ASSIGNMENT 5 - INTRO TO MACHINE LEARNING | Decision Tree, Bagging, Random Forest and Boosting

> **FULL MARKS = 100**

Note: To submit the assignment, please follow the same steps and in assignments 1, 2, 3, 4.

In this assignment we will cover the following topics:

1. **Decision Tree with KFold Cross Validation| SCORE : 40**

   References

   > https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
   >
   > https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressc
   >
   > https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

1. **Decision Tree with Bagging | SCORE : 20**

   References

   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegres
   >
   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html#sklearn.ensemble.BaggingClassifie

2. **Random Forest | SCORE : 20**

   References

   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomF
   >
   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomF

3. **Boosting | SCORE : 20**

   References

   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.Gradie
   >
   > https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.Gradier

1. Decision Tree with KFord Cross Validation

EXERCISE NO. 1.1 | Score :40

In [21]:
```
# Required Dataset

from sklearn.datasets import load_diabetes

# In the following exercises you will use the diabetes datasets for regression problems.
# Use your experience from previous exercises on how to use sklearn model, load data and do visualization.
# Please try to research about the datasets diabetes dataset

#Load necessary libraries for you
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import numpy as np

#Load the diabetes dataset
diabetes = load_diabetes()
#split the dataset with test_size = 0.20
```

```
X = diabetes.data
y = diabetes.target
regXtrain, regXtest, regYtrain, regYtest = train_test_split(X, y, test_size = 0.2)

print(f"Regression data : diabetes data")
print(f"xtrainShape : {regXtrain.shape}, ytrainShape : {regYtrain.shape}, xtestShape : {regXtest.shape}, ytestS
```

```
Regression data : diabetes data
xtrainShape : (353, 10), ytrainShape : (353,), xtestShape : (89, 10), ytestShape : (89,)
```

In [24]:

```
#You may print necessary dataset information to help you better understand this dataset

print(diabetes)
```

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990749, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06833155, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286131, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04688253,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452873, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00422151,  0.00306441]]), 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310
., 101.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTe
n baseline variables, age, sex, body mass index, average blood\npressure, and six blood serum measurements were
obtained for each of n =\n442 diabetes patients, as well as the response of interest, a\nquantitative measure o
f disease progression one year after baseline.\n\n**Data Set Characteristics:**\n\n:Number of Instances: 442\n\
n:Number of Attributes: First 10 columns are numeric predictive values\n\n:Target: Column 11 is a quantitative
measure of disease progression one year after baseline\n\n:Attribute Information:\n    - age     age in years\n
- sex\n    - bmi     body mass index\n    - bp      average blood pressure\n    - s1      tc, total serum chole
sterol\n    - s2      ldl, low-density lipoproteins\n    - s3      hdl, high-density lipoproteins\n    - s4
tch, total cholesterol / HDL\n    - s5      ltg, possibly log of serum triglycerides level\n    - s6      glu,
blood sugar level\n\nNote: Each of these 10 feature variables have been mean centered and scaled by the standar
d deviation times the square root of `n_samples` (i.e. the sum of squares of each column totals 1).\n\nSource U
RL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley Efron, Tre
vor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with di
scussion), 407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n', 'feature_names': ['
age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'], 'data_filename': 'diabetes_data_raw.csv.gz', 'ta
rget_filename': 'diabetes_target.csv.gz', 'data_module': 'sklearn.datasets.data'}
```

In [26]:

```
# Use the bosting diabetes dataset to achieve the following tasks
# An example link
# https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html#sphx-glr-auto-examples-tree-plot

# Tasks:
```

```
# 1. Create four decision tree regressor models, use max_depth size = 1,2,5,100
# 2. Fit model. You will have 4 different models for different depth sizes
# 3. Calculate the mean squared error of each model's prediction
# 4. Find the best max_depth

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

#create decision tree regressor
regr_1 = DecisionTreeRegressor(max_depth = 1)
regr_2 = DecisionTreeRegressor(max_depth = 2)
regr_3 = DecisionTreeRegressor(max_depth = 5)
regr_4 = DecisionTreeRegressor(max_depth = 100)

#fit regression model
regr_1.fit(regXtrain, regYtrain)
regr_2.fit(regXtrain, regYtrain)
regr_3.fit(regXtrain, regYtrain)
regr_4.fit(regXtrain, regYtrain)

