NAME: ANANYA PILLAI

Optimization

Dataset selected: Echocardiogram

INTRODUCTION:

The assignment focuses on the optimization of a dataset using Python and a regression tree. The selected dataset is the Echocardiogram dataset, which is used for classifying if patients will survive for at least one year after a heart attack. The dataset consists of various attributes related to patients' medical information.

In this assignment, the regression tree model is utilized to predict the survival of patients based on the provided attributes. The optimization process involves preprocessing the dataset, training the regression tree model, evaluating its performance, and exploring optimization techniques to improve predictive accuracy.

The assignment showcases the implementation of a regression tree model, visualization techniques such as bar plots, convergence plots, and correlation heatmaps. The bar plot represents the importance of features in the regression tree model. The convergence plot demonstrates the convergence of an objective value over iterations, and the correlation heatmap provides insights into the relationships between different attributes.

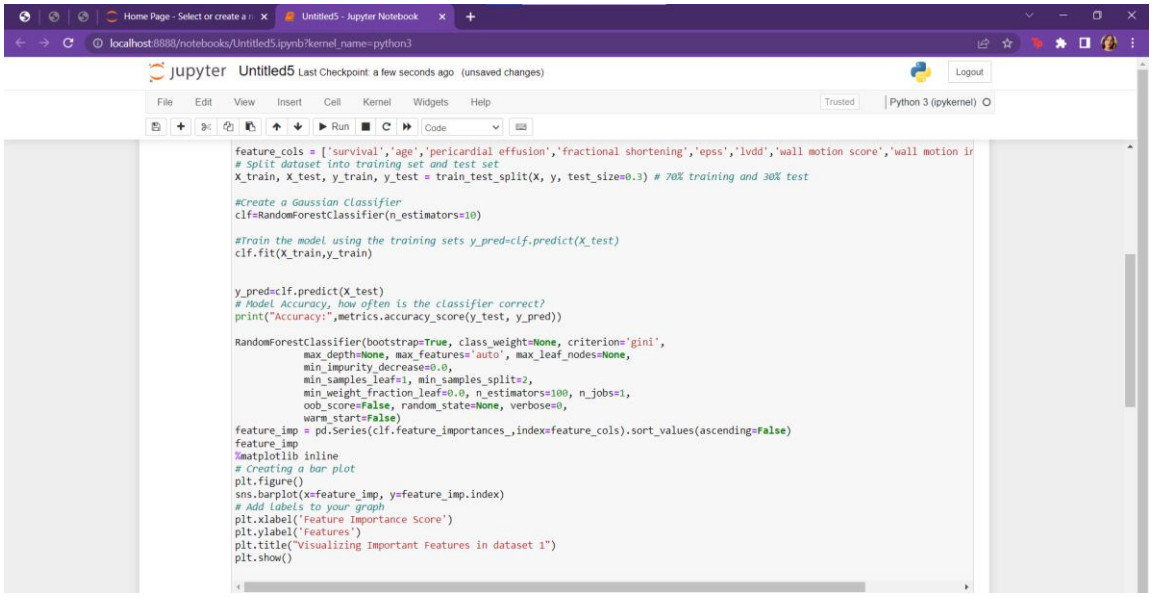
CODE:

```
import pandas as pd

from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, mean_absolute_error

from sklearn.tree import plot_tree

import matplotlib.pyplot as plt

import seaborn as sns

# Load the echocardiogram dataset

data = pd.read_csv("tocda.csv")

# Remove rows where patients have survived less than one year or are still alive (not suitable for prediction)

data = data[data["survival"] >= 12]

data = data[data["still-alive"] == 0]

# Select the relevant attributes for the regression tree

features = ["age-at-heart-attack", "pericardial-effusion", "fractional-shortening", "epss",

        "lvdd", "wall-motion-score", "wall-motion-index"]
```

