This assignment consists of four questions. The objective of the assignment is to use different machine learning algorithms. All the algorithms used were from Scikit and nltk packages. The steps used to solve question one and two fit to the following general procedures.

* Import a model. The general form of importing a model is:

**From sklearn.family** import **Model**

Example: **from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis

The imported models are our estimators.

* The second step is to instantiate the estimator models

Example: model = SVC (kernel=**"linear"**)

* The third step is to import the data we want to analyze. For most of the questions we used scikit built in data set.

**from** sklearn **import** datasets

**Example** cancer = datasets.load\_breast\_cancer()

* If the question is related to supervised learning, then we identified and labeled the independent and dependent variables
* The fourth step was to split the data into training and testing set and fit the training data into the model

**from** sklearn.cross\_validation **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size = 0.2)

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

* The final step was predicting the accuracy of our model using our test data

**from** sklearn **import** metrics

y\_pred = model.predict(X\_test)   
print(metrics.accuracy\_score(y\_test, y\_pred))

**Question 1**

**Work flow:**

* The sklearn libraries are imported for evaluation.
* The iris data set is loaded and set it into the variable “iris”.
* The variable “x” is set as the storing feature matrix and the variable “y” is set as the response vector for the variable “iris”.
* The data is split into x and y training and testing sets using the train\_test\_split() expression. Both the “x” and “y” variables are set and the test\_size is set to 0.2 and the data is split into 20% testing and 80% training.
* The LinearDiscriminantAnalysis() expression is used to set the number of components for classification for the model. The x and y training data is fit into that model using the fit() expression.
* To predict the value of the dependent variable, the testing data is used with the model.predict() function and set it in a variable called “y\_pred”.
* The accuracy of using linear discriminant analysis is output using the metrics.accuracy\_score() expression and the y testing and y predicting variables, “y\_test” and “y\_pred”.

**Logistic Regression vs Discriminant Analysis**

Logistic Regression is used to find the probability of success and failure. In logistic regression, the dependent variable is binary, the data is coded as 1 or 0. Logistic Regression measures the relationship between the dependent variable and the one or more independent variables. Discriminant Analysis is used to predict the value of the dependent variable given values of the independent variable. In Discriminant Analysis, the dependent variable is divided into a number of categories.

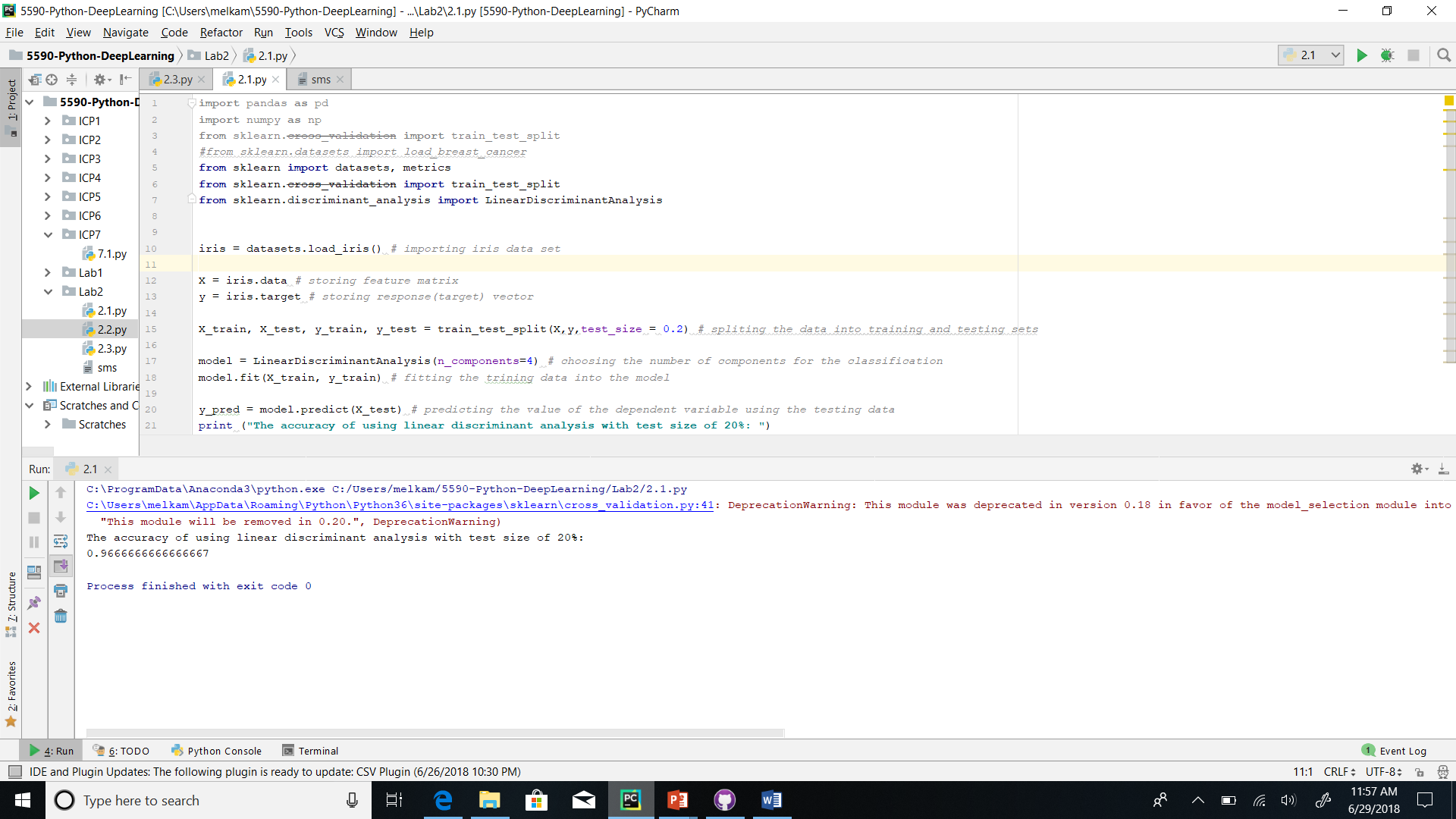
Sources:

<https://www.medcalc.org/manual/logistic_regression.php>

<http://www.statisticssolutions.com/what-is-logistic-regression/>

<http://www.statisticssolutions.com/discriminant-analysis/>

<https://www.researchoptimus.com/article/what-is-descriminant-analysis.php>

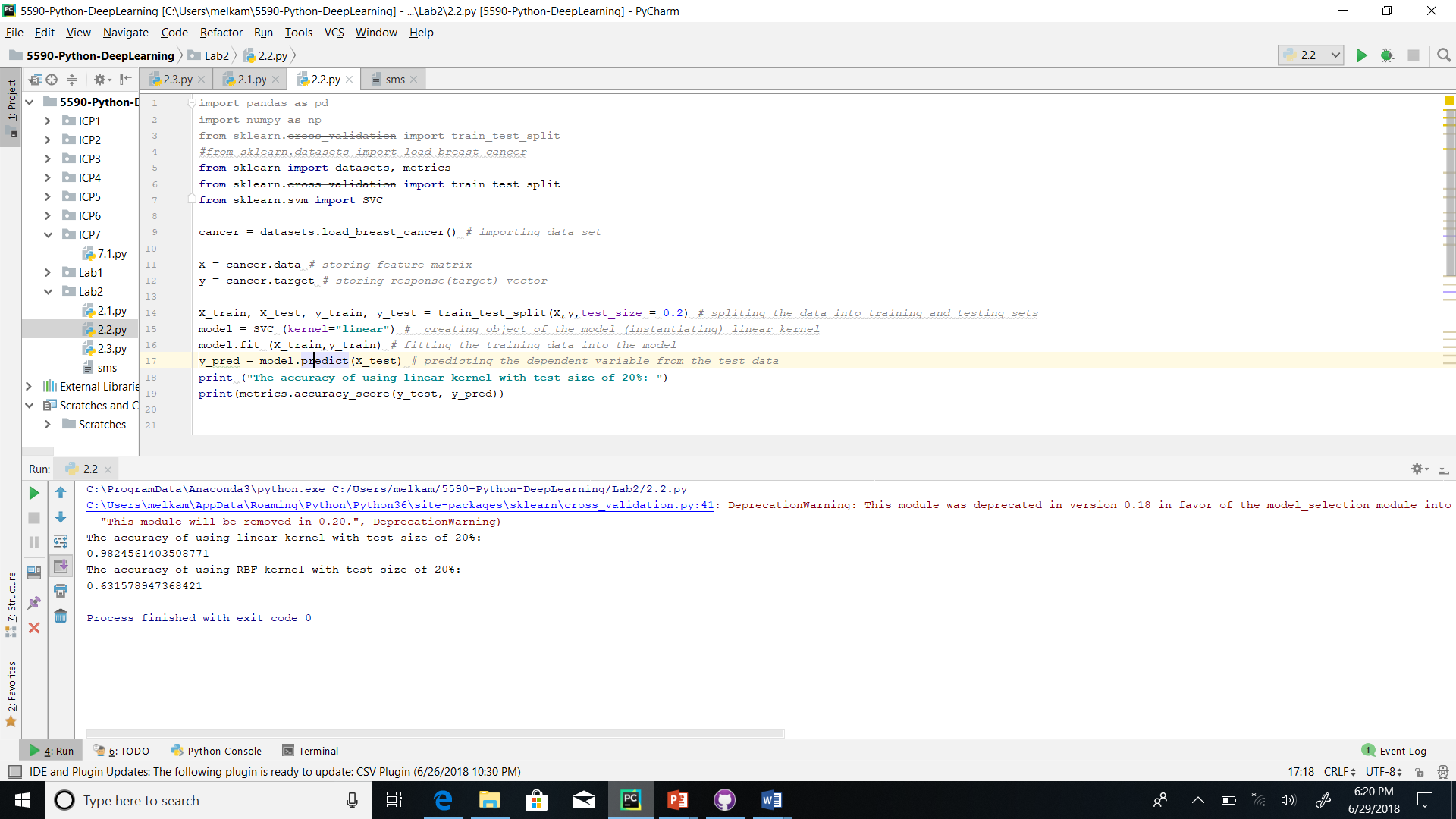


**Question 2**

**Work flow:**

* The sklearn libraries are imported for evaluation.
* The breast cancer data set is loaded and set it into the variable “cancer”.
* The variable “x” is set as the storing feature matrix and the variable “y” is set as the response vector for the variable “cancer”.
* The data is split into x and y training and testing sets using the train\_test\_split() expression. Both the “x” and “y” variables are used and the test\_size is set to 0.2 to split the data into 20% testing and 80% training.
* For the Linear kernel, an object of the model is created using the SVC() expression and the kernel is set to linear. The training data is fit into the model.
* The predicted value of the dependent variable is found using the testing data and is set to the variable “y\_pred”.
* The accuracy of the linear kernel is output using the metrics.accuaracy.score() function using the “y” testing and “y” predicting variables, “y\_test” and “y\_pred”.
* For the RBF kernel, an object of the model is created using the SVC() expression and the kernel is set to “rbf”. The training data is fit into the model.
* The predicted value of the dependent variable is found using the testing data and the result is set to the variable “y\_pred”.
* The accuracy of the RBF kernel is output using the metrics.accuaracy.score() function using the “y” testing and “y” predicting variables, “y\_test” and “y\_pred”.

**Screen Shot:**



**Question 3**

**Work flow:**

* The Natural Language Toolkit (nltk) libraries are imported for evaluation.
* A text file that the sentences will be read from is opened. Using the open() expression, the file is opened and set it as the variable “myfile”.
* Using a variable “input\_txt” the variable “myfile” is read. The “input\_txt” variable is split, storing each word from the text file as a list. The list is set to the variable “words”.
* To apply lemmatization, the WordNetLemmatizer() function from the nltk library is used. The function is set the variable “lemmatizer”.
* An empty list is set to store the lemmatized words is initialized and set to the variable “lemma”.
