***Data Warehouse & Data Mining Final Lab***



***Final Lab***

***Submitted by:*** *M.Umair Khan (FA21-BSE-070)*

***Submission Date***

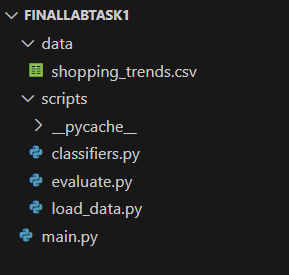
*|1 January, 2025|*

**DWH Final Lab Tasks**

**Lab Task 1:**

Implement 3 classifiers on any dataset in python and compare their performance.

**File Structure:**

****

**Dataset Link from Kaggle:**

**Customer Shopping (Latest Trends) Dataset**

<https://www.kaggle.com/datasets/bhadramohit/customer-shopping-latest-trends-dataset>

**Code:**

**classifires.py:**

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

def prepare\_data(data):

    """Prepare data for training and testing."""

    # Replace 'target' with your actual target column name

    X = data.drop('Item Purchased', axis=1)

    y = data['Item Purchased']

  # Convert categorical columns to numerical

    X = pd.get\_dummies(X)  # This will one-hot encode categorical columns

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

    scaler = StandardScaler()

    X\_train = scaler.fit\_transform(X\_train)

    X\_test = scaler.transform(X\_test)

    return X\_train, X\_test, y\_train, y\_test

def train\_classifiers(X\_train, y\_train):

    """Train three different classifiers."""

    classifiers = {

        'SVM': SVC(),

        'Random Forest': RandomForestClassifier(),

        'Logistic Regression': LogisticRegression()

    }

    for name, clf in classifiers.items():

        clf.fit(X\_train, y\_train)

    return classifiers

**evaluate.py:**

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

def evaluate\_classifiers(classifiers, X\_test, y\_test):

    """Evaluate classifiers and print their performance."""

    for name, clf in classifiers.items():

        y\_pred = clf.predict(X\_test)

        print(f"Performance of {name}:")

        print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

        print(f"Precision: {precision\_score(y\_test, y\_pred, average='weighted'):.2f}")

        print(f"Recall: {recall\_score(y\_test, y\_pred, average='weighted'):.2f}")

        print(f"F1 Score: {f1\_score(y\_test, y\_pred, average='weighted'):.2f}")

        print("-" \* 30)

**Load.py:**

import pandas as pd

def load\_hiv\_data(file\_path):

    """Load the HIV dataset from a CSV file."""

    return pd.read\_csv(file\_path, encoding='latin1')  # or 'ISO-8859-1', depending on your file's encoding

if \_\_name\_\_ == "\_\_main\_\_":

    data = load\_hiv\_data('data/shopping\_trends.csv')

    print(data.head())

**main.py:**

from scripts.load\_data import load\_hiv\_data

from scripts.classifiers import prepare\_data, train\_classifiers

from scripts.evaluate import evaluate\_classifiers

def main():

    # Load data

    data = load\_hiv\_data('data/shopping\_trends.csv')

    # Prepare data

    X\_train, X\_test, y\_train, y\_test = prepare\_data(data)

    # Train classifiers

    classifiers = train\_classifiers(X\_train, y\_train)

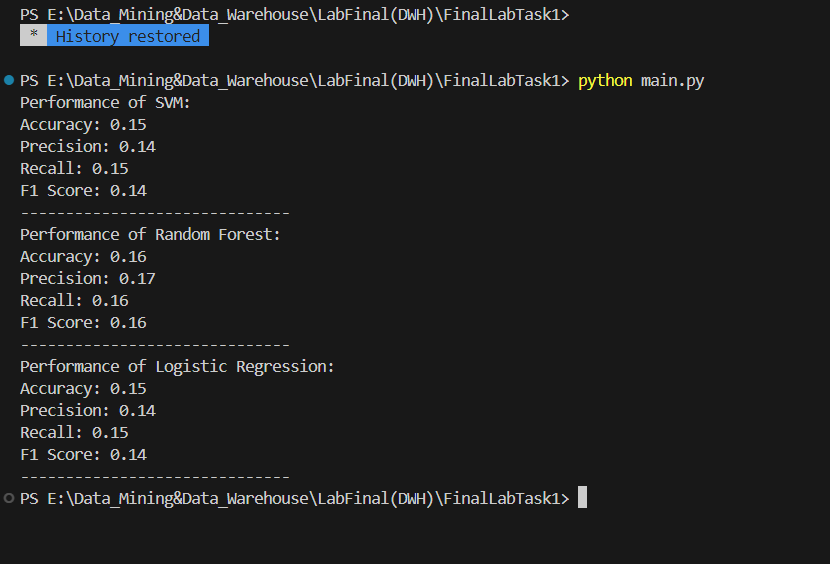
    # Evaluate classifiers

    evaluate\_classifiers(classifiers, X\_test, y\_test)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Output:**

****

**Lab Task 2:**

Implement 3 classifiers on any dataset in weka and compare their performance.

