**AI-BASED DIABETES PREDICTION USING MACHINE LEARNING ALGORITHM.**

**TEAM MEMBER: KRISHNAKANTH .S**

**REG.NO: 211521106088**

**NM ID: au211521106088**

**PHASE 4: Development part 2**

**TOPIC:** *continue building the diabetes prediction model by*

*Ml algorithm ,model training and evalution*

**INTRODUCTION:**

Diabetes is a chronic disease that affects how your body turns food into energy. With diabetes, your body either resists the effects of insulin—a hormone that regulates the movement of sugar into your cells—or doesn't produce enough insulin to maintain normal glucose levels.

Early detection and treatment of diabetes can help prevent serious health complications, such as heart disease, stroke, blindness, and kidney disease. Machine learning algorithms can be used to predict diabetes risk based on a variety of factors, including age, gender, family history, medical history, and lifestyle habits.

Selecting a machine learning algorithm

There are many different machine learning algorithms that can be used to predict diabetes. Some of the most common algorithms include:

* Logistic regression: A simple but effective algorithm for predicting binary outcomes, such as whether or not someone has diabetes.
* Support vector machines (SVM): A more complex algorithm that can be used to predict both binary and continuous outcomes.
* Random forests: An ensemble learning algorithm that combines the predictions of multiple decision trees to produce a more accurate prediction.
* Deep neural networks: A type of artificial intelligence that can learn complex patterns in data.

The best machine learning algorithm to use for diabetes prediction will depend on the specific dataset being used and the desired performance metrics.

Training the model

Once a machine learning algorithm has been selected, it needs to be trained on a dataset of labeled examples. Labeled examples are data points where the target variable (in this case, whether or not someone has diabetes) is known.

The training process involves feeding the algorithm the labeled examples and allowing it to learn the relationship between the input features (e.g., age, gender, medical history) and the target variable.

Evaluating the performance of the model

Once the model has been trained, it needs to be evaluated on a held-out test set. The test set is a dataset of labeled examples that was not used to train the model.

Evaluating the model on the test set provides an estimate of how well the model will perform on new data. Common performance metrics for diabetes prediction include accuracy, precision, recall, and F1 score.

Conclusion

Machine learning algorithms can be used to predict diabetes risk with high accuracy. By predicting diabetes risk, clinicians can identify people who are at high risk and provide them with early intervention and preventive care.

Here are some additional tips for selecting, training, and evaluating a machine learning model for diabetes prediction:

* Use a variety of features. The more features you use to train your model, the more accurate it will be. However, it is important to select features that are relevant to diabetes prediction and to avoid overfitting the model to the training data.
* Use cross-validation. Cross-validation is a technique that allows you to evaluate the performance of your model on multiple datasets. This is important to avoid overfitting the model to the training data.
* Tune the hyperparameters. Hyperparameters are parameters that control the behavior of the machine learning algorithm. Tuning the hyperparameters can improve the performance of the model.
* Use a variety of evaluation metrics. Accuracy is a common evaluation metric, but it is important to also consider other metrics, such as precision, recall, and F1 score. This is because accuracy can be misleading in some cases, such as when the dataset is imbalanced.

Once you have selected, trained, and evaluated your model, you can use it to predict diabetes risk in new patients. This information can be used to guide clinical decision-making and to provide patients with personalized recommendations for prevention and care.

**OVERVIEW OF THE PROCESS:**

Overview of the Process

To select a machine learning algorithm for diabetes prediction, you need to consider the following factors:

* Type of data: Is your data structured or unstructured? What are the features of your data?
* Complexity of the problem: How complex is the relationship between the features of your data and the target variable (i.e., whether or not a patient has diabetes)?
* Interpretability: How important is it to be able to interpret how the model makes its predictions?
* Computational resources: How much computational power do you have available?

Once you have considered these factors, you can start to narrow down your choices of machine learning algorithms. Some popular algorithms for diabetes prediction include:

* Logistic regression: A simple but effective algorithm for binary classification tasks.
* Support vector machines (SVMs): A powerful algorithm that can learn complex relationships between features and the target variable.
* Random forests: An ensemble learning algorithm that builds multiple decision trees and averages their predictions to produce a final prediction.
* Gradient boosting machines (GBMs): Another ensemble learning algorithm that builds sequential models to improve the performance of the previous model.
* Deep neural networks (DNNs): A type of machine learning model that can learn complex patterns from data.

Once you have selected a machine learning algorithm, you need to train the model on your data. This involves feeding the model your data and allowing it to learn the relationships between the features and the target variable.

Once the model is trained, you need to evaluate its performance on a held-out test set. This will give you an idea of how well the model will generalize to new data. If the model performs well on the test set, you can deploy it to production.

**Step-by-Step Procedure:**

To train a machine learning model for diabetes prediction, you can follow these steps:

1. Prepare your data. This includes cleaning the data, handling missing values, and converting categorical variables to numerical variables.
2. Split your data into training and test sets. The training set will be used to train the model, and the test set will be used to evaluate the model's performance.
3. Choose a machine learning algorithm. Consider the factors mentioned above when choosing an algorithm.
4. Train the model. Feed the training data to the model and allow it to learn the relationships between the features and the target variable.
5. Evaluate the model. Evaluate the model's performance on the test set.
6. Deploy the model. If the model performs well on the test set, you can deploy it to production.

Here is an example of how to train a random forest model for diabetes prediction using the Python programming language:

Python

import numpy as np

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

# Load the data

data = pd.read\_csv("diabetes.csv")

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.drop(columns=["Outcome"]), data["Outcome"], test\_size=0.25)

# Create the random forest model

model = RandomForestClassifier()

# Train the model

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

# Evaluate the model

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

accuracy = np.mean(y\_pred == y\_test)

