

# **Leveraging the power of Python, Octave and Matlab for Machine Learning**

**Mohammad Rafiq Muqri (Professor CEIS)**

**Seta Boghikian-Whitby (Professor and Department Chairperson)**

**Muiz Muqri**

**Zacki Muqri**

**Sarah Muqri**

# Leveraging the power of Python, Octave and Matlab for Machine Learning

## Abstract

The objective of this paper is to bring awareness, instigate interest, and promote the need of using Artificial Intelligence (AI) and machine learning algorithms for information and engineering technology students. This paper will also attempt to review some of the widely used methods for supervised and unsupervised machine learning and the key issues that students may come across, suggest some discrete resolutions so as to provide optimal results on the accuracy and validity of Train and Test methodology.

Operational Definitions: *Artificial intelligence* is the intelligence demonstrated by machines while *machine learning* is the ability of a computer to learn and make decisions the same as a human. *Data science* is the exploration and quantitative analysis of all available structured and unstructured data to develop understanding, extract knowledge, and formulate actionable results.

Machine learning (ML) process may be divided into three main steps: data cleansing, feature extraction and optimization, and train/test system modeling. Models are evaluated based on statistics about the errors, or residuals, in the predicted values. Evaluating models is challenging since there is no testing data with labels to determine the correctness. In Python, Principal Component Analysis (PCA) is used to evaluate clustering methods. Scikit-learn is a Python library that implements the various types of machine learning algorithms, such as classification, regression, clustering, decision tree, and more. Using Scikit-learn, implementing machine learning is now simply a matter of supplying the appropriate data to a function so that you can fit and train the model. The paper will explore selected programming tools, theoretical analysis of selected machine learning algorithms and demonstrate the three main ML steps with examples.

## Background

The past decade has brought tremendous advances in an exciting dimension of artificial intelligence — machine learning. This technique for taking data inputs and turning them into predictions has enabled technology (tech) giants such as Amazon, Apple, Facebook, and Google to dramatically improve their products. It has also spurred start-ups to launch new products and platforms, sometimes even in competition with Big Tech (Mishra, 2019).

Maple simulation can manipulate mathematical expressions and find symbolic solutions to certain problems, such as those arising from ordinary and partial differential equations.

Depending on the application in as to how you define a problem, for different people with their peculiar background, Java, Matlab, Maple, and Python may end up being 'best' for that application (Shakib & Muqri, 2010).

Actually, often, one will find that a mixture of a symbolic package, and a numeric package (or library), with a little glue programming, will be best. This is because for advanced applications, one probably really wants to do

1. symbolic model manipulation
2. symbolic model simplification
3. numeric model simulation
4. code generation (for efficiency)

Some experts have reported that the premise of Matlab is numerical computing. Depending on the application, say if one just wants to numerically compute eigenvalues, inverses, or numerically solve differential equations then probably Python is the way to go, because one can easily learn Python and make use of libraries like Numpy, SciPy, and Scikit rich numerical computing tools and abundant community support and best of all it is free (Muqri & Chang, 2015).

There are a number of software products that are add-ins to Matlab Machine Learning Toolbox that let you perform supervised machine learning. These are explained as well in Appendix A.

The paper will start with an introduction, followed with justifications of different methodologies such as Matlab and Maple. Breast cancer data will be used for demonstrations. The paper will conclude by providing students with five different lab experiments to practice on using Python. The paper ends by providing student feedback. The paper includes an appendix demonstrating various Machine Learning Algorithms.

### Introduction

Machine Learning Algorithms help computer system learn without being explicitly programmed. These algorithms are categorized into supervised or unsupervised. Let us now see a few algorithms:

# Machine Learning Problems

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

Figure 1. Machine learning Problems Categories

**I. Unsupervised machine learning algorithms:** The unsupervised learning problems can be divided into the following two kinds of problems: Clustering and Dimensionality reduction.

1. **Clustering:** Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on (UCI, Center for Machine Learning and Intelligent Systems, 2022).

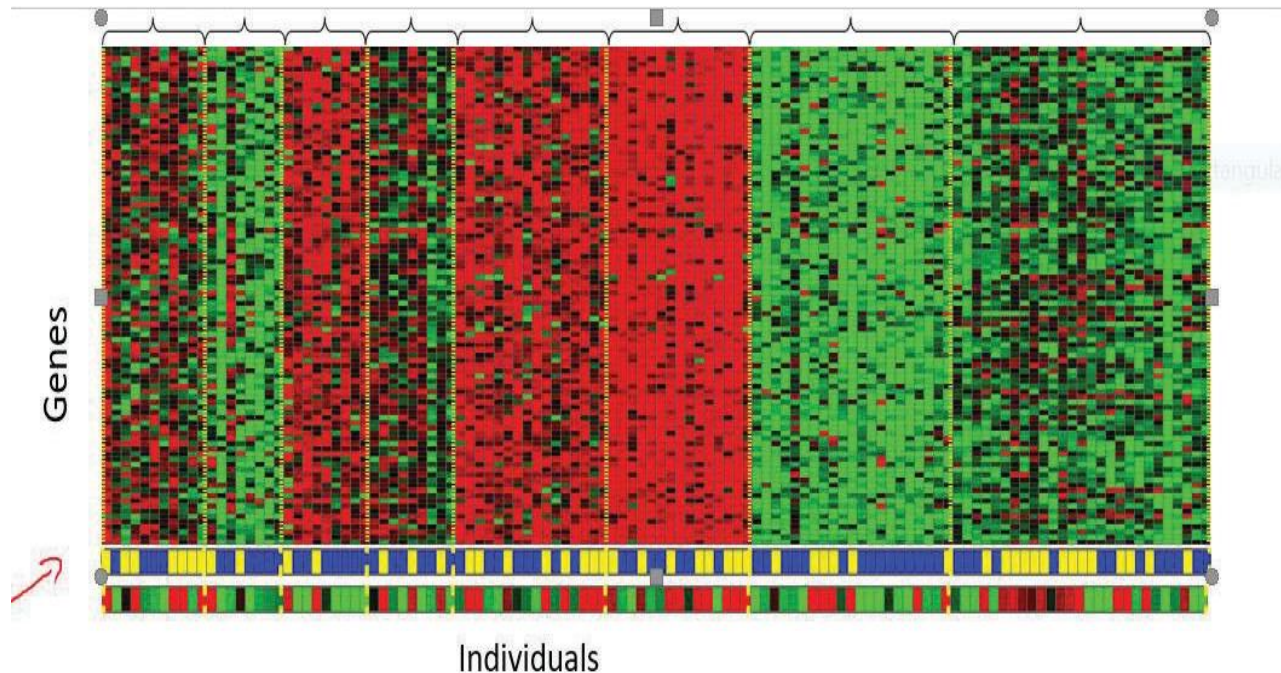


Figure 2. Unsupervised Machine learning, Gene analysis

2. **Dimensionality Reduction:** Dimensionality Reduction is under unsupervised Machine Learning

**II. Supervised machine learning algorithms:** This is the most commonly used machine learning algorithm. Supervised learning algorithms are trained with labeled data.

Mainly supervised learning problems can be divided into the following two kinds of problems:

1. **Classification:** A problem is called classification problem when we have the categorized output such as “black”, “teaching”, “non-teaching”, etc.

Suppose that you have a dataset containing the following:

Tumor size, age, Malignant. The Malignant field is a label indicating if a tumor is cancerous. When you visualize or plot the data set, you will be able to classify it into two distinctive groups “benign” or “Malignant” tumors. This type of problem is known as classification problem (Blinowska & Durka, 1994).

**2. Regression:** A problem is called regression problem when we have the real value output such as “distance”, “kilogram”, etc. Decision tree, random forest, knn, logistic regression are the examples of supervised machine learning algorithms.

The steps in machine learning are depicted below in Figure 3.

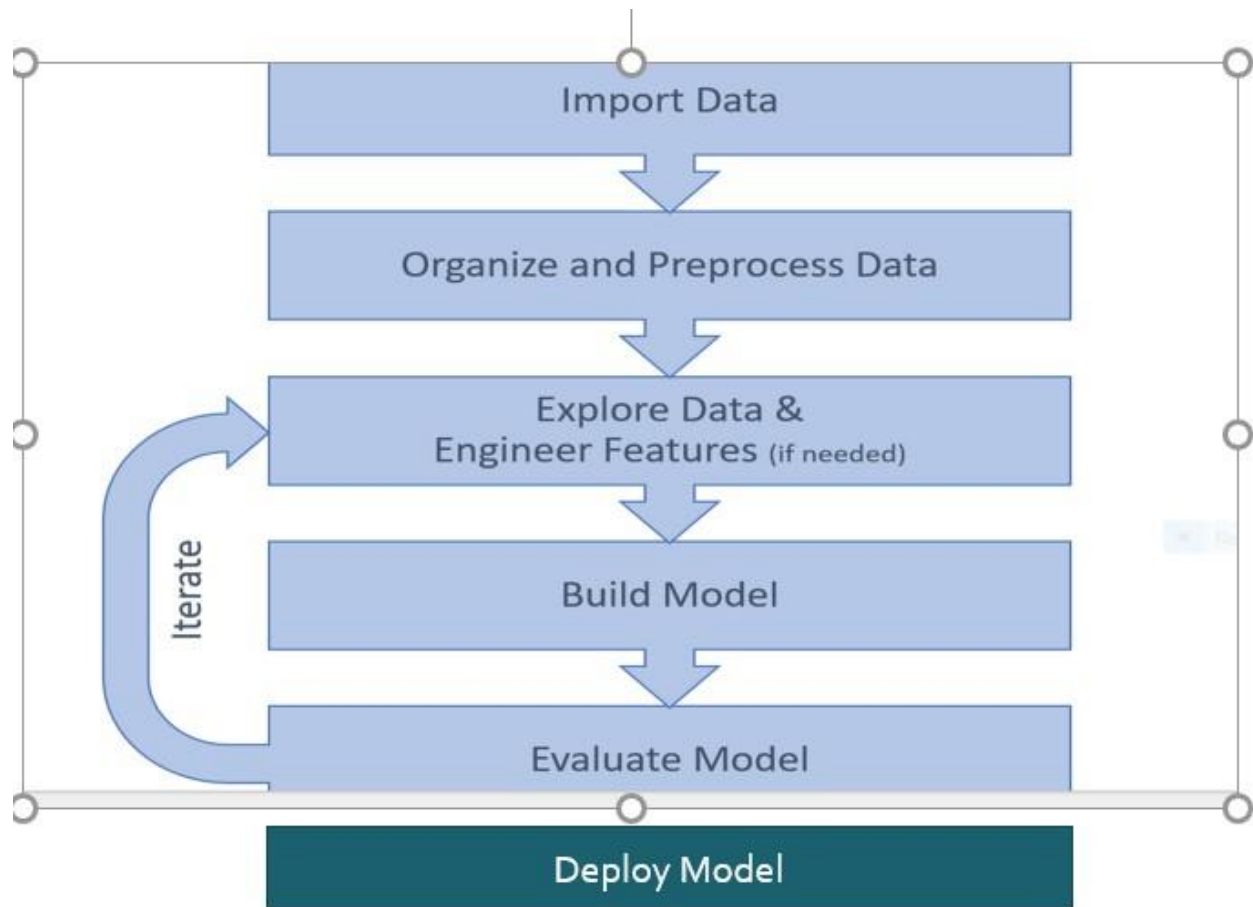


Figure 3. Machine Learning Workflow

To keep things simple we will delve into application of machine learning for cancer diagnosis. In machine learning, when you just get started with an algorithm, it is useful to get started with a simple dataset that one can create to test that the algorithm is working correctly according to your understanding, then it is time to work with relevant datasets for your machine learning model. In this paper we will show as to how to access different datasets and basics of feature engineering. A few other practical algorithms and techniques are discussed in detail in Appendix A.

