**Udacity – Machine Learning Nanodegree**

**Capstone Project Report – Instacart Market Basket Analysis**

1. **Definition**
   1. ***Project Overview***

Instacart is a same-day grocery delivery startup offering delivery in as little as one hour from a variety of local stores. Focused on delivering groceries and home essentials, Instacart already has over 500,000 items from local stores in its catalog. Customers can choose from a variety of local stores and can mix items from multiple stores into one order. This makes Instacart a unique player in e-commerce space where it combines online ordering experience and aggregates real world shoppers to procure and deliver the items.  The Instacart business model is really unique as the app enables customers to browse all the items available at local grocery stores and hence it is a mix of the online experience offered by Amazon, the In-Store delivery model, similar to most retail stores and same day delivery with crowd-sourced delivery mechanisms, somewhat similar to Uber's model of crowd-sourcing transportation. This also gives access to shopping data aggregated across retail stores across the country, which isn't there with anyone else and thus introducing problems which are unique.

In this project I am trying to solve some of the daily challenges faced by the data team at Instacart. I need to analyze the order history (in millions) of Instacart customers for thousands of products being sold through the Instacart platform, to predict what would be the next product that a customer could be buying. The dataset is obtained from kaggle.com at this link - <https://www.kaggle.com/c/instacart-marketbasket-analysis/data>

The data is divided into multiple relational csv files detailing the products, categories and customer orders over time. There are about 3 million orders from more than 200,000 users. The data dictionary with all the csv files and fields in them is described at the following link: <https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b>

The data set essentially consists of orders file with information about all the orders in the data set (about 3 million rows), each order could be classified as prior (for prior orders by a user), train (for training set orders which are ) and eval (evaluation set for submission to Kaggle). There are a couple of files orders\_products\_train (with more than 1 million rows) and orders\_products\_prior (with more than 30 million rows) detailing what all products are associated with each order part of the orders data file. There are three more files specifying products, aisles and categories separately. Given the computation challenge posed by the amount of data provided to work on this project, I’ve tried my best to use a combination of advanced hardware in AWS and software techniques to work out a reasonable solution at a low cost.

* 1. ***Problem Statement***

As quoted in the problem description section "*In this competition, Instacart is challenging the Kaggle community to use this anonymized data on customer orders over time to predict which previously purchased products will be in a user’s next order, or the products they'll try for the first time or add to cart during next session.*" The inputs for this problem will consist of features provided in the training data set which include order history per customer and what all products were part of the order. As a first step for working towards a solution I performed some exploratory data analysis along with preprocessing on the data if required. The strategy adopted for this project is to use the existing data provided to evaluate a couple of benchmark algorithms such as SVM and Random Forest and finally use one of the highly popular classification algorithms XGBoost and/or LightGBM (from Microsoft) and see the improvement in evaluation metric. If required I also considered engineering features based on the features that are already provided and reassess if the evaluation metric could be improved, before making a final recommendation as the algorithm of choice.

* 1. ***Metrics***

We can choose from multiple metrics but given that this can be treated as a classification problem we can use related measures such as Accuracy and F-Scores.  The F-Score defined by the following formula:

**F1 = 2 \* (precision \* recall) / (precision + recall)**

F1-score factors in both precision and recall to evaluate the performance of an algorithm and hence it is an appropriate choice for a metric for this project.

1. **Analysis**
   1. **Data Exploration**

Exploring the data to understand the input space and identify any anomalies is the first important step. The data set consists of five csv files. Here are my observations as I studied their structure correlating with the description provided by Instacart.

* + - * orders.csv – has the following structure and fields. Some of the important features that could be considered are order\_dow (day of week), order\_hour\_of\_day and days\_since\_prior\_order (representing difference between successive orders for a specific user identified by a user\_id field).

|  |
| --- |
|  |

eval\_set is another important field which differentiates a record among training, testing and prior as shown below. The test portion of the data set is used in the Kaggle competition. In this project I’ll be using prior as training data and train data as test data. An abnormality I would like to mention about this data set is that the orders\_df data frame from orders.csv has days\_since\_prior\_order set to null for any user's first order.

|  |  |
| --- | --- |
|  | Prior represents all the historic orders  Test is used for Kaggle evaluation  Train is used for training purposes but since it has information about product reordered for the given orders hence I’ll be using it for testing purpose. |

