

SMS Spam detection project

Springboard Data Science with Python Course





student: Reza nickmanesh

MENTOR: SERENA PERUZZO

April 3, 2018

**1. Project overview:**

The growth of mobile phone users has led to a dramatic increase in SMS spam messages. Therefore, there is a need for reliable filtering techniques capable of detecting spam messages from ham (legitimate) messages. The objective of the current capstone project is to develop an automatic filtering system using natural language processing (NLP) and machine learning techniques for SMS spam detection. Potential clients for an SMS spam detection system are mobile phone service providers. Mobile service providers can use this system to automatically filter out spam messages at the first place so that their clients do not receive those spam messages which potentially leads to more customer satisfaction. Using this filtering system, the mobile service providers can trace the sources of spam messages and implement strategies to eliminate/reduce the rate of spam messages which are generated/received. To achieve the goal of this project we take the following steps:

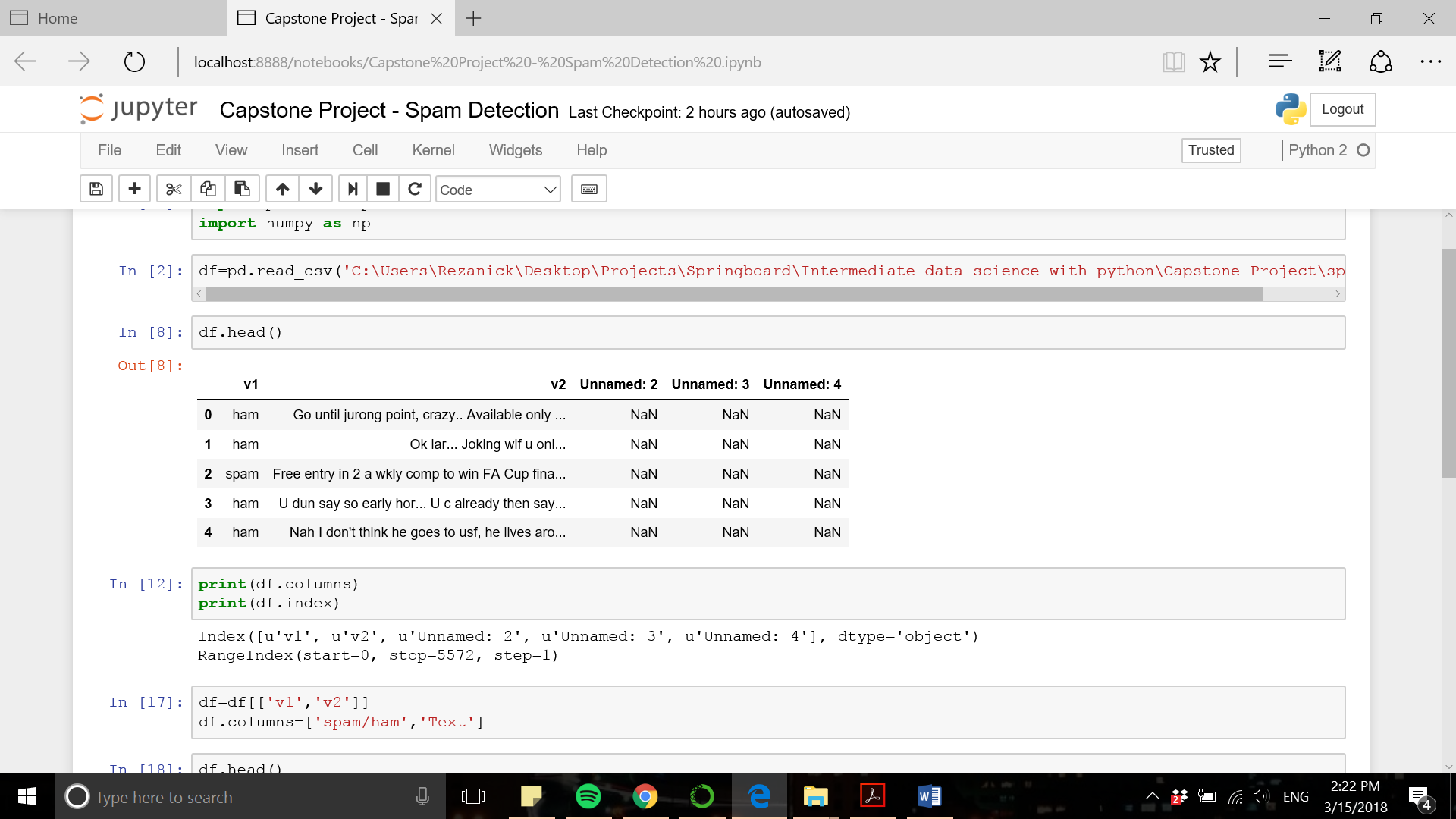
* Data exploration and cleaning
* Vectorization of text messages
* Training machine learning classifiers
* Model evaluation