The fact of the matter is the diagnosis of cancer, particularly solid tumors, relies heavily on the visual interpretation of histologic slides by pathologists who use their experience in pattern recognition to render a diagnosis.

It has long been established that the precise oncological diagnosis is a difficult and time-

consuming skill for humans to master, but an ideal task for machine learning paradigm, which can efficiently use millions of medical images to train algorithms in a relatively short period of time. It is no wonder that judicious use of machine learning as such holds promise to deliver fast and more-consistent cancer diagnoses.

**Test and Train:** Training data is a resource used by engineers to develop machine learning models. It is used to train algorithms by providing them with comprehensive, consistent information about a specific task. Training data is usually composed of large number of data points, each formatted with labels and other metadata.

**Data Normalization:** Data Normalization is a common practice in machine learning which consists of transforming **numeric columns** to a **common scale**. In machine learning, some feature values differ from others multiple times. The features with higher values will dominate the leaning process. However, it does not mean those variables are more important to predict the outcome of the model.

**Data normalization** transforms multiscale data to the same scale. After normalization, all variables have a **similar influence** on the model, improving the stability and performance of the learning algorithm.

There are multiple **normalization techniques** in statistics:

**The maximum absolute scaling.**

**The min-max feature scaling**

**The z-score method the robust scaling**

The following section presents five different Python activities for students to explore and investigate Machine Learning libraries (Wei-Ming, 2019).

### **Objectives:**

1. To access dataset using Python Scikit ML library
2. To study and analyze Data set features
3. To explore different resources for data sets including open source

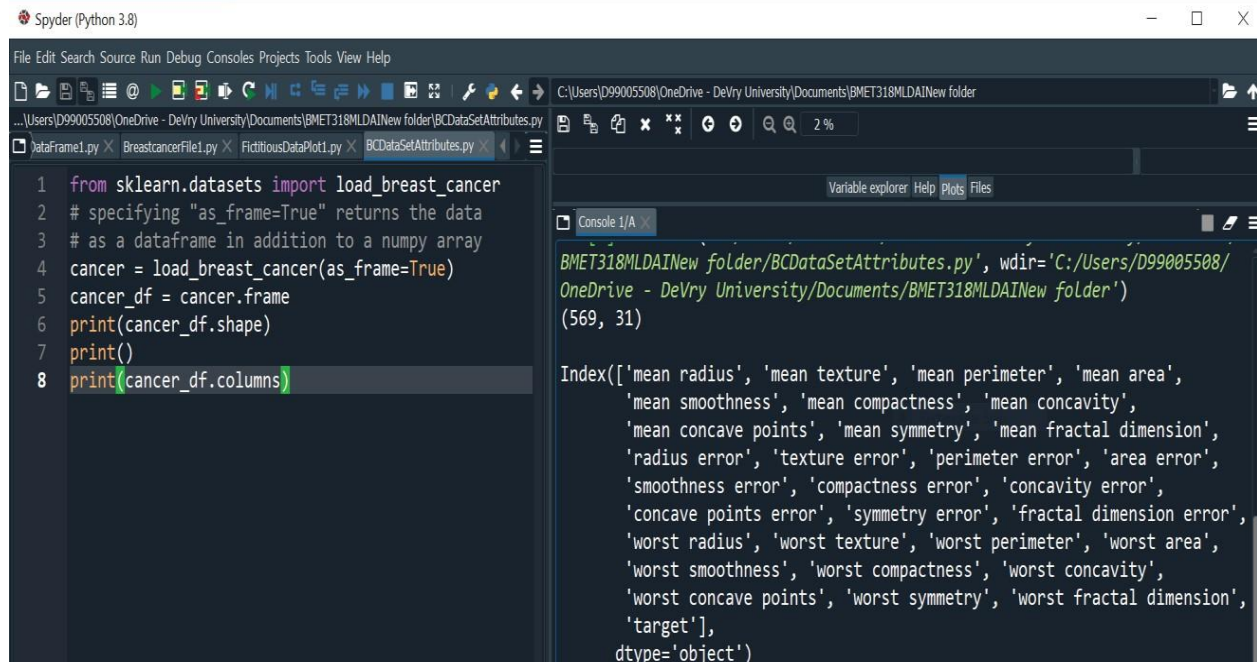
### **Student Activity I: Exploring the Scikit-learn Dataset**

To load the sample scikit data set,

1. import the datasets module and load the desired dataset.
2. Launch Spyder IDE from Anaconda.
3. Click File <New file...>
4. In the IDE window, enter Python code depicted below and
5. save as BCDDataAttributes.py
6. `from sklearn.datasets import load_breast_cancer`
7. `cancer = load_breast_cancer(as_frame=True)`
8. `cancer_df = cancer.frame`
9. `print(cancer_df.shape)`
10. `print(cancer_df.columns)# specifying "as_frame=True" returns the data as a dataframe in`



## addition to a numpy array



```
1 from sklearn.datasets import load_breast_cancer
2 # specifying "as_frame=True" returns the data
3 # as a dataframe in addition to a numpy array
4 cancer = load_breast_cancer(as_frame=True)
5 cancer_df = cancer.frame
6 print(cancer_df.shape)
7 print()
8 print(cancer_df.columns)
```

```
BMET318MLDAINew folder/BCDataSetAttributes.py', wdir='C:/Users/D99005508/
OneDrive - DeVry University/Documents/BMET318MLDAINew folder')
(569, 31)

Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
      'mean smoothness', 'mean compactness', 'mean concavity',
      'mean concave points', 'mean symmetry', 'mean fractal dimension',
      'radius error', 'texture error', 'perimeter error', 'area error',
      'smoothness error', 'compactness error', 'concavity error',
      'concave points error', 'symmetry error', 'fractal dimension error',
      'worst radius', 'worst texture', 'worst perimeter', 'worst area',
      'worst smoothness', 'worst compactness', 'worst concavity',
      'worst concave points', 'worst symmetry', 'worst fractal dimension',
      'target'],
      dtype='object')
```

Figure 4. Python script to access Breast Cancer data Attributes

The main problem with linear regression is that the predicted value does not always fall within the expected range. In such cases Logistic regression is more helpful. Following is a list of important dictionary keys for the Breast cancer Data set:

- a) Classification label names (target\_names)
- b) The actual labels (target)
- c) The attribute/feature names (feature\_names)
- d) Odds are defined as follows:
  - (1) Odds = Chances of something happening / Chances of something not happening
  - (2) Let P = Probability of success
  - (3) 1 – P = Probability of failure
  - (4) Odds = P / 1 – P

### **Student Activity II:** Exploring the Logit Function

**Logit Function:** The logit Function is the natural logarithm of the odds.

$$L = \ln [ P/(1 -P) ]$$

The logit function transfers a variable on (0, 1) into a new variable on  $(-\infty, \infty)$ .

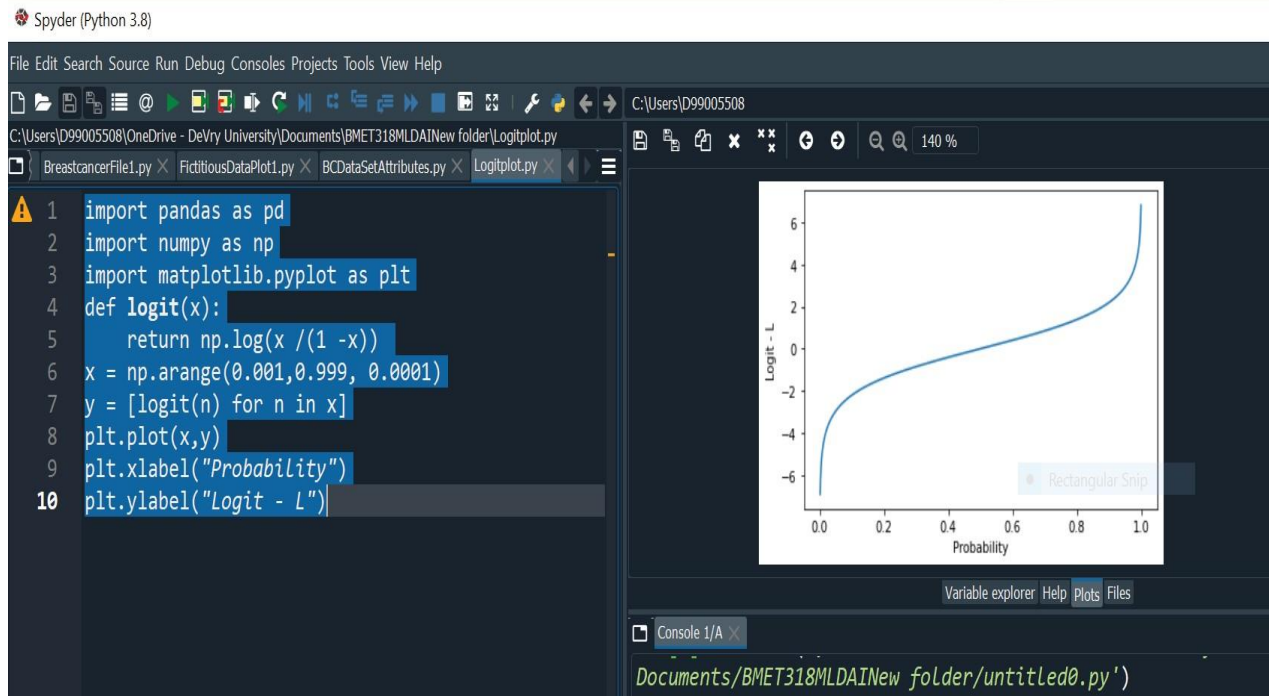


Figure 5. Code snippet to plot the Logit curve

### Sigmoid Curve:

For Logistic Regression what we need is the inverse of the logit function called the sigmoid curve. The Sigmoid function is used to transform values on  $(-\infty, \infty)$  into numbers on  $(0, 1)$ . The following code snippet shows how the sigmoid curve is obtained. Just like we try to plot a straight line that fits through all the points in linear regression, we attempt to plot a sigmoid curve that fits through all of the points.