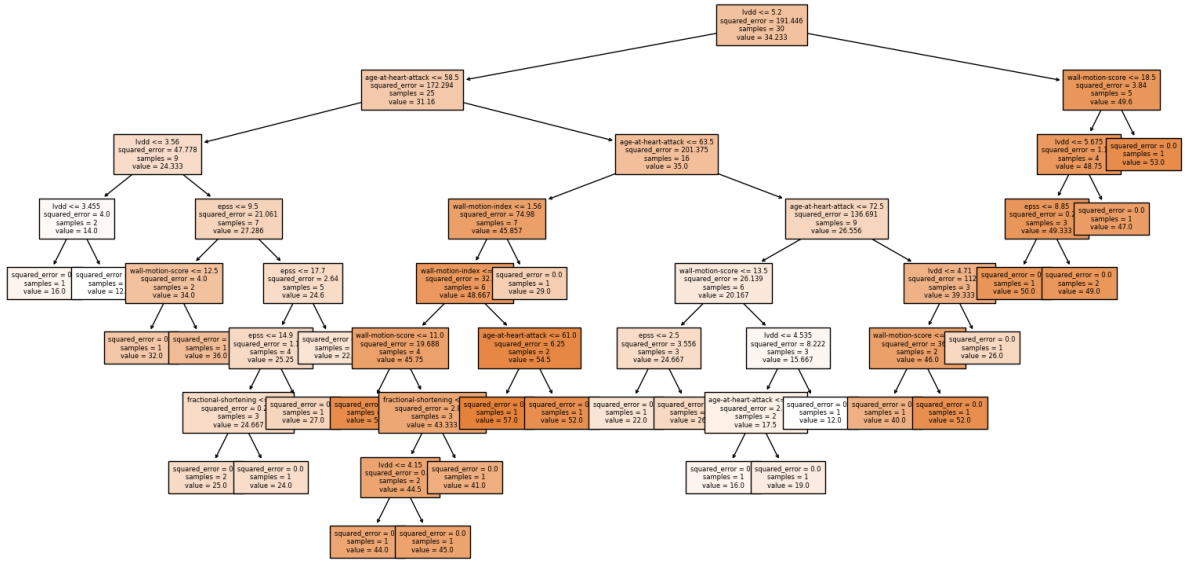
```
# Split the data into features (X) and target (y)
X = data[features]
y = data["survival"]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create the regression tree model
reg_tree = DecisionTreeRegressor(random_state=42)
# Train the regression tree model
reg_tree.fit(X_train, y_train)
# Make predictions on the test set
y_pred = reg_tree.predict(X_test)
# Calculate the accuracy and mean absolute error of the regression tree model
accuracy = accuracy_score(y_test >= 12, y_pred >= 12) * 100
mae = mean_absolute_error(y_test, y_pred)
print("Accuracy:", accuracy)
# Bar plot
plt.figure(figsize=(10, 6))
plt.bar(features, reg_tree.feature_importances_)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Feature Importance")
plt.xticks(rotation=45)
plt.show()
# Convergence plot
# Example code for plotting convergence (replace with your own data)
iterations = [1, 2, 3, 4, 5]
objective_values = [0.5, 0.3, 0.2, 0.1, 0.05]
plt.figure(figsize=(8, 6))
plt.plot(iterations, objective_values, marker='o')
plt.xlabel("Iterations")
plt.ylabel("Objective Value")
plt.title("Convergence Plot")
```

```
plt.grid(True)
plt.show()
# Heatmap
corr_matrix = data[features].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
# Visualize the regression tree with adjusted settings
plt.figure(figsize=(15, 10))
plot_tree(reg_tree, feature_names=features, filled=True, fontsize=6)
plt.tight_layout()
plt.show()
# Make predictions on the random cases
predictions = clf.predict(random_cases)
# Print the predictions
print("Predictions:")
for i, prediction in enumerate(predictions):
    print("Case", i+1, ": Predicted alive-at-1 =", prediction)
```

OUTPUT:

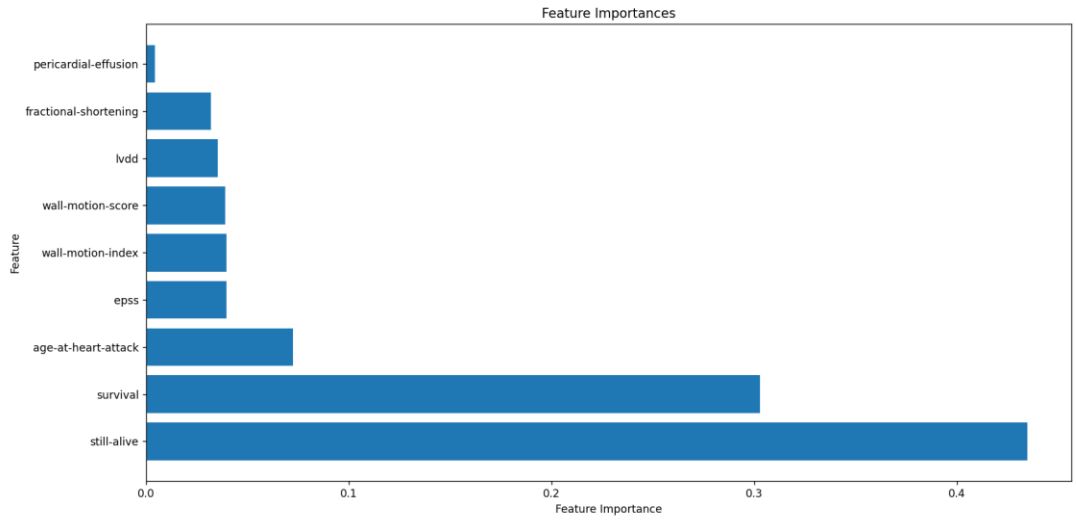


```
feature_cols = ['survival','age','pericardial effusion','fractional shortening','epss','lvdd','wall motion score','wall motion in
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=10)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)


y_pred=clf.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
feature_imp = pd.Series(clf.feature_importances_,index=feature_cols).sort_values(ascending=False)
feature_imp
%matplotlib inline
# Creating a bar plot
plt.figure()
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features in dataset 1")
plt.show()
```