**2. SMS spam collection dataset:**

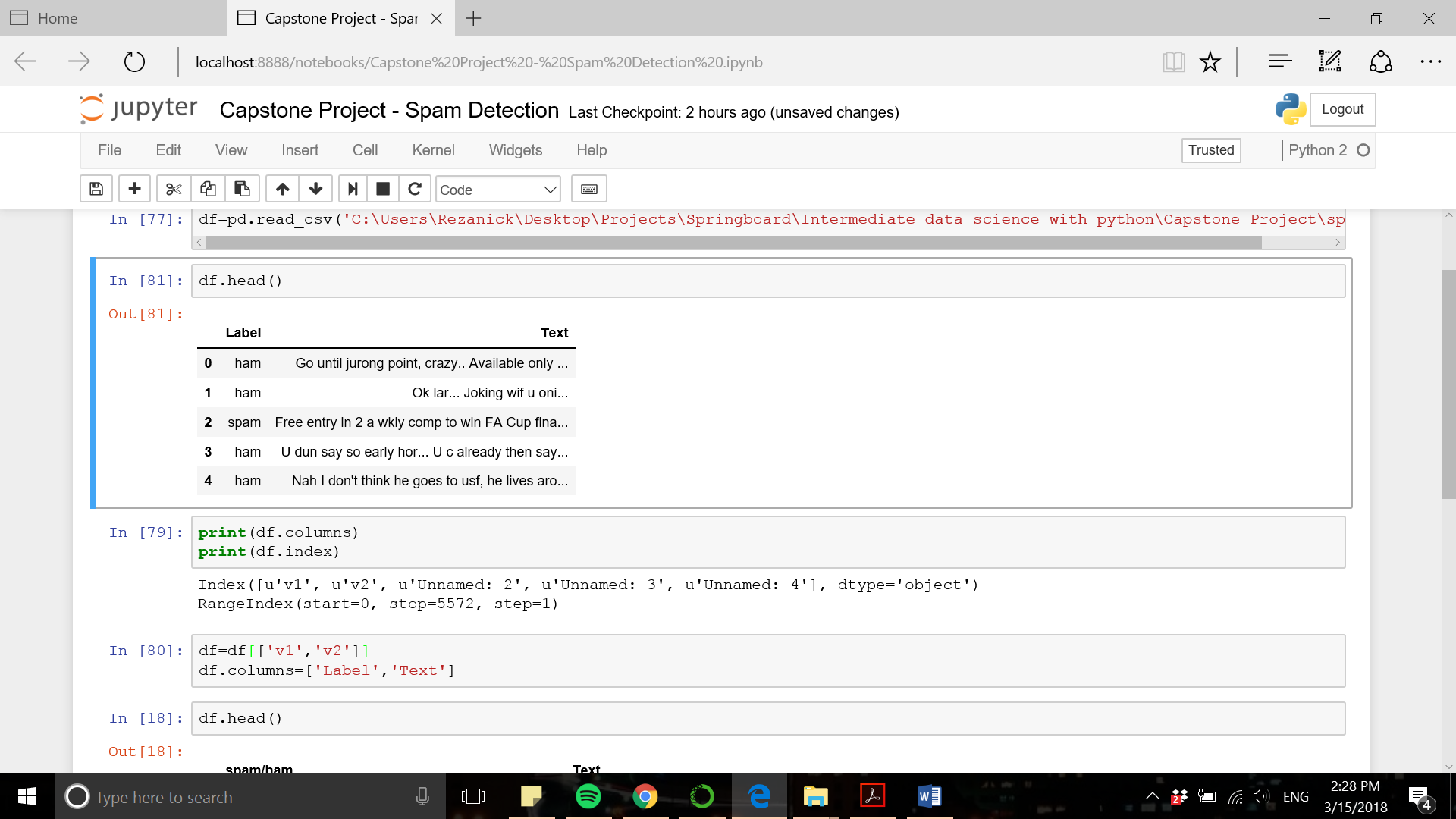
The dataset for this project originates from the UCI Machine Learning Repository [1] which is publicly available. The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

**3. Data wrangling**

SMS spam detection dataset consisted of 5572 entries (4825 ham messages and 747 spam messages). As shown below, the column ‘v1’ represented the tag for each message labeled as ‘ham’ or ‘spam’. The column ‘v2’ consisted of text messages in string format. There are three redundant columns (‘Unnamed: 2’, ‘Unnamed: 3’, ‘Unnamed: 4’) with NaN values.