Mathematically, this can be expressed by the formula shown as follows:

$$P = 1 / (1 + e^{- (\beta_0 + x \beta)})$$

Now we notice that  $L$  has been replaced by  $\beta_0$  and  $x\beta$ . The coefficients by  $\beta_0$  and  $\beta$  are not known, and they must be estimated based on the available training data using a technique known as Maximum Likelihood Estimation (MLE). In logistics regression,  $\beta_0$  is known as the intercept and  $x\beta$  is known as the coefficient.



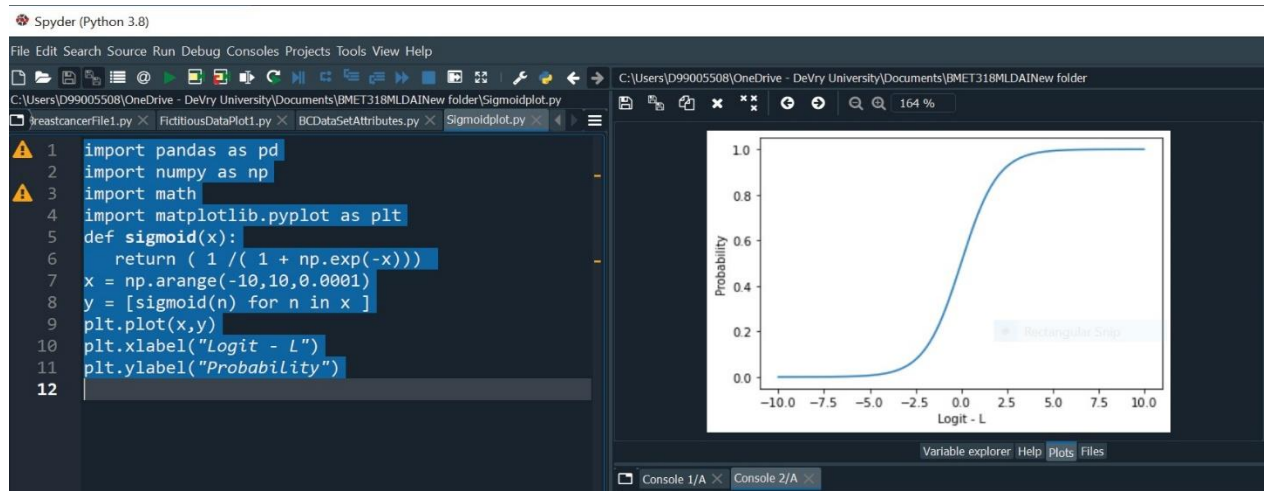


Figure 6. Flipping the Logit curve into Sigmoid curve.

In this project module you will discover how to use basic statistics and begin to prepare your data for machine learning in Python using Numpy and SciPy.

Both NumPy and SciPy are Python libraries used for used mathematical and numerical analysis. NumPy contains array data and basic operations such as sorting, indexing, etc. whereas, SciPy consists of all the numerical code. SciPy has a number of sub-packages for various scientific computations.

**Pandas** is a Python package proving fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both intuitive and easy. Pandas supports two key data structures:

- Pandas Series
- Pandas DataFrames

1. **A Pandas Series** is a one-dimensional NumPy like array, while each element having an Index (0, 1, 2, ...) by default.
2. **A Pandas DataFrame** is a two-dimensional NumPy like array and is very useful in the world of data science and machine learning.

Breast cancer is cancer that forms in the cells of the breasts. After skin cancer, breast cancer is the most common cancer diagnosed in women in the United States. Breast cancer can occur in both men and women, but it's far more common in women. Tests and procedures used to diagnose breast cancer include:

- Breast exam
- Mammogram
- Breast ultrasound
- Breast magnetic resonance imaging (MRI).

Biopsy (Removing a sample of breast cells for testing). A biopsy is the only definitive way to make a diagnosis of breast cancer. Biopsy samples are sent to a laboratory for analysis where experts determine whether the cells are cancerous. A biopsy sample is also analyzed to determine the type of cells involved in the breast cancer, the aggressiveness (grade) of the cancer, and whether the cancer cells have hormone receptors or other receptors that may influence your treatment options.

In a Fine Needle Aspiration (FNA) biopsy, a very thin, hollow needle attached to a syringe is used to withdraw (aspirate) a small amount of tissue from a suspicious area. The needle used for an FNA biopsy is thinner than the one used for blood tests.

The correct diagnosis of Breast Cancer (BC) and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

### **Examining the Relationship Between Features of Breast Cancer (BC) Diagnostic Dataset:**

Scikit-learn's Breast Cancer (BC) is a classic data set used to illustrate binary classifications. It consists of 30 features, which are computed from a digitized image of fine needle aspirate (FNA) of a breast mass. Since the Breast Cancer (BC) has shown to be loaded and its attributes listed, it is useful to examine the relationship between some of its features.

### **Student Activity III: Plotting the two Dataset features in 2D.**

The following code snippet performs 4 functions:

```
Loads the BC Diagnostic dataset with binary classification: M for Malignant, B for Benign
Copies first two dataset features namely : "mean radius" and "mean texture" in a 2-D List
Choose the tumor growth Display Colors : red (Malignant); blue (Benign)
Scatter plot depicts the distribution for the first two features of dataset
```

From the scatter plot shown in Figure 4, one will be able to view that **as the tumor grows in mean radius and increases in mean texture, the more likely it would be diagnosed as Malignant**