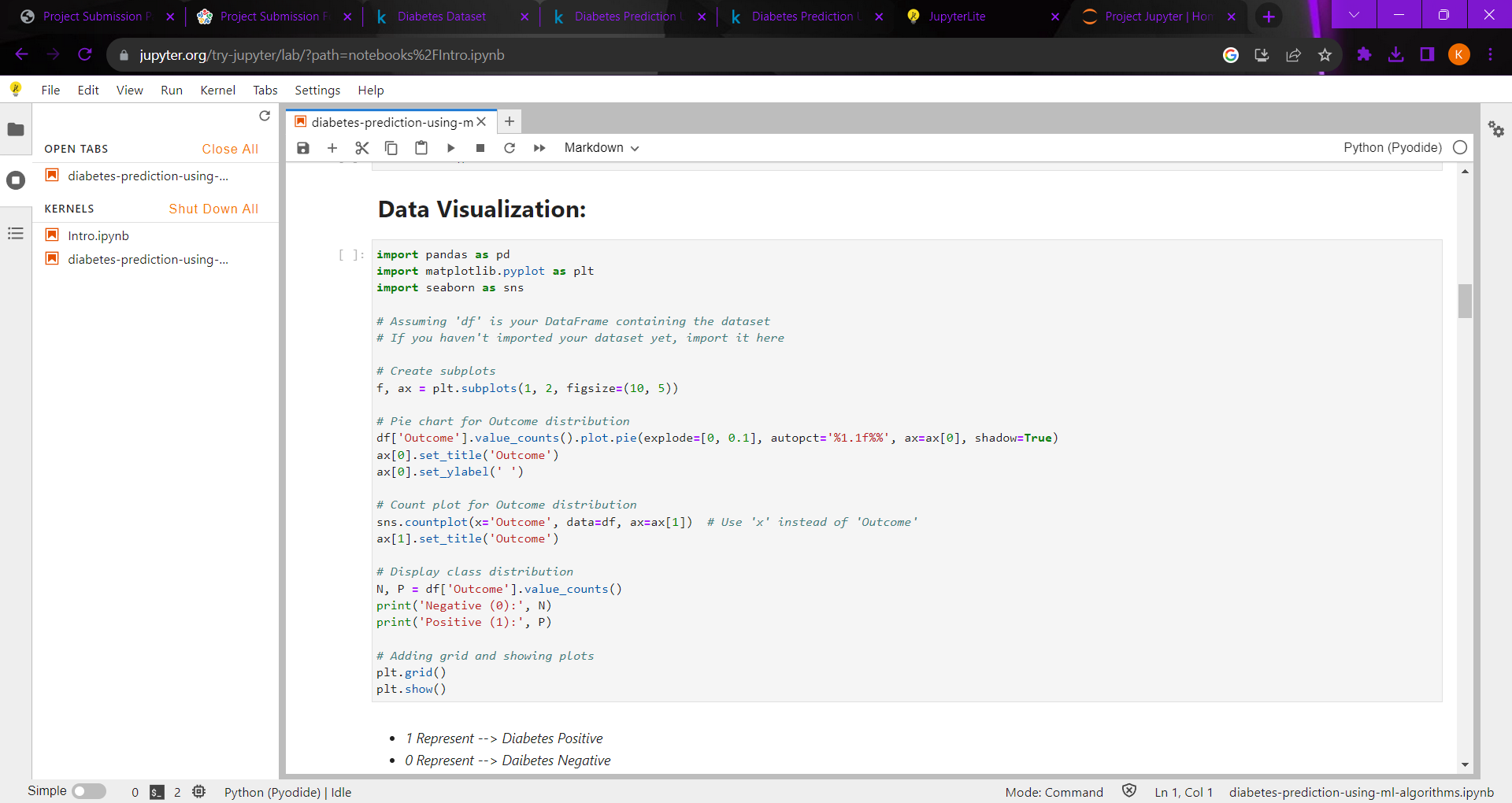
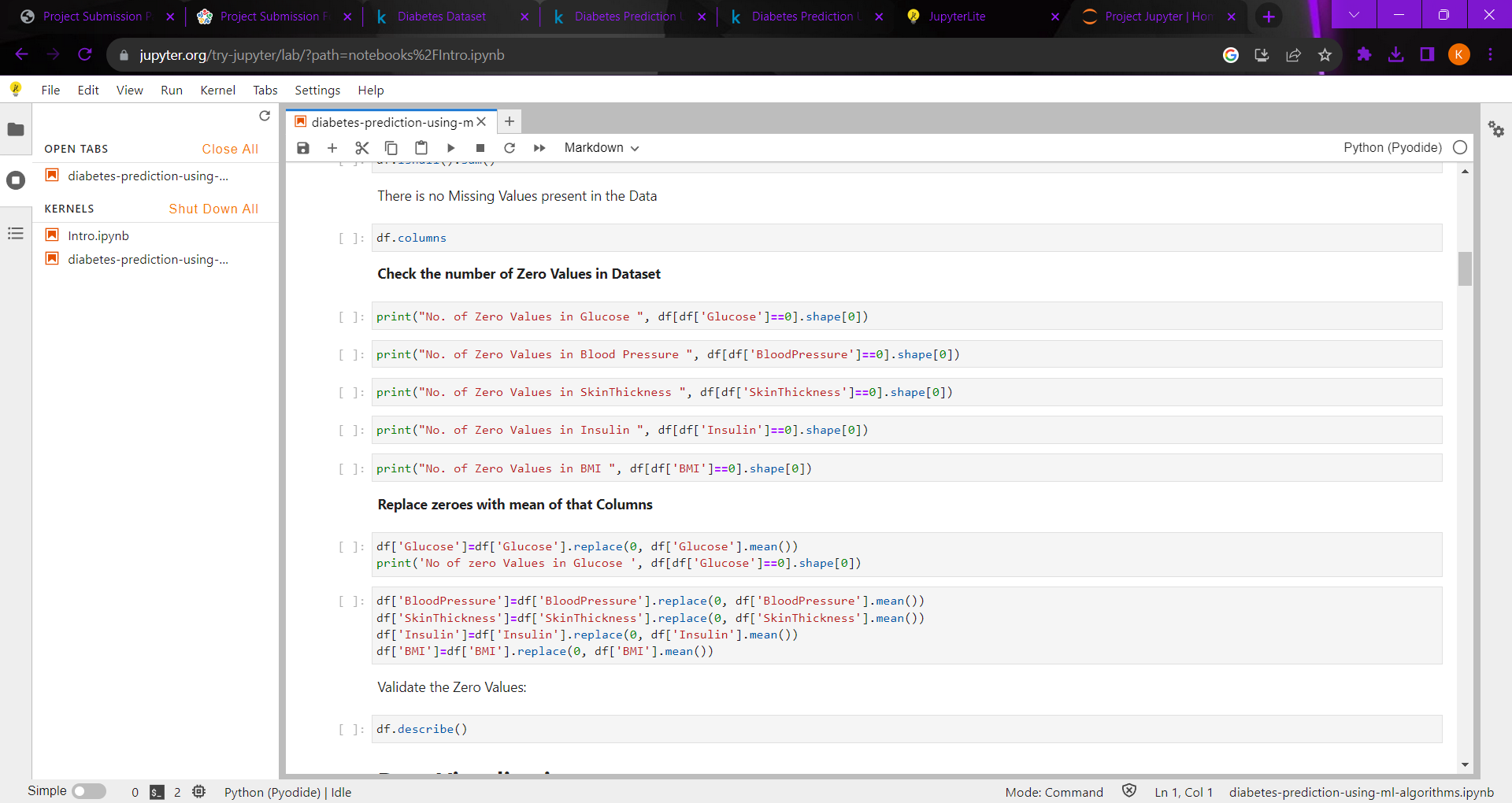
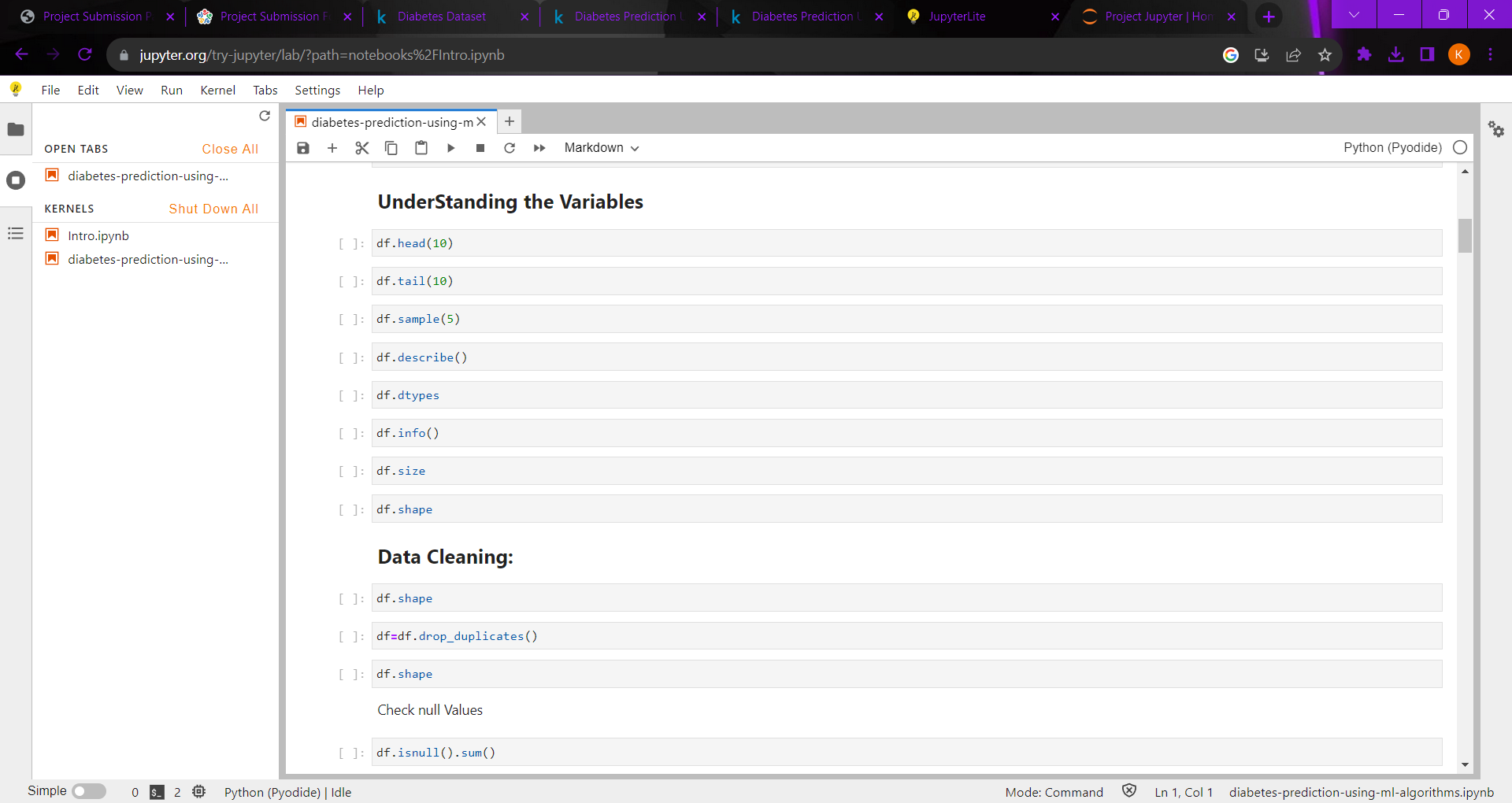
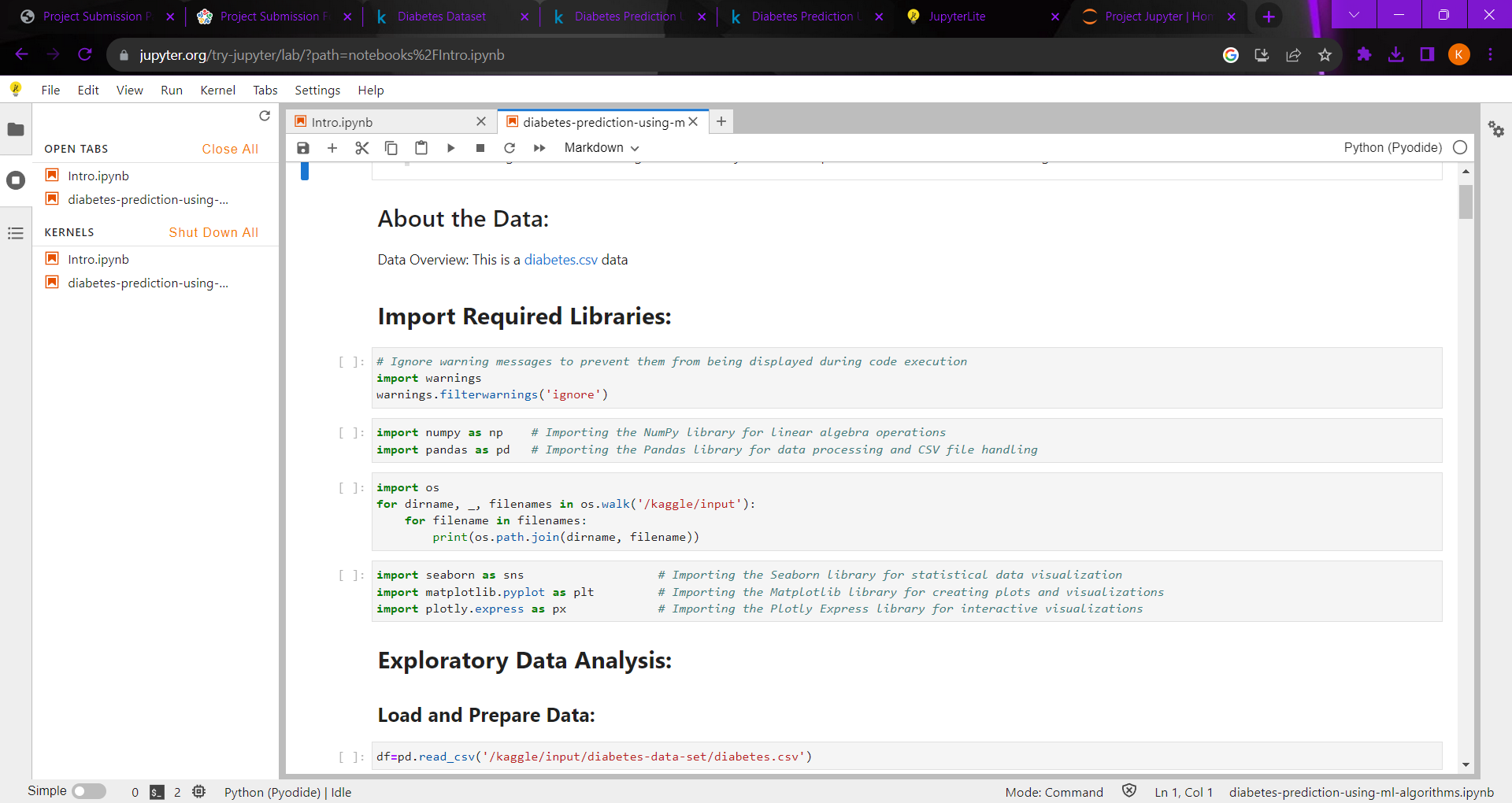
print("Accuracy:", accuracy)