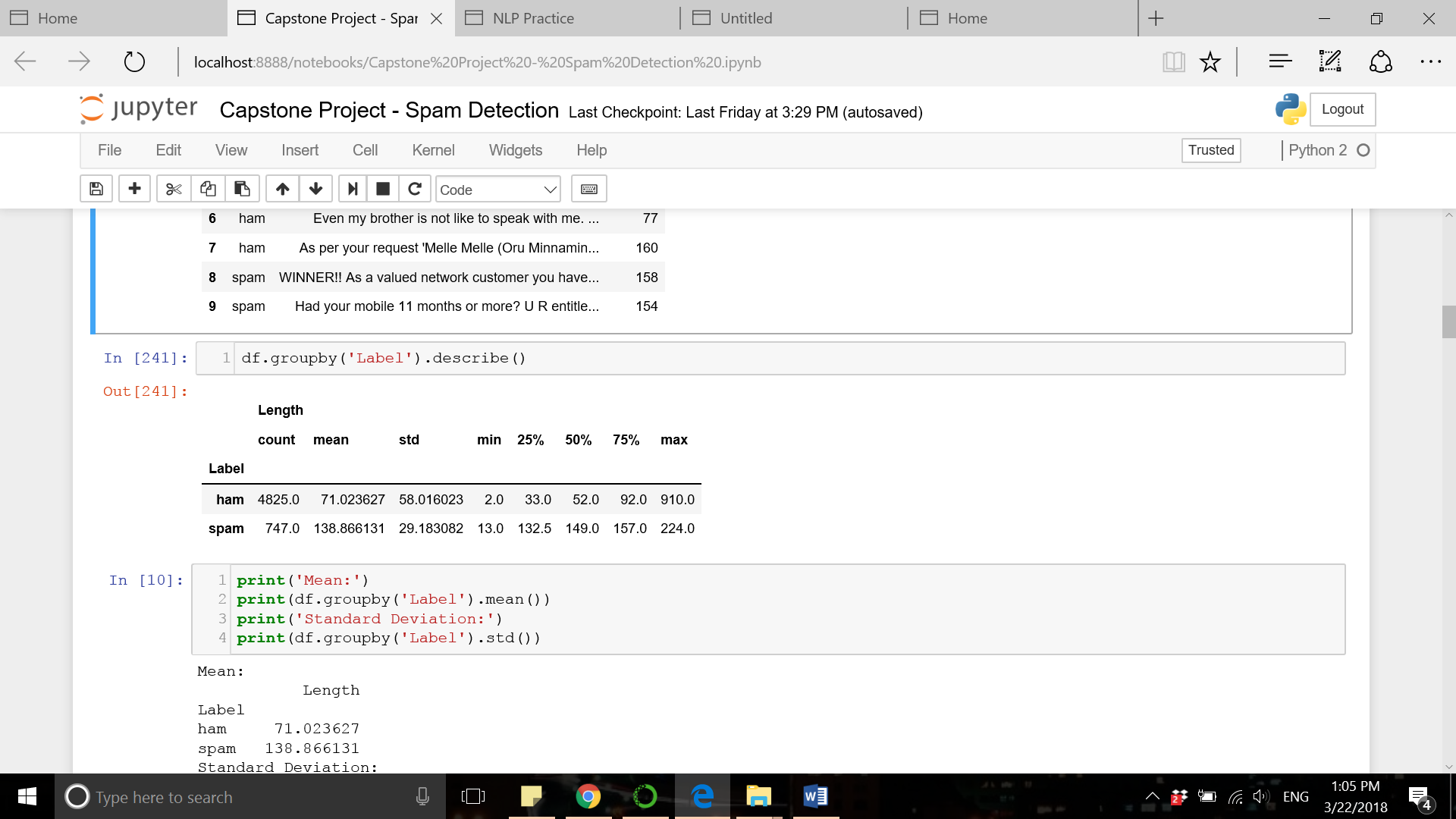


The first two columns (‘v1’ and ‘v2) are renamed into more meaningful column names (‘Label’ and ‘Text’, respectively) and the three redundant columns are dropped. The cleaned data frame is shown below and is ready for exploratory data analysis.