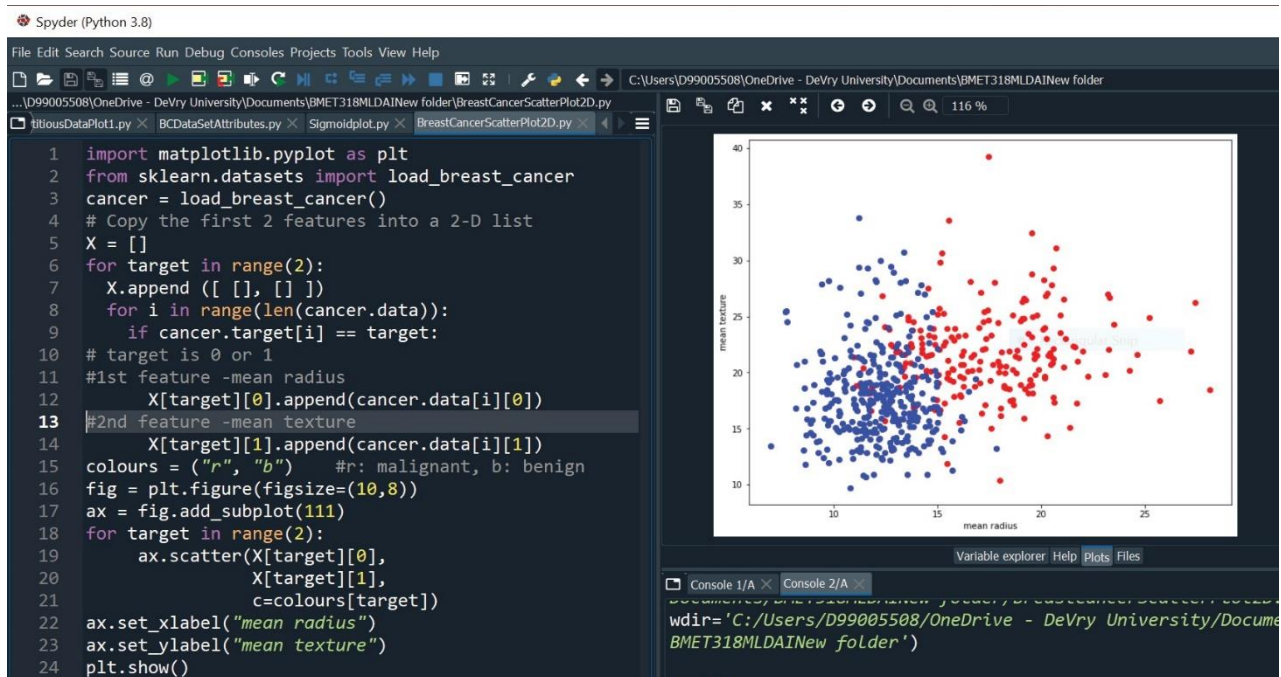


Figure 7. Scatter Plot demo of 2 Dataset features in 2D

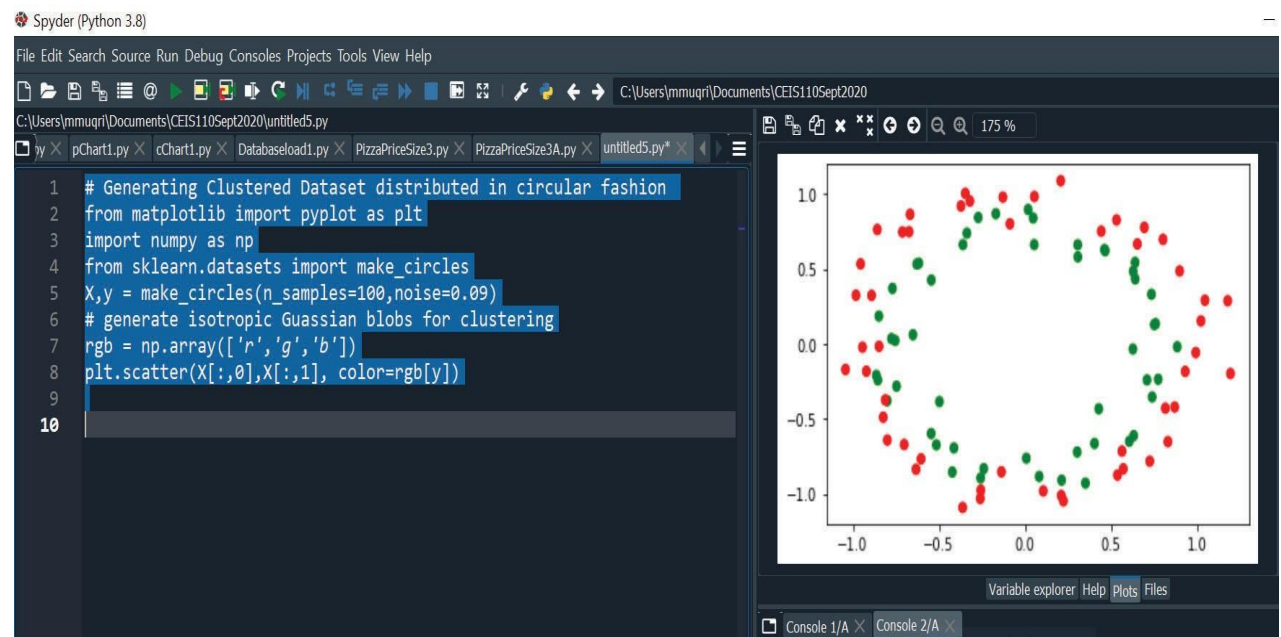


Figure 8. Plot of distributed cluster Dataset in circular fashion

## Train and Test Models:

### Training Using One Feature:

Let us now use Logistic Regression to try to predict if a breast tumor is benign or malignant. To keep it simple, we will begin with only first feature of the Wisconsin Dataset from scikit learn, namely mean radius. The code snippet given below depicts a scatter plot showing if a tumor is benign or malignant based on just one feature **mean radius**.

### Student Activity IV: Uni feature Scatter Plot

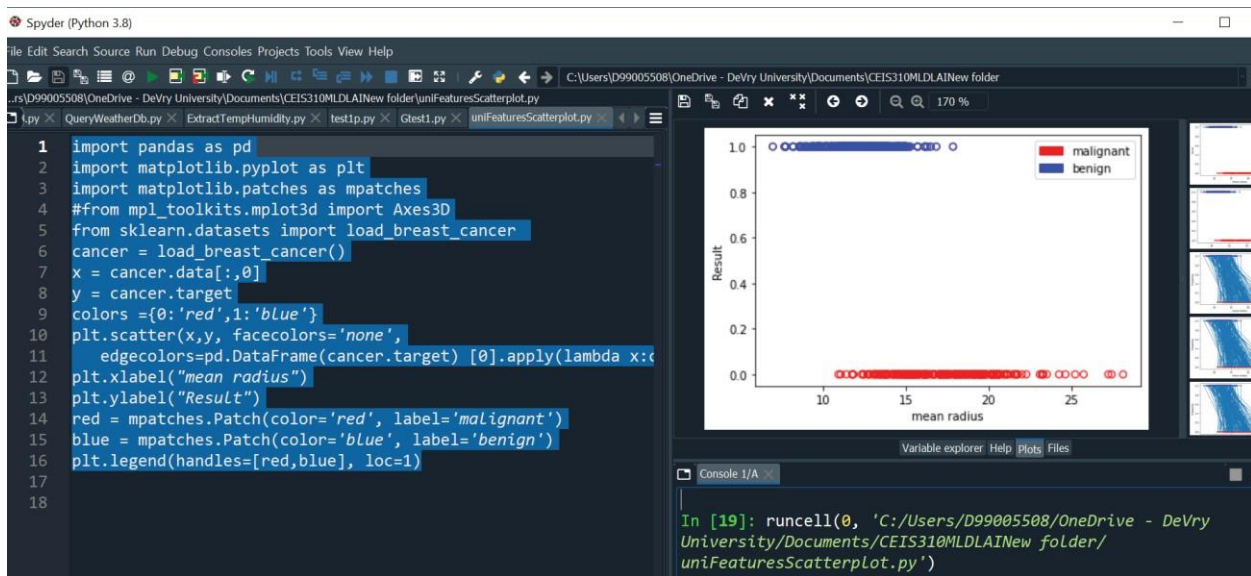


Figure 9. Plotting a scatter plot based on one feature (mean radius)

Scikit comes with the LogisticRegression class which allows us to apply logistic regression to train a model. The code given below we can use to train a model using the first feature (mean radius) of the dataset. It displays trained model's intercept as well as trained model's coefficients.

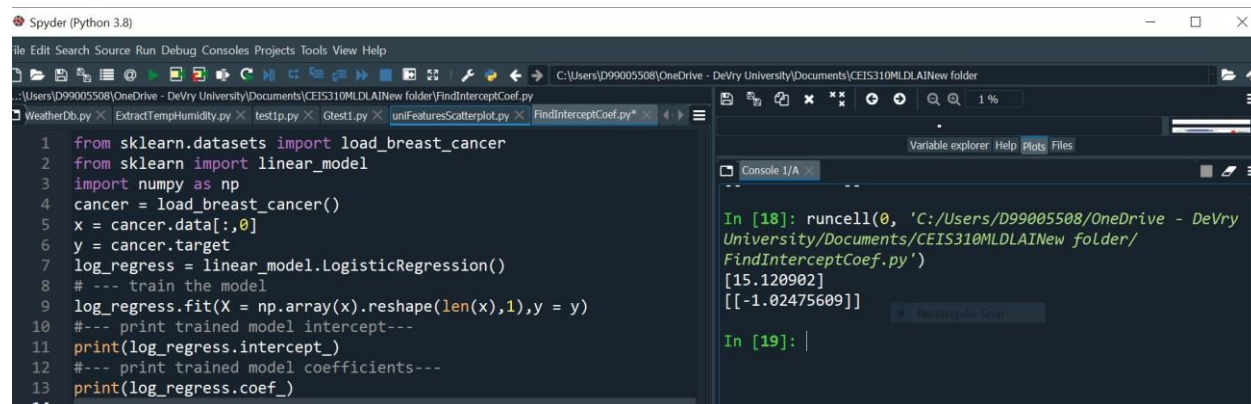


Figure 10. Display of Intercept and Coefficient using the code FindingInterceptCoef.py



Now that once the model is trained, our main interest at this point is the intercept  $\beta_0$  and the Coefficient  $x\beta$ , knowing these two values allows us to plot the sigmoid curve that tries to fit the points on the chart.

### Plotting the Sigmoid Curve:

With the values of the intercept  $\beta_0$  and the coefficients  $x\beta$  obtained we can now use the sigmoid curve `SigmoidCurveFittingTwoPoints`

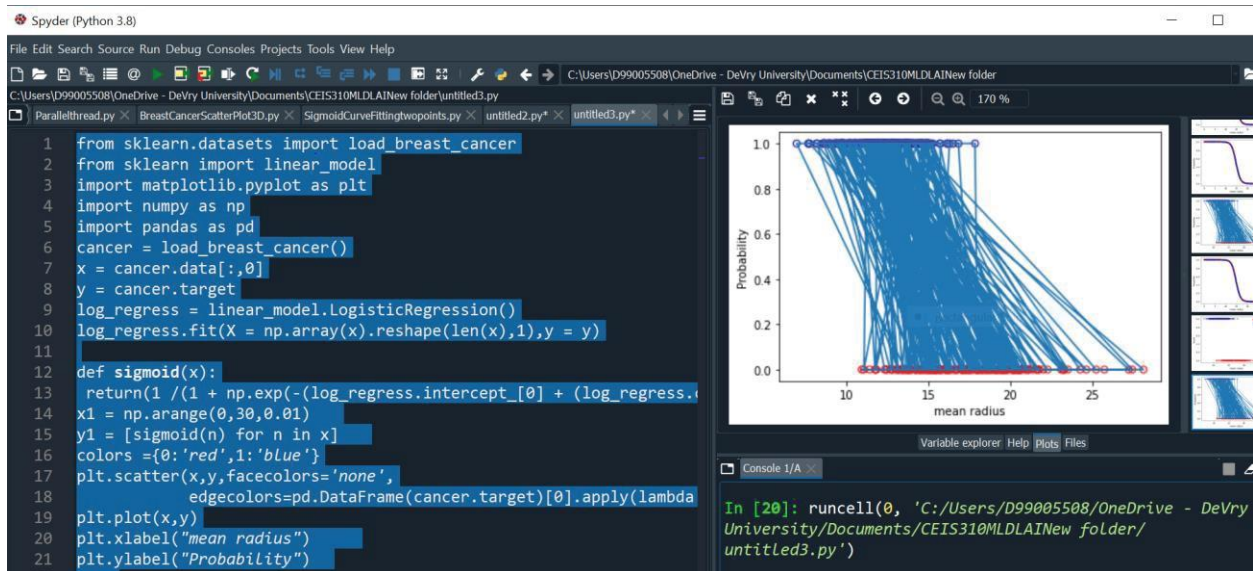


Figure 11. Sigmoid Curve Fitting to the two sets of points (substituting  $(x_1,y_1)$  with  $(x, y)$ )

