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.

This is a real business use case problem and there are several solutions available in the market today to solve these problems. Most of the solutions are using rules engine, a Google search for rules engine will give us several companies who do this. One famous rules engine used in FinTech space is Oracle BPM. It's a complicated and painful process to build, deploy and maintain BPM. To simplify, it's a graphical representation of nested if conditions and memory expensive.

These conditions rarely change, so most of the cases will be handled with a pre-defined set of rules. In cases where rules change, capability to turn off charge account predictions will be useful.

It is worth considering Machine Learning to solve this problem and replace heavy weight and memory expensive rules engine. Algorithm should look at the history of data and predict the final charge account.

Problem Statement

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

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

All the features included in this dataset contribute to the final charge account. Each feature is independent of other so Naïve Bayes classifier could be the right algorithm

for this problem. Other algorithm worth considering is Logistic regression but Naïve Bayes performs better as we assume the independence of each feature.

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 incorrectly predicted, it can be excluded. As the problem we are trying to solve is to reduce the time and improve the overall automation, we are interested in knowing only True Positives and exclude others for below reasons:

- True Negatives: Model couldn't predict the charge account because there are new values found in the data set, which were not available in Training set. This makes the users manually specify charge accounts on each Invoice line reducing automation.

- False Positives: Model predicted a charge account but it predicted incorrectly. This will make users remove the existing charge account and populate new values.

- False Negatives: Model couldn't predict a charge account despite of having sufficient features or model predicted a charge account with low score. This will make the user to either specify charge accounts manually or review the predicted results. This will increase the processing time of Invoices, as users need to validate the model results.

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. Features used for this project are entity id, vendor id, ship to, document type, and pay group.

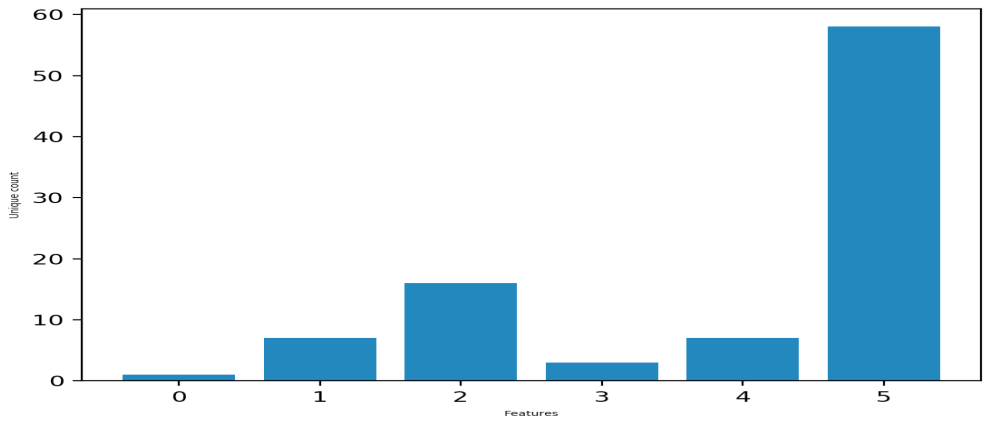
Ideally charge accounts are determined using 3-5 features. Features can be characters or numbers, so we will use label encoding so that all characters will be converted to numbers. 60,849 records are available for training. A split of 20% is used for testing. Data set has the following fields:

- Entity Id: Each Company has a unique id. As each company structure is different, it is ideal to use this as a feature.
- Vendor ID: This application is for Invoices without Purchase Orders. Few Vendors support only Purchase Order Invoices so including vendor id in the data set helps in getting better results. For PO Invoices charge account comes from PO so this application wouldn't be necessary for Invoices with Purchase Orders. In rare cases charge accounts are specified manually for PO Invoices too.
- Ship To: This is the ID, which identifies where the goods/services delivered from the Vendor. Charge Accounts will tell a company where the costs are occurred across the entire organization. So this is crucial to predict charge account.
- Document Type: This is usually an Invoice or Credit Memo. Companies usually have different charge accounts for Credit Memos.
- Pay Group: Grouping of Invoices also derives the charge account.

