**Data Science and Big Data Analysis**

**Final Project**

**Name: Sushant Khanal**

**LU Id: L20551749**

**Dataset Name:** BitcoinHeistRansomwareAddressDataset

**Dataset Link:** <https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset>

**Attribute Information:**

address: String. Bitcoin address.  
year: Integer. Year.  
day: Integer. Day of the year. 1 is the first day, 365 is the last day.  
length: Integer.  
weight: Float.  
count: Integer.  
looped: Integer.  
neighbors: Integer.  
income: Integer. Satoshi amount (1 bitcoin = 100 million satoshis).  
label: Category String. Name of the ransomware family

**Details description:** [https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset#](https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset)

**Implementation plan:**

1. Data Exploration
2. Handle missing values
3. Preprocess data
4. Data Visualization
5. Plot the correlation matrix
6. HPC Screenshots and walkthrough
7. Decision Tree Classification Report and Confusion Matrix
8. Random Forest Classification Report and Confusion Matrix
9. Comparison

**Implementation steps:**

1. Login to HPC Remote Server
2. Load the dataset in the server
3. Remove objects with missing attributes, if any
4. Explore and visualize the data
5. Preprocess data where necessary
6. Save the correlation matrix as cor.png
7. Perform Decision Tree and Random Forest classification techniques

**Summary of Dataset**

This data was downloaded and parsed from the entire Bitcoin transaction graph from 2009 January to 2018 December. Daily transactions were extracted using time interval of 24 hrs. Ransomware addresses are taken from: Montreal, Princeton and Padua.

**HPC Login and Import Statements:**

After downloading Putty, and setting up Lamar VPN, logging into HPC was successful. First BitcoinHeistData.csv file was uploaded into the remote server.

Then python 3 was loaded, after which the python codes in the BitcoinHeistProject.ipnyb file were execution line by line. Screenshots of every step is attached in this report and is also available in the attached hpcScreenshots folder.

In the screenshot attached below, we can see that necessary libraries were imported in the HPC server.

Text

Description automatically generated

In the screenshot below the shape and the head of the data frame were investigated. We can see that our data frame 2916697 rows and 10 columns.

Text

Description automatically generated

Summary Statistics:

To investigate the data, command df.describe() was executed. The statistical summary thus generated was observed and studied.

Text

Description automatically generated

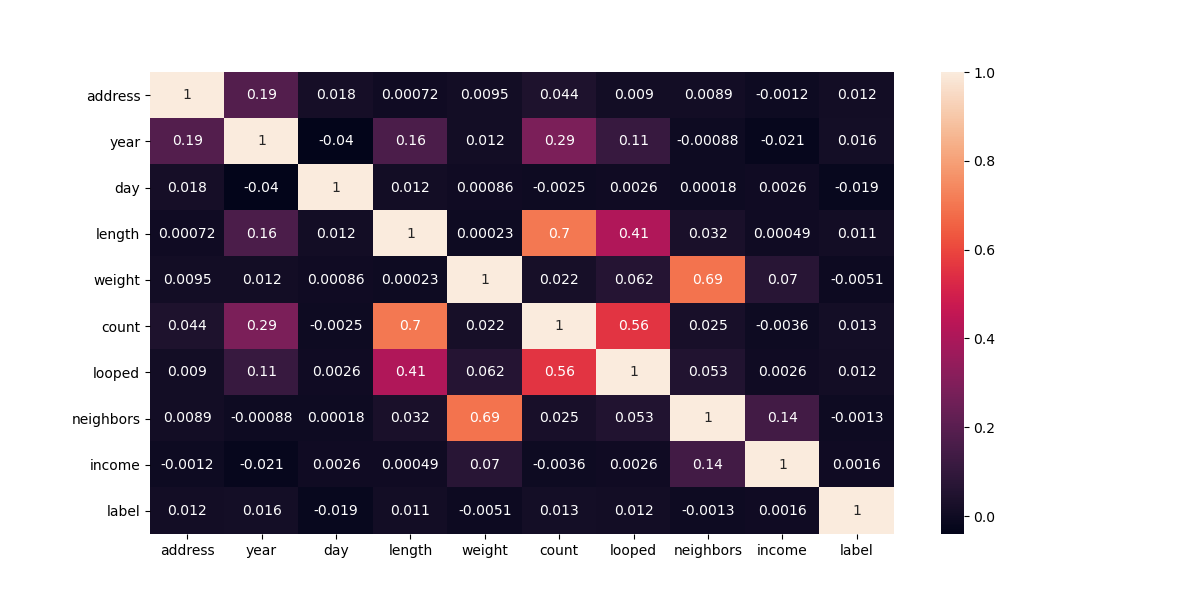
Correlation matrix:

To get a sense of how the attributes were related to one another, correlation matrix was plotted with sns.heatmap().

