**Insta Market Basket Analysis**

**Big Data and Intelligence Analytics**

**INFO 7245 - SPRING 2018**

PROFESSOR:

**Nicholas Brown**

TEAM MEMBERS:

**Sameer Suman (**[www.linkedin.com/in/sameersuman/](http://www.linkedin.com/in/sameersuman/)**)**

Table of Contents

[Problem Statement 3](#_Toc481448732)

[Part 1: Data wrangling and exploratory data analysis 5](#_Toc481448733)

[Data downloading and storing 5](#_Toc481448734)

[Data cleaning 7](#_Toc481448735)

[Exploratory data analysis in Power BI 17](#_Toc481448736)

[Storing clean data on Amazon S3 bucket 21](#_Toc481448737)

[Part 2: Building and Evaluating Models 25](#_Toc481448738)

[Classification 25](#_Toc481448739)

[Clustering 28](#_Toc481448740)

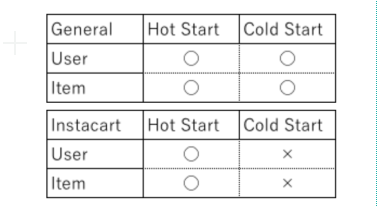
# **Problem Statement**

Recently [Instacart Market Basket Analysis competition](https://www.kaggle.com/c/instacart-market-basket-analysis) challenged Kagglers to predict which grocery products an Instacart consumer will purchase again and when. Imagine, for example, having milk ready to be added to your cart right when you run out, or knowing that it's time to stock up again on your favorite ice cream.

This focus on understanding temporal behavior patterns makes the problem fairly different from standard item recommendation, where user needs and preferences are often assumed to be relatively constant across short windows of time. Whereas Netflix might be fine assuming you want to watch another movie similar to the one you just watched, it's less clear that you'll want to reorder a fresh batch of almond butter or toilet paper if you bought them yesterday.

 The goal of this competition was to predict grocery reorders: given a user’s purchase history (a set of orders, and the products purchased within each order), which of their previously purchased products will they repurchase in their next order?

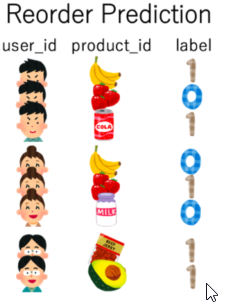
The problem is a little different from the general recommendation problem, where we often face a cold start issue of making predictions for new users and new items that we’ve never seen before. For example, a movie site may need to recommend new movies and make recommendations for new users.



The sequential and time-based nature of the problem also makes it interesting: how do we take the time since a user last purchased an item into account? Do users have specific purchase patterns, and do they buy different kinds of items at different times of the day? And the competition’s F1 evaluation metric makes sure our models have both high precision and high recall.

## Main Approach

I used XGBoost to predict: **Predicting reorders** - which previously purchased products will be in the next order? This model depends on both the user and product.



.

## **Data**

The dataset for this competition is a relational set of files describing customers' orders over time. The goal of the competition is to predict which products will be in a user's next order. The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. We also provide the week and hour of day the order was placed, and a relative measure of time between orders. For more information, see the [blog post](https://tech.instacart.com/3-million-instacart-orders-open-sourced-d40d29ead6f2) accompanying its public release.

Each entity (customer, product, order, aisle, etc.) has an associated unique id. Most of the files and variable names should be self-explanatory.

### **aisles.csv**

aisle\_id,aisle

1,prepared soups salads

2,specialty cheeses

3,energy granola bars

...

### **departments.csv**

department\_id,department

1,frozen

2,other

3,bakery

...

### **order\_products\_\_\*.csv**

These files specify which products were purchased in each order. order\_products\_\_prior.csv contains previous order contents for all customers. 'reordered' indicates that the customer has a previous order that contains the product. Note that some orders will have no reordered items. You may predict an explicit 'None' value for orders with no reordered items. See the evaluation page for full details.

order\_id,product\_id,add\_to\_cart\_order,reordered

1,49302,1,1

1,11109,2,1

1,10246,3,0

...

