**ONE CREDIT COURSE**

**PYTHON -DATA ANALYTICS**

**NAME: NABIL MOHAMED T**

**REG NO:927621BIT069**

**M.KUMARASAMY COLLEGE OF ENGINEERING ,**

**KARUR**

**HOUSING PRICE**

**Code:**

1. import numpy as np

import pandas as pd

2. dataset = pd.read\_csv("housing\_prices\_SLR.csv") # first we need upload the dataset

dataset.head() # will shoe first 5 rows

3. x = dataset.iloc[:,0:1]

y = dataset.iloc[:,1]

x.head()

4. y.head()

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,train\_size=0.8,random\_state=42)

6. from sklearn.linear\_model import LinearRegression

model = LinearRegression()

7. model.fit(x\_train,y\_train)

8. samplepredciton = model.predict([[1240]])

print(samplepredciton)

9. y\_pred = model.predict(x\_test) # predicting for all x\_test

y\_pred

10. y\_test

11. y\_pred

12. list(zip(y\_test,y\_pred))

13. import numpy as np

# Hypothetical housing dataset

areas = np.array([1000, 1030, 1060, 1090])  # House areas in square feet

prices = np.array([5618, 5201, 4779, 5245])  # House prices in dollars

# Example of a simple linear regression model (hypothetical coefficients)

m = 200  # slope

b = 50000  # intercept

# Make predictions

predicted\_prices = m \* areas + b

# Calculate squared differences

squared\_diff = (prices - predicted\_prices) \*\* 2

# Calculate mean squared error

mse = np.mean(squared\_diff)

print("Actual Prices:", prices)

print("Predicted Prices:", predicted\_prices)

print("Squared Differences:", squared\_diff)

print("Mean Squared Error:", mse)

14. from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

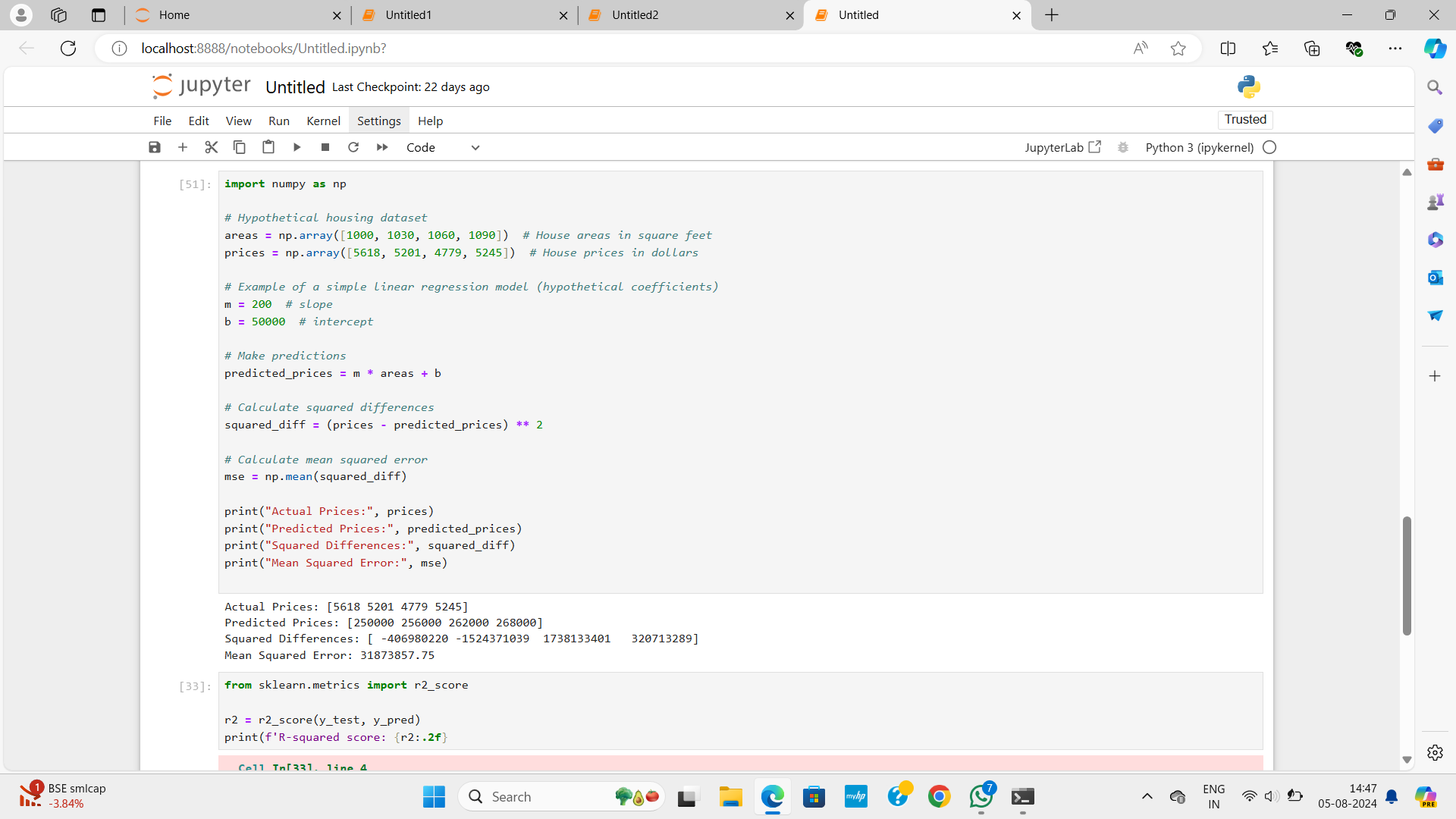
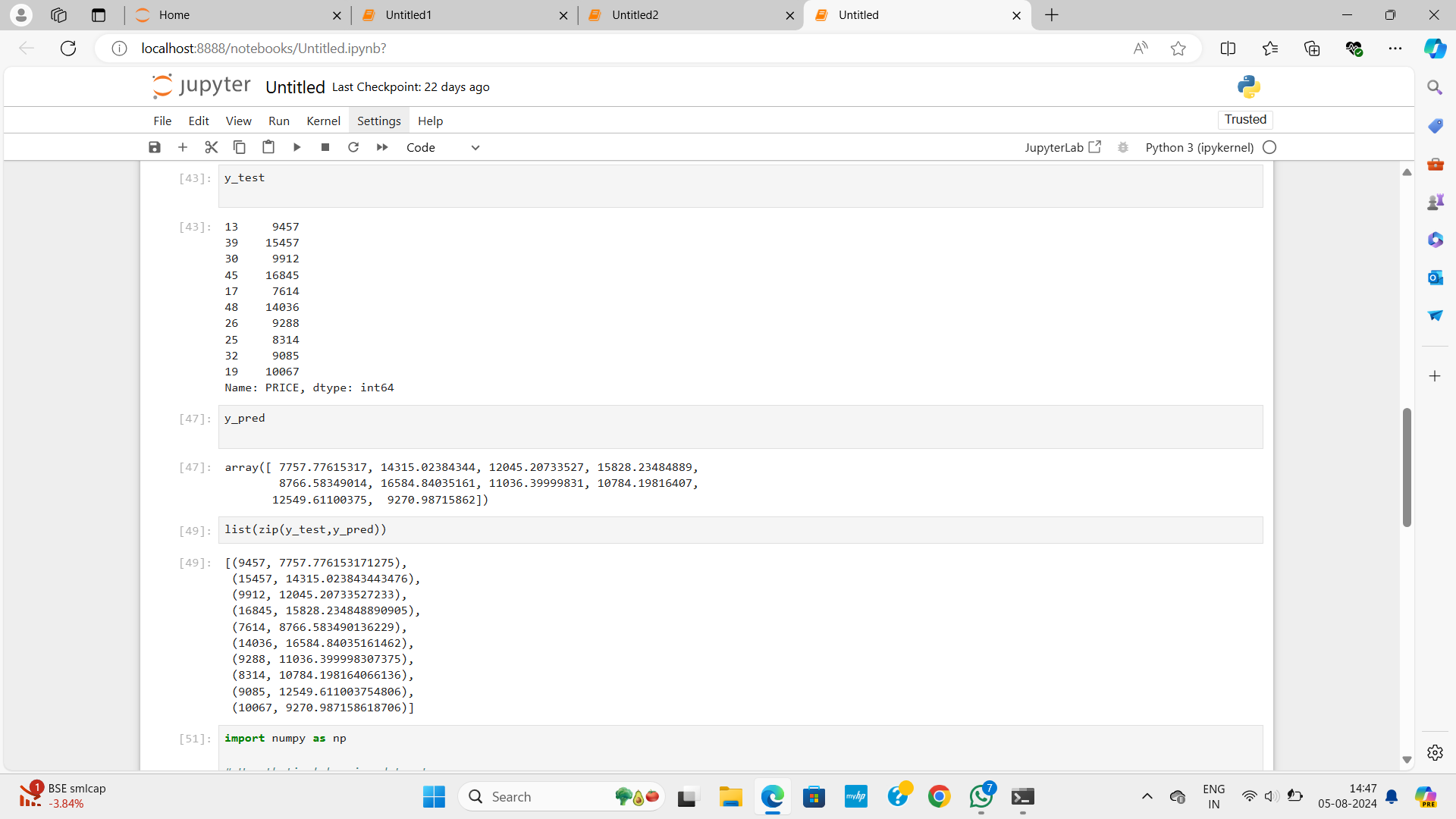
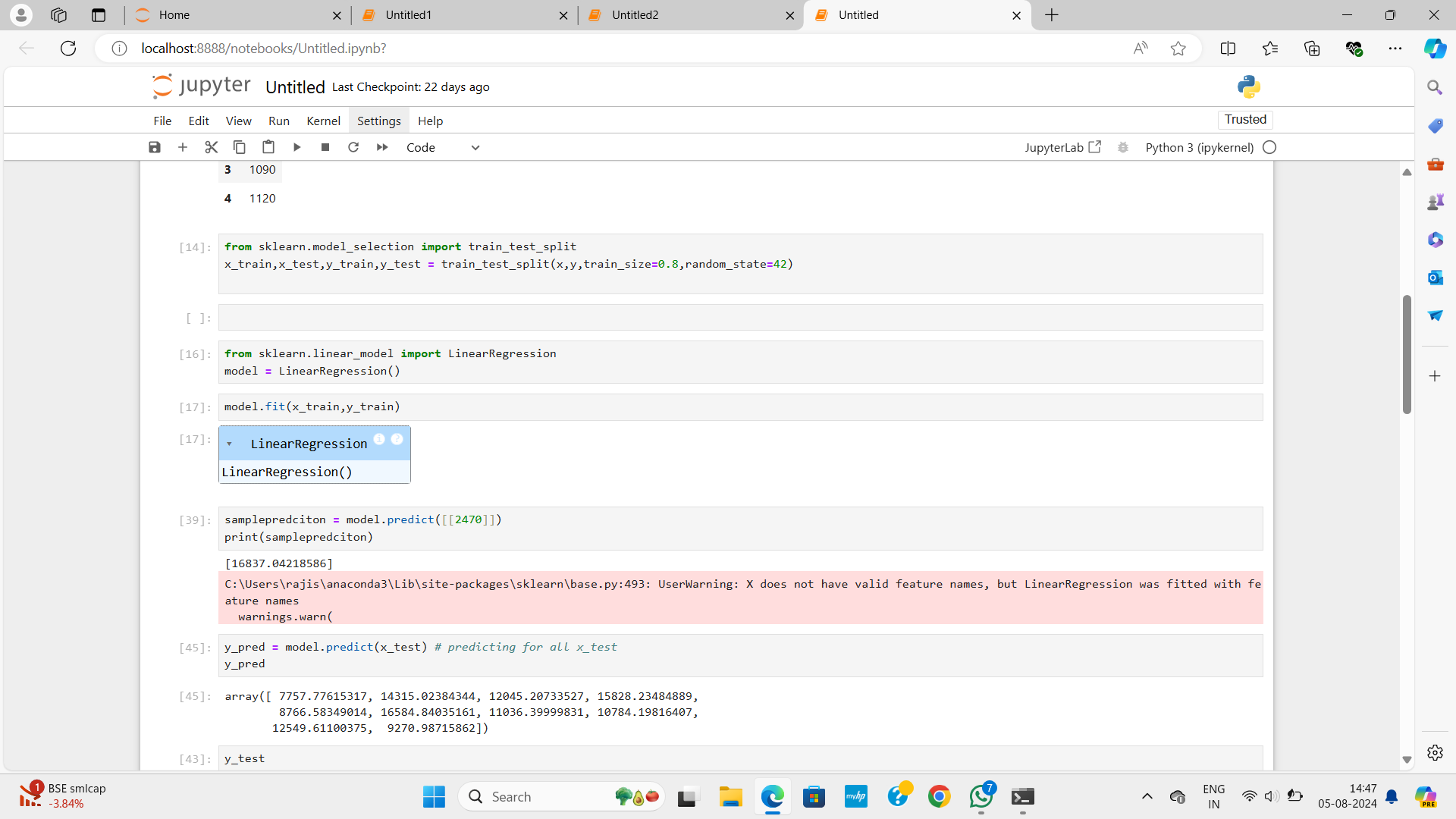
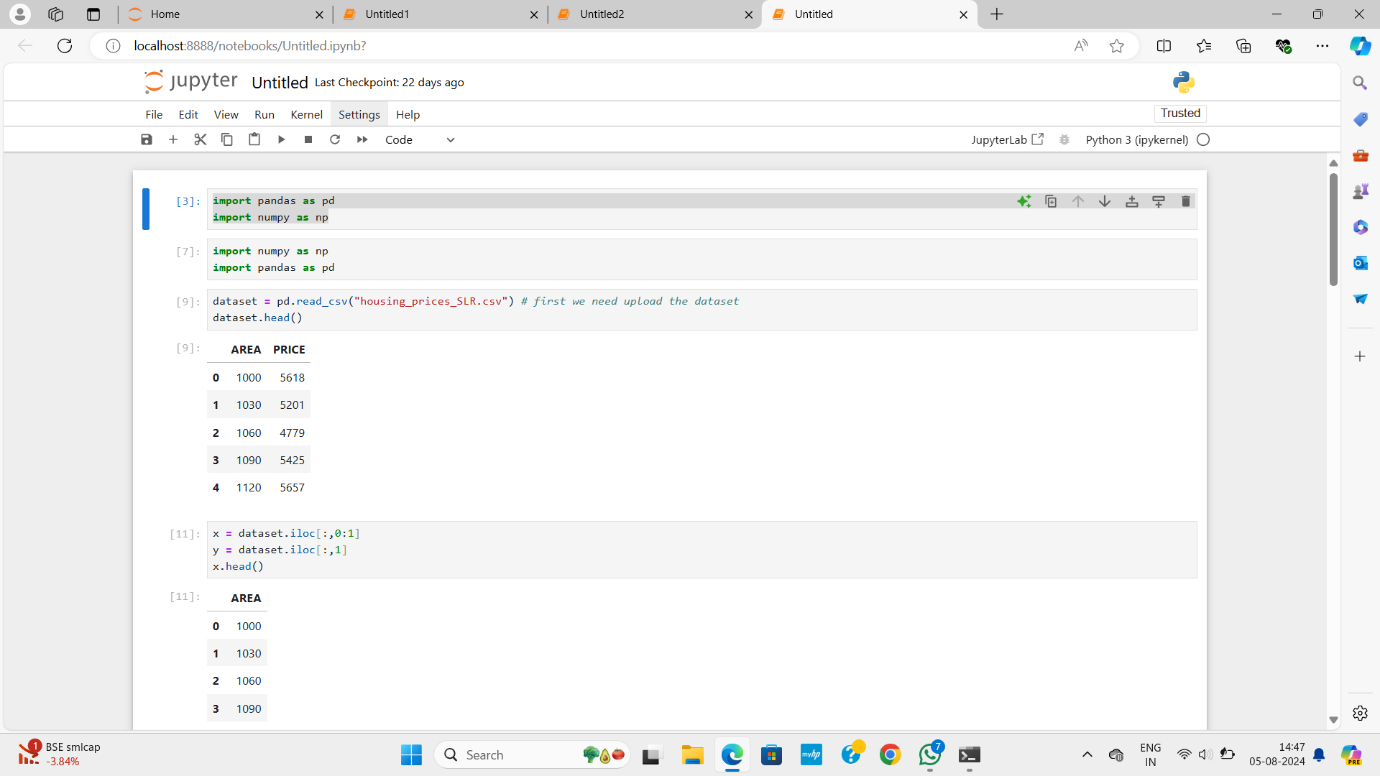
print(f'R-squared score: {r2:.2f}

