PAPER 3:

Machine Learning

Practical 2A:

Perform Data Loading, Feature selection .

Theory:

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression.

Three benefits of performing feature selection before modeling your data are:

* Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
* Improves Accuracy: Less misleading data means modeling accuracy improves.
* Reduces Training Time: Less data means that algorithms train faster.

Program:

from pandas import read\_csv

from sklearn.ensemble import ExtraTreesClassifier

# load data

url = "pima-indians-diabetes.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(url, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

# feature extraction

model = ExtraTreesClassifier(n\_estimators=10)

model.fit(X, Y)

print(names)

print(model.feature\_importances\_)

Output:

['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

[0.10725276 0.21109673 0.10056895 0.08541 0.07307367 0.14485006

0.11504959 0.16269824]

Practical 2B :

Perform Principal Component analysis

Theory:

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

Generally , this is called a data reduction technique. A property of PCA is that you can choose the number of dimensions or principal component in the transformed result.

Program:

import numpy

from pandas import read\_csv

from sklearn.decomposition import PCA

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

# load data

url = "pima-indians-diabetes.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = read\_csv(url, names=names)

array = dataframe.values

X = array[:,0:8]

Y = array[:,8]

# feature extraction

pca = PCA(n\_components=3)

fit = pca.fit(X)

# summarize components

print("Explained Variance: %s" % fit.explained\_variance\_ratio\_)

print(fit.components\_)

Output:

Explained Variance: [0.88854663 0.06159078 0.02579012]

[[-2.02176587e-03 9.78115765e-02 1.60930503e-02 6.07566861e-02

9.93110844e-01 1.40108085e-02 5.37167919e-04 -3.56474430e-03]

[-2.26488861e-02 -9.72210040e-01 -1.41909330e-01 5.78614699e-02

9.46266913e-02 -4.69729766e-02 -8.16804621e-04 -1.40168181e-01]

[-2.24649003e-02 1.43428710e-01 -9.22467192e-01 -3.07013055e-01

2.09773019e-02 -1.32444542e-01 -6.39983017e-04 -1.25454310e-01]]

Practical 3: Write a program to implement Decision Tree

Theory:

A Decision Tree is a supervised algorithm used in machine learning. It is using a binary tree graph (each node has two children) to assign for each data sample a target value. The target values are presented in the tree leaves. To reach to the leaf, the sample is propagated through nodes, starting at the root node. In each node a decision is made, to which descendant node it should go. A decision is made based on the selected sample’s feature. Decision Tree learning is a process of finding the optimal rules in each internal tree node according to the selected metric.

The decision trees can be divided, with respect to the target values, into:

Classification trees used to classify samples, assign to a limited set of values - classes. In scikit-learn it is DecisionTreeClassifier.

Regression trees used to assign samples into numerical values within the range. In scikit-learn it is DecisionTreeRegressor.

Program:

from matplotlib import pyplot as plt

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

# Prepare the data data

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# Fit the classifier with default hyper-parameters

clf = DecisionTreeClassifier(random\_state=1234)

model = clf.fit(X, y)

fig = plt.figure()

\_ = tree.plot\_tree(clf,

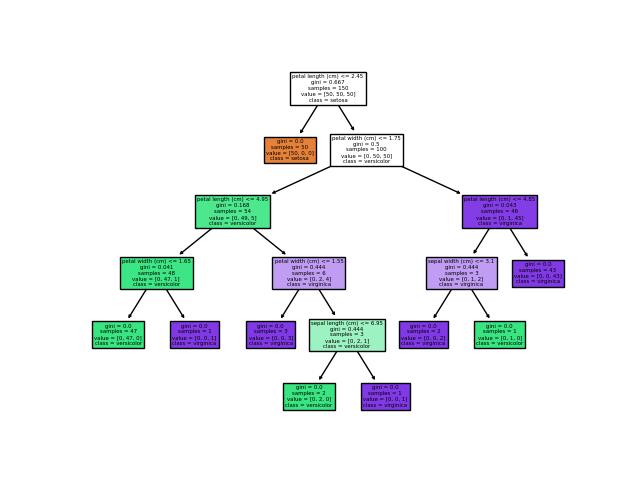
                   feature\_names=iris.feature\_names,

                   class\_names=iris.target\_names,

                   filled=True)

plt.show()

Output:



Practical 4.a :

For a given set of training data examples stored in a .CSV file implement Least Square Regression algorithm.

Theory:

In statistics, Linear Regression is a linear approach to model the relationship between a scalar response (or dependent variable), say Y, and one or more explanatory variables (or independent variables), say X.

Regression Line: If our data shows a linear relationship between X and Y, then the straight line which best describes the relationship is the regression line. It is the straight line that covers the maximum points in the graph.

A regression line is given as Y = a + b\*X where the formula of b and a are given as:

b = (nΣ(xiyi) – Σ(xi)Σ(yi)) ÷ (nΣ(xi2)-Σ(xi)2)

a = ȳ – b.x̄

where x̄ and ȳ are mean of x and y respectively.

* To find regression line, we need to find a and b.
* Calculate a, which is given by a = (\sum yi)/n - b \* (\sum xi)/n
* Calculate b, which is given by
* b = (n\*\sum(xi\*yi) - \sum (xi)\* \sum (yi))/(n\*\sum (xi)^{2}-(\sum xi)^{2})
* Put value of a and b in the equation of regression line.

Program:

"""

To find regression line, we need to find a and b.