### **orders.csv**

This file tells to which set (prior, train, test) an order belongs. You are predicting reordered items only for the test set orders. 'order\_dow' is the day of week.

order\_id,user\_id,eval\_set,order\_number,order\_dow,order\_hour\_of\_day,days\_since\_prior\_order

2539329,1,prior,1,2,08,

2398795,1,prior,2,3,07,15.0

473747,1,prior,3,3,12,21.0

...

### **products.csv**

product\_id,product\_name,aisle\_id,department\_id

1,Chocolate Sandwich Cookies,61,19

2,All-Seasons Salt,104,13

3,Robust Golden Unsweetened Oolong Tea,94,7

|  |  |
| --- | --- |
| **Variable** | **Description** |
| **Item\_Identifier** | Unique product ID |
| **Item\_Weight** | Weight of product |
| **Item\_Fat\_Content** | Whether the product is low fat or not |
| **Item\_Visibility** | The % of total display area of all products in a store allocated to the particular product |
| **Item\_Type** | The category to which the product belongs |
| **Item\_MRP** | Maximum Retail Price (list price) of the product |
| **Outlet\_Identifier** | Unique store ID |
| **Outlet\_Establishment\_Year** | The year in which store was established |
| **Outlet\_Size** | The size of the store in terms of ground area covered |
| **Outlet\_Location\_Type** | The type of city in which the store is located |
| **Outlet\_Type** | Whether the outlet is just a grocery store or some sort of supermarket |
| **Item\_Outlet\_Sales** | Sales of the product in the particulat store. This is the outcome variable to be predicted. |

# **Part 1: Data wrangling and exploratory data analysis**

In this section, we will perform the data the following operations:

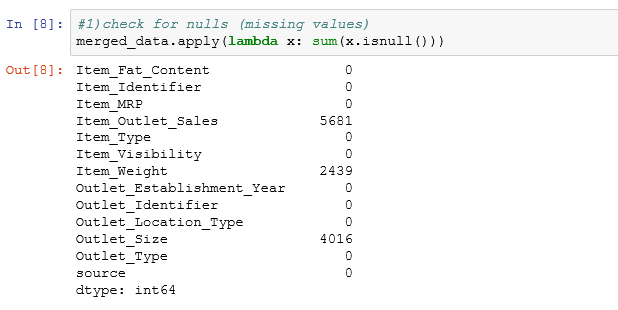
* Data cleaning
* Exploratory data analysis in Python
* Exploratory data analysis in Tableau

As per the problem statement, we have two files, train csv to train our model and test csv to test our model and predict the sales outcome. But, before we begin to clean our data or impute new features and values, we combined both the files onto one so that we don’t have to clean the data twice differently.

## **Data cleaning**

This section comprises of handling missing data. We get the concatenated .CSV file from the previous part, on which we perform data cleaning steps.