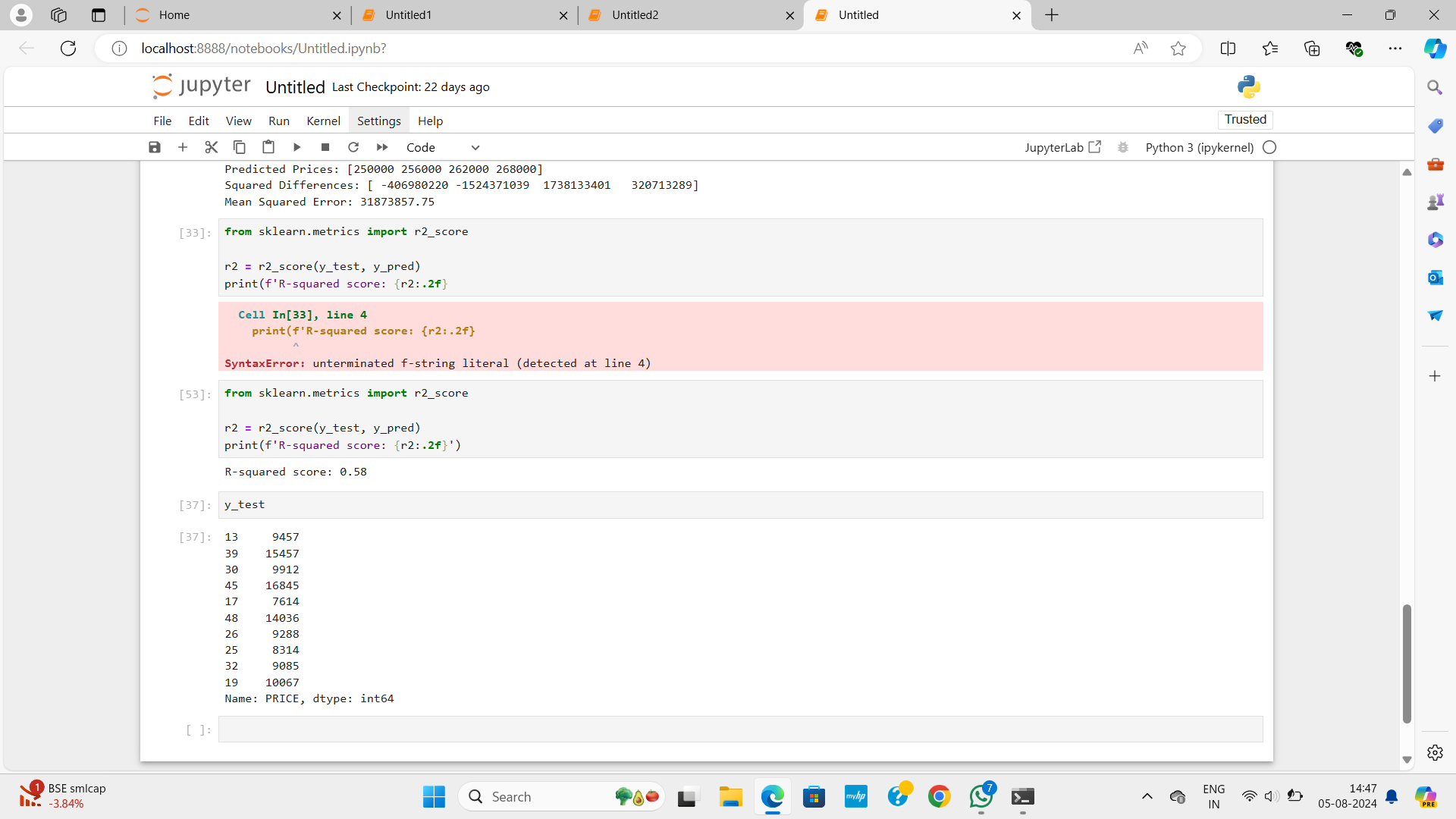
15. from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

