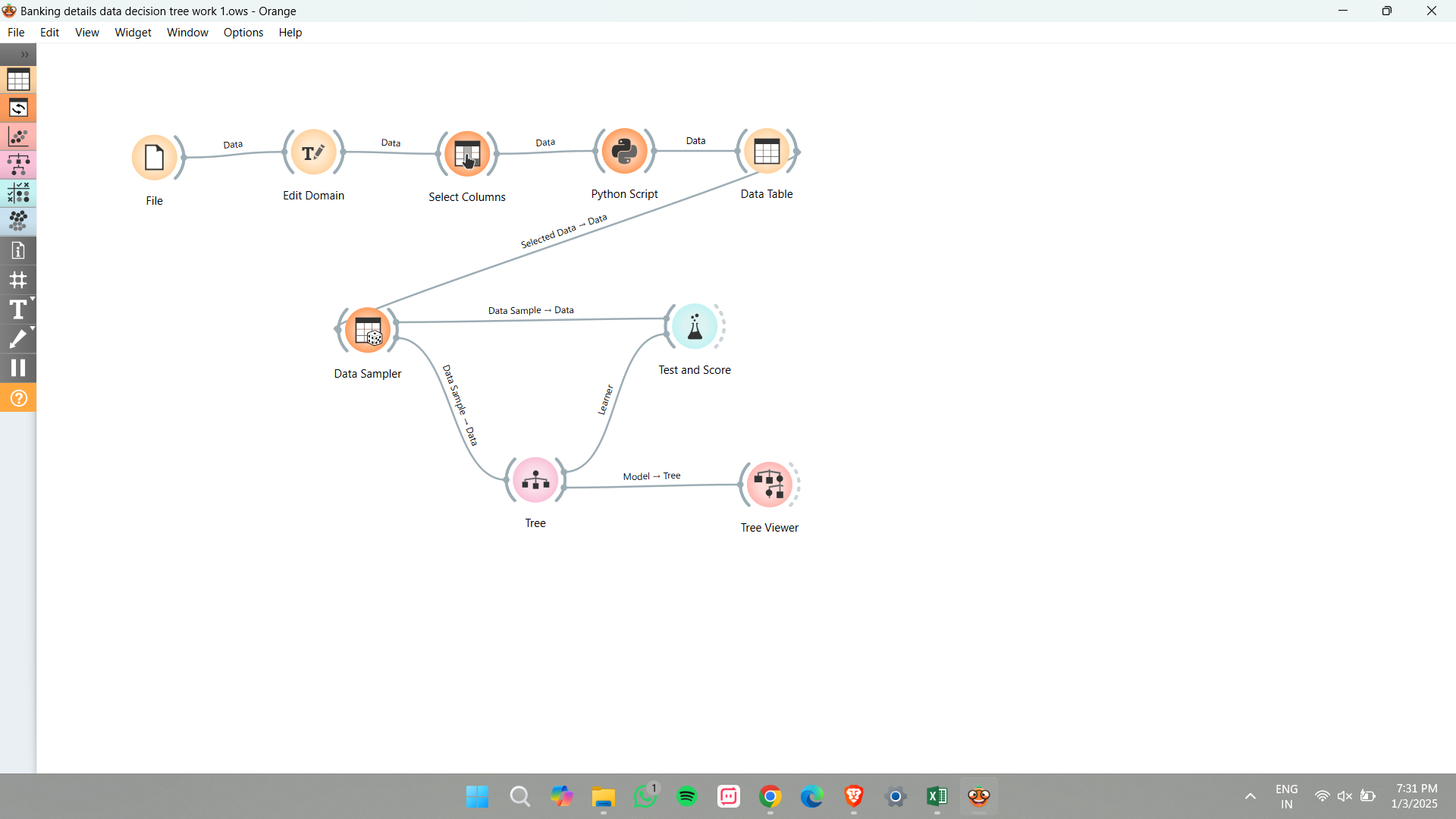
# **Overview of the task:**



**Orange** is a Data Mining Software that is a handy tool for forming a Decision tree from large datasets.

* First, the Data is taken from the CSV file.
* Then the Data file is connected to a widget **“Edit Domain”** where the given columns are formatted according to the data. (eg: age – numerical value, months – text, etc)
* The “Select Column” widget determines the target variable.
* Since the given dataset is imbalanced, the **SMOTE** (Synthetic Minority Over-Sampling Technique) script is used in Python which helps to balance the output (yes/no) for better fitting in the decision tree.

**PYTHON SCRIPT:**  
from imblearn.over\_sampling import SMOTE

import numpy as np

from Orange.data import Table, Domain

# Ensure in\_data is connected

if in\_data is None:

raise ValueError("Input data (in\_data) is not connected properly.")

# Extract features and labels

X = in\_data.X

y = in\_data.Y.flatten()

# Check the shape of y to ensure it's not empty

if len(y) == 0:

raise ValueError("No labels found. Please check the class variable.")

