Machine Learning Nanodegree Capstone Project

Spam Detection using Naive Bayes and Support Vector Machines

By

Sylvester Ranjith Francis

Domain Background:

- Natural language Processing:**Natural language processing** is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (**natural**) languages. As such, **NLP** is related to the area of human–computer interaction.
- Machine Learning:**Machine learning** is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. **Machine learning** focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.
- Support Vector Machines:**Support vector machines** (SVMs, also **support vector** networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- Naive Bayes:It is a classification technique based on **Bayes'** Theorem with an assumption of independence among predictors. In simple terms, a **Naive Bayes** classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
- Text Feature Extraction: the **term** frequency – inverse document frequency (also called tf-idf), is a well know method to evaluate how important is a word in a document

Problem Statement:

- The Proposed project involves classifying the SMS messages received in an end-user's phone as either a Spam message or not a Spam message (Ham)
- Spam messages are a nuisance which cause various problems such as inbox storage issues
- Unwanted messages lead to delay in delivery of important messages
- It also leads to loss of bandwidth in the carriers

- The project involves removal of spam messages by classifying them and segregating them into a separate bin

Datasets and Inputs:

This project utilises the SMS spam dataset found in the UCI Machine Learning Repository .

- The dataset looks like this:

	Label	Message
0	ham	Go until jurong point, crazy.. Available only
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...

- The dataset consists of Tab separated values

Solution Statement:

- The solution to this problem is to classify the texts as spam and ham,and create a prediction model to do the same
- The end result being the creation of the prediction model with two different approaches namely
  1. Using a Naive Bayes Classifier with an accuracy of 95%
  2. Using a Support Vector Machine with an accuracy of 98%

An Outline of Project Design:

- The project is implemented as an IPython notebook using python 2.7,numpy, pandas,Sklearn,Matplotlib and Textblob
- The basic goal of the project is to design a Naive Bayes Classifier to classify the text messages into Spam and Ham (not spam)
- Initially a naive bayes classifier approach was utilised.
- Step 1:Initial exploration of the data was performed
- Step 2:Once the data exploration was performed,I proceeded to convert the raw messages into vectors.
- Step 3:The mapping was not 1 to 1 hence the **Bag of Words** approach is used where each unique word in a text is represented by a number
- Step 4: Converting the text into a vector involves three steps namely,Counting the frequency of a word occurring in each message (term frequency),Weighting the counts, so that frequent tokens get lower weight (inverse document frequency),Normalizing the vectors to unit length, to abstract from the original text length (L2 norm)
- Reference : http://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-words-representation
- Step 5: The model has to trained using a naive bayes classifier to check if our model works
- Step 6: Split the data into testing and training data using Sklearn's train_test_split
- Step 7:The testing data is fed into the designed Naive bayes classifier,to find out that the prediction engine gave an accuracy of 95%
- Step 8: To improve the performance of the prediction engine,I tried designing a Support Vector Machine after many different trials and errors to improve the accuracy
- Step 9: The resulting Support Vector Machine gave an accuracy of 98% which is a huge improvement compared to the Naive bayes classifier
- The result being that we were able to classify the messages successfully as spam and ham with an accuracy of 98%

References:

Text Feature Extraction- Bag of Words Representation:
http://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-words-representation
http://scikit-learn.org/stable/auto_examples/model_selection/grid_search_text_feature_extraction.html
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
Naive Bayes - Sklearn Documentation:
http://scikit-learn.org/stable/modules/naive_bayes.html
Support Vector Machines -Sklearn Documentation:
http://scikit-learn.org/stable/modules/svm.html
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
Pipelines-Sklearn Documentation:
http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
Cross validation-Sklearn Documentation:
http://scikit-learn.org/stable/modules/cross_validation.html
http://scikit-learn.org/0.17/modules/generated/sklearn.grid_search.GridSearchCV.html
http://scikit-learn.org/0.17/modules/generated/sklearn.grid_search.GridSearchCV.html
Model Selection-Sklearn Documentation:
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold
http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
Accuracy score/Metrics-Sklearn Documentation:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score
Decision Tree Classifier-Sklearn Documentation:
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
Other References:
https://www.thinkful.com/projects/building-a-text-classifier-using-naive-bayes-499/
http://blog.fliptop.com/blog/2015/03/02/bias-variance-and-overfitting-machine-learning-overview/

Affirmation:
I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.

Sylvester Ranjith Francis