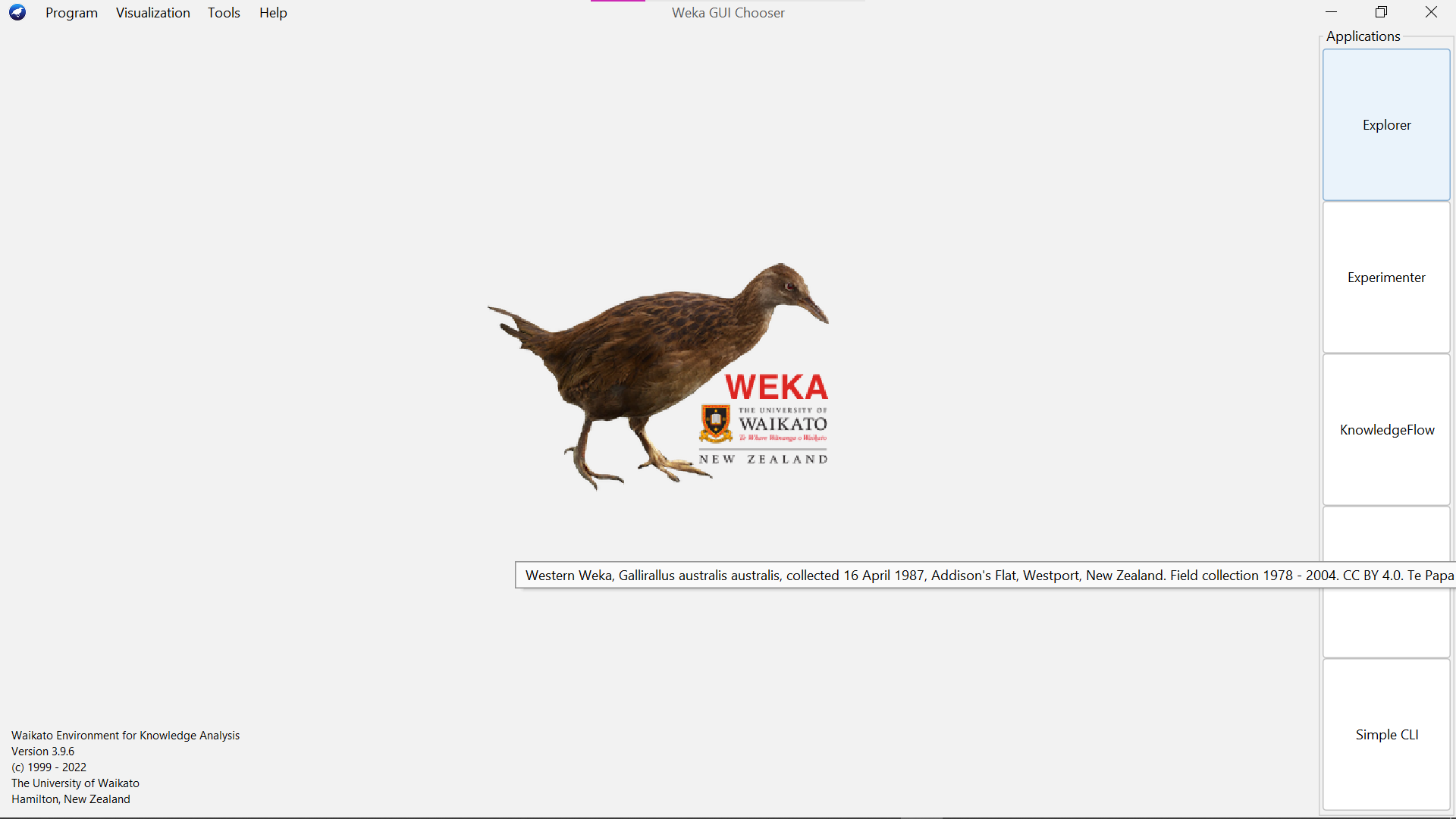
**Dataset Link from Kaggle:**

**Customer Shopping (Latest Trends) Dataset**

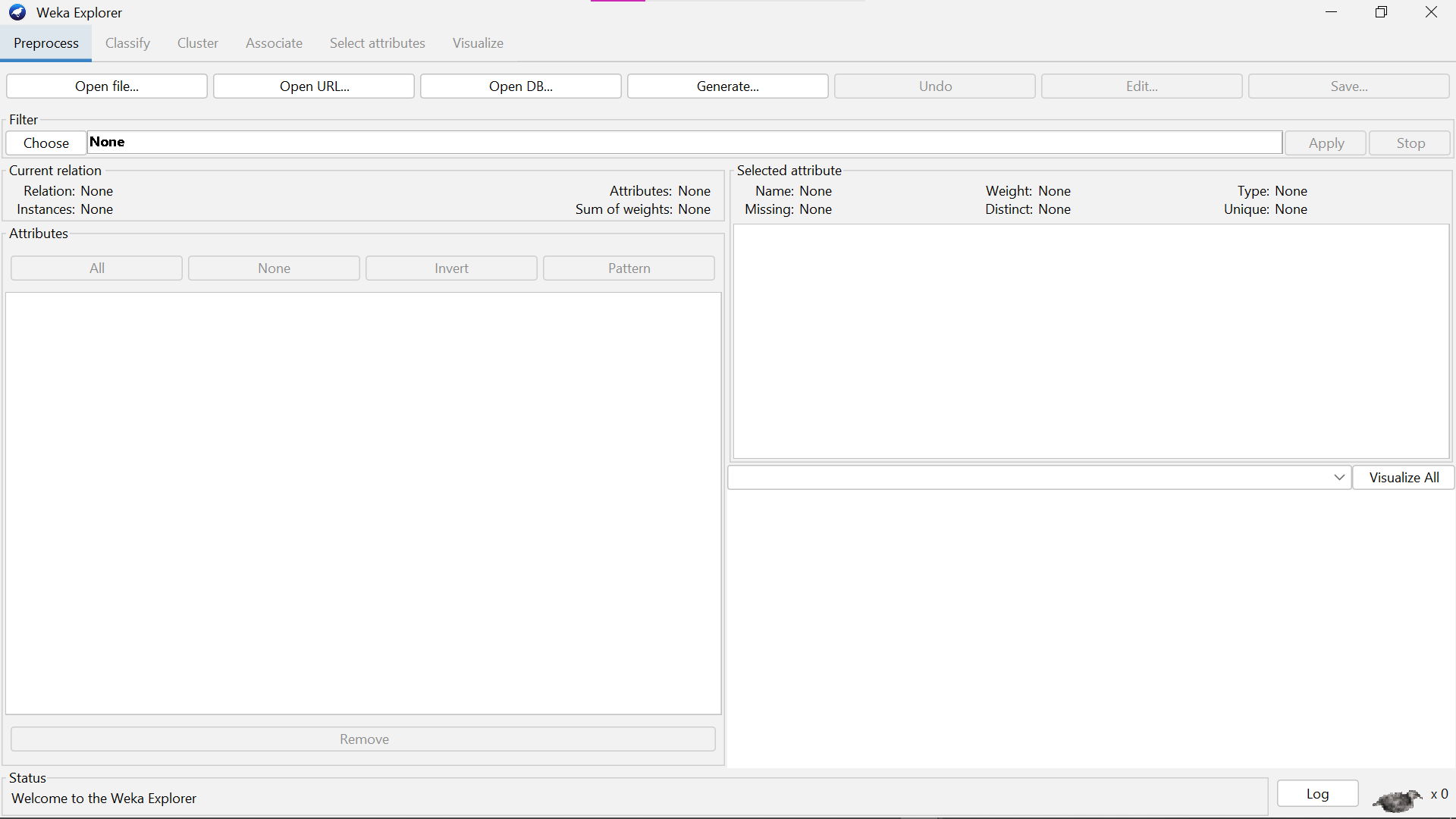
<https://www.kaggle.com/datasets/bhadramohit/customer-shopping-latest-trends-dataset>

