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)

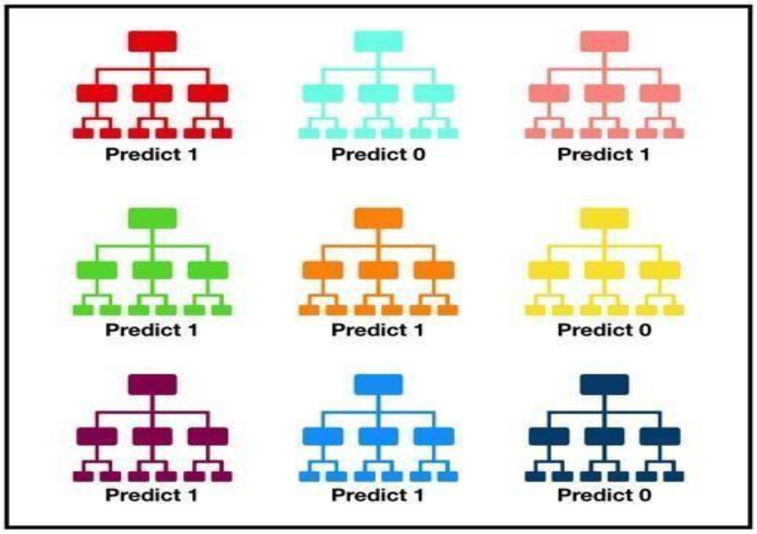
Department of Computer Science and Engineering (Data Science)
Subject: Machine Learning – I (DJ19DSC402)
AY: 2022-23
Experiment 6
(Random Forest)

NAME : Bhuvi Ghosh SAPID : 60009210191 BATCH : B/B3

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:

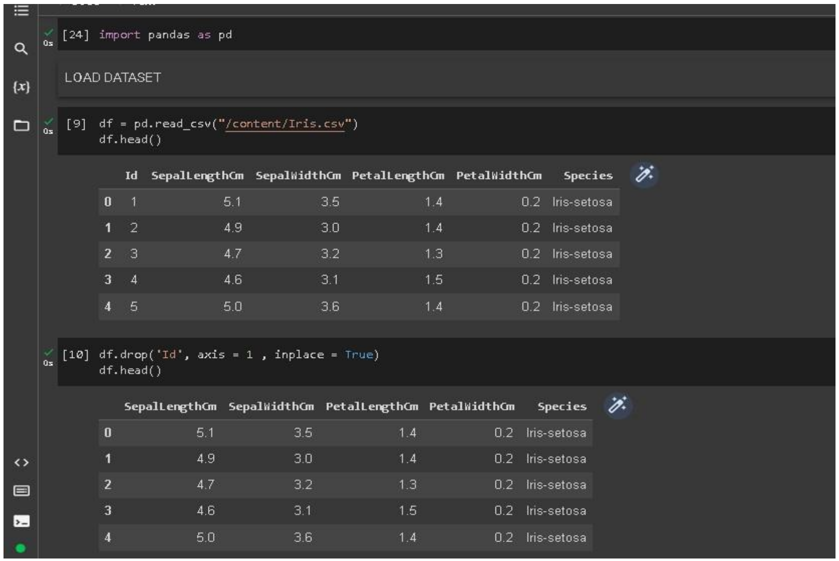
1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.

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.



```
[24] import pandas as pd
```