* + - * order\_products\_train.csv, order\_products\_prior.csv

Both the files, order\_products\_train.csv and order\_products\_prior.csv have a similar structure as shown above. Although \_prior file has more than 33 million records detailing the 3.2 million orders in the orders.csv with product details and an important feature ‘reordered’ which is actually the target field for prediction. The \_train file has about 1.3 million records showing the details of a customer’s last order with product details and reordered field value as 1 or 0. Another significant feature is add\_to\_cart\_order which shows the order of putting the products in the shopping cart.

|  |
| --- |
|  |

* + - * products.csv

This file has information about all the products in the Instacart catalog. There are about 49,688 products in the catalog. The data set also shows the aisle and department to which a product belongs. The sample structure of the file is:

|  |
| --- |
|  |

* + - * aisles.csv – The aisle file has about 130 records for aisle\_id and the aisle name. The aisle\_id 100 is mentioned as ‘missing’ which could need further pre-processing.
      * departments.csv – The departments file has 21 records with an id and name of the department. Department\_id 21 is mentioned as ‘missing’, which could need further pre-processing.
  1. **Exploratory Visualizations**

Following is a series of selected visualizations along with a brief description for specific features readily provided with the data set

|  |  |
| --- | --- |
| Orders by day of the week, where 0 represents Sunday, the busiest day on Instacart and the next one being Monday. |  |
| This chart shows order frequency by time of day through 7 days. The traffic is high from 9 am to 5 pm time frame. |  |
| Looking at only weekend orders, there is a slight variation and people start ordering in high numbers after 10 am and up to 4 pm. The window is smaller than rest of the days but number of orders is higher. |  |
| This chart shows days since prior order for a specific order. Apart from 30, which seems to be an artificial limit set by the Instacart team, the other highest number is 7 which could be consistent with weekly ordering habits of buyers. |  |
| This chart shows a heatmap of when the Instacart app is most busy, during Sunday morning and afternoons and Monday mornings. Over rest of the week the ordering is pretty consistent in 9 to 5 time frame. |  |
| These are the top 20 aisles by count of orders of products part of those aisles. Fresh produce is highly popular aisle, probably because it is fast moving and perishable good |  |

* 1. **Algorithms and Techniques**
  + I have used Support Vector Machines, Random Forest Classifier and Microsoft’s LightGBM, initially with the feature set that was provided with the data set. Later I’ve tried to engineer more features to enhance the model performance and done a GridSearchCV cross-validation on a parameter set for each algorithm to get the best features along with the best parameters for the optimal performing algorithm. Below are some of the reasons which I considered to chose the algorithms I’ve used for this project:
  + Support Vector Machine – is one of the popular classification algorithms with flexibility enabled by kernel methods. Although it can be inefficient with the amount of data we have which I have tried to address by limiting the amount of data fed to the algorithm.
  + Random Forest Method – is one of the classification algorithms where an ensemble of decision trees with different samples to build multiple models (called weak learners) and make predictions. The final prediction is a function of each prediction which could simply be a weighted mean.
  + Light Gradient Boosted Method – A recent challenger to the XGBoost with a unique way of splitting the decision tree, leaf wise instead of depth wise in case of XGBoost. The leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Further, the algorithm has much faster training speeds, lower memory usage, compatibility with larger data sets which all are important considerations given the data set size and the limited hardware capability I’m dealing with.
  1. **Benchmark**
  + I have chosen the scores reported on the Kaggle leaderboard for this competition. The top scores are in the range 0.404 to 0.409. Since these scores are based on the eval\_set = ‘test’ and are scored by Kaggle, I would assume there would be some over fitting in my solution and hence I think the benchmark should be higher, at around 0.6

