Team InstaMarketing Final Paper – Predicting Customer Order Date


**Georgetown Data Science Certificate Program**
**Summer 2019 – Cohort #15**

InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

## TABLE OF CONTENTS:

# Team InstaMarketing Final Paper – Predicting Customer Order Date

## Section I: Project Background

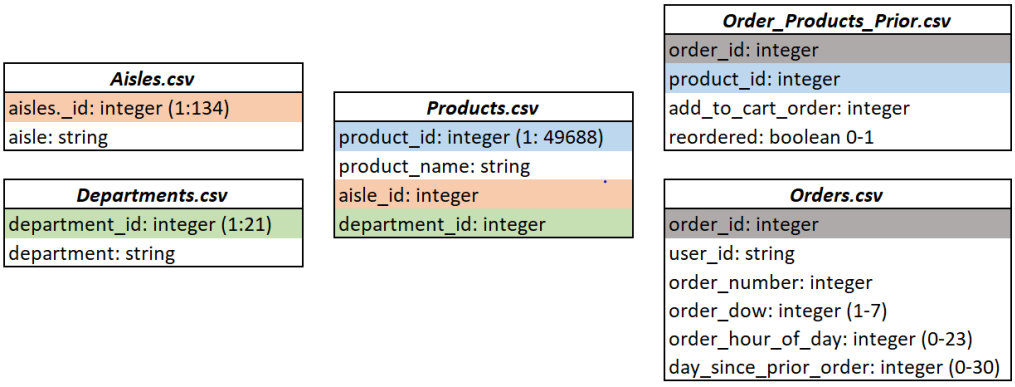
The retail domain comes with many challenges as a business. Managing inventory, balancing operational efficiencies, and recruiting high level talent are just a few of numerous tests businesses face as they compete in one of the most challenging markets to sustain consistent profitability. Add in the fierce competition retail businesses face and the question of where to invest resources to maximize return becomes more ambiguous. One direction many companies turn to for growth is reaching as many new customers as possible. According to a 2016 survey of 168 Chief Marketing Officers of U.S. companies the average marketing amount spent was 10% of overall budget with some approaching as high as 40% (smallbusiness.chron.com). More often than not these inflated amounts being spent on advertising are geared towards attracting new customers as a way to increase revenues, however according to MarketingMetrics.com the success rate of selling to a new customer ranges from 5% - 20% while shifting the strategy towards selling to existing customers can yield a sale frequency of 60% - 70%. With these facts in mind, it becomes intuitive that a retail business should focus largely on customer retention as a key to future growth.

Our team aims to utilize a dataset from a large online grocery delivery service consisting of information on customer order history to predict when a customer, or user as it is defined in our data, will be likely to order again. With this information in hand the business will in turn be able to focus a calculated amount of their advertising budget on users they suspect will be likely to have larger gaps between their orders placed.

## Section II: Data Source Summary

The dataset our team utilized was originally introduced as part of a 2017 Kaggle competition in which Instacart, an online grocery delivery service business, volunteered user order history. The original goal of the competition was to predict the products that will be part of a users next order, however our team decided that based upon the business case outlined above, predicting the days until a user makes their next order would also be a beneficial analysis.

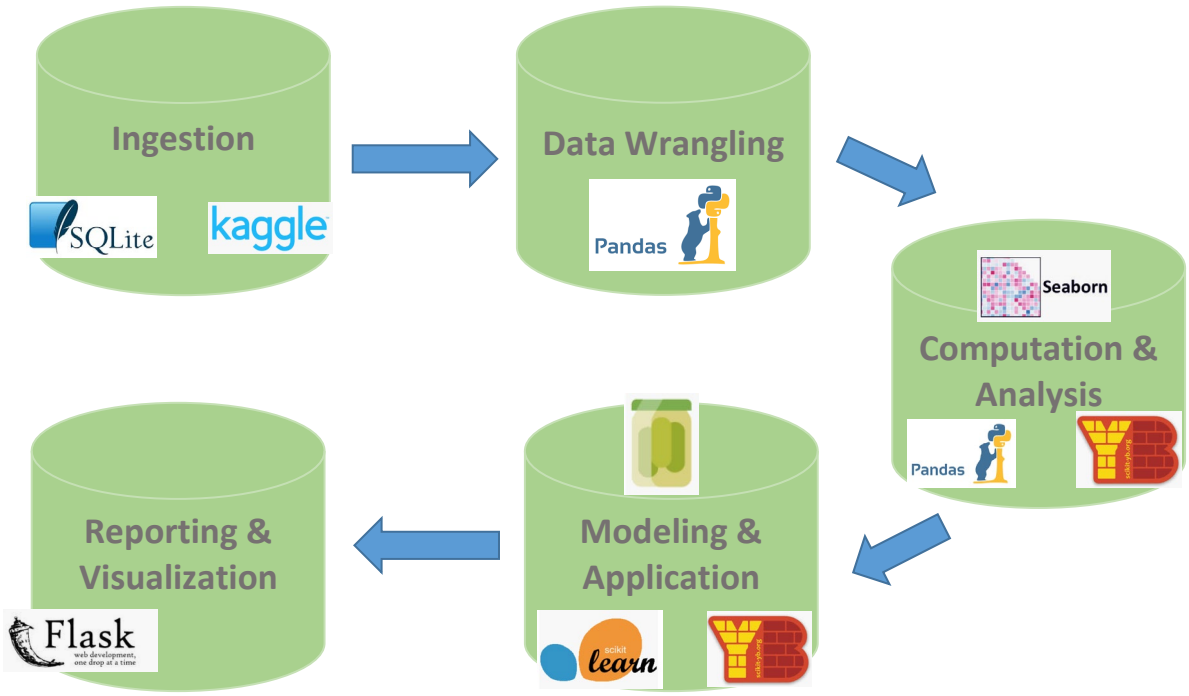
The dataset obtained contains a relational set of five csv files each with varying amount of information on a user's orders completed over time. In total, more than 200,000 users with over 3 million orders completed are represented in the data. The files are summarized below.




InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

# Team InstaMarketing Final Paper – Predicting Customer Order Date

With our dataset selected we then defined the necessary steps in transforming the raw user data into a final data product that could be utilized by the Instacart marketing team.