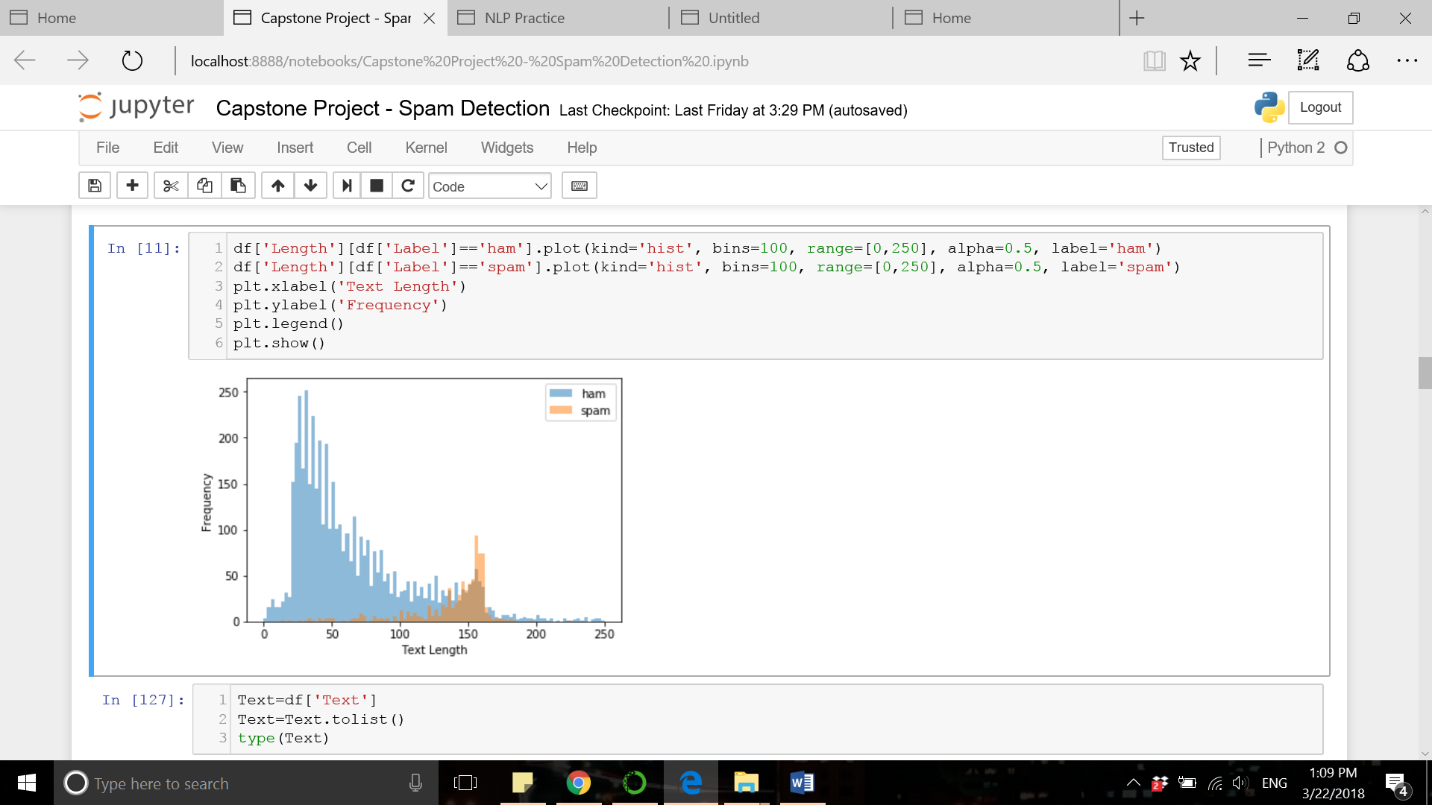


**4. Data exploration and visualization**

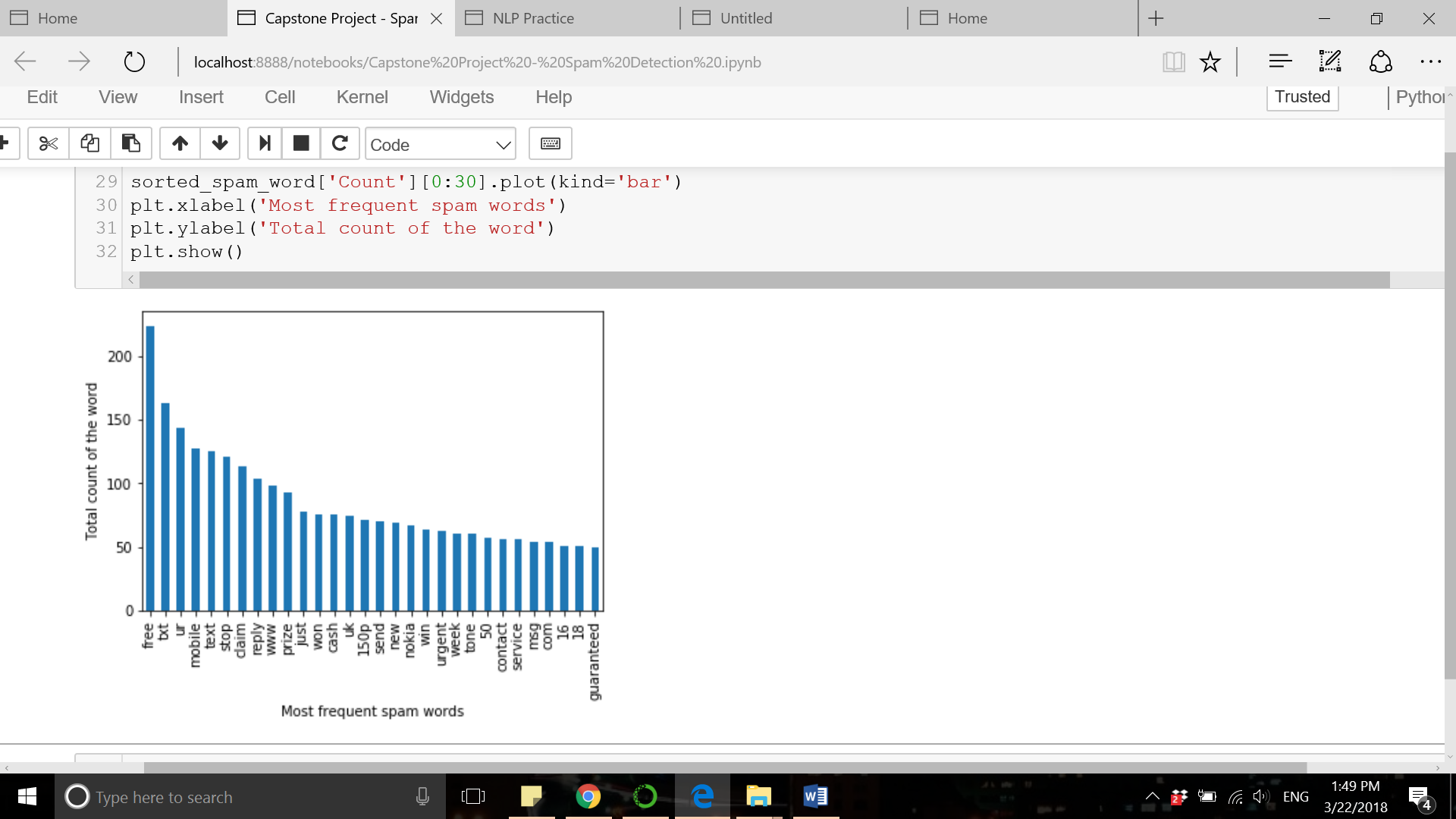
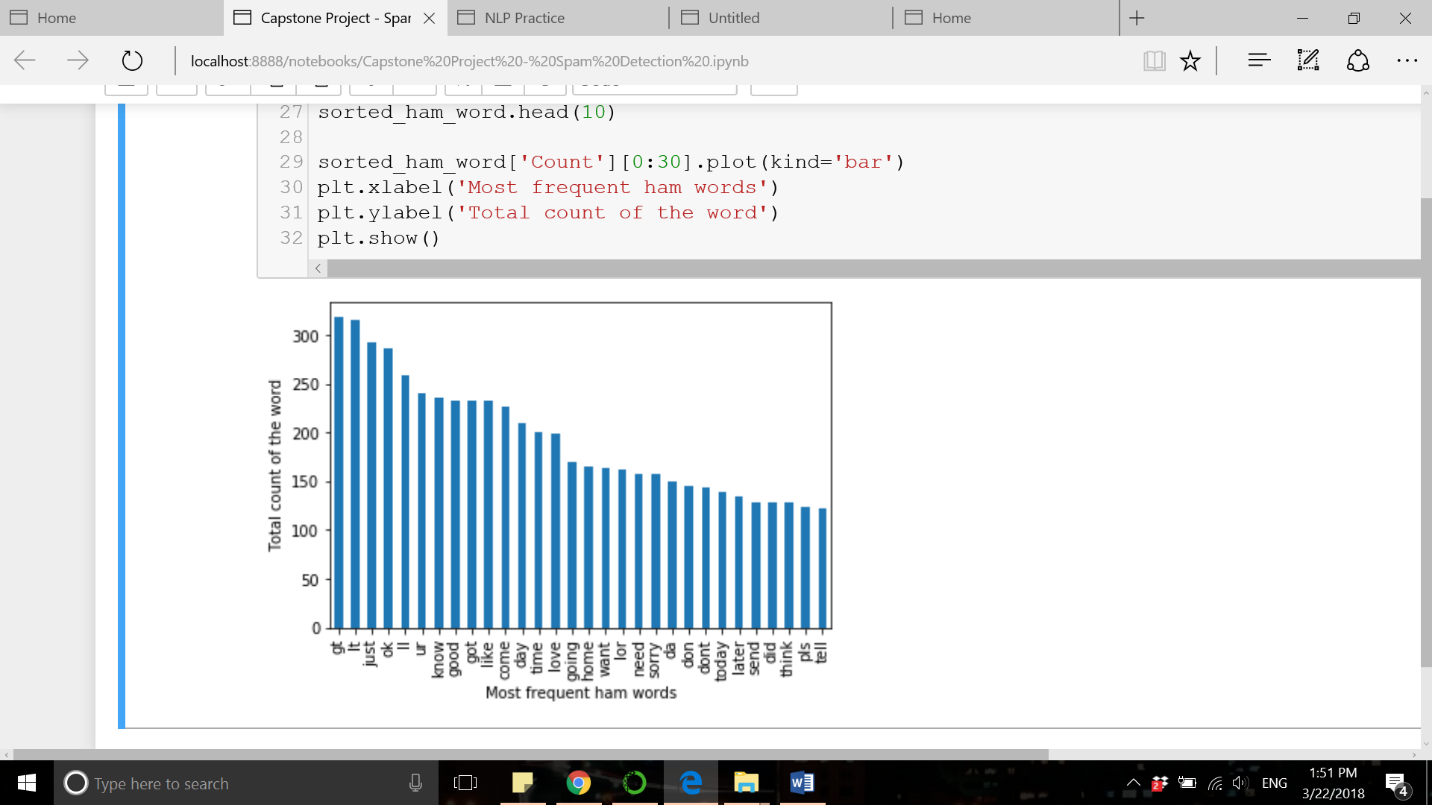
The question associated with this capstone project is which features in our data set could be used to detect spam messages from ham messages. The statistics about the length of each message clearly show that spam messages are longer than ham messages:



We visualize the distribution of message length by creating a histogram:



In addition to that, the word frequency data can provide further insights into the spam/ham messages and the features useful for detection purposes. The two charts below show the most frequent words in spam/ham messages and the total number of their occurrence at each group. It is evident that specific words are frequently found in spam messages which are used in ham messages less often. The most frequent word in spam messages is “free” which occurred 224 times while this word occurs only 60 times in ham messages (on average 1 out of 80 ham messages includes the word “free”, while 1 out of 3 spam messages includes this word). Therefore, vectorized representation of text messages created based on the vocabulary obtained from the existing corpus can provide the features for spam detection.

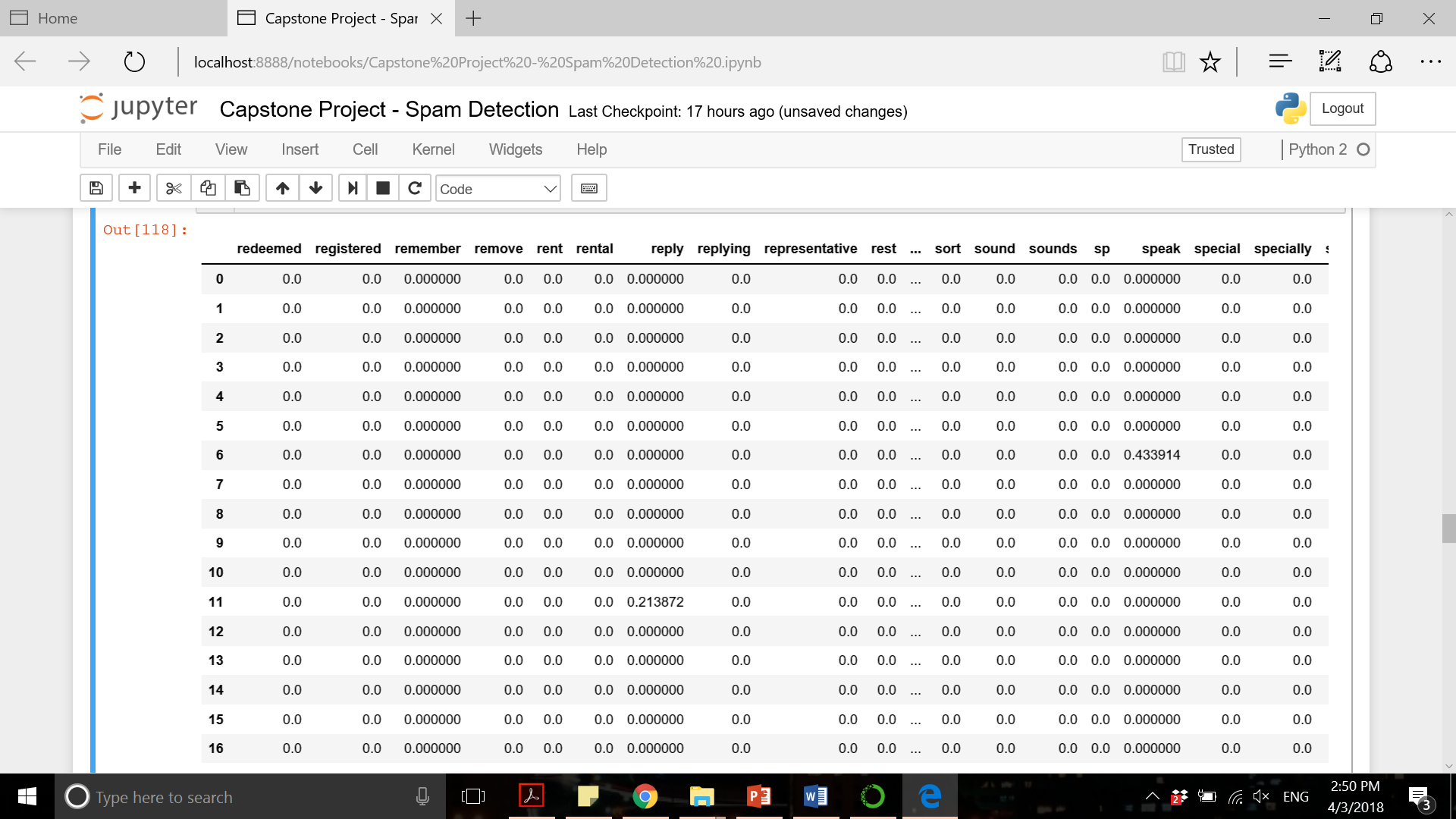


The bar charts below show the most frequent words in each group. Words like ‘free’, ‘claim’, ‘prize’, ‘reply’, ‘won’, ‘cash’, etc are frequent in spam messages and represent a common theme among spam messages.

**5. Vectorization of Text Messages using Tf-idf Representation**

Tf-idf, short for term frequency-inverse document frequency, is the numerical representation of text data which intends to reflect the importance of each word to a document in a corpus. The tf-idf value increases proportionally to the number of times a word appears in a document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general (useful to remove stop words).

