Importing Libraries

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Reading the data

In [ ]:

```
smoke_data=pd.read_csv("/content/drive/MyDrive/CSV files/train_dataset.csv")
```

Analysing the Data

In [ ]:

```
smoke_data.head()
```

Out[ ]:

	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	relaxation	...	HD
0	35	170	85	97.0	0.9	0.9	1	1	118	78	...	7
1	20	175	110	110.0	0.7	0.9	1	1	119	79	...	7
2	45	155	65	86.0	0.9	0.9	1	1	110	80	...	5
3	45	165	80	94.0	0.8	0.7	1	1	158	88	...	4
4	20	165	60	81.0	1.5	0.1	1	1	109	64	...	4

5 rows × 23 columns

In [ ]:

```
smoke_data.tail()
```

Out[ ]:

	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	relaxation	...
38979	40	165	60	80.0	0.4	0.6	1	1	107	60	...
38980	45	155	55	75.0	1.5	1.2	1	1	126	72	...
38981	40	170	105	124.0	0.6	0.5	1	1	141	85	...
38982	40	160	55	75.0	1.5	1.5	1	1	95	69	...
38983	55	175	60	81.1	1.0	1.0	1	1	114	66	...

5 rows × 23 columns

In [ ]:

```
smoke_data.shape
```

Out[ ]:

```
(38984, 23)
```

In [ ]:

```
smoke_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38984 entries, 0 to 38983
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  38984 non-null  int64  
 1   height(cm)           38984 non-null  int64  
 2   weight(kg)           38984 non-null  int64  
 3   waist(cm)            38984 non-null  float64
 4   eyesight(left)       38984 non-null  float64
 5   eyesight(right)      38984 non-null  float64
 6   hearing(left)        38984 non-null  int64  
 7   hearing(right)       38984 non-null  int64  
 8   systolic             38984 non-null  int64  
 9   relaxation           38984 non-null  int64  
 10  fasting blood sugar  38984 non-null  int64  
 11  Cholesterol          38984 non-null  int64  
 12  triglyceride         38984 non-null  int64  
 13  HDL                  38984 non-null  int64  
 14  LDL                  38984 non-null  int64  
 15  hemoglobin           38984 non-null  float64
 16  Urine protein        38984 non-null  int64  
 17  serum creatinine     38984 non-null  float64
 18  AST                  38984 non-null  int64  
 19  ALT                  38984 non-null  int64  
 20  Gtp                  38984 non-null  int64  
 21  dental caries        38984 non-null  int64  
 22  smoking              38984 non-null  int64  
dtypes: float64(5), int64(18)
memory usage: 6.8 MB
```

In [ ]:

```
smoke_data.isnull().sum()
```

Out[ ]:

```
age                    0
height(cm)             0
weight(kg)             0
waist(cm)              0
eyesight(left)         0
eyesight(right)        0
hearing(left)          0
hearing(right)         0
systolic               0
relaxation             0
fasting blood sugar    0
Cholesterol            0
triglyceride           0
HDL                    0
LDL                    0
hemoglobin             0
Urine protein          0
serum creatinine       0
AST                    0
ALT                    0
Gtp                    0
dental caries          0
smoking                0
dtype: int64
```

In [ ]:

```
smoke_data.describe()
```

Out[ ]:

	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)
count	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000	38984.000000
mean	44.127591	164.689488	65.938718	82.062115	1.014955	1.008768	1.025369	1.026190
std	12.063564	9.187507	12.896581	9.326798	0.498527	0.493813	0.157246	0.159703
min	20.000000	130.000000	30.000000	51.000000	0.100000	0.100000	1.000000	1.000000
25%	40.000000	160.000000	55.000000	76.000000	0.800000	0.800000	1.000000	1.000000
50%	40.000000	165.000000	65.000000	82.000000	1.000000	1.000000	1.000000	1.000000
75%	55.000000	170.000000	75.000000	88.000000	1.200000	1.200000	1.000000	1.000000
max	85.000000	190.000000	135.000000	129.000000	9.900000	9.900000	2.000000	2.000000