1. **Methodology**
   1. **Data Preprocessing**
   * Overall the data seemed to be pre-processed by Instacart, ready for the Kaggle competition, except a couple of points which I mention below.
   * As mentioned previously while discussing the orders.csv file, I had mentioned that for days\_since\_prior\_order the value was null for the first order which I filled up with a mean value of this feature.
   * I also had pointed about the aisle and department data frames which contained ‘missing’ category. I explored it further seeing that about 1258 out of about 50,000 products were part of this category, which I chose to leave as it is.
   * I combined products, aisle and department data sets into a single data set on product\_id. The new products data set was further combined with orders\_products\_prior and order\_products\_train files to create 2 new data sets. I chose *prior* file for training purpose and *train* file for testing purpose as first part of preparing the data set for model development.
   * For training purpose, for the first part of model exploration I just used pre-defined feature set which include the following - ['user\_id', 'order\_number', 'days\_since\_prior\_order', 'order\_dow', 'order\_hour\_of\_day', 'product\_id'] and the [‘reordered’] field was used as a target for model prediction and evaluation.
   1. **Implementation & Refinements**
   * I used this pre-processed data to find out how the models performed while searching for the best set of parameters with GridSearchCV cross validation

function.

* + Since the data set size is huge and it wasn’t possible to use a very high end computing platform for me, I used a standard AWS machine with 64 GiB RAM at a reasonable cost to train the models and had to limit the data I fed into the algorithms.
  + I have reported my finding on the model evaluation and validation on the first part of model development in the Results section under Part 1.
  + Next step was to perform more refinements in terms of limited feature development based on intuition. Since, I expected this to be computationally expensive, I limited the feature development to few
  + Following is the list of features developed along with a brief description of each
    - 'aisle\_id' – represents the aisle from which the product is ordered
    - 'mean\_hod' – this is average time (of all the instances of ordering a product) when a specific product is ordered
    - 'mean\_dow' – as above this is the average of all the days of week for ordering a specific product
    - 'is\_organic' – this is a flag for any product that has the word *organic* in the name
    - 'user\_prod\_count' – this represents the count of user\_id and product\_id combinations
    - 'user\_prod\_rate' – of the total orders by a user, this represents the number of times a specific product was ordered by a specific user.
    - 'order\_carts\_since' – number of orders since last order of a product
    - 'order\_carts\_since\_1st' – number of orders after 1st order of a product
    - 'cart\_position' – a simple mean of rank of every product in a cart for a user+product combination
    - 'order\_dow' – taken straight from orders.csv
    - 'order\_hour\_of\_day' - taken straight from orders.csv
    - 'days\_since\_prior\_order' - taken straight from orders.csv
    - 'user\_norders' – number of orders by a user
    - 'user\_reorder\_rate' - number of products reordered out of total orders by a user
    - 'user\_order\_dt' – average of days since prior order for a user
  + The results of models based on the new feature are also reported in the Results section under Part 2.
  + I was expecting that feature engineering would be helpful in improving the classification and a better accuracy, but just the opposite of what I was expecting happened as evident in the results in Part 2.
  + Feature engineering could be considered as a major refinement step that I tried to improve the model. The major challenge was again working with the huge amount of data with limited hardware to manipulate the data frames and develop the features.
  + I also tried to limit the data by discarding about 90% of the data, but since it was a random elimination hence the final combination of different data sets turned out to be very sparse. The final feature set after removing all the NaNs that got introduced resulted in only about 10,000 rows of data to work with. Hence I had to work with entire data set for feature development.
  + Finally for Part 2, I split the data in 70:30 training:testing combination using test\_train\_split from sklearn.
  + I think the LightGBM model from Part 1 is my final solution given that it was able to outperform the benchmark I had set and also performed better than SVC and Random Forest models in both Part 1 and Part 2 of the solutions I attempted.

1. **Results**
   1. **Model Evaluation, Validation & Justification**

As I have mentioned in the previous section, I have done a couple of experiments when training the same models and both are reported in Parts 1 and 2 of this section.

* + **Part 1**

|  |  |  |
| --- | --- | --- |
| **Model** | **F1 Score** | **Notes** |
| SVC | 0.9033 | Using the RBF kernel and Shrinking parameter set to True |
| Random Forest | 0.8284 | Using 50 estimators, 4 features, max depth of 15 and entropy criterion |
| LightGBM | 0.9033 | Using binary classification parameter, 10 rounds, binary logloss as the metric, 256 leaves, max depth of 1 |

* + **Part 2**

|  |  |  |
| --- | --- | --- |
| **Model** | **F1 Score** | **Notes** |
| SVC | 0.0032 | Using the RBF kernel and Shrinking parameter set to True |
| Random Forest | 0 | Using 20 estimators, 4 features, max depth of 10 and gini criterion |
| LightGBM | 0.043 | Using binary classification parameter, 10 rounds, binary logloss as the metric, 256 leaves, max depth of 1 |