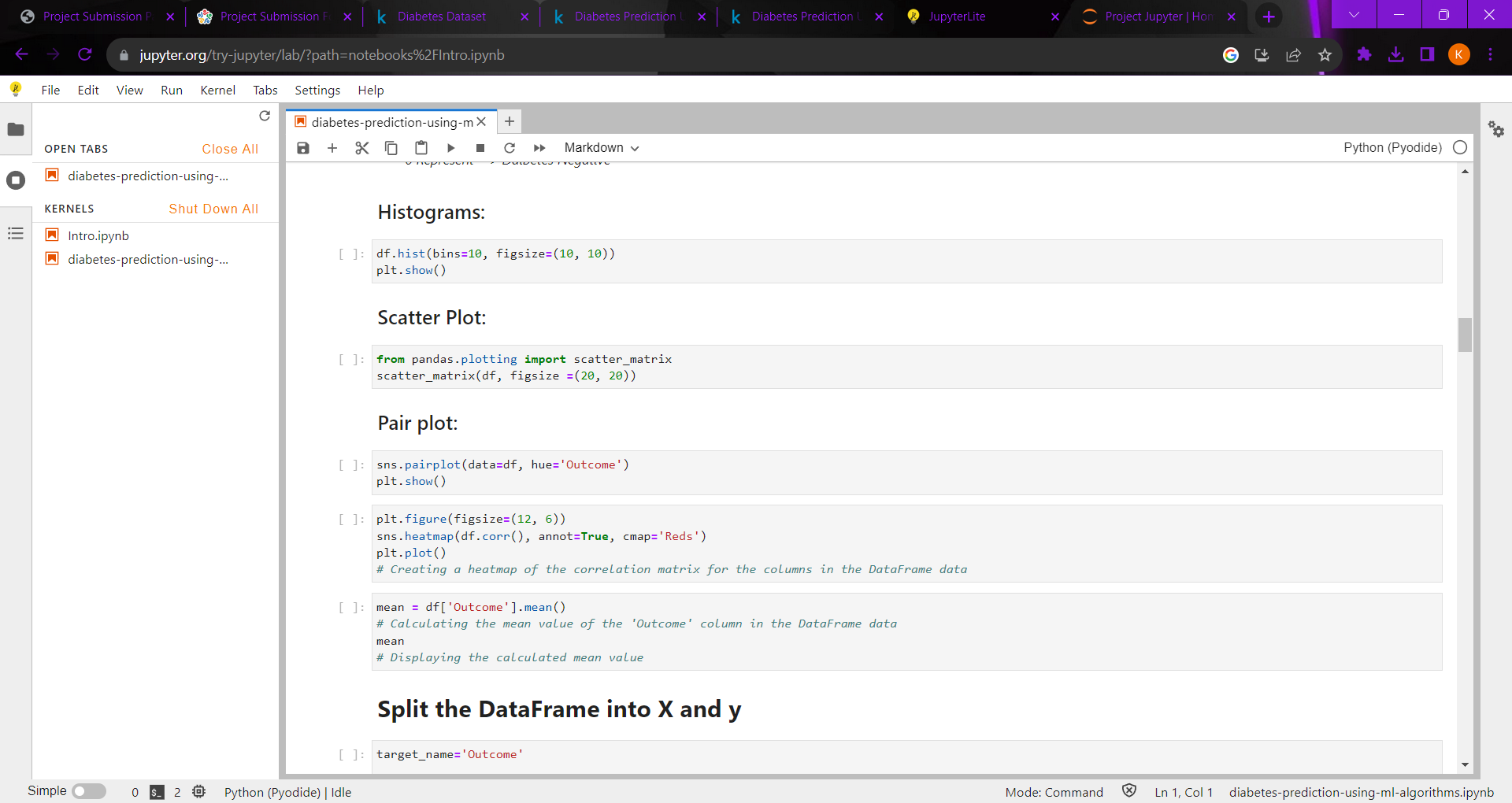
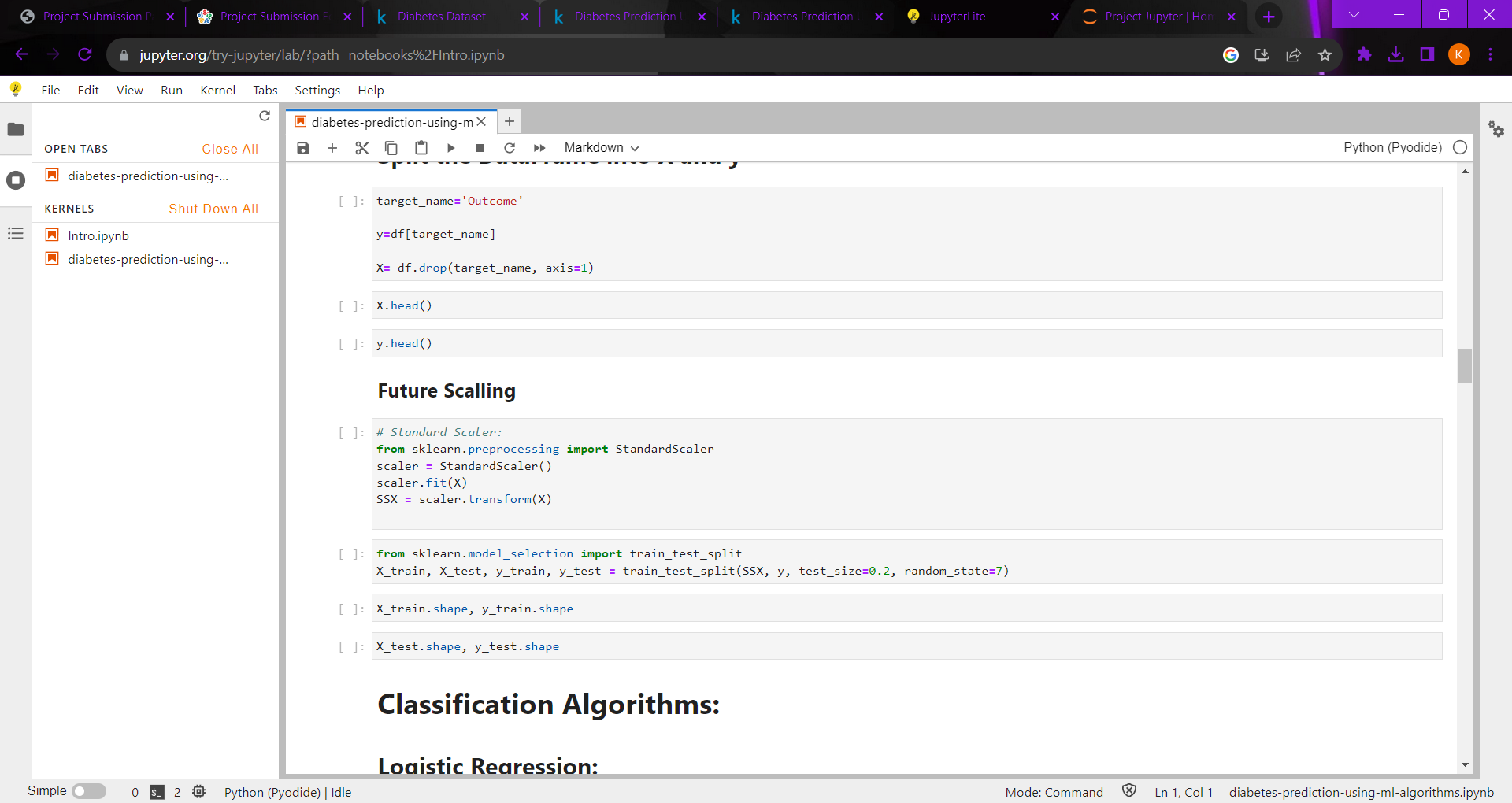
# Deploy the model