8 rows × 23 columns

In [ ]:

```
smoke_data['smoking'].value_counts()
```

Out[ ]:

```
0    24666
1    14318
Name: smoking, dtype: int64
```

In [ ]:

```
X= smoke_data.drop(columns='smoking',axis=1)
Y= smoke_data['smoking']
```

In [ ]:

```
print(X)
```

```
       age  height(cm)  weight(kg)  waist(cm)  eyesight(left)  \
0       35         170          85       97.0             0.9   
1       20         175         110      110.0             0.7   
2       45         155          65       86.0             0.9   
3       45         165          80       94.0             0.8   
4       20         165          60       81.0             1.5   
...    ...         ...         ...        ...             ...   
38979   40         165          60       80.0             0.4   
38980   45         155          55       75.0             1.5   
38981   40         170         105      124.0             0.6   
38982   40         160          55       75.0             1.5   
38983   55         175          60       81.1             1.0   

       eyesight(right)  hearing(left)  hearing(right)  systolic  relaxation  \
0                  0.9              1               1       118          78   
1                  0.9              1               1       119          79   
2                  0.9              1               1       110          80   
3                  0.7              1               1       158          88   
4                  0.1              1               1       109          64   
...                ...            ...             ...       ...         ...   
38979              0.6              1               1       107          60   
38980              1.2              1               1       126          72   
38981              0.5              1               1       141          85   
38982              1.5              1               1        95          69   
38983              1.0              1               1       114          66   

       ...  triglyceride  HDL  LDL  hemoglobin  Urine protein  \
0      ...           153   70  142        19.8              1   
1      ...           128   71  114        15.9              1   
2      ...           120   57  112        13.7              3   
3      ...           366   46   91        16.9              1   
4      ...           200   47   92        14.9              1   
...    ...           ...  ...  ...         ...            ...   
```

```
38979  ...             53   61   72       12.3              1
38980  ...            100   76  131       12.5              2
38981  ...            196   48  138       17.1              1
38982  ...             48   79  116       12.0              1
38983  ...             57   64  137       13.9              1

       serum creatinine   AST   ALT  Gtp  dental caries
0                   1.0    61   115  125              1
1                   1.1    19    25   30              1
2                   0.6  1090  1400  276              0
3                   0.9    32    36   36              0
4                   1.2    26    28   15              0
...                 ...   ...   ...  ...            ...
38979               0.5    18    18   21              1
38980               0.6    23    11   12              0
38981               0.8    24    23   35              1
38982               0.6    24    20   17              0
38983               1.0    18    12   16              0

[38984 rows x 22 columns]
```

In [ ]:

```
print(Y)
```

```
0        1
1        0
2        0
3        0
4        0
        ..
38979    0
38980    0
38981    1
38982    1
38983    1
Name: smoking, Length: 38984, dtype: int64
```

Training the Data

In [ ]:

```
X_train, X_test, Y_train,Y_test= train_test_split(X,Y,test_size=0.4,stratify=Y, random_s
tate=4)
```

In [ ]:

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(38984, 22) (23390, 22) (15594, 22)
```

Logistic Regression

In [ ]:

```
model = LogisticRegression()
```

In [ ]:

```
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[ ]:

```
LogisticRegression()
```

In [ ]:

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [ ]:

```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.7114151346729372
```

In [ ]:

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [ ]:

```
print('Accuracy on Test data : ', test_data_accuracy)
```

```
Accuracy on Test data :  0.7123252533025523
```

In [ ]:

```
input_data = (30,180,90,94.0,1.0,0.8,1,1,115,72,88,177,103,53,103,13.5,1,1.0,19,29,30,0)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not Smoke')
else:
  print('The Person Smokes')
```

```
[0]
The Person does not Smoke
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  "X does not have valid feature names, but"
```

Implementing Test Data on the created model

In [ ]:

```
new_test_data = pd.read_csv("/content/drive/MyDrive/CSV files/test_dataset.csv")
new_test_data.head()
```

Out[ ]:

	age	height(cm)	weight(kg)	waist(cm)	eyesight(left)	eyesight(right)	hearing(left)	hearing(right)	systolic	relaxation	...	trig
0	40	170	65	75.1	1.0	0.9	1	1	120	70	...	
1	45	170	75	89.0	0.7	1.2	1	1	100	67	...	
2	30	180	90	94.0	1.0	0.8	1	1	115	72	...	
3	60	170	50	73.0	0.5	0.7	1	1	118	78	...	
4	30	170	65	78.0	1.5	1.0	1	1	110	70	...	