As highlighted above, we first ingested the data into a SQLite database with each csv file saved into a separate dataframe. From here our team began the wrangling process using Python's Pandas library. Given that the data was obtained from a Kaggle competition, we did not encounter many of the usual issues such as missing values, strings requiring conversion, or parsing requirements. Although these steps were not needed our team did need to merge the fields required for our prediction of the number of days until a user will order again. For our analysis we identified the Orders_Products_Prior.csv, Orders.csv, and Products.csv as having necessary data points for feature generation. Once the files were merged, we uploaded the dataset to a single SQLite database for storage and performed data verification.

```
print('Total unique Count:')
orders_df.nunique()
```

```
Total unique Count:

9]: order_id                  3421083
    user_id                   206209
    eval_set                       3
    order_number                 100
    order_dow                      7
    order_hour_of_day             24
    days_since_prior_order        31
    dtype: int64
```

InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

In the above table it was confirmed that the number of users, which will be defined as instances for our analysis, totaled 206,209. Additionally, we were able to see that the users total number of orders totaled from 4 to 100 and that days since prior order ranged from 0 to 31 days. From this point our team's goal was to perform computational analysis, modeling and analysis, before finally creating a usable data product.

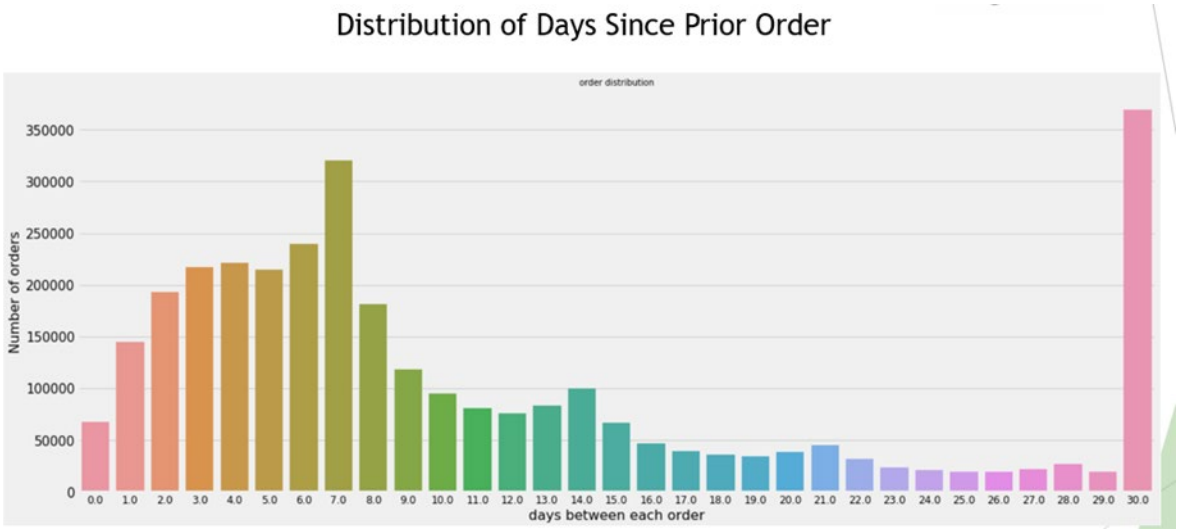
## Section III: Exploratory Data Analysis

Now that we had our merged dataset finalized, Exploratory Data Analysis (EDA) was performed to get a better understanding of the characteristics of the underlying data.

As we merged the dataset, we homed in on certain questions we felt were relevant to answer prior to generating our target and features for analysis:

- What was the distribution of the days since prior order for a user?
- How many orders did the average user have?
- What days of the week and time of day were most popular for users to order?
- What is the most popular department type of products ordered?
- How many unique products did a user order during the timeframe of our dataset?

First, our goal was to define what our target would be for our future model analysis. The below table was generated to gain a distribution of days since prior order for each user.



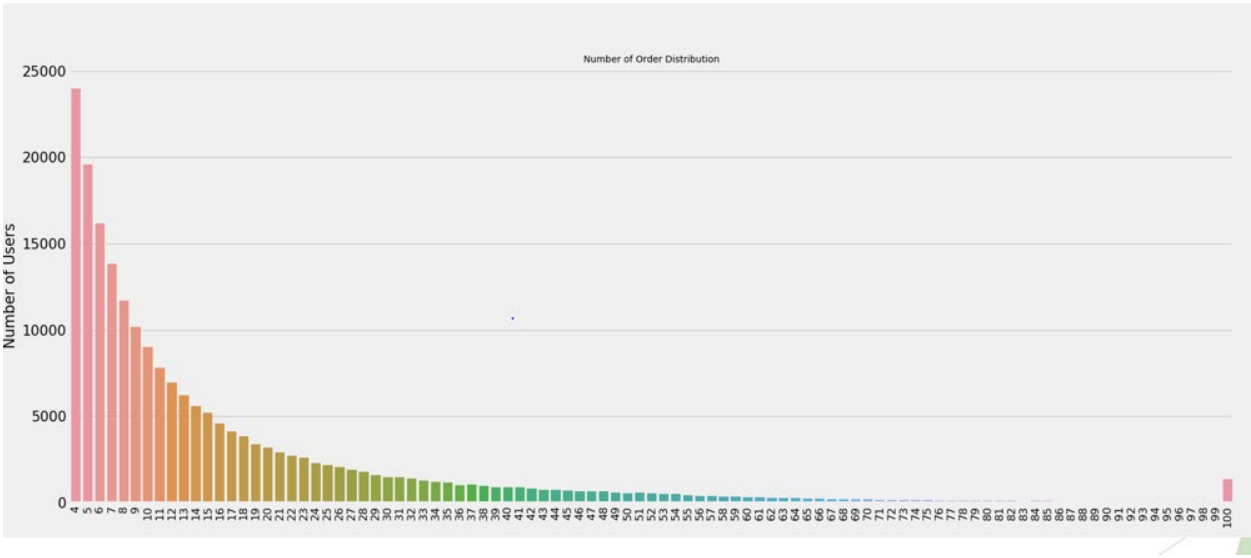
From this information, it became evident that the number of days since prior order for each user fell into the below distribution which became our target groups:

- A group of orders occurring within 0-7 days
- A group of orders occurring between 8-29 days
- A group of orders occurring beyond 30+ days

From here we performed additional analysis including the below findings.

An analysis of Number of Orders per User revealed order counts ranging from 4 to 100 orders each across a universe of 206,209 users (instances).