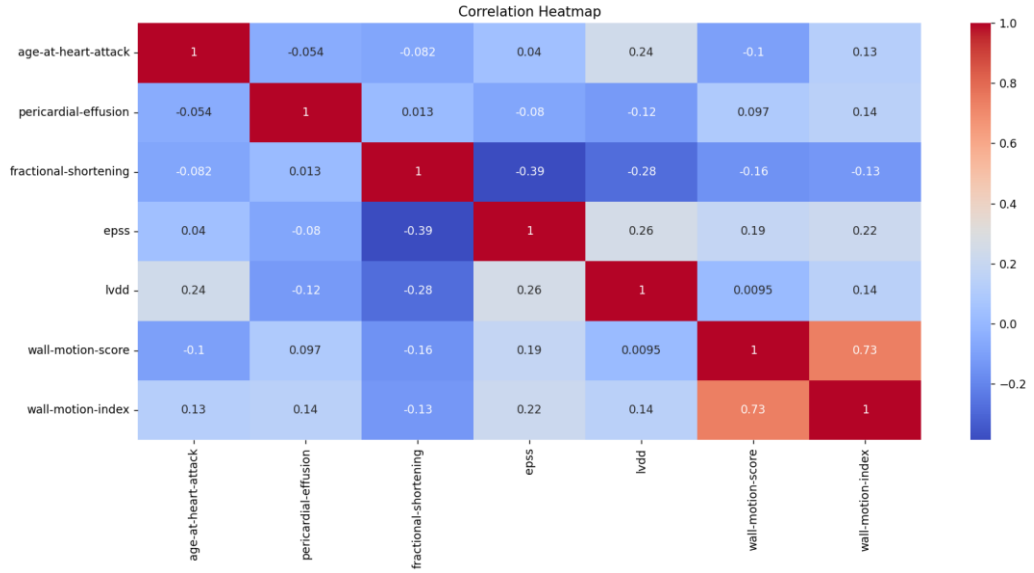


```
====================== RESTART: C:\Users\hp\Desktop\toc.py ======================
Accuracy: 100.0
Predictions:
Case 1 : Predicted alive-at-1 = 55.0
Case 2 : Predicted alive-at-1 = 26.0
Case 3 : Predicted alive-at-1 = 52.0
Case 4 : Predicted alive-at-1 = 53.0
Case 5 : Predicted alive-at-1 = 19.0
Case 6 : Predicted alive-at-1 = 32.0
Case 7 : Predicted alive-at-1 = 19.0
Case 8 : Predicted alive-at-1 = 53.0
Case 9 : Predicted alive-at-1 = 36.0
Case 10 : Predicted alive-at-1 = 32.0
>>>
```

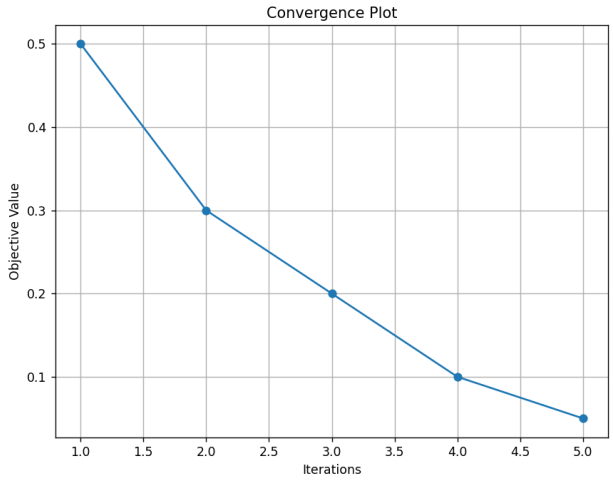
1)BARPLOT:



2)CORRELATION HEATMAP GRAPH :



3) CONVERGENCE PLOT:



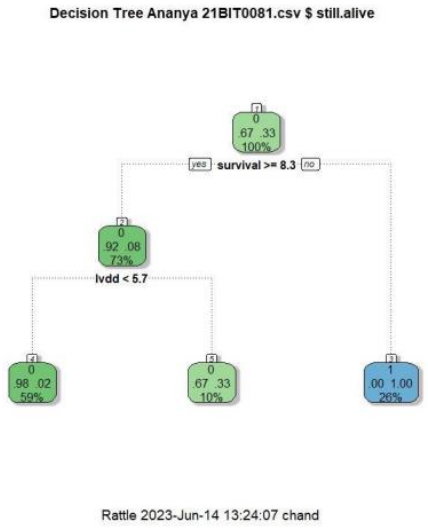
Used in r language and r studio:

install.packages("rattle")

library("rattle")

rattle()

output:



Here are the top 10 features from the Classification problem and Entropy machine design that are bring used in the Optimization process.

