|  |  |
| --- | --- |
| **Assignment #** | **4** |

**By**

**NAME: Ali Raza Asim**

**ROLL NO #:1220100037**

**BS.SE (4th)**

**Fall 2024**

**COURSE TITLE: Artificial Intelligence**

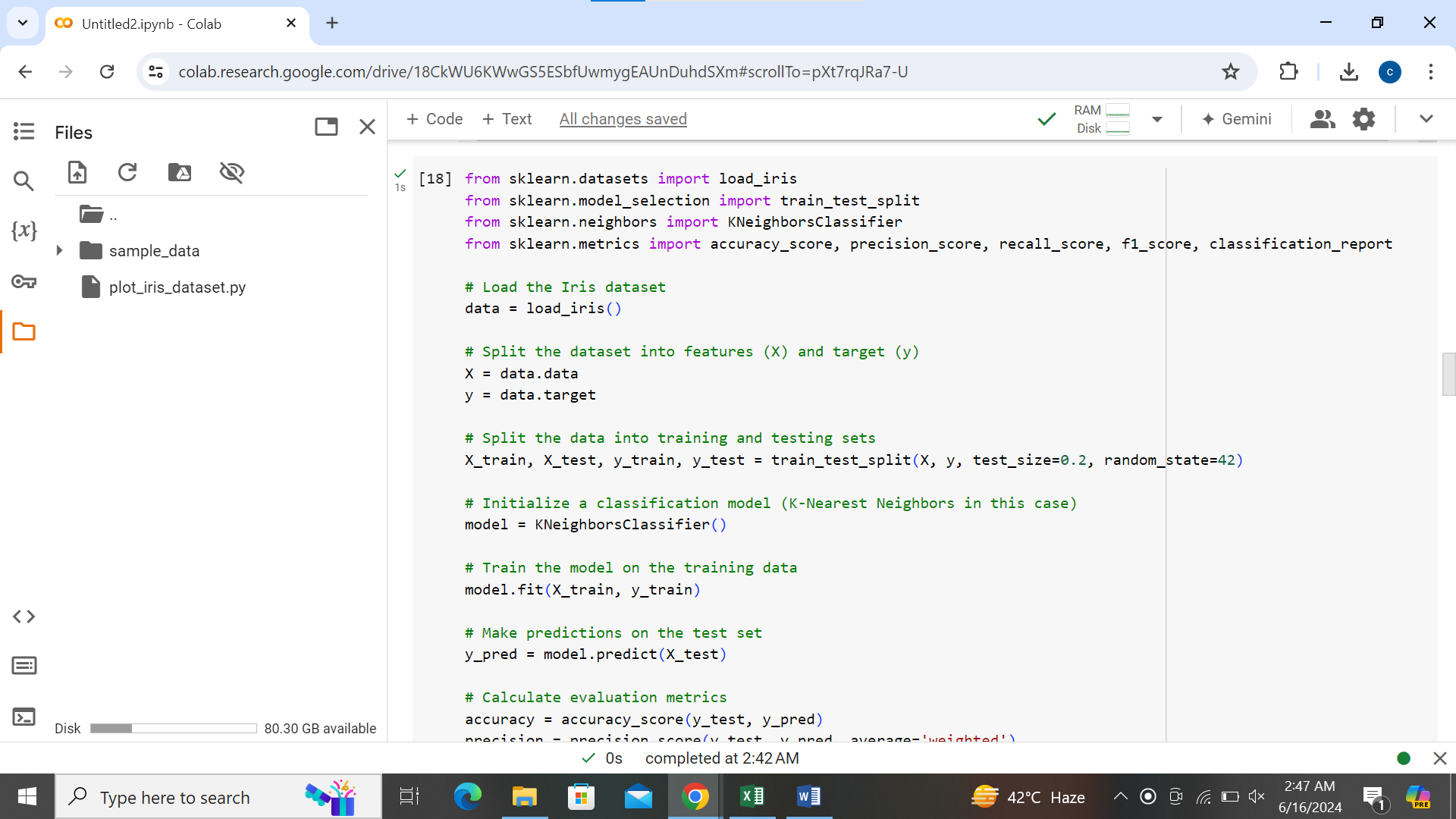
**SUBMITTED TO : Mr. Zubair**

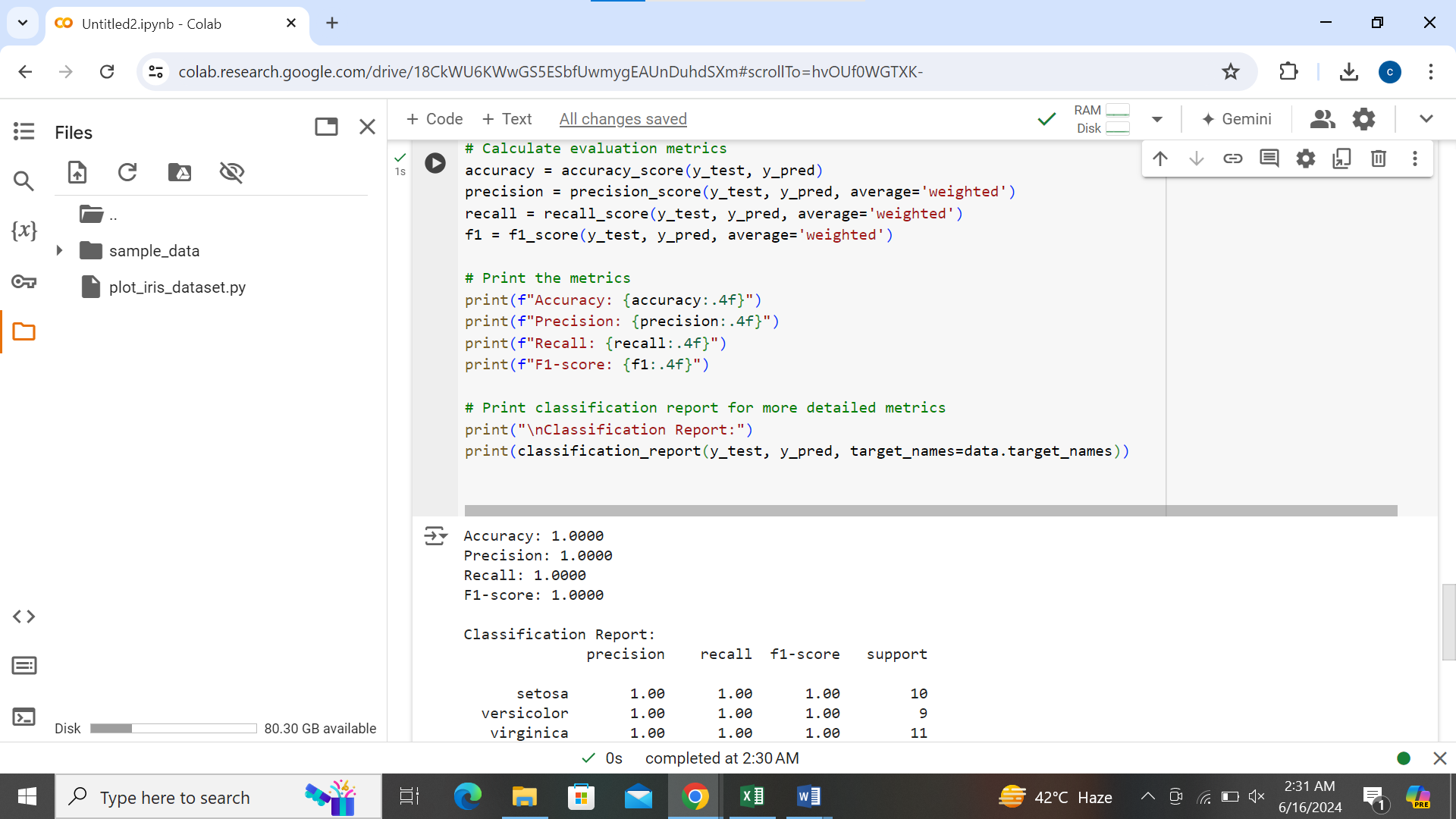
****

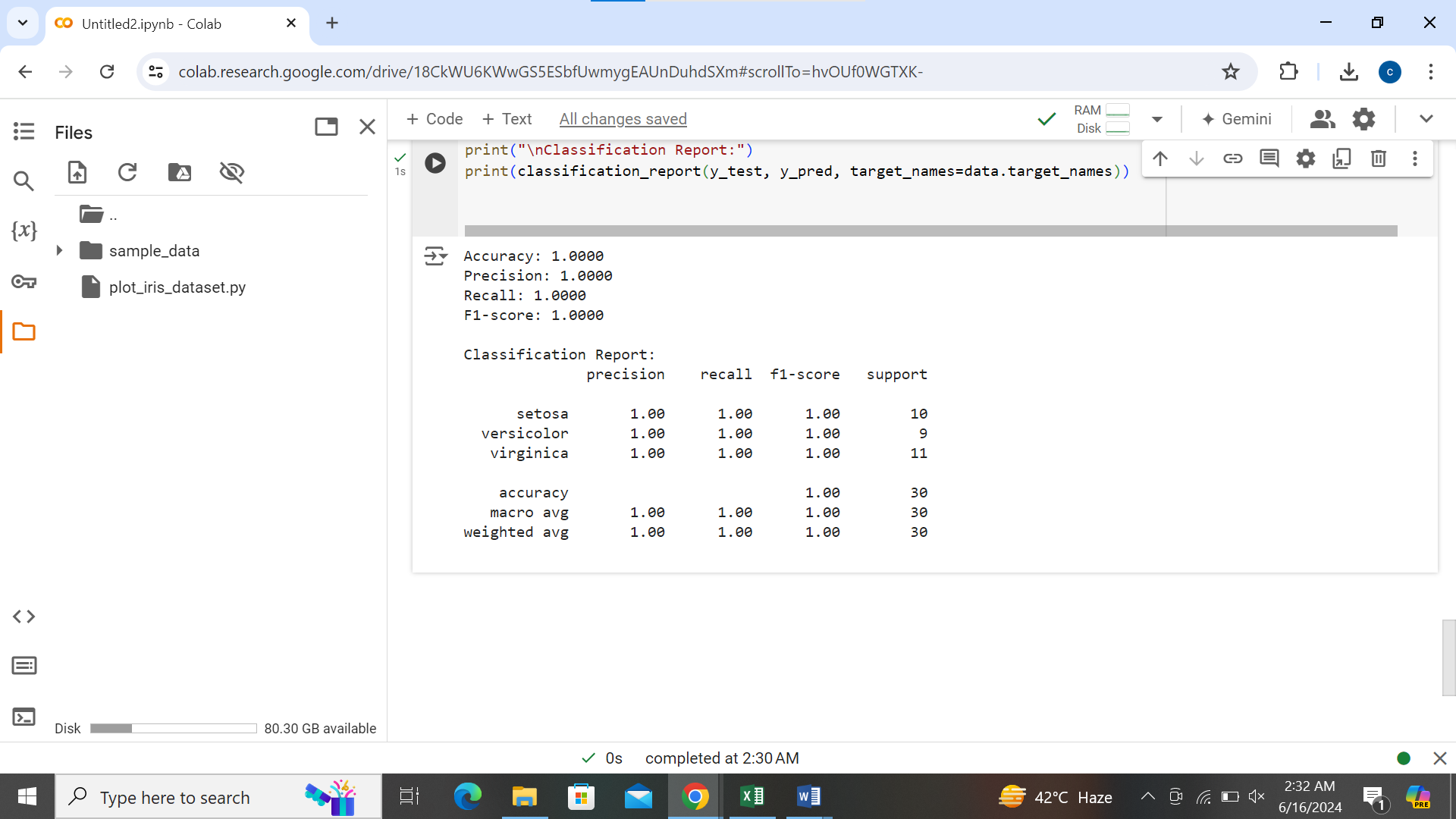
**Department of COMPUTER SCIENCE**

**International Institute of Science, Arts and Technology (IISAT), Gujranwala**

**QUESTION NO 1:**

****

****

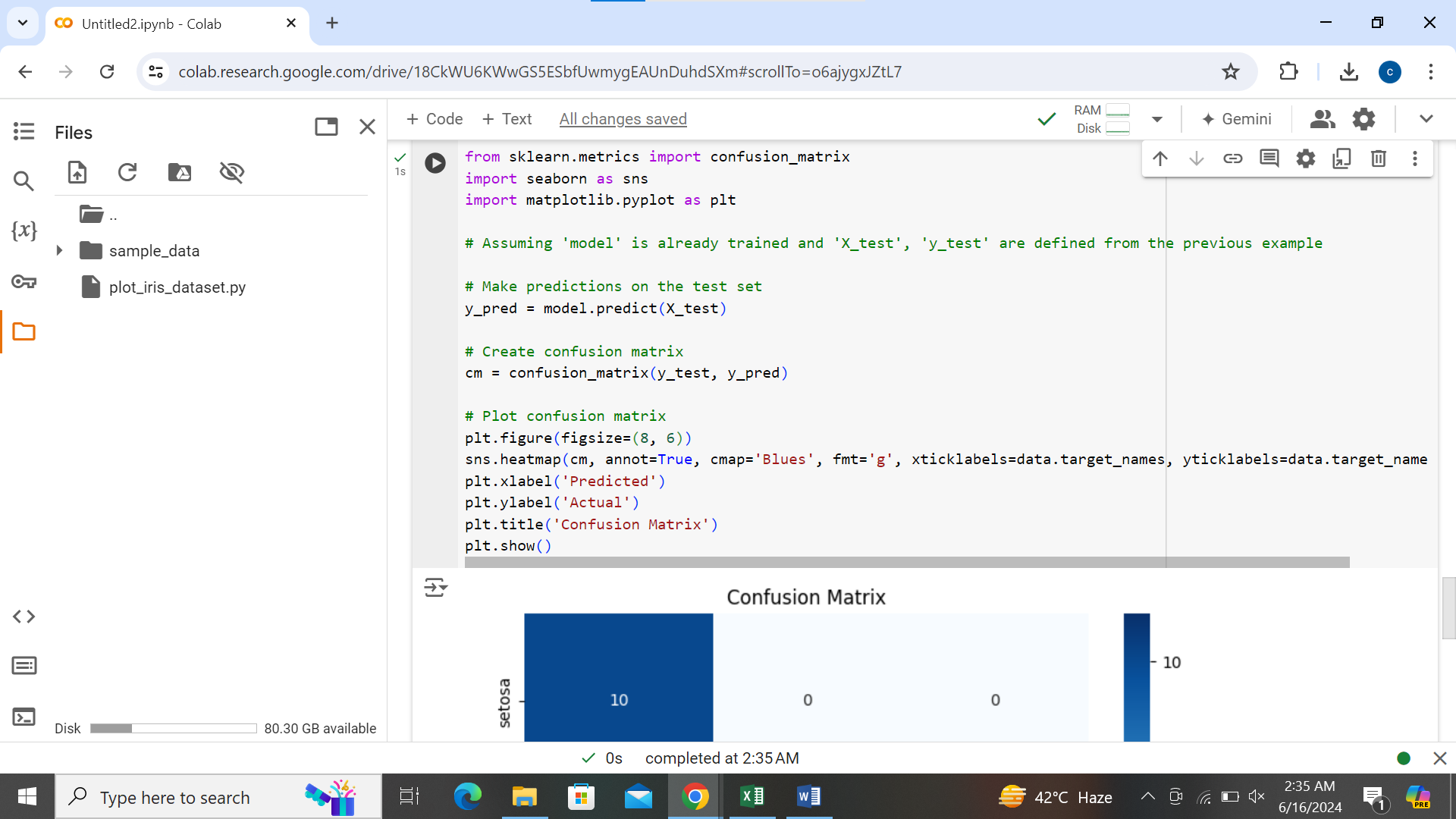
****

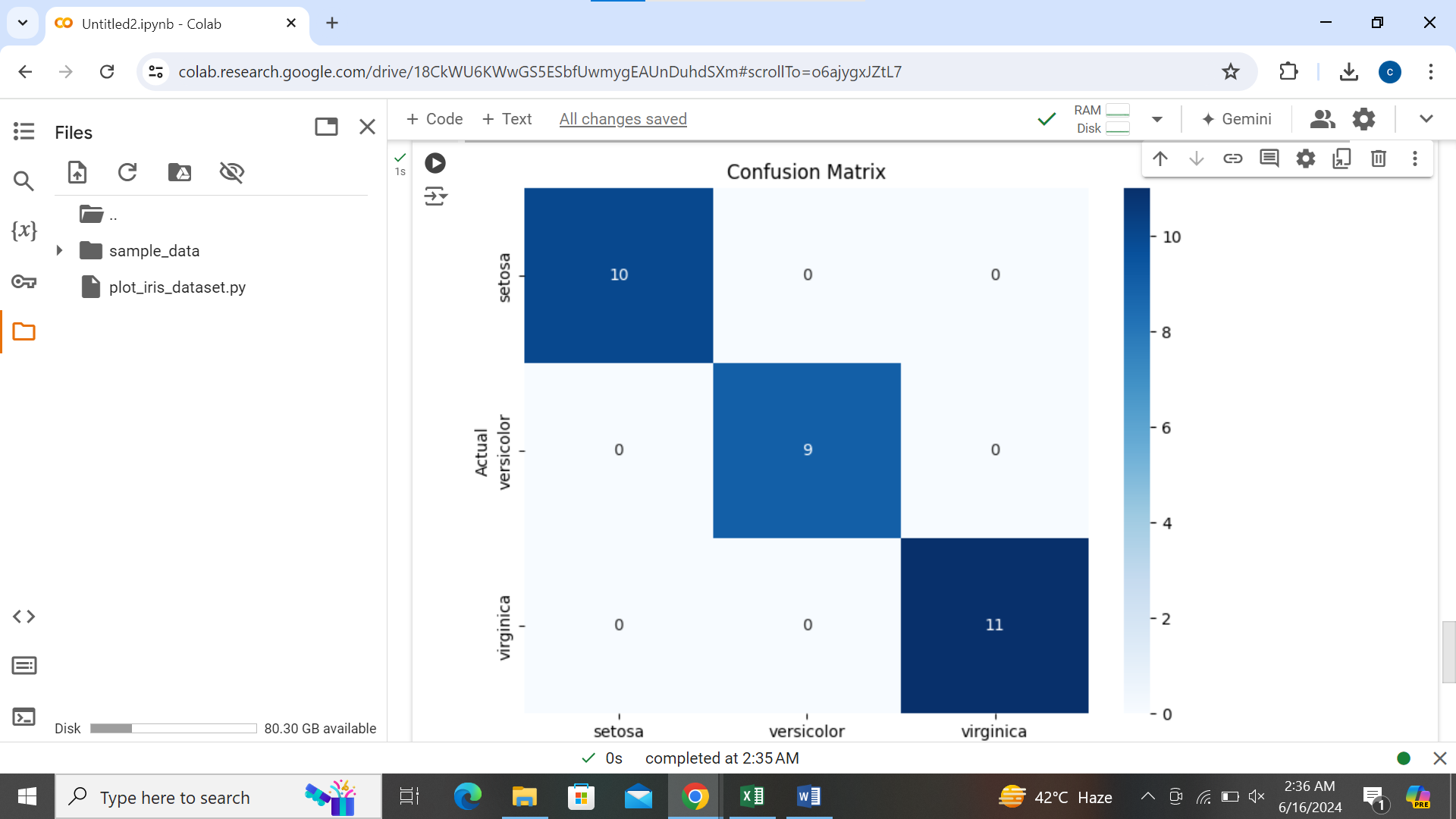
**Explanation:**

1. **Loading the Dataset**: We use load\_iris() from scikit-learn to load the Iris dataset. This dataset contains features related to iris flowers and a target variable indicating the species.
