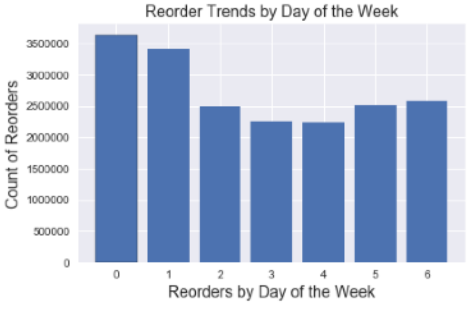
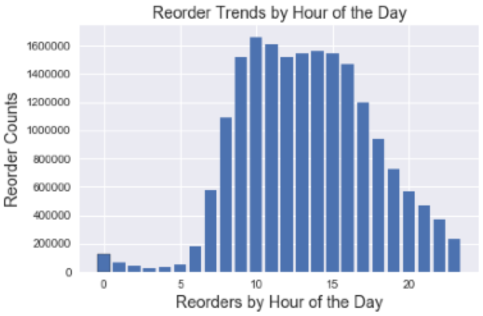
Customer Habits at the Grocery Store

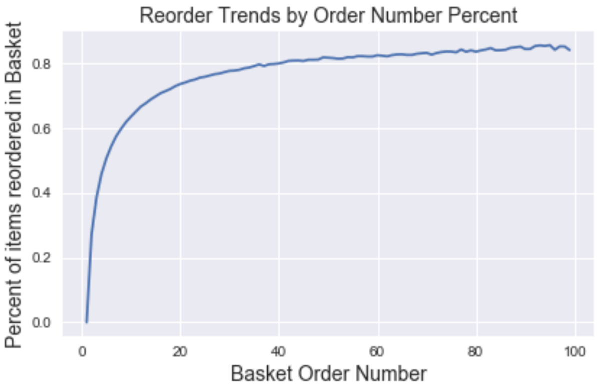
We are interested in how to predict reorders for a grocery store. The data is provided from InstaCart, however, the client can be any business looking to predict customer reorders. We will take the data wrangle and clean it for analysis and modeling. Once cleaned, we will explore the data in order to get a picture of how reorders behave and look for any informative trends. Once explored, we will use machine learning specifically a Support Vector Machine and a Random Forest model to predict future reorders. We will compare and analyze the results from these models. It is our goal that the data will provide insight into customers purchasing habits and future tendencies.

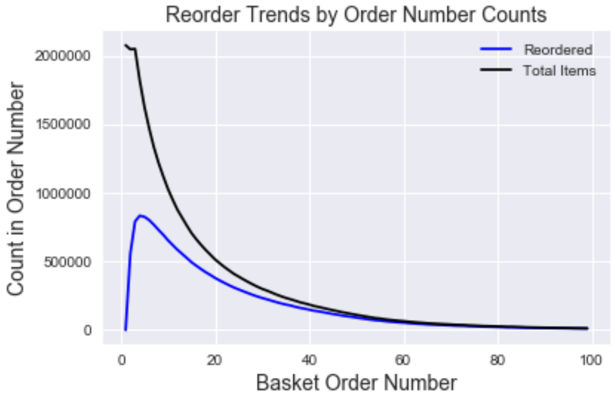
The data comes in the form of six comma separated variable (csv) files. We use Python and Pandas to wrangle these files into data frames. We see that only "Days since Prior Order" has missing values, which we replace with zero. This does cause some information loss. We no longer have access to same day reorders from customers, thus, we will only concern ourselves with reorders at least one day apart. The csv files are all related by the "order_id" reference number, which we will use to combine our data frames into a tidy format. Tidy format is one that has observations as rows and columns as features. The data is now in a condition to explore.





