accuracy of Decision Tree-Test:  0.9777777777777777
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_dt))

                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        18
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45


import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(10,5))
tree.plot_tree(dt,fontsize=10)
plt.show()
```





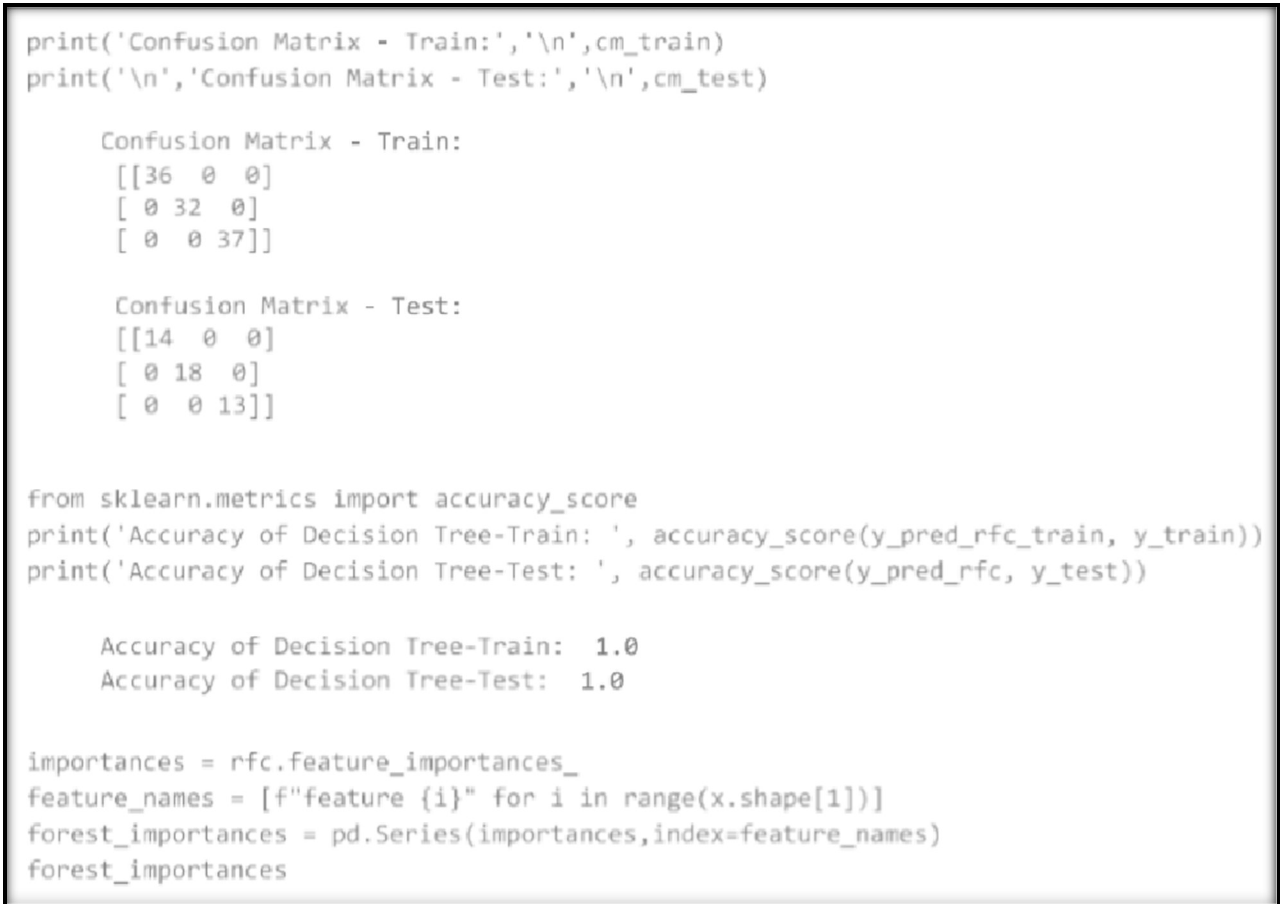
RANDOM FOREST CLASSIFIER ON IRIS DATASET

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 100)
rfc.fit(x_train, y_train)
```

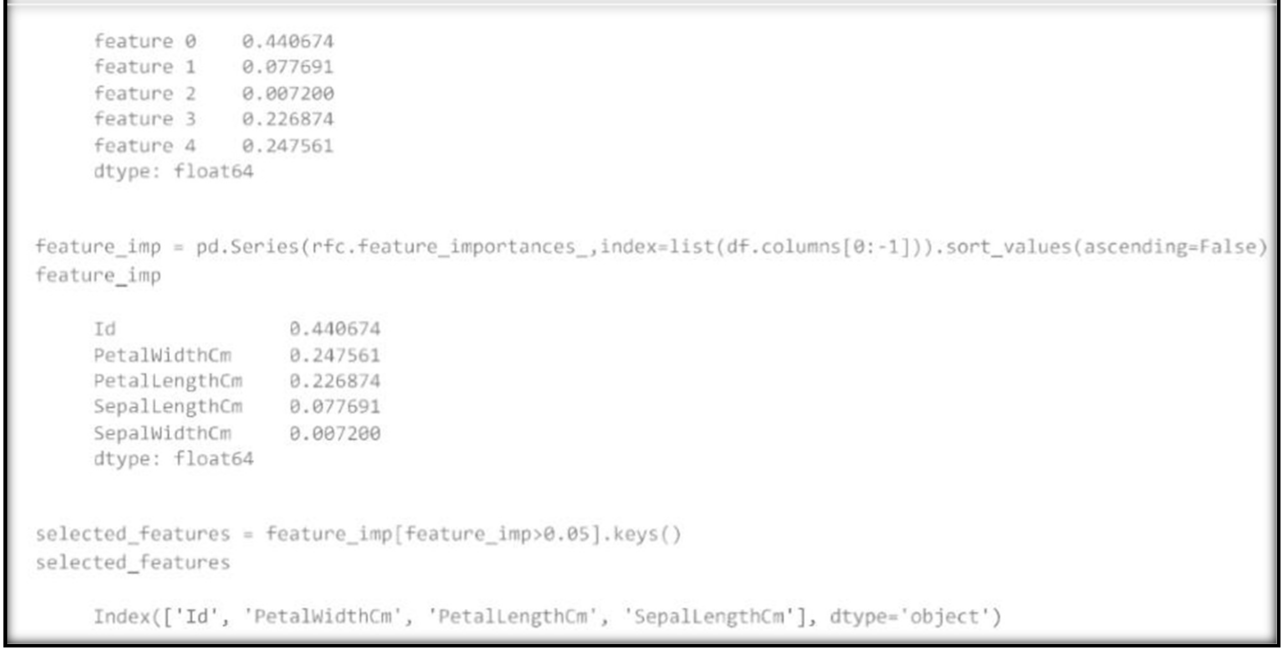
```
    ▾ RandomForestClassifier
    RandomForestClassifier()
```

```
y_pred_rfc_train = rfc.predict(x_train)
y_pred_rfc = rfc.predict(x_test)
```