2. **Splitting the Data**: The dataset is split into training (X\_train, y\_train) and testing (X\_test, y\_test) sets using train\_test\_split() function. Here, 80% of the data is used for training and 20% for testing.
3. **Training the Model**: We initialize a KNeighborsClassifier model, which is a simple classification algorithm based on neighbors' voting.
4. **Making Predictions**: Predictions are made on the test set (X\_test) using predict().
5. **Calculating Metrics**: Using scikit-learn metrics (accuracy\_score, precision\_score, recall\_score, f1\_score), we compute the accuracy, precision, recall, and F1-score based on the predictions (y\_pred) compared to the actual labels (y\_test). For multi-class classification like Iris dataset, we use average='weighted' for precision, recall, and F1-score to account for class imbalance.
6. **Interpreting Metrics**: These metrics provide insights into how well the K-Nearest Neighbors model performs in classifying iris flowers into their respective species. Higher values indicate better performance for each metric.

**Conclusion:**

You can use the Iris dataset or any other classification dataset available in scikit-learn or other sources to practice training models and evaluating their performance using metrics like accuracy, precision, recall, and F1-score. Adjust the model choice and dataset based on your specific learning or project requirements.**QUESTION NO 2:**





**Explanation of Confusion Matrix:**

Let's interpret what each part of the confusion matrix represents using the Iris dataset as an example:

* **True Positives (TP)**: Diagonal elements from top-left to bottom-right (e.g., cm[0,0], cm[1,1], cm[2,2]). These represent the number of correctly predicted instances for each class. For instance, cm[0,0] represents the number of correctly predicted instances of class 0 (setosa), cm[1,1] for class 1 (versicolor), and cm[2,2] for class 2 (virginica).
* **False Positives (FP)**: Columns where the actual class is different from the predicted class (e.g., cm[0,1], cm[0,2] for class 0). These indicate the number of instances wrongly predicted as belonging to that class.
* **False Negatives (FN)**: Rows where the predicted class is different from the actual class (e.g., cm[1,0], cm[2,0] for class 0). These represent instances that were wrongly predicted as not belonging to that class.
* **True Negatives (TN)**: Off-diagonal elements not in the same row or column as true positives for that class. In binary classification, it represents correctly predicted instances of the negative class, but in multi-class, it is less commonly used and not always defined in confusion matrix.