Calculate a, which is given by a = (\sum yi)/n - b \* (\sum xi)/n

Calculate b, which is given by

b = (n\*\sum(xi\*yi) - \sum (xi)\* \sum (yi))/(n\*\sum (xi)^{2}-(\sum xi)^{2})

Put value of a and b in the equation of regression line.

"""

# Function to calculate b

def calculateB(x, y, n):

  # sum of array x

  sx = sum(x)

  # sum of array y

  sy = sum(y)

  # for sum of product of x and y

  sxsy = 0

  # sum of square of x

  sx2 = 0

  for i in range(n):

    sxsy += x[i] \* y[i]

    sx2 += x[i] \* x[i]

  b = (n \* sxsy - sx \* sy)/(n \* sx2 - sx \* sx)

  return b

# Function to find the

# least regression line

def leastRegLine(X,Y,n):

  # Finding b

  b = calculateB(X, Y, n)

  meanX = int(sum(X)/n)

  meanY = int(sum(Y)/n)

  # Calculating a

  a = meanY - b \* meanX

  # Printing regression line

  print("Regression line:")

  print("Y = ", '%.3f'%a, " + ", '%.3f'%b, "\*X", sep="")

# Driver code

# Statistical data

import pandas as pd

# Step 1 :Import libraries and dataset

datas = pd.read\_csv('data.csv')

print(datas )

X = datas.TEMPERATURE

Y = datas.PRESSURE

n = len(X)

leastRegLine(X, Y, n)

Output:

SNO TEMPERATURE PRESSURE

0 1 0 0.0002

1 2 20 0.0012

2 3 40 0.0060

3 4 60 0.0300

4 5 80 0.0900

5 6 100 0.2700

Regression line:

Y = -0.117 + 0.002\*X

Practical 4b.:

For a given set of training data examples stored in a .CSV file implement Linear Regression algorithm.

Theory:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

**Hypothesis function for Linear Regression :**



While training the model we are given :  
**x:** input training data (univariate – one input variable(parameter))  
**y:** labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.  
**θ1:** intercept  
**θ2:** coefficient of x

Program:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Step 1 :Import libraries and dataset

datas = pd.read\_csv('data.csv')

print(datas )

#Step 2: Dividing the dataset into 2 components

X = datas.iloc[:, 1:2].values

y = datas.iloc[:, 2].values

#Step 3: Fitting Linear Regression to the dataset

from sklearn.linear\_model import LinearRegression

lin = LinearRegression()

lin.fit(X, y)

plt.scatter(X, y, color = 'blue')

plt.plot(X, lin.predict(X), color = 'red')

plt.title('Linear Regression')

plt.xlabel('Temperature')

plt.ylabel('Pressure')

plt.show()

Output:

====================

SNO TEMPERATURE PRESSURE

0 1 0 0.0002

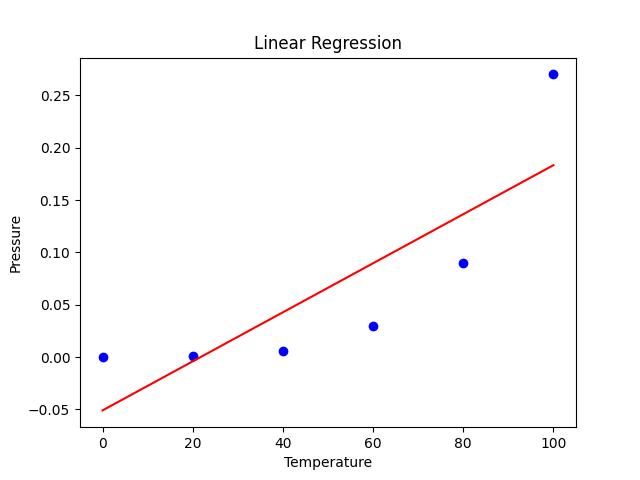
1 2 20 0.0012

2 3 40 0.0060

3 4 60 0.0300

4 5 80 0.0900

5 6 100 0.2700



Practical 6b.:

Implement the classification model using clustering for the following techniques with K means clustering with Prediction, Test Score and Confusion Matrix.

Theory:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

Program:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Step 1 :Import libraries and dataset

datas = pd.read\_csv('data.csv')

print(datas )

#Step 2: Dividing the dataset into 2 components

X = datas.iloc[:, 1:2].values

y = datas.iloc[:, 2].values

#Step 3: Fitting Linear Regression to the dataset

from sklearn.linear\_model import LinearRegression

lin = LinearRegression()

lin.fit(X, y)

plt.scatter(X, y, color = 'blue')

plt.plot(X, lin.predict(X), color = 'red')