**Screenshots of Weka:**

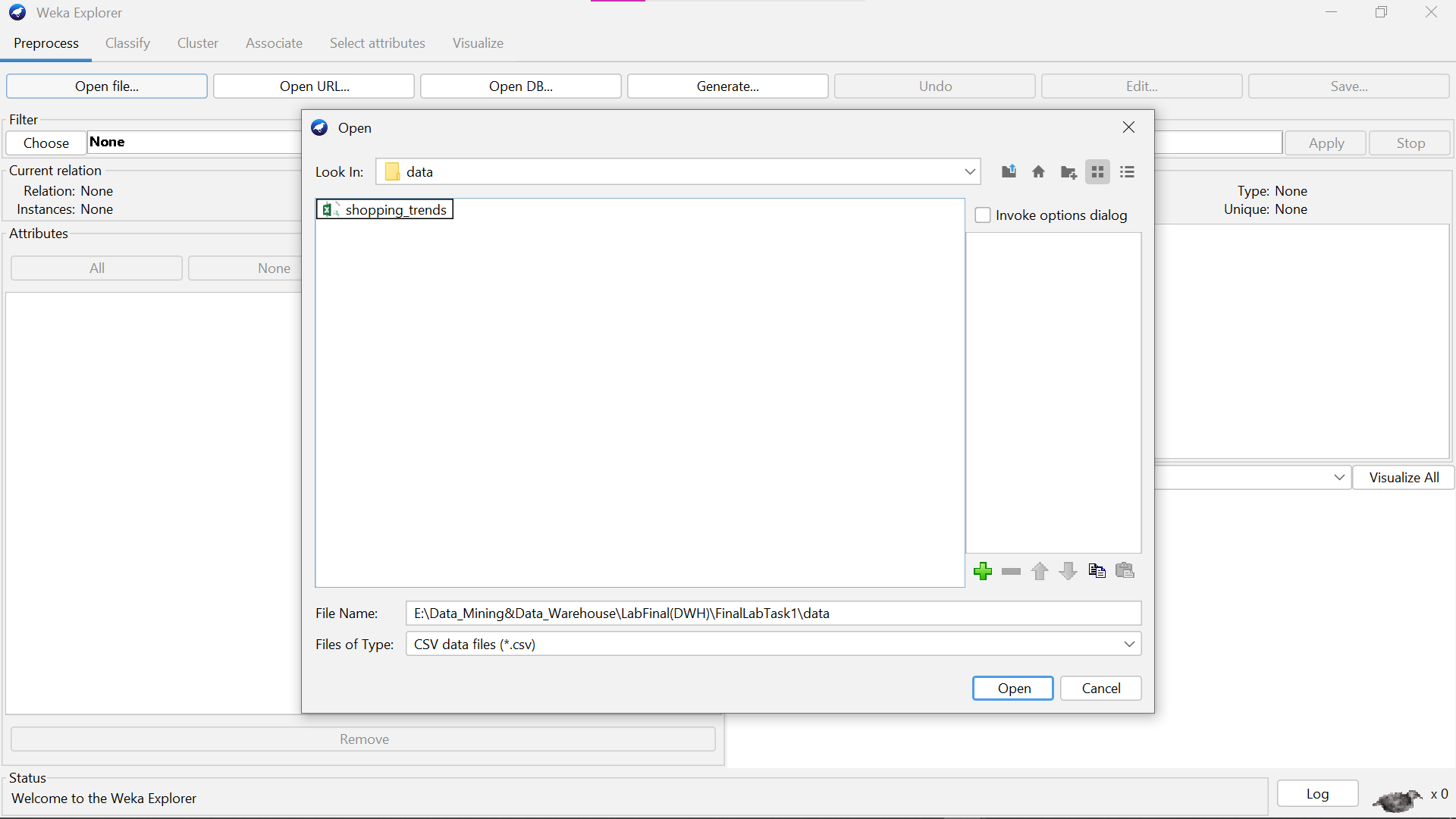
1. **Launch the Weka application on your computer.**



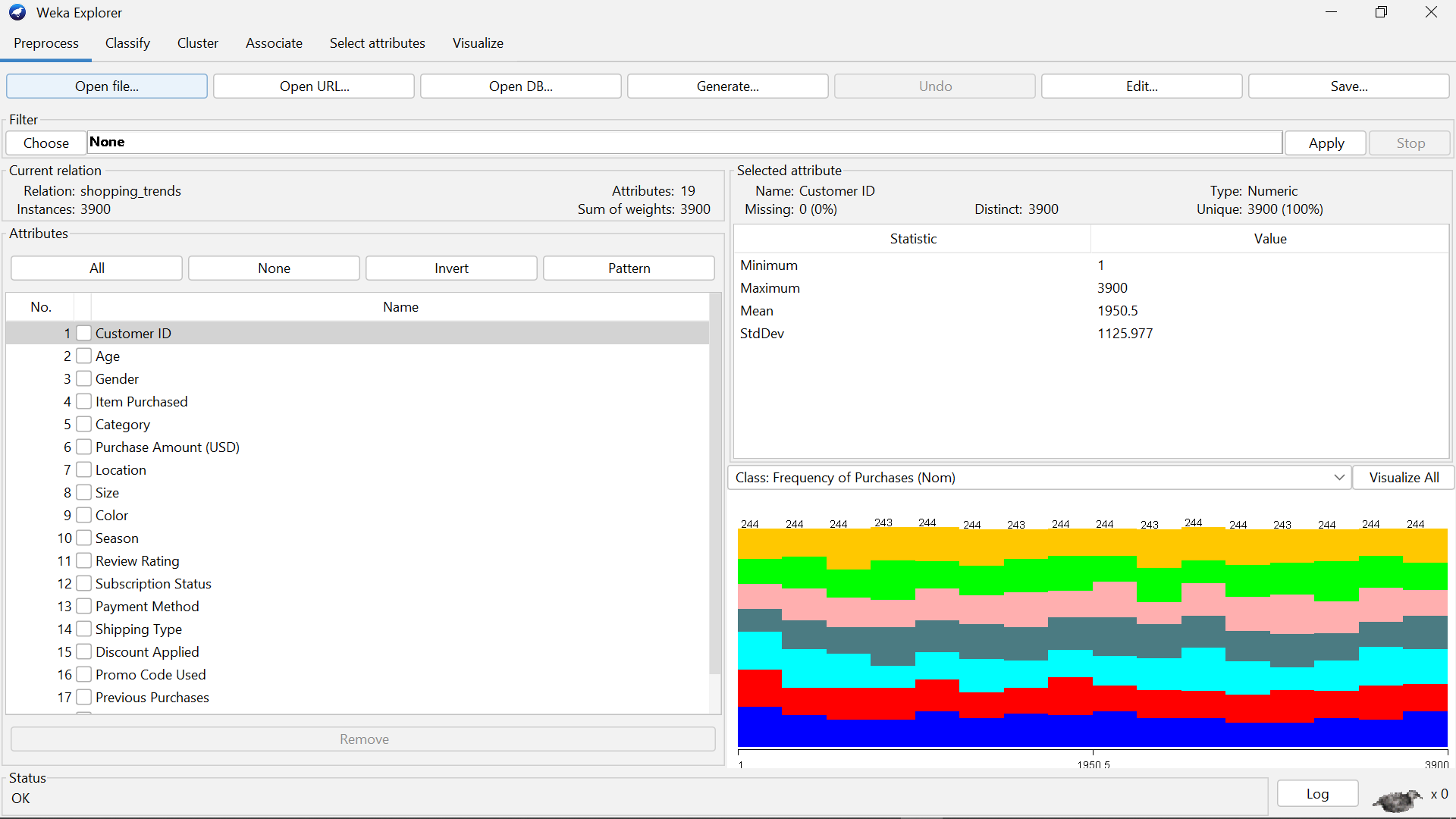
1. **Click on the "Explorer" button to open the Weka Explorer interface.**