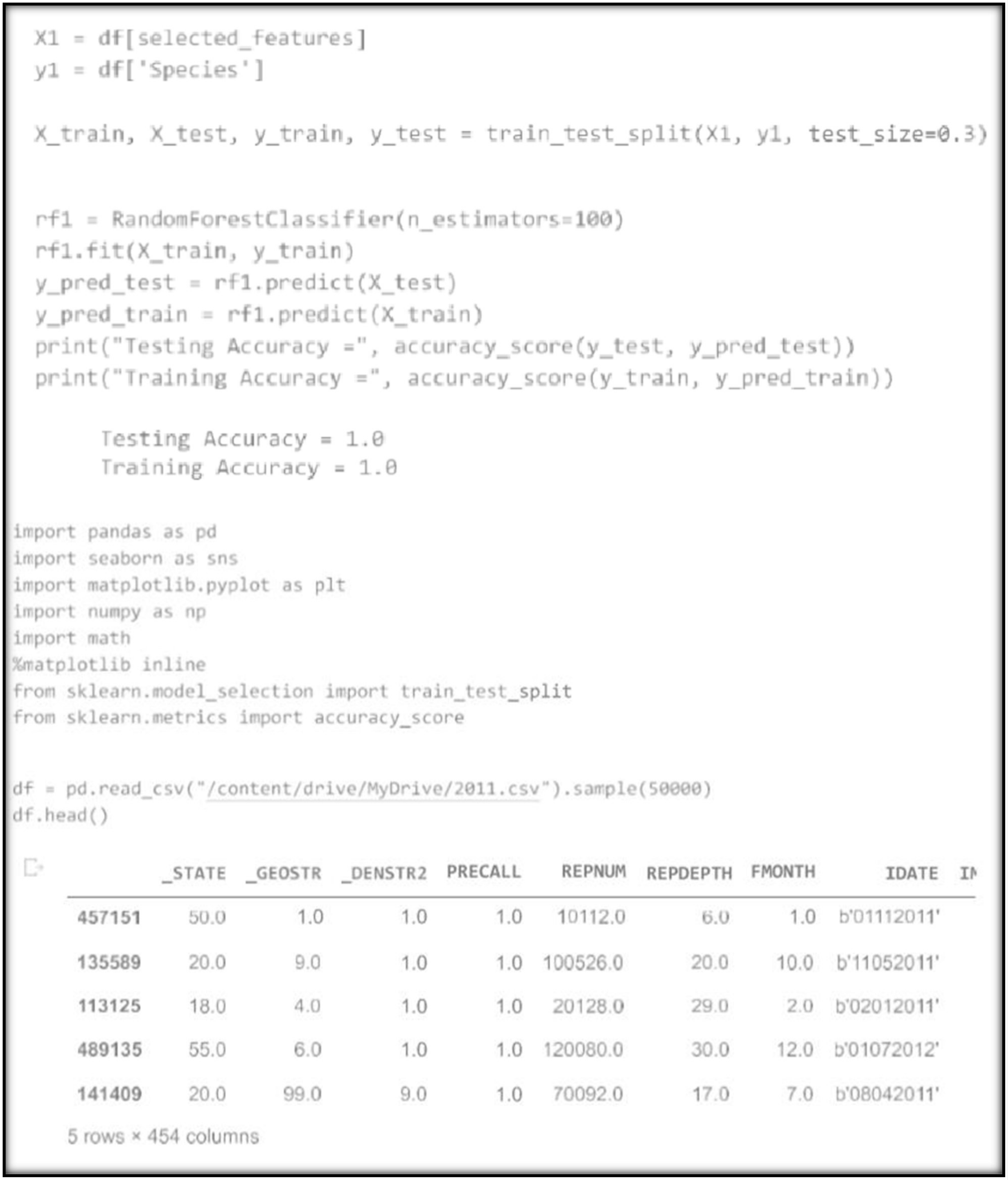
```
from sklearn.metrics import confusion_matrix
cm_train = confusion_matrix(y_train, y_pred_rfc_train)
cm_test = confusion_matrix(y_test, y_pred_rfc)
```



```
print('Confusion Matrix - Train:','\n',cm_train)
print('\n','Confusion Matrix - Test:','\n',cm_test)

    Confusion Matrix - Train:
     [[36  0  0]
     [ 0 32  0]
     [ 0  0 37]]

     Confusion Matrix - Test:
     [[14  0  0]
     [ 0 18  0]
     [ 0  0 13]]


from sklearn.metrics import accuracy_score
print('Accuracy of Decision Tree-Train: ', accuracy_score(y_pred_rfc_train, y_train))
print('Accuracy of Decision Tree-Test: ', accuracy_score(y_pred_rfc, y_test))

    Accuracy of Decision Tree-Train:  1.0
    Accuracy of Decision Tree-Test:  1.0

importances = rfc.feature_importances_
feature_names = [f"feature {i}" for i in range(x.shape[1])]
forest_importances = pd.Series(importances,index=feature_names)
forest_importances
```



```
    feature 0    0.440674
    feature 1    0.077691
    feature 2    0.007200
    feature 3    0.226874
    feature 4    0.247561
    dtype: float64


feature_imp = pd.Series(rfc.feature_importances_,index=list(df.columns[0:-1])).sort_values(ascending=False)
feature_imp

    Id               0.440674
    PetalWidthCm     0.247561
    PetalLengthCm    0.226874
    SepalLengthCm    0.077691
    SepalWidthCm     0.007200
    dtype: float64


selected_features = feature_imp[feature_imp>0.05].keys()
selected_features

    Index(['Id', 'PetalWidthCm', 'PetalLengthCm', 'SepalLengthCm'], dtype='object')
```