#predict
y_1 = regr_1.predict(regXtest)
y_2 = regr_2.predict(regXtest)
y_3 = regr_3.predict(regXtest)
y_4 = regr_4.predict(regXtest)

#calculate mean_squared_error
print(mean_squared_error(regYtest, y_1))
print(mean_squared_error(regYtest, y_2))
print(mean_squared_error(regYtest, y_3))
print(mean_squared_error(regYtest, y_4))
```

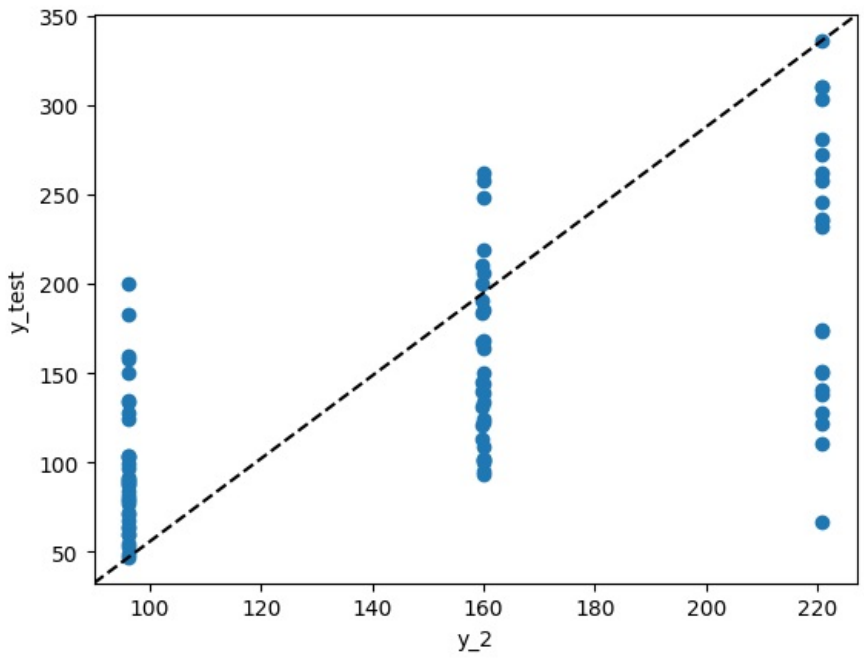
```
3634.6321565138746
2841.5483452967055
4449.8318549829755
6431.91011235955
```

In [27]:

```
plt.scatter(y_2,
            regYtest,
            label = 'medv')

plt.plot([0, 1],
         [0, 1],
         '--k',
         transform = plt.gca().transAxes)

plt.xlabel('y_2')
plt.ylabel('y_test')
```

Out[27]:

```
Text(0, 0.5, 'y_test')
```



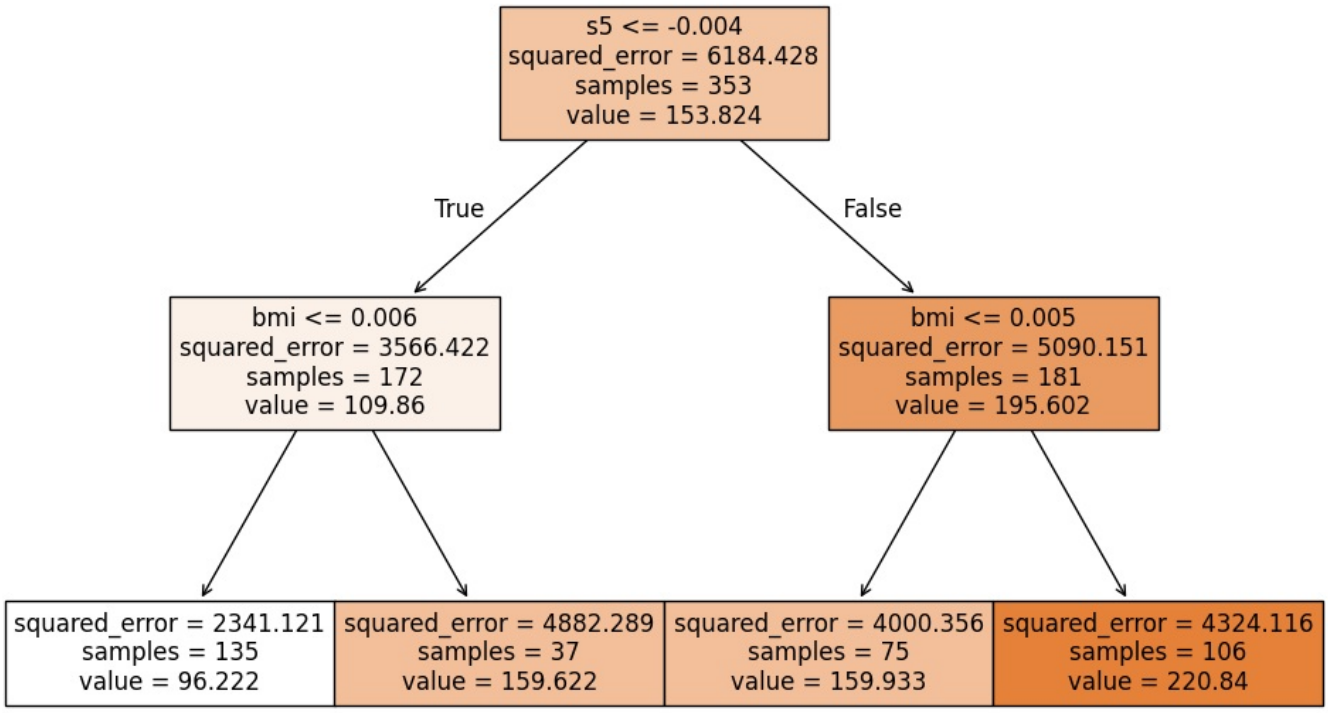
In [28]:

```
# Finally pick the best max_depth you got
# Use this max_depth, and use cross_val_score and fit your model with k = 10 fold size
# Calculate average scores in kfold
'''
Best max_depth = 2

'''

from sklearn.model_selection import KFold, cross_val_score
```

```
kfold = KFold(n_splits=10)
scores = cross_val_score(regr_2, diabetes.data, diabetes.target, cv=kfold)
print("Cross-validation scores: {}".format(scores))
print("Average cross-validation score: {:.2f}".format(scores.mean()))
```

```
Cross-validation scores: [0.28897364 0.16168834 0.30122639 0.48723453 0.30735793 0.5135879
 0.29810844 0.01757394 0.15153891 0.55875239]
Average cross-validation score: 0.31
```

In [30]:
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# If regr_4 is a trained DecisionTreeRegressor (or DecisionTreeClassifier), just do:
# NOTE: replace the regr_4 with the regr model with the lowest MSE
plt.figure(figsize=(12, 8))  # optional, sets a larger figure size
plot_tree(regr_2,
          feature_names=diabetes.feature_names,
          filled=True)
plt.show()
```



In [31]:
```
# Answer the questions:

'''
1. What is the most important indicator that indicate diabetes?
'''

'''
Your answer goes here:
BMI being less than or equal to 0.006


'''
print()
```

2. Decision Tree with Bagging

EXERCISE NO. 2.1 | Score :20

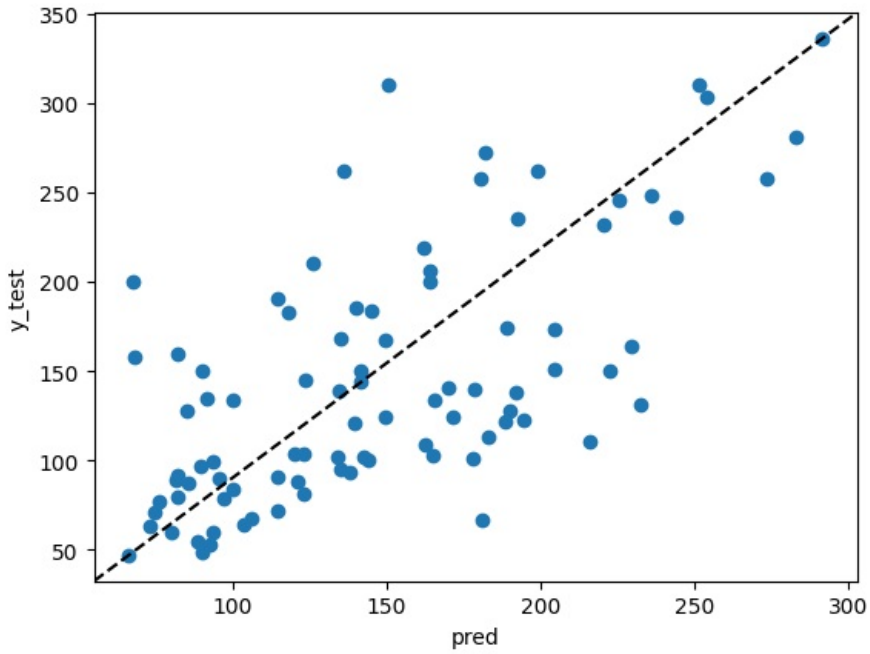
In [32]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