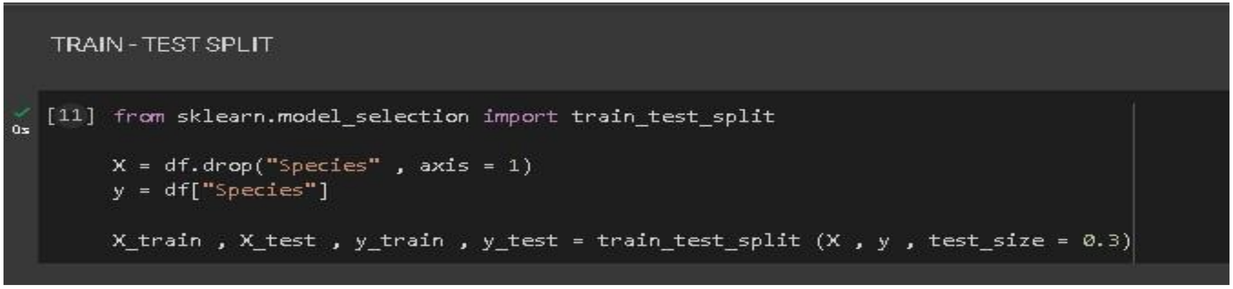
LOAD DATASET

```
[9]  df = pd.read_csv("/content/Iris.csv")
     df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	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

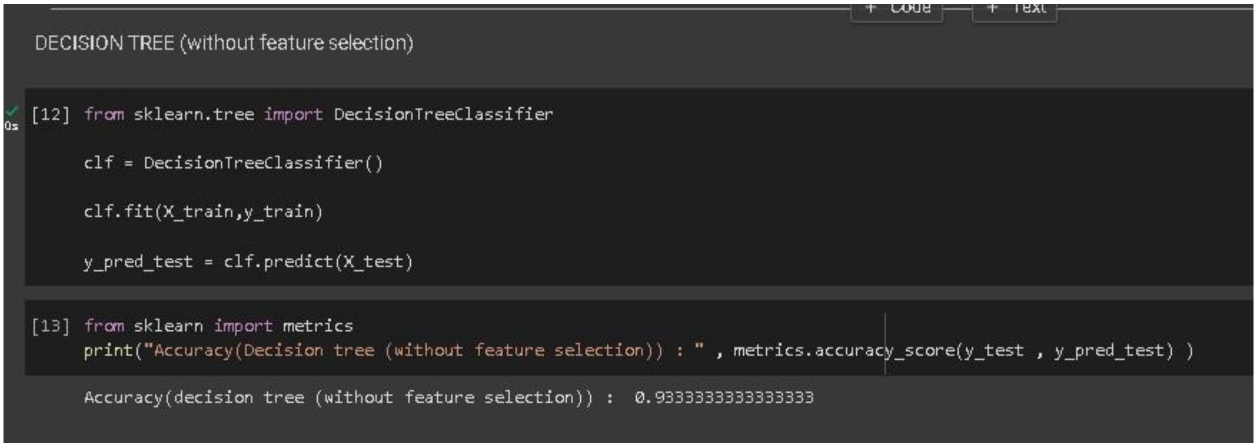
```
[10] df.drop('Id', axis = 1 , inplace = True)
     df.head()
```

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



TRAIN - TEST SPLIT

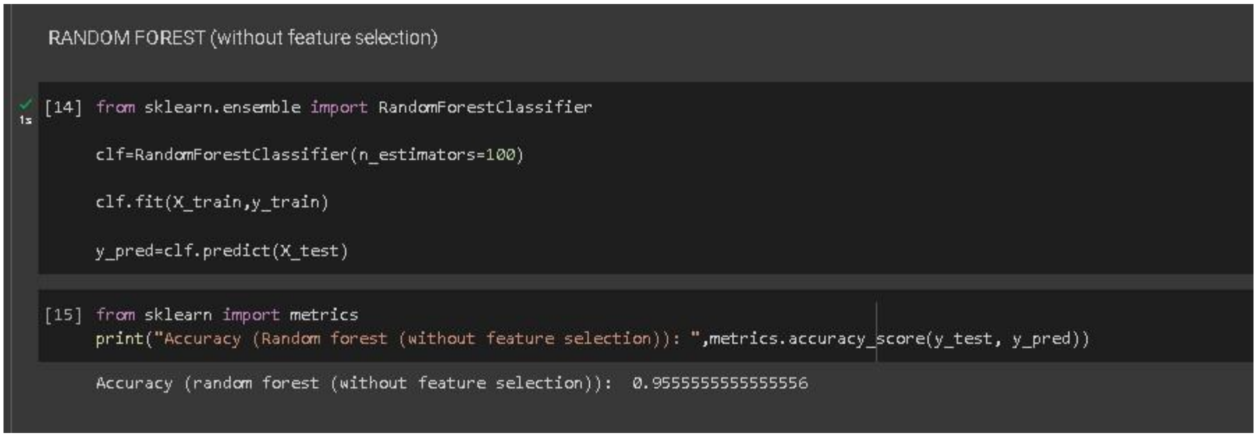
```
[11] from sklearn.model_selection import train_test_split

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

     X_train , X_test , y_train , y_test = train_test_split (X , y , test_size = 0.3)
```



DECISION TREE (without feature selection)

```
[12] from sklearn.tree import DecisionTreeClassifier

     clf = DecisionTreeClassifier()

     clf.fit(X_train,y_train)

     y_pred_test = clf.predict(X_test)
```

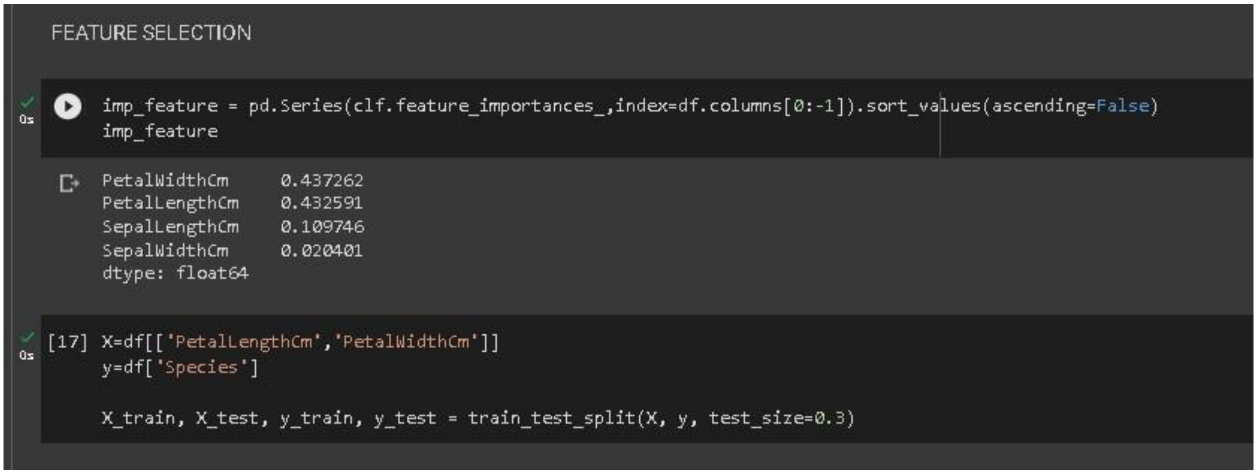
```
[13] from sklearn import metrics
     print("Accuracy(Decision tree (without feature selection)) : " , metrics.accuracy_score(y_test , y_pred_test) )

     Accuracy(decision tree (without feature selection)) :  0.9333333333333333
```



RANDOM FOREST (without feature selection)

```
[14] from sklearn.ensemble import RandomForestClassifier

     clf=RandomForestClassifier(n_estimators=100)

     clf.fit(X_train,y_train)

     y_pred=clf.predict(X_test)
```

```
[15] from sklearn import metrics
     print("Accuracy (Random forest (without feature selection)): ",metrics.accuracy_score(y_test, y_pred))