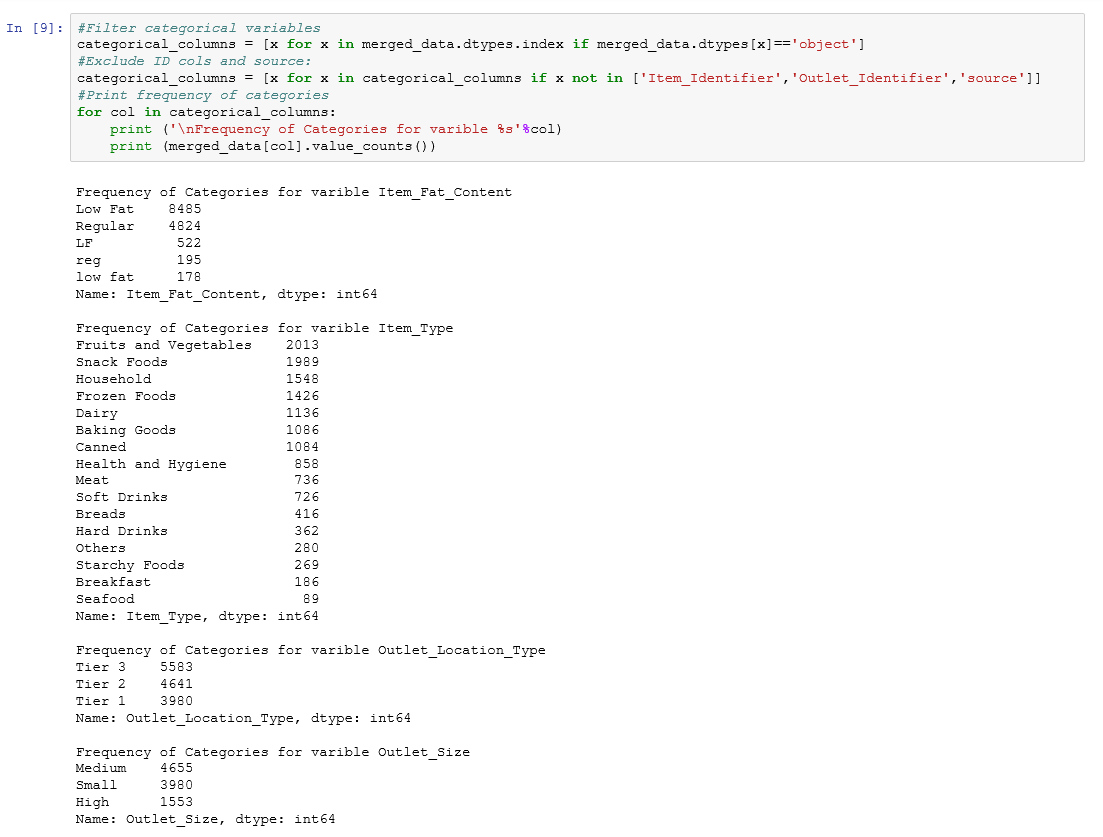
I first analyzed all the columns having empty values as shown below:



I have implemented this by writing a lambda function, which gives us a .CSV file having the count of null values for each variable.

I will clean these null values in the upcoming section

Prior to cleaning, just to understand how the data is flowing, so I got a count of all the different categories in all the variables.



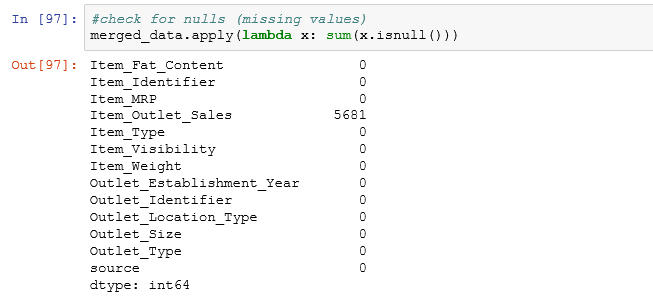
**Columns Updated and cleaned:**

* **Item\_Weight** – Ideally, a weight of an item can never be zero. So, replacing all the nulls with the average weight of a particular item across all outlets.



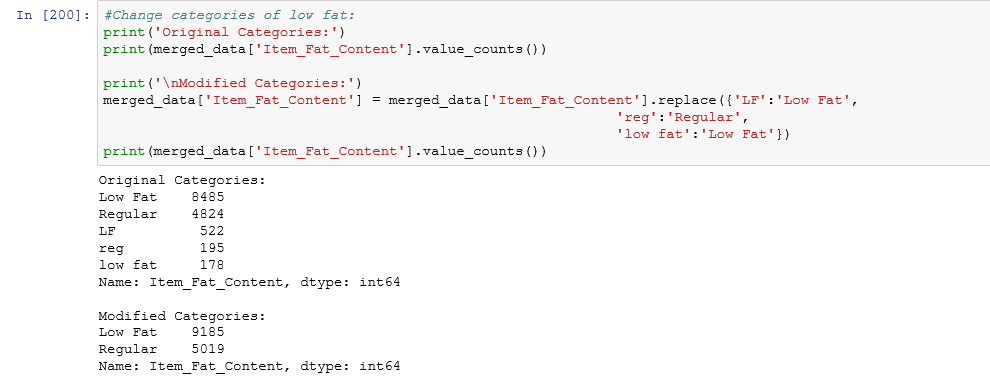
* **Outlet\_Size** –
* **Item\_Outlet\_Sales** – We need not clean these, as these are nulls from the test file, as we need to predict these values based on our model

We check for nulls again after cleaning and below are the results:

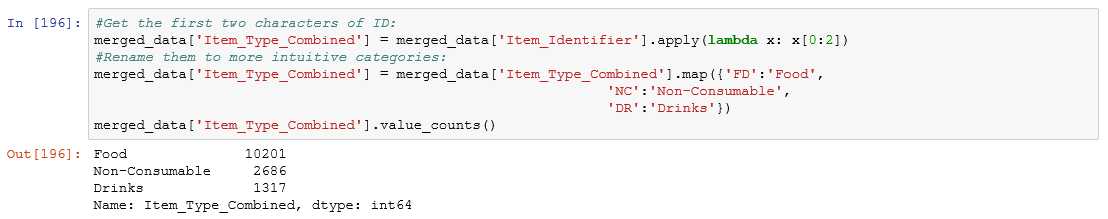


**Data pre-processing**

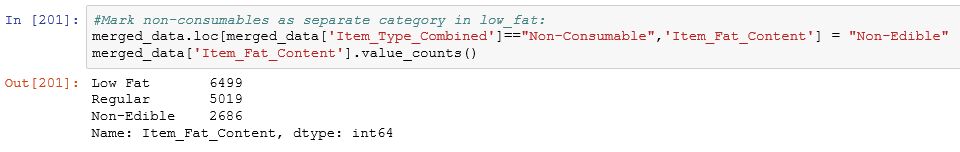
* On observing, I found out that there are 5 different categories present, out of which 3 are similar and all combined falls under two categories in the data. So treated this bad data and combined the groups together as below:



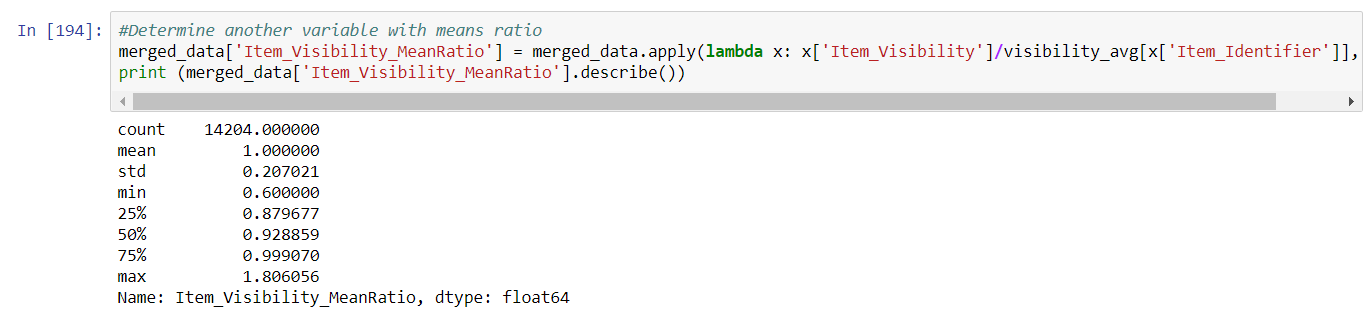
* The data dictionary says there are three types of items – Food items, Non-consumable and Drinks.



* The data says one category as Non consumable, but still filters it using Regular and Nonfat. Since, some are non-consumable, it doesn’t fall under any fat/nonfat category, so replacing all non-consumable items as “Non-edible”



