COMP 4448: Data Science Tools II Assignment 5

Sean Wendlandt

**Directions:** Do this assignment in Jupyter Notebook and provide screenshots of code and output in this word document wherever required. You will upload this word document containing screenshots of code and answers as well as your Jupyter Notebook to Canvas. All assignments will be submitted and graded through canvas and grades will be transferred to the 2U platform.

**Goal:** The goal of this assignment is to give you the opportunity to implement the Naïve Bayes Algorithm from scratch as well as using tools built into sklearn.

**Packages:** Core packages you may need for this assignment include numpy, pandas, sklearn, matplotlib.pyplot and/or seaborn, nltk, string, and re.

Note: In sklearn, there are different types of Naive Bayes constructors for fitting Naïve Bayes models, dependent on the nature of the data. For example:

* **MultinomialNB()** is used for text classification when data is represented as feature vectors.
* **ComplementNB()** is an adaptation of the standard MultinomialNB() for imbalance data.
* **GaussianNB()** is used if the features are assumed numerical and are assumed to follow a Gaussian or normal distribution.
* **BernoulliNB()** is used when each feature follows a Bernoulli distribution. That is, the data or all features are binary with values 0 or 1.
* **CategoricalNB()** is used when each feature has its own categorical distribution.

(<https://scikit-learn.org/stable/modules/naive_bayes.html>)

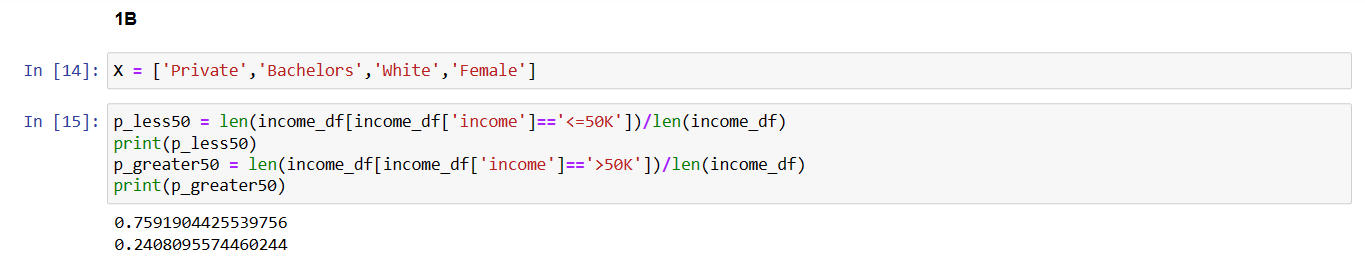
Question 1:

1. Upload the income\_evaluation\_cat.csv provided on canvas. The features in this data include **workclass, education, race, and gender.** The output variable is **income** and contains two categorical values (<=50k or >50k) indicating whether the income of an individual is less than/equal to $50,000 or greater than $50,000 respectively. Print the unique values of each variable in this data.

