```
import pandas as pd
```

```
df = pd.read_csv('Metabolic Syndrome.csv')
```

```
df.head()
```

	seqn	Age	Sex	Marital	Income	Race	WaistCirc	BMI	Albuminuria	UrAlbCr	UricAcid	BloodGlucose	HDL	Triglycerides	Metaboli
0	62161	22	Male	Single	8200.0	White	81.0	23.3	0	3.88	4.9	92	41	84	
1	62164	44	Female	Married	4500.0	White	80.1	23.2	0	8.55	4.5	82	28	56	
2	62169	21	Male	Single	800.0	Asian	69.6	20.1	0	5.07	5.4	107	43	78	
3	62172	43	Female	Single	2000.0	Black	120.4	33.3	0	5.22	5.0	104	73	141	
4	62177	51	Male	Married	NaN	Asian	81.1	20.1	0	8.13	5.0	95	43	126	

```
df.columns
```

```
Index(['seqn', 'Age', 'Sex', 'Marital', 'Income', 'Race', 'WaistCirc', 'BMI',
       'Albuminuria', 'UrAlbCr', 'UricAcid', 'BloodGlucose', 'HDL',
       'Triglycerides', 'MetabolicSyndrome'],
      dtype='object')
```

```
df.axes
```

```
[RangeIndex(start=0, stop=2401, step=1),
 Index(['seqn', 'Age', 'Sex', 'Marital', 'Income', 'Race', 'WaistCirc', 'BMI',
        'Albuminuria', 'UrAlbCr', 'UricAcid', 'BloodGlucose', 'HDL',
        'Triglycerides', 'MetabolicSyndrome'],
       dtype='object')]
```

```
df.size
```

```
36015
```

```
df.values
```

```
array([[62161, 22, 'Male', ..., 41, 84, 0],
       [62164, 44, 'Female', ..., 28, 56, 0],
       [62169, 21, 'Male', ..., 43, 78, 0],
       ...,
       [71909, 28, 'Male', ..., 47, 84, 0],
       [71911, 27, 'Male', ..., 41, 124, 1],
       [71915, 60, 'Male', ..., 36, 226, 1]], dtype=object)
```

```
df.shape
```

```
(2401, 15)
```

```
df.describe()
```

	seqn	Age	Income	WaistCirc	BMI	Albuminuria	UrAlbCr	UricAcid	BloodGlucose	HDL
count	2401.000000	2401.000000	2284.000000	2316.000000	2375.000000	2401.000000	2401.000000	2401.000000	2401.000000	2401.000000
mean	67030.674302	48.691795	4005.253940	98.307254	28.702189	0.154102	43.626131	5.489046	108.247813	53.369429
std	2823.565114	17.632852	2954.032186	16.252634	6.662242	0.422780	258.272829	1.439358	34.820657	15.185537
min	62161.000000	20.000000	300.000000	56.200000	13.400000	0.000000	1.400000	1.800000	39.000000	14.000000
25%	64591.000000	34.000000	1600.000000	86.675000	24.000000	0.000000	4.450000	4.500000	92.000000	43.000000
50%	67059.000000	48.000000	2500.000000	97.000000	27.700000	0.000000	7.070000	5.400000	99.000000	51.000000
75%	69495.000000	63.000000	6200.000000	107.625000	32.100000	0.000000	13.690000	6.400000	110.000000	62.000000
max	71915.000000	80.000000	9000.000000	176.000000	68.700000	2.000000	5928.000000	11.300000	382.000000	156.000000

```
df.dropna()
```

	seqn	Age	Sex	Marital	Income	Race	WaistCirc	BMI	Albuminuria	UrAlbCr	UricAcid	BloodGlucose	HDL	Triglyceride
0	62161	22	Male	Single	8200.0	White	81.0	23.3	0	3.88	4.9	92	41	8
1	62164	44	Female	Married	4500.0	White	80.1	23.2	0	8.55	4.5	82	28	5
2	62169	21	Male	Single	800.0	Asian	69.6	20.1	0	5.07	5.4	107	43	7
3	62172	43	Female	Single	2000.0	Black	120.4	33.3	0	5.22	5.0	104	73	14
5	62178	80	Male	Widowed	300.0	White	112.5	28.5	0	9.79	4.8	105	47	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
2394	71895	31	Male	Married	2500.0	Asian	74.0	20.6	0	2.00	6.7	95	64	8
2395	71898	65	Female	Married	5400.0	MexAmerican	98.5	29.4	0	5.51	6.7	114	49	16
2398	71909	28	Male	Single	800.0	MexAmerican	100.8	29.4	0	2.78	6.2	99	47	8
2399	71911	27	Male	Married	8200.0	MexAmerican	106.6	31.3	0	4.15	6.2	100	41	12
2400	71915	60	Male	Single	6200.0	White	106.6	27.5	0	12.82	5.2	91	36	22

```
print(df.dtypes)
```

```
seqn                int64
Age                 int64
Sex                object
Marital            object
Income            float64
Race               object
WaistCirc         float64
BMI               float64
Albuminuria         int64
UrAlbCr           float64
UricAcid          float64
BloodGlucose        int64
HDL                 int64
Triglycerides       int64
MetabolicSyndrome   int64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder

# Example categorical data
categorical_data = ['Sex', 'Marital', 'Race']

# Create a LabelEncoder instance
label_encoder = LabelEncoder()

# Use fit_transform on the categorical data
numerical_labels = label_encoder.fit_transform(categorical_data)

# Print the result
print(numerical_labels)
```

```
[2 0 1]
```