     Accuracy (random forest (without feature selection)):  0.9555555555555556
```

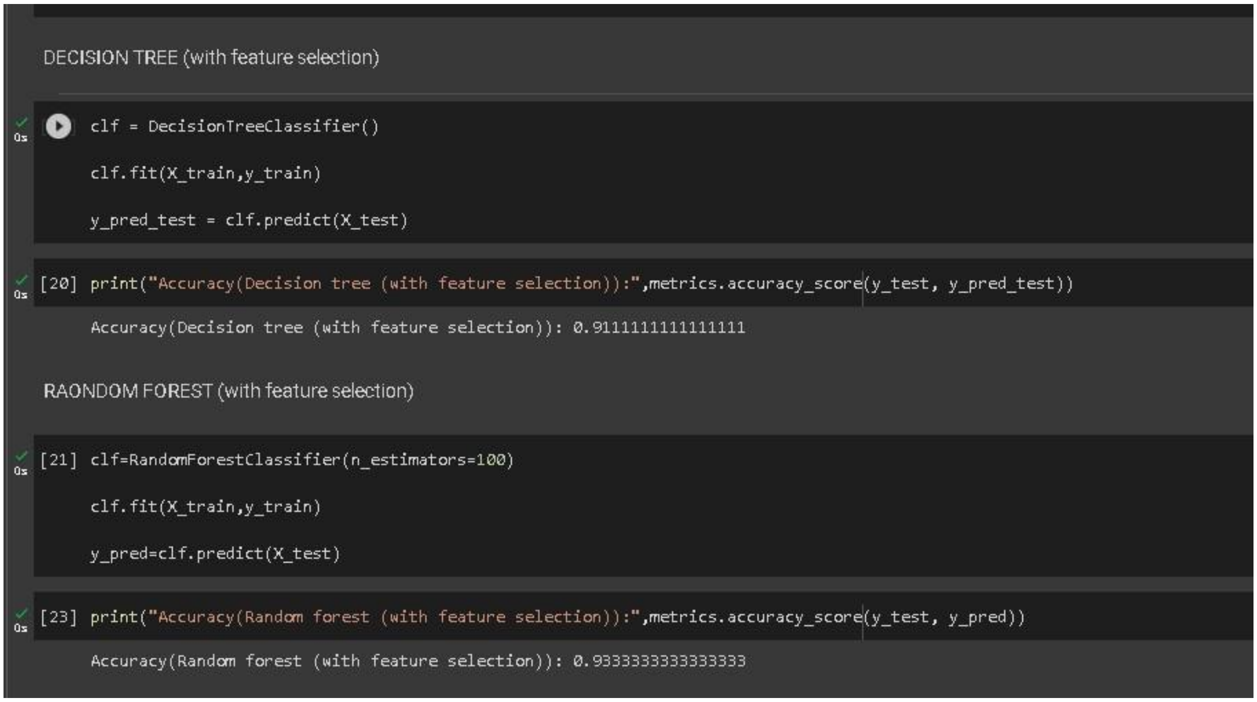


FEATURE SELECTION

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

```
PetalWidthCm     0.437262
PetalLengthCm    0.432591
SepalLengthCm    0.109746
SepalWidthCm     0.020401
dtype: float64
```

```
[17] X=df[['PetalLengthCm','PetalWidthCm']]
     y=df['Species']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```



DECISION TREE (with feature selection)

```
clf = DecisionTreeClassifier()

clf.fit(X_train,y_train)

y_pred_test = clf.predict(X_test)
```

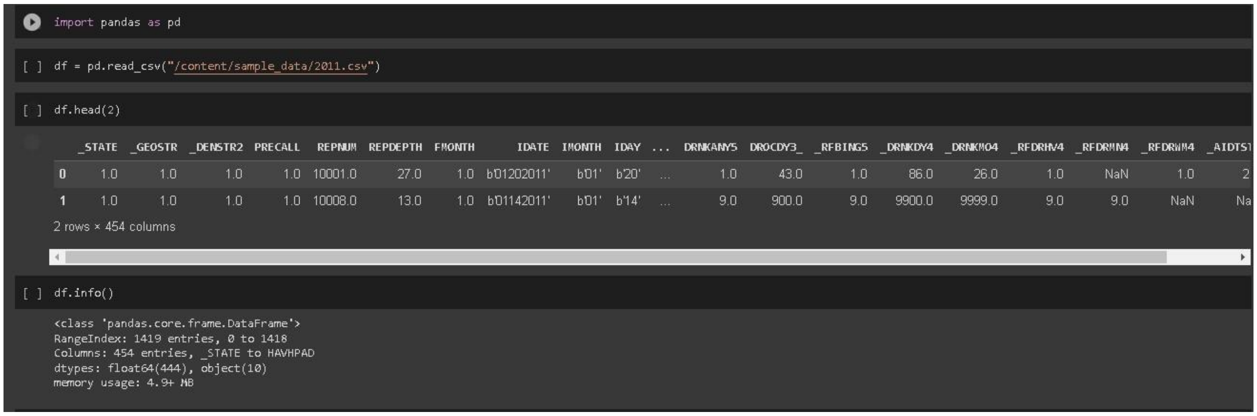
```
[20] print("Accuracy(Decision tree (with feature selection)):",metrics.accuracy_score(y_test, y_pred_test))

     Accuracy(Decision tree (with feature selection)): 0.9111111111111111
```

RAONDOM FOREST (with feature selection)

```
[21] clf=RandomForestClassifier(n_estimators=100)

     clf.fit(X_train,y_train)

     y_pred=clf.predict(X_test)
```

```
[23] print("Accuracy(Random forest (with feature selection)):",metrics.accuracy_score(y_test, y_pred))

     Accuracy(Random forest (with feature selection)): 0.9333333333333333
```

Q2] Compare the results of random forest with and without selecting important features only for building the classifier on dataset 2 and 3.



```
import pandas as pd
```

```
[ ] df = pd.read_csv("/content/sample_data/2011.csv")
```