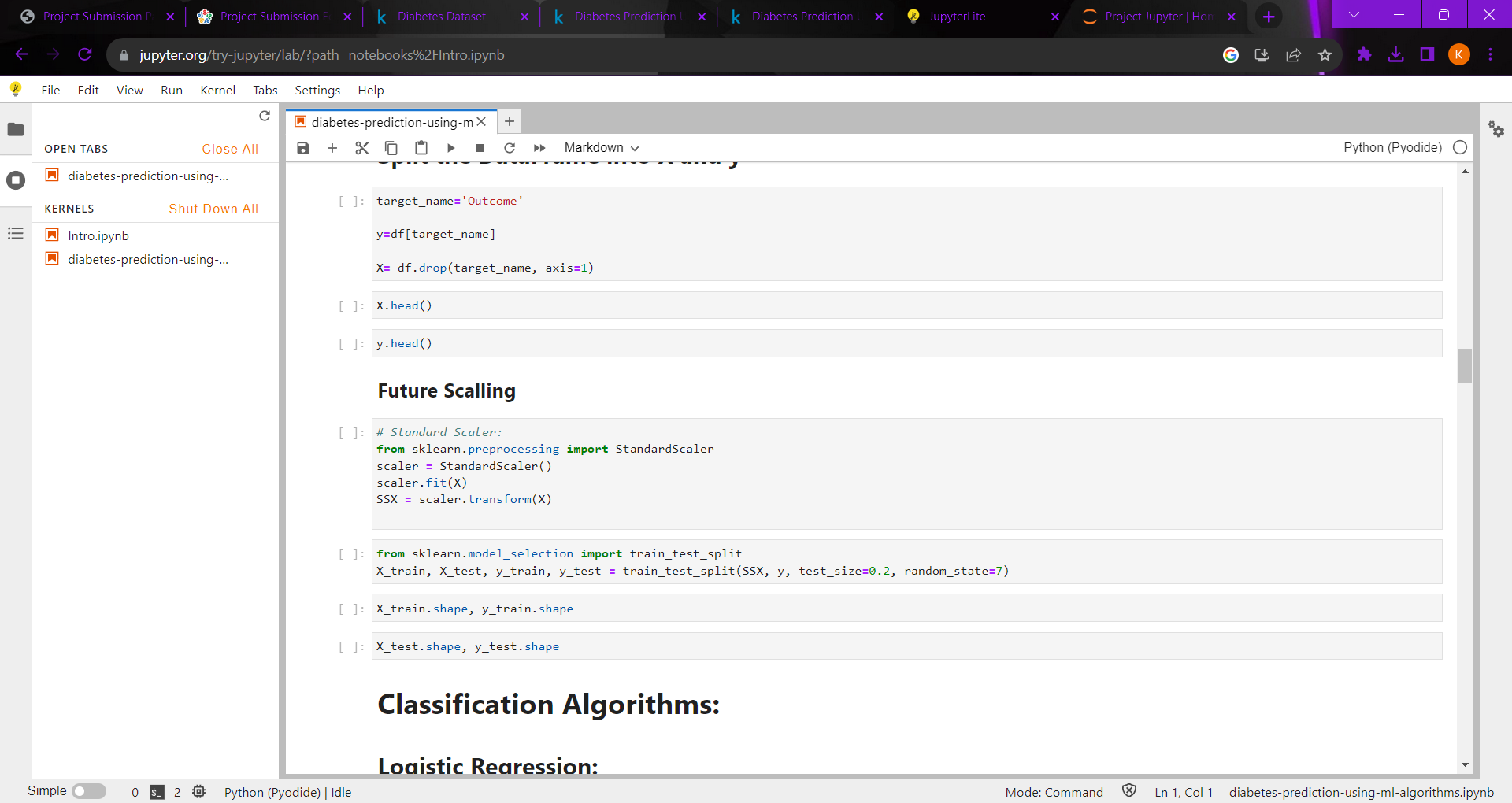
# Save the model to a file or deploy it to a web service

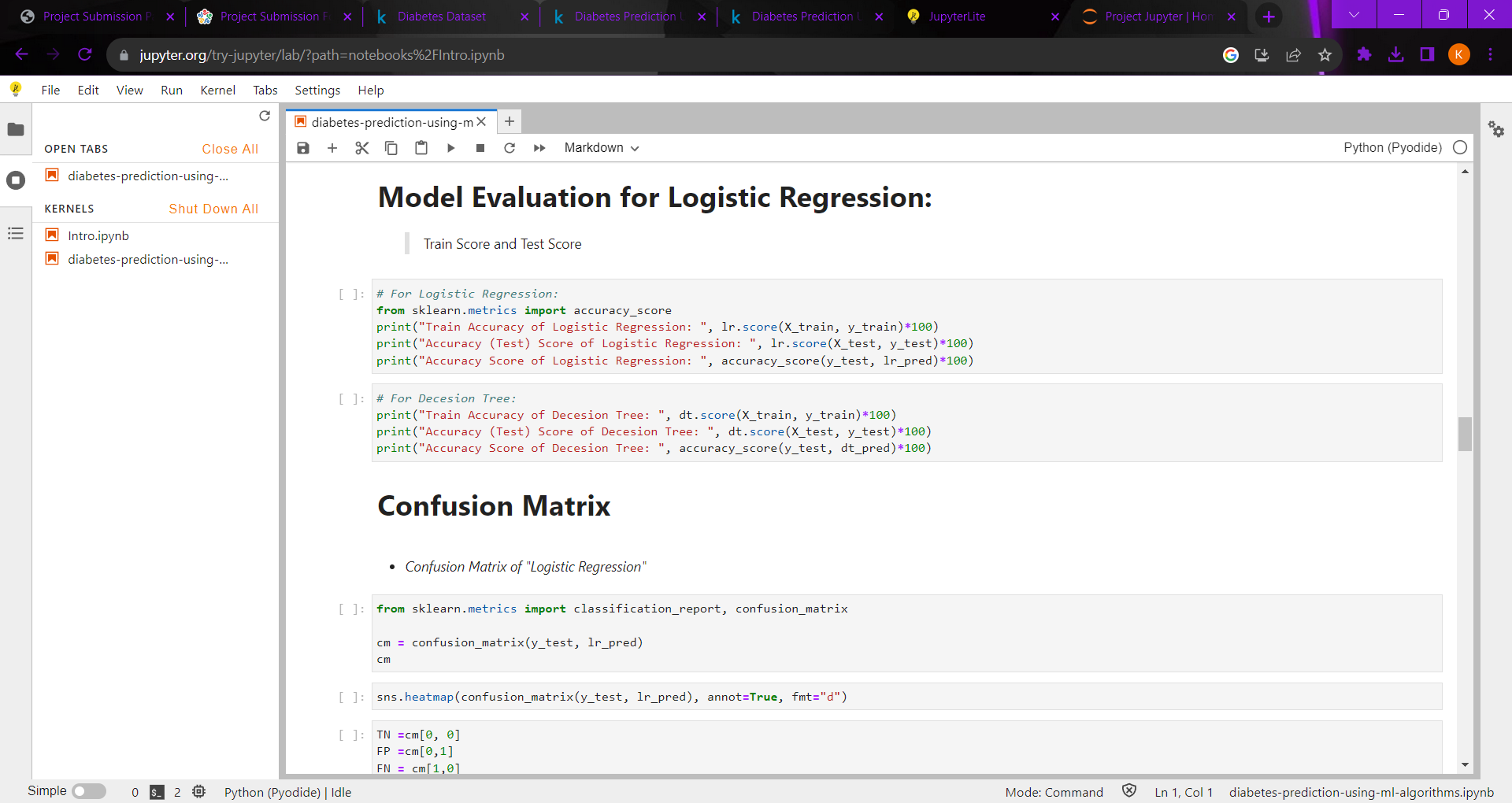
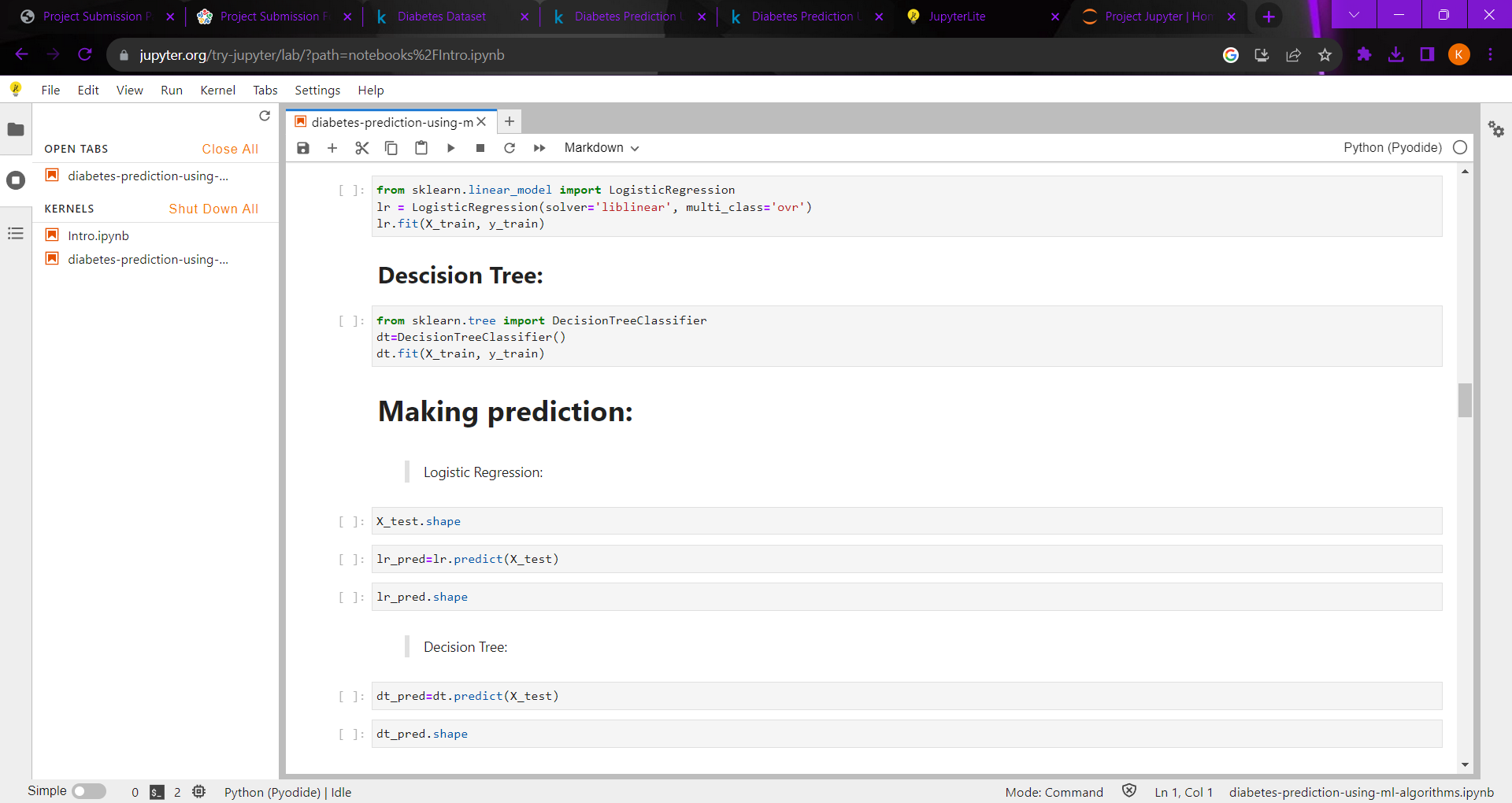
This is just a basic example, and you may need to adjust the parameters of the random forest model or try other algorithms to get the best results.