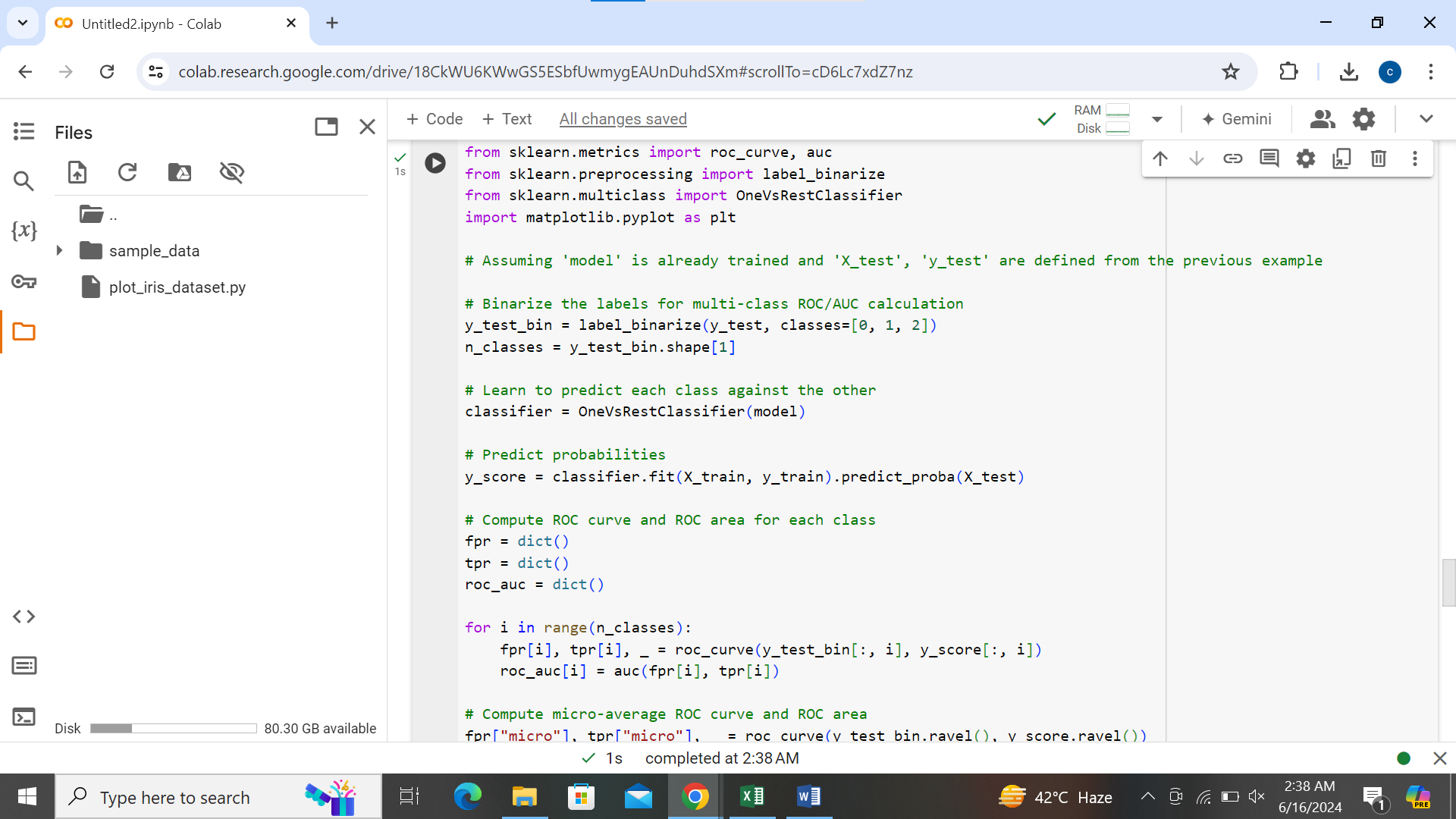
**Importance of Confusion Matrix:**

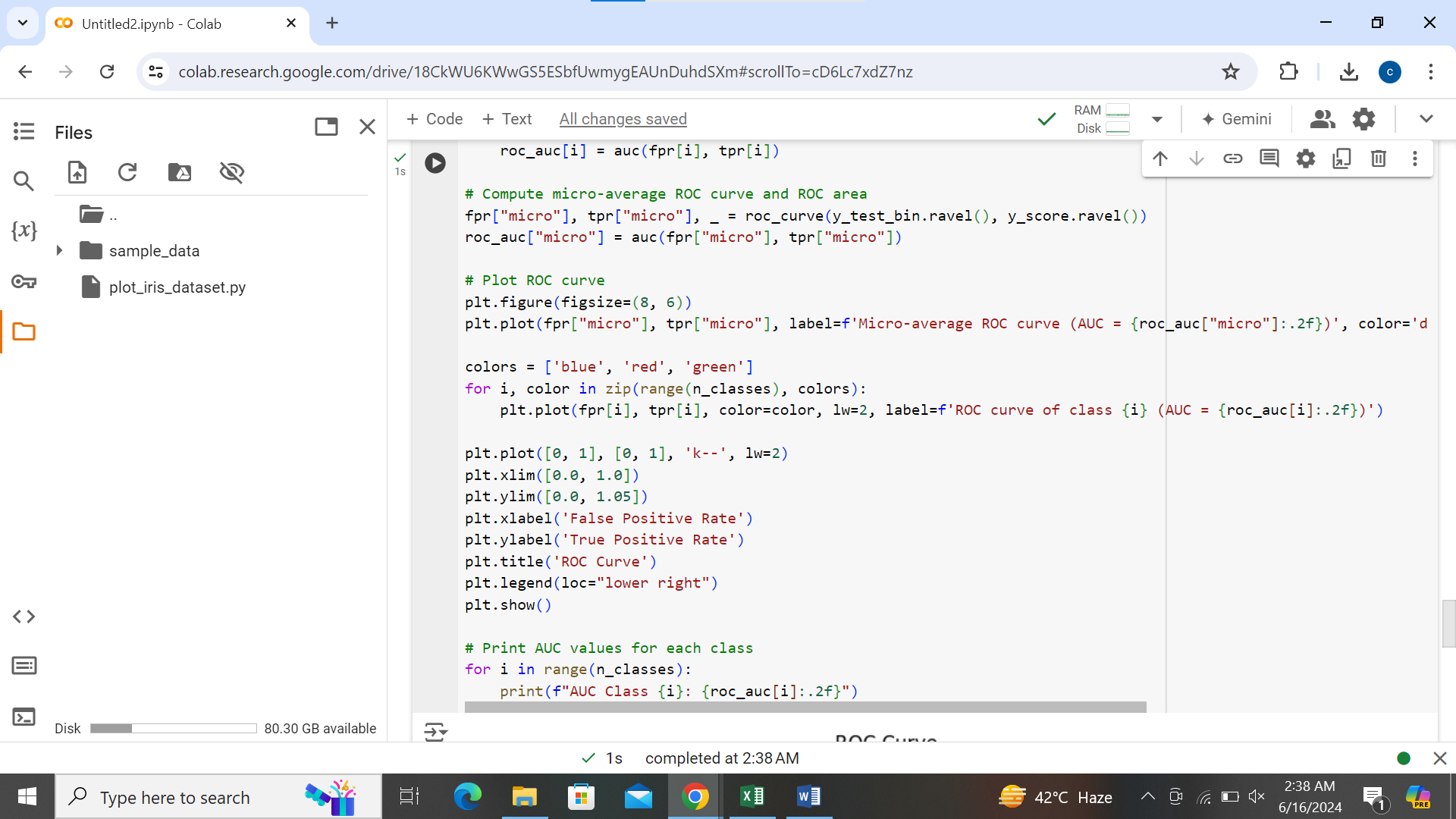
* **Understanding Errors**: Helps in understanding where the model is making mistakes (e.g., confusing one class for another) and how these errors are distributed across different classes.
* **Performance Metrics**: From the confusion matrix, you can derive metrics such as accuracy, precision, recall, and F1-score. For example:
  + **Accuracy**: Overall correctness of predictions.
  + **Precision**: Proportion of true positive predictions out of all positive predictions.
  + **Recall**: Proportion of true positives correctly identified.
  + **F1-score**: Harmonic mean of precision and recall.
* **Model Improvement**: Allows for targeted improvements such as focusing on reducing false positives or false negatives based on the application's requirements.