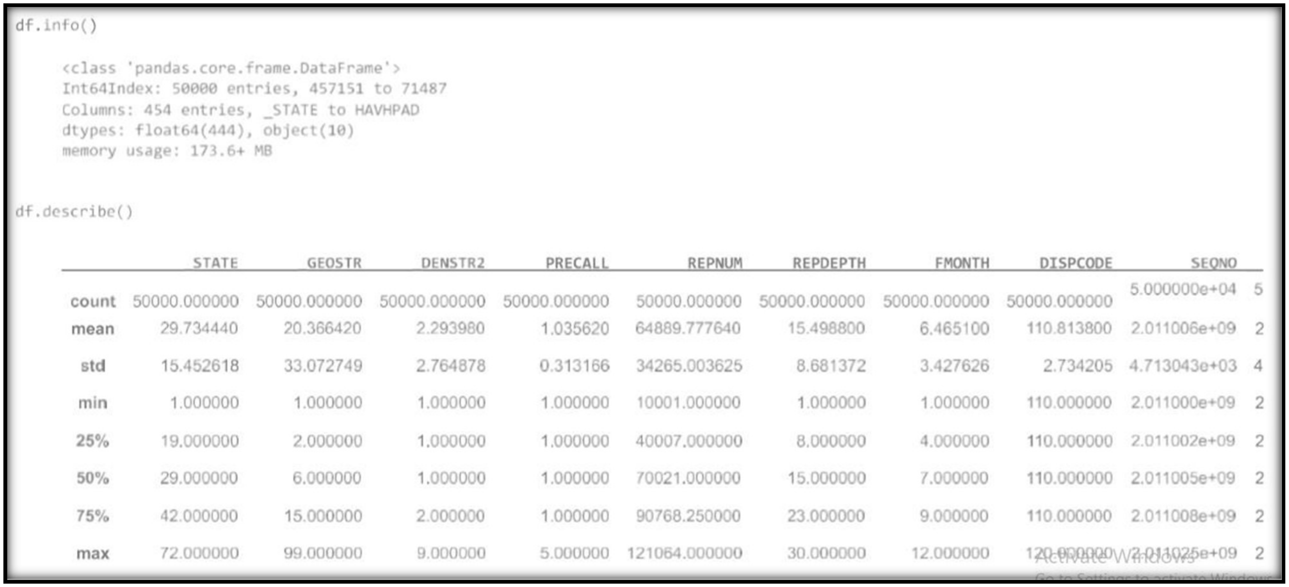
```
X1 = df[selected_features]
y1 = df['Species']

X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3)


rf1 = RandomForestClassifier(n_estimators=100)
rf1.fit(X_train, y_train)
y_pred_test = rf1.predict(X_test)
y_pred_train = rf1.predict(X_train)
print("Testing Accuracy =", accuracy_score(y_test, y_pred_test))
print("Training Accuracy =", accuracy_score(y_train, y_pred_train))
```

```
    Testing Accuracy = 1.0
    Training Accuracy = 1.0
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import math
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


df = pd.read_csv("/content/drive/MyDrive/2011.csv").sample(50000)
df.head()
```

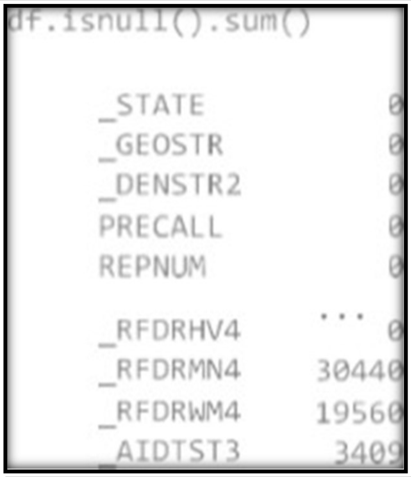
	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	IDATE	I
457151	50.0	1.0	1.0	1.0	10112.0	6.0	1.0	b'01112011'	
135589	20.0	9.0	1.0	1.0	100526.0	20.0	10.0	b'11052011'	
113125	18.0	4.0	1.0	1.0	20128.0	29.0	2.0	b'02012011'	
489135	55.0	6.0	1.0	1.0	120080.0	30.0	12.0	b'01072012'	
141409	20.0	99.0	9.0	1.0	70092.0	17.0	7.0	b'08042011'	

5 rows × 454 columns

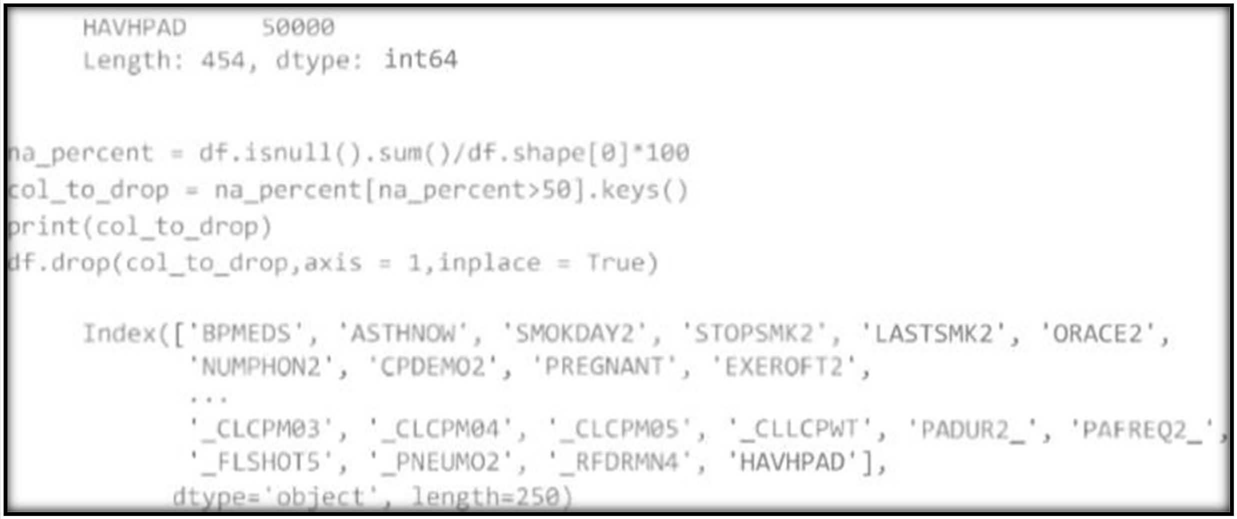


```
df.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 50000 entries, 457151 to 71487
    Columns: 454 entries, _STATE to HAVHPAD
    dtypes: float64(444), object(10)
    memory usage: 173.6+ MB


df.describe()
```