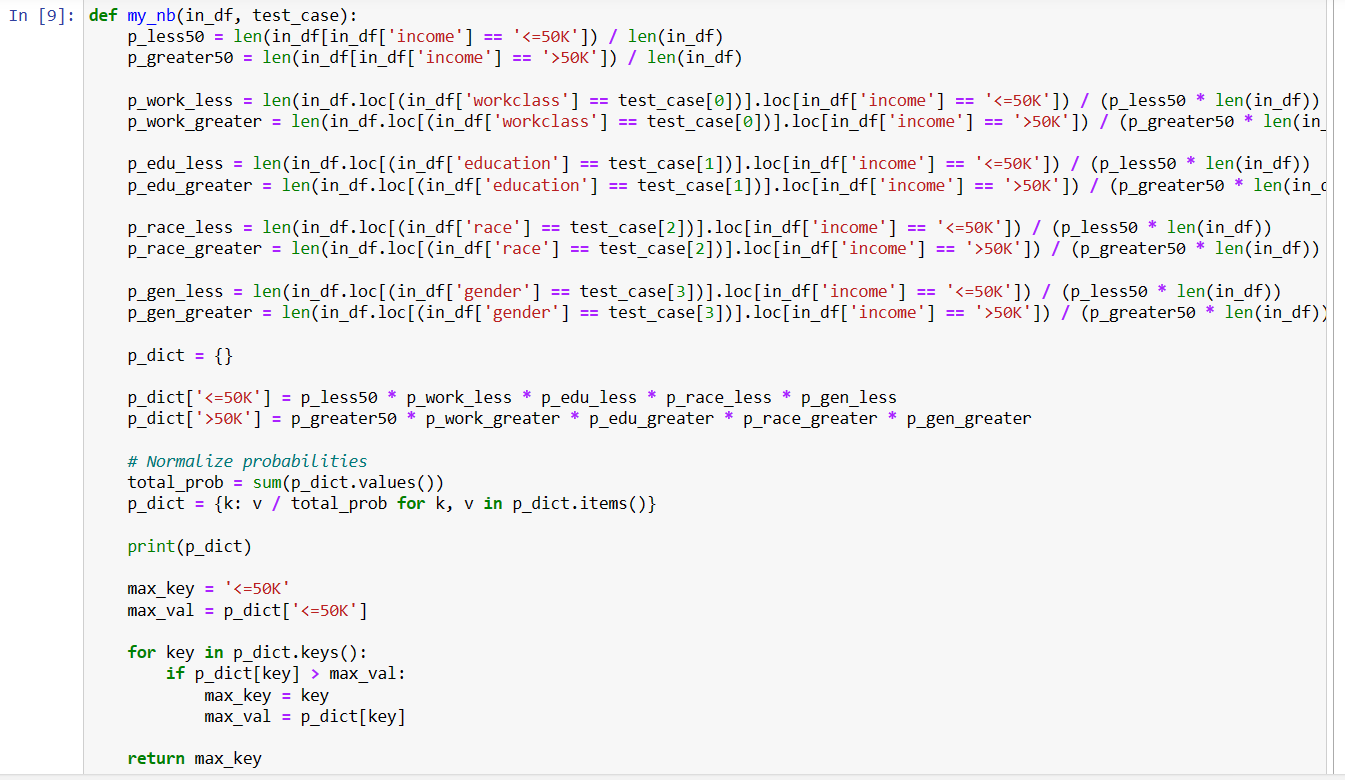


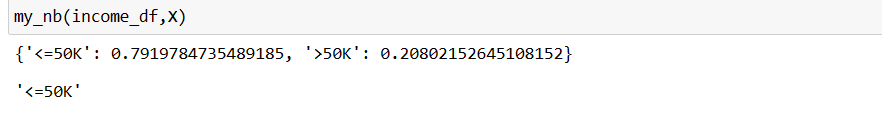
1. You will implement Naïve Bayes from scratch using Bayes’ rule. You can do your calculations in Python, but you would not use the sklearn package. Suppose that all the income\_evaluation\_cat.csv data you uploaded is the training data , classify a test instance, **X = [“Private”, “Bachelors”, “White”, “Female”]** into the class **income<=50** or **income>50k.** Compute prior and posterior probabilities that are needed to classify the test instance.

* What are the prior probabilities, **P(income<=50)** and **P(income>50k)?**

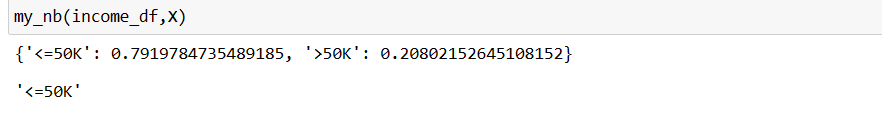


* What are the posterior probabilities, **P(income<=50/X)** and **P(income>50k/X)?**