A general conclusion can be derived that LightGBM seems to be much more efficient, which I observed while performing the fit, than the other 2 classifiers. Overall LightGBM and SVC have performed well compared to Random Forest although the F1 scores in 2nd part are too low compared to the benchmark. For Part 1 the important features identified by LightGBM are order\_number, day\_since\_prior\_order. For Part 2 the important features identified are 'order\_carts\_since' (number of orders since last order of a product), 'user\_prod\_rate' (this represents the number of times a specific product was ordered by a specific user), 'order\_carts\_since\_1st' (number of orders after 1st order of a product) and 'user\_prod\_count' (this represents the count of user\_id and product\_id combinations).

The final solution in my opinion would be the LightGBM from Part 1 given the efficiency, F1 score and ability to achieve it with a limited data set of only about 20,000 records.

1. **Conclusion**
   1. **Visualization and Reflection**

|  |
| --- |
|  |

Feature importance ranking from Part 1

For Part 1, the pre-defined features were used after merging all the data sets provided. Order\_number is a bit counter intuitive but days\_since\_prior\_order is a strong determinant of the reorder prediction if we go by intuition. Another important and intuitive determinant is the order\_hour\_of\_day which was evident in the exploratory visualization I plotted in the initial section of this report.

For this project, I worked on the usual Exploratory Data Analysis with the data provided, chose models to work on the problem and evaluated each. I also tried to engineer a good 15 feature Features which involved multiple slow grouping and aggregation functions. Feature development was done in hope of improving the performance, only to learn about the significance of domain expertise.

Working on such a problem definitely requires infrastructure that a large corporation can be easily afford and employees working there can use the infrastructure to solve such a problem. But for someone like me armed with a humble laptop and more recently Google Colaboratory, it is very difficult unless I am ready to spend on cloud infrastructure to work on such a problem. This was a major difficulty for me and the reason it took so much time for me to reach this almost final stage for this project. But I learned how to be more efficient with whatever you have, work with smaller sets of data, manage memory at each step of the notebook, how to use cloud most efficiently. Another important take away for me is that one needs to get to a substantial size of data, which may not be too less, but neither 100% of the data. Going with the 80:20 rule, a fraction of total data should be good enough to solve a problem satisfactorily, but one has to determine what is that fraction of data and the rest can be discarded.

|  |
| --- |
|  |

Feature importance ranking from Part 2

A striking observation I would like to repeat is that feature engineering didn’t lead to improvement in performance, rather a decline. Hence it is highly important to think through the features to be included in a model, and for that domain knowledge of the industry is highly desired. Features seem to be the most important determinant of the success of models. One may not be able to come up with the right set in a short span of time and requires multiple iteration of applying thought, experimentation and incremental improvement.

* 1. **Improvement**

There are multiple improvement areas that I identified for me while working on this project. Here are some of them:

Working efficiently with large amounts of data, specially when aggregating and grouping data to develop new features. Current implementation is too slow and time + resource consuming.

Able to come up with useful features, probably by discussion with someone who has some experience in the domain or just brainstorming

Once optimal features are identified, try to add dimensionality reduction as well using techniques such as PCA.

Try to get access to powerful infrastructure through multiple avenues.

1. **References**:
   1. “The Instacart Online Grocery Shopping Dataset 2017”, Accessed from <https://www.instacart.com/datasets/grocery-shopping-2017> on 2/11/18
   2. Kaggle Feature Prediction Competition - <https://www.kaggle.com/c/instacart-market-basket-analysis>
   3. Model evaluation - <https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/>
   4. <http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>
   5. <https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost>
   6. <https://www.kaggle.com/nschneider/gbm-vs-xgboost-vs-lightgbm>
   7. <https://www.kaggle.com/c/instacart-market-basket-analysis/leaderboard>

**Submission**

* proposal.pdf  + review link in the student submission notes. [https://review.udacity.com/#!/reviews/1024537](https://review.udacity.com/" \l "!/reviews/1024537)
* A project report (in PDF format only) + five project development stages. (nine to fifteen pages.) (not iPython Notebook as PDF) + why you made the choices you made?
* Code - iPython Notebooks
* A README file - briefly describes the software and libraries + references to supporting material + necessary instructions + datasets, images, or input files