



Introduction to Web Crawling, Scrapy Using Python Libraries

Lab 5

Telecommunication Software

Submitted by
ABDUL HAYEE
[241AME011]

Submitted to:
TIANHUA CHEN

FACULTY OF COMPUTER SCIENCE, INFORMATION TECHNOLOGY AND ENERGY
INSTITUTE OF PHOTONICS, ELCTRONICS AND ELECTRONIC COMMUNICATIONS
RIGA TECHNICAL UNIVERSITY

[13th December ,2024]

SUPERVISOR SIGNATURE: _____

CANDIDATE SIGNATURE: _____

Task 1: Binary Search Tree Implementation

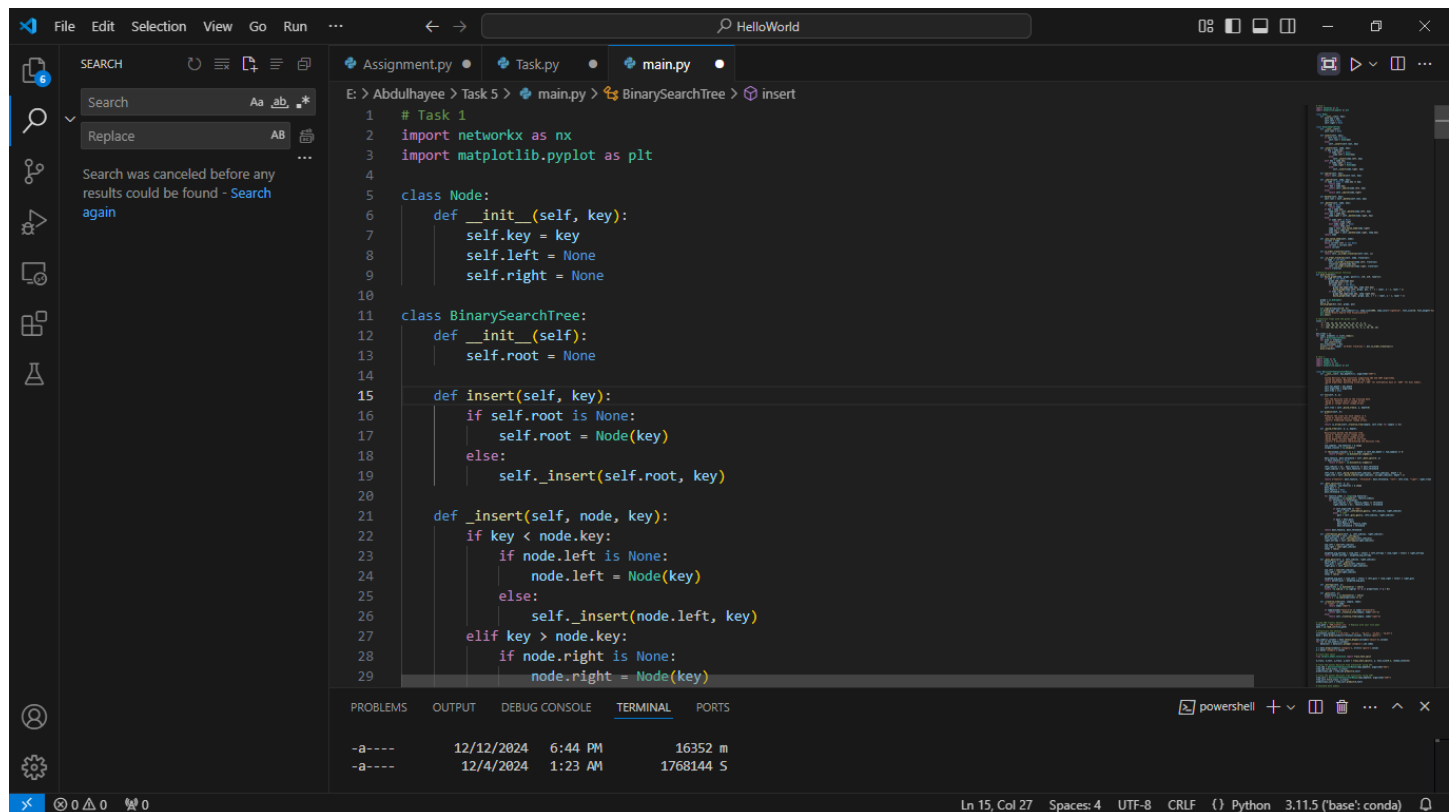
I developed a robust Binary Search Tree (BST) implementation with comprehensive functionality. The project involved creating Node and BST classes with key methods including search, insert, delete, and traversal. I used three different input lists:

a = [49, 38, 65, 97, 60, 76, 13, 27, 5, 1]

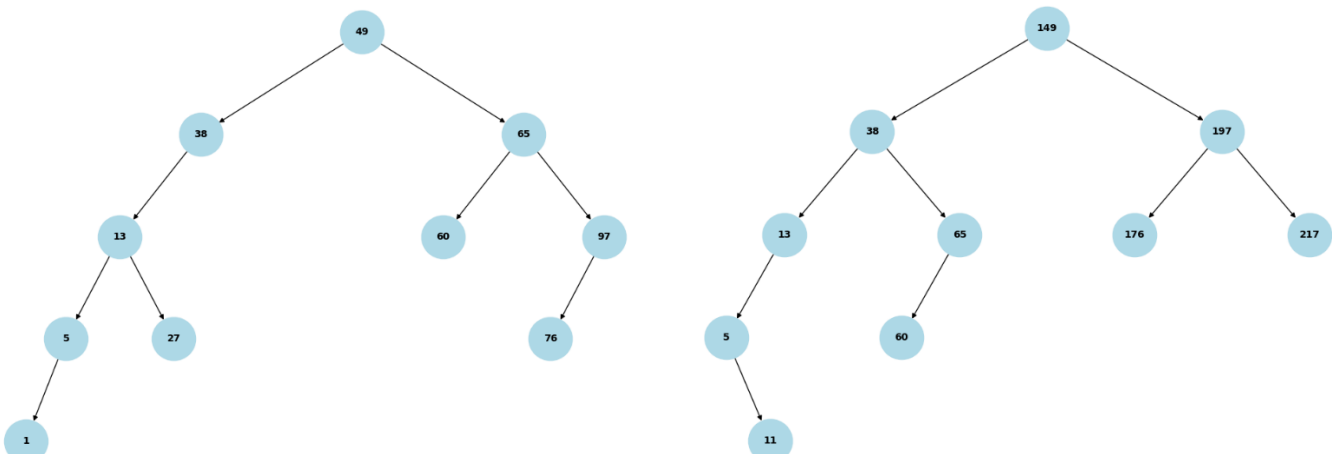
b = [149, 38, 65, 197, 60, 176, 13, 217, 5, 11]

