**Machine Learning Engineer Nanodegree**

**Capstone Project**

Ramakanth Vanga,

May 27th, 2018

**Definition**

**Project Overview**

Specifying charge account on Invoice Lines is a daily activity by Accounts Payables team in any medium-large organizations. AP teams receive Invoices from Vendors regularly and users manually specify the charge accounts. This is time taking, repetitive and painful process.

**Problem Statement**

The goal is to create an application, which will look at the history of Invoices from a Vendor and try to predict charge accounts for a new Invoice from Same Vendor.

1. Read charge accounts for Invoice lines in past 1 year.
2. Assuming that vendor has been extracted correctly.
3. Features will not change from Vendor to Vendor for an organization.

**Metrics**

Only True Positives are important for this Metric. We want the charge account to be correctly predicted, anything else is a negative. So calculation can be as follows:

Metric = True Positives/ Total Dataset size

If a charge account is not predicted at all or predicted but wrong, it can be excluded.

**Analysis**

**Data Exploration**

Every customer will have a process to determine a charge account. Several data points related to an Invoice and company’s hierarchy would determine a charge account. As we are not sure which data points a customer uses, we need a generic feature name. Hence I am using the column names as feature1, feature2 … feature10.

In a typical case charge accounts are determined using 3-5 features. Hence I am limiting to a maximum of 10 features. Features can be characters or numbers, so we will use label encoding so that all characters will be converted to numbers.

**Exploratory Visualization**

**Algorithms and Techniques**

In several cases each feature is independent of other. However there could be few cases where a feature is dependent on the other. To support most of the cases, it can be assumed each feature is independent of the other.

Naïve Bayes algorithm is an ideal algorithm for this problem, where probability of each feature is calculated independently and joint probability is applied for the final outcome.

**Benchmark**

**Methodology**

**Data Preprocessing**

Each feature has to be treated as a character. So each unique feature needs to be converted to a numerical value using label encoding or one-hot encoding.

One-hot encoding: It represents categorical variables as binary vectors. As we are not sure of the data size and considering at least 1000 lines for a Vendor it grows the matrix size exponentially.

Label Encoding: It gives each unique feature an increment value. As we are using Naïve Bayes Algorithm there is no calculation like average etc. So we can use label encoding for preprocessing.

**Implementation**

Following are the sequence of steps on how this is implemented:

* Read the training data from requests as json object.
* Read the prediction data from requests as json object.
* Use label encoder and transform all the features.
* All the predictions data have LineID column populated which will be used to map the charge account predicted.
* Move the charge account id to a different data frame so that it can be treated as Y values.
* Split the dataset using cross validation.
* Fit the model using Naïve Bayes and predict the score.
* Do the inverse transform from the predictions so that actual charge account is retrieved.

**Refinement**

Without filtering the data based on vendor, model results were poor. Filtering data based on Vendor and then applying the data to a model gave better results.

As this is a sample set, we couldn’t come with a vendor specific data. In a real environment customers will chose the features needed to predict charge account.

**Results**

**Model Evaluation and Validation**

Model score without filtering vendors was 16%. This is very low which lead to me removing features, which are not related to vendor. This improved the score significantly.

I see wide range of results from 32-100%. Customers would have to come up with better rules in order to improve predictions for low score vendors. Instead of specifying charge accounts randomly, they need to follow a consistent process so that predictions will get better.

**Justification**

In real environment we have the capability to turn charge account predictions at a Vendor level. This will help customers to turn charge account predictions for Vendors where scores are high. For the vendors with low scores, predictions will get better and accurate when a consistent process is followed and new Invoices are processed with consistent charge accounts.

**Conclusion**

**Free-Form Visualization**

**Reflection**

Following are the steps followed for this project:

* Read the training data and predictions data from json requests. For the project I am reading it from csv file.
* Converting each feature to numerical encoding. As the predictions set is in a different object, it was tricky to use the same label encoder, which was used for training set. This is to make sure training and predictions data have same numerical values for each character.
* Filling Na or NaN values with 1. I considering by removing them completely but scores were very low. It made senses to just replace them with one value so that algorithm will treat is one combination.

**Improvement**

It would be better to have a model being generated and updated periodically. Instead of creating the model for every new Invoice, it would make sense to reuse the model for every new Invoice coming in. Advantages with this approach would be:

* Low Turnaround time for the predictions
* Less computing power for model creations

**Domain Background**

Charge Account Prediction for Invoice lines is the project I would like to work on. Typical medium and large business companies have an ERP system and an Accounts Payable team who receives Invoices from Vendors, create Invoices in the ERP system and pays Vendors against the Invoices created. Based on the services/goods provided by Vendor Accounts Payable team will specify appropriate GL Account (Charge account) on each line on the Invoice. This is a repetitive process every day and specifying the appropriate GL Account comes with experience and memory.

**Problem Statement**

Specifying Charge Account on Invoice line is a repetitive task where several parameters define the final charge account. These final values don’t change and users keep on doing the same task over and over. When AP receives an Invoice from Vendor, it makes sense to look at the history of Invoices from the same Vendor and predict the charge account. This will save a lot of time in processing Invoices and will increase the overall productivity.

Charge account is dependent on several factors like Vendor Country, Payment Method, and Payment Terms etc. These factors are independent from each other so Naïve Bayes algorithm might be a good solution for this problem.

**Datasets and Inputs**

When a new Invoice is received from a vendor, all the old Invoices lines for the same vendor can be considered as training set. Current Invoice lines will be considered as prediction/test set.

I have 673,150 records available for training. All the features are characters so one hot encoding is necessary for all the features. Will use only last one-year data, using more or less than that might result in over fitting or under fitting.

**Benchmark Model**

For every new Invoice when user manually specifies a charge account, there is an entry in audit table, which has a record for old and new value. With this algorithm predicted results we can compare the value predicted with the charge account value for recent Invoice in audit table. The count of entries in field audit table will be less for a new Invoice when charge accounts prediction is successful.

For the features of test set, count of distinct charge accounts can be fetched and compared to predicted charge account. In most of the cases, there should be only 1 or 2 different charge accounts, if the results are not correct it simply means we didn’t account for a feature or a mistake in data preprocessing.

**Solution Statement**

Naïve Bayes algorithm could be used to predict charge account, as occurrence of each feature in the input data set is independent of other features. New Invoices from Vendors keeps coming in and model gets better with more available training data.

Users spend a lot of time in specifying a charge account on each Invoice line, which is time consuming. Auto predicting charge accounts will decrease the overall processing time and improves the efficiency.

**Evaluation Metrics**

We can measure the model by comparing the predicted charge accounts and charge accounts of the old Invoices. We only look at previous Invoices for the same Vendor as comparison metric if needed.

**Project Design**

1. N features are needed to predict charge account, N value varies from customer to customer based on the business requirement. This will be crucial to come up with a design where these N features can be dynamically changed from customer to customer.
2. Python script should be able to access these features and charge account id values either by reading from Oracle database directly or CSV file or from JSON requests.
3. These features could be characters so it is needed to convert them to a numerical value. Converting this can be done using label encoder or hot encoding.
4. Y value which is charge account are characters too so converting them to numerical is important and have the capability to get the character value for every numerical value is needed.
5. Model score is important which will help users to understand how good or bad a model is.
6. Naïve Bayes algorithm seems to be the right choice for this. It would be interesting to see the results of Logistic Regression too.