from sklearn.ensemble import BaggingRegressor

# Use BaggingRegressor to fit the training data
# Calculate the mean squared error
```

```
#load BaggingRegressor model and pass n_estimators=20, random_state=1
bagged_regr = BaggingRegressor(estimator=DecisionTreeRegressor(), n_estimators=20, random_state=1)
bagged_regr.fit(regXtrain, regYtrain)
pred = bagged_regr.predict(regXtest)
mean_squared_error(regYtest, pred)
```

Out[32]:
```
2829.643848314607
```

In [33]:
```
plt.scatter(pred,
            regYtest,
            label = 'medv')

plt.plot([0, 1],
         [0, 1],
         '--k',
         transform = plt.gca().transAxes)

plt.xlabel('pred')
plt.ylabel('y_test')
```

Out[33]:
```
Text(0, 0.5, 'y_test')
```



In [36]:
```
# Answer the following questions:
'''
1. Does Begging improve the MSE(mean squared error) from a single decision tree with best max_depth, and why?
2. Comparing to the single decision tree scatter plot, what do you observed from the begging decision trees sca
'''

'''
Your answer goes here:
Yes, the mean squared error has improved by indroducing randomness in each base regressor then averaging,
the regression has less chances to cause variance
The bagging decision tree scatter plot actually resembles a scatter plot over the single decision tree scatter
'''
print()
```

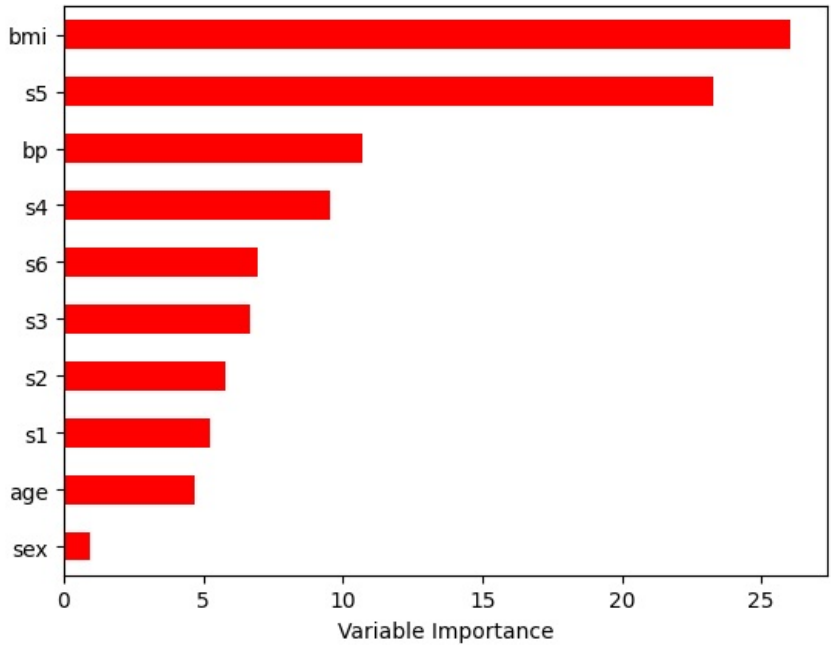
3. Random Forest

EXERCISE NO. 3.1 | Score :20

In [35]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

from sklearn.ensemble import RandomForestRegressor

# Use RandomForestRegressor to fit the training data
# Calculate the mean squared error

# load RandomForestRegressor model and pass max_features=6, n_estimater=20, random_state=1
random_forest_regr = RandomForestRegressor(max_features=6, n_estimators=20, random_state=1)
random_forest_regr.fit(regXtrain, regYtrain)
pred = random_forest_regr.predict(regXtest)

mean_squared_error(regYtest, pred)
```