print(f'R-squared score: {r2:.2f}')

16. y\_test





**MATRIX**

**Code :**

1. import pandas as pd

import numpy as np

2. dataset=pd.read\_csv("Iris\_dataset.csv")

3.dataset

4. x=dataset.iloc[:,1:5]

5.x

6. y=dataset.iloc[:,5:]

7.y

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,train\_size=0.8,random\_state=42)

9. from sklearn.linear\_model import LogisticRegression

model =  LogisticRegression()

10. model.fit(x\_train,y\_train)

11. model.predict([[5.1,3.5,1.4,0.2]])

12. y\_test

13. y\_pred

14. y\_test

15. y\_pred

16. from sklearn.metrics import confusion\_matrix

conf\_mat = confusion\_matrix(y\_test,y\_pred)

print("Confusion Matrix:")

print(conf\_mat)

17. import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report, precision\_score, recall\_score, f1\_score

conf\_mat = confusion\_matrix(y\_test, y\_pred)

plt.imshow(conf\_mat, interpolation='nearest')

plt.title("Confusion Matrix")

plt.xlabel("Predicted Class")

plt.ylabel("Actual Class")

plt.colorbar()

plt.show()

plt.plot([accuracy, precision, recall, f1])

plt.xlabel("Evaluation Metric")

plt.ylabel("Value")

plt.title("Evaluation Metrics")

plt.legend(["Accuracy", "Precision", "Recall", "F1-score"])

plt.show()

18. import matplotlib.pyplot as plt

conf\_mat = confusion\_matrix(y\_test, y\_pred)

plt.imshow(conf\_mat, interpolation='nearest')

plt.title("Confusion Matrix")

plt.xlabel("Predicted Class")

plt.ylabel("Actual Class")

plt.colorbar()

plt.show()

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)

plt.plot([accuracy])

plt.xlabel("Evaluation Metric")

plt.ylabel("Value")

plt.title("Accuracy")

plt.show()

plt.subplot(1,2,2)

plt.plot([precision, recall, f1])

plt.xlabel("Evaluation Metric")

plt.ylabel("Value")

plt.title("Precision, Recall, F1-score")

plt.legend(["Precision", "Recall", "F1-score"])

plt.show()

19. import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

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

from sklearn.metrics import confusion\_matrix

from sklearn.linear\_model import LogisticRegression

iris = load\_iris()

X = iris.data[:, :2]  # we only take the first two features.

y = iris.target

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)

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

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

conf\_mat = confusion\_matrix(y\_test, y\_pred)

plt.imshow(conf\_mat, interpolation='nearest')

plt.title("Confusion Matrix")