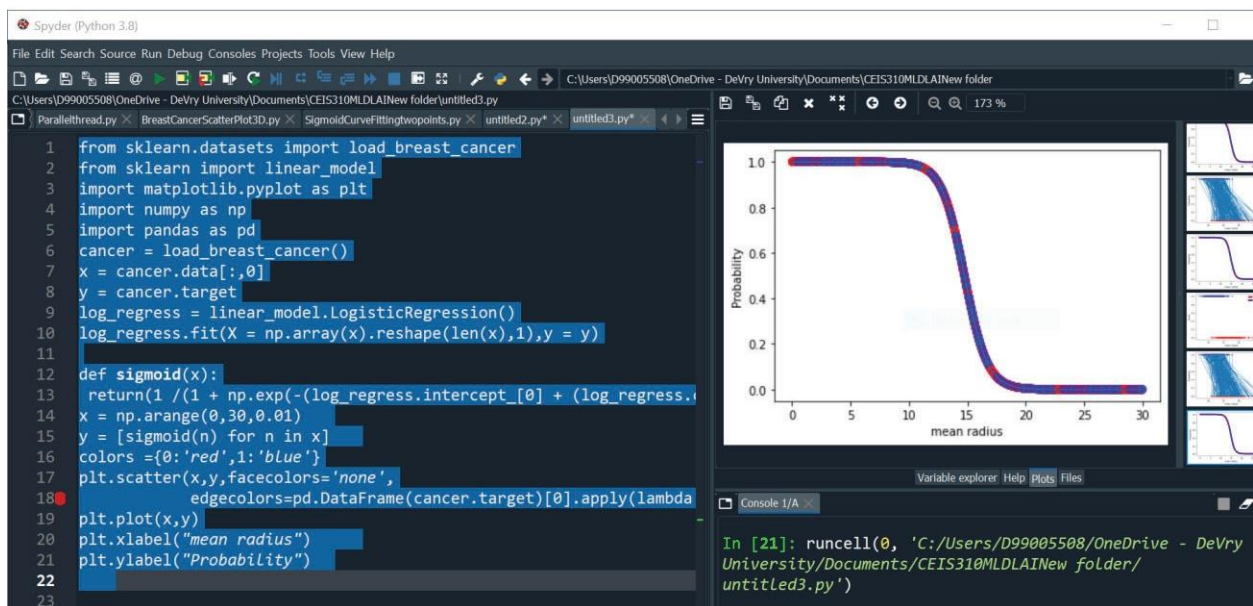
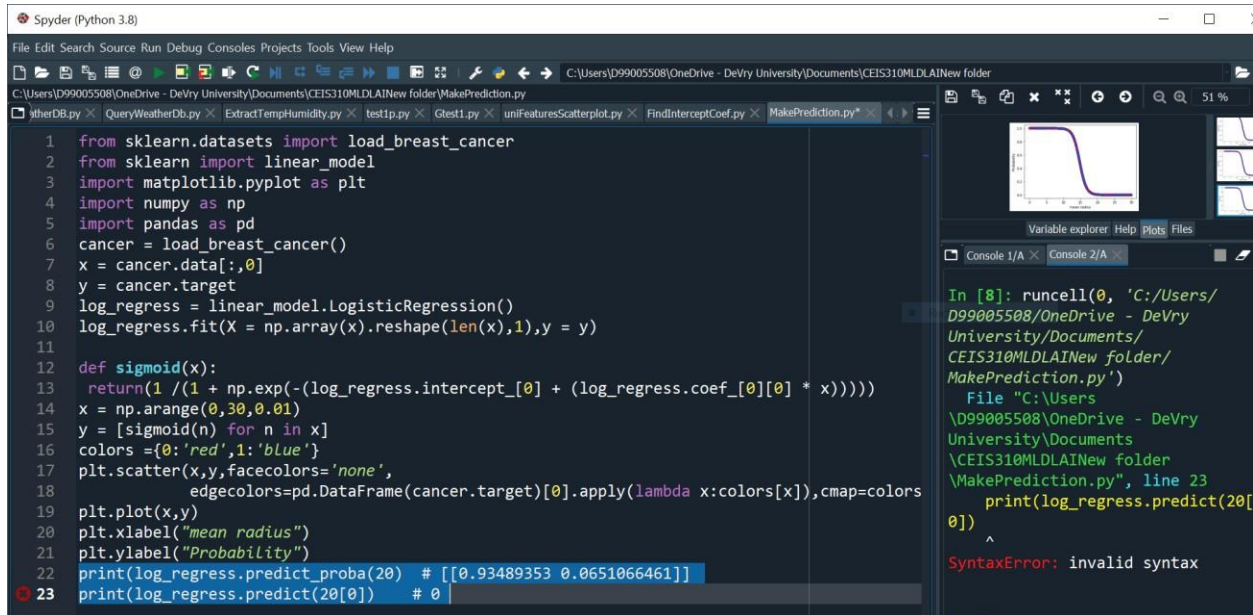


Figure 12. Sigmoid Curve Fitting to the two sets of points obtained by substituting  $(x_1, y_1)$  with  $(x, y)$

Making Predictions:

### **Student Activity V: Making Predictions using the trained Model**

Using the trained model, let us try to make some predictions. We will try to predict the result if the mean radius is 20.



```
1 from sklearn.datasets import load_breast_cancer
2 from sklearn import linear_model
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import pandas as pd
6 cancer = load_breast_cancer()
7 x = cancer.data[:,0]
8 y = cancer.target
9 log_regress = linear_model.LogisticRegression()
10 log_regress.fit(X = np.array(x).reshape(len(x),1),y = y)
11
12 def sigmoid(x):
13     return(1 / (1 + np.exp(-(log_regress.intercept_[0] + (log_regress.coef_[0][0] * x))))))
14 x = np.arange(0,30,0.01)
15 y = [sigmoid(n) for n in x]
16 colors = {0: 'red', 1: 'blue'}
17 plt.scatter(x,y,facecolors='none',
18            edgecolors=pd.DataFrame(cancer.target)[0].apply(lambda x: colors[x]), cmap=colors)
19 plt.plot(x,y)
20 plt.xlabel("mean radius")
21 plt.ylabel("Probability")
22 print(log_regress.predict_proba(20) # [[0.93489353 0.06510646]]
23 print(log_regress.predict(20[0]) # 0
```

```
In [8]: runcell(0, 'C:/Users/D99005508/OneDrive - DeVry University/Documents/CEIS310MLDLAINew folder/MakePrediction.py')
File "C:/Users/D99005508/OneDrive - DeVry University/Documents/CEIS310MLDLAINew folder/MakePrediction.py", line 23
print(log_regress.predict(20[
^
SyntaxError: invalid syntax
```

Figure 13. Machine learning Error Display

### **Prediction Result and Error Analysis:**

The **predict\_proba()** function is supposed to return a two dimensional array. It should have displayed `#[[0.93489354 0.06510646]]`

The result for example 0.93489354 indicates the probability that the prediction is 0 (malignant) while the result of 0.06510646 indicates the probability that the prediction is 1. Based on the default threshold of 0.5, the prediction is that the tumor is malignant (value of 0), since its predicted probability (0.93489354) of 0 is more than 0.5.

The **predict()** function in the (last) statement returns the class that the result. (which in this case can be 0 or 1).

The result of 0 indicates that the prediction is that the **tumor is malignant**.

Another typical example: With mean radius of 8 this time may yield:  
`print(log_regress.predict_proba(8)) # [[0.020812411 0.97917589]]`

`print(log_regress.predict (20[0])) # 1`

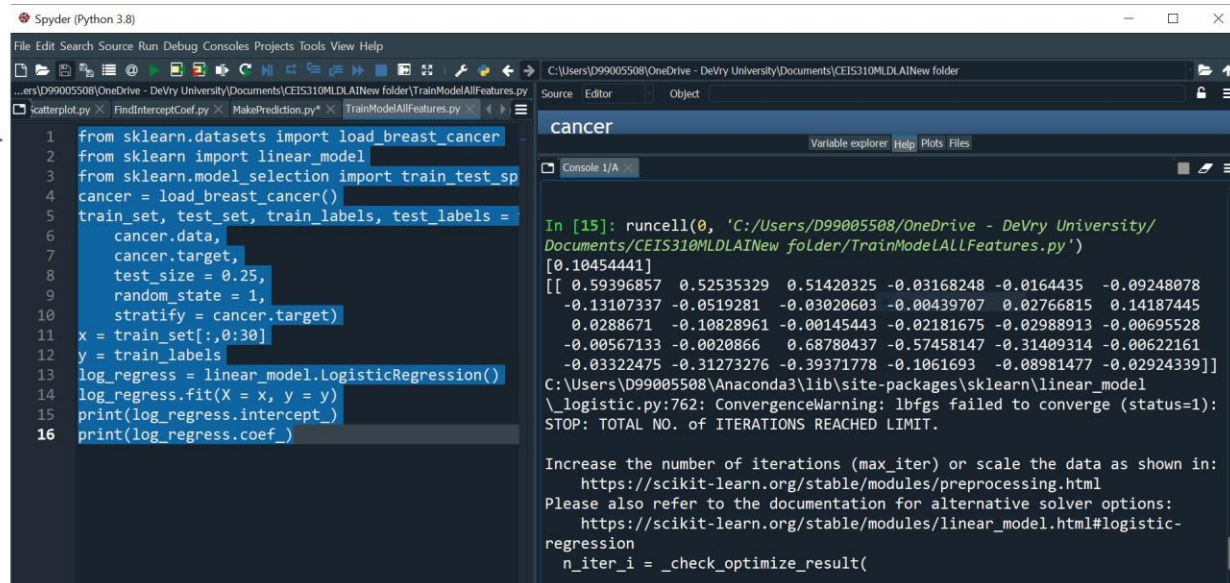
The result of 1 indicates that the prediction is that the **tumor is benign**.



## Train the Model using all Features:

Let us train the model using all of the 30 features in the data set.