5 rows × 22 columns

In [ ]:

```
new_test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16708 entries, 0 to 16707
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  16708 non-null  int64  
 1   height(cm)           16708 non-null  int64  
 2   weight(kg)           16708 non-null  int64  
 3   waist(cm)            16708 non-null  float64
 4   eyesight(left)       16708 non-null  float64
 5   eyesight(right)      16708 non-null  float64
 6   hearing(left)        16708 non-null  int64  
 7   hearing(right)       16708 non-null  int64  
 8   systolic             16708 non-null  int64  
 9   relaxation           16708 non-null  int64  
 10  fasting blood sugar  16708 non-null  int64  
 11  Cholesterol          16708 non-null  int64  
 12  triglyceride         16708 non-null  int64  
 13  HDL                  16708 non-null  int64  
 14  LDL                  16708 non-null  int64  
 15  hemoglobin           16708 non-null  float64
 16  Urine protein        16708 non-null  int64  
 17  serum creatinine     16708 non-null  float64
 18  AST                  16708 non-null  int64  
 19  ALT                  16708 non-null  int64  
 20  Gtp                  16708 non-null  int64  
 21  dental caries        16708 non-null  int64  
dtypes: float64(5), int64(17)
memory usage: 2.8 MB
```

In [ ]:

```
input_data=new_test_data
prediction = model.predict(input_data)
#print("\n",prediction)
res=pd.DataFrame(prediction)
res.index=new_test_data.index
res.columns=["prediction"]
res.to_csv("prediction_results_smoke_data.csv")
#files.download("prediction_results_smoke_data.csv")
#if (prediction[0]== 0):
#  print('The Person does not Smoke')
#else:
#  print('The Person Smokes')
```

Downloading the Prediction file

In [ ]:

```
from google.colab import files
files.download("prediction_results_smoke_data.csv")
```

Random Forest Classifier

In [88]:

```
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier
```

In [95]:

In [95]:

```
forest = RandomForestClassifier(random_state=1, n_estimators=1000, max_depth=5)

forest.fit(X_train, Y_train)
```

Out[95]:

```
RandomForestClassifier(max_depth=5, n_estimators=1000, random_state=1)
```

Boruta

In [ ]:

```
!pip install boruta
```

```
Collecting boruta
  Downloading Boruta-0.3-py3-none-any.whl (56 kB)
     |████████████████████████████████| 56 kB 2.6 MB/s
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.7/dist-packages (f
rom boruta) (1.21.6)
Requirement already satisfied: scikit-learn>=0.17.1 in /usr/local/lib/python3.7/dist-pack
ages (from boruta) (1.0.2)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (f
rom boruta) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (fr
om scikit-learn>=0.17.1->boruta) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pack
ages (from scikit-learn>=0.17.1->boruta) (3.1.0)
Installing collected packages: boruta
Successfully installed boruta-0.3
```

In [104]:

```
from boruta import BorutaPy
```

In [111]:

```
boruta_selector = BorutaPy(forest, n_estimators='auto', verbose=2, random_state=1)   # i
nitialize the boruta selector
boruta_selector.fit(np.array(X_train), np.array(Y_train))        #
```

```
Iteration:  1 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  2 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  3 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  4 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  5 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  6 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  7 / 100
Confirmed:  0
Tentative:  22
Rejected:  0
Iteration:  8 / 100
Confirmed:  19
Tentative:  0
```

```
Tentative:  0
Rejected:   3


BorutaPy finished running.

Iteration:  9 / 100
Confirmed:  19
Tentative:  0
Rejected:   3
```