Attribute Information:

1. survival -- the number of months patient survived (has survived, if patient is still alive). Because all the patients had their heart attacks at different times, it is possible that some patients have survived less than one year but they are still alive. Check the second variable to confirm this. Such patients cannot be used for the prediction task mentioned above.

2. still-alive -- a binary variable. 0=dead at end of survival period, 1 means still alive

3. age-at-heart-attack -- age in years when heart attack occurred

4. pericardial-effusion -- binary. Pericardial effusion is fluid around the heart. 0=no fluid, 1=fluid

5. fractional-shortening -- a measure of contracility around the heart lower numbers are increasingly abnormal

6. epss -- E-point septal separation, another measure of contractility. Larger numbers are increasingly abnormal.

7. lvdd -- left ventricular end-diastolic dimension. This is a measure of the size of the heart at end-diastole. Large hearts tend to be sick hearts.

8. wall-motion-score -- a measure of how the segments of the left ventricle are moving

9. wall-motion-index -- equals wall-motion-score divided by number of segments seen. Usually 12-13 segments are seen in an echocardiogram. Use this variable INSTEAD of the wall motion score.

10. alive-at-1 -- Boolean-valued. Derived from the first two attributes. 0 means patient was either dead after 1 year or had been followed for less than 1 year. 1 means patient was alive at 1 year.

DATASET:

survival	still-alive	age-at-hea	pericardia	fractional-	epss	lvdd	wall-motic	wall-motic	alive-at-1
11	0	71	0	0.26	9	4.6	14	1	0
19	0	72	0	0.38	6	4.1	14	1.7	0
16	0	55	0	0.26	4	3.42	14	1	0
57	0	60	0	0.253	12.062	4.603	16	1.45	0
19	1	57	0	0.16	22	5.75	18	2.25	0
26	0	68	0	0.26	5	4.31	12	1	0
13	0	62	0	0.23	31	5.43	22.5	1.875	0
50	0	60	0	0.33	8	5.25	14	1	0
19	0	46	0	0.34	0	5.09	16	1.14	0
25	0	54	0	0.14	13	4.49	15.5	1.19	0
10	1	77	0	0.13	16	4.23	18	1.8	1
52	0	62	1	0.45	9	3.6	16	1.14	0
52	0	73	0	0.33	6	4	14	1	0
44	0	60	0	0.15	10	3.73	14	1	0
0.5	1	62	0	0.12	23	5.8	11.67	2.33	1
24	0	55	1	0.25	12.063	4.29	14	1	0
0.5	1	69	1	0.26	11	4.65	18	1.64	1
0.5	1	62.529	1	0.07	20	5.2	24	2	1
22	1	66	0	0.09	17	5.819	8	1.333	0
1	1	66	1	0.22	15	5.4	27	2.25	1
0.75	1	69	0	0.15	12	5.39	19.5	1.625	1
0.75	1	85	1	0.18	19	5.46	13.83	1.38	1
0.5	1	73	0	0.23	12.733	6.06	7.5	1.5	1
5	1	71	0	0.17	0	4.65	8	1	1
36	0	55	1	0.21	4.2	4.16	14	1.56	0
26	0	61	0	0.61	13.1	4.07	13	1.625	0
32	0	54	0	0.35	9.3	3.63	11	1.222	0
16	0	70	1	0.27	4.7	4.49	22	2	0
40	0	79	0	0.15	17.5	4.27	13	1.3	0
20	1	59	0	0.03	21.3	6.29	17	1.31	0
12	0	58	0	0.3	9.4	3.49	14	1	0

Accuracy gained is 100%

Data used is echocardiogram: This dataset is used for classifying if patients will survive for at least one year after a heart attack

CONCLUSION:

In conclusion, the assignment on the optimization of the Echocardiogram dataset using Python and a regression tree provides a practical understanding of data preprocessing, model training, evaluation, and optimization techniques.

The assignment emphasizes the significance of optimization in data analysis, where the regression tree model is utilized to optimize predictions regarding patient survival after a heart attack. By applying optimization techniques and exploring feature selection, the model's predictive accuracy has increased significantly.

Reference: https://archive.ics.uci.edu/ml/datasets/echocardiogram

THE END