# Apply SMOTE if data is valid

smote = SMOTE(sampling\_strategy='auto', random\_state=42)

X\_res, y\_res = smote.fit\_resample(X, y)

# Convert back to Orange Table

domain = Domain(in\_data.domain.attributes, in\_data.domain.class\_var)

out\_data = Table(domain, X\_res, y\_res)

# Set the output data

out\_data # This makes the data available to the widget output

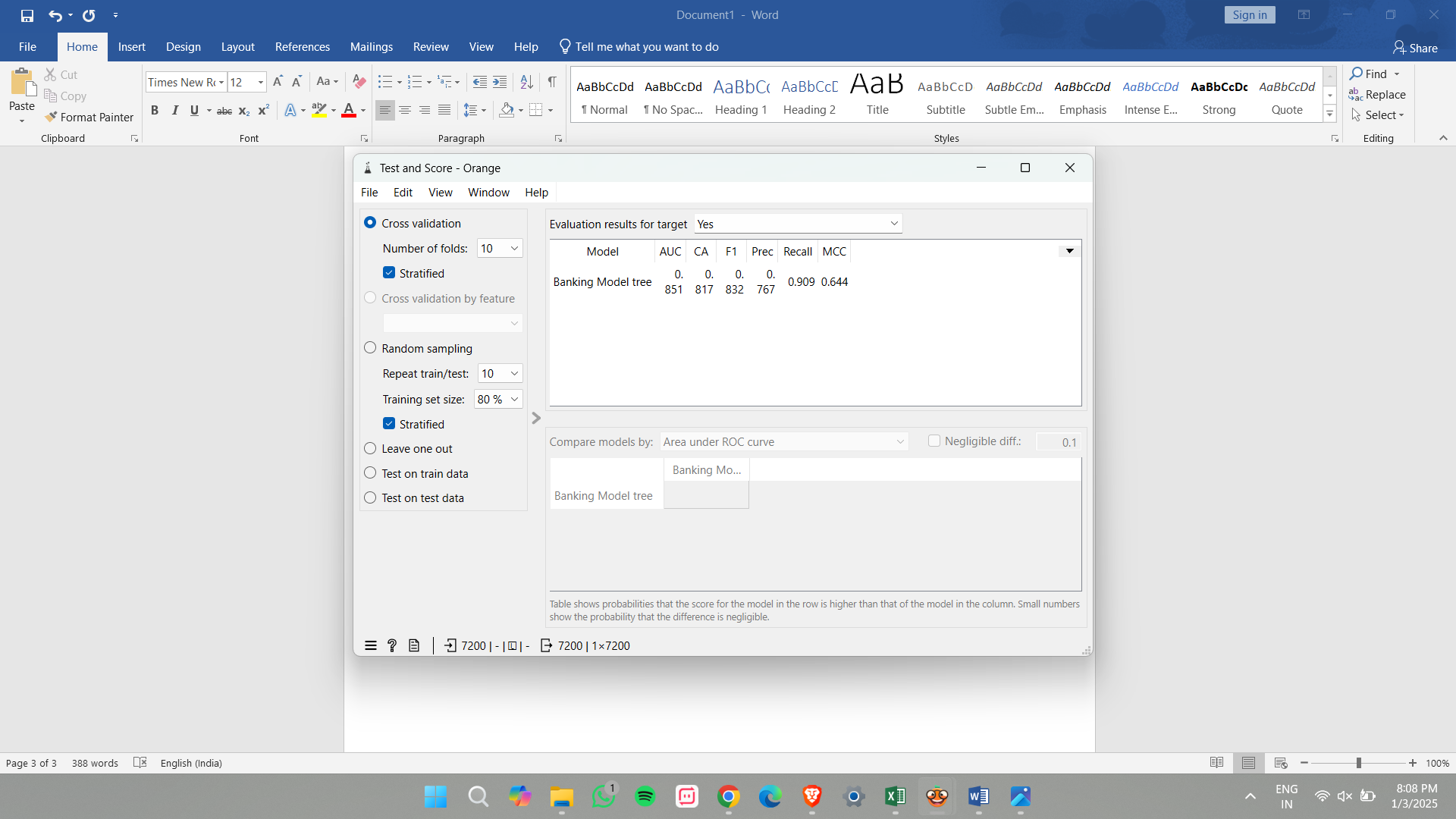
print("Starting script...")

print(f"Shape of X: {X.shape}")

print(f"Shape of y: {y.shape}")