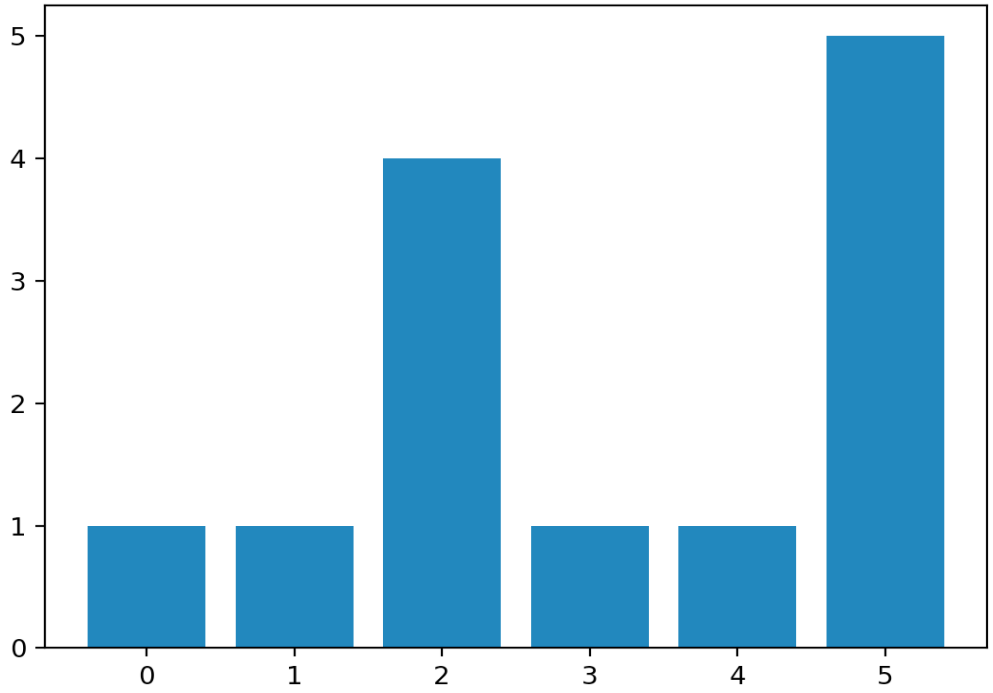
Exploratory Visualization

The plot below shows the count of each unique feature from the entire dataset. Index 5 is the final value predicted. The numbers of unique final values are very high compared to other features, which might result in a lower accuracy. Following is what each feature represents:

Index 0 – Entity Id
Index 1 – Vendor Id
Index 2 – Ship To Id
Index 3 – Document Type
Index 4 – Pay Group Code
Index 5 – Charge Account Id



When the data is filtered based on Vendors, count of unique features looks like below:



For this vendor only 5 different combinations are possible, running a model with features just for the specific vendor would give better results. Following are the model scores for each Vendor in dataset:

Vendor 1 (23348)	-	100%
Vendor 2 (462)	-	68%
Vendor 3 (959)	-	31%
Vendor 4 (5628)	-	68%

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.

Formulae for Bayes Theorem:

$$P(h/d) = (P(d/h) * P(h)) / P(d)$$

P(h/d): Probability of hypothesis h give the data d.
P(d/h) : Probability of data d given that hypothesis H was true.
P(h): Probability of hypothesis h being true regardless of data.
P(d): Probability of data regardless of hypothesis.

Bayes Algorithm calculates the probability of each feature, which occur independently, and all the features contribute to the final outcome of the charge accounts. It is simple and can handle huge data loads. As the algorithm calculates the probability and no statistical calculations are involved like Standard deviation etc, it is perfect algorithm to this problem where all features are treated as characters.

Benchmark

We have an audit for each Invoice line and charge account predicted. If an Invoice has 5 lines and user manually specified a charge account on the line we would see 5 entries in audit with line Id, old value as null and new value with the charge account. Further Audit data is used to calculate the automation % for each Invoice.

$$\text{Automation \%} = (1 - \text{Count of Audit Entries} / \text{Total Fields}) * 100$$

With this application the number of audit entries decreases and automation score should improved.

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 Line ID 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.

The training data and prediction data was in two different objects. Applying a label encoder on training data is simple but using the same encoder on prediction set, which is in a different object to get the transformed values, is tricky. If label encoder is initialized we will get a different categorical variables and results were very bad. To solve this problem dictionary is used for each feature label encoder and same dictionary is used to transform and inverse transform for the final charge accounts on predictions set.

Implementing security is critical for any application. Token validation using AES encryption and Token expiration is implemented so that all the requests are trusted and validated. This python application is called from a java application; current time in milliseconds is calculated both in java and python. As long as the difference between two times is less than 60 seconds we assume token is not expired. Then a fixed key is used to decrypt the token to look for data format and as long as the data is in a expected format we assume token is from a trusted source.

Refinement

Few features might have null or Nan values. Initially model was built by removing all rows, which have any Nan values. It reduced the data size from 365088 to 221694. Model score is 31.2 %. Even if one feature is null it removed the entire row.

Next attempt was to fill null or Nan values with 1, so that we have more data to train. This may not be a right thing as a particular feature might be always null for one vendor and mandatory for another vendor. So replacing all null values with '1' might lead to confusion. Model score is 23.3%. This is very low score compared to previous model.