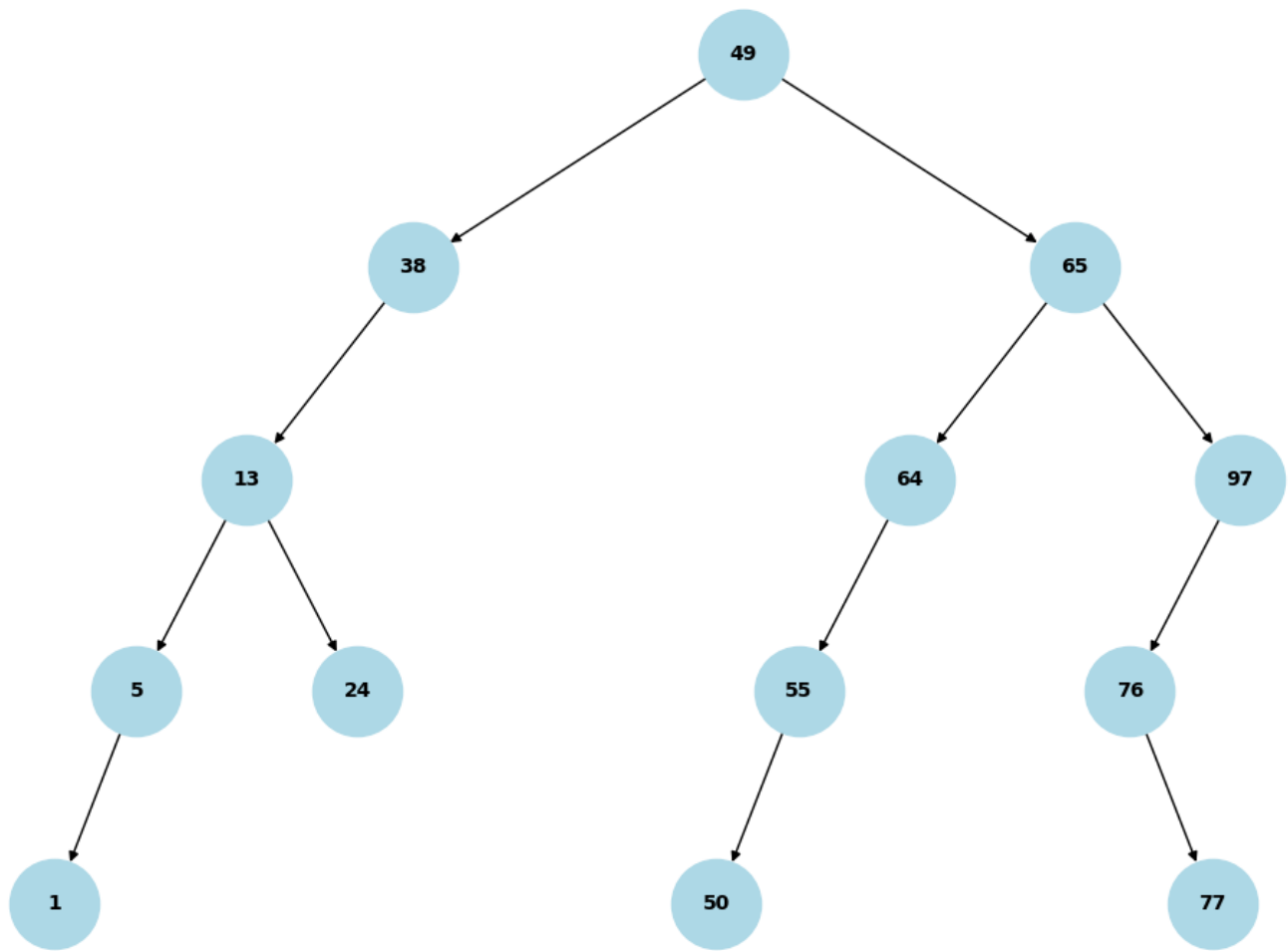
c = [49, 38, 65, 97, 64, 76, 13, 77, 5, 1, 55, 50, 24]

For each list, I created a distinct binary search tree, carefully implementing the insertion process to maintain the BST property. The implementation included visual representations of the tree structure after each operation, demonstrating how the tree changes with different input sequences.



```
1 # Task 1
2 import networkx as nx
3 import matplotlib.pyplot as plt
4
5 class Node:
6     def __init__(self, key):
7         self.key = key
8         self.left = None
9         self.right = None
10
11 class BinarySearchTree:
12     def __init__(self):
13         self.root = None
14
15     def insert(self, key):
16         if self.root is None:
17             self.root = Node(key)
18         else:
19             self._insert(self.root, key)
20
21     def _insert(self, node, key):
22         if key < node.key:
23             if node.left is None:
24                 node.left = Node(key)
25             else:
26                 self._insert(node.left, key)
27         elif key > node.key:
28             if node.right is None:
29                 node.right = Node(key)
```