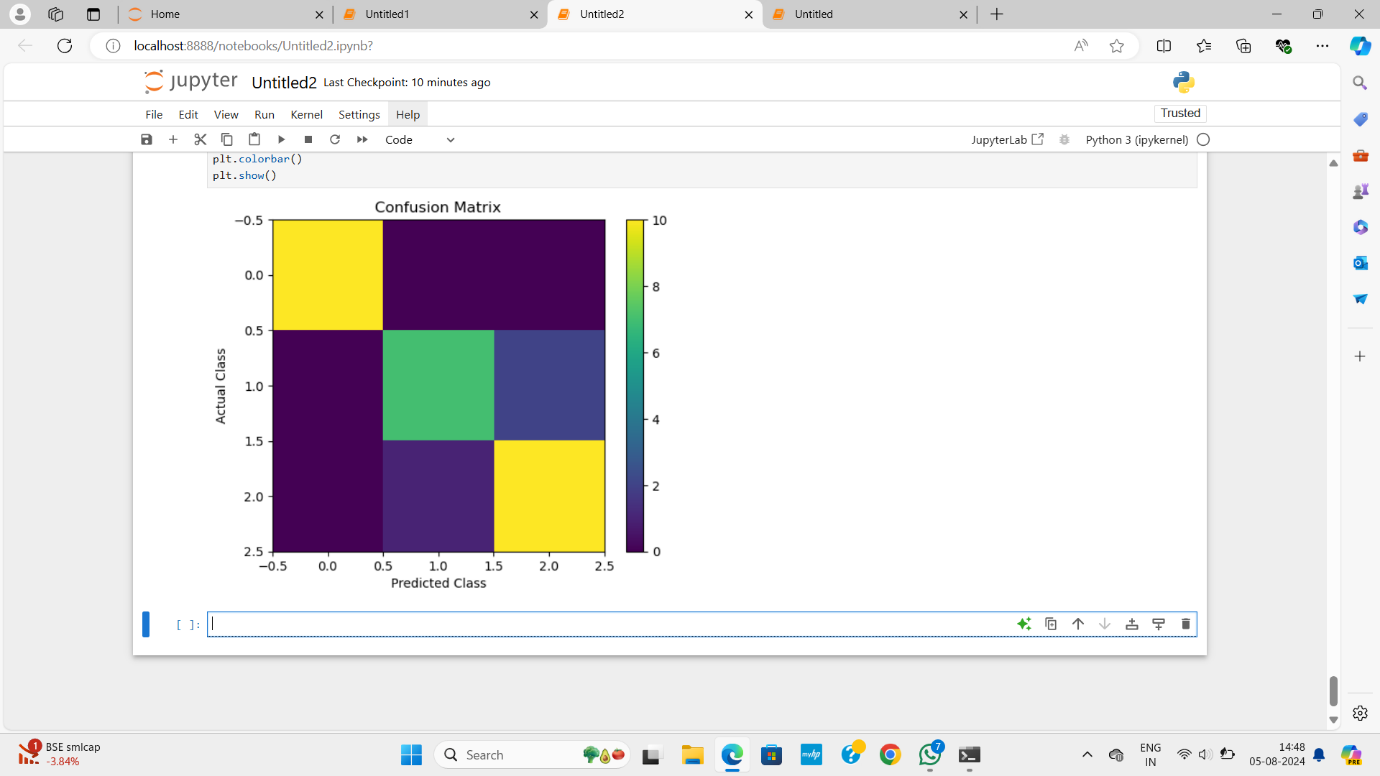
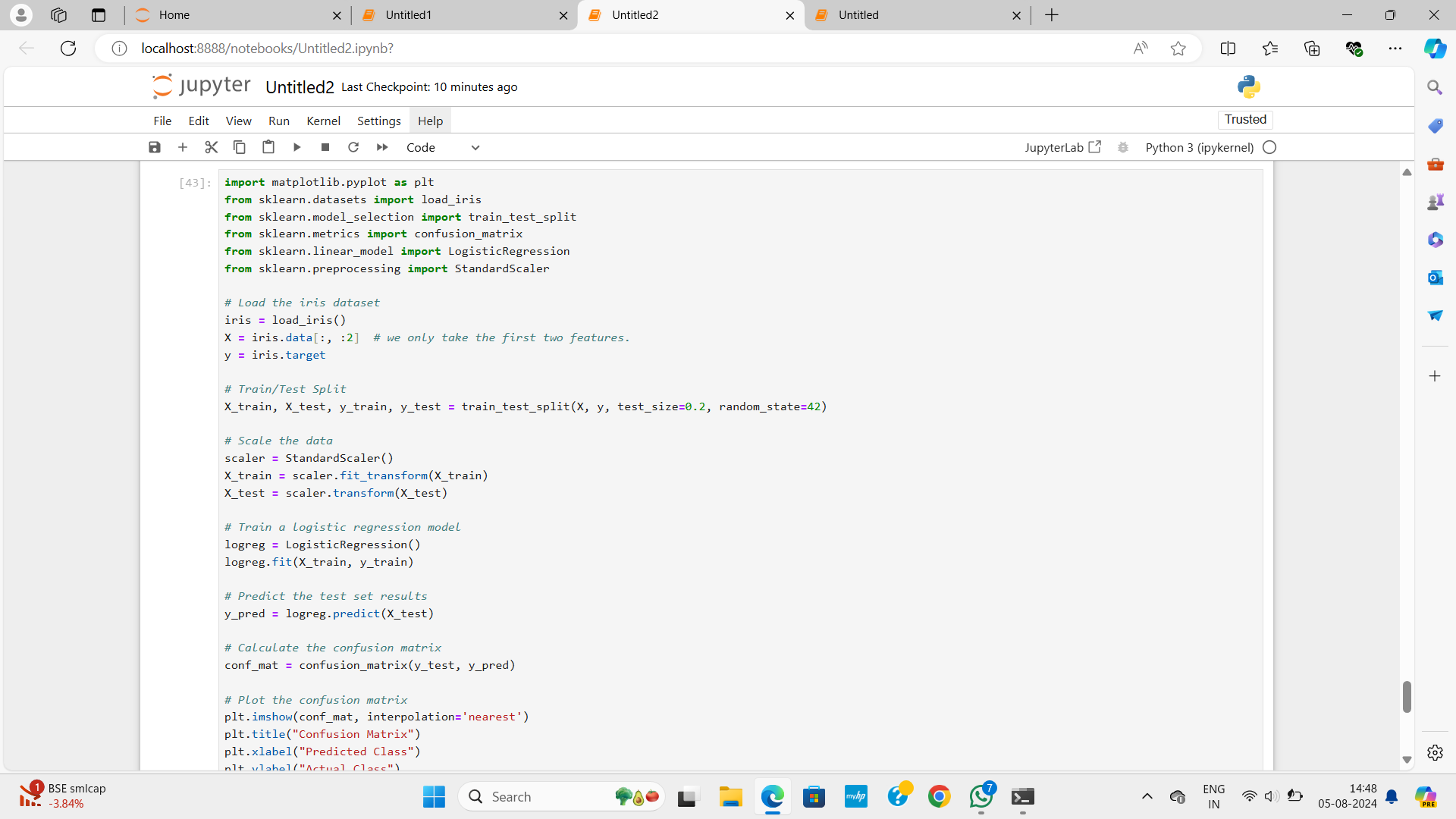
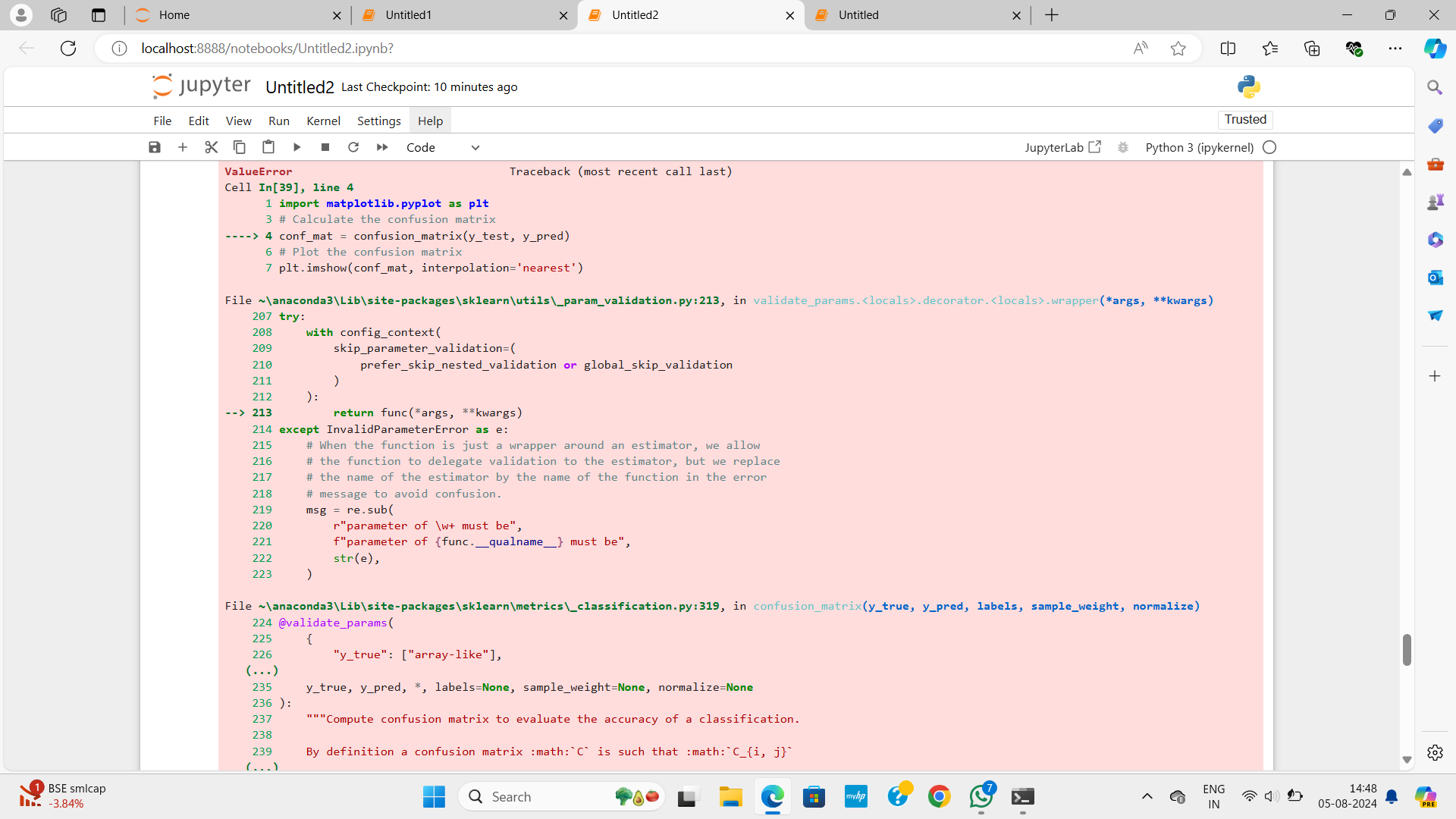
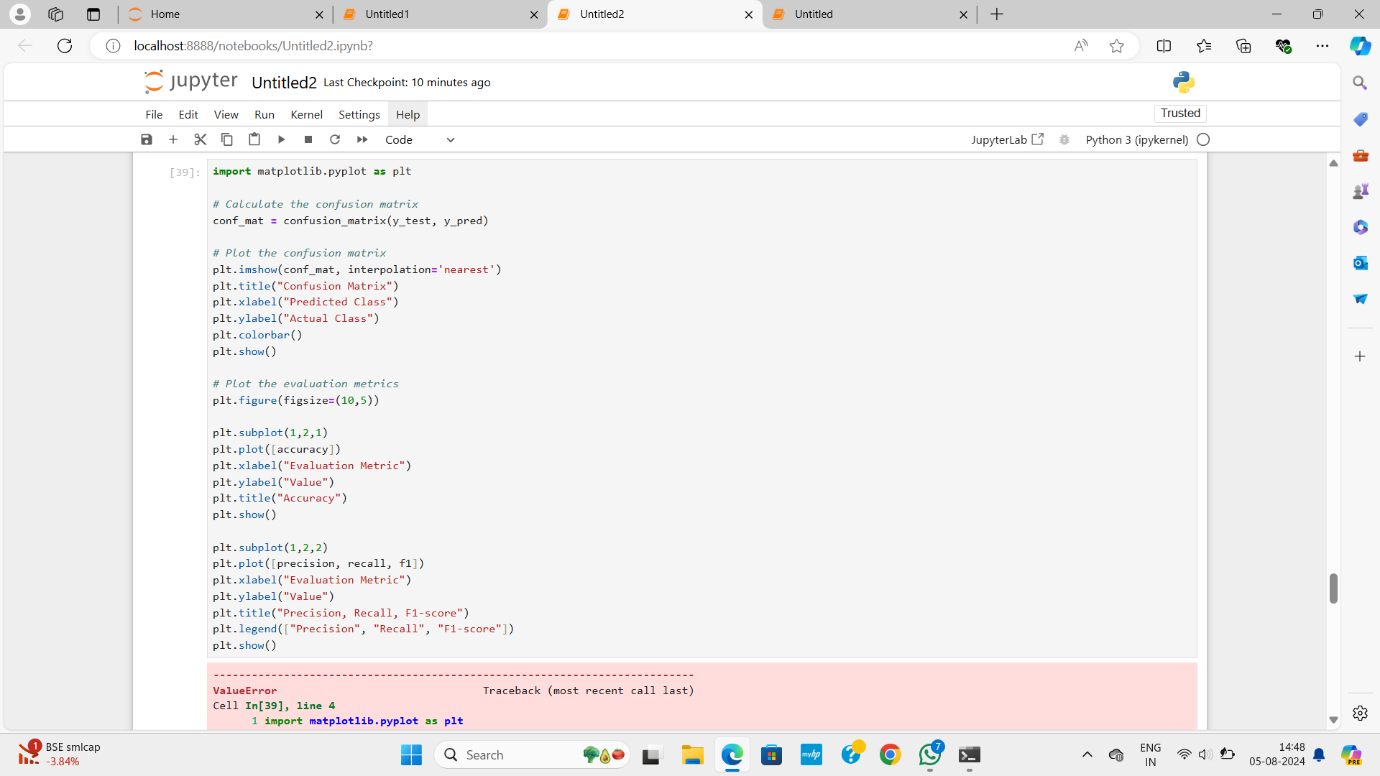