When the training is done, we will print the intercept and model coefficients. Because we have trained the model using 30 features, there are 30 coefficients as depicted in Figure 14.



```
1 from sklearn.datasets import load_breast_cancer
2 from sklearn import linear_model
3 from sklearn.model_selection import train_test_split
4 cancer = load_breast_cancer()
5 train_set, test_set, train_labels, test_labels =
6 cancer.data,
7 cancer.target,
8 test_size = 0.25,
9 random_state = 1,
10 stratify = cancer.target)
11 x = train_set[:,0:30]
12 y = train_labels
13 log_regress = linear_model.LogisticRegression()
14 log_regress.fit(X = x, y = y)
15 print(log_regress.intercept_)
16 print(log_regress.coef_)
```

```
In [15]: runcell(0, 'C:/Users/D99005508/OneDrive - DeVry University/
Documents/CEIS310MLDLAINew folder/TrainModelAllFeatures.py')
[0.10454441]
[[ 0.59396857  0.52535329  0.51420325 -0.03168248 -0.0164435 -0.09248078
 -0.13107337 -0.0519281 -0.03020603 -0.00439707  0.02766815  0.14187445
  0.0288671 -0.10828961 -0.00145443 -0.02181675 -0.02988913 -0.00695528
 -0.00567133 -0.0020866  0.68780437 -0.57458147 -0.31409314 -0.00622161
 -0.03322475 -0.31273276 -0.39371778 -0.1061693 -0.08981477 -0.02924339]]
C:\Users\D99005508\Anaconda3\lib\site-packages\sklearn\linear_model
\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
  https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Figure 14. Prediction Testing the Model

## Testing the Model:

It is time to make a prediction. The code snippet uses the test set and feeds it into the model to obtain the predictions. The results of the predictions can then be printed out. It may not give a clear picture of how good the model in predicting if a tumor is cancerous. A more scientific way is to use the confusion matrix. The confusion matrix shows the number of actual and predicted labels and how many of them are classified correctly. We can use Pandas's crosstab() function to print out the confusion matrix. An interesting picture of the confusion matrix is given in the Appendix A.

## Persisting the Model:

Once you have trained, you can save it for later use. This avoids retraining the model every time and their immediate use and in making predictions.

There are two ways to save your trained model in python:

- 1) Using the standard pickle module in python to serialize and deserialize objects.
- 2) Using the joblib module in the SciKit-learn that is optimized to save and load Python objects that deal with NumPy data.

## Challenges in Machine Learning

### Hard to get started

Steps	Challenge
Access, explore and analyze data	<b>Data diversity</b> Numeric, Images, Signals, Text – not always tabular
Preprocess data	<b>Lack of domain tools</b> Filtering and feature extraction Feature selection and transformation
Train models	<b>Time consuming</b> Train several models to find the “best”
Assess model performance	<b>Avoid pitfalls</b> Over Fitting Speed-Accuracy-Complexity tradeoffs

### Conclusion:

ML-powered systems hold promise for delivering faster and more-consistent cancer diagnoses. The lack of a gold standard in cancer diagnosis is a well-known problem as such, technology has important limitations. The reliability of ML algorithms depends on the validity of the data on which they are trained. In the diagnosis of cancer, this means a reliance on data that are correctly labeled with the ground truth about what is and what is not cancer.

Researchers and physicians eager to make machine learning driven cancer diagnosis a clinical reality are concerned that the technology’s reliance on pathologists as an external standard could lead to an increase in over diagnosis. This is a observed fact that has been observed in other cancer types as well. Machine learning ascertains the diagnosis of disease that meets the pathologic definition of cancer but never would have caused morbidity or mortality in a patient’s lifetime. Like any other technology, ML needs to be adequately vetted before it is widely adopted, given the potential for unintended harms such as over diagnosis in oncological applications.

### Student Feedback

We administered an anonymous questionnaire to obtain feedback from the students in relation to the choice of using python and Matlab for machine learning. We have provided some selected questions from the questionnaire and their respective responses that apply particularly to the choice of programming language and lessons learned by early exposure to this module. Note that all the students’ comments were encouraging and positive; there were few negative responses obtained from students with no background in statistics and linear algebra using Matlab tools for machine learning..

- *Do you feel the learning ML module reinforced class material in a good, bad, or indifferent way?*

**Sample Responses:**

- ✓ Excellent. The python programs have greatly enhanced the overall class information.
- ✓ In an indifferent way. I like the computing with Matlab but it was quite cumbersome, may be because I am more comfortable with python scikit library.
- ✓ Why python, why not Octave or Matlab ? I was able to retrofit them using Matlab's Machine tool kit.
- ✓ Matlab's Classification Learner was daunting at the onset, but it let me choose from several classification types including decision trees, SVM and k-nearest neighbors.
- ✓ These python lab modules were really helpful and set the pace for the class and enhance the rich experience.

- *Has the teaching module been straight forward and easy to understand? Do you have suggestions for improvement?*

**Sample Responses:**

- ✓ Using Scikit learn for linear regression was Pretty straight forward, but logical regression based model was confusing in the beginning but felt straightforward after few iterations.
- ✓ Add a section on use of Octave for machine learning because I don't have a copy of Matlab Machine learning Toolbox

- *Did the Python software in conjunction with Matlab aid in understanding of the lab exercises?*

**Sample Responses:**

- ✓ MatLab was not that easy, I felt lost in vectors and matrices. I got my results quickly using python modules.
- ✓ Yes it did. The first few labs were OK, I had to brush up on my python lists, tuples and file input/output for retrieving files.
- ✓ Yes. The use of python was a life saver. More choices using Matlab gave a fresh different outlook, but probably not the best when we delved into matrices and vectors.

- *What other aspects do you feel would have been useful in this teaching module?*

**Sample Responses:**

- ✓ When to use `r2_score`, SVM, and k-nearest neighbors and when not to use it.
- ✓ Few more examples explaining decision trees, support vector machines and k-nearest neighbors

In summary, we can see from the above responses that students enjoyed this teaching module overall. Their responses to the choice of programming language were fairly positive, but more mixed as expected. The incorporation of many demonstrations and interactive lab experiences with python and Matlab into this teaching module proved to be effective.

I was fortunate to monitor and discuss the experiences of these students who took the senior project capstone class with my colleague last year. As kind of expected they were very positive with the outcome and obtained an enhanced understanding of data exploration, feature selection and classification algorithms, even model comparison and assessment which they attributed to their early exposure to python as well as the reinforcement of the usage of vectors, matrices and Matlab functions for statistical applications.

In conclusion, it can be stated that with proper guidance, monitoring, and diligent care, engineering technology students can be exposed earlier to manipulating tabular data using python's pandas, data visualization using matplotlib, supervised learning using linear regression, classification using logistic regression, and the basics of Matlab, and Octave (if they don't have Matlab). This will go a long way in motivating students, eliminating their fear, improving their understanding, and making them study the public cloud, Amazon Web Services (AWS), Microsoft Azure and enhancing their quality of education (Abell & Braselton, 1999).

## **Bibliography**

- Abell, M., & Braselton, J. (1999). *Maple V By Example*. Academic Press.
- Blinowska, K., & Durka, P. (1994). *The Application of Wavelet Transforms and Matching Pursuit to the Time - Varying EEG Signals, in Intelligent Engineering Systems Through Artificial Neural Netowks* (Vol. 4). (Dag, & o. & Fernandez, Eds.) New York, NY: ASME Press.
- Mishra, A. (2019). *Machine Learning in the AWS Cloud*. Sybex.
- Muqri, M., & Chang, S. (2015). MRI and Bioinstrumentation Using Matlab and Simulink, Transactions on Techniques in STEM Education. *Port Royal, 1*, 16.
- Murach, J., & Urban, M. (2016). *Python Programming*. Mike Murach & Associates.
- Shakib, J., & Muqri, M. (2010). Leverraging the Power of Java in the Enterprise. *American Society of Engineering Education*, 1701.
- UCI, Center for Machine Learning and Intelligent Systems. (2022, May 15). Retrieved from Machine Learning Repository:  
[https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(diagnostic\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic))
- Wei-Ming, L. (2019). *Python Machine Learning*. Wiley.

## Appendix A: Selected Machine Learning Algorithms

### Polynomial Regression

Polynomial Regression is used when your data points clearly will not fit a linear regression model. Polynomial regression, like linear regression, uses the relationship between the variables  $x$  and  $y$  to find the best way to draw a line through the data points.

Let us choose an example of **predicting the number of pizzas price sold based on the time of the day which is plotted using military time**. As you will notice the data set for this demo does not fit into simple linear regression, as such we will use the polynomial regression algorithm furnished by Scikit-learn machine learning Python library

We can also use the NumPy method that lets us make a polynomial model:

```
mymodel = numpy.poly1d(numpy.polyfit(x,y,3))
```

We just specify how the line will display, we start at position 1, and end at position 22:

```
myline = numpy.linspace(1,22,160)
```

```
# Performance Evaluation for Regression-based ML Models
```

```
import numpy
```

```
from scipy import stats
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import r2_score
```

```
x = [1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
```

```
y = [60,50,30,30,40,55,65,70,74,95,100,110,96,124,140,160,150,130,100]
```

```
plt.title('Pizzas Delivered plotted against Military Time')
```

```
plt.xlabel('Military Time')
```

```
plt.ylabel('Pizza Qty delivered')
```

```
mymodel = numpy.poly1d(numpy.polyfit(x,y,3))
```

```
myline = numpy.linspace(1,22,160)
```

```
plt.scatter(x,y)
```

```
plt.plot(myline,mymodel(myline))
```



```

plt.show()

print(" Coefficient of Determination:", r2_score(y,mymodel(x)))

#predict the number of Pizzas delivered at military time 17:00 hr

x = round(mymodel(17))