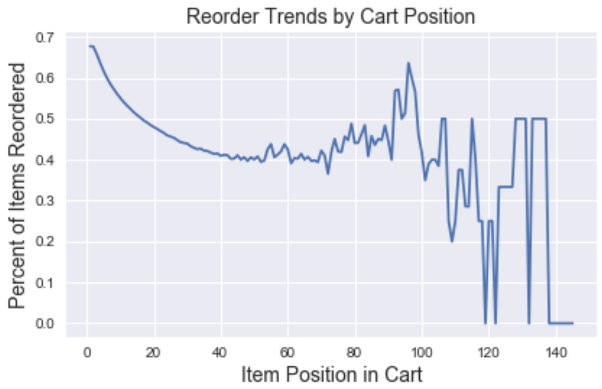
We will focus our exploration on what the data tells about customer reorder habits. We see that overall 60% of purchases contain reorders. The first feature "order_dow", which is the day of the week customers make purchases, tells us customers make more purchases and more reorders on "day 0" and "day 1" of the week. The next feature "order_hour_of_day" shows customers typically place orders reorders between 8 am and 5 pm. The feature “order_number” tracks the orders chronologically by customer. We see that reorders as a percent of the basket does increase as customer stay with the company. The first order, as expected, has no reorders, hence a percent of 0%. Afterward, the percent of reorders in the basket climbs exponentially for each customer leveling off at near 85% (left graph). If we look at the raw values of reorders compared to the basket items instead of percent we see the reorders climb shortly and then start to decline at the same rate as items in the basket decreases (right graph).

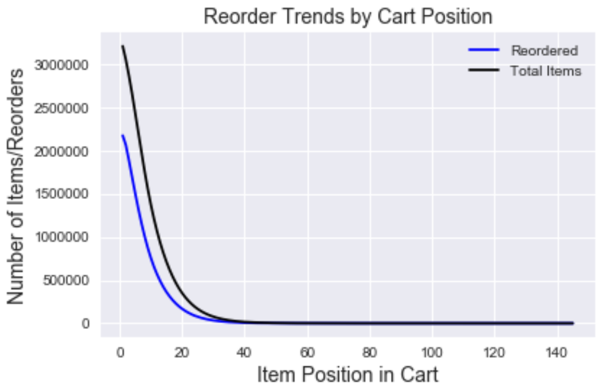




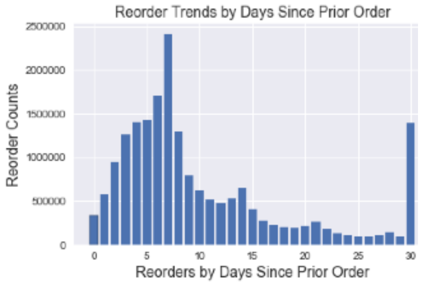
Continuing to the next feature “add to cart order," we can answer an interesting question. Do customers buy reorders first or wait until the end? We see that most customer’s first item added to the cart is a reorder at just under 70%. Followed closely behind by the second item near the same rate. This begins to trail off to around 40% up to the 55th position in the cart. Afterward, things get a bit more erratic do most likely do to the low number of orders with basket items this large.