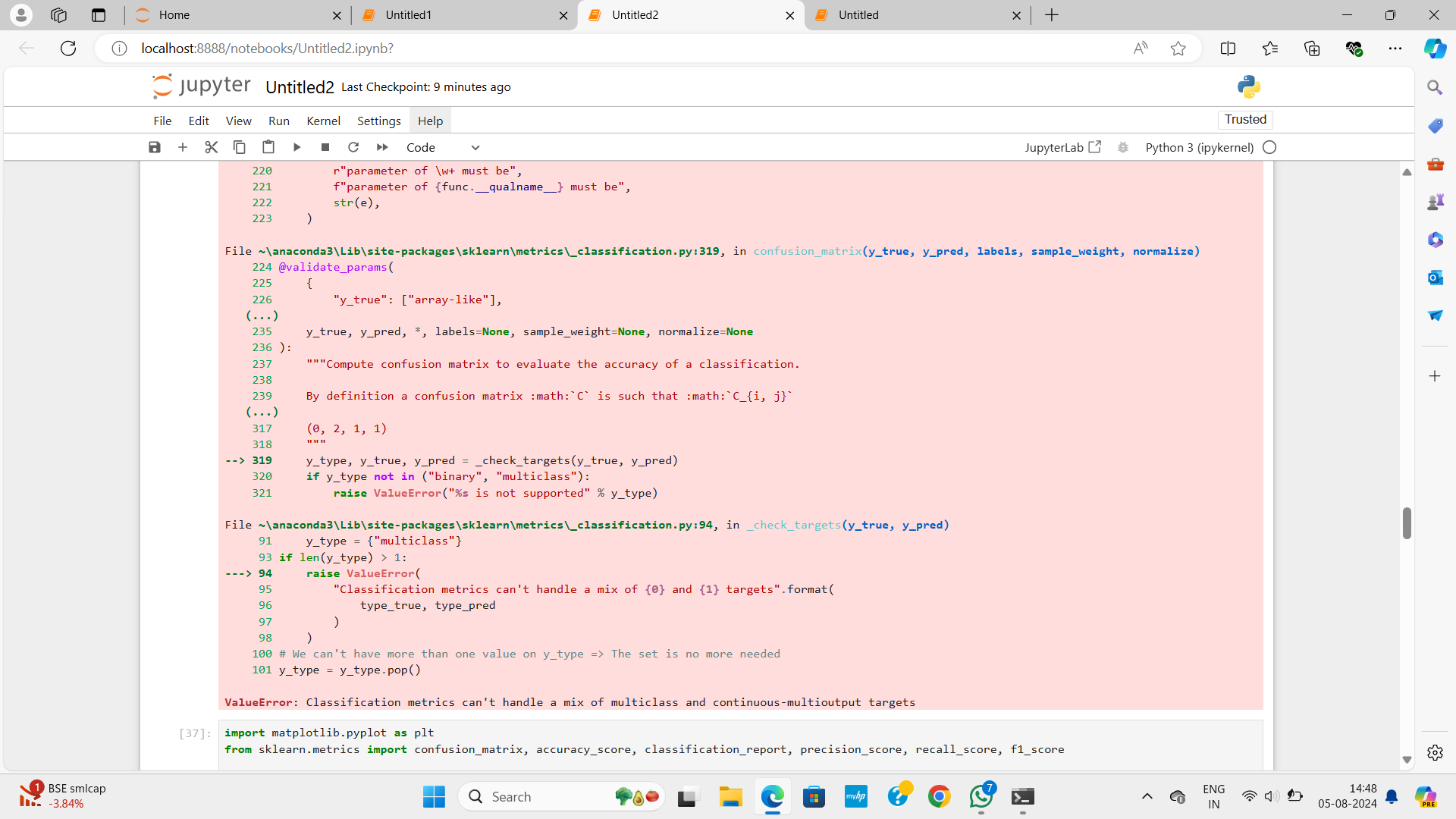
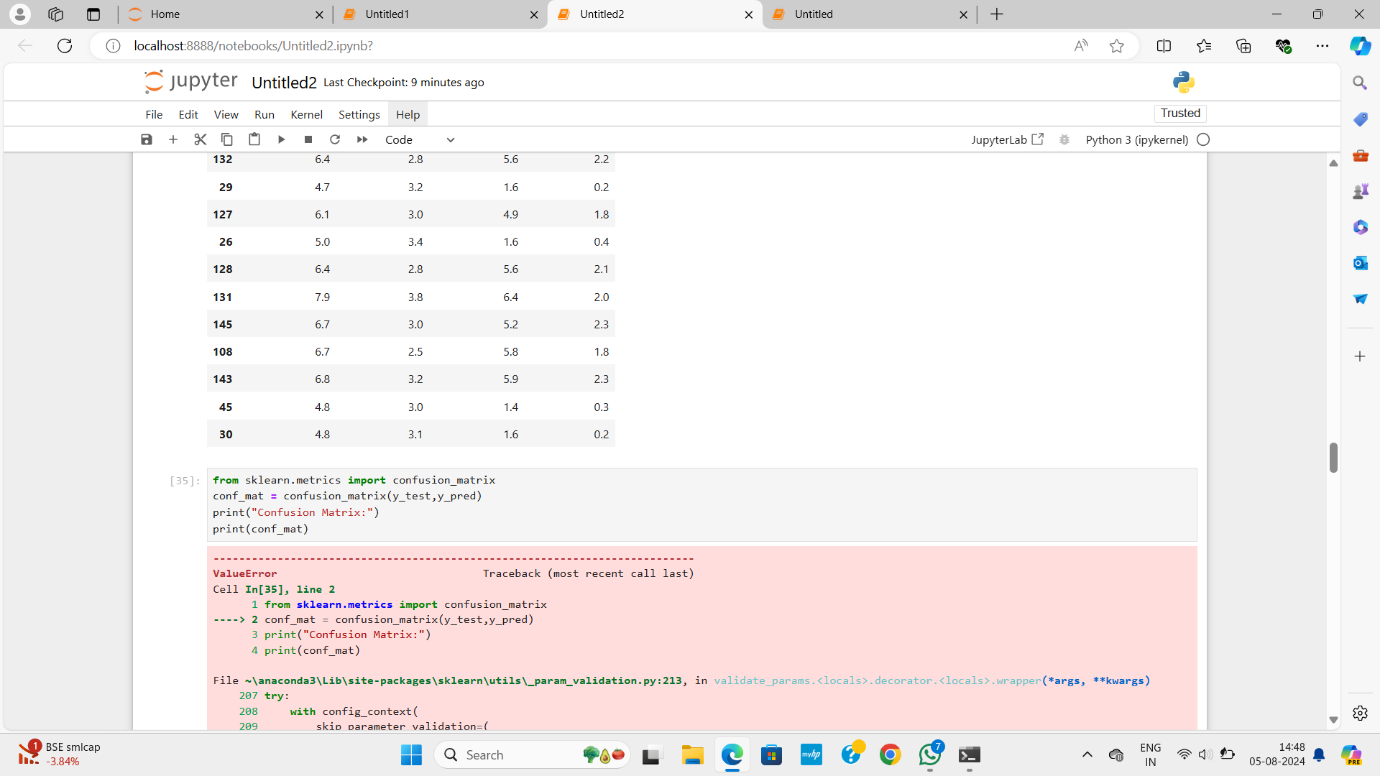
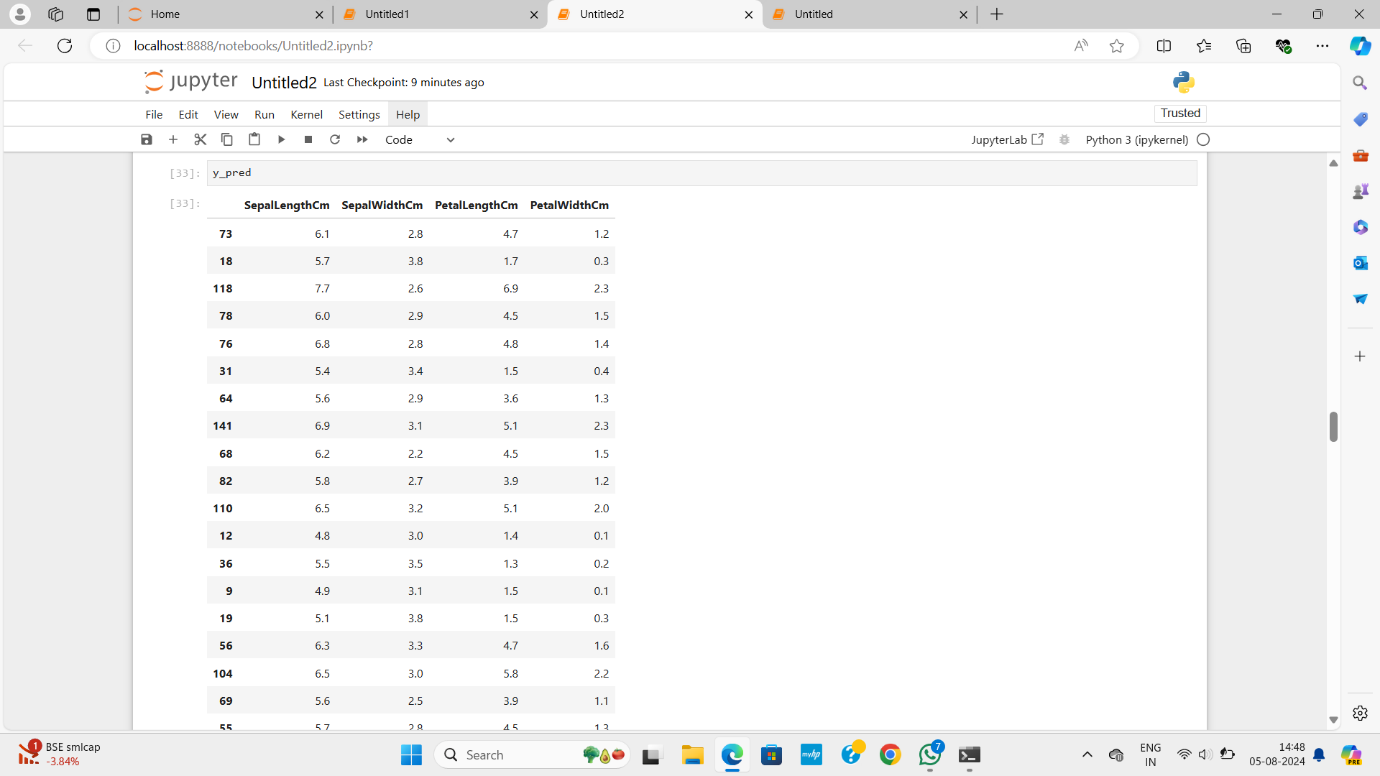