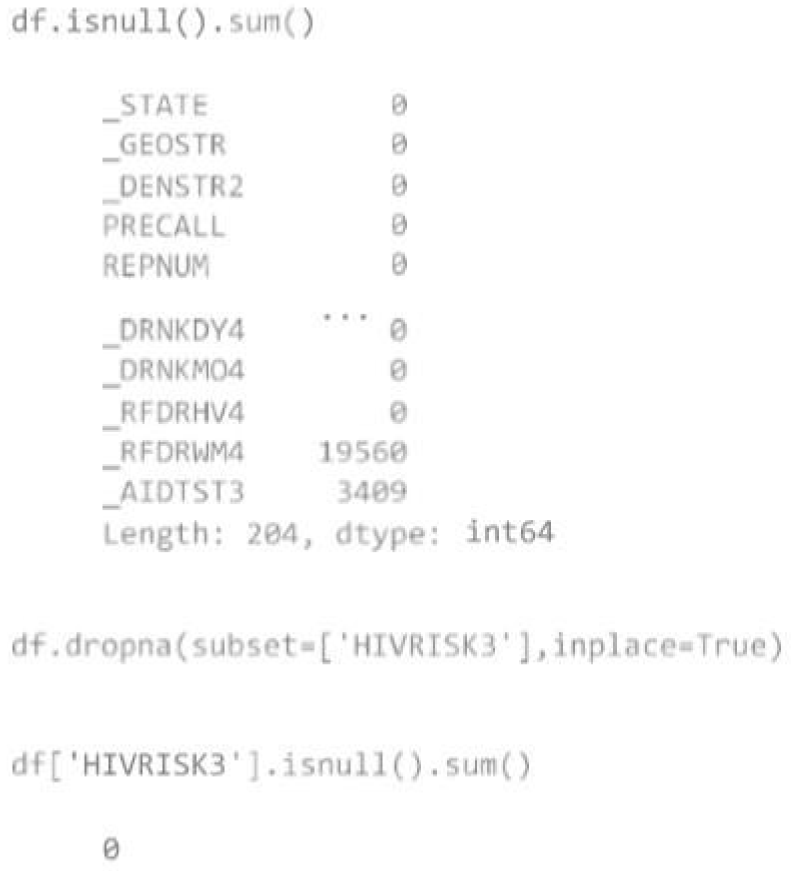
	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	DISPCODE	SEQNO	
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	5.000000e+04	5
mean	29.734440	20.366420	2.293980	1.035620	64889.777640	15.498800	6.465100	110.813800	2.011006e+09	2
std	15.452618	33.072749	2.764878	0.313166	34265.003625	8.681372	3.427626	2.734205	4.713043e+03	4
min	1.000000	1.000000	1.000000	1.000000	10001.000000	1.000000	1.000000	110.000000	2.011000e+09	2
25%	19.000000	2.000000	1.000000	1.000000	40007.000000	8.000000	4.000000	110.000000	2.011002e+09	2
50%	29.000000	6.000000	1.000000	1.000000	70021.000000	15.000000	7.000000	110.000000	2.011005e+09	2
75%	42.000000	15.000000	2.000000	1.000000	90768.250000	23.000000	9.000000	110.000000	2.011008e+09	2
max	72.000000	99.000000	9.000000	5.000000	121064.000000	30.000000	12.000000	120.000000	2.011025e+09	2



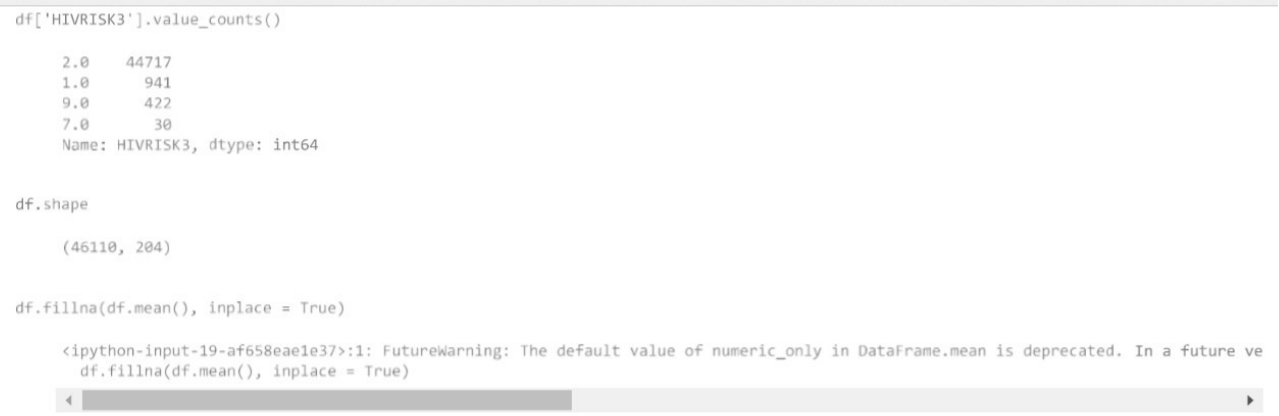
```
df.isnull().sum()

    _STATE          0
    _GEOSTR         0
    _DENSTR2        0
    PRECALL         0
    REPNUM          0
                  ...
    _RFDRHV4        0
    _RFDRMN4    30440
    _RFDRWM4    19560
    _AIDTST3     3409
```



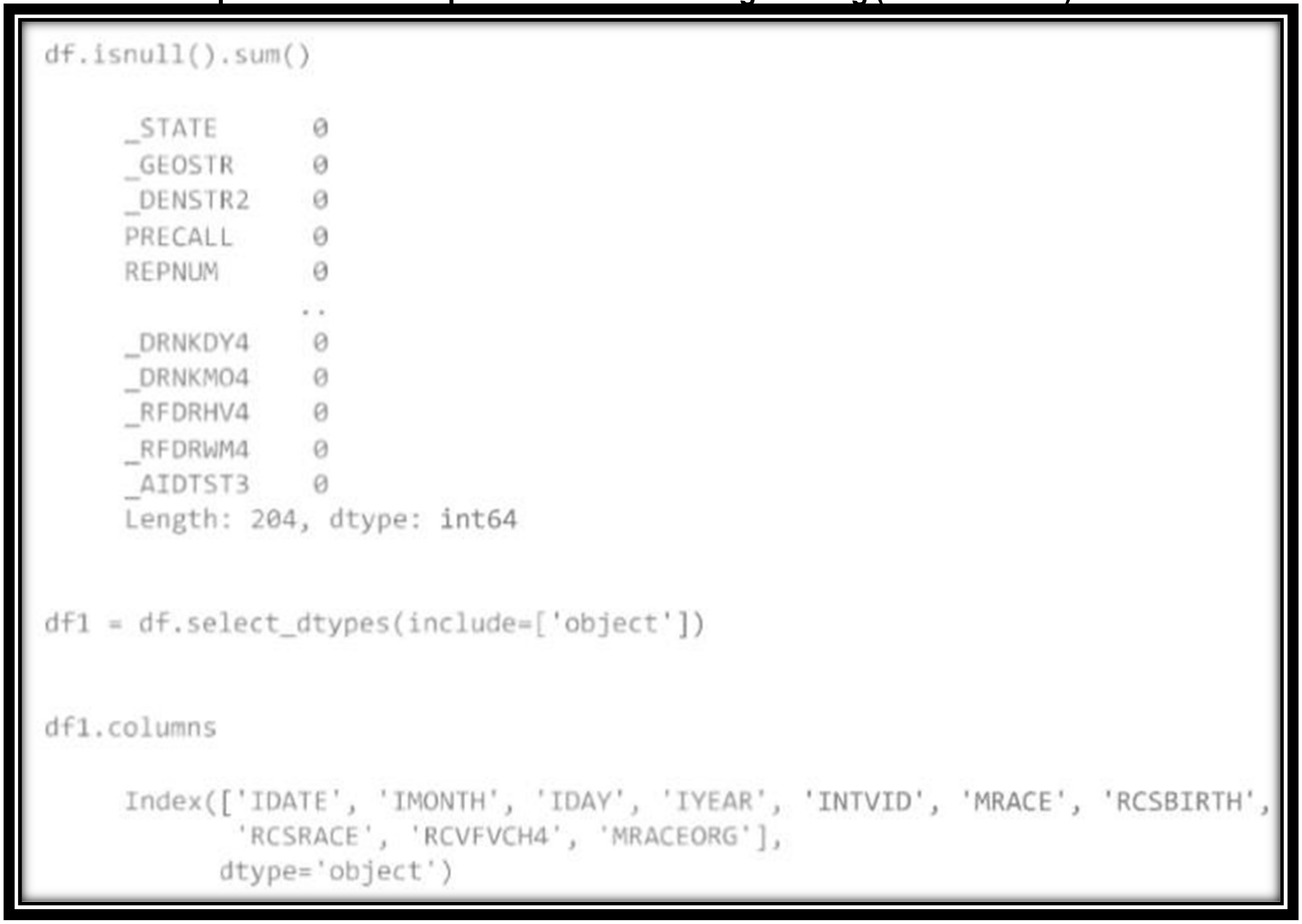
```
    HAVHPAD     50000
    Length: 454, dtype: int64


na_percent = df.isnull().sum()/df.shape[0]*100
col_to_drop = na_percent[na_percent>50].keys()
print(col_to_drop)
df.drop(col_to_drop,axis = 1,inplace = True)

    Index(['BPMEDS', 'ASTHNOW', 'SMOKDAY2', 'STOPSMK2', 'LASTSMK2', 'ORACE2',
           'NUMPHON2', 'CPDEMO2', 'PREGNANT', 'EXEROFT2',
           ...
           '_CLCPM03', '_CLCPM04', '_CLCPM05', '_CLLCPWT', 'PADUR2_', 'PAFREQ2_',
           '_FLSHOT5', '_PNEUMO2', '_RFDRMN4', 'HAVHPAD'],
          dtype='object', length=250)
```