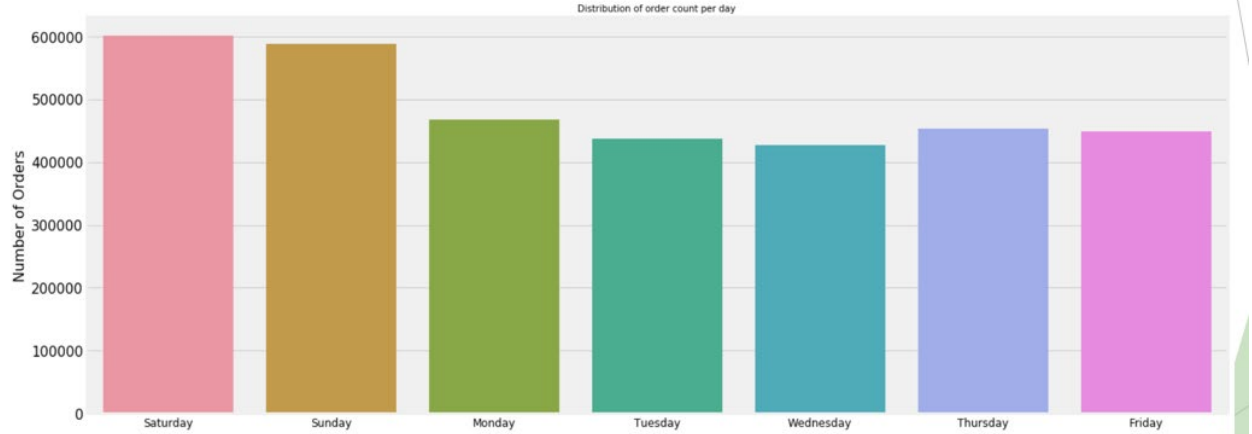
Number of Orders per User



# Team InstaMarketing Final Paper – Predicting Customer Order Date

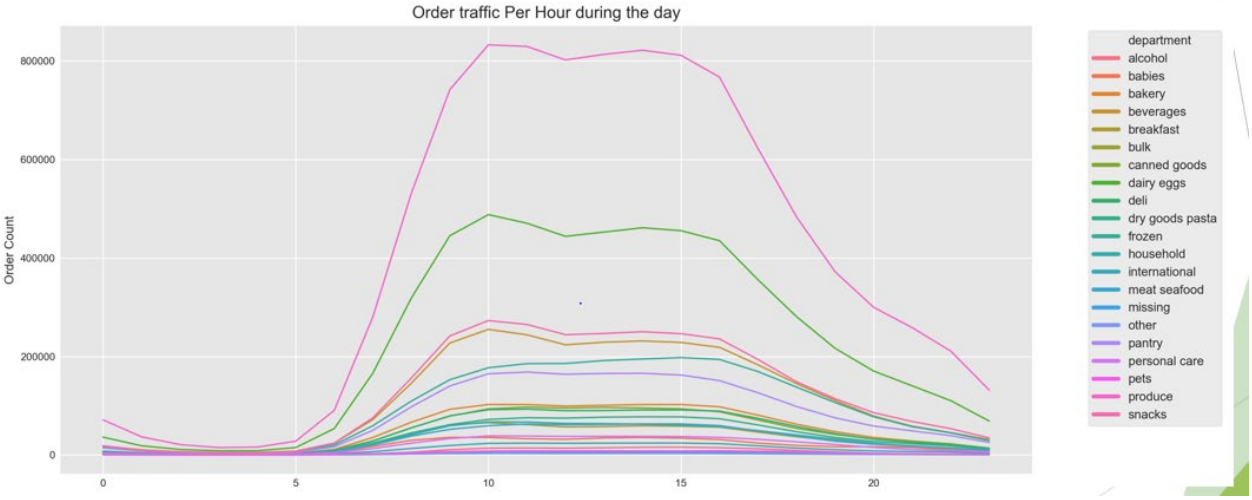
The Instacart data revealed that higher volumes of orders to be placed on Saturday or Sunday, with the least amount of orders being placed on Wednesday.

Order Count per Day of the Week



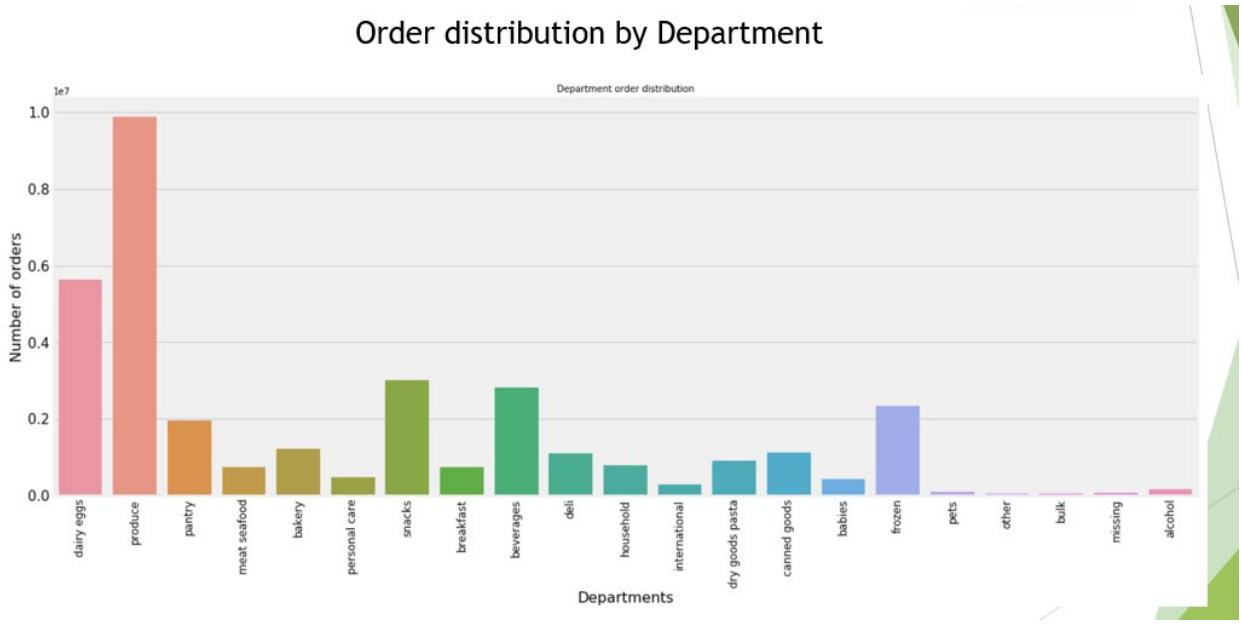

Our analysis of the Instacart data indicated that of the 21 departments within the dataset, Produce had the highest Order traffic per hour by Department type, with Personal Care receiving the lowest volume of traffic per hour.

