Capstone Project Report
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 original datasets are not exposed, however the provided datasets are resembling the original datasets.

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.

Problem Statement

Aim of this project is 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. The problem of stamping every product in our catalogue into a category is a **Multi Class Classification** problem. In this submission I have built a micro version of this classifier where I will predict 4 classes.

Evaluation Criteria

Accuracy:

When we want to evaluate a set of predicted labels or performance of Machine Learning models we use different performance measures. Accuracy, Precision, Recall, F-beta (usually people use F-1) or etc. But none of the afore mentioned methods except Accuracy work for Multi-class data where class labels tend to have more than two (binary) different values.

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:

- F1 score treats both precision and recall with same importance, 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.**

$$F_\beta \text{ SCORE} = (1+\beta^2)\beta^2 \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

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 skewed towards the precision.

- **predicted_output.csv** is the final predicted output file, which contains the test set with an additional column called **label being added** which that will have the predicted label of the product.

II Analysis

Data Exploration

Feature set Exploration:

1. **storeId** - a unique number for identifying a website, *numerical data, discrete*

2. **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"}

3. **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

4. **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.

Training data has the following fields and contains total 603201 records.

Sample Data:

	storeId	url	additionalAttributes	breadcrumbs	label
0	23	http://www.walmart.com/ip/best-of-toto%3A-pian...	Contributed by=Toto;Format=Paperback;Number of...	books > art music & photography > music	books
1	NaN	NaN	NaN	best buy > computers & tablets > computer card...	rest
2	23	https://www.walmart.com/ip/34441317	NaN	music on cd or vinyl > rock music on cd or vin...	music
3	26	https://www.overstock.com/Books-Movies-Music-G...	NaN	books & media > books > cooking & food books >...	books
4	22	http://www.target.com/p/amore-version-ii/-/A-1...	NaN	target > movies, music & books > music > class...	music

Features Description:

	storeId	url	additionalAttributes	breadcrumbs	label
count	1770	1094	772	4874	4896
unique	9	1094	771	2700	4
top	23	http://www.barnesandnoble.com/w/golfing-in-haw...	Country of Origin=United States	books	rest
freq	425	1	2	146	3126

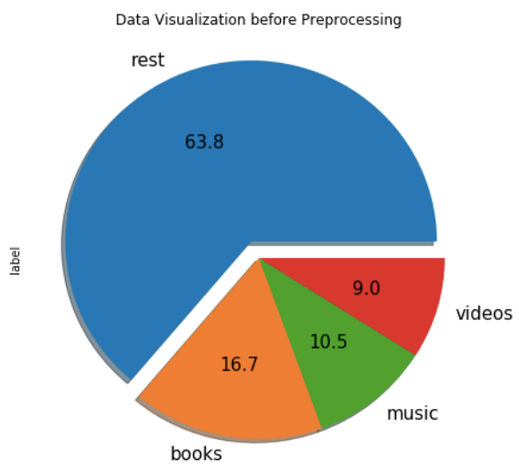
Total records of training dataset 4896 .

- From the above feature description, it's clear that features storied, url , additionalAttributes are having lot of missing information.
- Since all the features are having categorical data, it needs to be transformed into numerical form.

Exploratory Visualization

- From the below Pie char visualization it is clear that more than **63%** of data is labeled as **rest** and on the other hand in remaining data, we could see that more samples are related to books category. So categories rest and books have more samples than other categories.

- From the pie chart, it is also clear that the **dataset is unbalanced.**



Algorithms and Techniques:

As per the project problem statement, it is clear that we need to employ the Supervised Machine Learning Algorithms for predicting label of unknown product label. So, I would like to try with the following algorithms and based on the performance metrics mentioned above, would like to pick the best one.

Multi Nomial Naive Bayes: Super simple, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so we need less training data. And even if the NB assumption doesn't hold, a NB classifier still often performs surprisingly well in practice.

Decision Trees: Easy to interpret and explain. Non-parametric, so we don't have to worry about outliers or whether the data is linearly separable (e.g., decision trees easily take care of cases where you have class A at the low end of some feature x, class B in the mid-range of feature x, and A again at the high end).