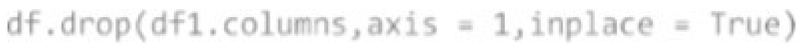
```
df.isnull().sum()

    _STATE          0
    _GEOSTR         0
    _DENSTR2        0
    PRECALL         0
    REPNUM          0
                  ...
    _DRNKDY4        0
    _DRNKMO4        0
    _RFDRHV4        0
    _RFDRWM4    19560
    _AIDTST3     3409
    Length: 204, dtype: int64


df.dropna(subset=['HIVRISK3'],inplace=True)


df['HIVRISK3'].isnull().sum()

    0
```



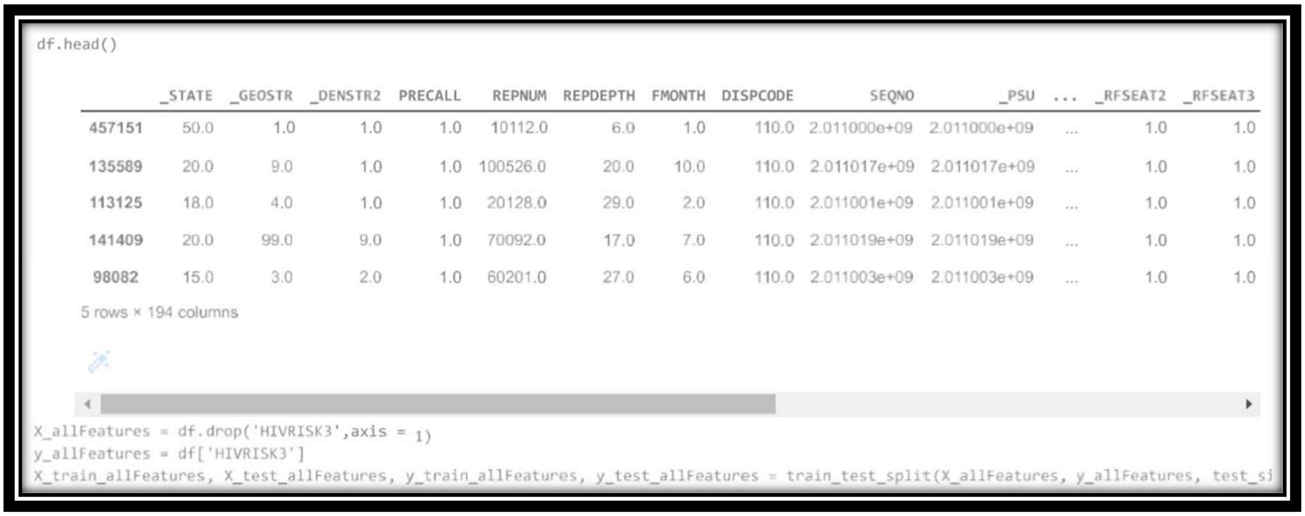
```
df['HIVRISK3'].value_counts()

    2.0    44717
    1.0      941
    9.0      422
    7.0       30
    Name: HIVRISK3, dtype: int64


df.shape

    (46110, 204)


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

    <ipython-input-19-af658eae1e37>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future ve
      df.fillna(df.mean(), inplace = True)
```



```
df.isnull().sum()

     _STATE       0
     _GEOSTR      0
     _DENSTR2     0
     PRECALL      0
     REPNUM       0
                 ..
     _DRNKDY4     0
     _DRNKMO4     0
     _RFDRHV4     0
     _RFDRWM4     0
     _AIDTST3     0
     Length: 204, dtype: int64


df1 = df.select_dtypes(include=['object'])


df1.columns

     Index(['IDATE', 'IMONTH', 'IDAY', 'IYEAR', 'INTVID', 'MRACE', 'RCSBIRTH',
            'RCSRACE', 'RCVFVCH4', 'MRACEORG'],
           dtype='object')
```



```
df.drop(df1.columns,axis = 1,inplace = True)
```