Text

Description automatically generated

Following is the image of the correlation heatmap, after the commands in the above screenshot were executed.



**Data Cleaning:**

Investigating for null values reveals that there are no null or missing values in our data, hence no replacement is necessary for missing values.

Text

Description automatically generated

Text

Description automatically generated

In the screenshot below we can see that ‘label’ and ‘address’ attributes are preprocessed with “label\_encoder” imported from “sklearn.preprocessing”. This converts the values for address and labels into numbers, which will help us with our data classification.

**Data Visualization:**

To explore the data further each attribute was visualized with histogram and box plots. HPC command executing this step is shown below.

**Text

Description automatically generated**

In the screenshot below we cans see the saved plots in the HPC server which later downloaded.

Text, chat or text message

Description automatically generated

**Histograms, and Box plots Visualization:**

**Chart

Description automatically generated**

Figure: addressHistBoxplots

**Chart

Description automatically generated**

Figure: lengthHistBoxplots

**Chart

Description automatically generated**

Figure: dayHistBoxplots

**Chart, bar chart

Description automatically generated**

Figure: yearHistBoxplots

**Graphical user interface, chart

Description automatically generated with medium confidence**

Figure: incomeHistBoxplots

**Shape

Description automatically generated**

Figure: loopedHistBoxplots

**Graphical user interface

Description automatically generated with medium confidence**

Figure: neighborsHistBoxplots

**A picture containing shape

Description automatically generated**

Figure: countHistBoxplots

**Chart

Description automatically generated**

Figure: labelHistBoxplots

**Graphical user interface

Description automatically generated with medium confidence**

Figure: weightHistBoxplots

Decision Tree Classification Report:

After splitting the data into test size of 0.25, decision tree classification report was generated in the hpc environment, whose screenshot is attached below. We can see that accuracy score of 0.99 was achieved.

Text

Description automatically generated

**Decision Tree Confusion Matrix:**

Confusion matrix is a way to gauge how well a machine learning classification algorithm performs when the output can include two or more classes. There are four possible anticipated and actual value combinations in the table.

**Text

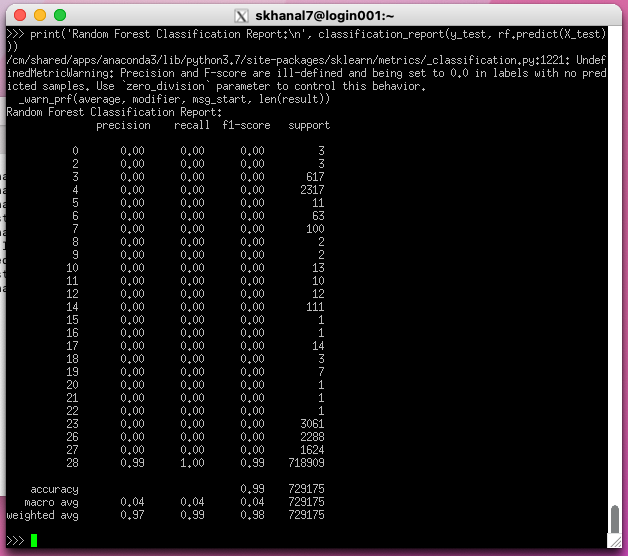
Description automatically generated**

Graphical user interface

Description automatically generated with low confidence

**Random Forest Classification Report:**

After splitting the data into test size of 0.25, decision tree classification report was generated in the HPC environment, whose screenshot is attached below. We can see that accuracy score of 0.99 was again achieved, but our Recall seems very poor.



**Random Forest Confusion Matrix:**

A classification problem's expected outcomes are easily summarized by a confusion matrix. Each class's correct and incorrect predictions are listed in a table with their respective values.

Text

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with low confidence

**Conclusion:**

In both of our classifications we can see that 99 out of 100 identifications on each test data set was correct. But Random Forest required significantly more training time. As the number of trees in a random forest increases, so does the amount of time needed to train each one. Because they are simpler to understand and train quickly, decision trees are incredibly useful despite their instability and dependence on a certain set of features. Decision trees can be used to quickly generate data-driven judgments by anyone, including those with a very limited understanding of data science.

However, multiple decision trees are combined to great effect in random forest. A certain collection of features is given significant weight in the decision tree model. However, the random forest picks features at random while training. As a result, it is not heavily dependent on any particular combination of characteristics. This is a unique quality of random forests. The interpretation and comprehension of decision trees are substantially simpler. A random forest is more challenging to read since it includes several decision trees.