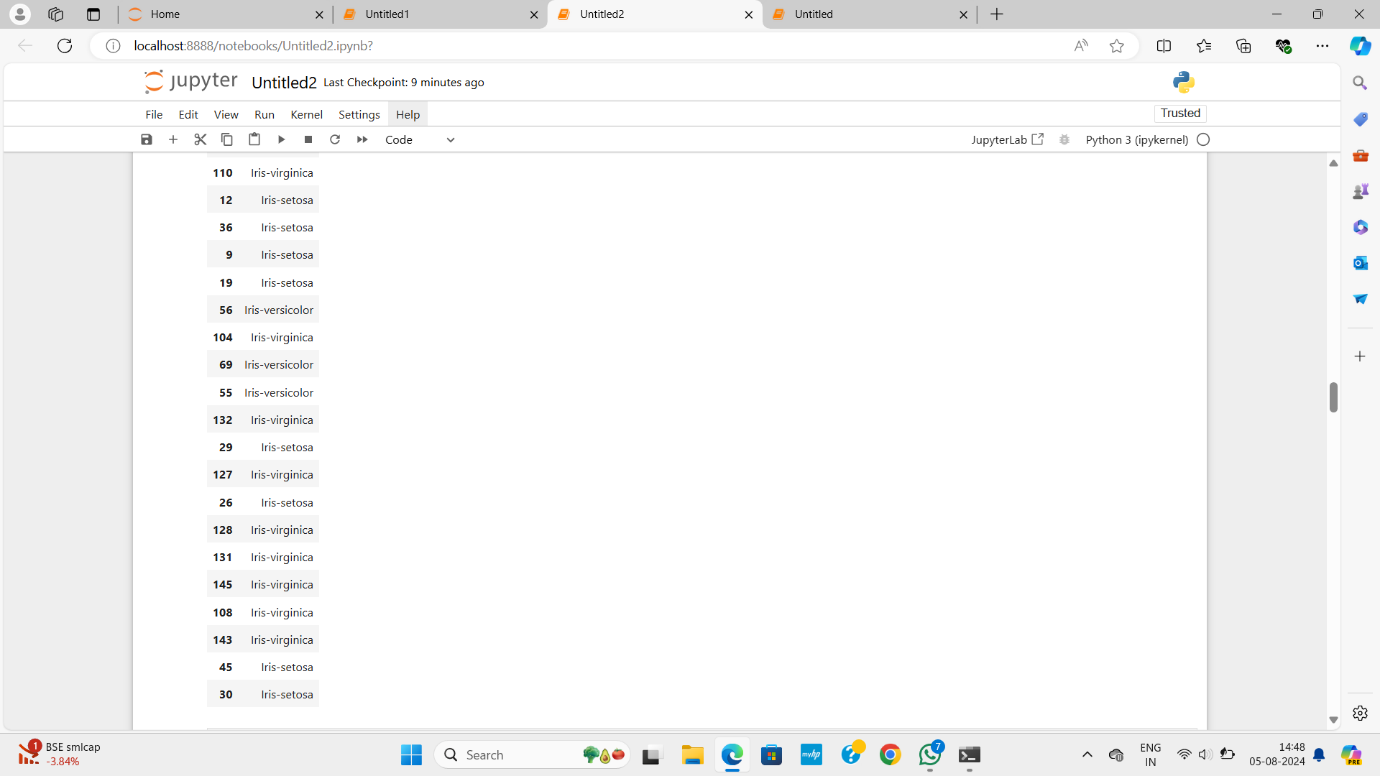
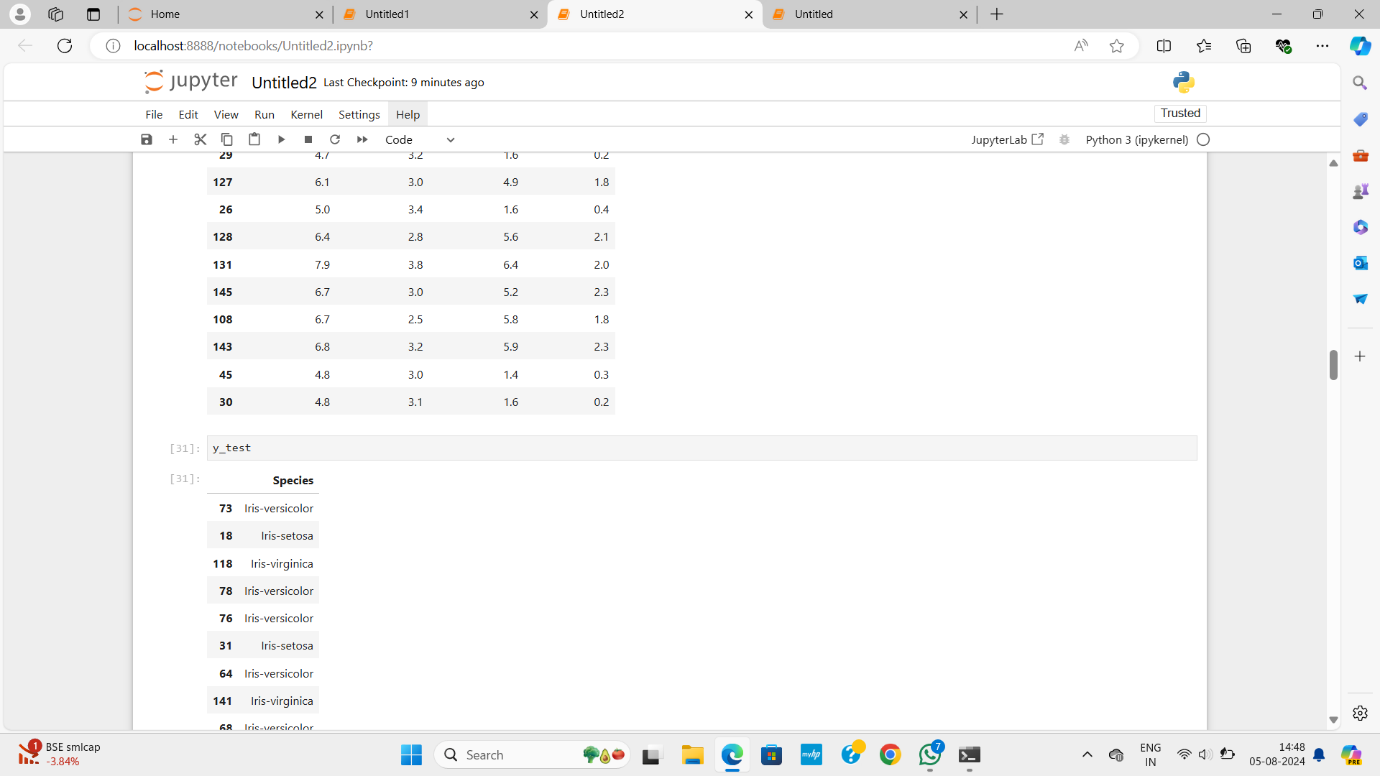
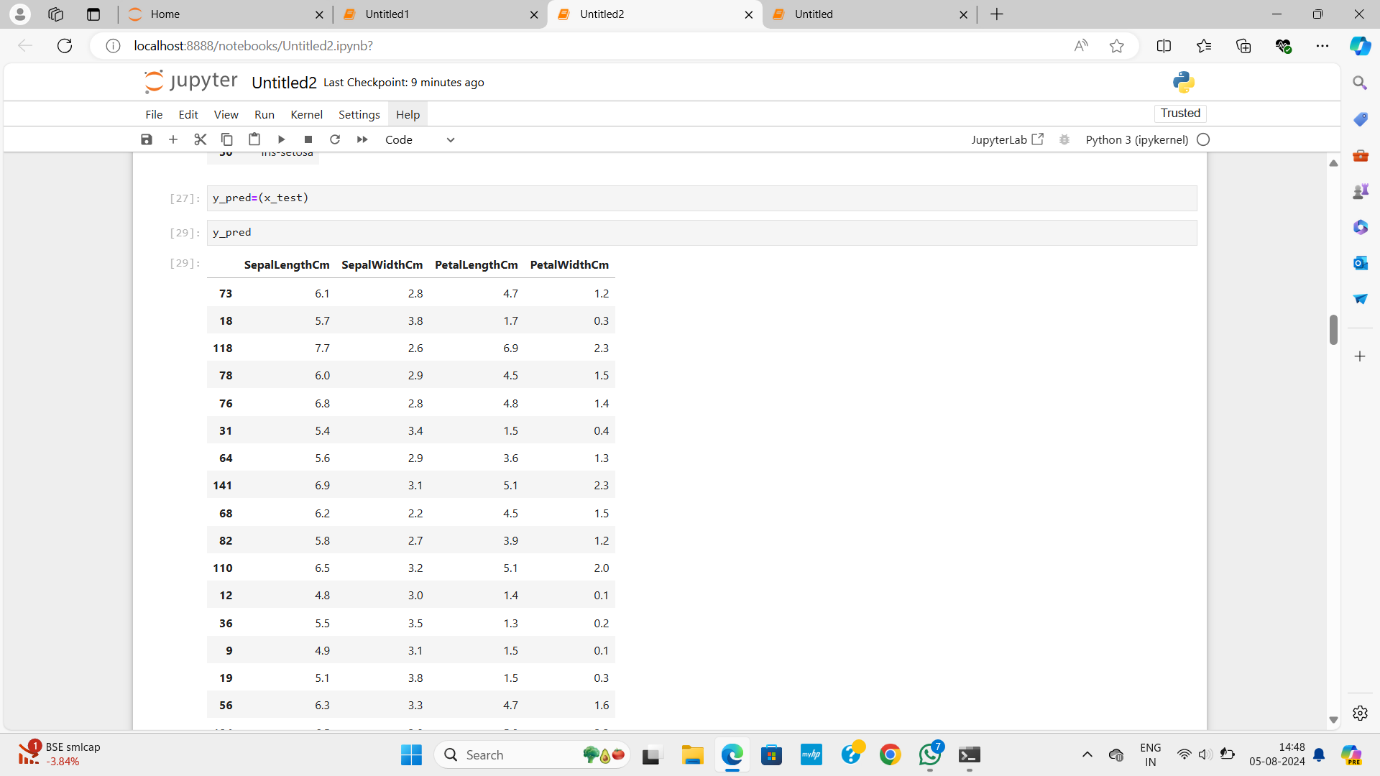
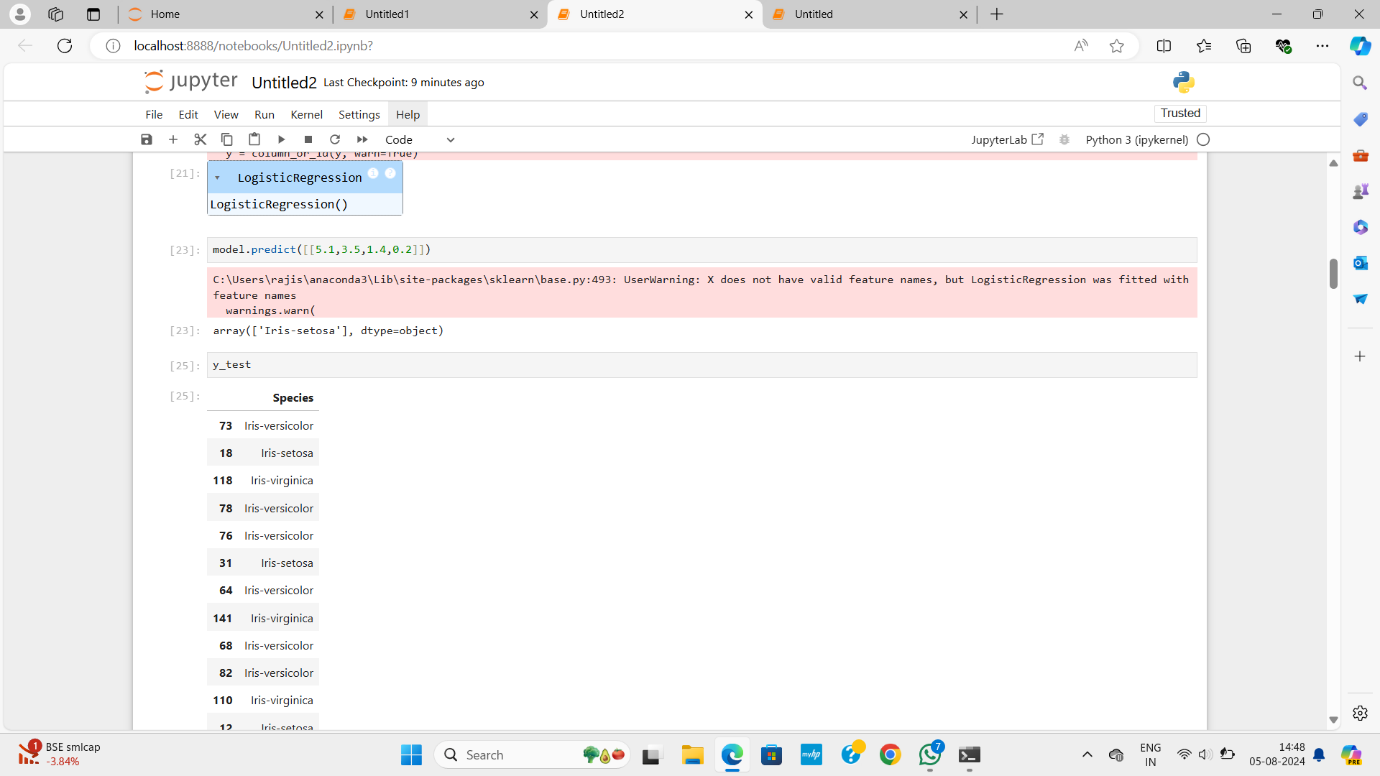