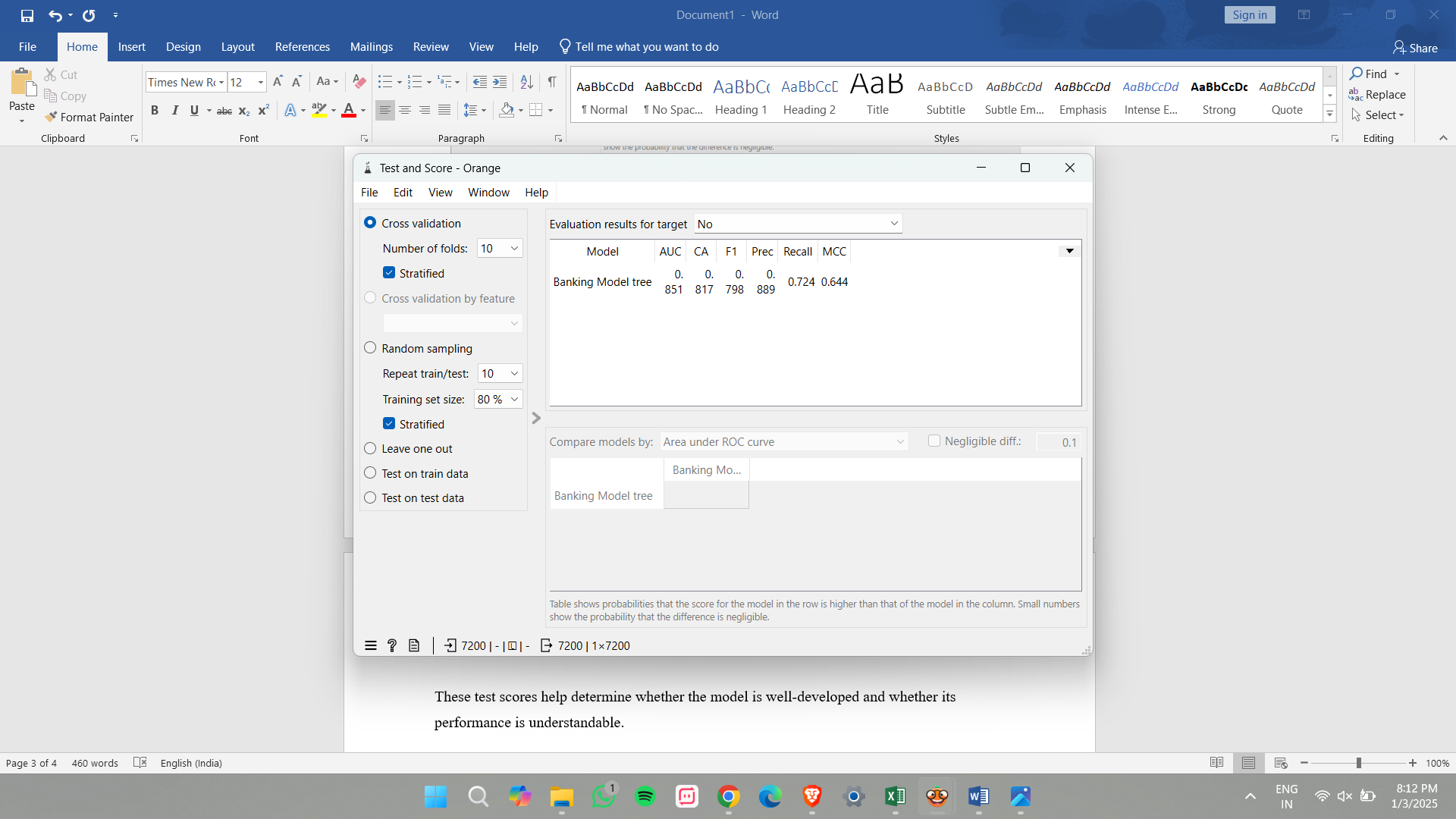
* The **“Data Table”** widget gives the complete output of the Python script.
* The **“Data Sampler”** widget is used to determine the type of sampling to be done (fixed sample or crossing validation sample)
* The Data sampler is connected to a **“Tree”** where the depth of the tree, maximum number of instances, and number of leaves values are adjusted. This adjustment is made based on the tree's fitting.
* The **“Test and Score”** widget is an important widget where the tree score is determined based on

**For Yes:**

* **AUC (Area Under the Curve)** – 0.851 (very good score)
* **CA (Classification Accuracy)-** 0.817 (81.7% of the predictions were correct)
* **F1 Score –** 0.832 (83.2% well balanced)
* **Precision –** 0.762 (76.2% of the model is correctly predicted)
* **Recall-** 0.909 (90.9% of the model shows actual positive instances)
* **MCC (The Matthews correlation coefficient) –** 0.644 (Decent score which shows that the model is performing better than random guessing)

**For No:**

* **AUC (Area Under the Curve) –** 0.851 (very good score)
* **CA (Classification Accuracy)-** 0.817 (81.7% of the predictions were correct)
* **F1 Score –** 0.798 (79.8% well balanced)
* **Precision –** 0.889 (88.9% of the model is correctly predicted)
* **Recall-** 0.724(72.4% of the model shows actual positive instances)
* **MCC (The Matthews correlation coefficient) –** 0.644 (Decent score which shows that the model is performing better than random guessing)

These test scores help determine whether the model is well-developed and whether its performance is understandable.

* After these considerations, the tree is visualized using the **“Tree Viewer”** widget.