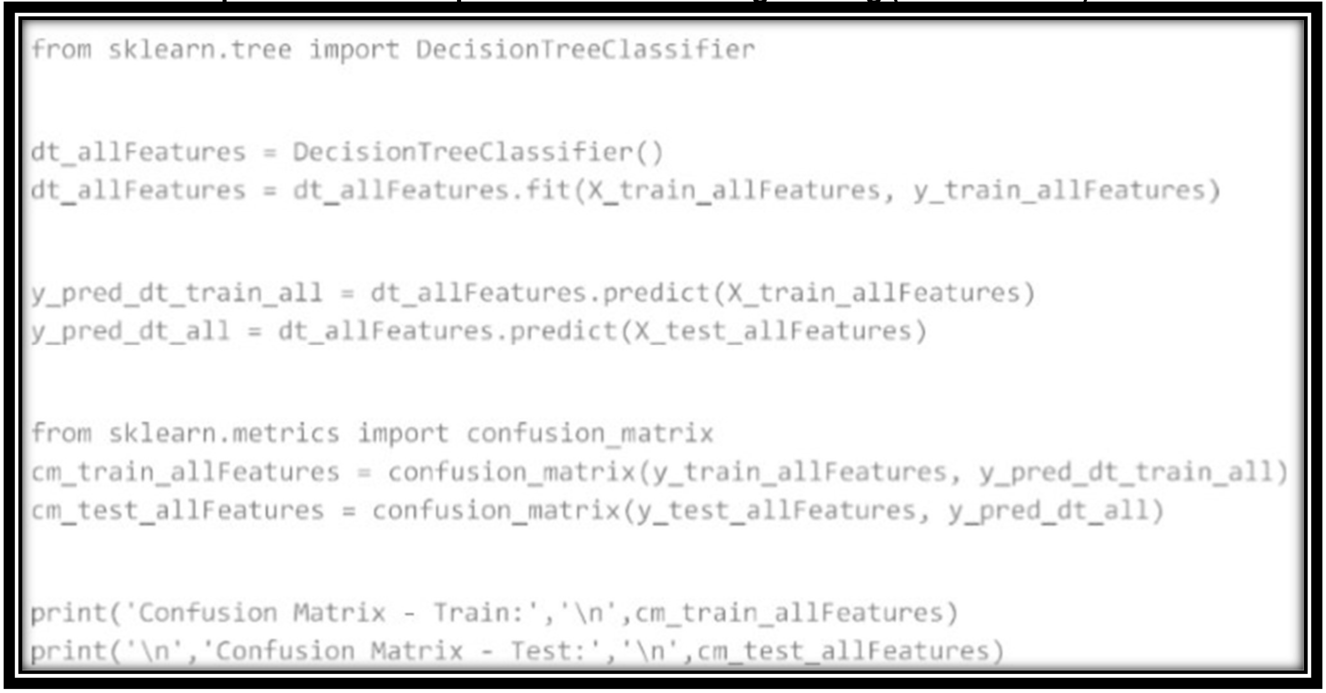


```
df.head()
```

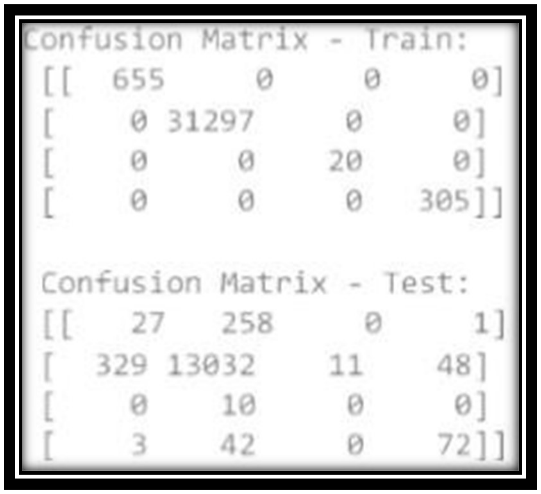
	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	DISPCODE	SEQNO	_PSU	...	_RFSEAT2	_RFSEAT3
457151	50.0	1.0	1.0	1.0	10112.0	6.0	1.0	110.0	2.011000e+09	2.011000e+09	...	1.0	1.0
135589	20.0	9.0	1.0	1.0	100526.0	20.0	10.0	110.0	2.011017e+09	2.011017e+09	...	1.0	1.0
113125	18.0	4.0	1.0	1.0	20128.0	29.0	2.0	110.0	2.011001e+09	2.011001e+09	...	1.0	1.0
141409	20.0	99.0	9.0	1.0	70092.0	17.0	7.0	110.0	2.011019e+09	2.011019e+09	...	1.0	1.0
98082	15.0	3.0	2.0	1.0	60201.0	27.0	6.0	110.0	2.011003e+09	2.011003e+09	...	1.0	1.0

5 rows × 194 columns

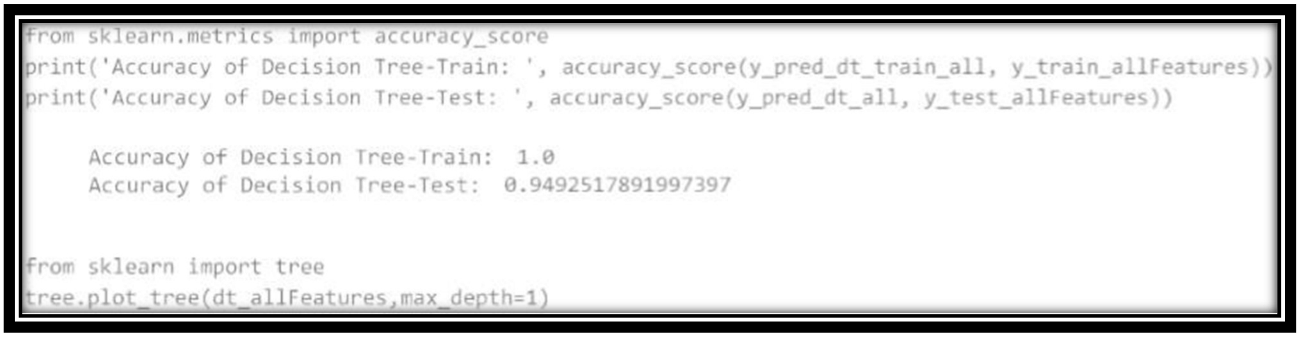
```
X_allFeatures = df.drop('HIVRISK3',axis = 1)
y_allFeatures = df['HIVRISK3']
X_train_allFeatures, X_test_allFeatures, y_train_allFeatures, y_test_allFeatures = train_test_split(X_allFeatures, y_allFeatures, test_si
```



