Capstone Project Proposal
Machine Learning Engineer Nanodegree

Media Product Classification

Nagaraju Budigam
November 28th, 2017

Project Overview

Note: This is the original problem that I am developing at production level. Due to company compliance policies original datasets are not exposed, however the provided datasets are resembling the original datasets.

Indix is a **Product Intelligence** company based in Seattle, that currently offers a cloud-based product information platform. It is also building a broad and deep product catalogue to enable mobile and desktop apps and websites to become product-aware. Indix provides access to APIs that enable developers to build product-aware applications.

As mentioned above, Indix hosts the world's largest collection of programmatically accessible structured product information in the cloud. The products in our database belong to 25 verticals and that translates approximately to 6000 sub-categories. Every product that we carry in our database gets stamped with information about the "category" it belongs to.

Indix's services are centred on proprietary algorithms that structure crawled product data and a data-as-a-service business model.

The database offers coverage for most consumer retail product categories. The database also includes many industrial and business-to-business products. Indix provides brands and retailers with access to data such as specifications, facets, availability, assortment, promotions, and real-time pricing information. In comparison to offerings from Google, which are influenced by Google's utilization of relevance algorithms, or Amazon, which is limited to only the products in its own catalogue, Indix's infinite product catalogue helps all client-facing digital media and environments become more product-aware.

Problem Statement

As Indix has the tons of data collected programmatically from web, it has become a challenging task to classifying a product into a particular category, which is very important to serve various use cases – like, helping search, performing product matching, providing category specific insights, and so on.

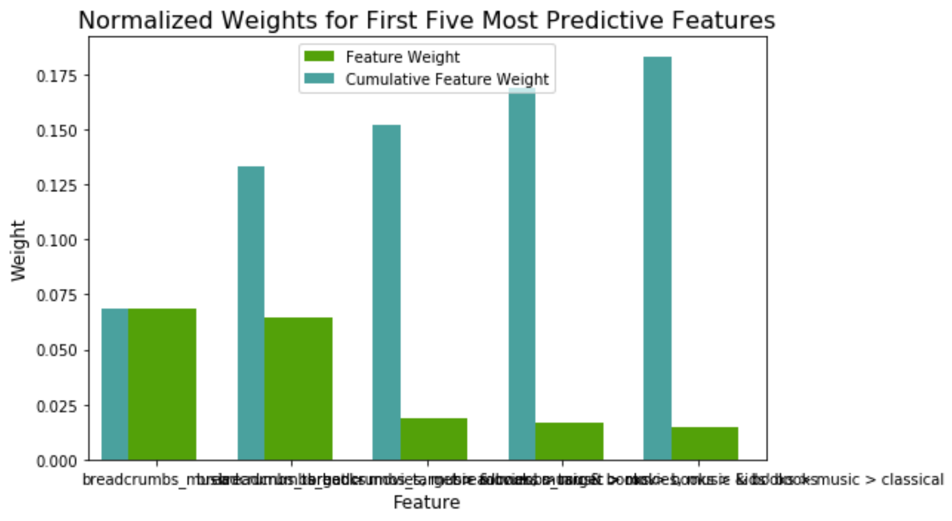
From the project overview and project statement it is clear that, the problem of stamping every product in our catalogue into a category is a **Classification** problem in **Machine Learning.** We have a handful of techniques and methods in Machine Learning to tackle this kind of real world problems.

To be more specific, here the stamping every product in our catalogue is a **Multi Class Classification** problem.

All the features in the training data set are categorical (text data) and it is clear that this is problem requires to apply **Natural Language Processing** techniques, as we are dealing with textual data to convert this textual data into a numerical form. For achieving this we are using library Countvectorizer, provided by sklearn, which will lower case the input, remove all punctuation marks, remove stop words, performs tokenization and constructs a bag of words model, by using which we construct a vector for each document of the input and feed it into a machine learning algorithm.

As mentioned below, training dataset contains storeid, url, additionalAttributes, breadcrumbs and label features. However, we are considering **breadcrumbs** as the input features because most of the values of other features are missing (null and NaNs) and they are very less informative, hence not contributing to predict the label of an unknown product. This statement is being justified in the free form visualization section, where we plotted a graph to understand the feature importance. So, it’s clear that breadcrumbs will our input feature and lable would be output that will be predicted by our machine learning model.

Feature Importance Plot



Datasets and Inputs

The training dataset contains storeid, url, additionalAttributes, breadcrumbs and label features.

	storeId	url	additionalAttributes	breadcrumbs	label
0	23	http://www.walmart.com/ip/best-of-toto%3A-piano---vocal---guitar/12005673	Contributed by=Toto;Format=Paperback;Number of Pages=66;Walmart No.=9781603781558;ISBN-13=9781603781558;Publish Date=Aug 2009;ISBN-10=1603781552;Publisher=Hal Leonard Corp	books > art music & photography > music	books
1	NaN	NaN	NaN	best buy > computers & tablets > computer cards & components > memory ram > desktop memory	rest
2	23	https://www.walmart.com/ip/34441317	NaN	music on cd or vinyl > rock music on cd or vinyl > alternative rock music on cd or vinyl	music
3	26	https://www.overstock.com/Books-Movies-Music-Games/The-New-Cooks-Tour-of-Sonoma/523652/product.html	NaN	books & media > books > cooking & food books > general cooking	books
4	22	http://www.target.com/p/amore-version-ii/-/A-16771282	NaN	target > movies, music & books > music > classical	music