print("The number of Pizzas delivered at 17:00 PM: ", x)

```

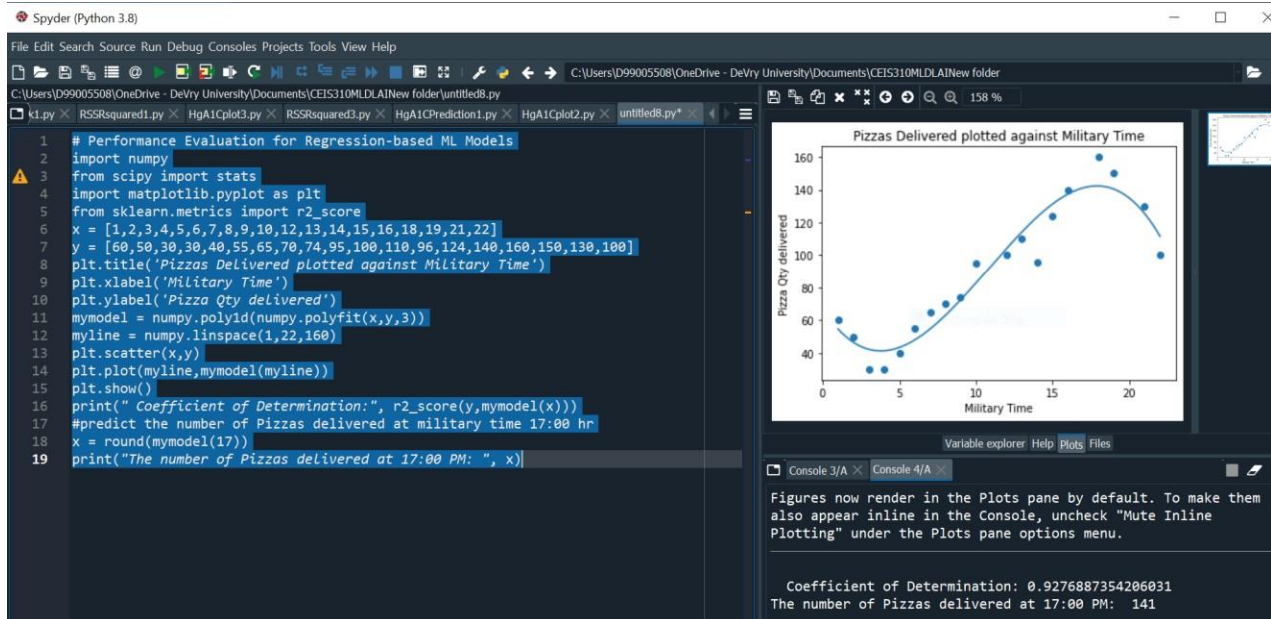


Figure1. Polynomial Regression model

## Train and Test Models:

In Machine Learning we create models to predict the outcome of certain events. For instance for a given dataset, we can predict the CO2 emission of a car when we knew the weight and engine size.

To measure if the model is good enough, we can use a method called Train/Test. Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the data set into two sets: a training set and a testing set. 80% for training and 20% for testing.

You train the model using the training set, and you test the model using the testing set. Train the model means to create the model. Test the model means test the accuracy of the model

Suppose we have a simple data set which illustrates 100 customers in a shop, and their shopping habits.

The y axis represents the amount of money spent on the purchase.

The x axis represents the number of minutes before making a purchase.

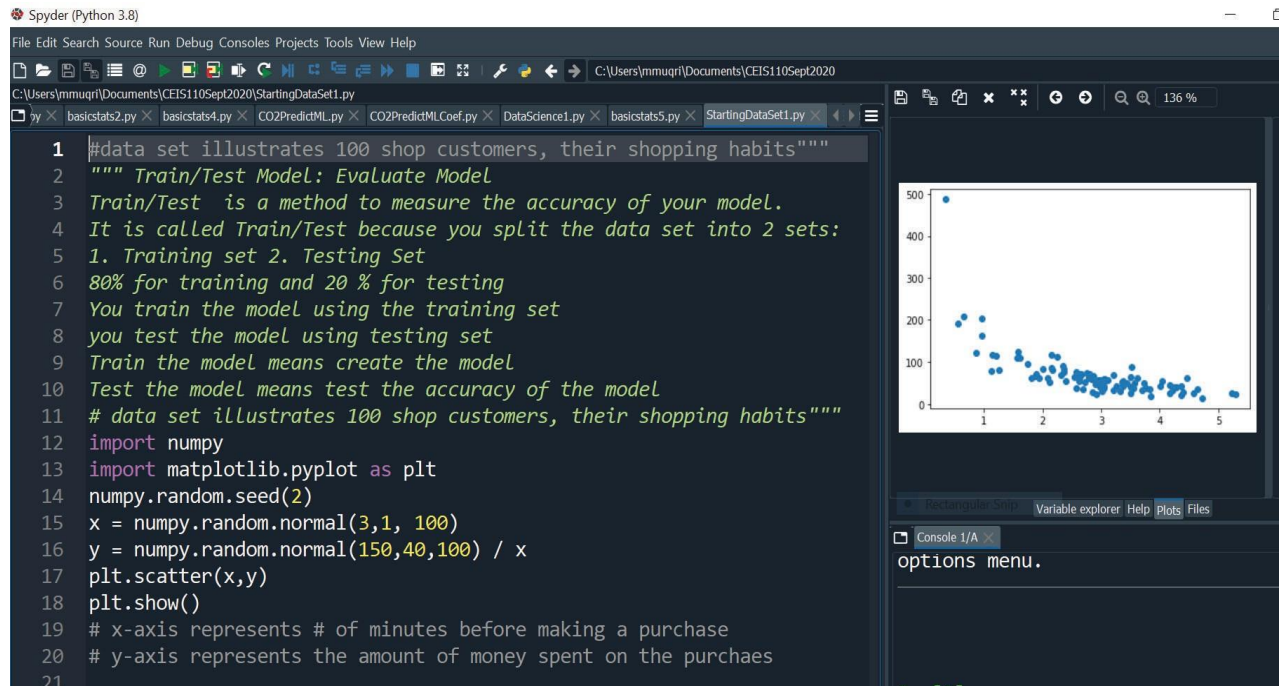


Figure 2. Test and Train Model

### Fitting the Data Set

To draw a line through the data points we use plot() method of the matplotlib module

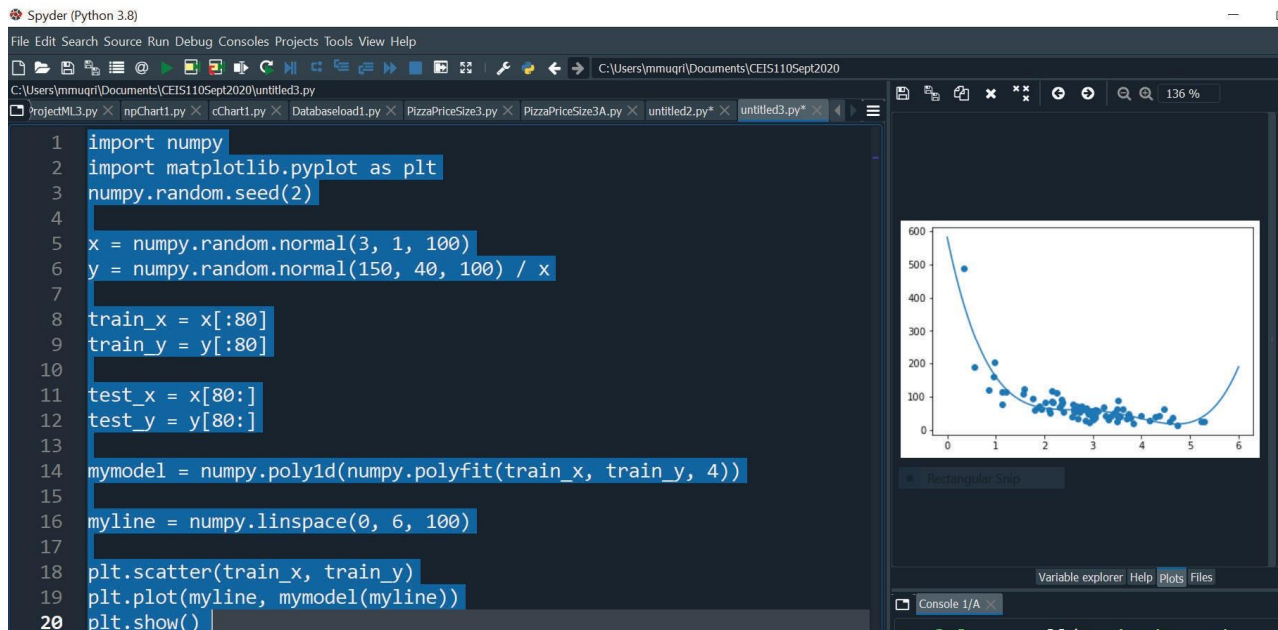


Figure 3. Polynomial regression Fit

If we look at the previous Figure carefully, we notice the display kind of suggests the data set fitting a polynomial regression.

The sklearn module has a method called `r2_score()` that will help us find this relationship.

Apparently in this case we will explore rather measure the relationship between the minutes a customer stays in the shop and how much money they spend.

Let us test how does our training data fit in a polynomial regression?

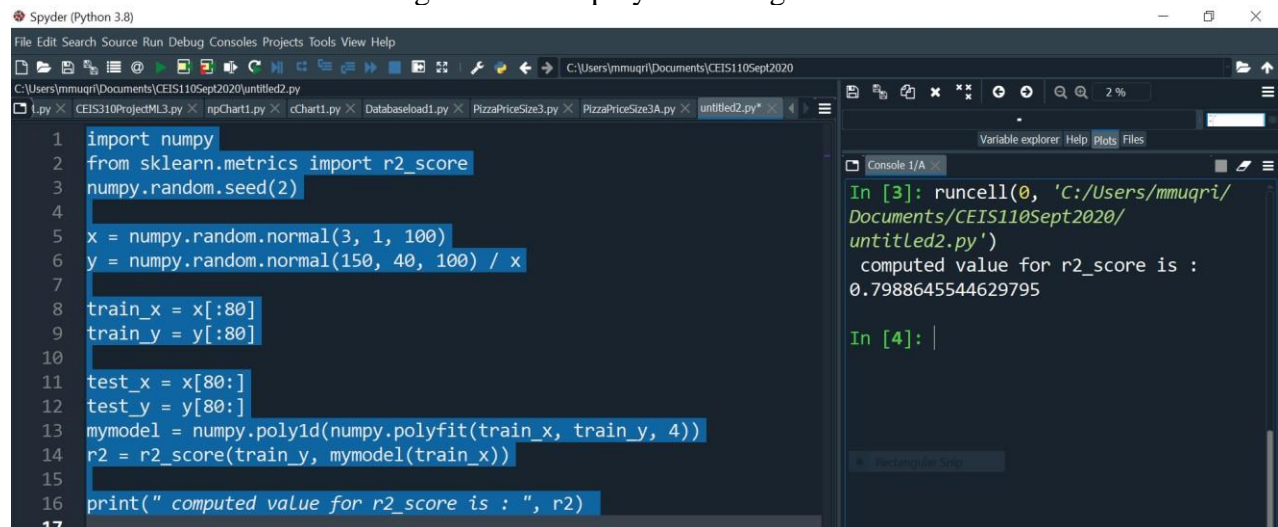
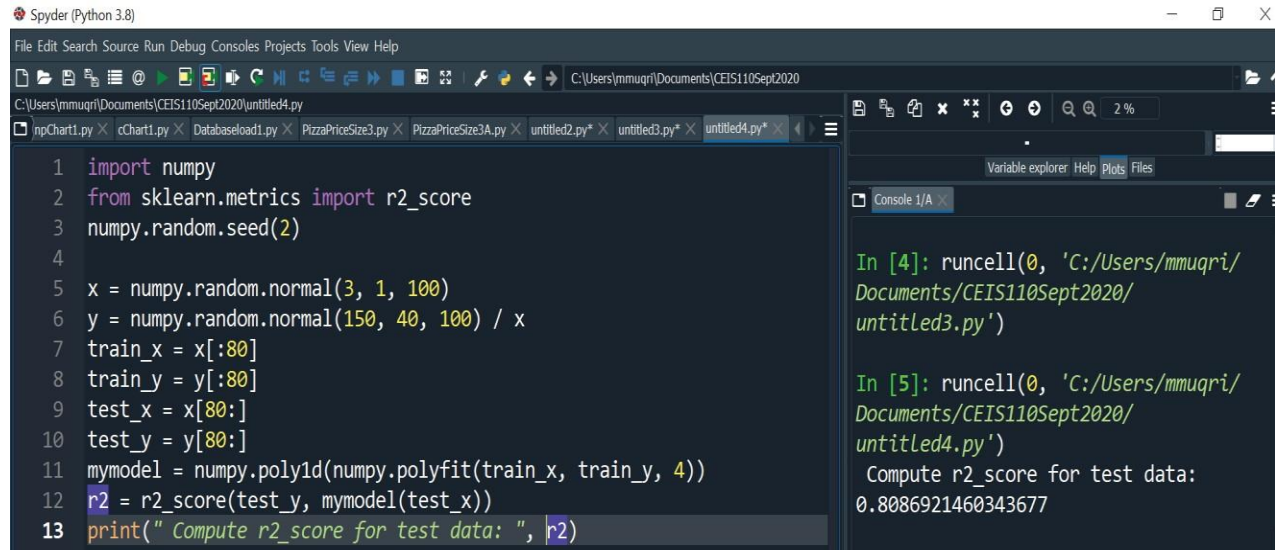


Figure 4. Train model r2 score Computations

## How well does our training data fit in a polynomial regression?

From our python program run, we get the result of 0.799 for Train Model which is not perfect but kind of shows that there is a OK relationship. It is time to evaluate our Test data set. We have made a model that is OK, at least when it comes to training data (Murach & Urban, 2016).

Now we want to test the model with the testing data as well, and check our findings to see if gives us the same result.