* A for loop is used to go through the elements in the list “words”. Inside the for loop, the elements in the list are lemmatized and appended to the “lemma” list. This is done using the append() function, the “lemmitizer” variable, and the lemmatize() function.
* After the for loop ends, the resulting “lemma” list is output.
* To apply Bigram, the ngrams() function from the nltk library is used. The ngrams() function, the “words” list, and an integer value of two is used and is set into the variable “bigrams”. The integer value is used to set the number of items.
* An empty list is initialized to store the list of tuples of two elements. A for loop is used to go thorough the elements in the list “bigrams” to append every tuple into the “bi” list.
* The resulting “bi” list is output after the for loop ends.
* To calculate the bigram frequency, an empty dictionary called “dic” is initialized.
* A for loop is used to go through every element in the “bi” list. Inside that for loop, another for loop in range(0, 2) is used to count the number of times a word is found in the bigram, then appended into the dictionary “dic”.
* After the nested for loop ends, the resulting dictionary is output.
* To find the top five bigrams repeated the most, an empty list called “lis2” is initialized. This list is used to store the tuple of the dictionary’s values and keys.
* A for loop for both the key and values in “dic.items()” is used the append each into “lis2” in the format (value, key).
* The resulting list is then sorted in increasing order. The sorted list is split from the element in the -1 index ending at the element in the -5 index and set into a variable named “top5\_bigrams”.
* The resulting list of tuples is output.
* To concatenate the sentences with the most repeated bigrams, a variable “sentences” is used to set the sentence tokenizing of the “input\_txt” variable.
* An empty string variable called “fin\_sent” is initialized.
* A for loop is used to go through each of the sentences in the variable “sentences”. Inside that for loop another for loop is used to go thorough each of the elements in the “top5\_bigrams” tuple list.
* If any of the words from the tuple are found in the tokenized sentences, it is added to the end of the empty string and an empty space is added on the end. The resulting string is output.

Screen shot