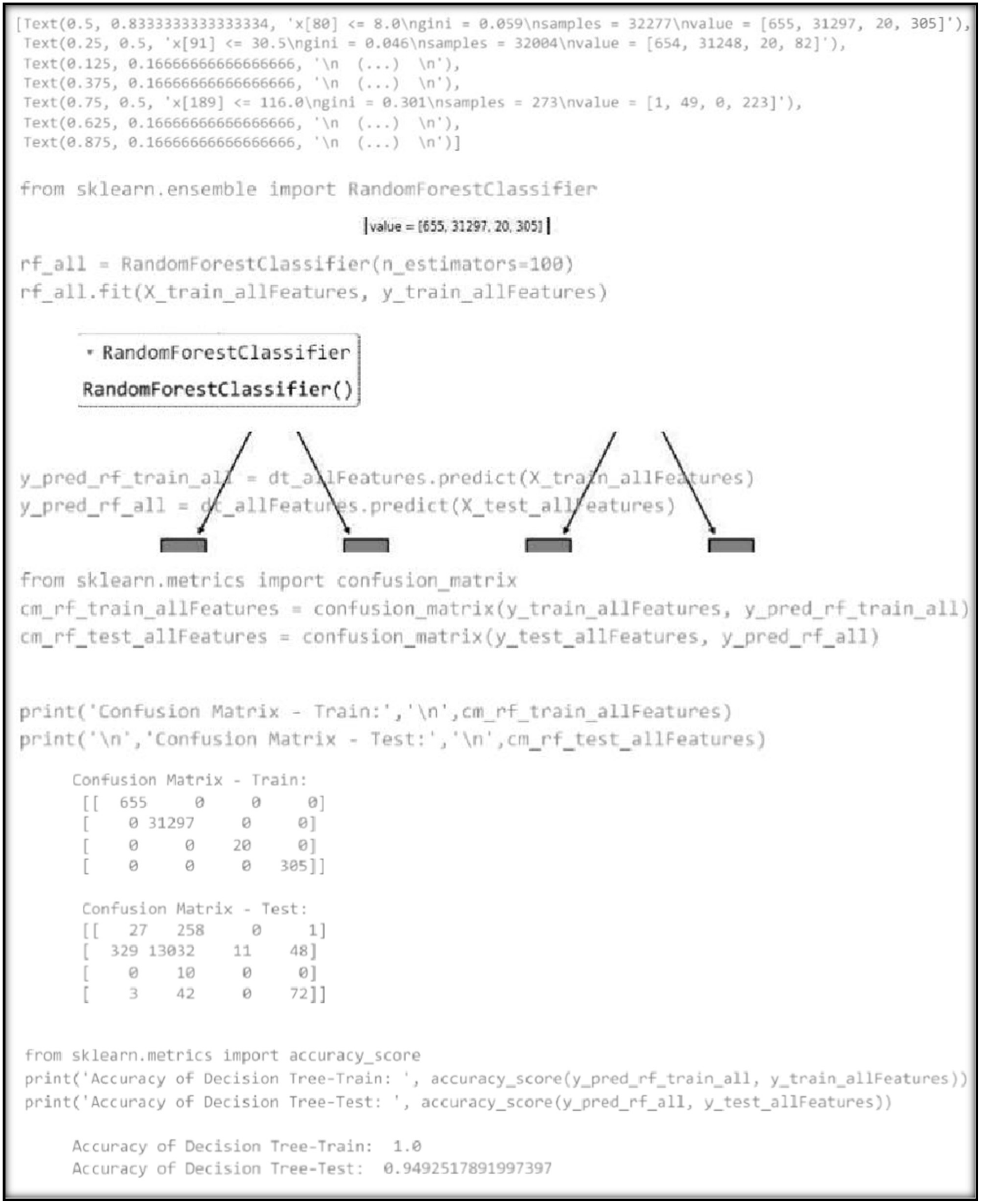
```
from sklearn.tree import DecisionTreeClassifier


dt_allFeatures = DecisionTreeClassifier()
dt_allFeatures = dt_allFeatures.fit(X_train_allFeatures, y_train_allFeatures)


y_pred_dt_train_all = dt_allFeatures.predict(X_train_allFeatures)
y_pred_dt_all = dt_allFeatures.predict(X_test_allFeatures)


from sklearn.metrics import confusion_matrix
cm_train_allFeatures = confusion_matrix(y_train_allFeatures, y_pred_dt_train_all)
cm_test_allFeatures = confusion_matrix(y_test_allFeatures, y_pred_dt_all)


print('Confusion Matrix - Train:','\n',cm_train_allFeatures)
print('\n','Confusion Matrix - Test:','\n',cm_test_allFeatures)
```



```
Confusion Matrix - Train:
 [[  655      0      0      0]
 [     0  31297      0      0]
 [     0      0     20      0]
 [     0      0      0    305]]

 Confusion Matrix - Test:
 [[    27    258      0      1]
 [   329  13032     11     48]
 [     0     10      0      0]
 [     3     42      0     72]]
```



```
from sklearn.metrics import accuracy_score
print('Accuracy of Decision Tree-Train: ', accuracy_score(y_pred_dt_train_all, y_train_allFeatures))
print('Accuracy of Decision Tree-Test: ', accuracy_score(y_pred_dt_all, y_test_allFeatures))

    Accuracy of Decision Tree-Train:  1.0
    Accuracy of Decision Tree-Test:  0.9492517891997397


from sklearn import tree
tree.plot_tree(dt_allFeatures,max_depth=1)
```



```
[Text(0.5, 0.8333333333333334, 'x[80] <= 8.0\ngini = 0.059\nsamples = 32277\nvalue = [655, 31297, 20, 305]'),
 Text(0.25, 0.5, 'x[91] <= 30.5\ngini = 0.046\nsamples = 32004\nvalue = [654, 31248, 20, 82]'),
 Text(0.125, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.375, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.75, 0.5, 'x[189] <= 116.0\ngini = 0.301\nsamples = 273\nvalue = [1, 49, 0, 223]'),
 Text(0.625, 0.16666666666666666, '\n  (...)  \n'),
 Text(0.875, 0.16666666666666666, '\n  (...)  \n')]
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf_all = RandomForestClassifier(n_estimators=100)
rf_all.fit(X_train_allFeatures, y_train_allFeatures)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

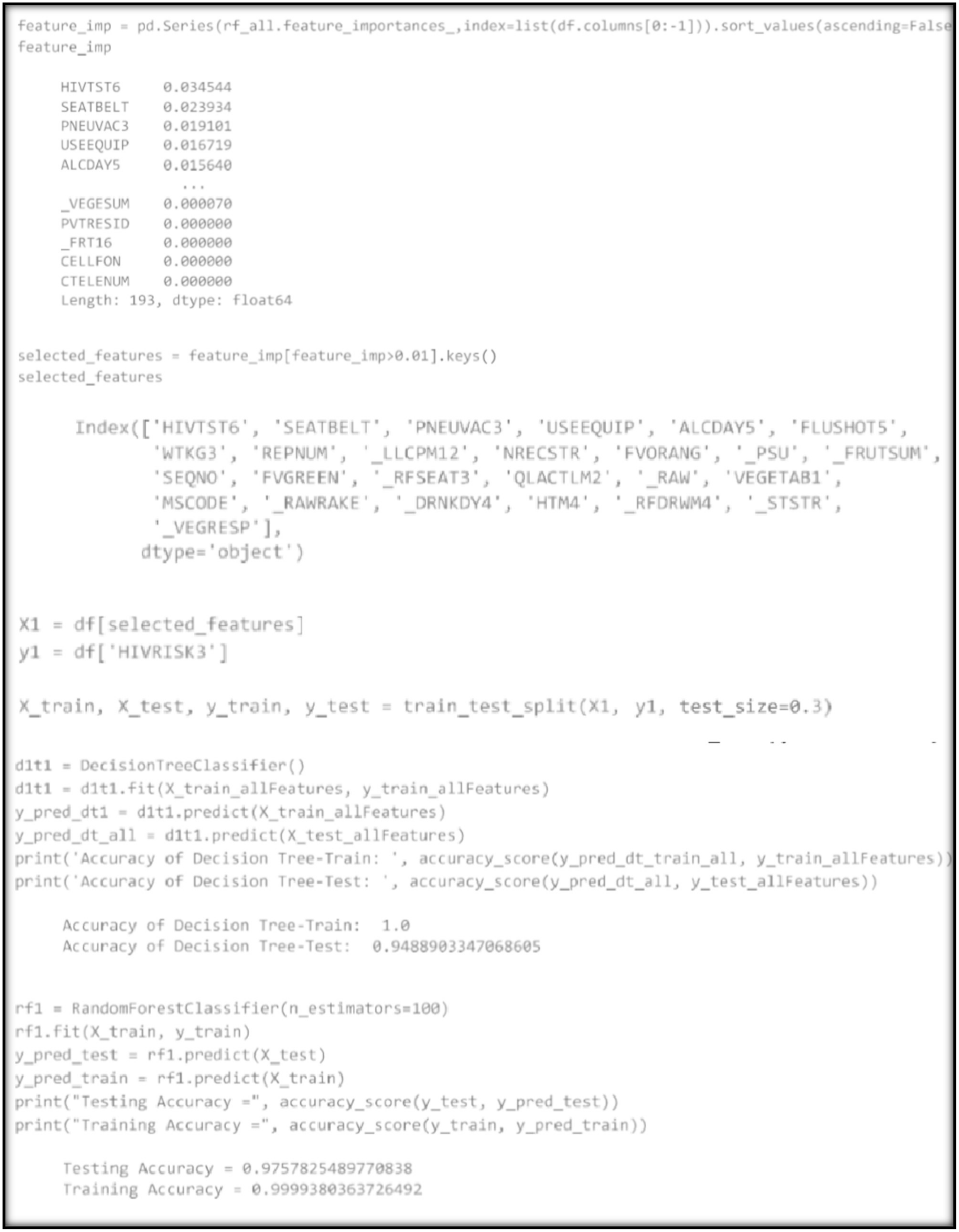
```
y_pred_rf_train_all = dt_allFeatures.predict(X_train_allFeatures)
y_pred_rf_all = dt_allFeatures.predict(X_test_allFeatures)
```