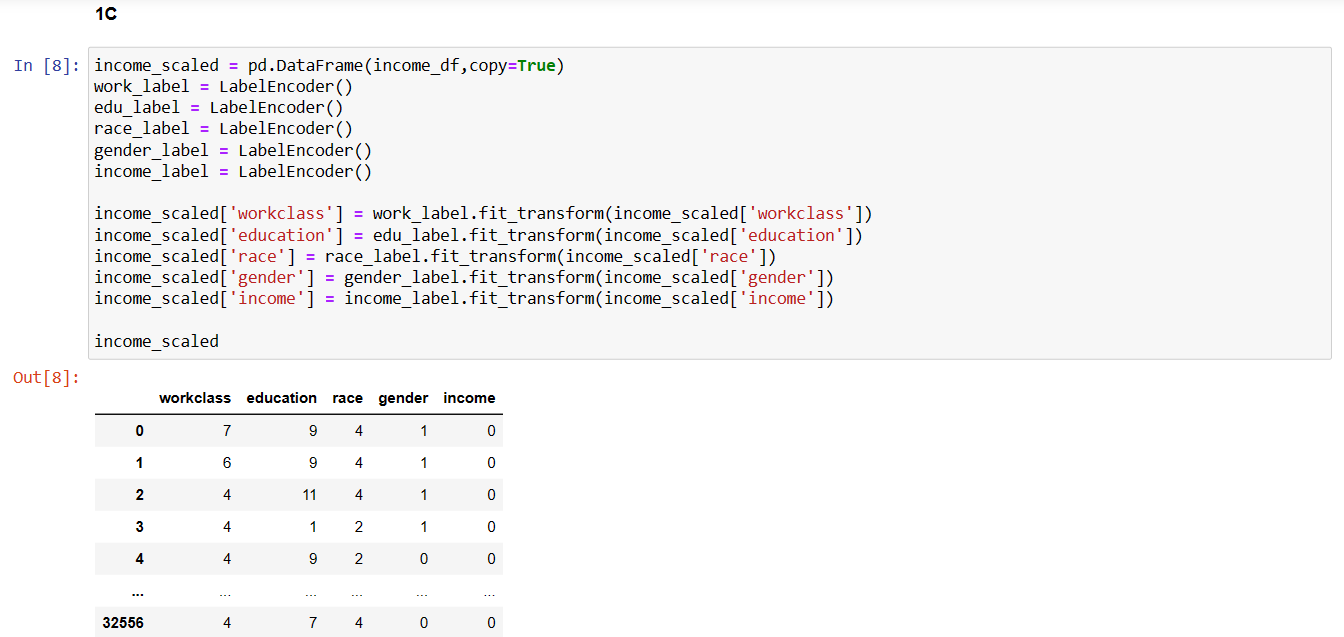




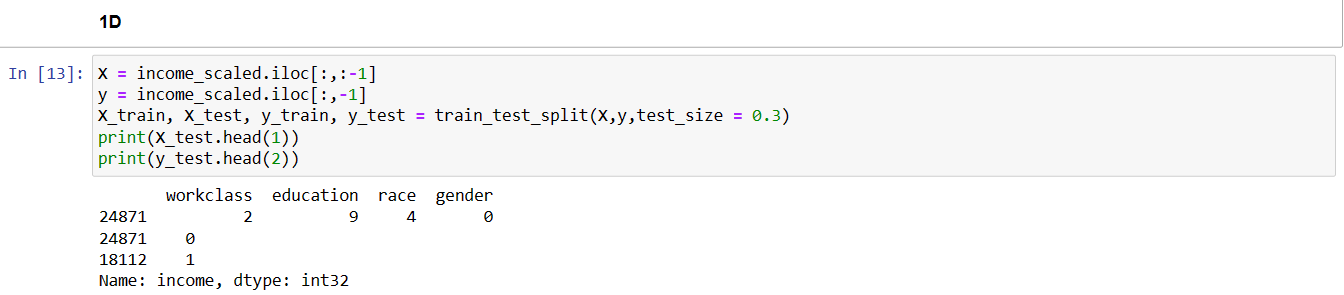
* Based on your posterior probabilities, which class should the test instance be classified into?



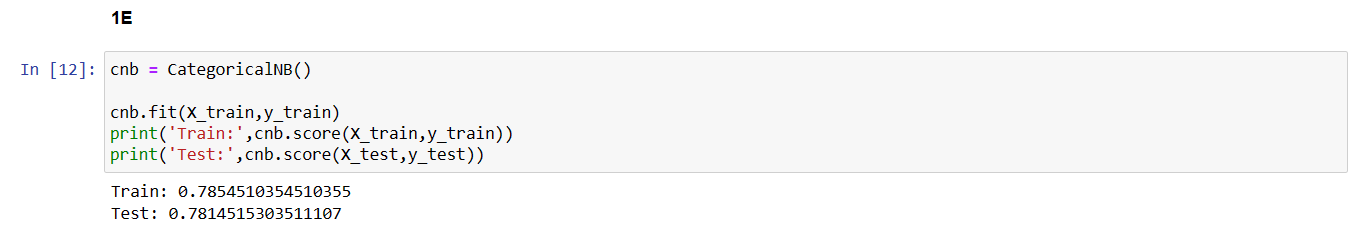
1. Preprocess or transform the features in the income\_evaluation\_cat.csv data using an appropriate scaler in sklearn. You don’t need to transform the output variable; it should still work fine in a text format.



1. Randomly split the transformed input data and the output data into X\_train, y\_train, X\_test and y\_test using tools in sklearn.