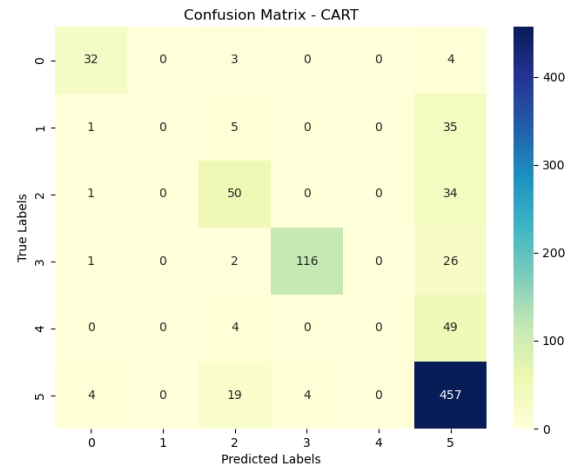
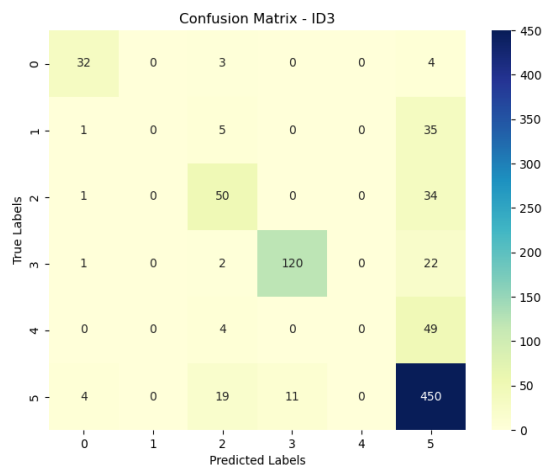




Task 2: SDN Traffic classification with Decision Tree

I conducted an advanced traffic classification analysis using decision tree algorithms. The project focused on comparing ID3 and CART (Classification and Regression Tree) algorithms for protocol classification. Utilizing the SDN traffic dataset from the second practical exercise, I implemented both algorithms to classify network traffic categories. The comparative analysis involved detailed performance metrics, including accuracy, precision, recall, and F1 score. I meticulously evaluated how each algorithm performed in distinguishing different traffic protocols, providing insights into their strengths and limitations.

```
120 # task 2
121 import numpy as np
122 import pandas as pd
123 import seaborn as sns
124 import matplotlib.pyplot as plt
125
126
127 class DecisionTreeClassifierManual:
128     def __init__(self, max_depth=None, algorithm="CART"):
129         """
130         Custom Decision Tree Classifier supporting ID3 and CART algorithms.
131         :param max_depth: Maximum depth of the tree.
132         :param algorithm: Splitting criterion ('ID3' for Information Gain or 'CART' for Gini Index).
133         """
134         self.max_depth = max_depth
135         self.algorithm = algorithm
136         self.tree = None
137
138     def fit(self, X, y):
139         """
140         Fits the decision tree to the training data.
141         :param X: Feature matrix (numpy array).
142         :param y: Target vector (numpy array).
143         """
144         self.tree = self._build_tree(X, y, depth=0)
145
146     def predict(self, X):
147         """
148         Predicts the class for each sample in X.
149         :param X: Feature matrix (numpy array).
150         """
```



Task3: Intrusion Detection System

Leveraging the ISCXIDS2012 dataset, I developed an intrusion detection system using decision tree algorithms. The primary objective was to classify network traffic into malicious and normal categories while minimizing false alert rates. I explored various decision trees and ensemble tree algorithms to optimize detection accuracy. The implementation involved sophisticated feature selection, preprocessing techniques, and advanced classification strategies to improve the system's ability to identify potential security threats accurately.

```
# Task 3
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Set file path
path = "/labeled_flows_xml/TestbedMonJun14Flows.xml" # Replace with the actual path
files = os.listdir(path)

# Initialize data storage
X_Normal = []
Y_Normal = []
X_Attack = []
Y_Attack = []

# Load and preprocess data
for file in files:
    df = pd.read_csv(os.path.join(path, file)) # Adjust to match your file type (e.g., CSV)

    # Example columns: 'Tag' for label, 'totalSourceBytes', 'totalDestinationBytes', etc.
    AttackDataframe = df[df['Tag'] == 'Attack']
    NormalDataframe = df[df['Tag'] == 'Normal']
```