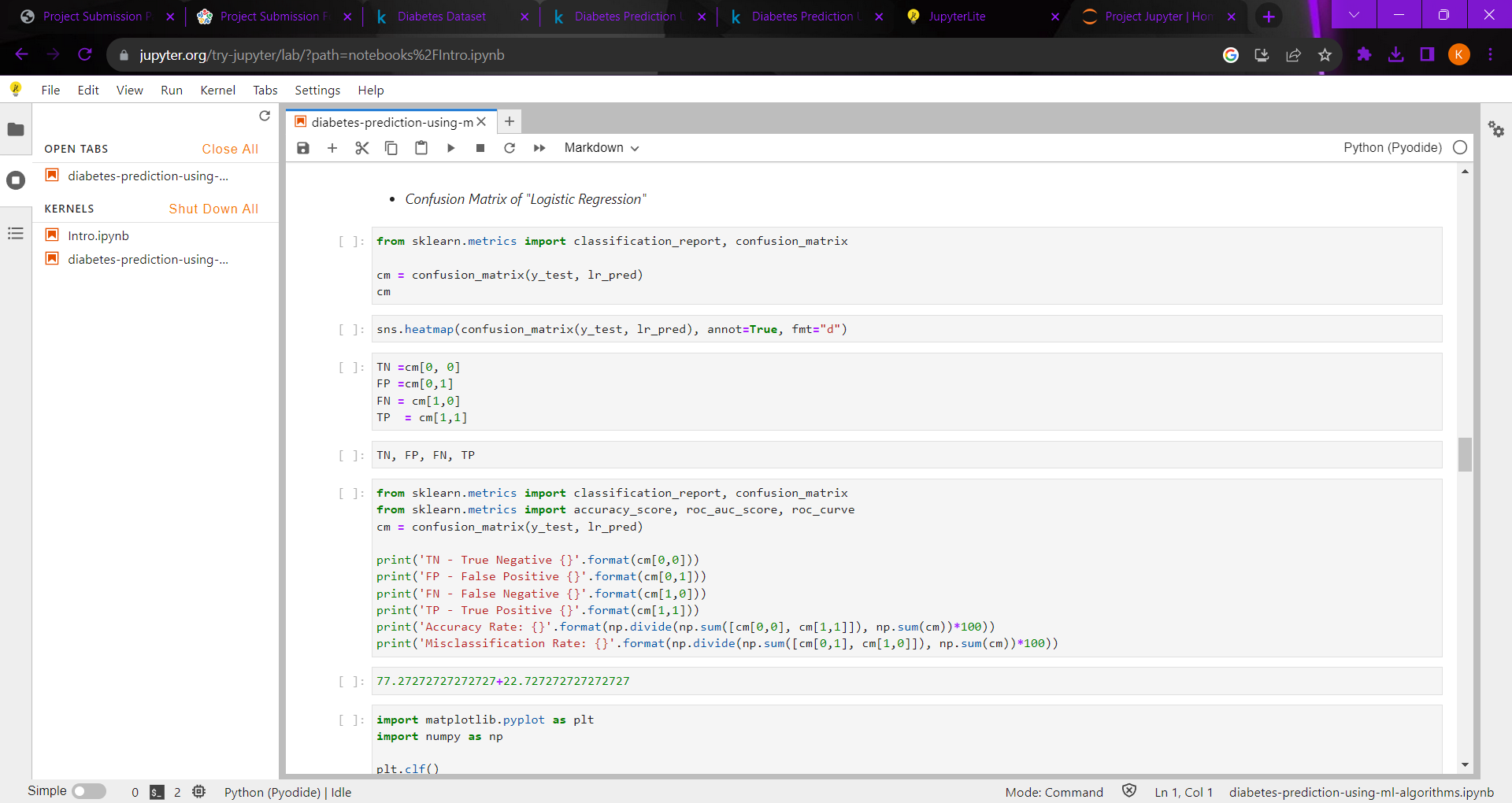
**PROGRAMS:**

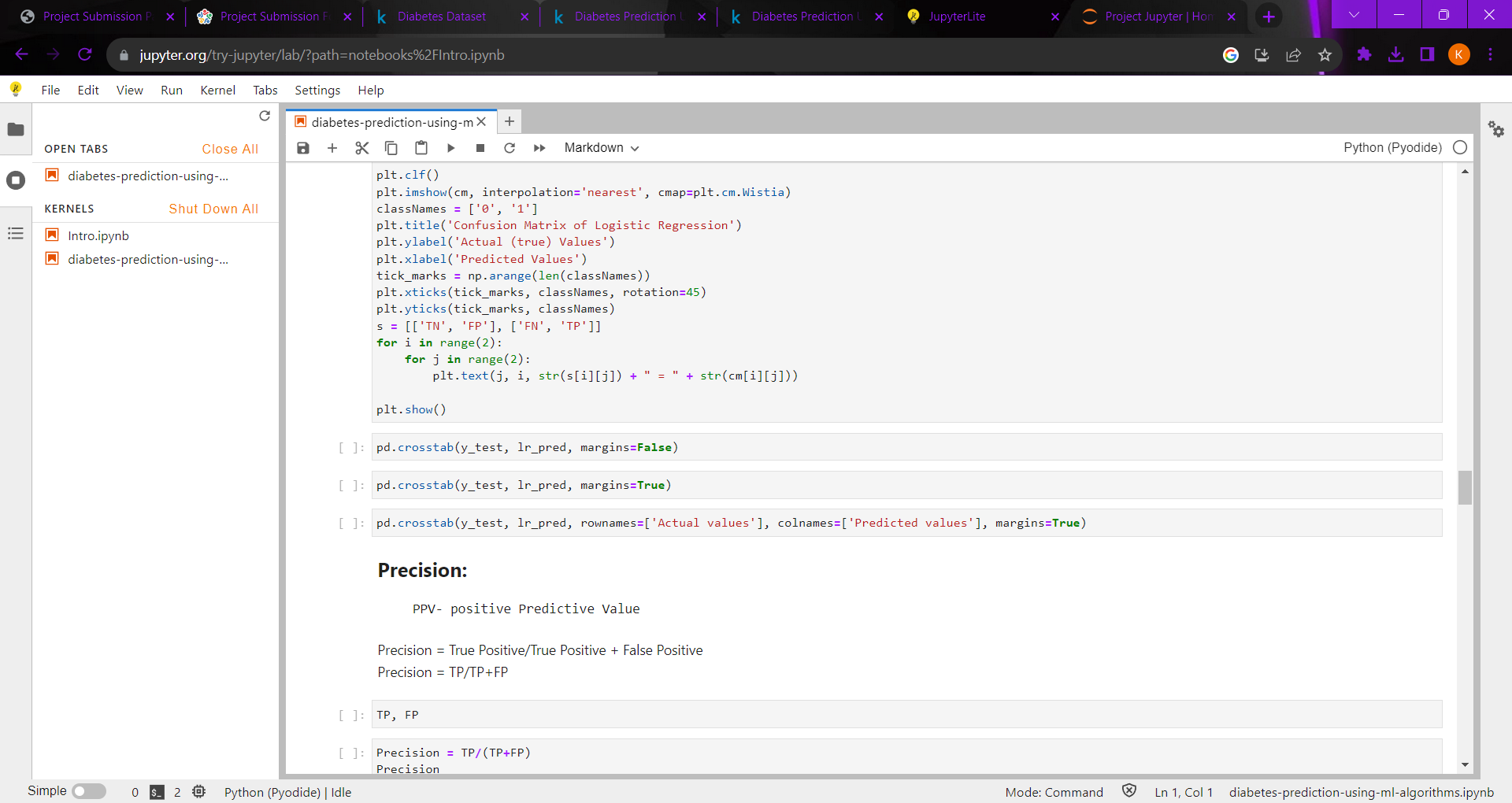