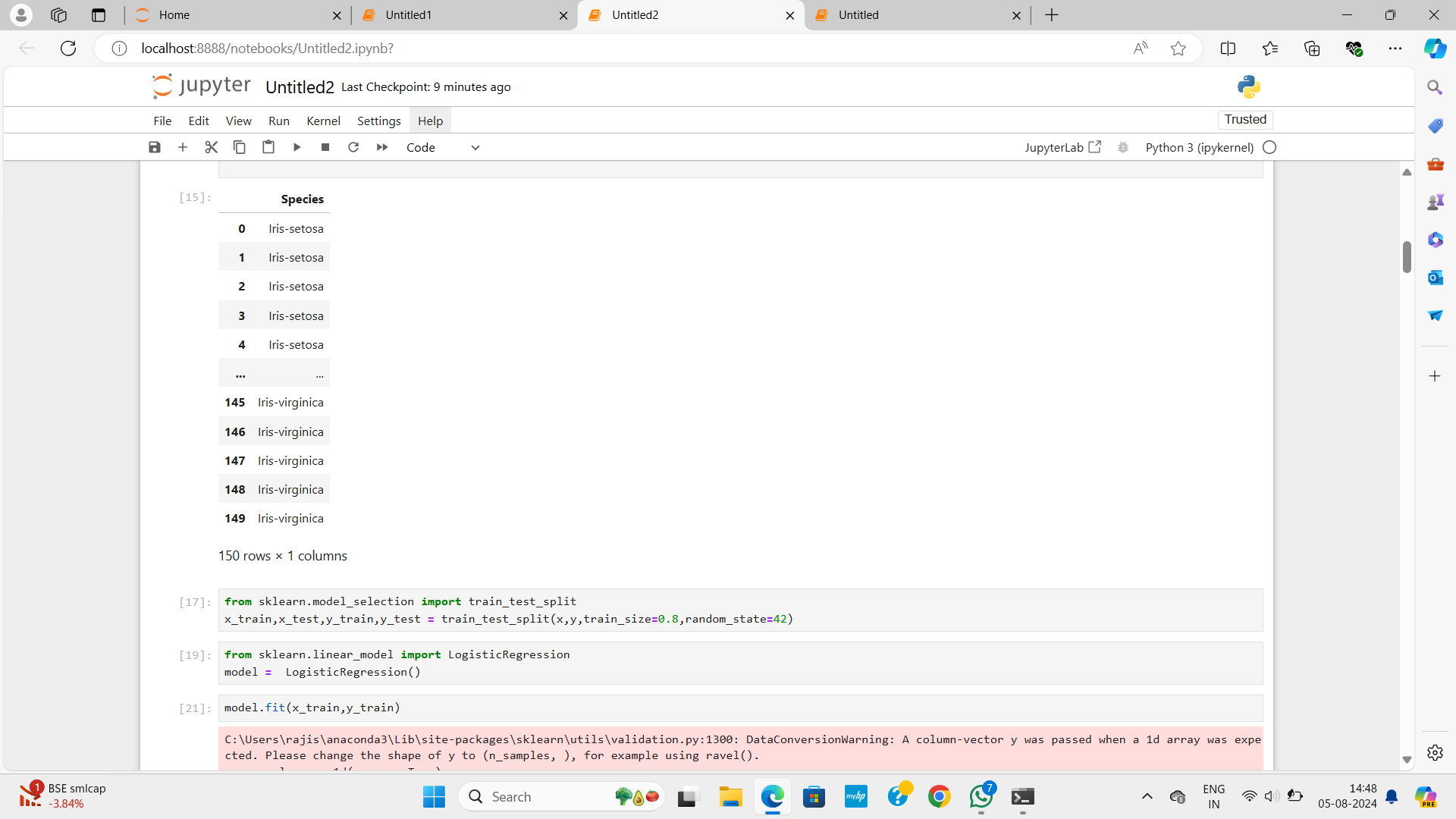
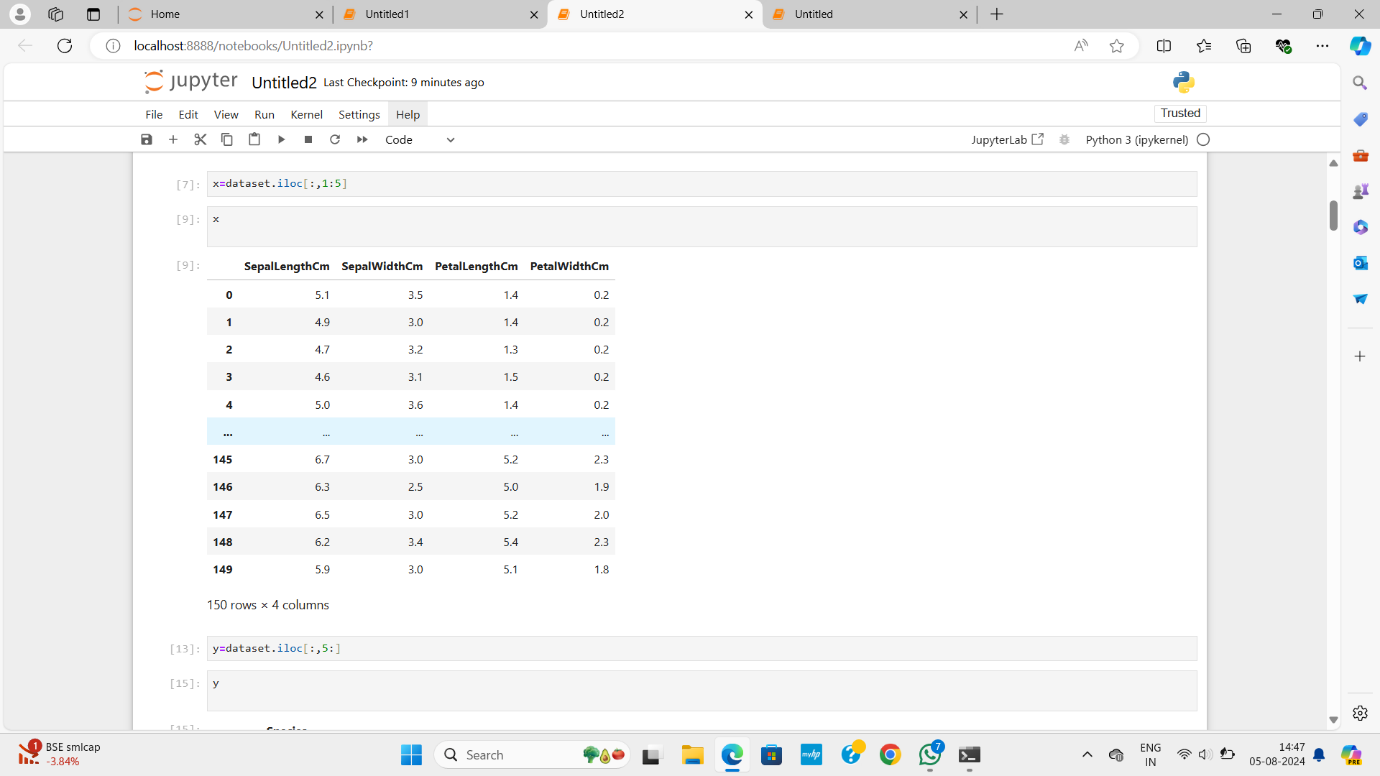
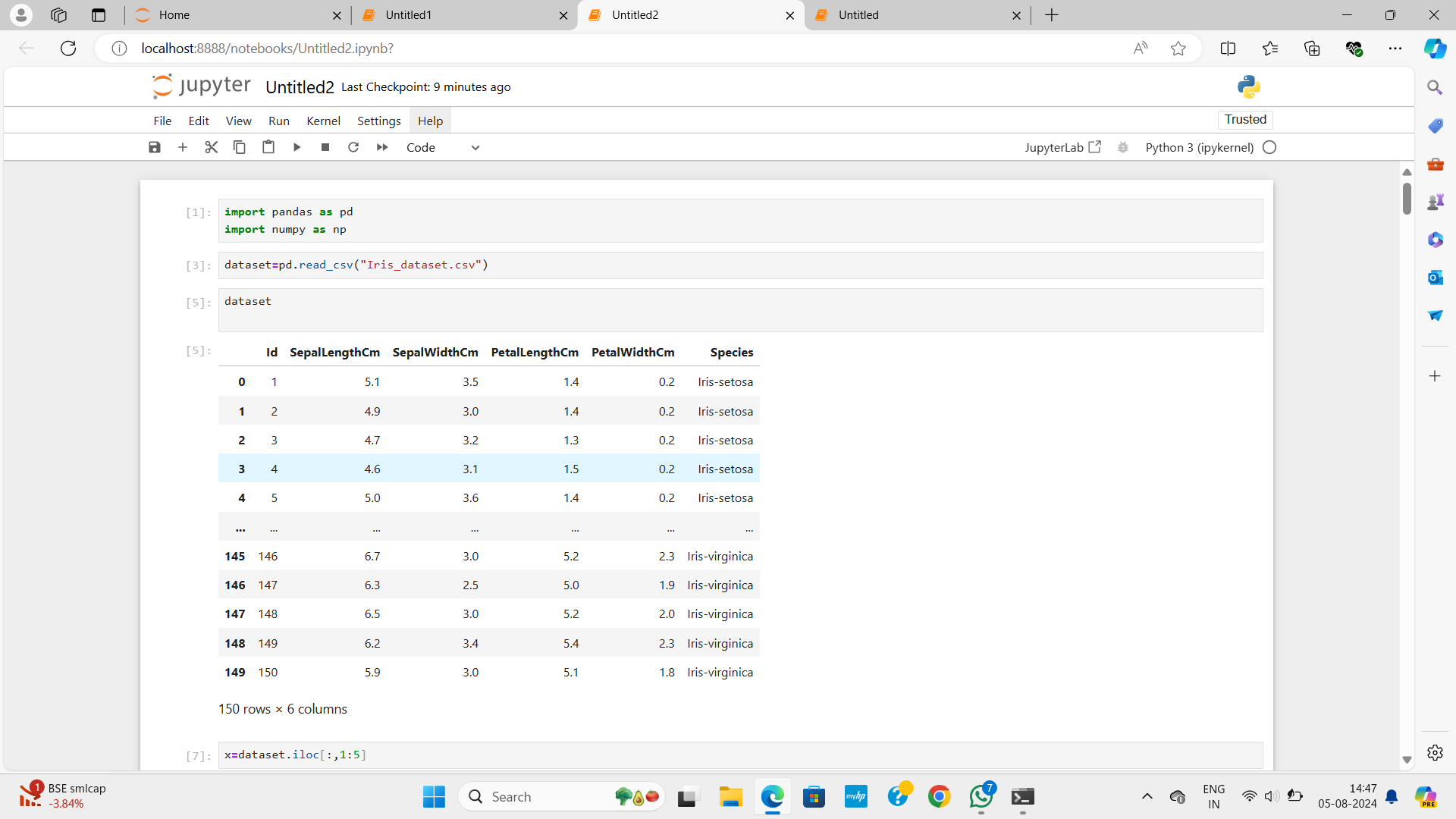
plt.xlabel("Predicted Class")