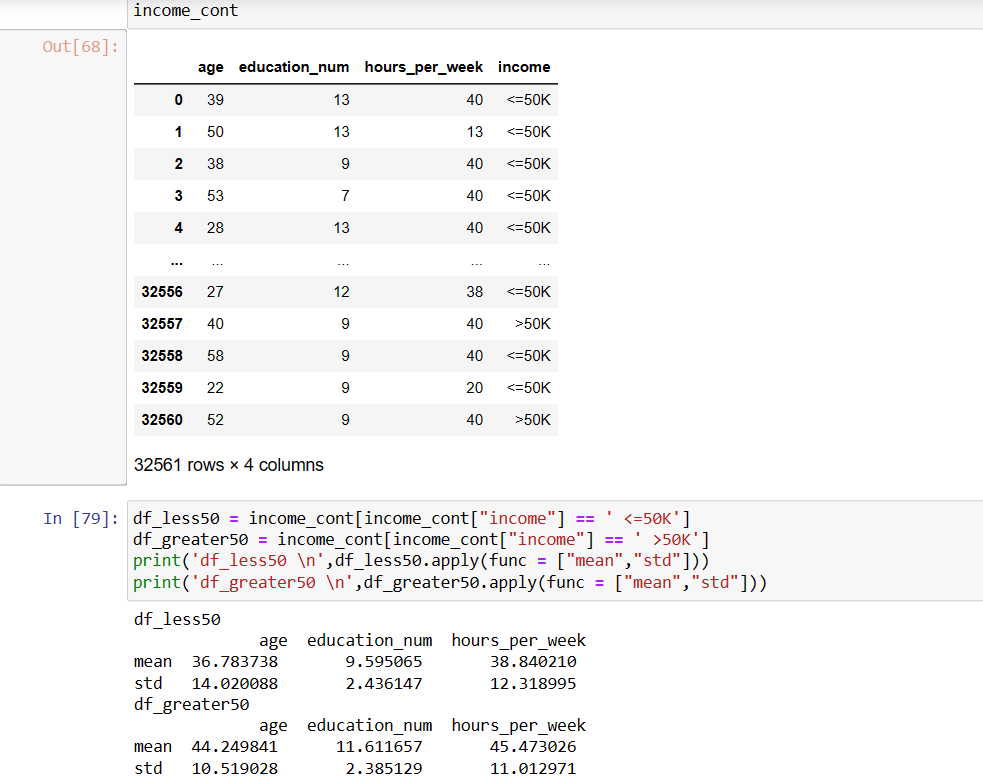


1. Use an appropriate Naïve Bayes constructor in sklearn to construct and fit a Naïve Bayes model on the training data, then use the model to compute the accuracy score of the training and test set.



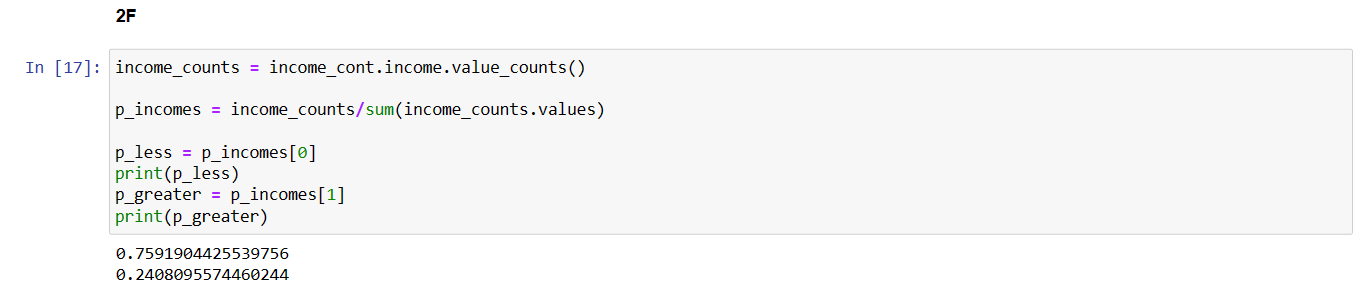
**Question 2**

1. Upload the income\_evaluation\_continuous.csv data provided on canvas. The features in this data include **age, education\_num,** and **hours\_per\_week.** The output variable is **income** and contains two categorical values (<=50k or >50k) indicating whether the income of an individual is less than/equal to $50,000 or greater than $50,000 respectively. Compute the mean and standard deviation of each input variable such that the results are presented on the same table or data frame. You can call the **.apply()** function on the pandas DataFrame.



1. You will implement Naïve Bayes from scratch using Bayes’ rule. Assume that all the features or input variables follow a normal distribution. You can do your calculations in Python, but you would not use the sklearn package. You can use the density function inside the stats module in the SciPy package. Given that all the income\_evaluation\_continuous.csv data you uploaded is the training data, classify a test instance, **X = [30, 10, 45]**,into the class **income<=50** or **income>50k.** Compute prior and posterior probabilities that are needed to classify the test instance.