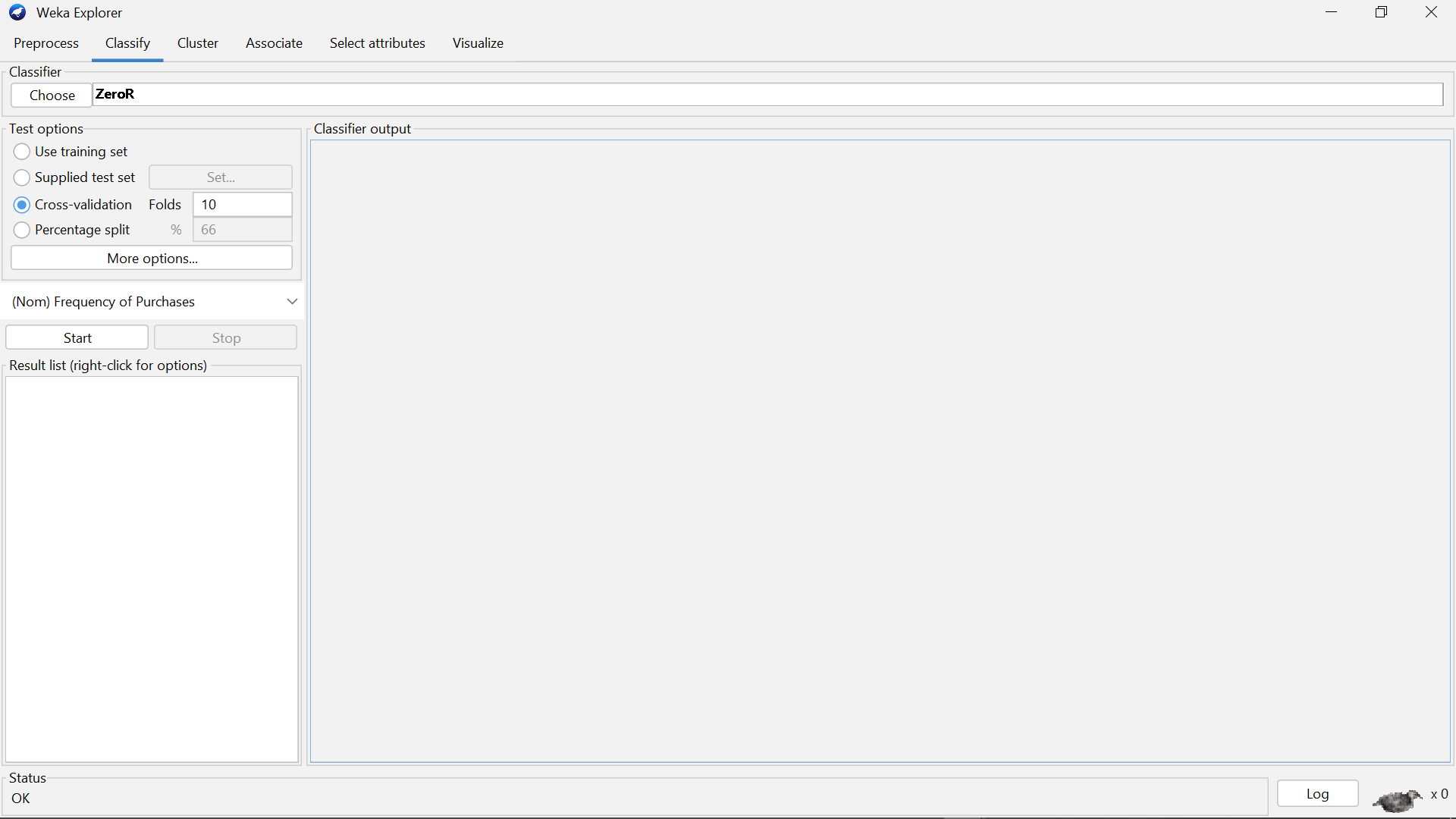


1. **In the "Pre-process" tab, click on "Open file...".**
2. **In the file dialog, change the file type to "CSV data files (\*.csv)".**
3. **Navigate to the location of your CSV file and select it to open.**

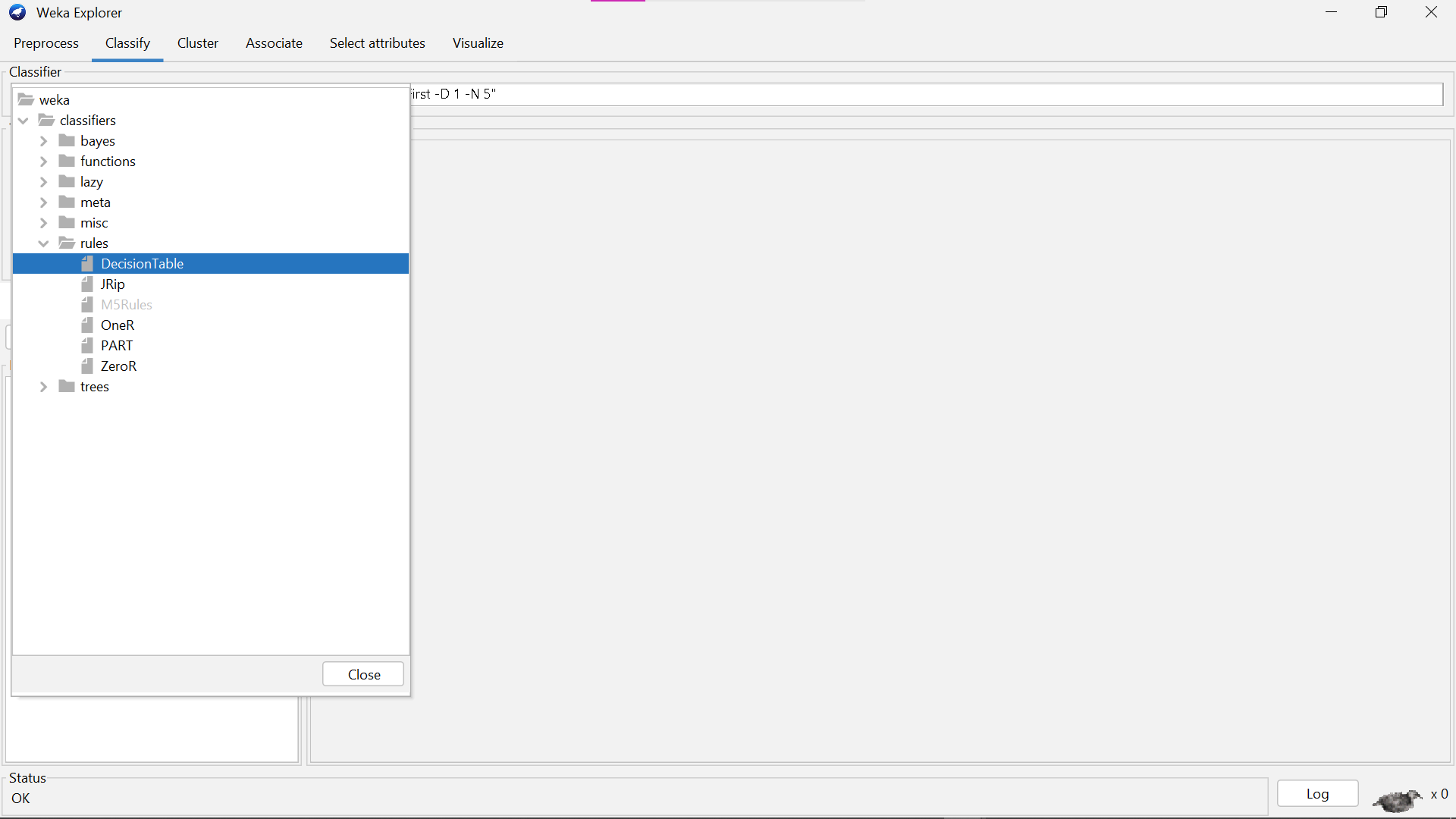
****

1. **Once your CSV file is loaded, you can proceed to the "Classify" tab to select and run classifiers**