Order traffic per hour by Department type



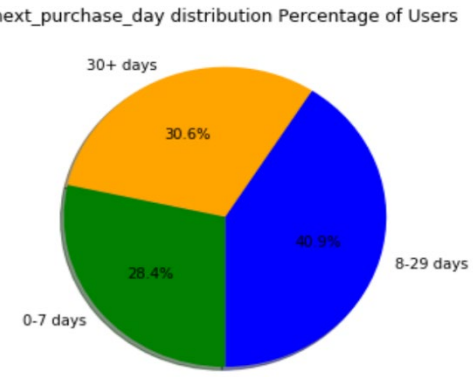

InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

In terms of the Department with the number of identified orders, Produce ranked highest, while Bulk and Other ranked lowest.




## Section IV: Feature Generation and Ranking

Through preliminary EDA, three distinct groups of next purchase day became evident. Target defined as next purchase day for users were grouped by 0-7 days, 8-29 days, and 30+days. The divisions gave each of three groups a relatively balanced weight to each other (ranged between 28%-41%), as reflected in the pie chart below:




Then each group was assigned an integer: 0-7 days assigned “0”; 8-29 days assigned “1”; and 30+ days assigned “2”.

<u>Feature generation based upon intuitive predictive qualities</u>

# Team InstaMarketing Final Paper – Predicting Customer Order Date

When it comes to select features to predict how soon would a customer comes back and makes the next purchase, it is intuitive to come up with the features like: what is the average days between a customer's any two consecutive purchases, how many items he/she usually purchased in one individual shop or which items he/she repeatedly ordered, etc. There are also some other features whose connection to target are more subtle but does exist, such as the time or day of week a customer prefers to make the purchase or even the first item he put into shopping cart.
We studied 42 features to predict the target including:

Number of inactive days per user;

Number of unique products ordered by user;

Number of reordered products by user;

Preferred day of week for order by user;

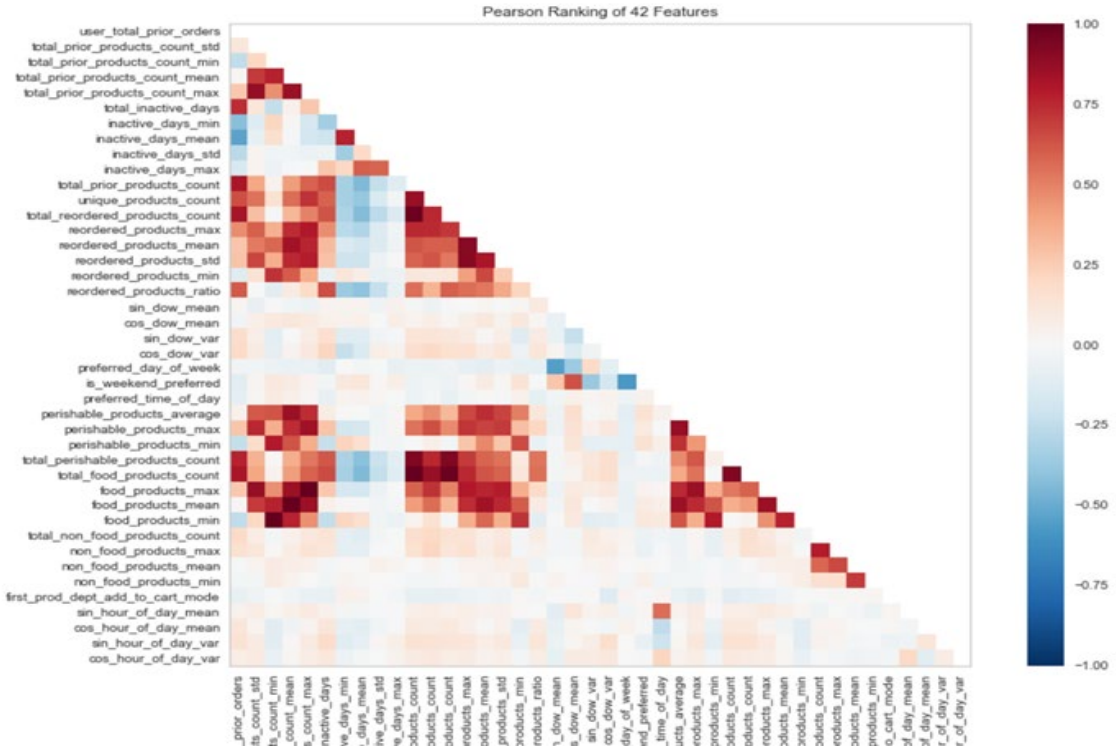
Preferred hour of day by user, encoded (Morning=0; Afternoon=1; Night =2);

Number of perishable items by user, with Bakery, Produce, Meat, Seafood and Dairy defined as perishable;

Number of food vs nonfood items purchased by user;

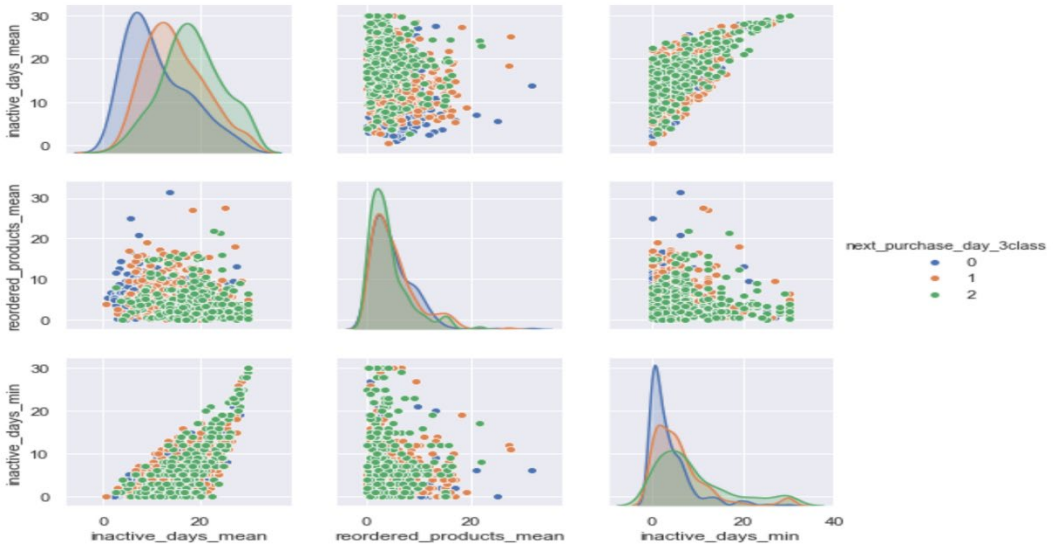
Mode of department type of first product added to cart by user; and so on.