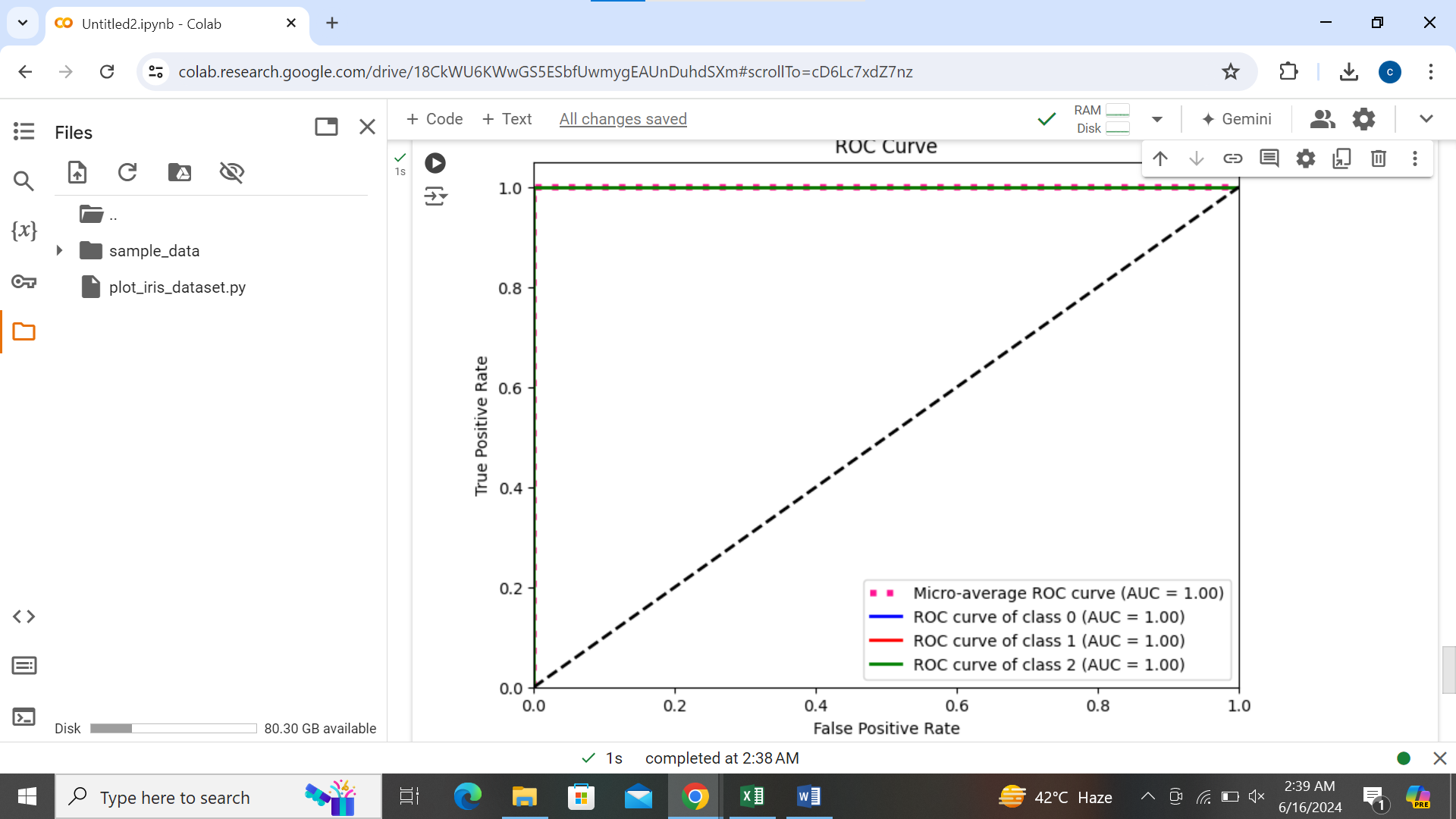
**Conclusion:**

The confusion matrix is a powerful tool for assessing the performance of a classification model by providing a detailed breakdown of prediction outcomes across different classes. It not only quantifies the model's accuracy but also helps in diagnosing specific areas of improvement for the model, leading to more effective machine learning solutions.