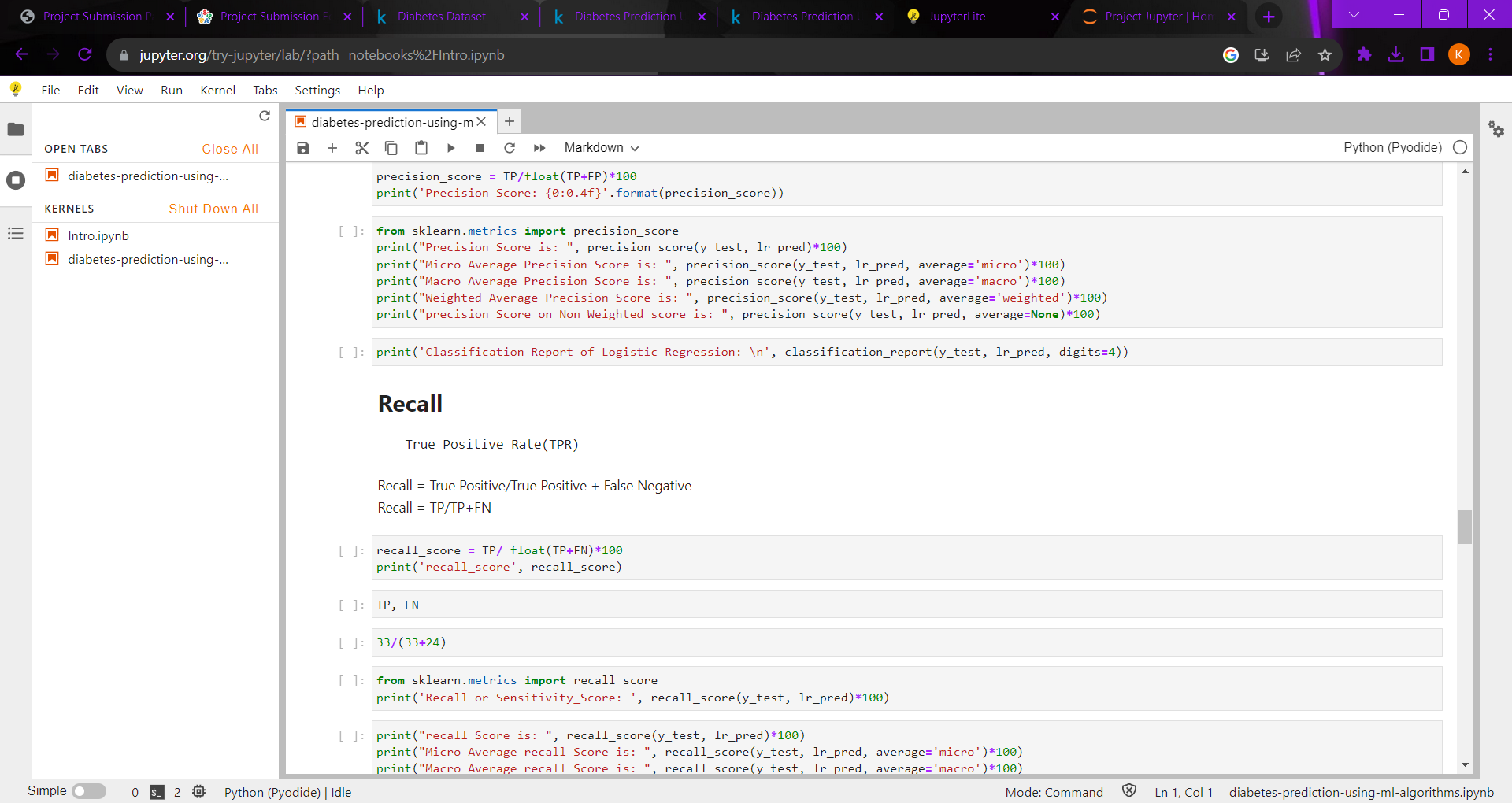
******

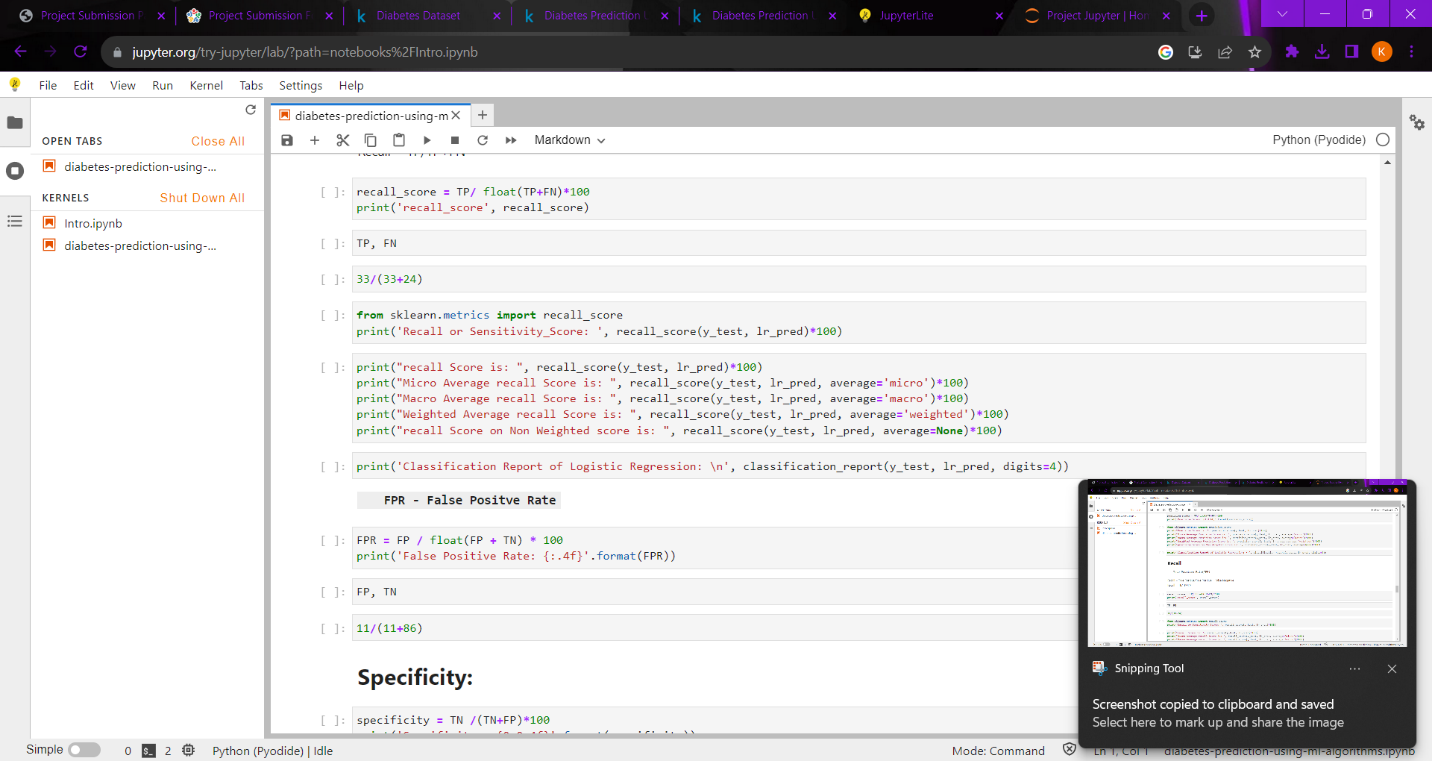
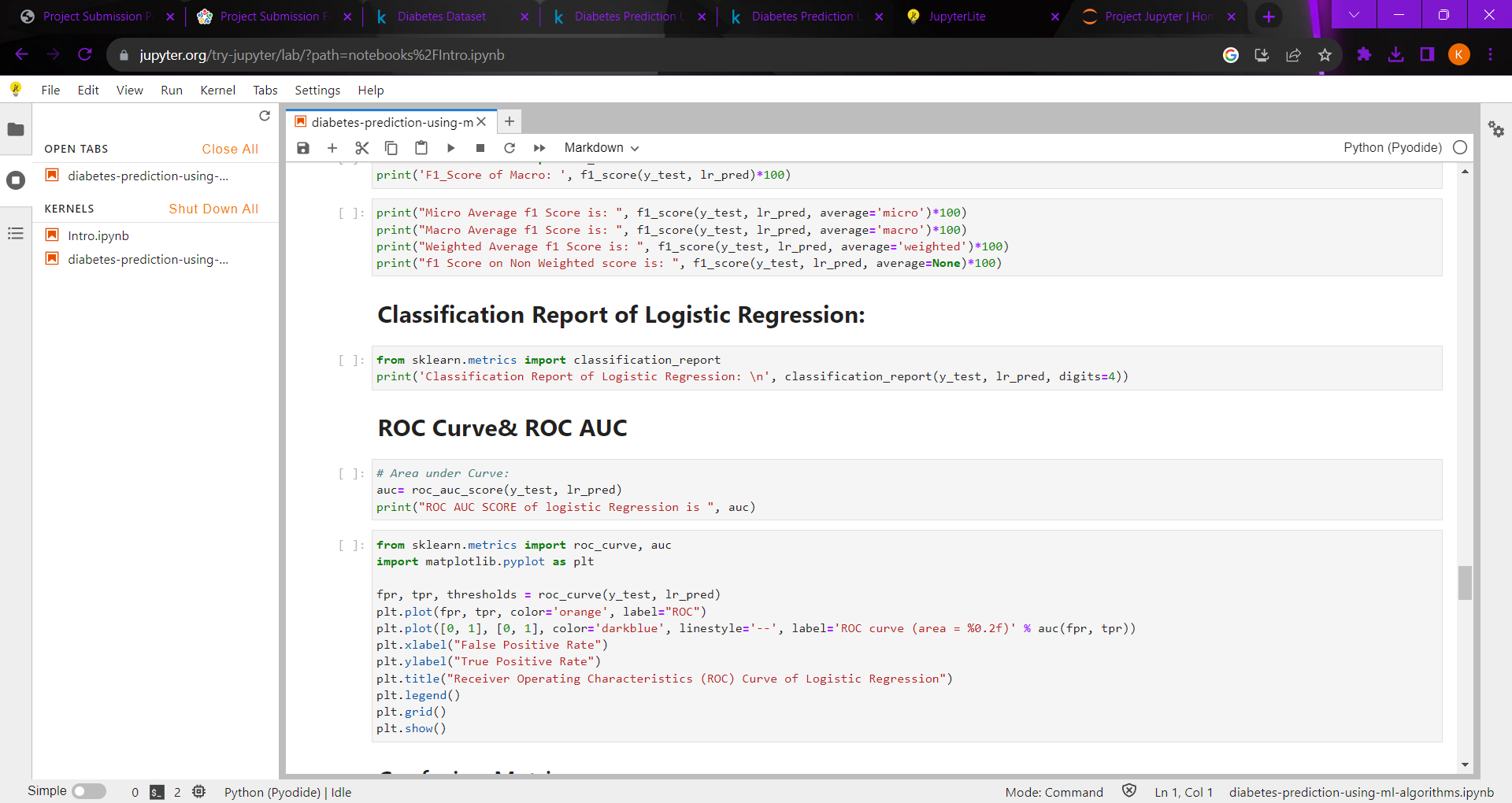