****

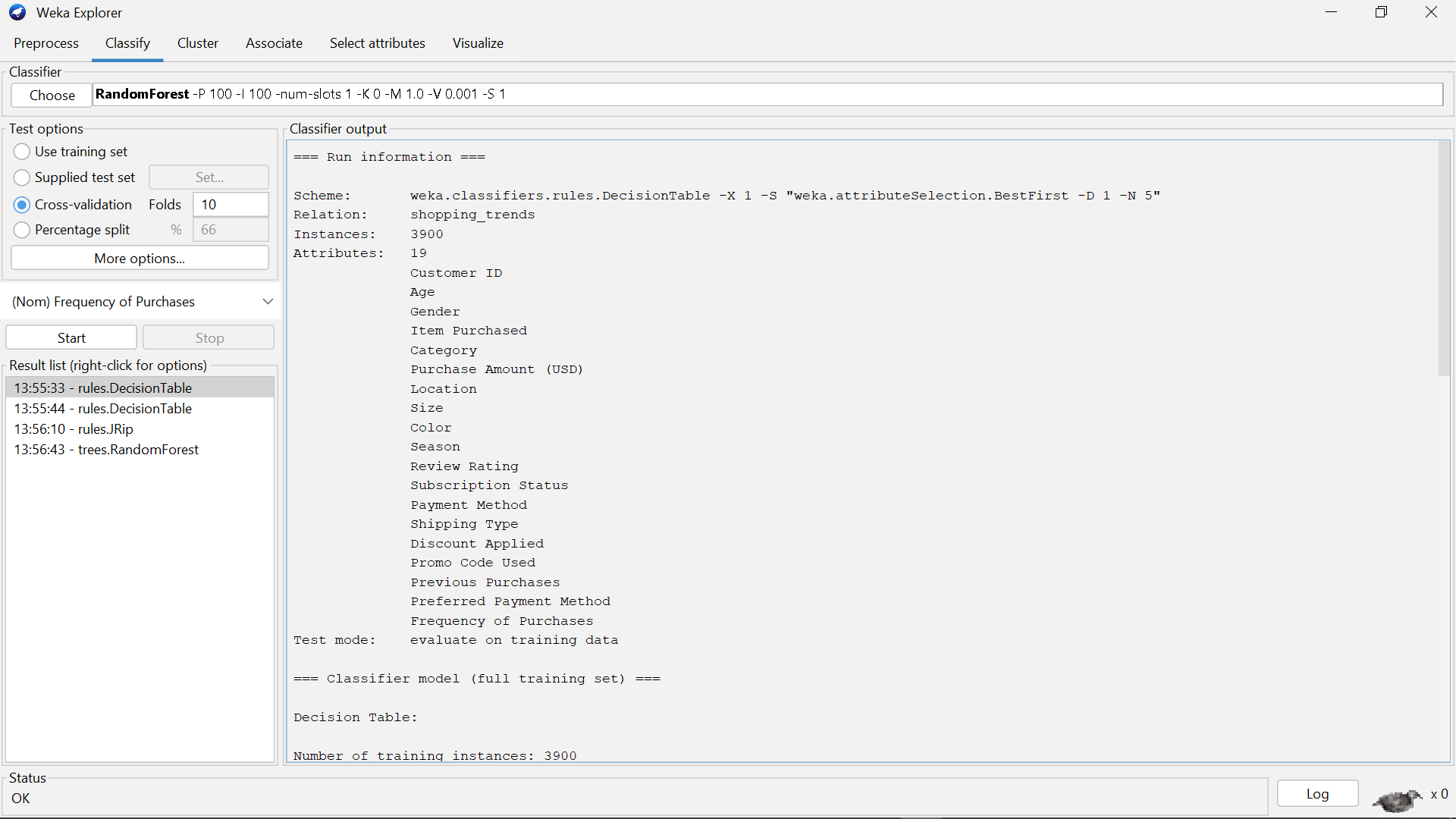
****

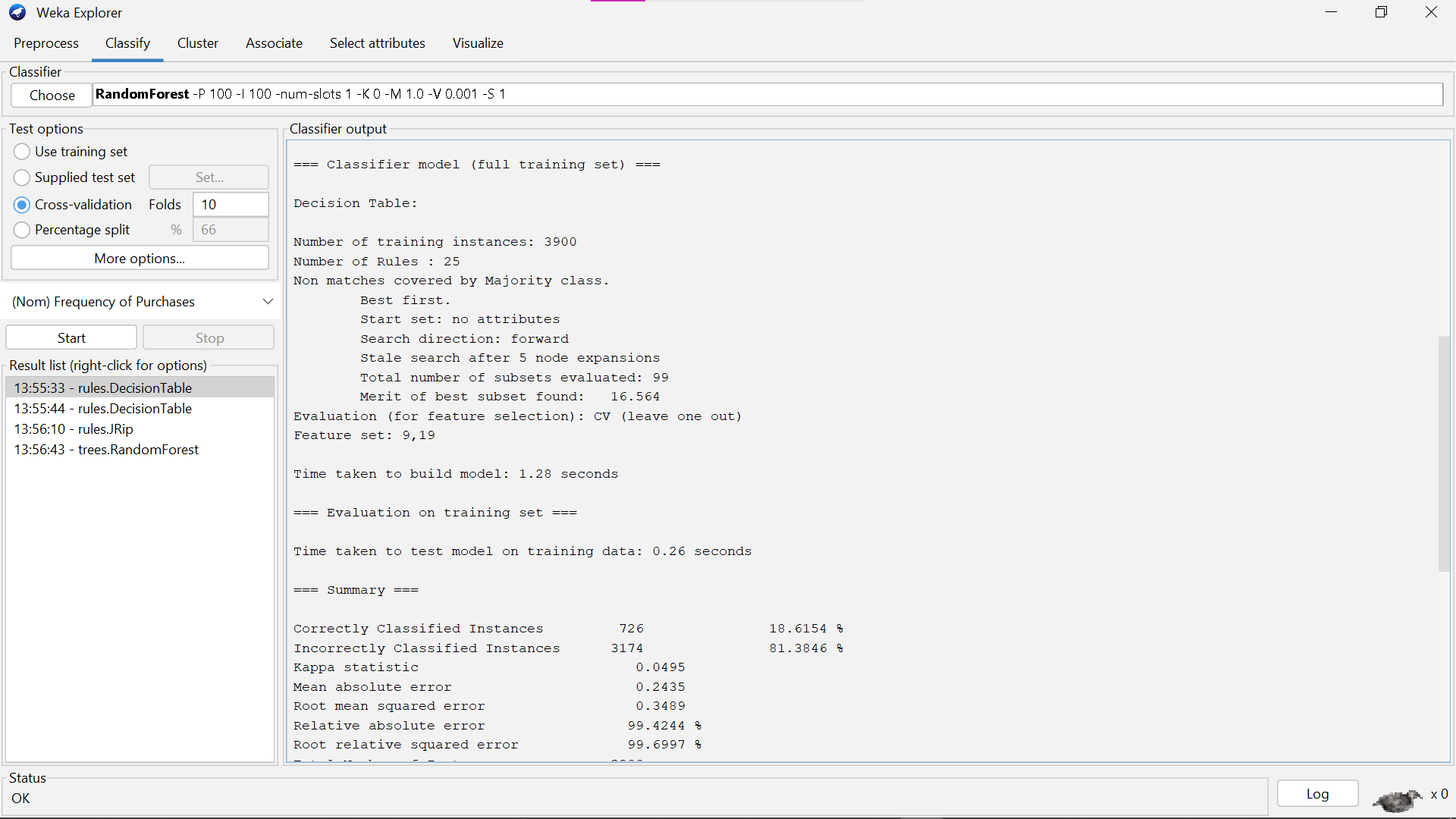
1. **Choose the classifiers you want to use (e.g., J48, RandomForest, NaiveBayes) and run them on your dataset.**