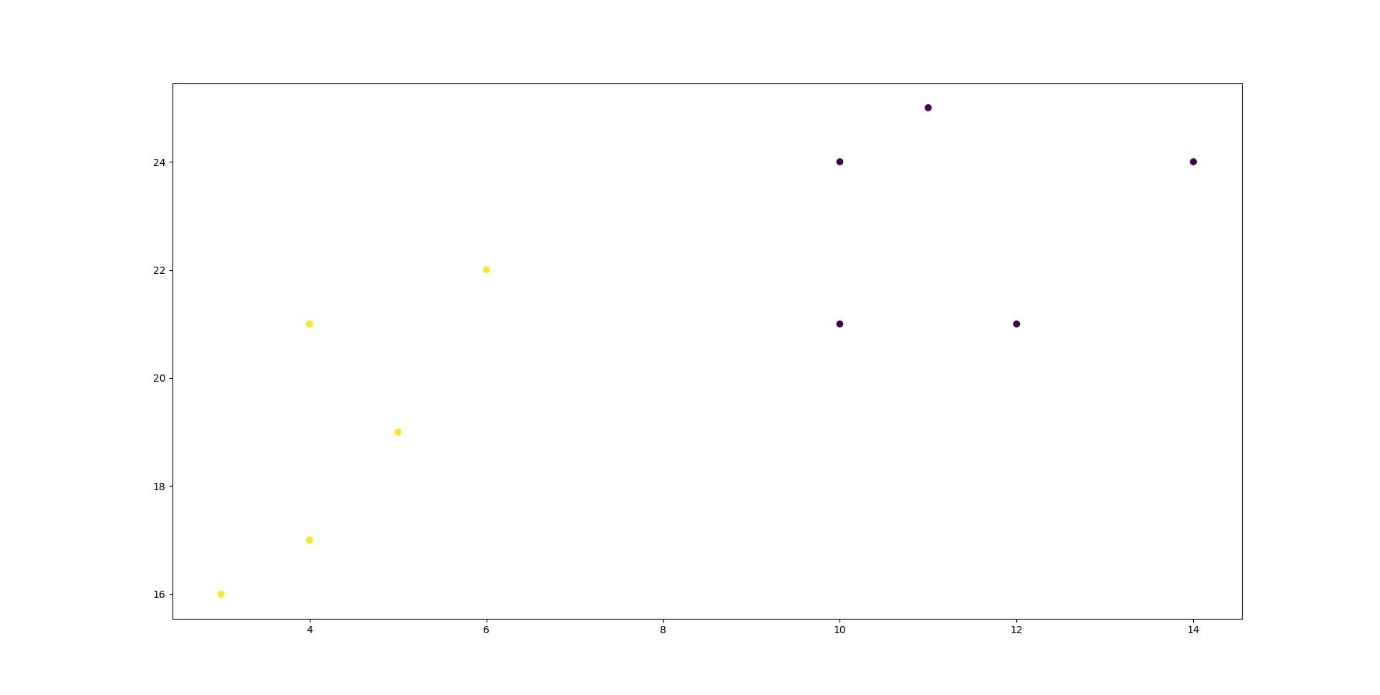
plt.title('Linear Regression')

plt.xlabel('Temperature')

plt.ylabel('Pressure')

plt.show()

Output:



Practical 7:

Implement the classification model using clustering for the following techniques with hierarchical clustering with Prediction, Test Score and Confusion Matrix.

Theory:

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

Agglomerative: Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach.

Program:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

from scipy.cluster.hierarchy import dendrogram, linkage

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))

hierarchical\_cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

labels = hierarchical\_cluster.fit\_predict(data)

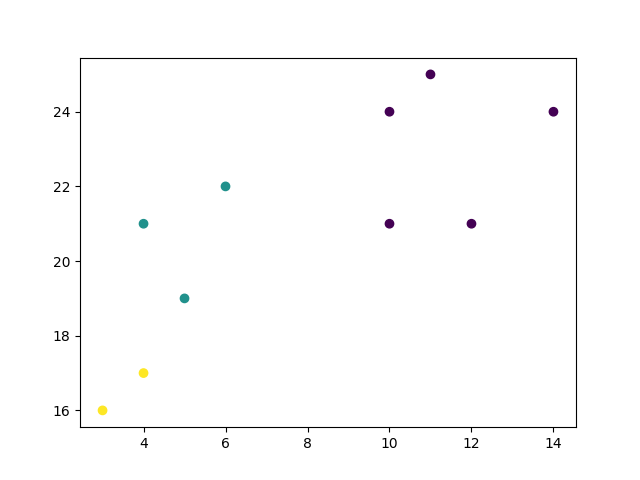
print(labels)

plt.scatter(x, y, c=labels)

plt.show()

Output:

[1 1 0 2 2 0 0 1 0 0]



PAPER 4:

ROBOTIC PROCESS AUTOMATION

Practical 1

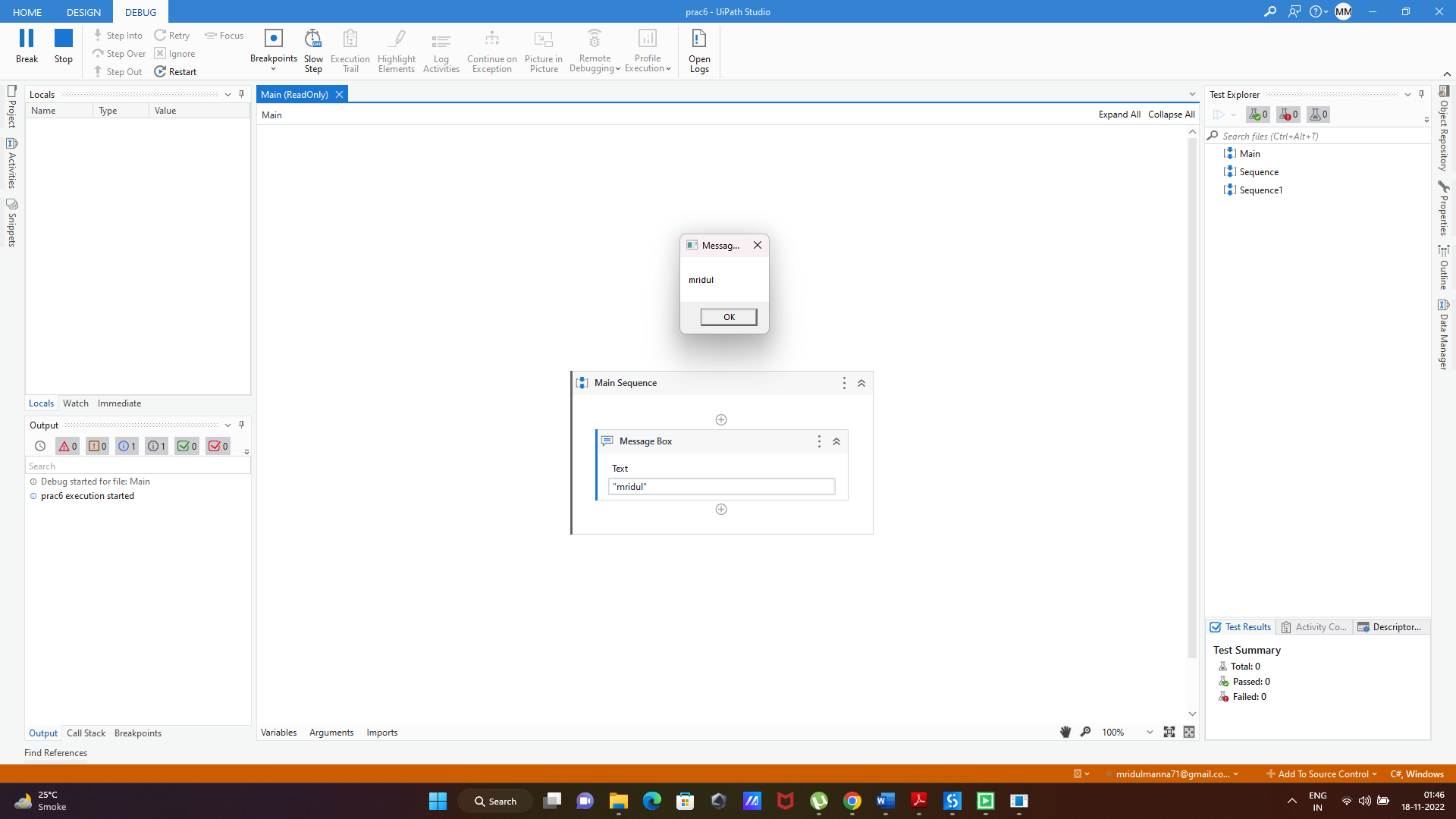
Create a simple sequence-based project

Theory:

A sequence is a basic automation process in RPA. It is used when the process is divided into steps and sequential. It is simpler and easier to use.

Steps:

1. Open UI path studio.
2. In uipath studio start page, under new project click on process
3. A dialog box appears, give a name to the process and select the language as C# and click on create.
4. Once a process is created, go to activity panel and add new sequence
5. Add name to the sequence and click on create. A sequence is created.
6. Once the sequence is created, add a message box.
7. Add the text you want to display and click on debug at the top ribbon.
8. This is how we create a simple sequence.

Output:

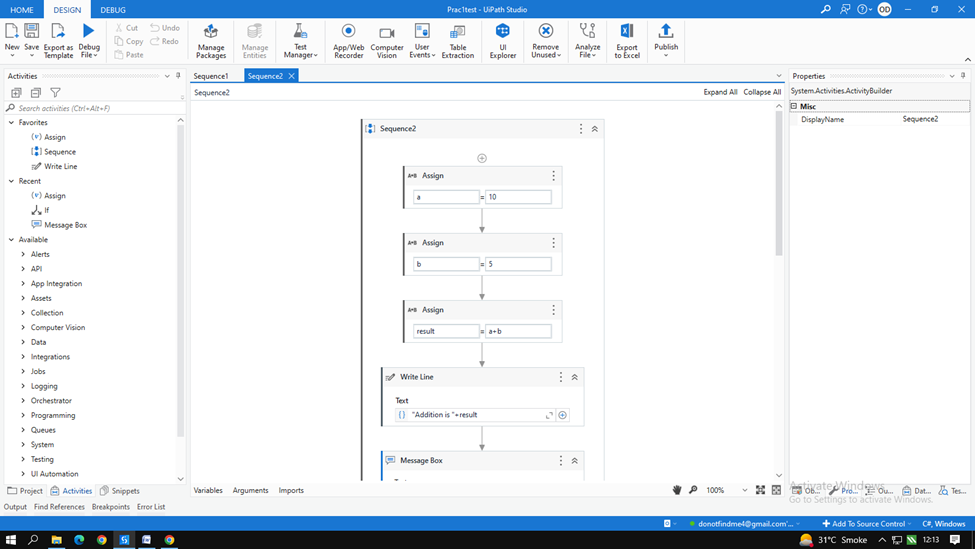
Practical 2a

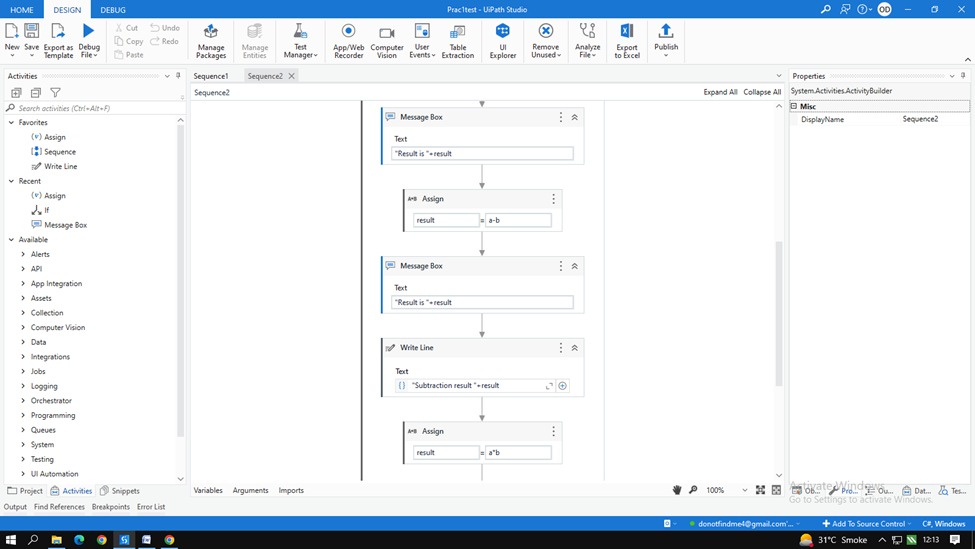
Automate UiPath Number Calculation (Subtraction, Multiplication, Division of numbers).

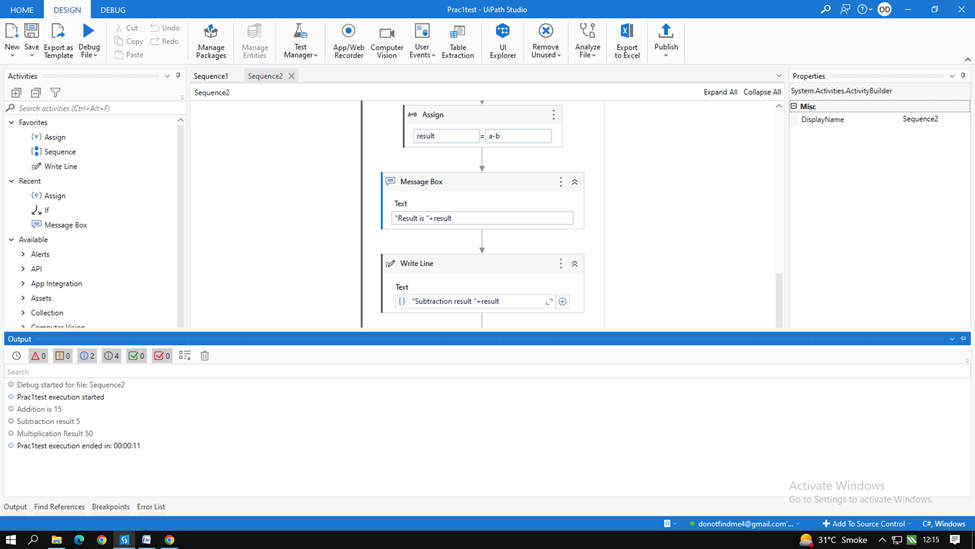
Steps:

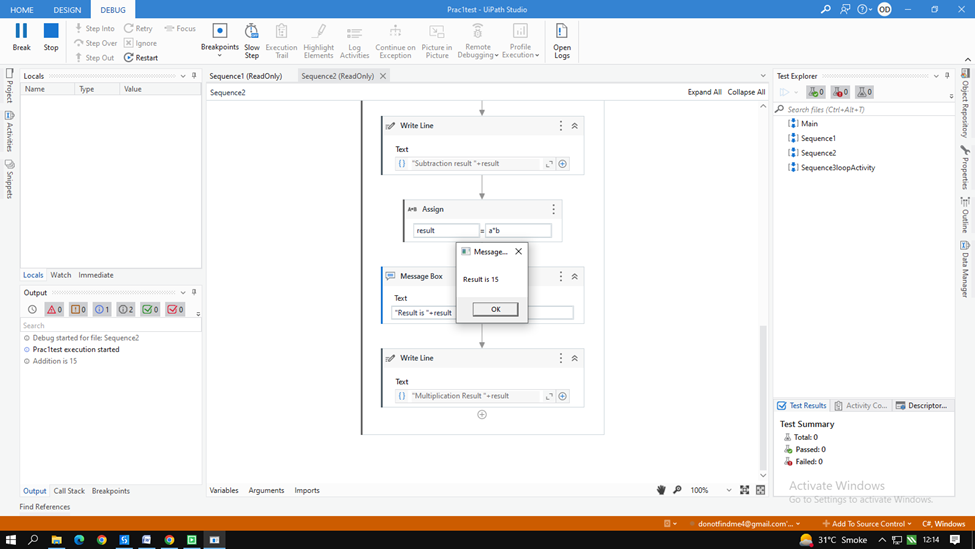
1. Create a new sequence.
2. In the sequence, drag and drop an assign box to assign a variable. Assign a variable “a” with value 10.
3. On the variables tab, create the variable and assign the variable type as Int32.
4. Repeat steps 2 and 3 to create another variable “b” of integer type and give it the value 15.
5. Add a third assign box and create a variable “result” that will store the value of the operation between a and b. (a+b). This variable will also be integer.
6. Once done, drag and drop a write line box and add: “Addition is” + result since the output will be in string format.
7. Repeat steps from 5 and 6 thrice and change the operation as c=a-b, c=a\*b, c=a/b.
8. Run the automation.