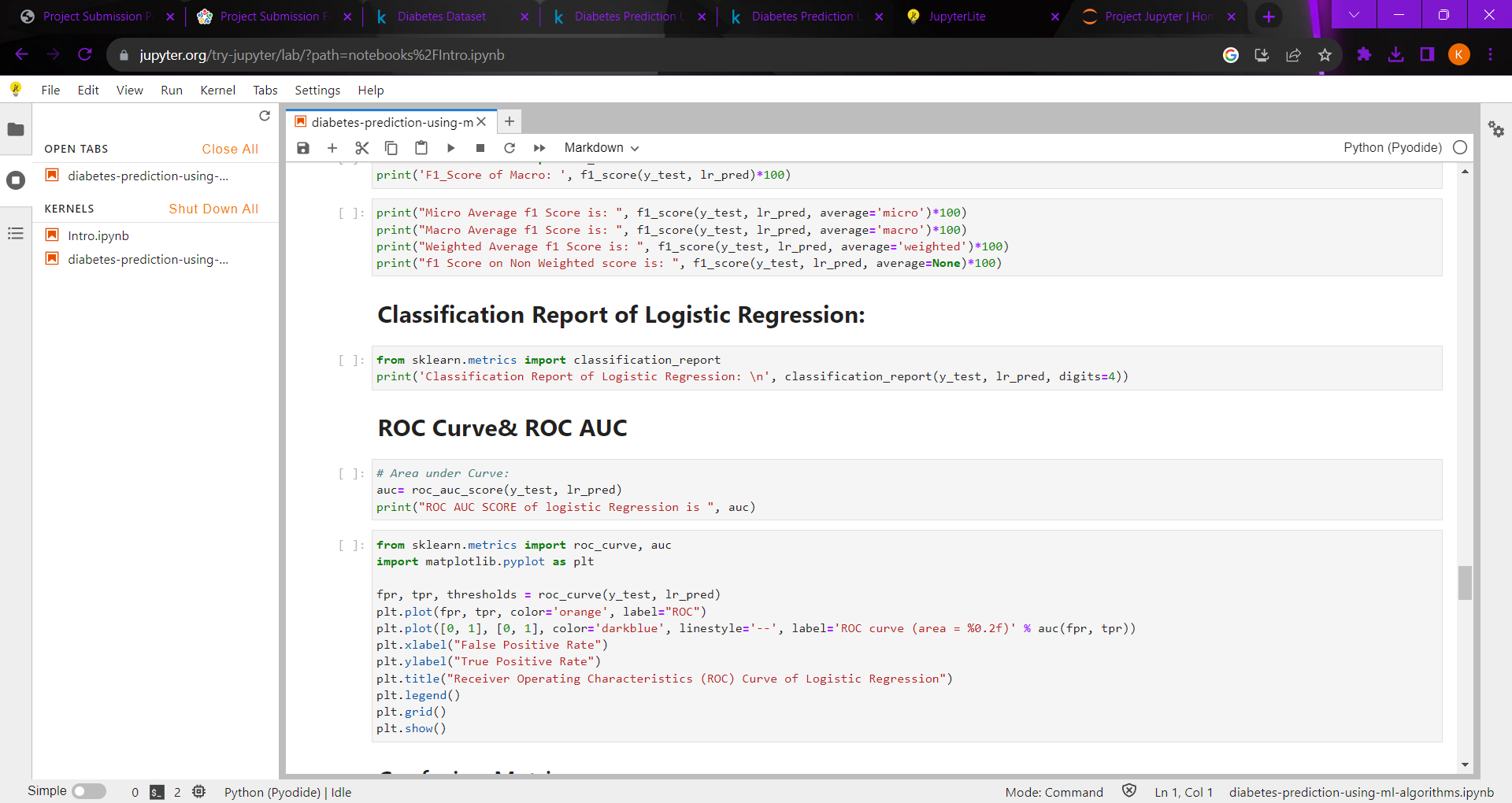
******

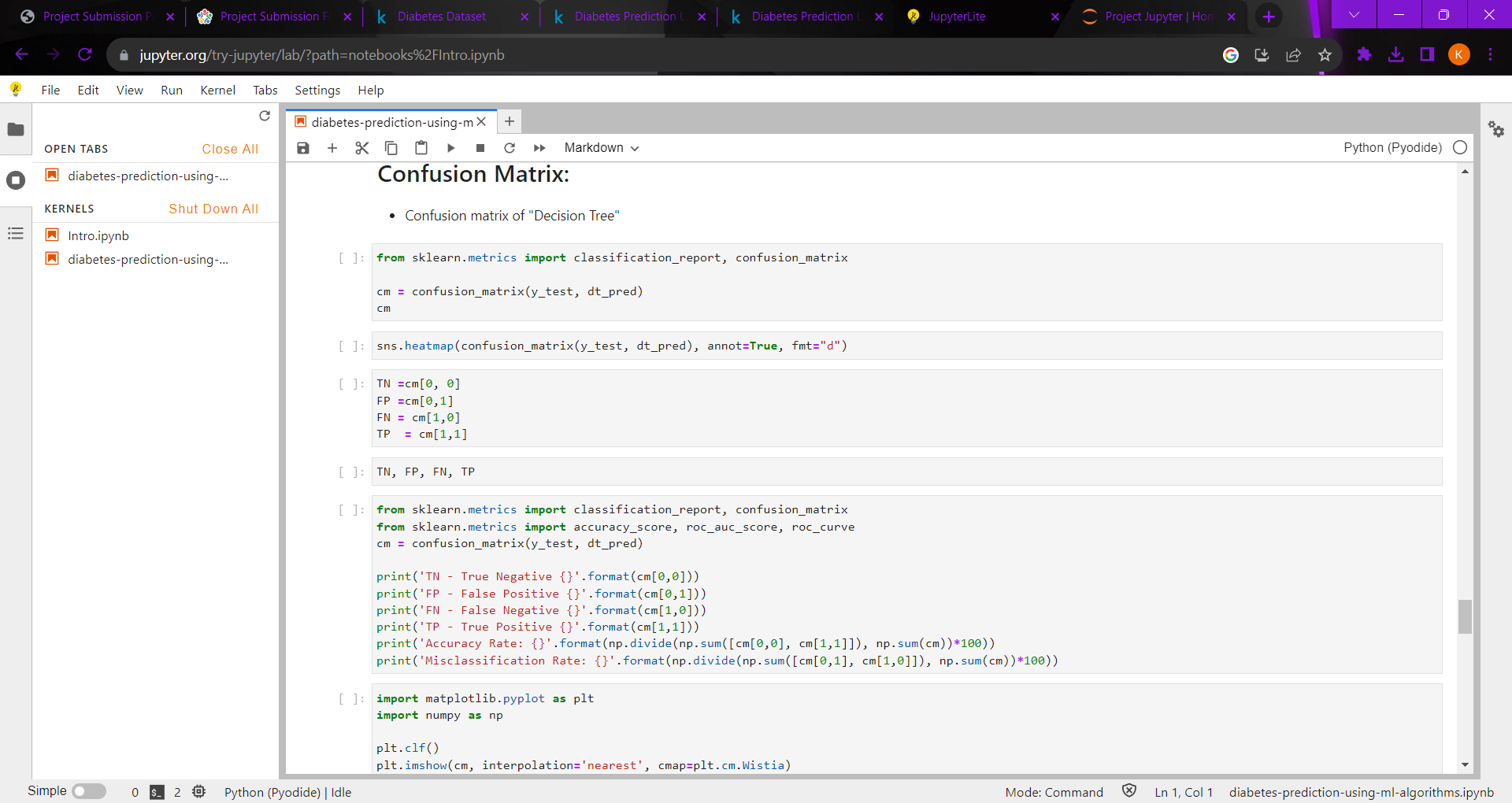
******

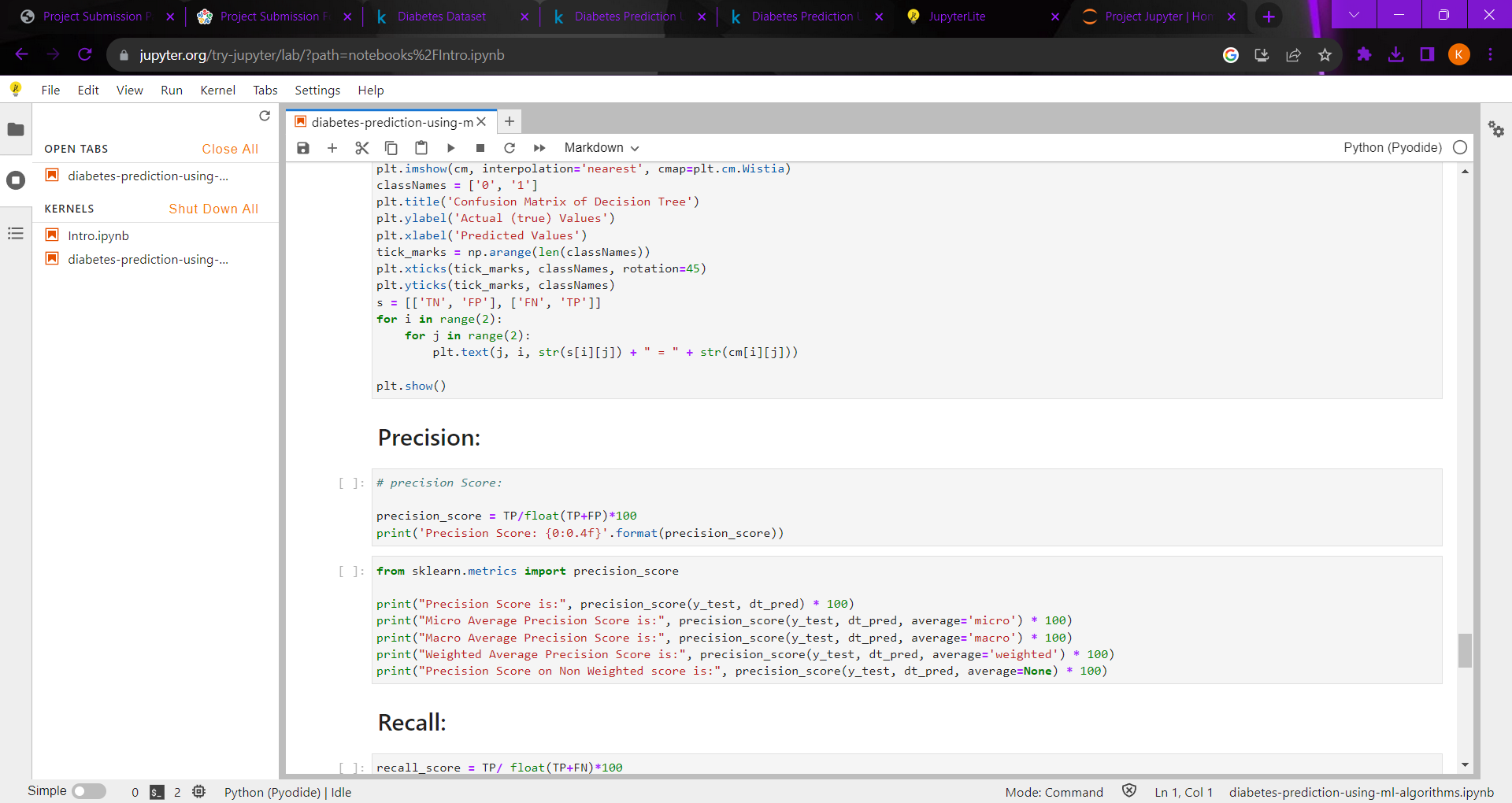
******

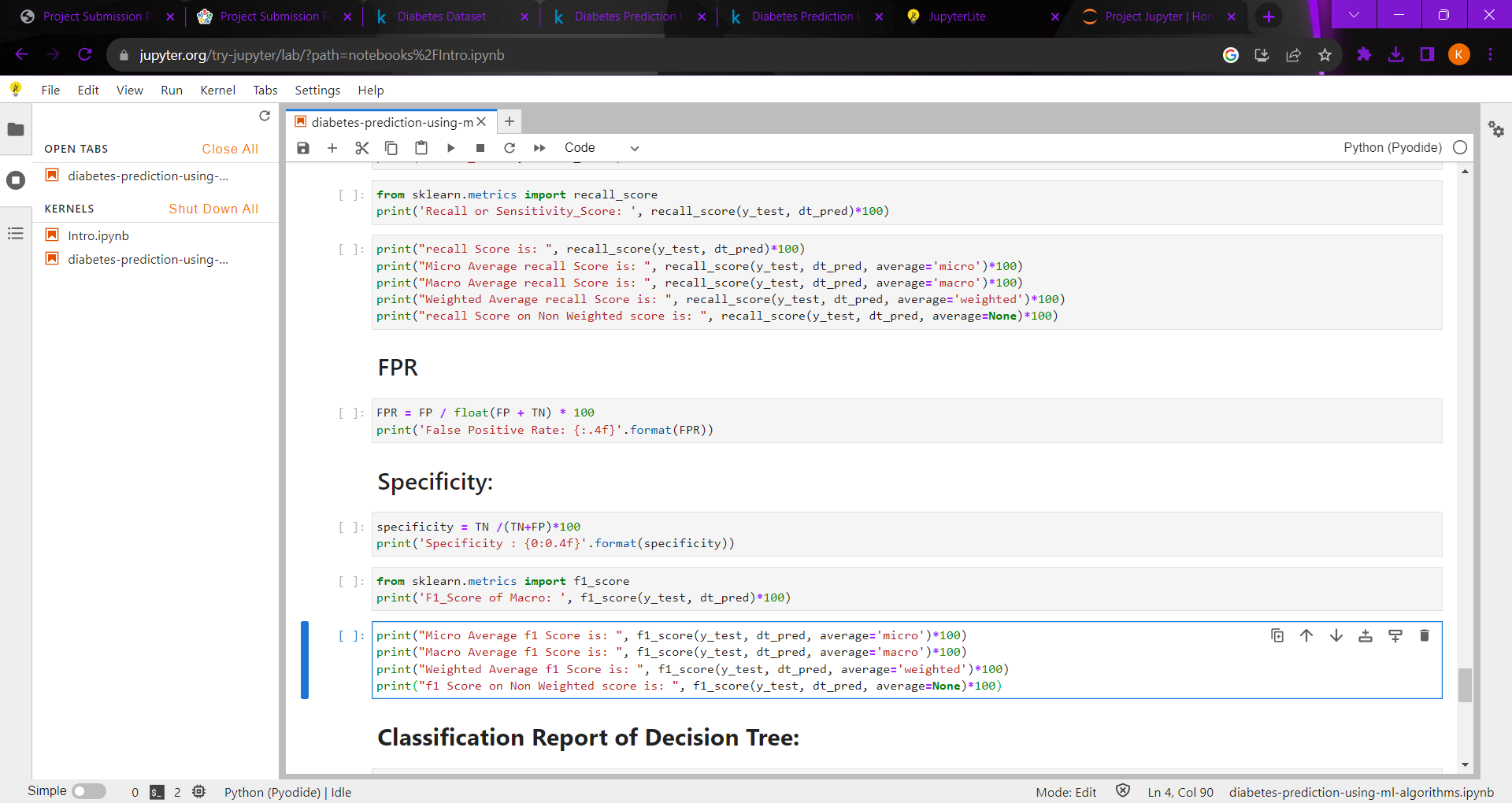
******

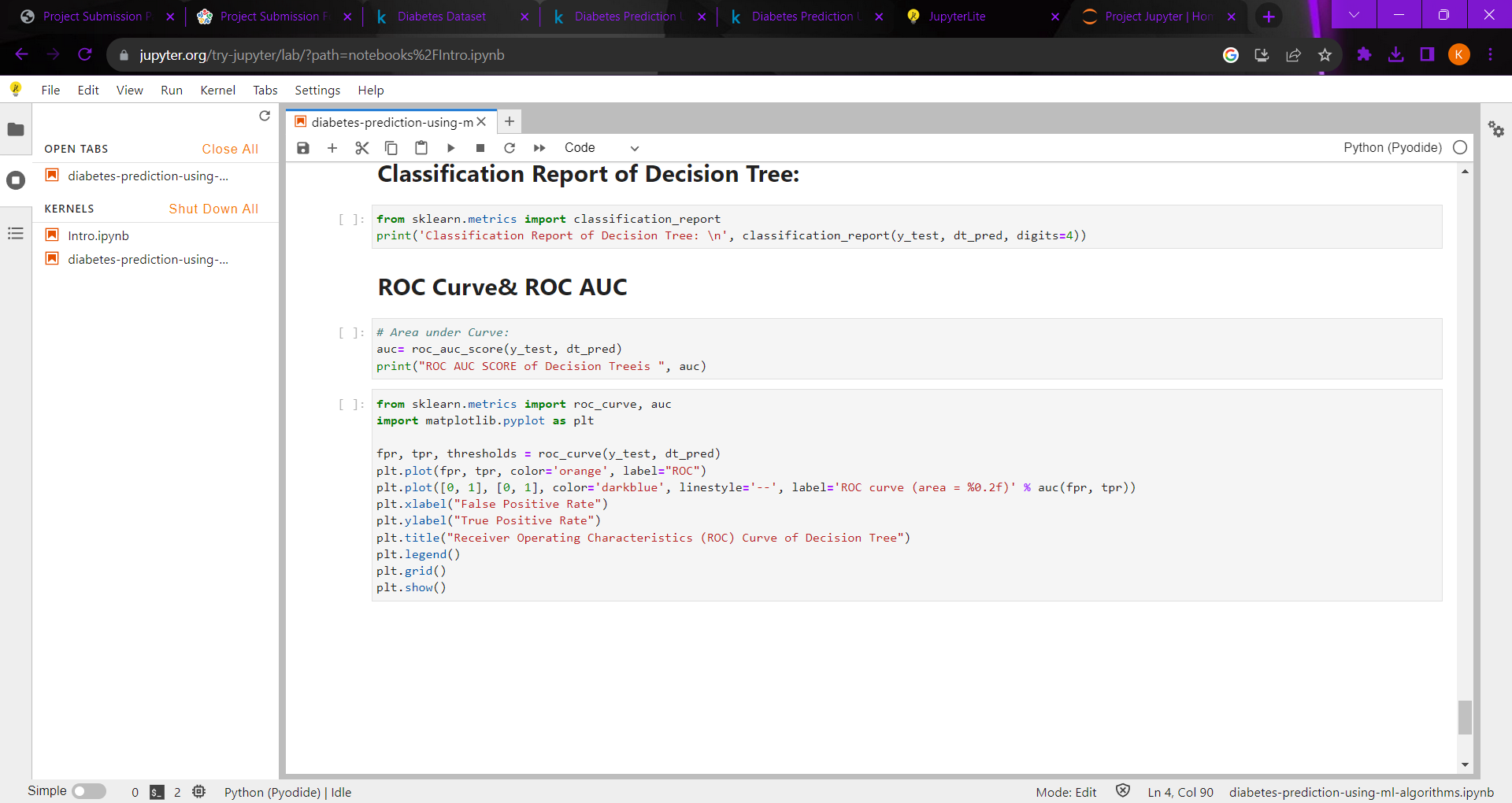
******

******

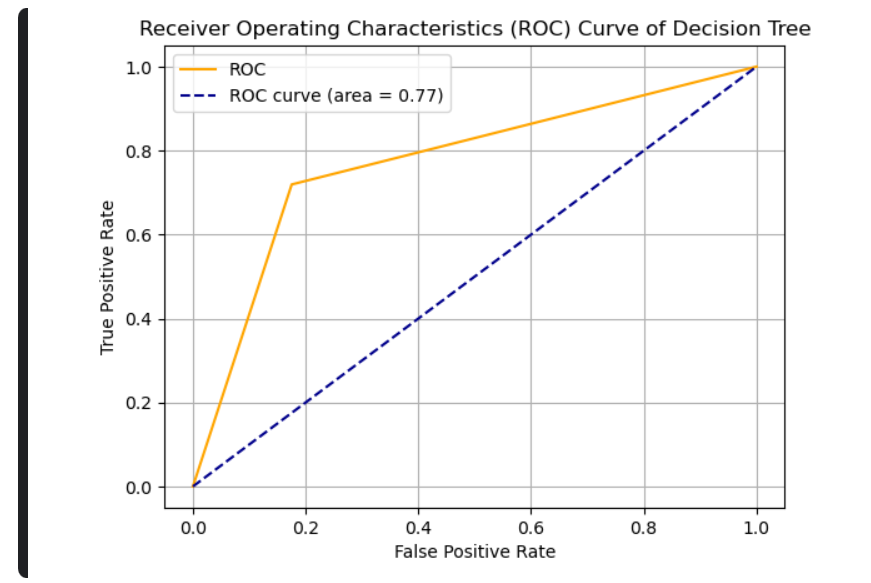
******

******

******

******

***ACCURACY WITH GRAP PLOTTED:***



Precision Score: 70.6897

Micro Average Precision Score is: 78.57142857142857

Macro Average Precision Score is: 77.01149425287358

Weighted Average Precision Score is: 78.65353037766832

**CONCLUSION:**

**I HAVE TYPED AND EXECUTED THE CODE USING JUPYTERLITE NOTEBOOK WITH RUN TIME OF 76.8S BY USING THE MACHINE LEARNING ALGORITM.**