**QUESTION NO 3:**







**Explanation:**

1. **Binarizing Labels**: In order to calculate ROC curve and AUC for multi-class classification, we binarize the labels (y\_test\_bin) using label\_binarize to transform each class into a binary format.
2. **One-vs-Rest Strategy**: We use OneVsRestClassifier to compute the ROC curve and AUC for each class against the rest.
3. **ROC Curve Calculation**: For each class, roc\_curve computes the false positive rate (fpr) and true positive rate (tpr). auc calculates the area under each ROC curve.
4. **Micro-average ROC Curve**: The micro-average aggregates the contributions of all classes to compute the average ROC curve.
5. **Plotting ROC Curve**: We plot the micro-average ROC curve along with individual ROC curves for each class, showing the AUC value in the legend for each curve.
6. **Interpreting ROC Curve and AUC**:
   * **ROC Curve**: Represents the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) across different thresholds.
   * **AUC (Area Under the Curve)**: Provides a single number that quantifies the overall performance of the classifier. A higher AUC indicates better discrimination ability of the model across all possible thresholds.

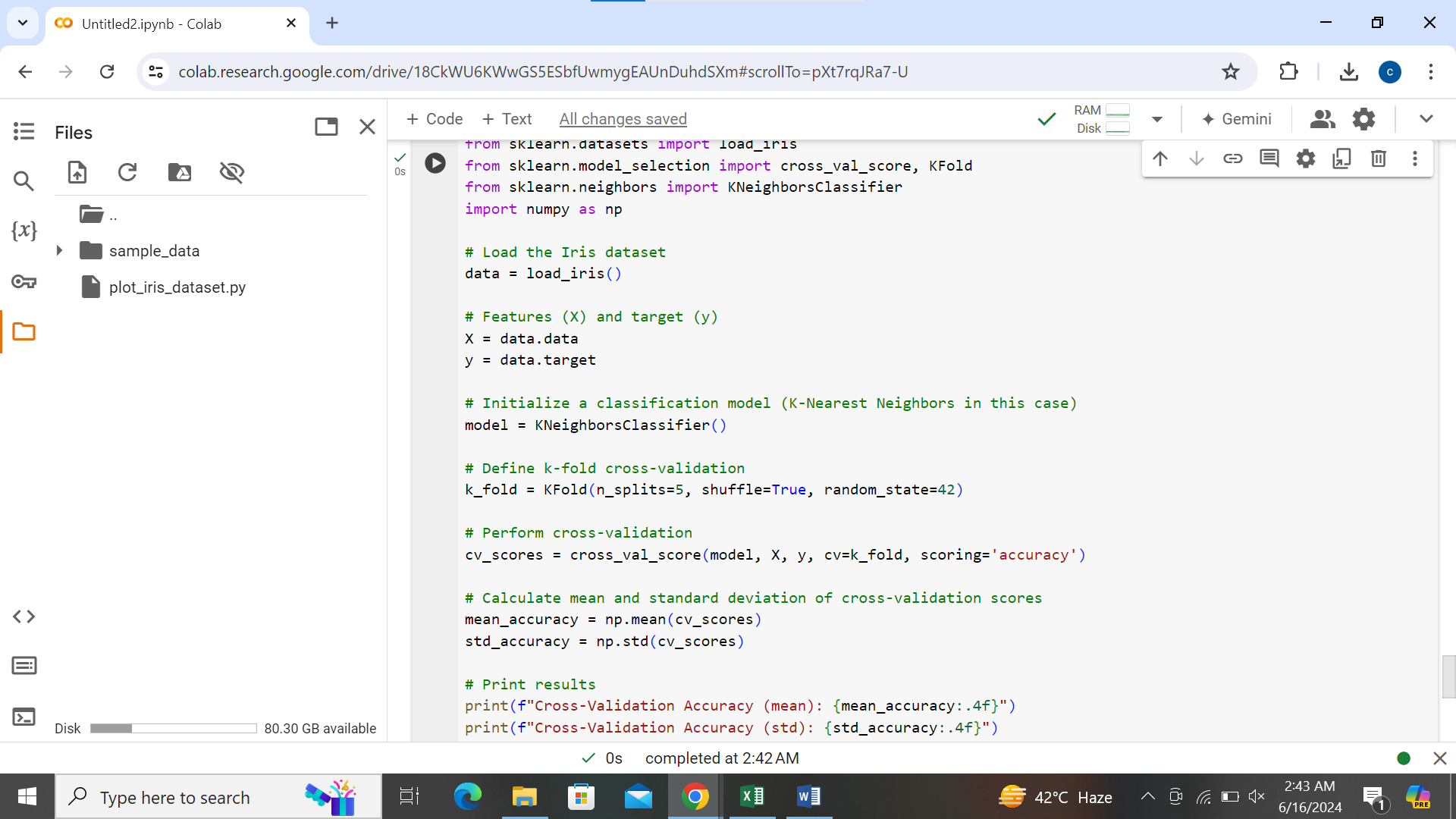
**How These Metrics Help in Evaluating Model Performance:**