SVMs: High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. Memory-intensive and kind of annoying to run and tune, though, so I think random forests are starting to steal the crown.

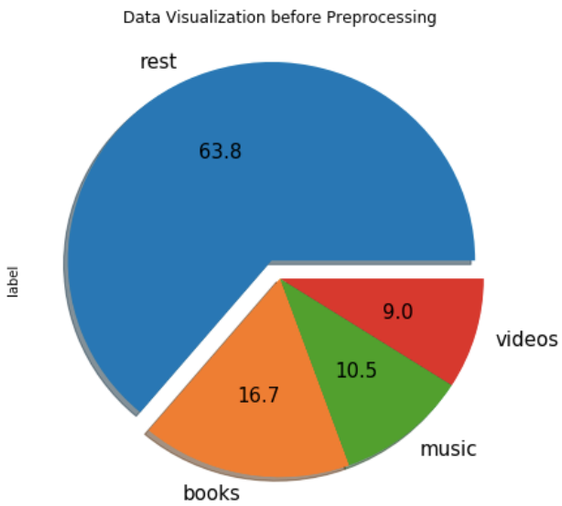
For all the above mentioned algorithms we feed in a vector constructed usning Bag of Words of Model of the training dataset.

III. Methodology

Feature Selection

- From the data exploration, it is straight forward that store id, url and additional attribute features (most of these features are NaNs). Hence they don't contribute to predict the label of a product.

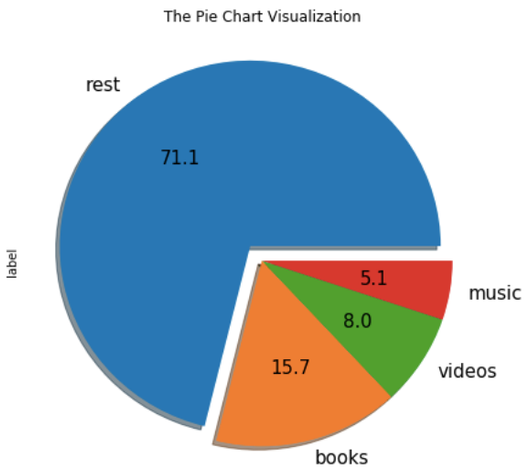
- I believe that, the feature breadcrumbs are right feature to choose to determine label of the unknown product.



Data Pre-processing

- In the data processing we **drop all the null or NaN rows** of breadcrumb and lable columns, and also we can ignore other columns they are not contributing to predict the output label.
- In the second step we **remove all the duplicate** rows by considering label and breadcrumb columns as they are our point of interest.
- As the data of breadcrumb feature contains special characters, numbers, it would be wise to remove all such noise from the data, I have taken care of this step during the tokenization.

Training Dataset Visualization after Data pre-processing



Data Pre-processing Description:

- `Original Dataset size: 4896`
- `Dataset size after noise removal: 4874`
- `Dataset size after duplicate removal: 2700`
- `Total noise removed from Training Dataset is: 2196`

`Sample Data after Pre-processing:`

	breadcrumbs	label
0	books > art music & photography > music	books
1	best buy > computers & tablets > computer card...	rest
2	music on cd or vinyl > rock music on cd or vin...	music
3	books & media > books > cooking & food books >...	books
4	target > movies, music & books > music > class...	music

Training and Testing Data Split

As it is a rule of thumb, we split the original input data set into training and testing data sets, where training dataset size should be 80% of the original data and testing dataset size would be remaining 20%.

```
Training set has 2160 samples.
Testing set has 540 samples.
```