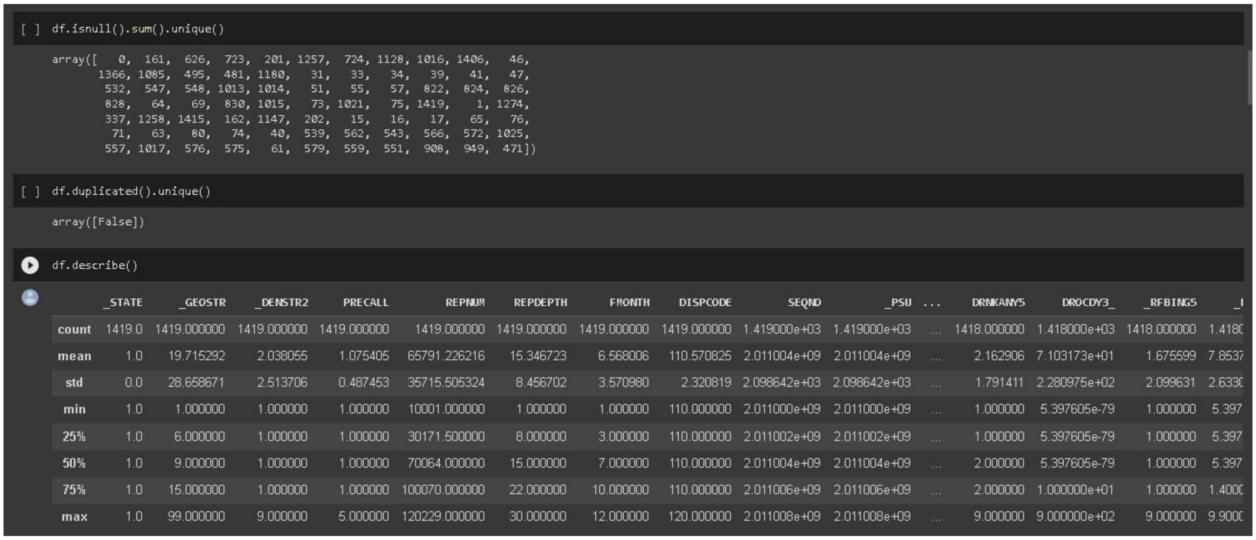
```
[ ] df.head(2)
```

	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	IDATE	IMONTH	IDAY	...	DRNKANY5	DROCDY3_	_RFBING5	_DRNKDY4	_DRNKMO4	_RFDRHV4	_RFDRMN4	_RFDRWM4	_AIDTS1
0	1.0	1.0	1.0	1.0	10001.0	27.0	1.0	b'01202011'	b'01'	b'20'	...	1.0	43.0	1.0	86.0	26.0	1.0	NaN	1.0	2
1	1.0	1.0	1.0	1.0	10008.0	13.0	1.0	b'01142011'	b'01'	b'14'	...	9.0	900.0	9.0	9900.0	9999.0	9.0	9.0	NaN	Na

2 rows × 454 columns

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1419 entries, 0 to 1418
Columns: 454 entries, _STATE to HAVHPAD
dtypes: float64(444), object(10)
memory usage: 4.9+ MB
```



```
[ ] df.isnull().sum().unique()
```

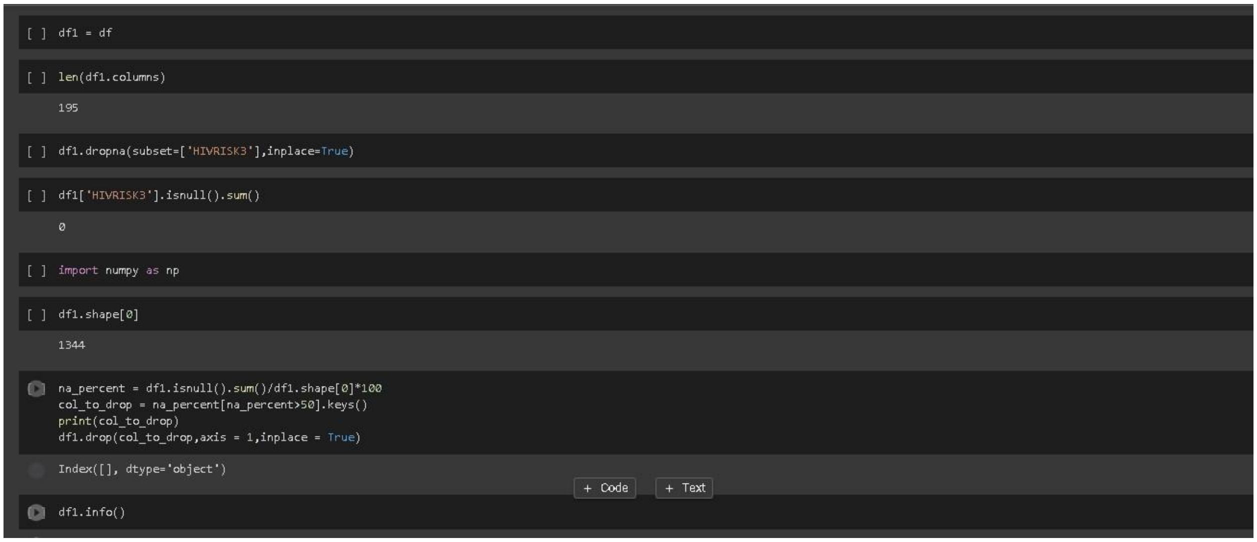
```
array([   0,  161,  626,  723,  201, 1257,  724, 1128, 1016, 1406,   46,
       1366, 1085,  495,  481, 1180,   31,   33,   34,   39,   41,   47,
        532,  547,  548, 1013, 1014,   51,   55,   57,  822,  824,  826,
        828,   64,   69,  830, 1015,   73, 1021,   75, 1419,    1, 1274,
        337, 1258, 1415,  162, 1147,  202,   15,   16,   17,   65,   76,
         71,   63,   80,   74,   40,  539,  562,  543,  566,  572, 1025,
        557, 1017,  576,  575,   61,  579,  559,  551,  908,  949,  471])
```