* What are the prior probabilities, **P(income<=50)** and **P(income>50k)?**



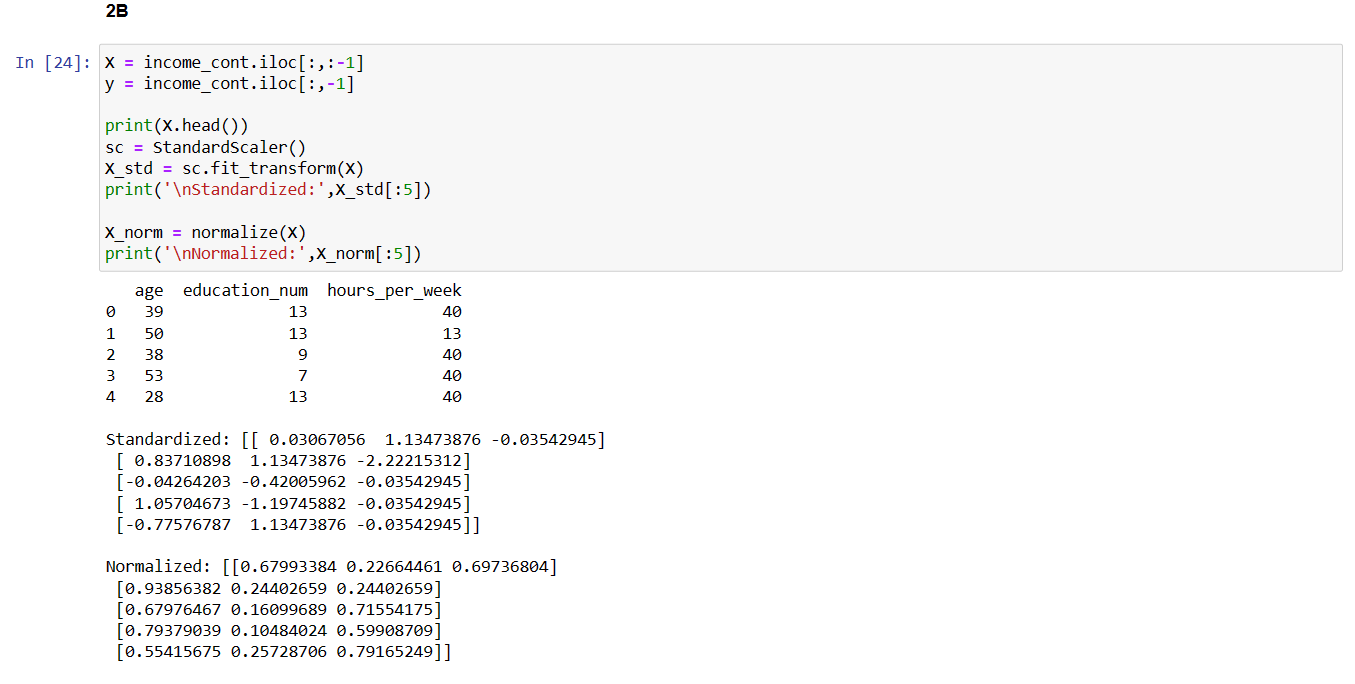
* What are the posterior probabilities, **P(income<=50/X)** and **P(income>50k/X)?**



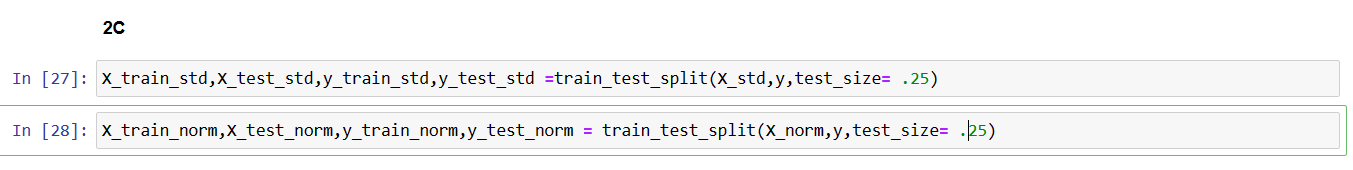
* Based on your posterior probabilities, which class should the test instance be classified into?

See screenshot above^^^ (It’s ‘<=50K’)

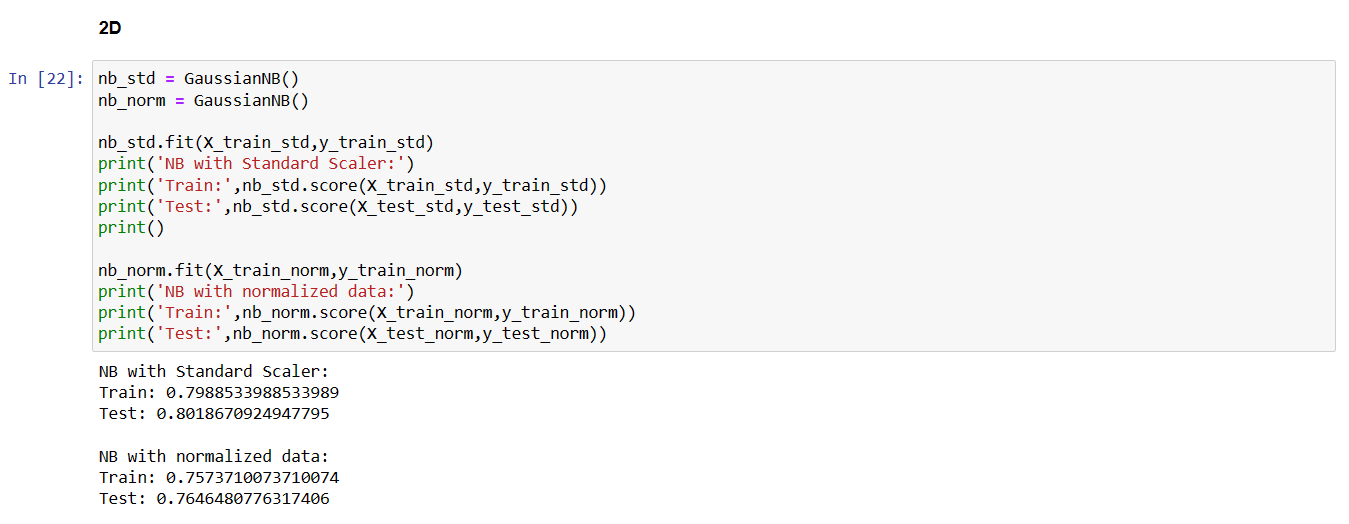
1. Preprocess or transform the features in the income\_evaluation\_cont.csv data using an appropriate scaler in sklearn. You don’t need to transform the output variable; it should still work fine in a text format.