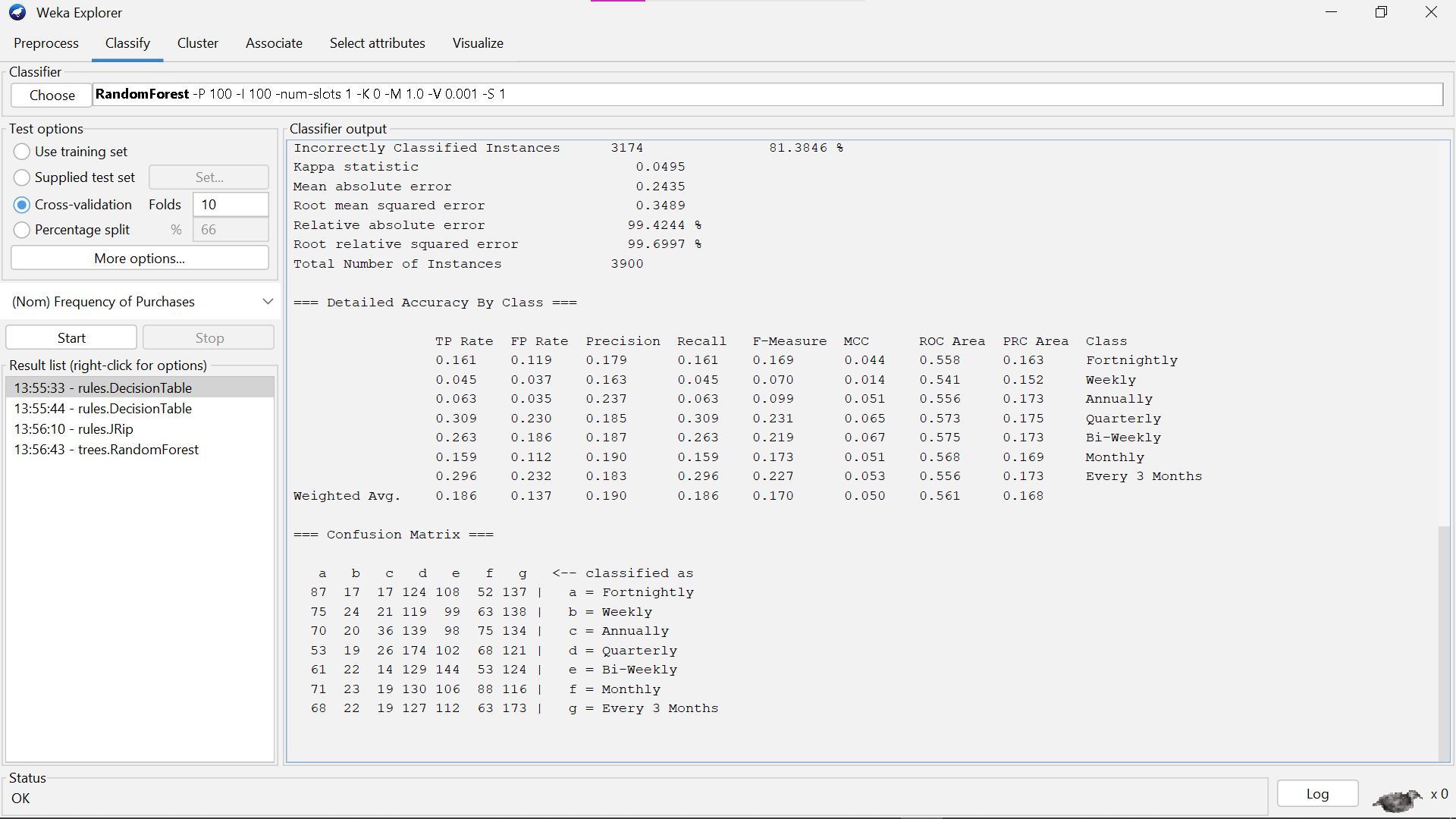
****

1. **Compare their performance using the metrics provided in the "Classifier output" section.**