Feature Transformation

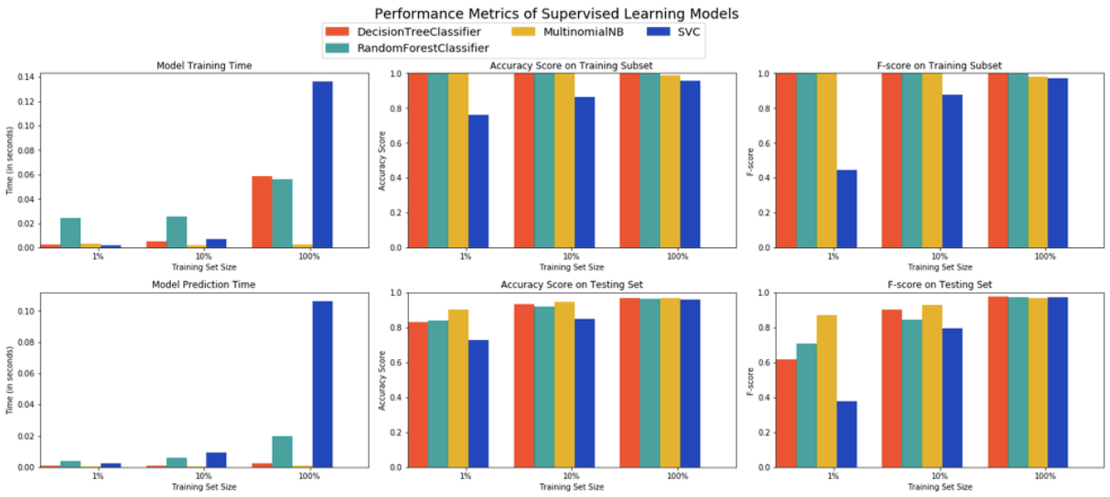
- As we are dealing with the text or categorical data it is a rule of thumb to transform all such data into numerical form so we can feed it into the machine learning algorithms, as they work only on numerical inputs.
- We use CountVectorizer of sklearn machine learning library, where it construct a Document Term Matrix and construct Bag of Words for the input data.
- After transforming the breadcrumb data we have to transform the labels of each category to a numerical form, We achieve this by using the Label Encoding module of sklearn.

Sample Data after Feature Transformation:

	breadcrumbs	label
0	books > art music & photography > music	0
1	best buy > computers & tablets > computer card...	2
2	music on cd or vinyl > rock music on cd or vin...	1
3	books & media > books > cooking & food books >...	0
4	target > movies, music & books > music > class...	1

Model Training

Once we perform data pre-processing and transform, we try to fit or train a couple of classification models and check which is performing good on the given dataset. You could see the performance metrics of various models fit on the 10%, 50% and 100% of dataset.



IV. Results

Model Evaluation and Validation

- I would like to use accuracy and f-beta score as an evaluation metrics to assess the performance of the classifier.
- As the dataset is unbalanced I would like to use f-beta score as the another evaluation metric, with beta being 0.5.

Model Selection and Hyper Parameter Tuning

- By taking the models accuracy and f-beta score into consideration, I have selected Decision Tree Classifier is the right fit for my problem.

- So, by using gird search I have performed the hyper parameter tuning to get the best parameter combination of the model.

- It is noticed that even after taking the best parameter combination returned by grid search, the accuracy and f-beta scores of the optimized model is not improved, however by using Decision Tree Classifier I could achieve 96.87% accuracy and 0.977 f-beta score.

Classifier Performance before and after Hyper Parameter Tuning

Unoptimized model

Accuracy score on testing data: 0.9685
F-score on testing data: 0.9778

Optimized Model

Final accuracy score on the testing data: 0.9833
Final F-score on the testing data: 0.9772

Justification

Final Model Selection

We could see the significant improvement in the accuracy of the classifier after the grid search and now we've got the best parameters combination to build the final model.

The best Parameter combination for the Final Model.
{'class_weight': None,
 'criterion': 'entropy',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'presort': False,
 'random_state': 1,
 'splitter': 'best'}

Final Model Training

Train the final model by usng best paremeter combination as mentioned above.

Predict using Final Model

Perform the prediction and label the products with the most relevant label appropriately.

Test Dataset Exploration:

- Original test dataset size: 4517
- Test dataset size after dropping NaNs: 4508
- Test dataset size after dropping duplicates: 2669

Wrote predicted output to the file predicted_output.csv Successfully!!!

Visualization of Final Models Prediction