1. Randomly split the input and output data into X\_train, y\_train, X\_test and y\_test using tools in sklearn.



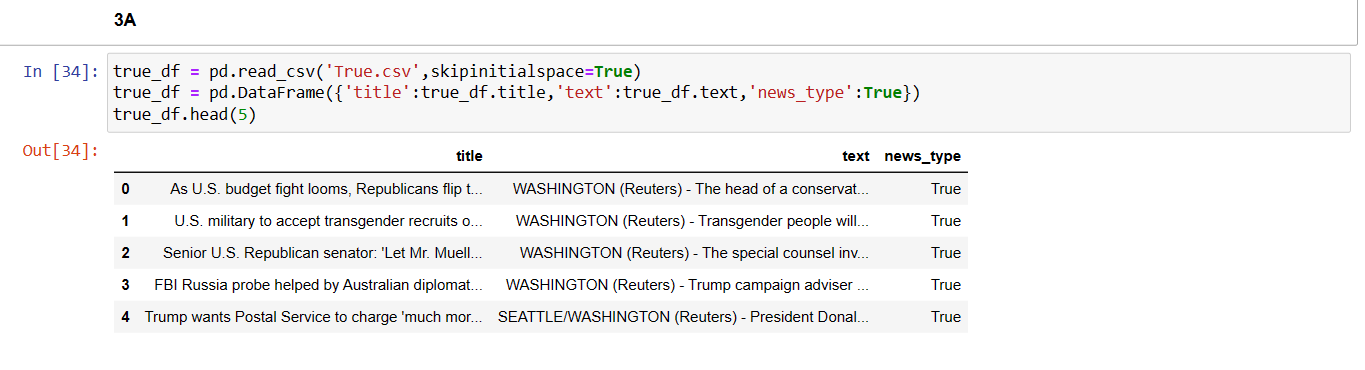
1. Use an appropriate Naïve Bayes constructor in sklearn to construct and fit a Naïve Bayes model on the training data, then use the model to compute the accuracy score of the training and test set.



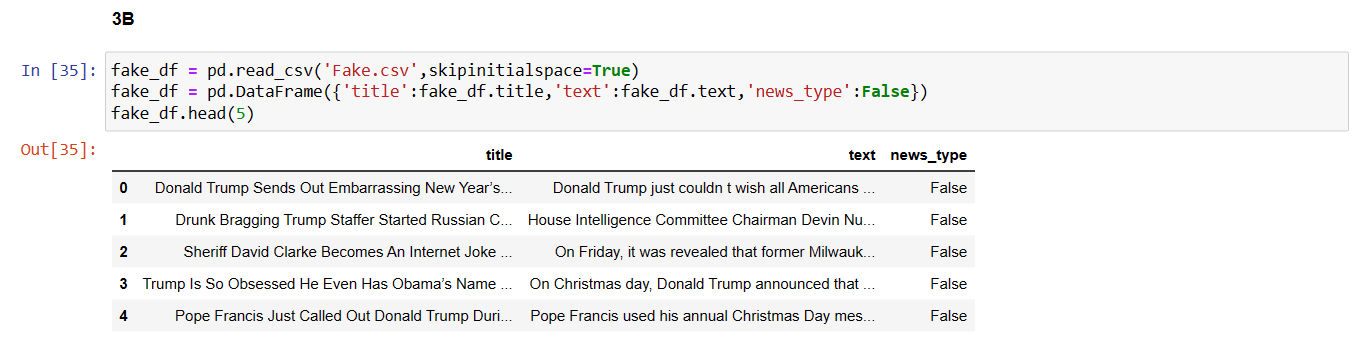
Question 3:

You will now implement a Naïve Bayes for text classification to detect fake or true news.

1. Upload the **True.csv** data provided on canvas into Python. You will create a new data frame by selecting the “title” and “text” columns, then, adding a new column called “news\_type” where all the values on this new column are “True”. So, your new data frame should have three columns; “title”, “text” and “news\_type”.