```
[ ] df.duplicated().unique()
```

```
array([False])
```

```
df.describe()
```

	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	DISPCODE	SEQNO	_PSU	...	DRNKANY5	DROCDY3_	_RFBING5	_I
count	1419.0	1419.000000	1419.000000	1419.000000	1419.000000	1419.000000	1419.000000	1419.000000	1.419000e+03	1.419000e+03	...	1418.000000	1.418000e+03	1418.000000	1.418C
mean	1.0	19.715292	2.038055	1.075405	65791.226216	15.346723	6.568006	110.570825	2.011004e+09	2.011004e+09	...	2.162906	7.103173e+01	1.675599	7.8537
std	0.0	28.658671	2.513706	0.487453	35715.505324	8.456702	3.570980	2.320819	2.098642e+03	2.098642e+03	...	1.791411	2.280975e+02	2.099631	2.633C
min	1.0	1.000000	1.000000	1.000000	10001.000000	1.000000	1.000000	110.000000	2.011000e+09	2.011000e+09	...	1.000000	5.397605e-79	1.000000	5.397
25%	1.0	6.000000	1.000000	1.000000	30171.500000	8.000000	3.000000	110.000000	2.011002e+09	2.011002e+09	...	1.000000	5.397605e-79	1.000000	5.397
50%	1.0	9.000000	1.000000	1.000000	70064.000000	15.000000	7.000000	110.000000	2.011004e+09	2.011004e+09	...	2.000000	5.397605e-79	1.000000	5.397
75%	1.0	15.000000	1.000000	1.000000	100070.000000	22.000000	10.000000	110.000000	2.011006e+09	2.011006e+09	...	2.000000	1.000000e+01	1.000000	1.400C
max	1.0	99.000000	9.000000	5.000000	120229.000000	30.000000	12.000000	120.000000	2.011008e+09	2.011008e+09	...	9.000000	9.000000e+02	9.000000	9.900C



```
[ ] df1 = df
```

```
[ ] len(df1.columns)

    195
```

```
[ ] df1.dropna(subset=['HIVRISK3'],inplace=True)
```

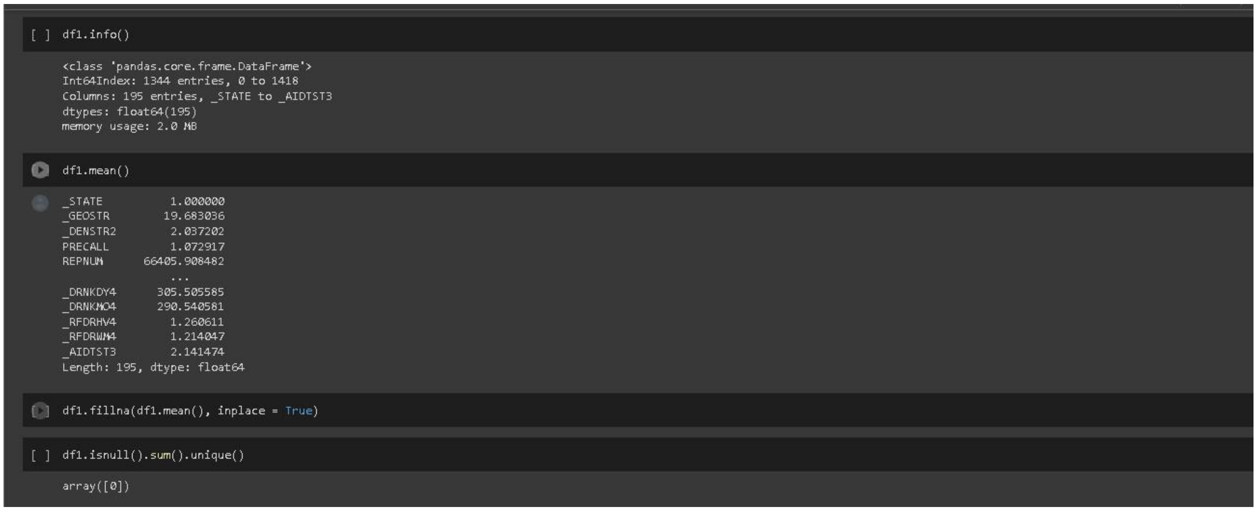
```
[ ] df1['HIVRISK3'].isnull().sum()

    0
```

```
[ ] import numpy as np
```

```
[ ] df1.shape[0]

    1344
```

```
    na_percent = df1.isnull().sum()/df1.shape[0]*100
    col_to_drop = na_percent[na_percent>50].keys()
    print(col_to_drop)
    df1.drop(col_to_drop,axis = 1,inplace = True)