plt.ylabel("Actual Class")

plt.colorbar()

plt.show()

**Screenshots**



**PENGUINS\_BINARY\_CLASSIFICATION:**

**Code :**

    1.import numpy as np

import pandas as pd

2. dataset = pd.read\_csv("penguins\_binary\_classification.csv") # first we need upload the dataset

dataset.head() # will shoe first 5 rows

3. x = dataset.iloc[:,0:1]

y = dataset.iloc[:,1]

x.head()

y.head()

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

x\_train,x\_test,y\_train,y\_test =

train\_test\_split(x,y,train\_size=0.8,random\_state=42)

5. from sklearn.linear\_model import LinearRegression

model = LinearRegression()

6.  model.fit(x\_train,y\_train)

7. samplepredciton = model.predict([[1240]])

print(samplepredciton)

y\_pred = model.predict(x\_test) # predicting for all x\_test

y\_pred

y\_test

y\_pred

list(zip(y\_test,y\_pred))

8. import numpy as np # Hypothetical housing dataset

areas = np.array([1000, 1030, 1060, 1090])  # House areas in square feet

prices = np.array([5618, 5201, 4779, 5245])  # House prices in dollars

# Example of a simple linear regression model (hypothetical coefficients)

m = 200  # slope

b = 50000  # intercept

# Make predictions

predicted\_prices = m \* areas + b

# Calculate squared differences

squared\_diff = (prices - predicted\_prices) \*\* 2

# Calculate mean squared error

mse = np.mean(squared\_diff)

print("Actual Prices:", prices)

print("Predicted Prices:", predicted\_prices)

print("Squared Differences:", squared\_diff)

print("Mean Squared Error:", mse)

9.from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

print(f'R-squared score: {r2:.2f}')

y\_test

**Screenshots**