Dividing the dataset in X and Y

```
array  = df.values


x = array[:,0:14]
y = array[:,14]


# importing libraries

from pandas import read_csv
from sklearn.model_selection import KFold,cross_val_score

from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
kfold = KFold(n_splits=10)
cart = DecisionTreeClassifier()
num_trees = 100
```

Build the model

```
x
```

```
array([[62161, 22, 'Male', ..., 92, 41, 84],
       [62164, 44, 'Female', ..., 82, 28, 56],
       [62169, 21, 'Male', ..., 107, 43, 78],
       ...,
       [71909, 28, 'Male', ..., 99, 47, 84],
       [71911, 27, 'Male', ..., 100, 41, 124],
       [71915, 60, 'Male', ..., 91, 36, 226]], dtype=object)
```

```
y
```

```
array([0, 0, 0, ..., 0, 1, 1], dtype=object)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split
```

```
df.head()
```

	seqn	Age	Sex	Marital	Income	Race	WaistCirc	BMI	Albuminuria	UrAlbCr	UricAcid	BloodGlucose	HDL	Triglycerides	Metaboli
0	62161	22	Male	Single	8200.0	White	81.0	23.3	0	3.88	4.9	92	41	84	
1	62164	44	Female	Married	4500.0	White	80.1	23.2	0	8.55	4.5	82	28	56	
2	62169	21	Male	Single	800.0	Asian	69.6	20.1	0	5.07	5.4	107	43	78	
3	62172	43	Female	Single	2000.0	Black	120.4	33.3	0	5.22	5.0	104	73	141	
4	62177	51	Male	Married	NaN	Asian	81.1	20.1	0	8.13	5.0	95	43	126	

```
df =read_csv('Metabolic Syndrome.csv')
```

```
X = df.iloc[:,0:14]
Y = df.iloc[:,14]
```

```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.4)
```

```
print(X.shape)
```

```
(2401, 14)
```

```
print(y_train.shape)
```

```
(1440,)
```

```
X_train = pd.get_dummies(X_train, columns=['Sex'], prefix=['Sex'])
```

```
X_train = pd.get_dummies(X_train, columns=['Marital'], prefix=['Marital'])
```

```
X_train = pd.get_dummies(X_train, columns=['Race'], prefix=['Race'])
```

```
import pandas as pd
```

```
new_df = df.dropna()
```

```
new_df.head(10)
```

	seqn	Age	Sex	Marital	Income	Race	WaistCirc	BMI	Albuminuria	UrAlbCr	UricAcid	BloodGlucose	HDL	Triglycerides
0	62161	22	Male	Single	8200.0	White	81.0	23.3	0	3.88	4.9	92	41	84
1	62164	44	Female	Married	4500.0	White	80.1	23.2	0	8.55	4.5	82	28	56
2	62169	21	Male	Single	800.0	Asian	69.6	20.1	0	5.07	5.4	107	43	78
3	62172	43	Female	Single	2000.0	Black	120.4	33.3	0	5.22	5.0	104	73	141
5	62178	80	Male	Widowed	300.0	White	112.5	28.5	0	9.79	4.8	105	47	100
6	62184	26	Male	Single	9000.0	Black	78.6	22.1	0	9.21	5.4	87	61	40
7	62189	30	Female	Married	6200.0	Asian	80.2	22.4	0	8.78	6.7	83	48	91
11	62202	36	Male	Married	8200.0	MexAmerican	97.0	24.7	1	62.14	6.7	94	58	182
12	62205	28	Male	Single	9000.0	White	106.0	28.9	0	7.24	8.8	101	40	129
13	62208	38	Male	Married	3500.0	Hispanic	82.7	22.2	0	5.53	4.5	91	46	62

```
from sklearn.impute import SimpleImputer

# Create an imputer
imputer = SimpleImputer(strategy='mean')  # You can choose other strategies like 'median' or 'most_frequent'

# Fit and transform the imputer on your data
X_train_imputed = imputer.fit_transform(X_train)

# Now, you can use X_train_imputed for model training


X_train.dropna(inplace=True)
y_train = y_train[X_train.index]  # Make sure to align the target variable


sel = SelectFromModel(RandomForestClassifier(n_estimators = 100,max_features = 4))
sel.fit(X_train,y_train)
```

```
SelectFromModel(estimator=RandomForestClassifier(max_features=4))
estimator: RandomForestClassifier
RandomForestClassifier(max_features=4)
RandomForestClassifier
RandomForestClassifier(max_features=4)
```

```
sel.get_support()
```

```
array([False,  True, False,  True,  True, False,  True,  True,  True,
        True,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False])
```

```
selected_feat= X_train.columns[(sel.get_support())]
len(selected_feat)
```

```
8
```

```
print(selected_feat)
```

```
Index(['Age', 'WaistCirc', 'BMI', 'UrAlbCr', 'UricAcid', 'BloodGlucose', 'HDL',
       'Triglycerides'],
      dtype='object')
```