Output:









Practical 2b

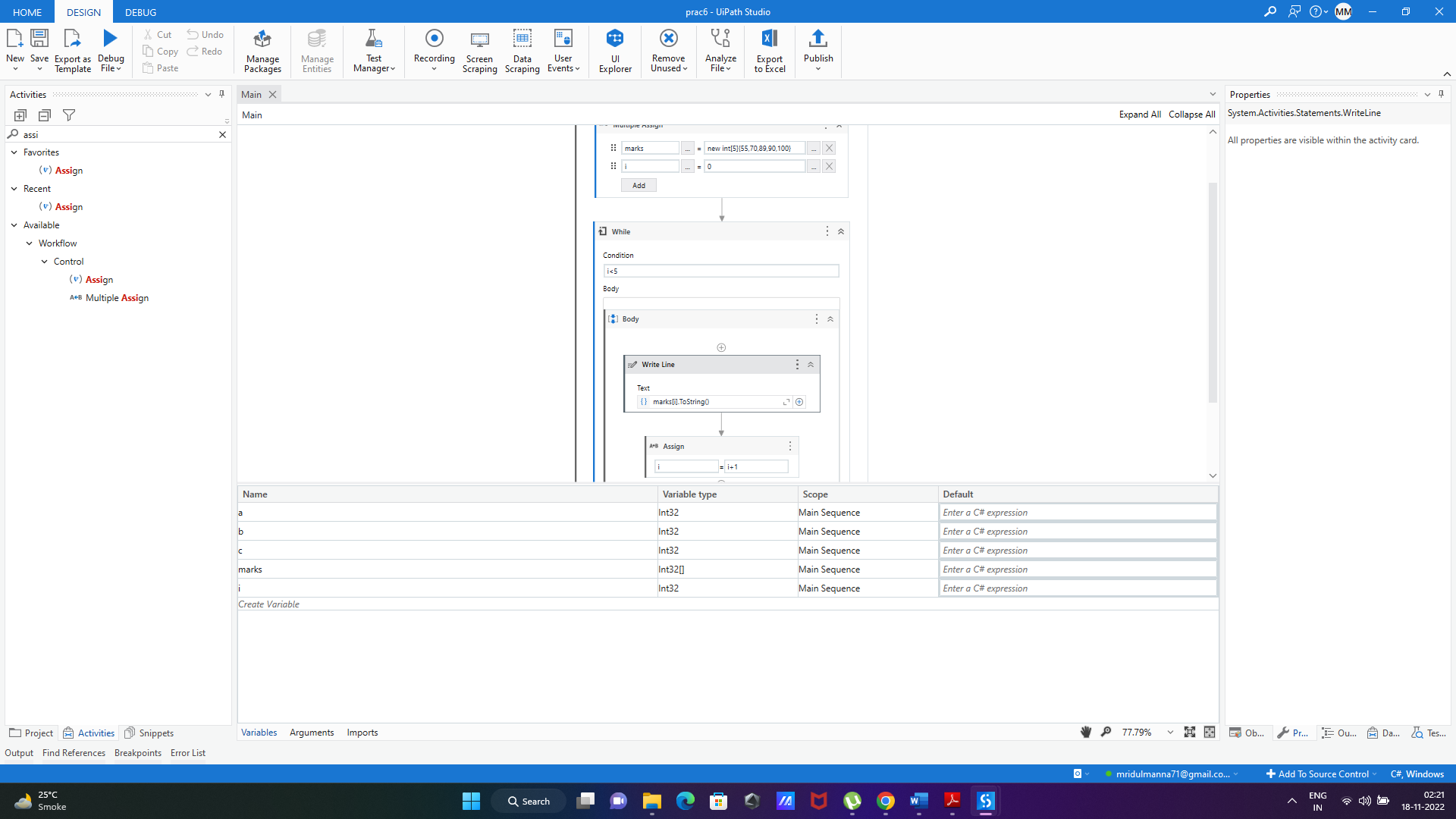
Create an automation UiPath project using different types of variables (array, data table)