It is necessary to not remove the data because of null or Nan values as we are losing data for training, so filtering the data for every vendor and building the model might help. We will only look for features related to the specific vendor and use the vendor data to build the model. With this approach scores are:

Vendor	Model Score
23348	100
462	68
959	31
5628	68

Results

Model Evaluation and Validation

Final model and data cleanup is chosen as it performed better. Following would be the final model description:

- Naïve Bayes algorithm is the right solution for this problem as each feature occurrence is independent of other feature.
- Don't remove features with null or Nan values. Rather fill them with value '1' so that we are not losing data for training.
- Filter the data based on vendor to avoid over fitting.
- Label Encoder object should be same for training and prediction set so that transform and inverse transform will use the same categorical variable.
- Split the data into training and test set so that model score can be calculated and user can see the confidence score before applying the results.
- Request comes with a line id (line number) in prediction set. Map the line id from request and results from model so that we know the charge accounts predicted for each line. It is possible to have different charge accounts on an Invoice but at most only one charge account on an Invoice line.

This model can be called as Robust for the below reasons:

- Customer can determine the count of features.
- Customer can pick the features to determine the charge account.
- We see results up to 100% for few vendors, optimizing the features by following few rules consistently will help in getting these high scores consistently.

Justification

Audit data would look like below before this application:

	LINE_ID	FIELD_NAME	OLD_VALUE	NEW_VALUE
1	107979	CHARGE_ACCOUNT_ID	(null)	01-210-6100-0000-000
2	107980	CHARGE_ACCOUNT_ID	(null)	01-210-6100-0000-000
3	(null)	INV_NUMBER	2600220	RV_0628_1

After the application is built it looks like below:

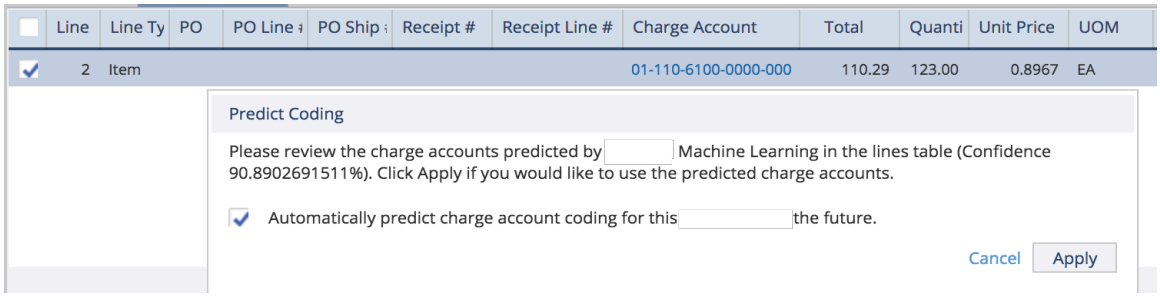
	LINE_ID	FIELD_NAME	OLD_VALUE	NEW_VALUE
1	(null)	INV_NUMBER	2600220	RV_0628_2

As we can see from the images entries in audit table are reduced and it will improve the overall automation.

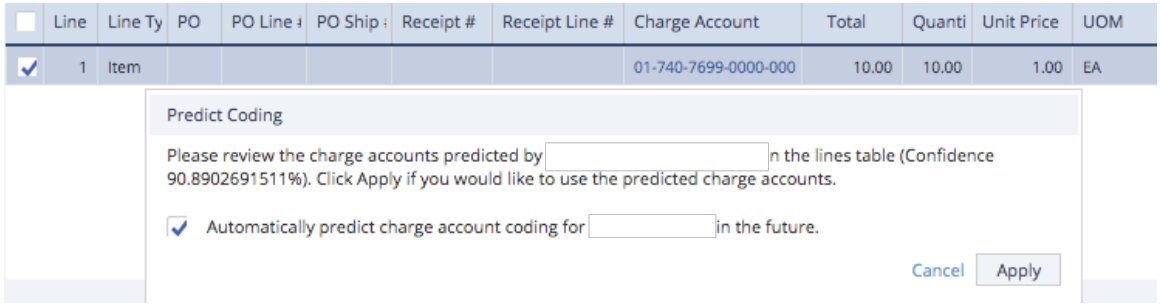
Conclusion

Free Form Visualization

Results from the final application:



Charge Account Prediction with 90% confidence for Vendor 'A'



Charge Account Prediction with 90% confidence for Vendor 'B'

We can see models with scores of above 90% for different Vendors. There could be few Vendors with low scores and we can expect scores to improve with more training data.

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 null or Nan values with 1. Considered by removing them completely but scores were very low. It made sense to just replace them with one value so that algorithm will treat them as 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
- Building a perceptron might be worth considering as an improvement. It would be tricky as there would be a lot of noise with several vendors data but a model which filters data by a vendor in first layer might a good solution.

References

1. https://machinelearningmastery.com/naive-bayes-for-machine-learning/