Out[111]:

```
BorutaPy(estimator=RandomForestClassifier(max_depth=5, n_estimators=132,
                                          random_state=RandomState(MT19937) at 0x7FC0774
E1380),
         n_estimators='auto',
         random_state=RandomState(MT19937) at 0x7FC0774E1380, verbose=2)
```

In [112]:

```
print("Selected Features: ", boruta_selector.support_)    # check selected features


print("Ranking: ",boruta_selector.ranking_)               # check ranking of features

print("No. of significant features: ", boruta_selector.n_features_)
```

```
Selected Features:  [ True  True  True  True  True  True False False  True  True  True  T
rue
  True  True  True  True False  True  True  True  True  True]
Ranking:  [1 1 1 1 1 1 4 3 1 1 1 1 1 1 1 1 2 1 1 1 1 1]
No. of significant features:  19
```

In [113]:

```
selected_rfe_features = pd.DataFrame({'Feature':list(X_train.columns),
                                      'Ranking':boruta_selector.ranking_})
selected_rfe_features.sort_values(by='Ranking')
```

Out[113]:

	Feature	Ranking
0	age	1
19	ALT	1
18	AST	1
17	serum creatinine	1
15	hemoglobin	1
14	LDL	1
13	HDL	1
12	triglyceride	1
11	Cholesterol	1
20	Gtp	1
10	fasting blood sugar	1
8	systolic	1
5	eyesight(right)	1
4	eyesight(left)	1
3	waist(cm)	1
2	weight(kg)	1
1	height(cm)	1
9	relaxation	1

	Feature	Ranking
21	dental caries	1
16	Urine protein	2
7	hearing(right)	3
6	hearing(left)	4

In [114]:

```
X_important_train = boruta_selector.transform(np.array(X_train))
X_important_test = boruta_selector.transform(np.array(X_test))
```

In [115]:

```
# Create a new random forest classifier for the most important features
rf_important = RandomForestClassifier(random_state=1, n_estimators=1000, n_jobs = -1)

# Train the new classifier on the new dataset containing the most important features
rf_important.fit(X_important_train, Y_train)
```

Out[115]:

```
RandomForestClassifier(n_estimators=1000, n_jobs=-1, random_state=1)
```

In [117]:

```
y_important_pred = rf_important.predict(X_important_test)
rf_imp_fscore = f1_score(Y_test, y_important_pred)
```

In [118]:

```
print(rf_imp_fscore)
```

```
0.7165043432482232
```

Hyper Parameter Tunning

In [119]:

```
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True, False],
    'max_depth': [5, 10, 15],
    'n_estimators': [500, 1000]}
```

In [120]:

```
rf = RandomForestClassifier(random_state = 1)

# Grid search cv
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 2, n_jobs = -1, verbose = 2)
```

In [121]:

```
grid_search.fit(X_important_train, Y_train)
```

```
Fitting 2 folds for each of 12 candidates, totalling 24 fits
```

Out[121]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(random_state=1), n_jobs=-1,
             param_grid={'bootstrap': [True, False], 'max_depth': [5, 10, 15],
                         'n_estimators': [500, 1000]},
             verbose=2)
```

In [122]:

```
grid_search.best_params_
```

Out[122]:

Out[122]:

```
{'bootstrap': True, 'max_depth': 15, 'n_estimators': 1000}
```

In [124]:

```
pred = grid_search.predict(X_important_test)

f1_score(Y_test, pred)
```

Out[124]:

```
0.7094635848233756
```

In [125]:

```
imp_test_features = boruta_selector.transform(np.array(new_test_data))
```

In [126]:

```
prediction = grid_search.predict(imp_test_features)
```

In [128]:

```
res = pd.DataFrame(prediction)
res.index = new_test_data.index
res.columns = ["prediction"]

res.to_csv("prediction_results_smoke_data.csv")
files.download("prediction_results_smoke_data.csv")
```