The 2D Rank powered by Yellowbrick shows the correlation coefficient between any two features, and not a surprising to find many of them are highly correlated to each other.




InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

To illustrate the relationship between two features and relative to target, we used the plot chart powered by seaborn to depict the top 3 features that have the largest influence to predicted result:




Take the lower middle chart for example, in the horizontal axis, it tells us that the more items a customer reordered, the more likely that he would return sooner.

## Section VI: Data Modeling & Analysis

For this problem, we want to use the model which gives the highest accuracy. In order to choose the most effective model, we experimented with the different classification models available in sklearn library. Through our research, we learned that scaling our dataset can help to normalize the data within a range and may help in speeding up the calculations in an algorithm. Thus, we also applied different scaler techniques during our experiment with the different models.

To ensure the stability of our machine learning model across different datasets and avoid any overfitting problems, we used cross validation method with kfold=12 in our model selection. K-fold Cross validation is the process during which the data is divided into k subsets of training learners using one set of data and testing it using a different set. The data is divided into k subsets.

| model<br>scaler | Bagging | CART | ExtraTrees | KNN | LDA | LR | LinearSVC | NB | RF | SVM | XGB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.476 | 0.432 | 0.504 | 0.442 | **0.532** | 0.525 | 0.531 | 0.481 | 0.522 | 0.375 | 0.518 |
| MaxAbsScaler | 0.483 | 0.432 | **0.514** | 0.445 | 0.532 | **0.531** | 0.532 | 0.481 | 0.522 | **0.53** | 0.518 |
| MinMaxScaler | 0.48 | 0.431 | 0.507 | 0.445 | 0.532 | 0.53 | 0.532 | 0.481 | 0.522 | 0.527 | 0.518 |
| Normalizer | 0.462 | 0.41 | 0.505 | 0.448 | 0.522 | 0.494 | 0.511 | 0.484 | 0.504 | 0.469 | 0.498 |
| PowerTransformer-Yeo-Johnson | 0.484 | 0.431 | 0.511 | 0.442 | 0.522 | 0.528 | 0.522 | **0.495** | 0.522 | 0.514 | **0.522** |
| QuantileTransformer-Normal | 0.466 | **0.439** | 0.503 | 0.432 | 0.512 | 0.51 | 0.511 | 0.478 | **0.523** | 0.485 | 0.519 |
| QuantileTransformer-Uniform | **0.49** | 0.438 | 0.511 | 0.441 | 0.52 | 0.522 | 0.512 | 0.49 | 0.523 | 0.525 | 0.519 |
| RobustScaler | 0.472 | 0.432 | 0.51 | **0.455** | 0.532 | 0.529 | **0.533** | 0.481 | 0.522 | 0.521 | 0.516 |
| StandardScaler | 0.482 | 0.432 | 0.512 | 0.44 | 0.532 | 0.527 | 0.532 | 0.481 | 0.522 | 0.523 | 0.517 |

Next, we considered hyperparameter tuning since the scores from the cross-validation experiment are low. Parameter tuning is the process of selecting the values for a model's parameters that maximize the accuracy of the model. We picked the top three models with the highest score and applied the Grid Search technique to improve the scores. GridSearchCV from sklearn combines both Cross Validation with Parameter Tuning Using Grid Search

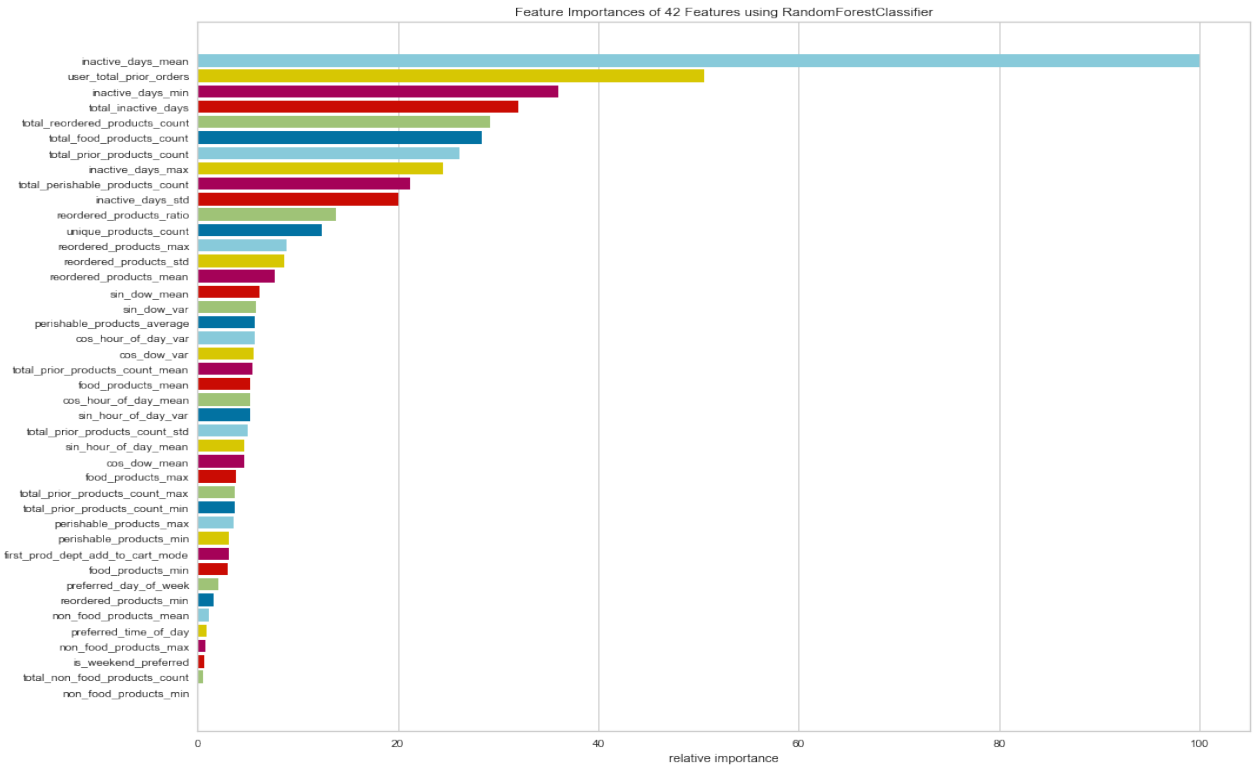
| model | CV_Score_with_hyperparamter_tuning |
|---|---|
| Random Forest | 0.533 |
| Logistic Regression | 0.532 |
| SVM | 0.531 |

We selected Random Forest classifier model as the model for this problem since it scored the highest. The most critical parameters for a Random Forest are max depth (how deep we want to grow the trees) and number of estimators (trees).