    Index([], dtype='object')
```

```
    df1.info()
```

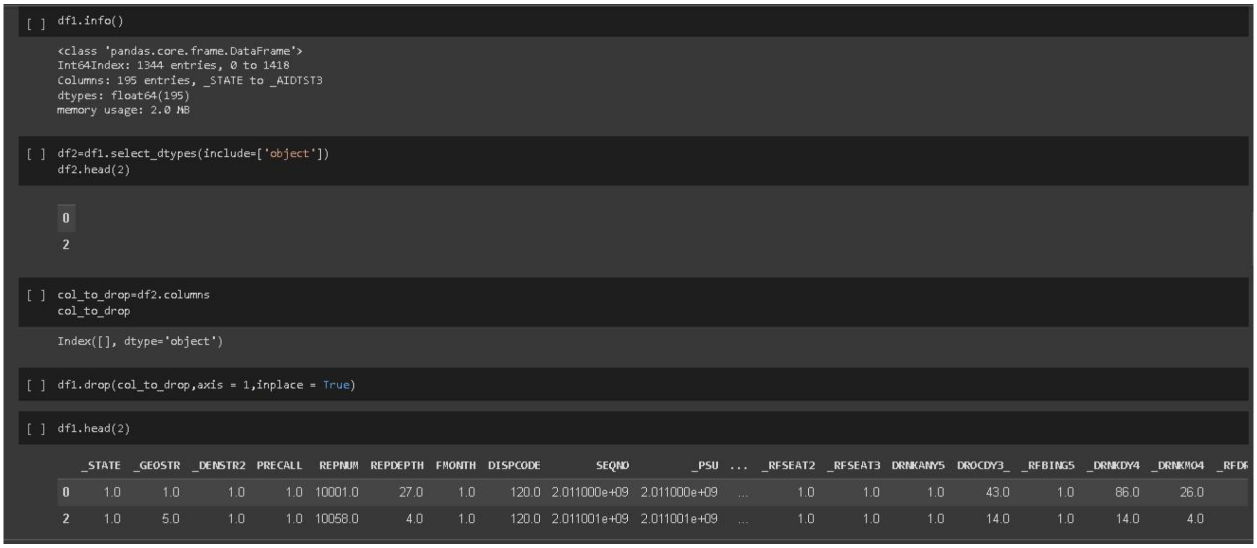


```
[ ] df1.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1344 entries, 0 to 1418
    Columns: 195 entries, _STATE to _AIDTST3
    dtypes: float64(195)
    memory usage: 2.0 MB
```

```
    df1.mean()

    _STATE          1.000000
    _GEOSTR        19.683036
    _DENSTR2        2.037202
    PRECALL         1.072917
    REPNUM      66405.908482
                   ...
    _DRNKDY4      305.505585
    _DRNKMO4      290.540581
    _RFDRHV4        1.260611
    _RFDRWM4        1.214047
    _AIDTST3        2.141474
    Length: 195, dtype: float64
```

```
    df1.fillna(df1.mean(), inplace = True)
```

```
[ ] df1.isnull().sum().unique()

    array([0])
```



```
[ ] df1.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1344 entries, 0 to 1418
    Columns: 195 entries, _STATE to _AIDTST3
    dtypes: float64(195)
    memory usage: 2.0 MB
```

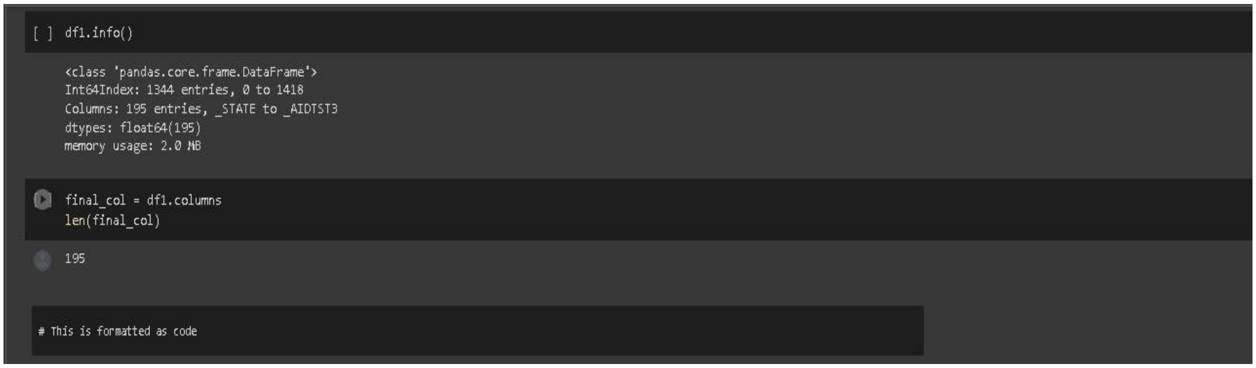
```
[ ] df2=df1.select_dtypes(include=['object'])
    df2.head(2)

    0
    2
```

```
[ ] col_to_drop=df2.columns
    col_to_drop

    Index([], dtype='object')
```

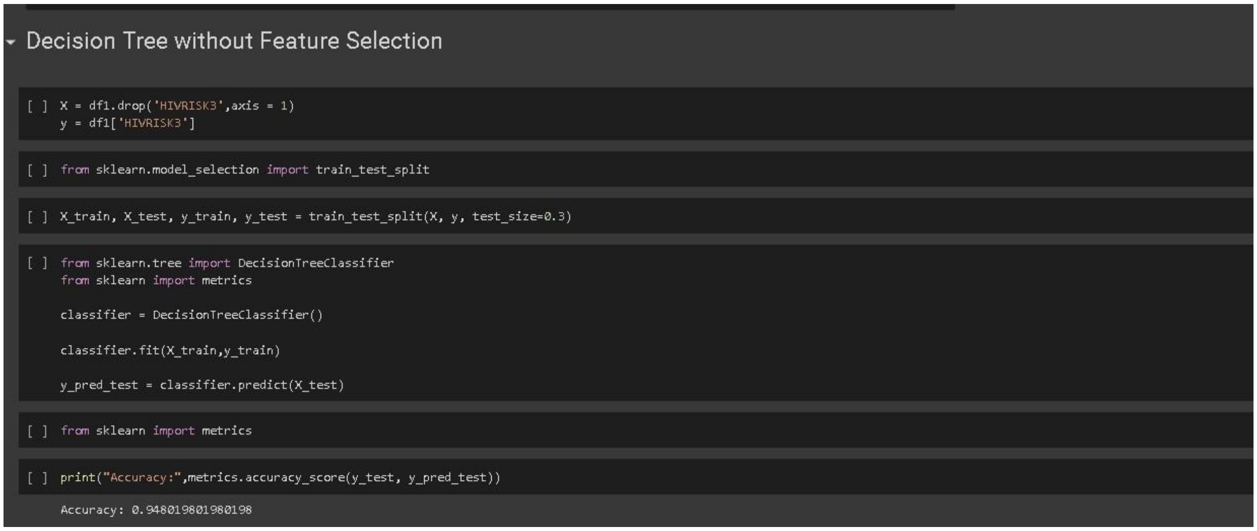
```
[ ] df1.drop(col_to_drop,axis = 1,inplace = True)
```