```
1 import numpy
2 from sklearn.metrics import r2_score
3 numpy.random.seed(2)
4
5 x = numpy.random.normal(3, 1, 100)
6 y = numpy.random.normal(150, 40, 100) / x
7 train_x = x[:80]
8 train_y = y[:80]
9 test_x = x[80:]
10 test_y = y[80:]
11 mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))
12 r2 = r2_score(test_y, mymodel(test_x))
13 print(" Compute r2_score for test data: ", r2)
```

```
In [4]: runcell(0, 'C:/Users/mmuqri/
Documents/CEIS110Sept2020/
untitled3.py')

In [5]: runcell(0, 'C:/Users/mmuqri/
Documents/CEIS110Sept2020/
untitled4.py')
Compute r2_score for test data:
0.8086921460343677
```

Figure 5. Test Data: r2 score Computations

The r2\_score result 0.80869 shows that the model fits the testing set as well, and builds our confidence that we can use this model to predict future values.

## Prediction Test:

How much money will a customer spend, if she or he stays in the shop for say 5 minutes? Our result of customer spending about \$22.88 seems to reflect our curve relationship.

```

1 import numpy
2 from sklearn.metrics import r2_score
3 numpy.random.seed(2)
4
5 x = numpy.random.normal(3, 1, 100)
6 y = numpy.random.normal(150, 40, 100) / x
7 train_x = x[:80]
8 train_y = y[:80]
9 test_x = x[80:]
10 test_y = y[80:]
11 mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))
12 r2 = r2_score(test_y, mymodel(test_x))
13 print(" Compute r2_score for test data: ", r2)
14 print ("Amount of money Customer will spend: ", mymodel(5))

```

```

In [7]: runcell(0, 'C:/Users/mmuqri/
Documents/CEIS110Sept2020/
untitled4.py')
Compute r2_score for test data:
0.8086921460343677
Amount of money Customer will spend:
22.87962591812061

In [8]:

```

Figure 6. Customer Spending if she or he stays in the shop for say 5 minutes

### Matlab Machine Learning Tools:

The Classification Learner app lets you train models to classify data using supervised machine learning. Using classification learner , one can perform machine learning tasks such as interactively exploring your data, selecting features, specifying validation schemes, training models and assessing results.

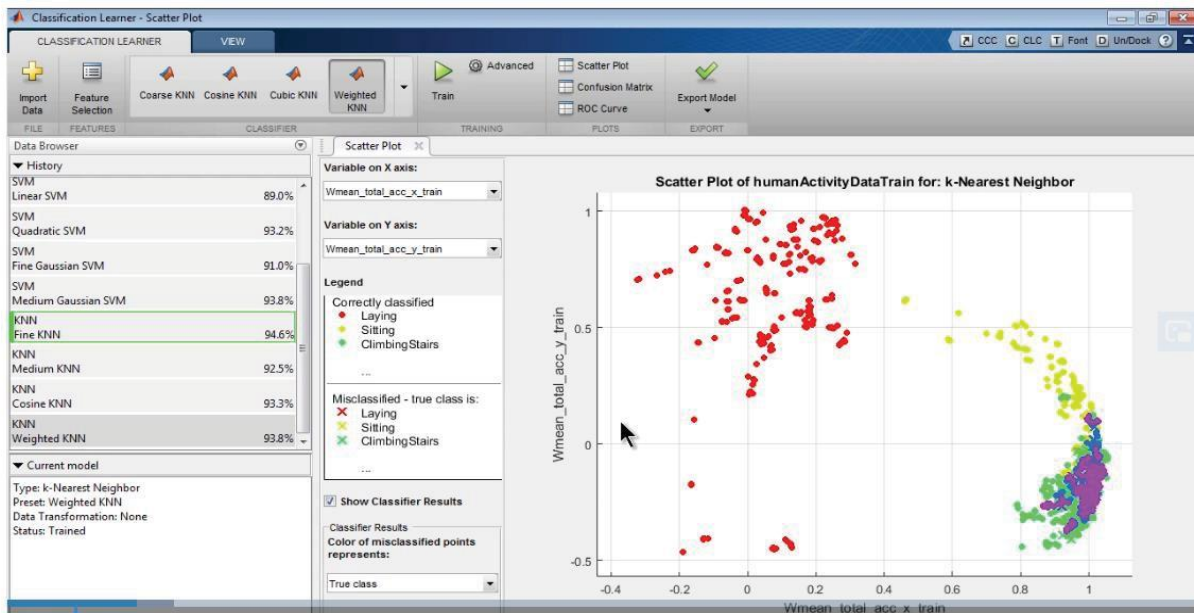


Figure 7. Classification Learner



One can choose from several classification types including decision trees, support vector machines(SVM), and k-nearest neighbors, and select from ensemble methods such as bagging, boosting and random subspaces just to name a few.

Classification learner helps one choose the best model for the data by letting one perform model assessment and model comparisons using confusion matrices and ROC curves as shown in figure below.

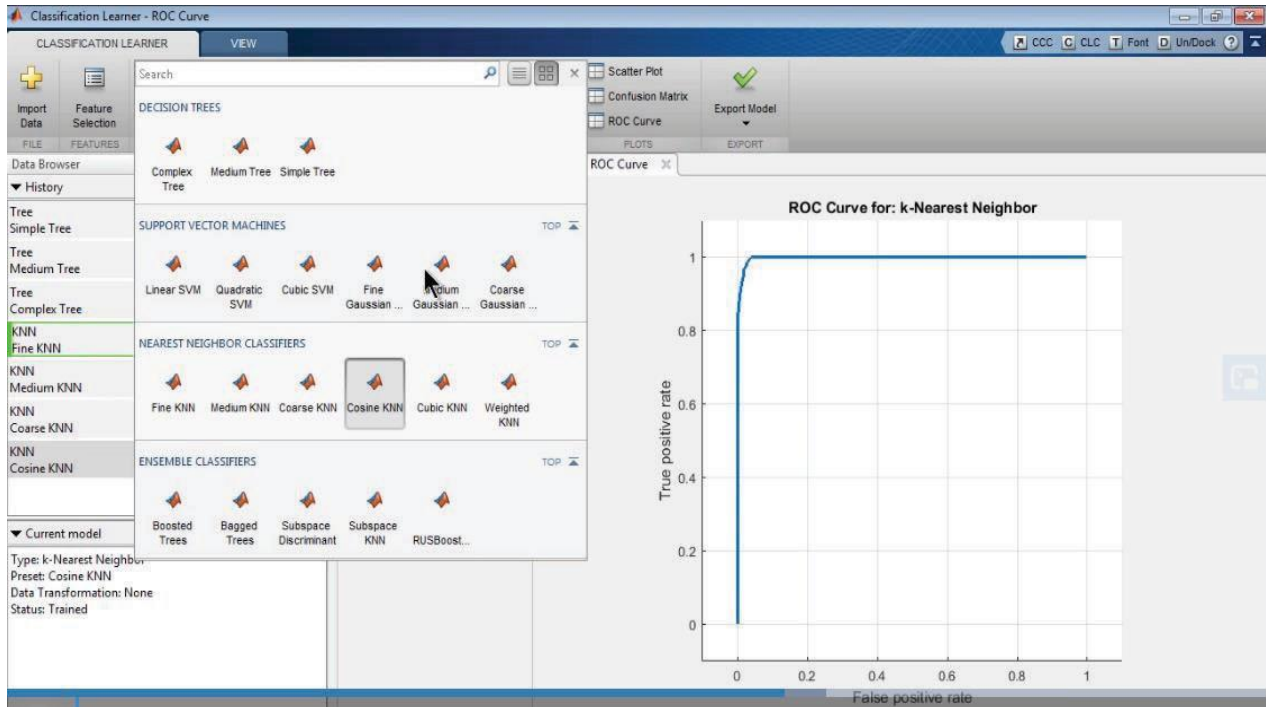


Figure 8. An ROC curve for k-Nearest Neighbor

The *confusion matrix* displays the total number of observations in each cell. The rows of the *confusion matrix* correspond to the true class, and the columns.

**An example of confusion matrix using Matlab is also shown below:**

	<b>Cat</b>	<b>42</b>	<b>6</b>	<b>9</b>
	<b>Dog</b>	<b>5</b>	<b>31</b>	<b>9</b>
<b>True Class</b>	<b>Penguin</b>	<b>7</b>	<b>1</b>	<b>41</b>
		<b>Cat</b>	<b>Dog</b>	<b>Penguin</b>
		<b>Predicted Class</b>		

Figure 9. Depiction of typical Confusion Matrix