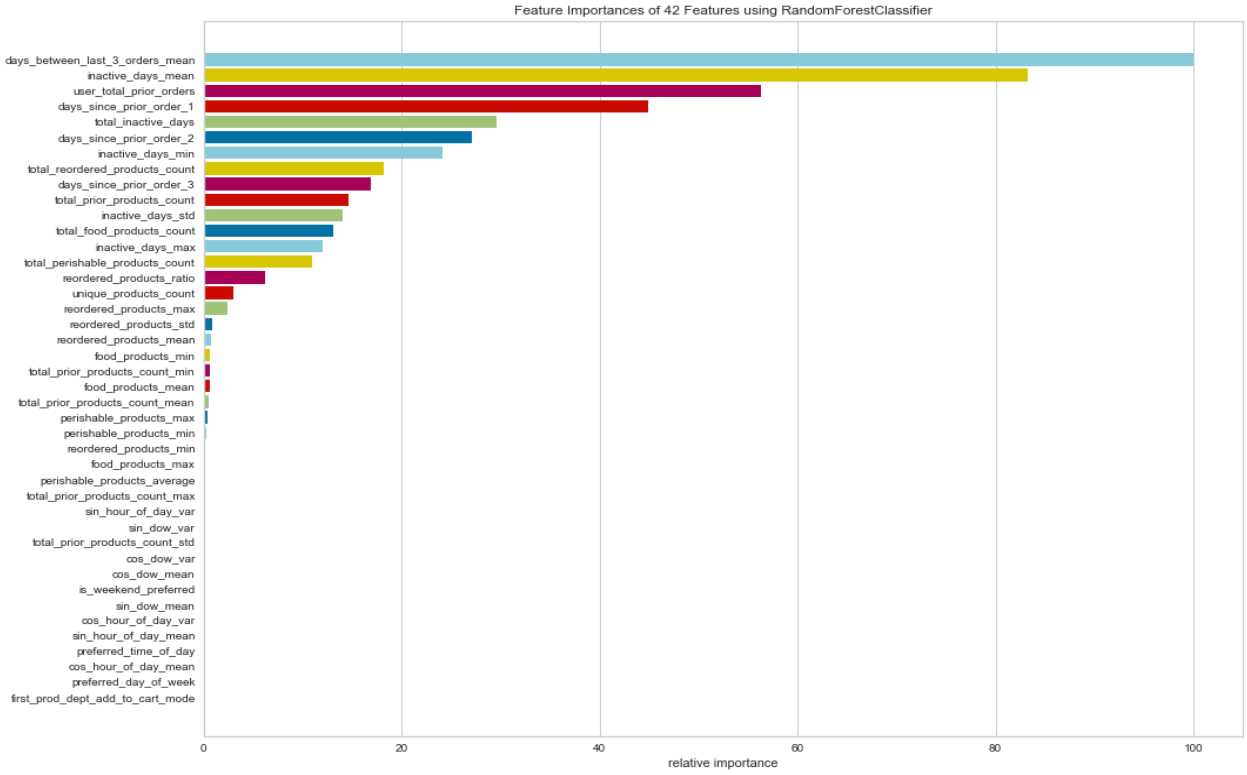
```
Best Estimator learned through GridSearch:

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='auto', 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, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=seed, verbose=0,
            warm_start=False)
```

To get a better understanding of the of the model's logic and identify which features are important, we investigated the feature importance using yellowbrick FeatureImportances visualizer

# Team InstaMarketing Final Paper – Predicting Customer Order Date




Using that above graph, we focused on the important variables and removed the less important variables from our dataset. Also, we added 3 new features that we think slightly improve the performance of the model.