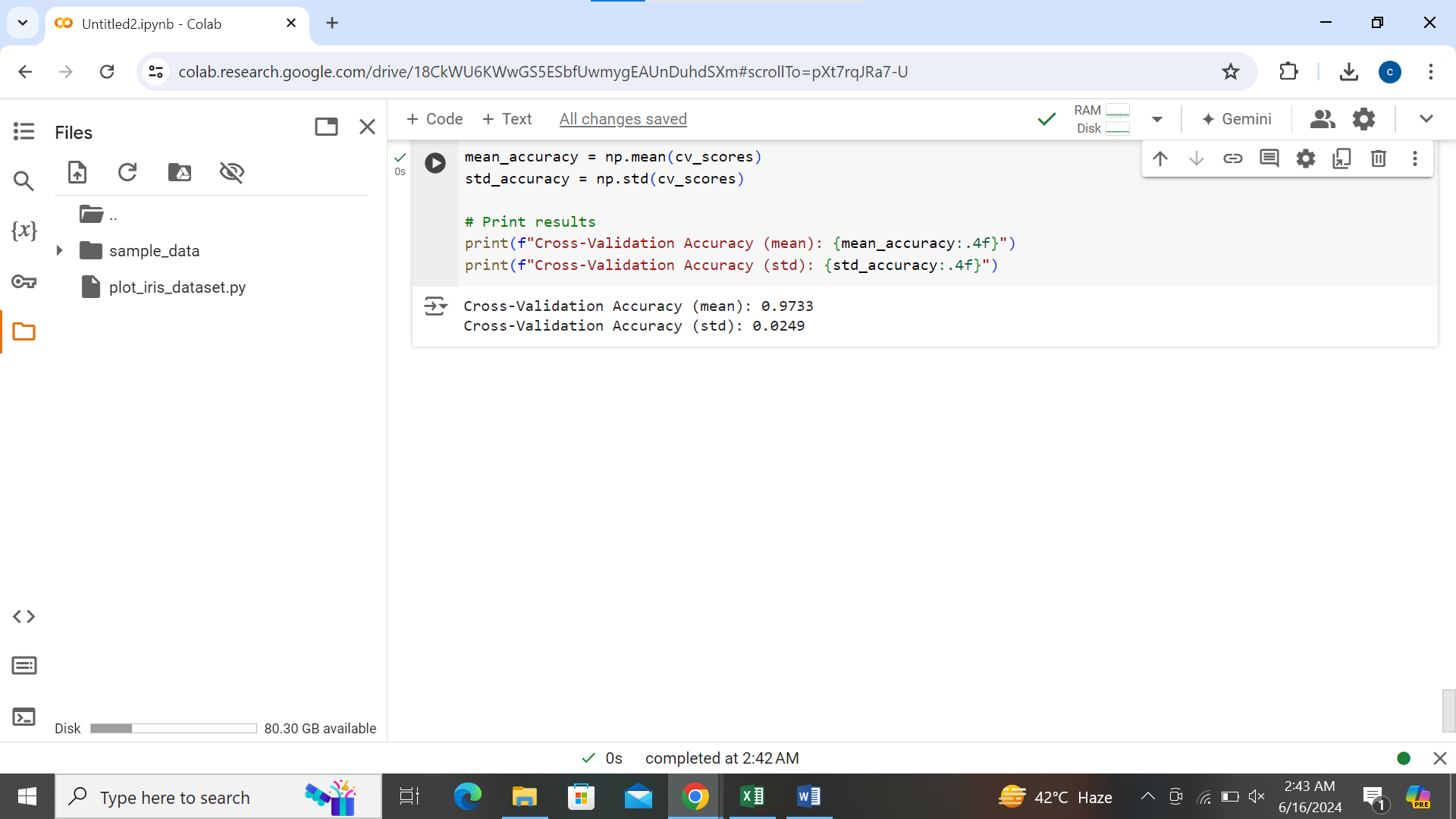
* **ROC Curve**: Helps visualize the performance of the classifier at various thresholds. A curve that is closer to the top-left corner indicates better performance.
* **AUC**: Provides a single scalar value (between 0 and 1) representing the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Higher AUC values indicate better model performance.
* **Evaluation Across Classes**: For multi-class problems, ROC curves and AUC values provide insights into how well the model discriminates between classes. It helps in understanding which classes are easier or harder to distinguish.

**Conclusion:**

The ROC curve and AUC are powerful tools for evaluating the performance of classification models, especially in scenarios where you need to compare the trade-offs between sensitivity and specificity or when dealing with multi-class classification tasks. These metrics provide a comprehensive view of the model's discriminatory ability and help in making informed decisions about model selection and tuning.

**QUESTION NO 4:**





**Explanation:**

1. **Loading the Dataset**: We load the Iris dataset using load\_iris() from scikit-learn.
2. **Features and Target**: X contains the features (sepal length, sepal width, petal length, petal width), and y contains the target variable (0 for setosa, 1 for versicolor, 2 for virginica).
3. **Initialization**: We initialize a KNeighborsClassifier model for classification. Replace this with any other classifier of your choice.
4. **k-fold Cross-Validation**: KFold is used to define 5-fold cross-validation (n\_splits=5). shuffle=True shuffles the data before splitting, and random\_state=42 sets the random seed for reproducibility.
5. **Cross-validation Score Calculation**: cross\_val\_score computes the accuracy of the model for each fold (cv=k\_fold) using the scoring='accuracy' parameter.
6. **Mean and Standard Deviation**: np.mean and np.std calculate the mean and standard deviation of the cross-validation scores, respectively.

**Conclusion:**

In this example, the Iris dataset is used to demonstrate k-fold cross-validation for evaluating a classification model's performance. The mean accuracy and standard deviation of the cross-validation scores provide insights into how well the model is expected to generalize to new, unseen data. Cross-validation is crucial for obtaining reliable estimates of model performance and guiding decisions in model selection and tuning across various datasets and algorithms.

Top of Form

Bottom of Form