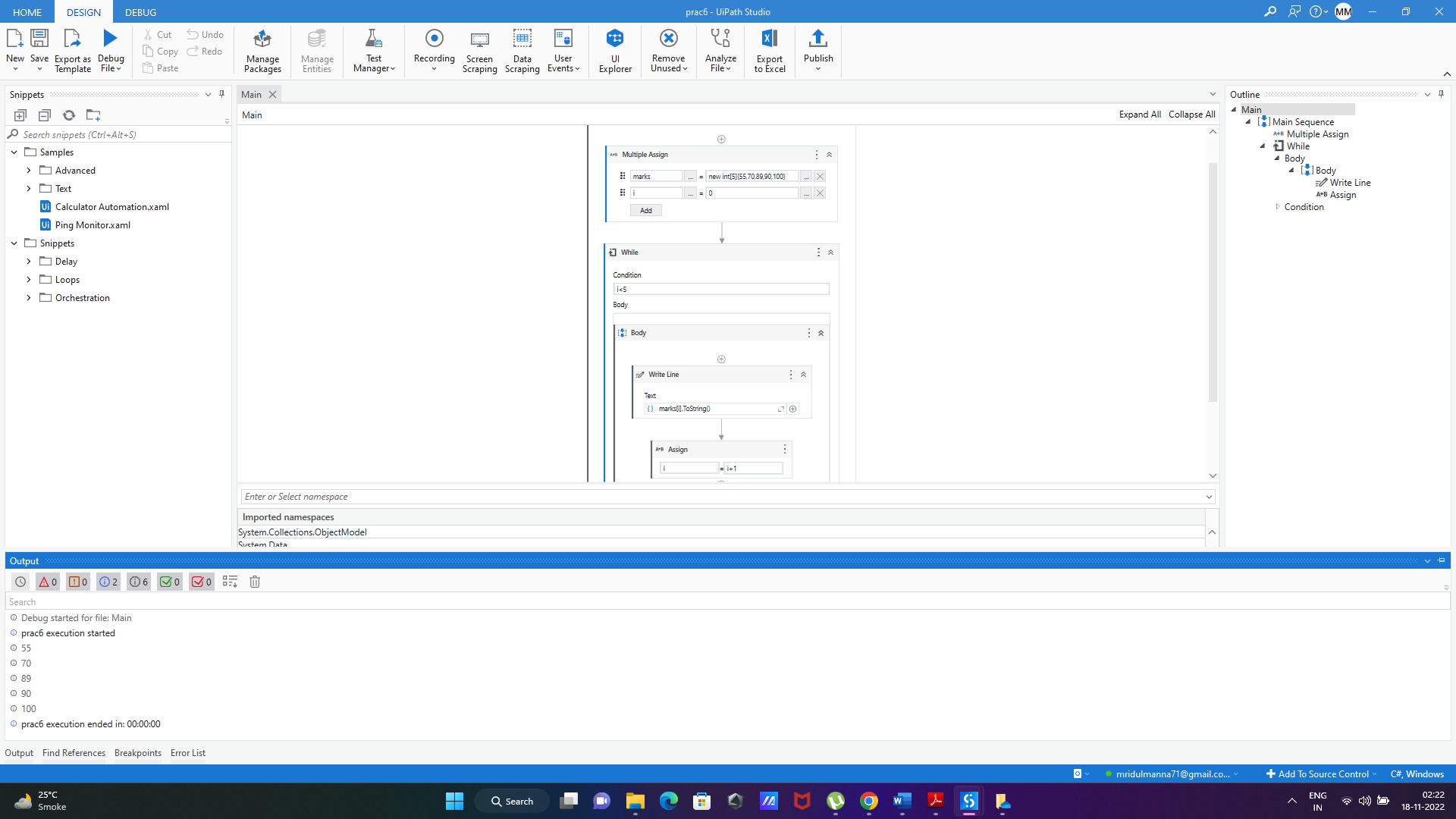
Steps: To create array

1. Create a new sequence.
2. Drag and drop multiple assign activity.
3. Create an array variable a where,

* marks = new int[5]{99,100,50,22,98}

1. Once done, go to variable option and change the data type of the array to array of [T]. Click on the option and select the array type as int 32.
2. Create another variable i=0 where I will be the iteration.
3. Drag and drop while activity. In the condition part add i<5
4. In the body drag and drop writeline and add marks[i].ToString() to display the array
5. Finally drag and drop a new assign and add i=i+1 to increase the iteration by 1. This will display all the members of the array.
6. Run the automation and check the output.

Output:



Practical 2b

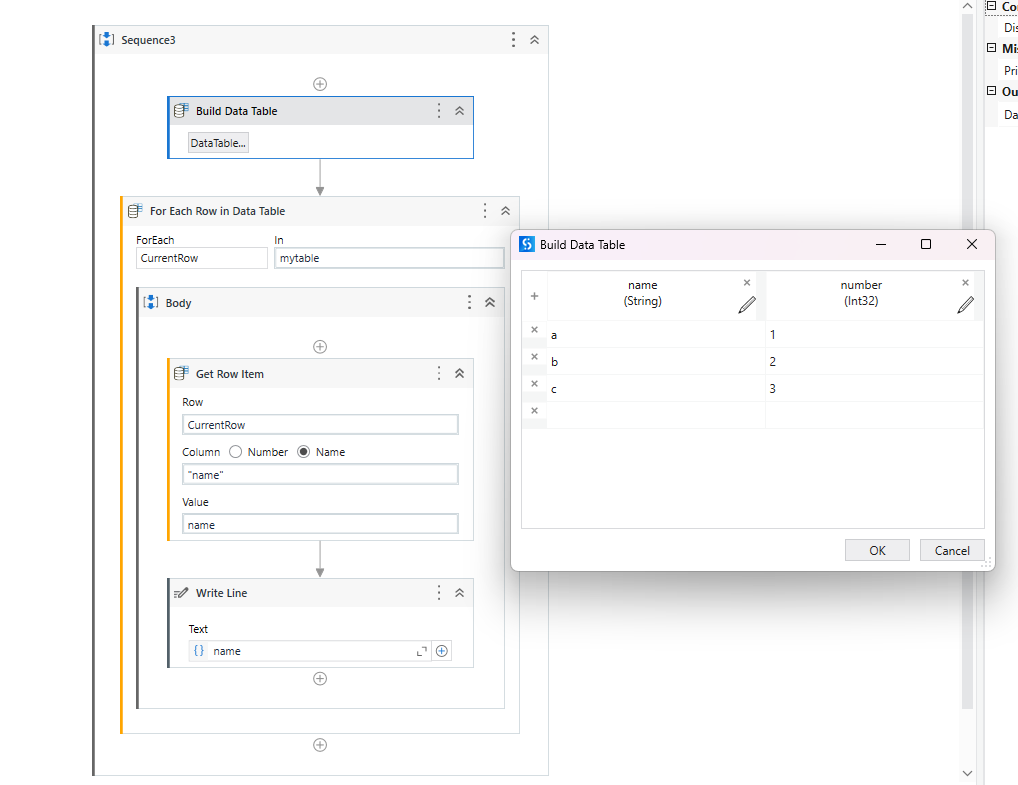
Create an automation UiPath project using different types of variables (array, data table)