```
[ ] df1.head(2)
```

	_STATE	_GEOSTR	_DENSTR2	PRECALL	REPNUM	REPDEPTH	FMONTH	DISPCODE	SEQNO	_PSU	...	_RFSEAT2	_RFSEAT3	DRNKANY5	DROCDY3_	_RFBING5	_DRNKDY4	_DRNKMO4	_RFDF
0	1.0	1.0	1.0	1.0	10001.0	27.0	1.0	120.0	2.011000e+09	2.011000e+09	...	1.0	1.0	1.0	43.0	1.0	86.0	26.0	
2	1.0	5.0	1.0	1.0	10058.0	4.0	1.0	120.0	2.011001e+09	2.011001e+09	...	1.0	1.0	1.0	14.0	1.0	14.0	4.0	

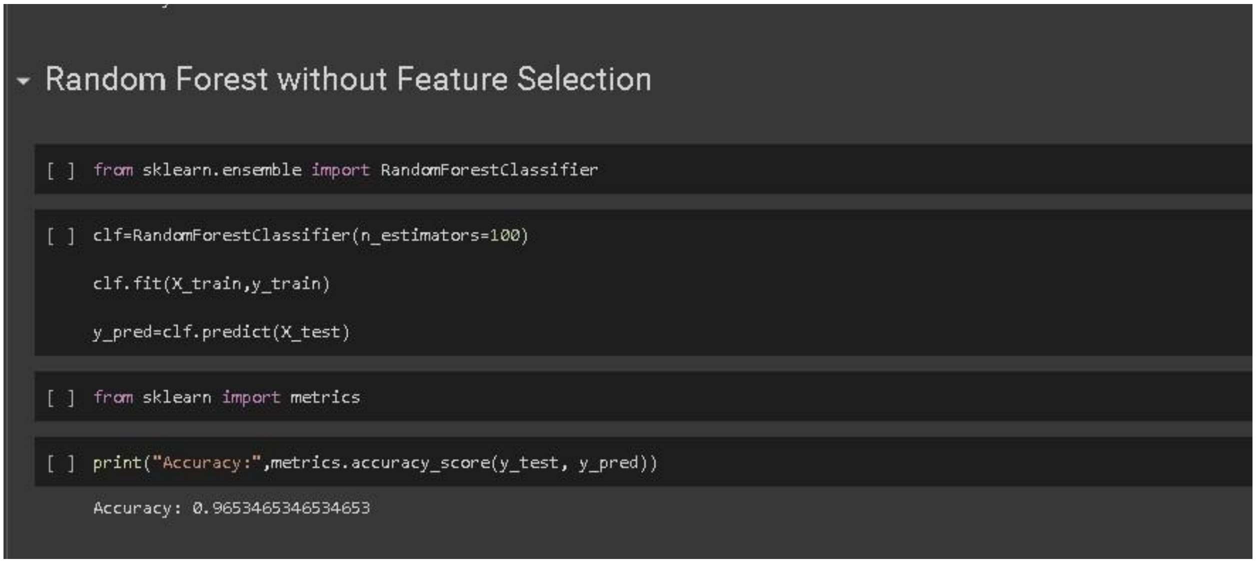


```
[ ] df1.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1344 entries, 0 to 1418
    Columns: 195 entries, _STATE to _AIDTST3
    dtypes: float64(195)
    memory usage: 2.0 MB

    final_col = df1.columns
    len(final_col)

    195

# This is formatted as code
```



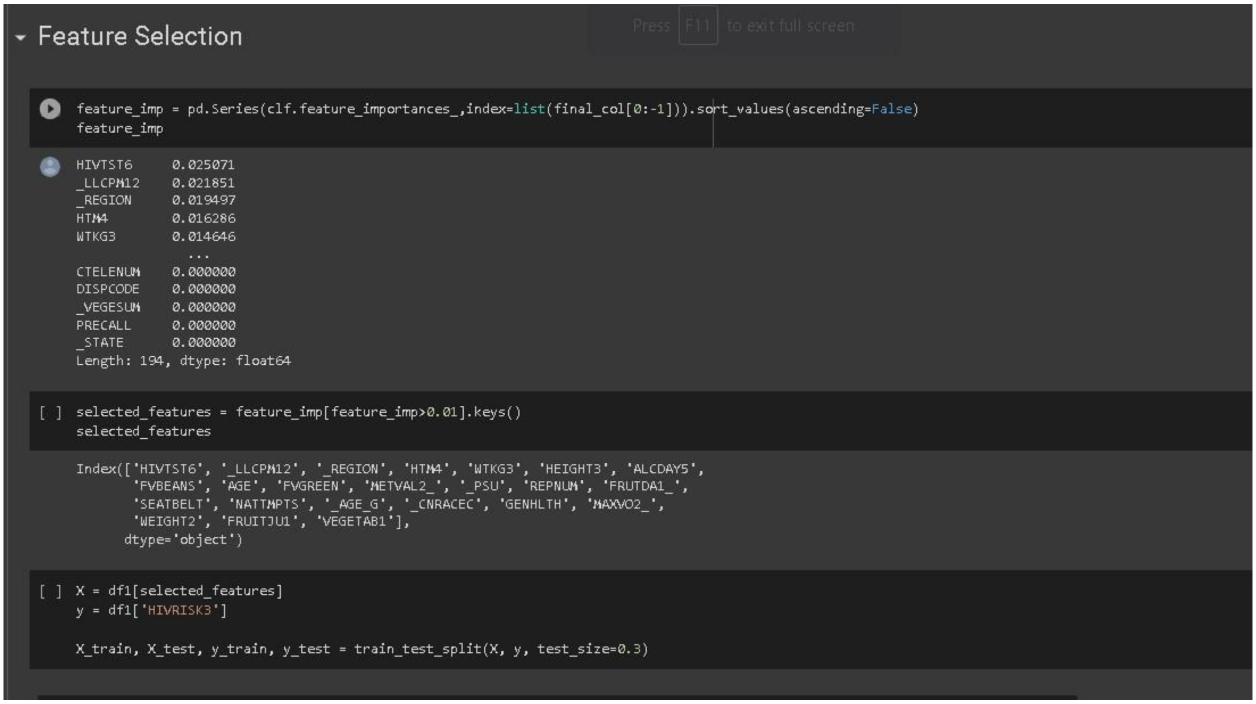
Decision Tree without Feature Selection

```
[ ] X = df1.drop('HIVRISK3',axis = 1)
    y = df1['HIVRISK3']