```
from sklearn.metrics import confusion_matrix
cm_rf_train_allFeatures = confusion_matrix(y_train_allFeatures, y_pred_rf_train_all)
cm_rf_test_allFeatures = confusion_matrix(y_test_allFeatures, y_pred_rf_all)


print('Confusion Matrix - Train:','\n',cm_rf_train_allFeatures)
print('\n','Confusion Matrix - Test:','\n',cm_rf_test_allFeatures)
```

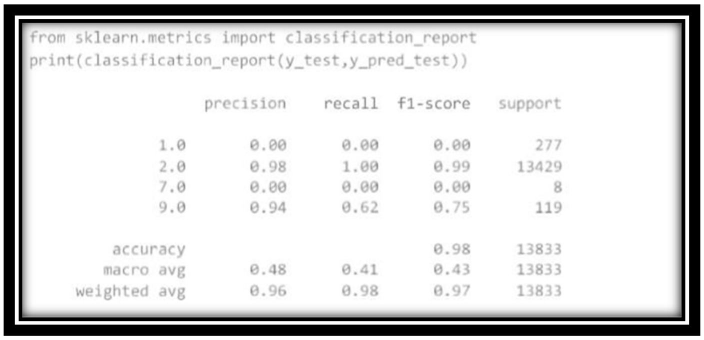
```
Confusion Matrix - Train:
 [[  655     0     0     0]
 [     0 31297     0     0]
 [     0     0    20     0]
 [     0     0     0   305]]

 Confusion Matrix - Test:
 [[   27   258     0     1]
 [  329 13032    11    48]
 [     0    10     0     0]
 [     3    42     0    72]]
```

```
from sklearn.metrics import accuracy_score
print('Accuracy of Decision Tree-Train: ', accuracy_score(y_pred_rf_train_all, y_train_allFeatures))
print('Accuracy of Decision Tree-Test: ', accuracy_score(y_pred_rf_all, y_test_allFeatures))
```

```
Accuracy of Decision Tree-Train:  1.0
Accuracy of Decision Tree-Test:  0.9492517891997397
```



```
feature_imp = pd.Series(rf_all.feature_importances_,index=list(df.columns[0:-1])).sort_values(ascending=False
feature_imp

     HIVTST6     0.034544
     SEATBELT    0.023934
     PNEUVAC3    0.019101
     USEEQUIP    0.016719
     ALCDAY5     0.015640
                   ...
     _VEGESUM    0.000070
     PVTRESID    0.000000
     _FRT16      0.000000
     CELLFON     0.000000
     CTELENUM    0.000000
     Length: 193, dtype: float64


selected_features = feature_imp[feature_imp>0.01].keys()
selected_features

     Index(['HIVTST6', 'SEATBELT', 'PNEUVAC3', 'USEEQUIP', 'ALCDAY5', 'FLUSHOT5',
            'WTKG3', 'REPNUM', '_LLCPM12', 'NRECSTR', 'FVORANG', '_PSU', '_FRUTSUM',
            'SEQNO', 'FVGREEN', '_RFSEAT3', 'QLACTLM2', '_RAW', 'VEGETAB1',
            'MSCODE', '_RAWRAKE', '_DRNKDY4', 'HTM4', '_RFDRWM4', '_STSTR',
            '_VEGRESP'],
           dtype='object')


X1 = df[selected_features]
y1 = df['HIVRISK3']

X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3)


d1t1 = DecisionTreeClassifier()
d1t1 = d1t1.fit(X_train_allFeatures, y_train_allFeatures)
y_pred_dt1 = d1t1.predict(X_train_allFeatures)
y_pred_dt_all = d1t1.predict(X_test_allFeatures)
print('Accuracy of Decision Tree-Train: ', accuracy_score(y_pred_dt_train_all, y_train_allFeatures))
print('Accuracy of Decision Tree-Test: ', accuracy_score(y_pred_dt_all, y_test_allFeatures))

     Accuracy of Decision Tree-Train:  1.0
     Accuracy of Decision Tree-Test:  0.9488903347068605


rf1 = RandomForestClassifier(n_estimators=100)
rf1.fit(X_train, y_train)
y_pred_test = rf1.predict(X_test)
y_pred_train = rf1.predict(X_train)
print("Testing Accuracy =", accuracy_score(y_test, y_pred_test))
print("Training Accuracy =", accuracy_score(y_train, y_pred_train))

     Testing Accuracy = 0.9757825489770838
     Training Accuracy = 0.9999380363726492
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_test))

              precision    recall  f1-score   support

         1.0       0.00      0.00      0.00       277
         2.0       0.98      1.00      0.99     13429
         7.0       0.00      0.00      0.00         8
         9.0       0.94      0.62      0.75       119

    accuracy                           0.98     13833
   macro avg       0.48      0.41      0.43     13833
weighted avg       0.96      0.98      0.97     13833
```