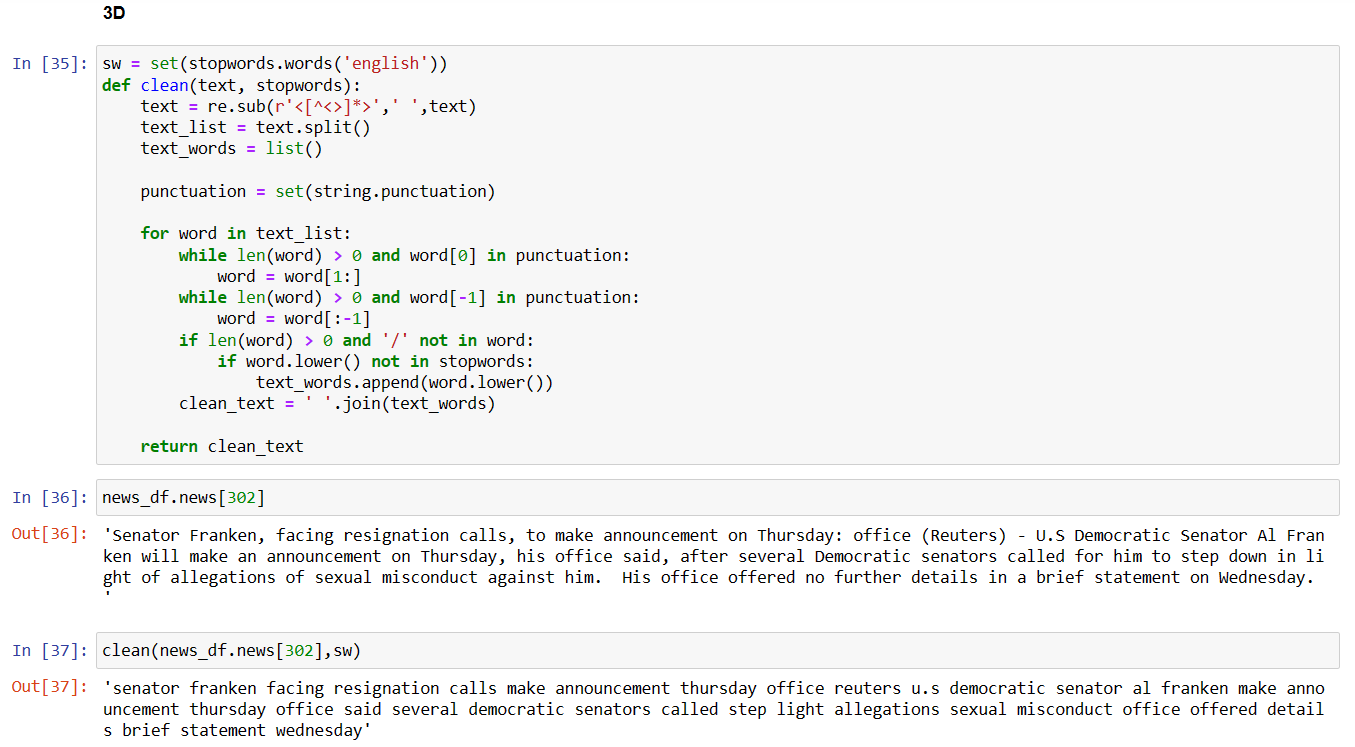
1. Upload the **Fake.csv** data provided on canvas into Python. You will create a new data frame by selecting the “title” and “text” columns, then, adding a new column called “news\_type” where all the values on this new column are “Fake”. So, your new data frame should have three columns; “title”, “text” and “news\_type”.

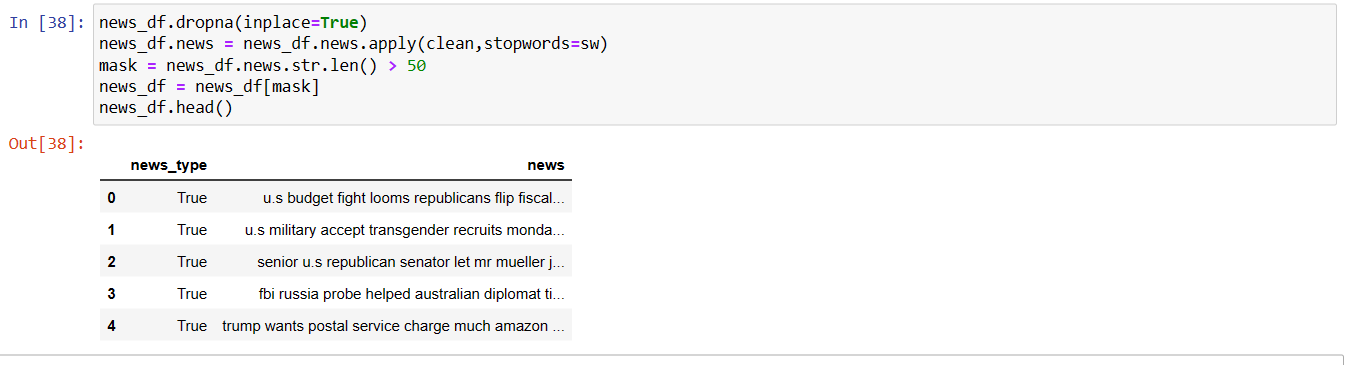


1. Merge the data frame in a) and b) so that one of the data frames is stacked vertically on top of the other. Combine the text in the “title” and “text” columns of the merged data frame into another column called “news”. Drop the “title” and “text” columns so that your final data frame is has only two columns, “news” and “news\_type”. Print the first five rows of your final data frame.