[ ] 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)

[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn import metrics

    classifier = DecisionTreeClassifier()

    classifier.fit(X_train,y_train)

    y_pred_test = classifier.predict(X_test)

[ ] from sklearn import metrics

[ ] print("Accuracy:",metrics.accuracy_score(y_test, y_pred_test))

    Accuracy: 0.948019801980198
```



Random Forest without Feature Selection

```
[ ] from sklearn.ensemble import RandomForestClassifier

[ ] clf=RandomForestClassifier(n_estimators=100)

    clf.fit(X_train,y_train)

    y_pred=clf.predict(X_test)

[ ] from sklearn import metrics

[ ] print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

    Accuracy: 0.9653465346534653
```



Feature Selection

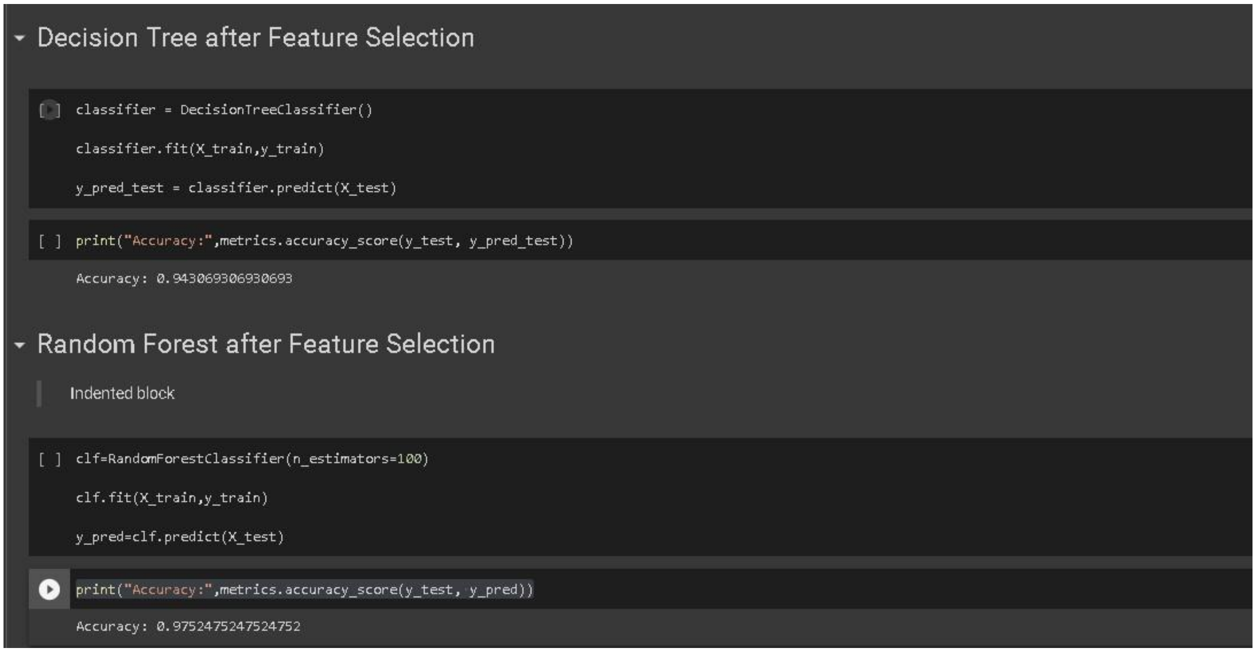
```
feature_imp = pd.Series(clf.feature_importances_,index=list(final_col[0:-1])).sort_values(ascending=False)
feature_imp
```

```
HIVTST6     0.025071
_LLCPM12    0.021851
_REGION     0.019497
HTM4        0.016286
WTKG3       0.014646
              ...
CTELENUM    0.000000
DISPCODE    0.000000
_VEGESUM    0.000000
PRECALL     0.000000
_STATE      0.000000
Length: 194, dtype: float64
```

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

```
Index(['HIVTST6', '_LLCPM12', '_REGION', 'HTM4', 'WTKG3', 'HEIGHT3', 'ALCDAY5',
       'FVBEANS', 'AGE', 'FVGREEN', 'METVAL2_', '_PSU', 'REPNUM', 'FRUTDA1_',
       'SEATBELT', 'NATTMPTS', '_AGE_G', '_CNRACEC', 'GENHLTH', 'MAXVO2_',
       'WEIGHT2', 'FRUITJU1', 'VEGETAB1'],
      dtype='object')
```

```
[ ] X = df1[selected_features]
    y = df1['HIVRISK3']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```



Decision Tree after Feature Selection

```
classifier = DecisionTreeClassifier()

classifier.fit(X_train,y_train)

y_pred_test = classifier.predict(X_test)
```

```
[ ] print("Accuracy:",metrics.accuracy_score(y_test, y_pred_test))
```

```
Accuracy: 0.9430693069306931
```

Random Forest after Feature Selection

> Indented block

```
[ ] clf=RandomForestClassifier(n_estimators=100)

    clf.fit(X_train,y_train)

    y_pred=clf.predict(X_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9752475247524752
```