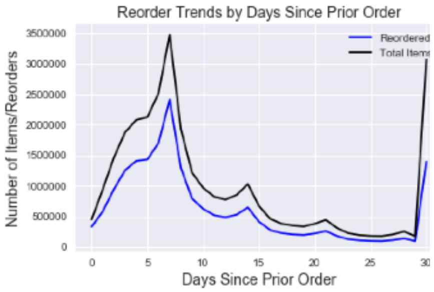
Customer Habits at the Grocery Store

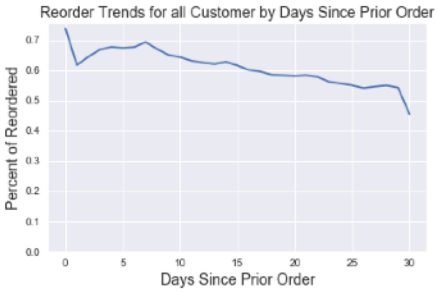




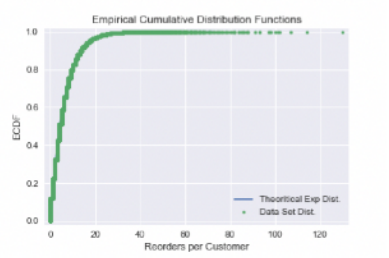
The next feature “days since prior order” gives us a picture that most reorders occurs within the first week of the last order. In fact, reorders increase as the week goes on peaking seven days from the previous order after which reorders drop off substantially. The fact that anything over 30 is lumped into the 30th day accounts for the spike at the end of the graph. We see the percent of reorders start at close to 80% and drop to just above 40%. We can make some propositions from this that customers are more likely to reorder items if they make another order within a week of their previous order. Logically, this might be because customers remember more easily things they enjoyed from their last purchase and purchase those items again.

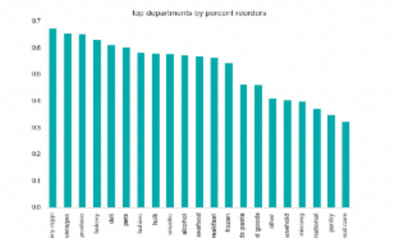






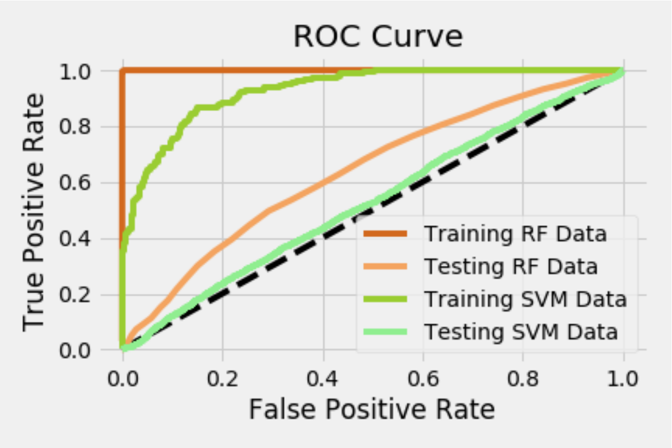
Pulling in some basic statistics by counting up and plotting purchases and reorders, we see a classic exponential decay suggesting an exponential distribution might be a good approximation of the reorder distributions. Plotting these overlaid





on one another we see that an exponential distribution does fit the data well.Using the properties of the exponential distribution, some probability questions can be easily answered. We see that 99% of customers reorders at least 28 items, 95% reordered 18 items and 66% reordered at least 6 items. We can also look at what department is the top reorders by percent or volume. By taking into account customer's historical reordering habits by product, we train our machine learning model to make predictions off of these habits.

The approach in making these predictions was to use two different machine learning algorithms in SciKit Learn. One model using a Support Vector Machine (SVM) with a radial basis function and the other using a Random Forest Classifier. In order to save computational time and money we reduced the data set to a smaller sample of 550 customers. We tuned both models for their respective parameters. We achieved decent "Area Under the Curve" (AUC) results of .527 with the SVM and .637 with the Random Forest, using just the basic features innate in the data set. We compare these two on a Receiver Operating Characteristic Curve (ROC), a Precision-Recall Curve and by Area Under the Curve (AUC). The best model will depend on the business application we choose to use it for. We will have to take cost into account and then choose one of these models.



Tentatively, the best one looks to be Random Forest.

Many companies already use a recommender on their online platform to recommend products. Using this method they could better target recommendations that their customers might want. We saw earlier how reorders fell after a week from a customer last order. If the recommended items were tailored to customer preferences, it would service as a catalyst to remind customers of items they enjoyed and possibly improve sells for the company.

We could improve this approach by including some feature engineering and fine tuning some more model parameters. We would also like to expand the size of the training data by improving the cost of computation.