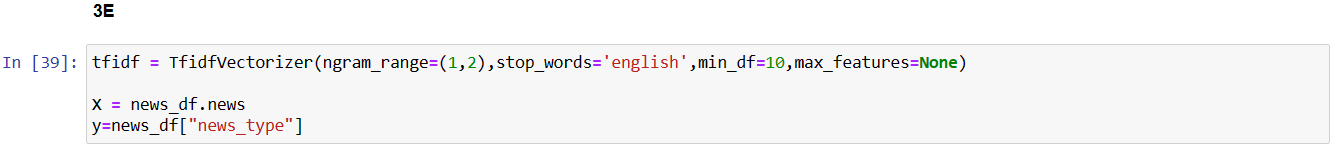


1. Preprocess your data by cleaning the textual data in the “news” column and removing the stop words, special characters, punctuations, etc especially at the beginning and end of each word. You can display any messy news text before you clean the data, then display the messy news text again after cleaning the data to see if your data cleaning worked well. Also, drop instances where the news text is less than 50 words.

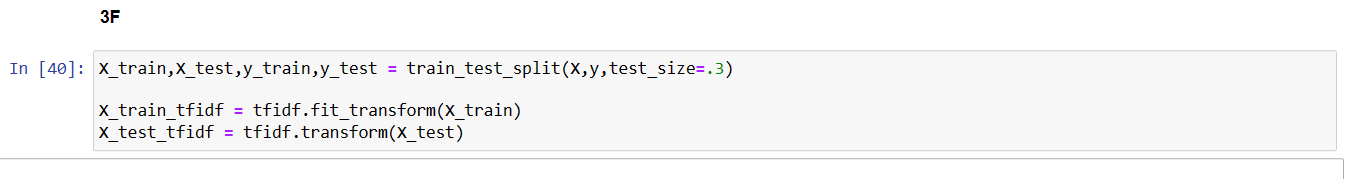




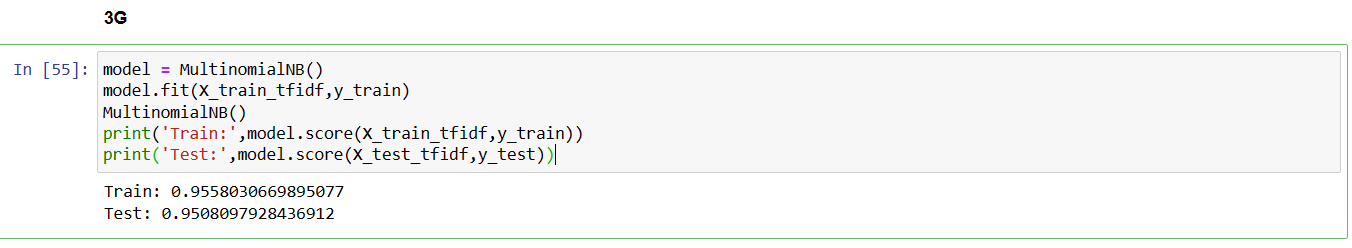
1. Transform the input text data into feature vectors where the entries of the feature vectors are term-frequency-inverse-document-frequency. Use the TfidfVectorizer() in sklearn.



1. Spit the feature vectors and the output variable into into X\_train, y\_train, X\_test and y\_test, you can let the test set be 30% of the entire data.

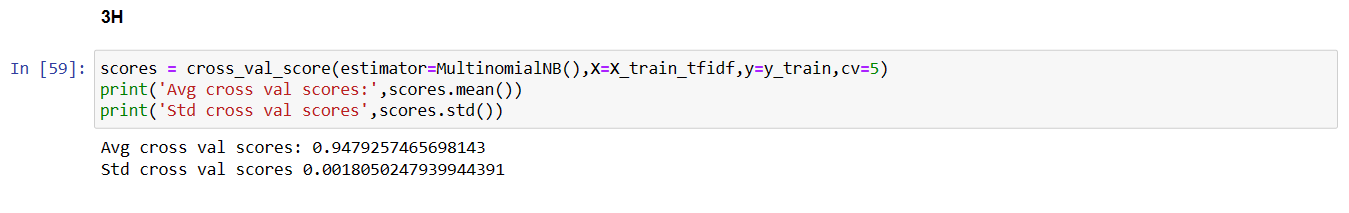


1. Fit an appropriate Naïve Bayes model and compute the training and test accuracy of the model. Is there overfitting?



The model is not overfitting as there is not much accuracy loss between train and test sets.

1. Fit a Naïve Bayes using cross validation and print the average cross validation score as well as the standard deviation of the cross-validation scores.



1. Select some hypermeters of your choice and tune using the grid search cross validation. Use some other hyperparameters than those used in class examples.