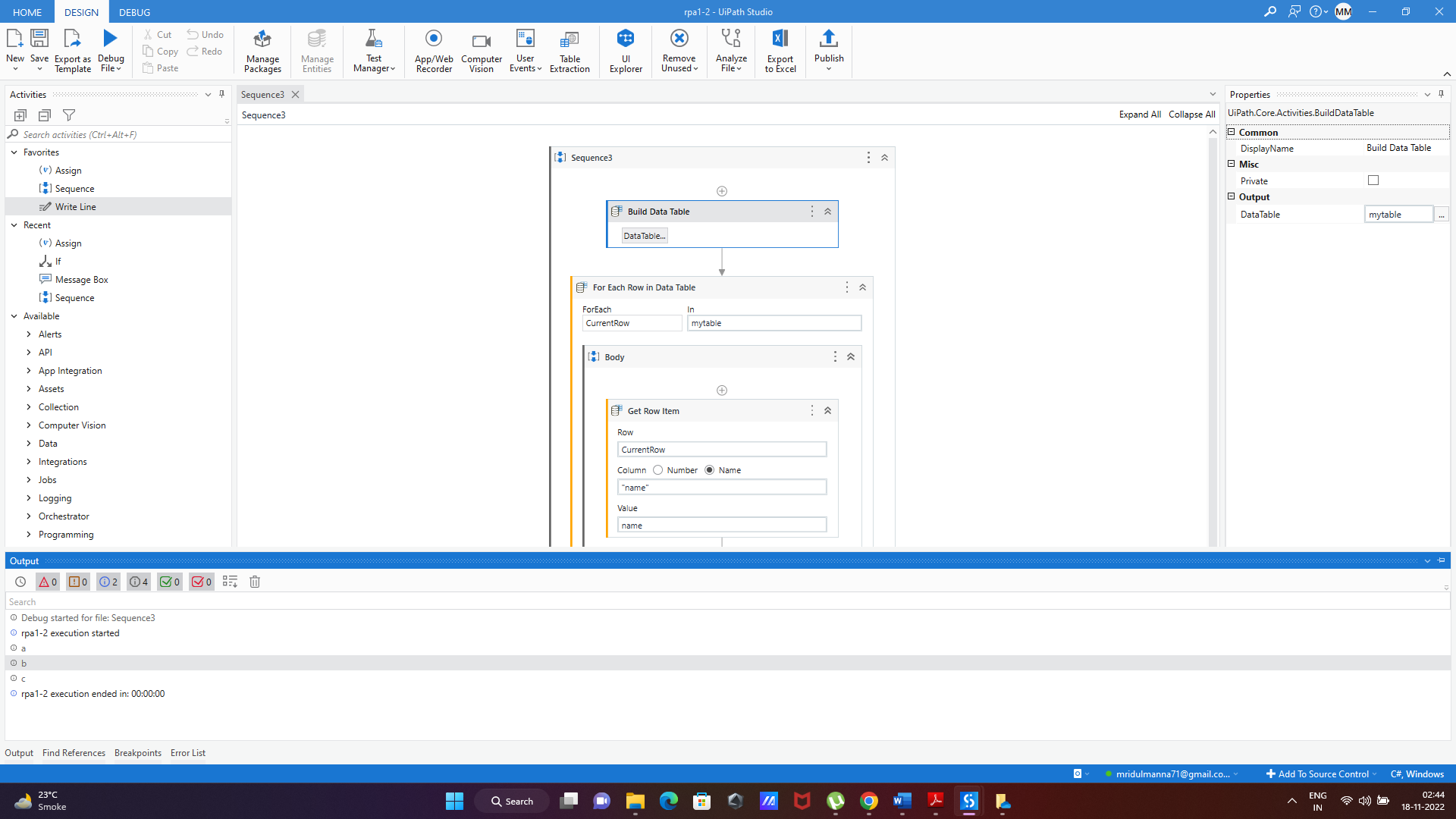
Steps: To create datatable.

1. Create a new sequence.
2. Drag and drop build data table activity from activity panel.
3. In the build data table acclivity, click on data table button and add the data you want to add in the data table.
4. Once done, on the right side in the properties tab, right click on the output option and click on create variable and add the variable name as mytable.
5. Drag and drop the for each row in data table activity in the sequence.
6. ForEach= CurrentRow and In=mytable. This will take the value in each row in the data table.
7. Once done, in the body of the for each activity, drag and drop the “get row item activity” and add the following:

* Row= CurrentRow
* Choose column as Name
* Add column name as “name”
* Add value = name

1. On the right-hand side in the properties panel, right click on output and select create variable
2. Create the variable as name.
3. Once done, drag and drop a writeline activity and add “name” as the column name.
4. Run the automation and check the output.



Output: This will display the data in the name column of the datatable.