Input Features: storeid, url, additionalAttributes, breadcrumbs

Output: Product Label

Featureset Description:

- **storeId** - a unique number for identifying a website, numerical data, discrete
- **url** - url of a product page.
- **additionalAttributes** - Product attribute related to a particular product. These are key, value pairs that can be found in tabular format as product information for most products in e-commerce websites. This is a categorical data.
    - **An example of additionalAttributes**

      {"ASIN": " B000JJRY9M",

      "Amazon Bestsellers Rank": " in DVD & Blu-ray (See Top 100 in DVD & Blu-ray)",

      "Average Customer Review": " Be the first to review this item",

      "Classification": " Exempt",

      "DVD Release Date": " 26 Feb. 2007",

      "Format": " AC-3, Colour, Dolby, DVD-Video, PAL",

      "Language": " English",

      "Number of discs": " 1",

      "Region": " Region 2 (This DVD may not be viewable outside Europe. Read more about

      DVD formats.)",

      "Run Time": " 287 minutes",

      "Studio": " Hip-O Records"}
- **breadcrumbs** - breadcrumb captured at the page. Breadcrumbs typically appear horizontally across the top of a Web page, often below title bars or headers. This is a categorical data.
    - **An example of breadcrumb** subjects > travel > world regions > europe > european nations > france

- **label** - The class to which a product belongs. Values belong to the finite set (‘books’,’music’,‘videos’,’rest’). This is a categorical data.

  It is possible that for some products only one among (2) or (3) might be available. It means that we may not have data for features 2 and 3 some times.

  The problem statement is to classify the products into any one of the buckets (i) Books (ii) Music (iii) Videos (iv) Rest - A default class for products which doesn’t belong to (i),(ii) or (iii) category.

Solution Statement

In this submission I have built a Classifier, which lables a unknown product by considering the breadcrumb as input to the model.

Steps followed for building an efficient Multi Class Classifier:

- Data Exploaration - Data Exploration provides good insights of the training dataset, such noise rate in the dataset, no.of samples avaialble for each category/label.
- Algorithms and Techniques - I have picked up 3 supervised algorithms DecisionTreeClassifier, MultinomialNB and SVC.
- Evaluation Metrics Selection - I am using Accuracy and F-beta Score as the evalaution metrics to decide and pick a good model among the 3, I mentioned above.
- Data Preprocessing - Remove noise, such as stop words, special characters, numbers.
- Feature Selection - Select the right features that better describes the label of a product, here **breadcrumbs** feature is more helpful in predcting the label of a product than other features as most their values are NaNs or null.
- Feature Transformation - Transform the categorical data into numerical form.
- Train and Test Data Split - Split the traing dataset into trainig and test set, 80% and 20% respectively.
- Intial Model Training and Evaluation - Train the 3 models, DecisionTreeClassifier, MultinomialNB and SVC and Calculate the performance metrics accuracy and f-beta score.
- Model Selection - Select the model that has the best accuracy and f-beta score.
- Hyper Parameter Tuning - Perform the hyper parameter tuning using grid search and find out the best parameter combination.
- Training the Final Model - Train the selected model by passing the best parameters combination obtained in Hyper paramter tuning step.
- Prediction - Perform the prediction on the unknown dataset.

Metrics

Accuracy:

Accuracy measures how often the classifier makes the correct prediction. It's the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

Accuracy = true positives + true negatives/dataset size

Well, Accuracy is calculated as the portion of true labelled instances to total number of instances. The questions are what is wrong with accuracy that we need other performance measures? The problem is that in some datasets we can achieve high accuracy with weak models such as a dummy classifier to classify instances with the most frequent label. In cases such as outlier detection or in any dataset that a large portion of samples are of one class label the dummy classifier can achieve a high accuracy such as 80%(in cases that 80% of data are of the majority class label). This is while stronger models may even have lower accuracy. This is called the Accuracy Paradox. Hence, we usually prefer to use other performance measures such as Precision, Recall, F-measure or etc.

Hence I feel that accuracy is not enough and we need a report of classifier about the precision and recall that can better understood through f-beta score.

F-beta Score:

- F-beta score treats both precision and recall with same importance, when beta=1, let's say we need model which care a bit more about precision than recall, then we want something more skewed towards precision.
- Smaller the beta the model more skewed towards precision.
- Larger the beta the model more skewed towards recall.
- Note: Finding a good value of beta requires a lot of intuition of data and a lot of experimentation.

Precision = `[True Positives/(True Positives + False Positives)]`

Example: In case of a spam filter, it tells us what proportion of messages we classified as spam, actually were spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all positives(all words classified as spam, irrespective of whether that was the correct classification), in other words it is the ratio of

Recall(sensitivity) = `[True Positives/(True Positives + False Negatives)]`

Example: In case of a spam filter, tells us what proportion of messages that actually were spam were classified by us as spam. It is a ratio of true positives(words classified as spam, and which are actually spam) to all the words that were actually spam.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

So, I would like to use accuracy and f-beta score as an evaluation metrics to access the performance of the classifier. I would like to use the beta value as 0.5, so my classifier is a bit