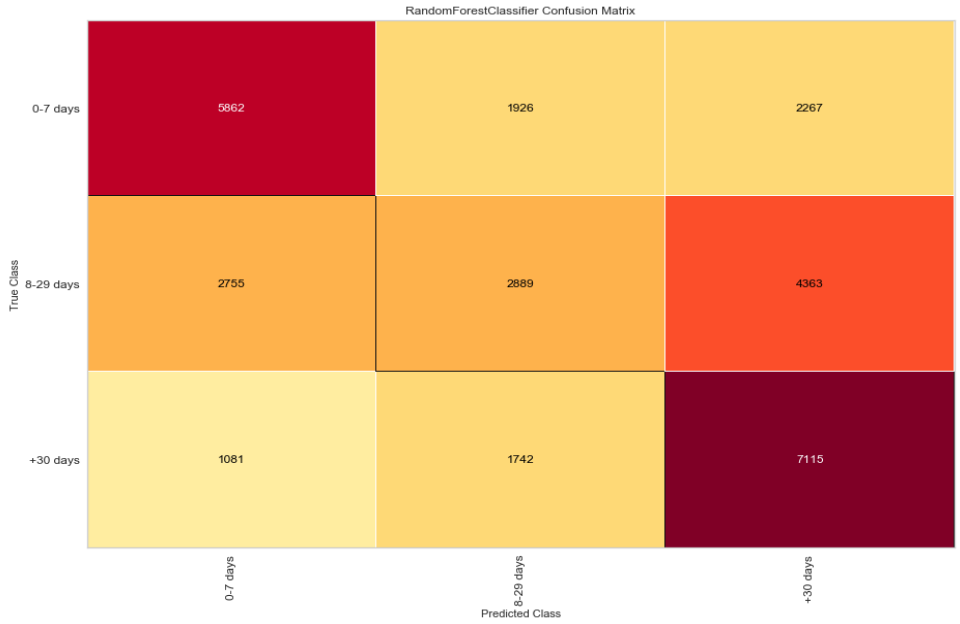



InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

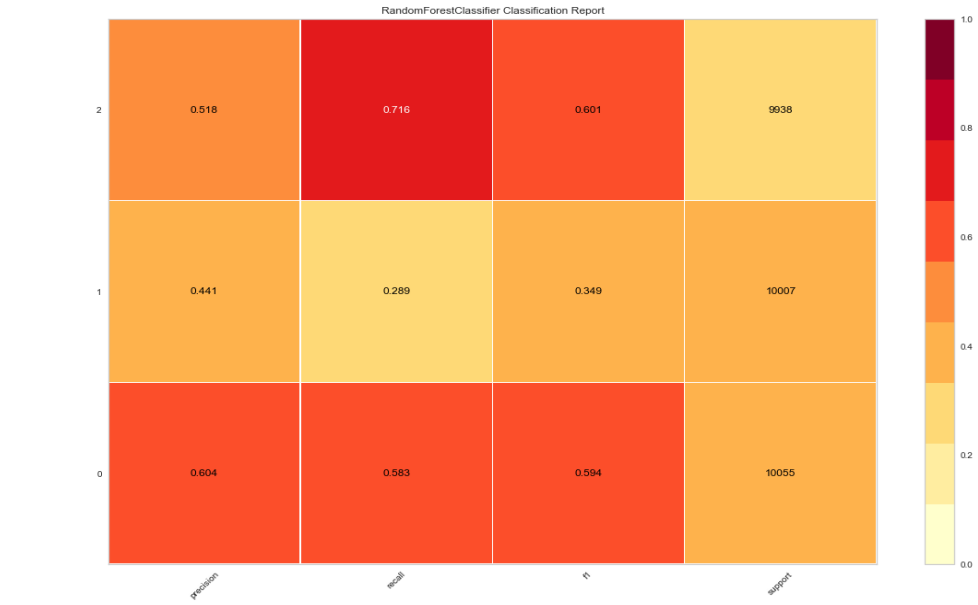
# Team InstaMarketing Final Paper – Predicting Customer Order Date

The new feature importance plot show that our 3 new added features have high relative importance. We proceeded with eliminating the less important features in order to have fewer complex data and run the algorithm faster.

Classification accuracy alone can be misleading since we have more than two classes in the dataset. Since our data has three classes, we didn't know if the accuracy score achieved was due to all classes are being predicted equally well or whether one or two classes are being neglected by the model. Therefore, we plotted the confusion matrix to give us a better idea of what the classification model is getting right and what types of errors it is making. In addition, we plotted the classification report to show a representation of the main classification metrics on a per-class basis. We used ConfusionMatrix and ClassificationReport visualizers from yellowbrick library to generate these plots.




```
Accuracy0.5288666666666667
```




InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

From the plots above, we can see that the recall score of class 2 (+30 days) is significantly higher (~0.716) than the overall accuracy score of the model. This is important because from the business need, these are the users to which we like to reach out more often. Also, class 1 (8-29 days) scored very low in comparison to the other two classes. In the confusion matrix, we can see that most of the class 1 was mistaken for class 2 ~43% of the time whereas it was mistaken for class 0 ~27% of the times and only correctly predicted ~28% of the times.

This observation led us to explore another approach in terms on how to group our users and solving this problem as a binary classification problem instead of a multiclass. We redefined the target (next purchase day) for users and regrouped them by less than or equal to 14 days as 0 and greater than 14 days as 1. The benefit of this new grouping is to have more balanced data set. For our model selection and evaluation, we applied the same steps we discussed above in the multiclass problem.

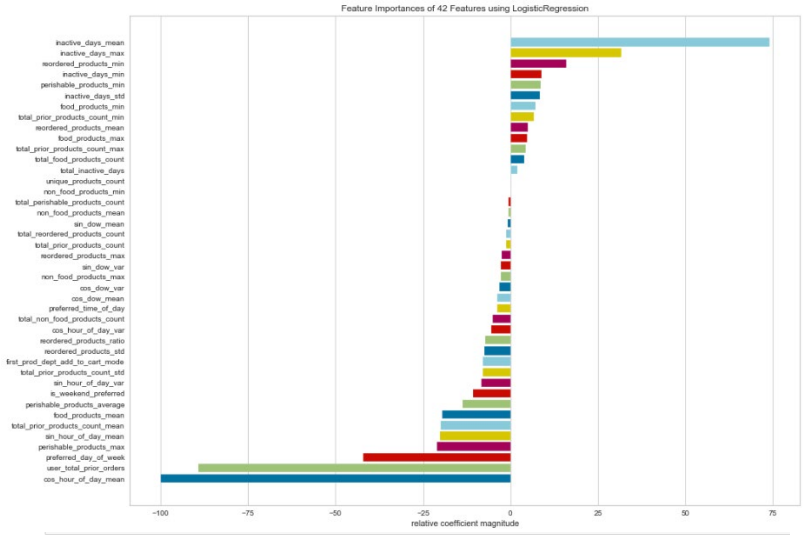
Below is the different scores achieved from the Cross Validation process:

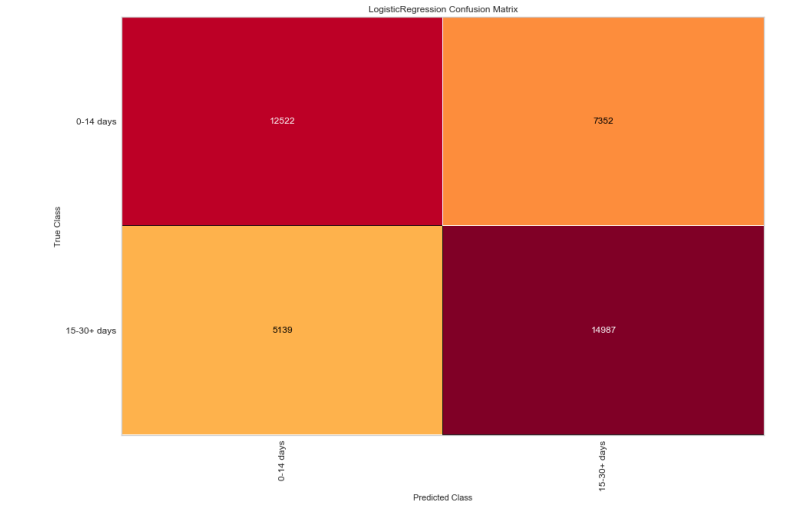
| model<br>scaler | Bagging | CART | ExtraTrees | KNN | LDA | LR | LinearSVC | NB | RF | SVM | XGB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.636 | 0.592 | 0.656 | 0.613 | 0.671 | 0.669 | 0.671 | 0.648 | **0.676** | 0.499 | 0.647 |
| **MaxAbsScaler** | 0.624 | 0.591 | 0.659 | 0.596 | 0.671 | 0.678 | **0.676** | 0.648 | 0.676 | **0.679** | 0.647 |
| **MinMaxScaler** | 0.632 | 0.592 | 0.658 | 0.612 | 0.671 | 0.677 | 0.676 | 0.648 | 0.676 | 0.676 | 0.645 |
| **Normalizer** | 0.632 | 0.586 | 0.663 | 0.619 | 0.674 | 0.651 | 0.673 | 0.628 | 0.673 | 0.641 | **0.664** |
| **PowerTransformer-Yeo-Johnson** | 0.626 | 0.592 | 0.666 | 0.62 | **0.678** | 0.676 | 0.675 | **0.657** | 0.676 | 0.67 | 0.638 |
| **QuantileTransformer-Normal** | 0.636 | 0.594 | 0.664 | 0.628 | 0.674 | 0.668 | 0.674 | 0.651 | 0.676 | 0.667 | 0.649 |
| **QuantileTransformer-Uniform** | 0.627 | 0.594 | 0.666 | **0.634** | 0.671 | **0.682** | 0.672 | 0.646 | 0.676 | 0.675 | 0.649 |
| **RobustScaler** | **0.637** | 0.592 | **0.667** | 0.632 | 0.671 | 0.68 | 0.674 | 0.648 | 0.676 | 0.673 | 0.646 |
| **StandardScaler** | 0.637 | **0.595** | 0.652 | 0.616 | 0.671 | 0.681 | 0.674 | 0.648 | 0.676 | 0.664 | 0.646 |