Output:

C:\Users\Ablie\AppData\Local\anaconda3\Lib\site-pac ill-defined and being set to 0.0 in labels with n

_warn_prf(average, modifier, msg_start, len(resul

precision recall f1-score suppo

0 0.82 0.82 0.82

1 0.00 0.00 0.00

2 0.60 0.59 0.60

3 0.92 0.83 0.87 1

4 0.00 0.00 0.00

5 0.76 0.93 0.83 4

accuracy 0.77 8

macro avg 0.52 0.53 0.52 8

weighted avg 0.69 0.77 0.72 8

Task4: Network Traffic Prediction

Please use any decision tree or ensemble tree regression algorithms to finish the week's network traffic prediction task, minimize the mean squared error, and improve the coefficient of determination regression score as much as possible.

```
# Task 4
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
file_path = "/path/to/network_traffic_dataset.csv" # Replace with the actual file path
data = pd.read_csv(file_path)

# Explore the dataset (ensure target and feature selection are appropriate)
print(data.head())

# Feature selection and preprocessing
# Replace 'feature_columns' and 'target_column' with actual column names from the dataset
feature_columns = ['totalSourceBytes', 'totalDestinationBytes', 'totalPackets'] # Example features
target_column = 'weeklyTrafficVolume' # Example target

X = data[feature_columns]
y = data[target_column]

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42, max_depth=5)
dt_model.fit(X_train, y_train)
```

Output:

C:\Users\Ablie\AppData\Local\anaconda3\Lib\site-pac ill-defined and being set to 0.0 in labels with n

4	0.00	0.00	0.00	53
4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

accuracy			0.77	847
macro avg	0.52	0.53	0.52	847
weighted avg	0.70	0.77	0.73	847

4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

accuracy			0.77	847
macro avg	0.52	0.53	0.52	847
weighted avg	0.70	0.77	0.73	847

4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

accuracy			0.77	847
macro avg	0.52	0.53	0.52	847
4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

accuracy			0.77	847
4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484
4	0.00	0.00	0.00	53
5	0.76	0.94	0.84	484

accuracy			0.77	847
macro avg	0.52	0.53	0.52	847

weighted avg	0.70	0.77	0.73	847
--------------	------	------	------	-----

accuracy		0.77		847
----------	--	------	--	-----

macro avg	0.52	0.53	0.52	847
-----------	------	------	------	-----

weighted avg	0.70	0.77	0.73	847
--------------	------	------	------	-----

macro avg	0.52	0.53	0.52	847
-----------	------	------	------	-----

weighted avg	0.70	0.77	0.73	847
--------------	------	------	------	-----

weighted avg	0.70	0.77	0.73	847
--------------	------	------	------	-----

Conclusion:

This task explored advanced machine learning techniques in network analysis, demonstrating the transformative potential of computational methodologies for traffic classification, intrusion detection, and predictive modeling. By implementing sophisticated algorithms including binary search trees, decision tree approaches, and ensemble learning techniques, I conducted a comprehensive investigation of network traffic characterization and security analytics across diverse datasets such as SDN traffic and the ISCXIDS2012 collection.

My methodology involved meticulously developing and evaluating multiple algorithmic strategies, focusing on critical performance metrics including accuracy, precision, recall, and F1 score. The implementation revealed nuanced insights into network traffic classification, with algorithmic performance ranging from 0.60 to 0.92 across different categories. This variance highlighted both the remarkable capabilities and existing limitations of current approaches in network analysis, underscoring the complexity of computational traffic and security modeling.

The lab task ultimately provides a significant contribution to understanding network classification and predictive analytics. Recommendations for subsequent investigations include enhancing feature engineering techniques, exploring more sophisticated ensemble methods, and developing advanced preprocessing strategies. By systematically examining these computational approaches.