The table below shows the vectorization of text messages. Each row represents a text message and the column names correspond to selected words.



**6. Algorithms and techniques:**

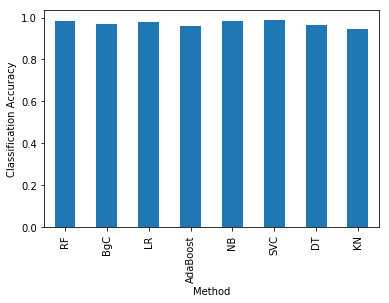
We approach our text classification problem as a supervised learning problem as the labeled data (spam vs. ham) is available. We select the first 1000 important words (according to tf-idf values) as the features used to classify spam and ham messages. Each text is represented by a vector of length 1000 with each element having the tf-idf weight for the word (value is zero if the word does not exist in the text). We examine the use of multiple machine learning algorithms in this project and evaluate their performances according to defined metrics (accuracy, precision, f1, and recall). The data is first split into a training set and test set with 80% of data being used for training classifiers and 20% for model evaluation. Cross-validation technique is also used to tune the hyperparameters for each of these machine learning algorithms to find the optimal parameters with the highest score (accuracy). Below is the list machine learning techniques implemented in current project:

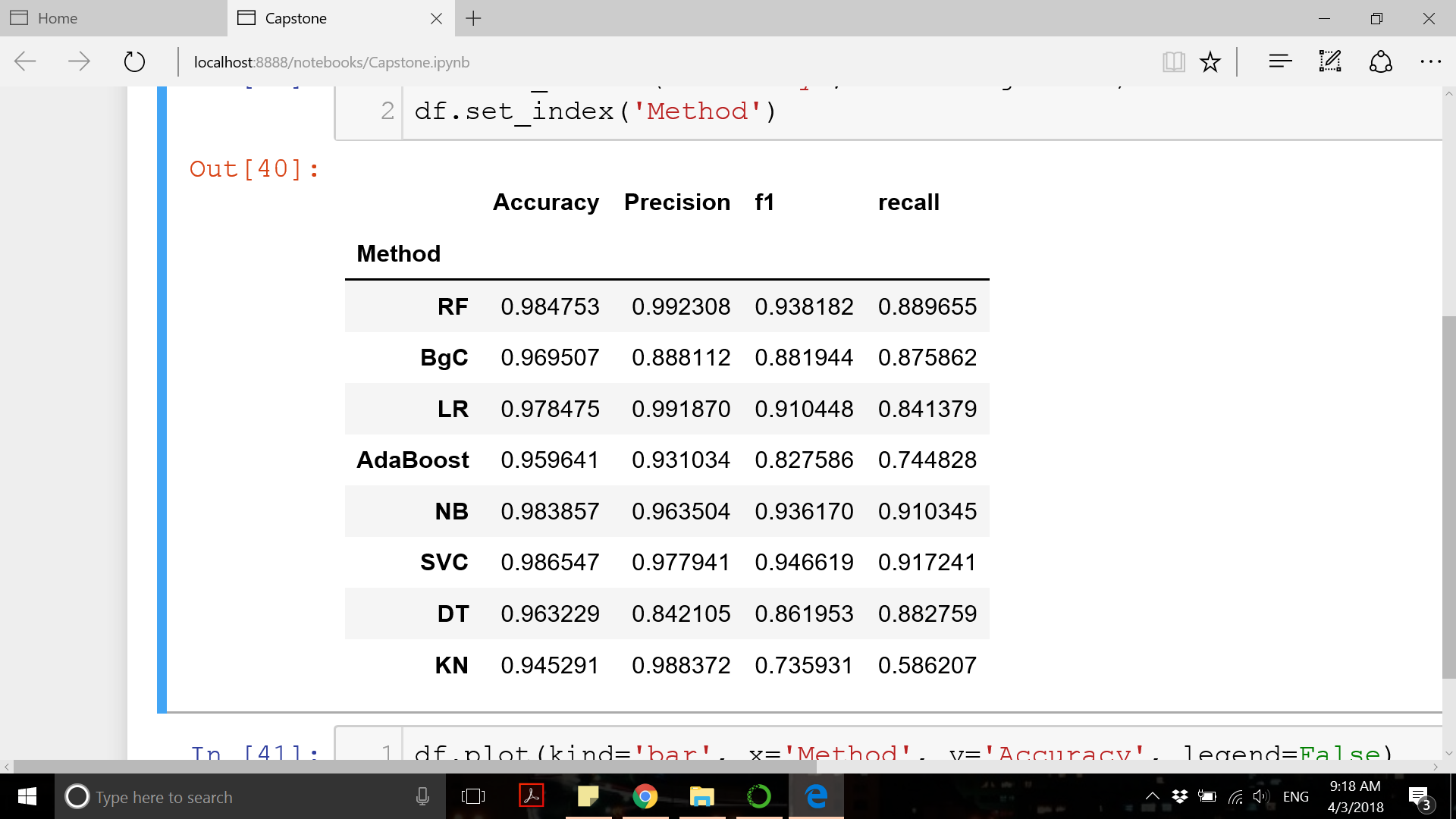
* Support vector machine
* K nearest neighbors
* Naïve Bayes
* Decision tree
* Logistic regression
* Random forest
* Adaboost
* Bagging