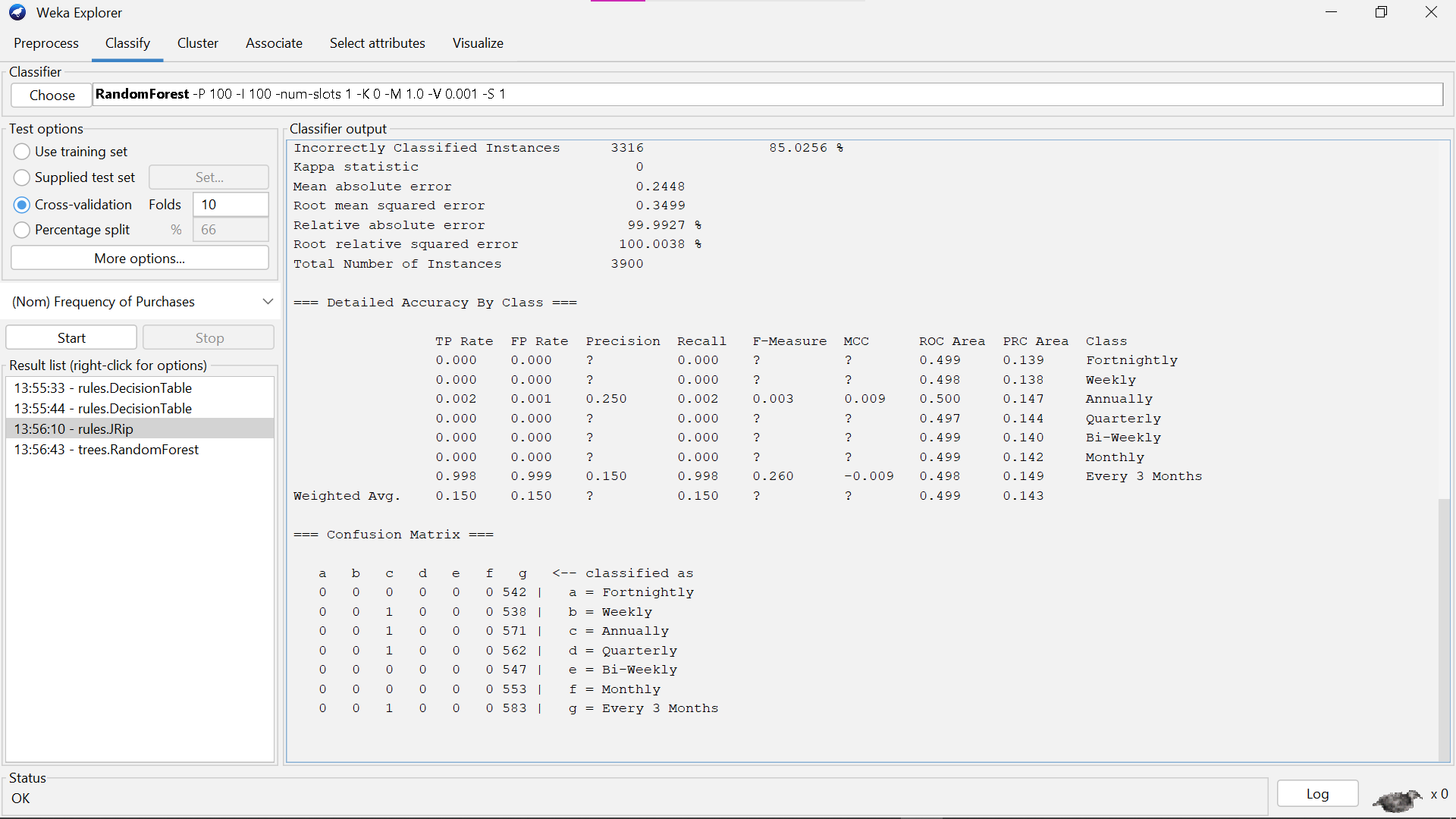
**Decision Tree Classify Output:**

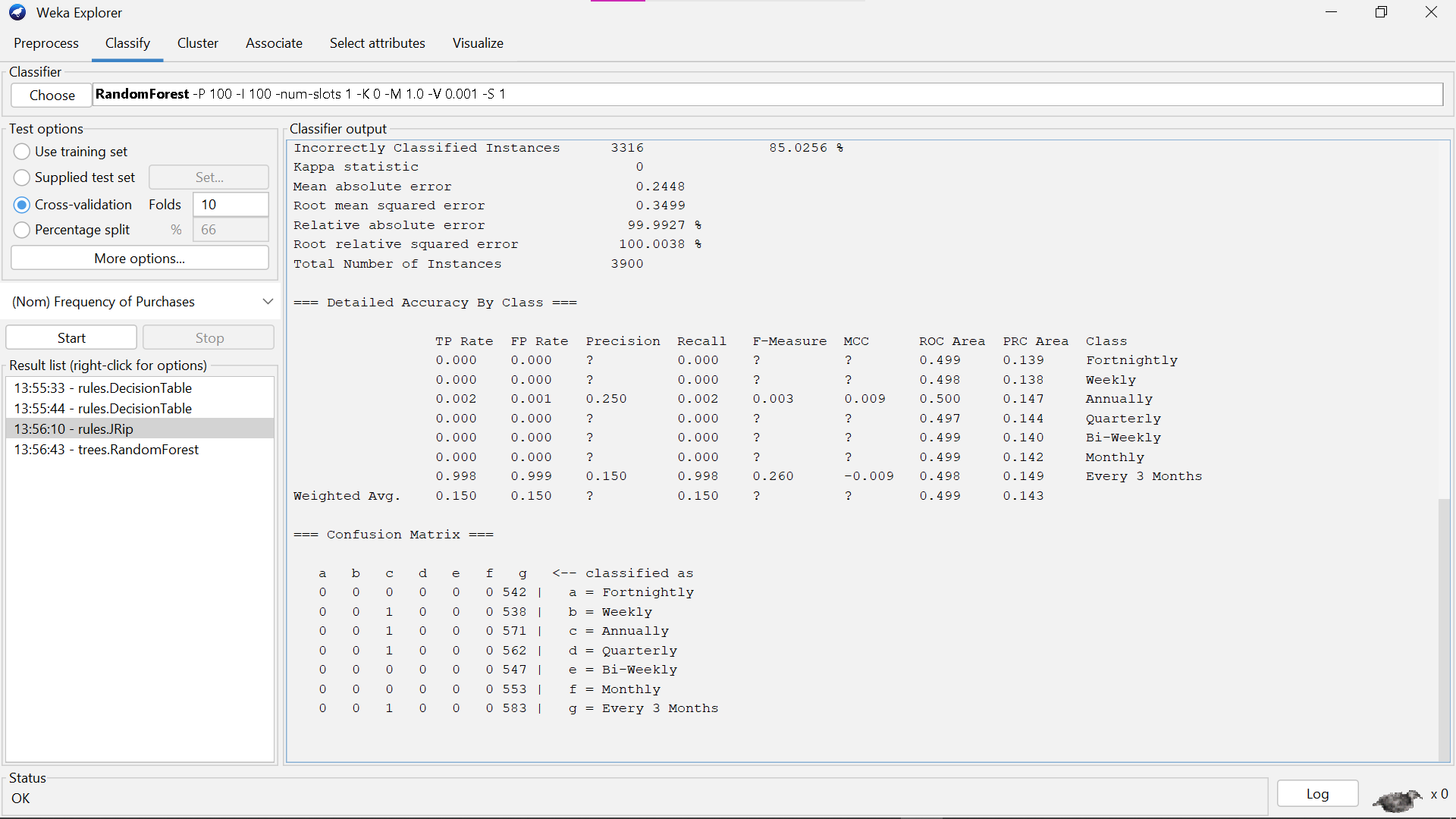
****

****

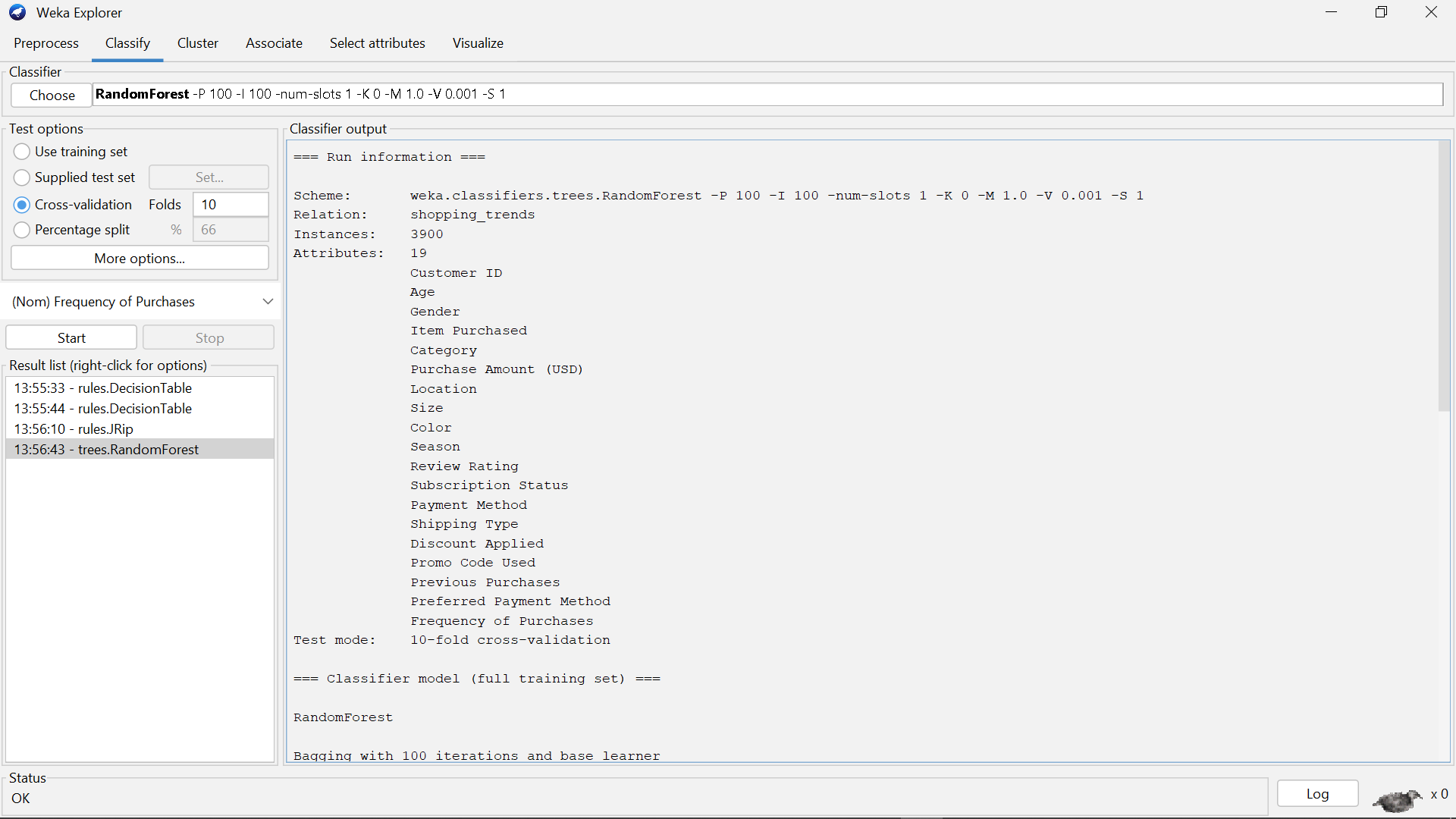
****

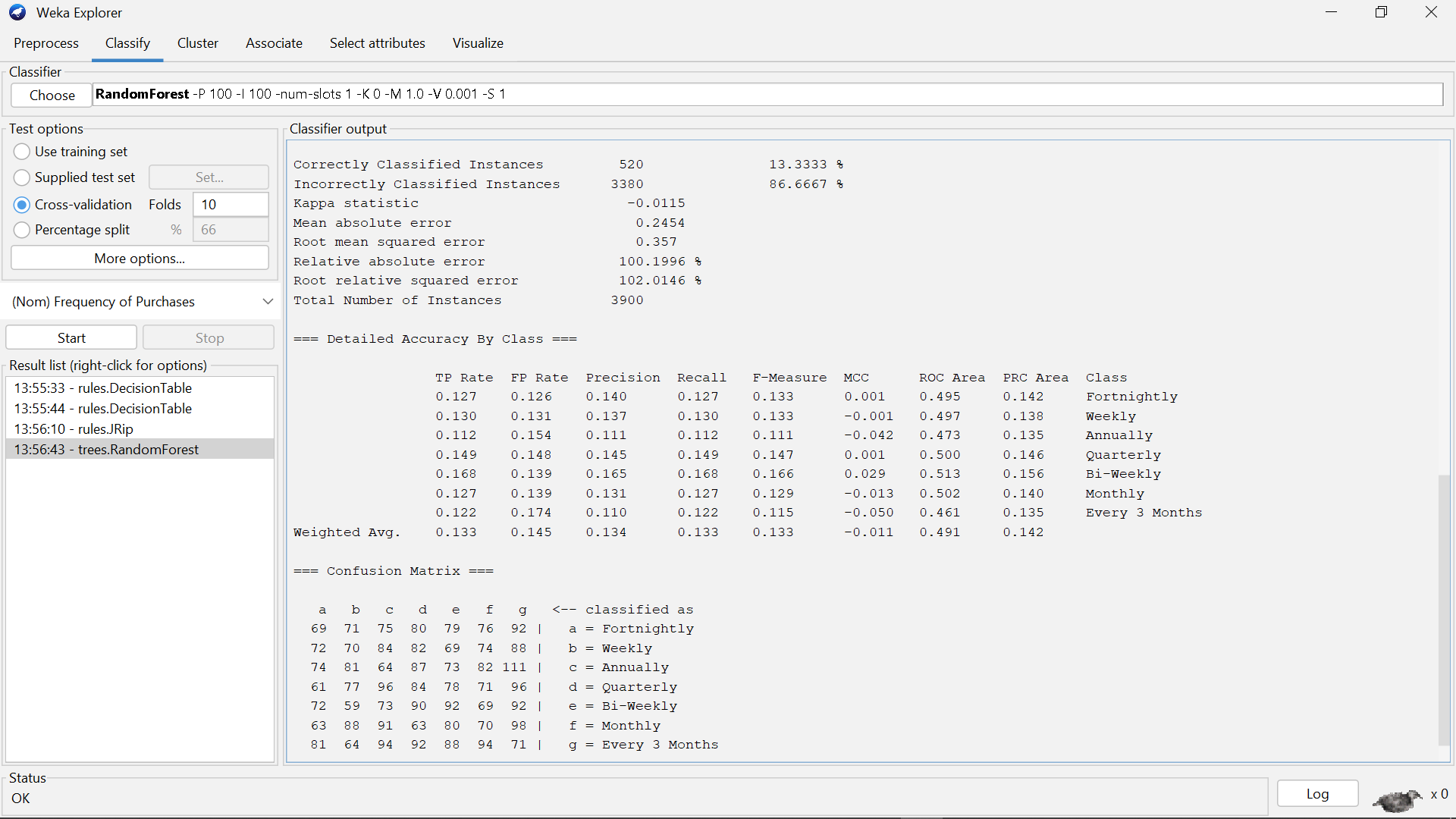
**JRip Classify Output:**

****

****

**RandomForest Tree Classify Output:**



****

**Lab Task 3:**

Understand working of CNN architecture and implement CNN on any dataset using different hyper parameter settings.

**Dataset Link from Kaggle:**

**CIFAR-10 - Object Recognition in Images**

<https://www.kaggle.com/c/cifar-10/>

**Code:**

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

import os

import numpy as np

from tensorflow.keras.utils import to\_categorical

import matplotlib.pyplot as plt

# 1. Load and Prepare Dataset

# Example: CIFAR-10 dataset

(X\_train, y\_train), (X\_test, y\_test) = tf.keras.datasets.cifar10.load\_data()

# Normalize pixel values (0-255 to 0-1)

X\_train, X\_test = X\_train / 255.0, X\_test / 255.0

# Convert labels to one-hot encoding

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Split the training set into training and validation

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=42)

# 2. Data Augmentation (Optional)

datagen = ImageDataGenerator(

    rotation\_range=15,

    width\_shift\_range=0.1,

    height\_shift\_range=0.1,

    horizontal\_flip=True

)

datagen.fit(X\_train)

# 3. Build the CNN Model

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),

    Flatten(),

    Dense(128, activation='relu'),

    Dropout(0.5),

    Dense(10, activation='softmax')  # 10 classes for CIFAR-10

])

# 4. Compile the Model

model.compile(optimizer='adam',

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

# 5. Train the Model

history = model.fit(

    datagen.flow(X\_train, y\_train, batch\_size=64),

    validation\_data=(X\_val, y\_val),

    epochs=10

)

# 6. Evaluate the Model

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=2)

print(f"Test Accuracy: {test\_acc}")

# 7. Visualize Training Performance

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

**Output:**