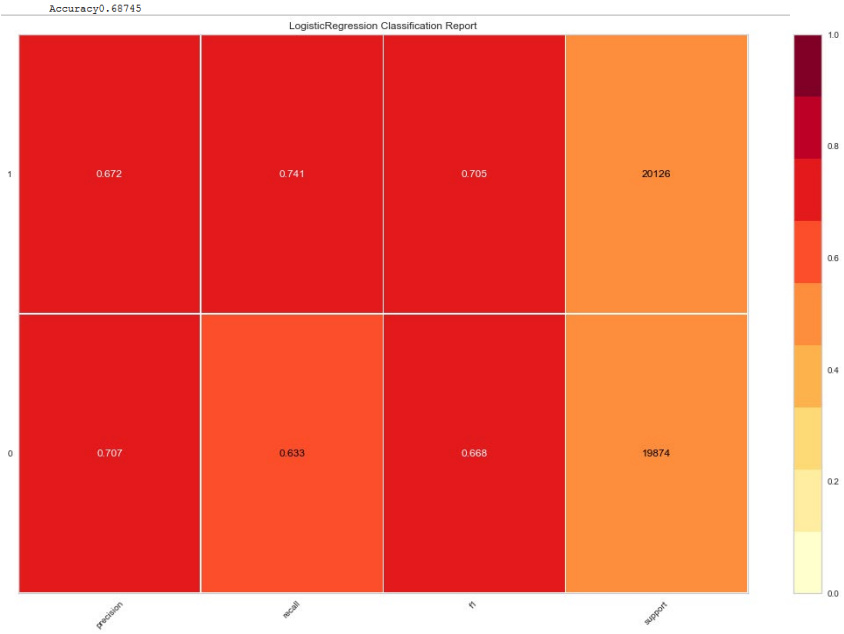
This shows that we scored a higher score across the different classifiers and scalers. Similarly, we applied hyperparameter tuning and selected Logistic Regression model since it scored higher (0.682).

In the same way we did in the multiclass problem, we evaluated the logistic regression model using feature importance, confusion matrix, and classification report. As expected, we ended up with a higher precision, recall and f1 score in comparison to the multiclass problem.

# Team InstaMarketing Final Paper – Predicting Customer Order Date




InstaMarketing - Summer 2019 Cohort #15
Charles Ping, Joe DeRose, Mohamed Osman, Raymond Stanley, Richard Colvin

## Section VII: Conclusion and Best Practices

Multiclass Random Forest model with hyper-tuned score of 0.53 may not seem high enough to deploy in real world setting; however, having a high recall score for the +30 days group is important for our business case. In addition, grouping users into two groups instead got us a higher score when comparing it to the multiclass problem.
Deciding the grouping of our users is a question for both statistics and business needs. It should make sense in terms of the distribution of the data and in the same time should be easy to act in terms of business needs.

Finding ways to increase the model score would be priority to improve score. The biggest flaw was most likely lack of strong features. We considered scrapping the web for product prices in order to calculate the total order price for each user, but we couldn't due to the lack of time.
For future work, we are looking into obtaining more predictive features. Brining in data from other sources outside of the Instacart csv set could enhance outcome. Features such as user demographics, spending habits, and from where grocery store are ordered.

Finally, we can expand on this problem to try to predict other data products that can help company's growth such as sales prediction, and Market Response Models.

## Section VIII: Data Product

The data product for our team consisted of a tailored web application exhibiting the production output from our machine learning estimator. Our product simplifies complex analysis in order to provide the Instacart marketing department (aka InstaMarketing) with predictions on consumer purchasing behavior. When designing the application, we took into consideration the needs of the department and the potential users of the app.

Definition of customer requirements is a crucial factor when creating a viable data product. In this scenario, our customer is requesting a report generator that would allow them to view, edit, and search user IDs that will likely be making their next purchase in a specified timeframe. In order to accomplish this task two things needed to be defined: 1) How does our team translate the results of the machine learning (ML) model into a useable product, and 2) What platform do we use to deliver the product to our customer.

Translation of the ML predictions to the preferred product format required over 6 python libraries (Flask/Pandas/Scikit-learn/Numpy/Sqlite3/Json), 3 javascript libraries (AngularJS/JQuery/D3), HTML and CSS.

The first step in accomplishing this task was creating a new column in a pandas dataframe that held the prediction group number assigned to the user by the ML model. Once this column was created, we created a script that took the dataframe and ingested it into a new table in a SQLite database table.

Next, we needed to connect to the database and convert the table data into a dictionary in order to then convert the data into json format in order to send to our api. In Flask, we used decorators to define the api route in the URI. We then created additional routes to render our html pages for the user interface to the app.

We used AngularJS to create a controller with a function that specified the API route and https request type. We then used interpolateprovider to allow angular to function without interference from the Jinga2 templating system that comes standard with Flask. We then used ng-repeat to loop through the results from the api and print the selected data to the html page. The semantic css library was used to define design elements of the site.