Out[35]:
```
2751.859606741573
```

In [37]:
```
# Plot the importance rank of each feature in the diabeties dataset
import pandas as pd
Importance = pd.DataFrame({'Importance':random_forest_regr.feature_importances_*100},
                          index = diabetes.feature_names)

Importance.sort_values(by = 'Importance',
                       axis = 0,
                       ascending = True).plot(kind = 'barh',
                                              color = 'r', )

plt.xlabel('Variable Importance')
plt.gca().legend_ = None
```



In [38]:
```
#Answer the following questions:
'''
1. Does Random Forest improve the MSE(mean squared error) from a single decision tree with best max_depth and B
2. What is the most important indicator that indicate diabetes?
'''

'''
Your answer goes here:
Yes, because we introduce randomness in each tree node further reducing variance and over-fitting
The most important indicator is BMI
'''
print()
```

4. Boosting

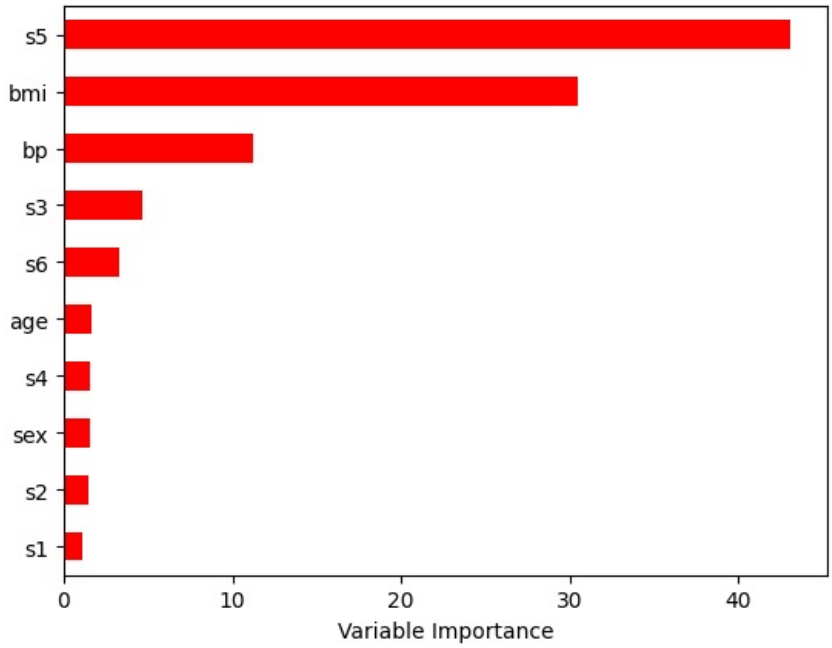
EXERCISE NO. 4.1 | Score :20

In [40]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
from sklearn.ensemble import GradientBoostingRegressor

# Use GradientBoostingRegressor to fit the training data
# Calculate the mean squared error

# load GradientBoostingRegressor model and pass n_estimators = 500, learning_rate = 0.01, max_depth = 2, random
boosted_regr = GradientBoostingRegressor(n_estimators = 500, learning_rate = 0.01, max_depth = 2, random_state
boosted_regr.fit(regXtrain, regYtrain)
pred = boosted_regr.predict(regXtest)

mean_squared_error(regYtest, pred)
```

Out[40]:
```
2490.4587852242803
```

In [41]:
```
feature_importance = boosted_regr.feature_importances_*100

rel_imp = pd.Series(feature_importance,
                    index = diabetes.feature_names).sort_values(inplace = False)

rel_imp.T.plot(kind = 'barh',
               color = 'r', )
```

```
plt.xlabel('Variable Importance')

plt.gca().legend_ = None
```



In [42]:
```
#Answer the following questions:
'''
1. Does Boosting improve the MSE(mean squared error) from a single decision tree with best max_depth, Bagging,
2. What is the most important indicator that indicate diabetes?
'''

'''
Your answer goes here:
It does improve the MSE by creating new training sets from misclassified trees to build upon errors, learning f
The importance now is s5 over bmi previously
'''
print()
```

In [ ]:
```

```

In [ ]:
```

```