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classifier** | **Abbreviation** | **Hyperparameters** | **Value range/Options** | **Tuned value** |
| Support Vector Machine | SVC | Kernel | 'rbf', 'poly', 'sigmoid' | ‘sigmoid’ |
| Gamma | np.linspace(0.05, 1, num=10) | 0.78 |
| K Nearest Neighbors | KN | N\_neighbors | range(3,25) | 3 |
| Naïve Bayes | NB | Alpha | np.linspace(0.05, 1, num=10) | 0.05 |
| Decision Tree | DT | Min\_samples\_split | range(2,21) | 15 |
| Logistic Regression | LR | Solver | 'newton-cg', 'lbfgs', 'liblinear', 'sag' | ‘newton-cg’ |
| Random Forest | RF | N\_estimators | range(2,36) | 27 |
| AdaBoost | AdaBoost | N\_estimators | range(2,21) | 19 |
| Bagging | BgC | N\_estimators | range(2,21) | 16 |

The table below demonstrates in detail the hyperparameters associated with each machine learning algorithm, their corresponding range of values/options examined to tune the algorithm, and the final tuned hyperparameters. A 5-fold cross validation method was used on the train dataset to tune the hyperparameters.

We determined the benchmark model based on a paper [2] which reviewed works related to spam detection problem. Considering the average and range of accuracies that previous investigators have achieved, we set the 95% accuracy as our benchmark.

The optimal values for each machine learning technique was used to train the classifier and the performance of the classifier was examined in the test dataset. The table below illustrates the performance of classifiers based on the defined metrics:



All classifiers showed an accuracy of above 94% with support vector machine classifier having the highest accuracy (98.6%) among all. Support vector classifier also showed good performance according to other metrics (precision=99%, f1=97.8%, and recall=91.7%).

**7. Conclusion and Future Work:**

In the present project, we aimed to train classifiers to detect spam messages. Computers do not understand text data and the challenge is to convert text data into a readable format (numerical format) for the computer. Term frequency-inverse document frequency (tf-idf) is a common vectorization method which gives weights to words according to the frequency they appear in the documents and the whole corpus. This numerical representation of text data was used to train multiple machine learning classifiers. Among all classifiers, the highest accuracy scores were for support vector machine, random forest, and naïve bayes classifiers (all above 98% accuracy).

The accuracy of the spam detection models obtained in this project is good and is accurate enough for a variety of applications and has huge value for potential business clients. There are potential ways to improve the spam detection models. First, a more detailed analysis of the features used for training the classifier can be helpful. Though stop words were removed in our analysis, there are still misspelled words and numbers that exist in our dataset. Second, neural network models have proved to be powerful models for classification problems. A well-tuned neural network might provide a model with better performance.

**8. Acknowledgement:**

I would like to thank my mentor, Serena Peruzzo, for her valuable feedback throughout this course.

**8. References:**

[1] <https://archive.ics.uci.edu/ml/datasets/sms+spam+collection>

[2] Sarah Jane Delany et al. “SMS spam filtering: Methods and data”. Expert Systems with Applications