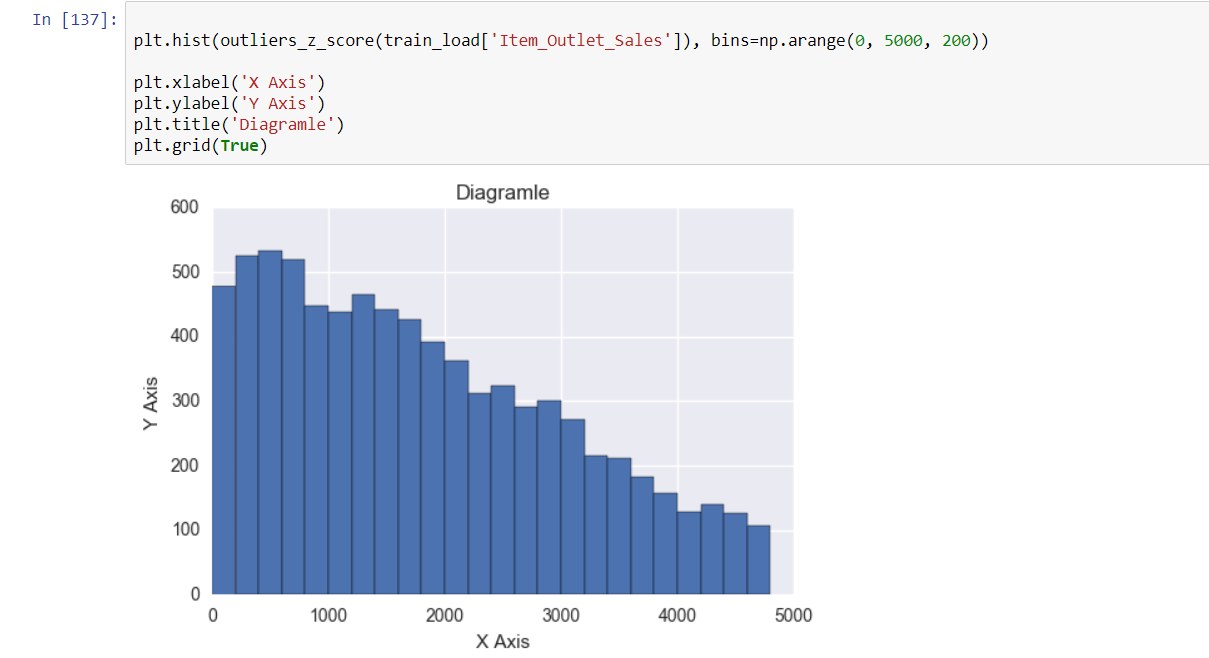
* Determine visibility ratio. (Standardize the visibility ratio)



**Exploratory data analysis in Python**

**Analysis 1:** The distribution of sales data

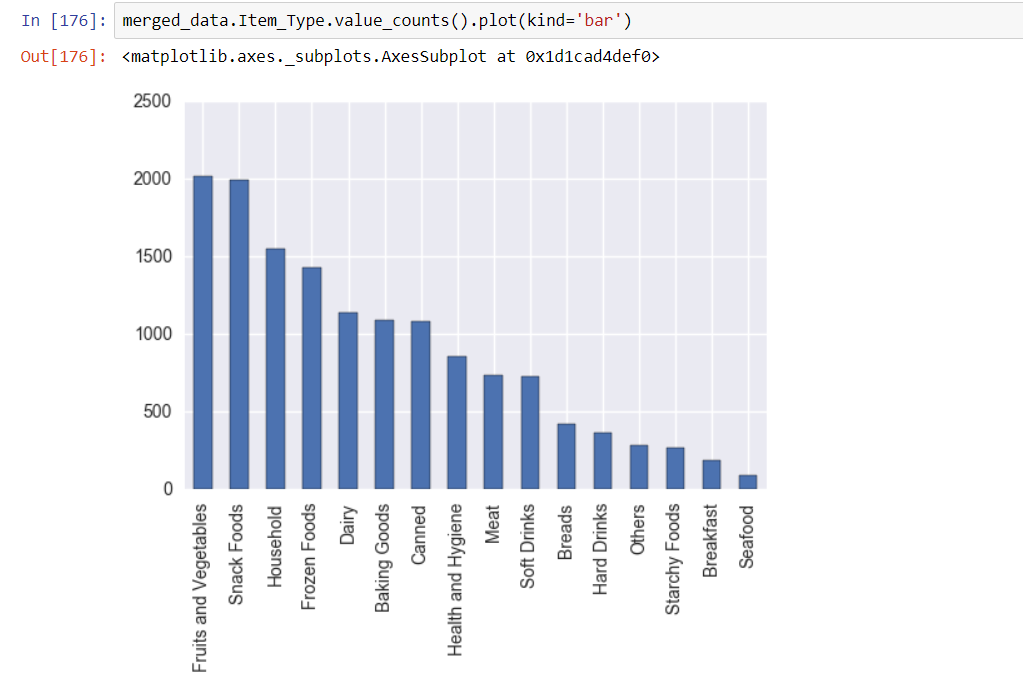
**Approach:**



**Conclusion:** The data is left skewed based on the total sum of sales

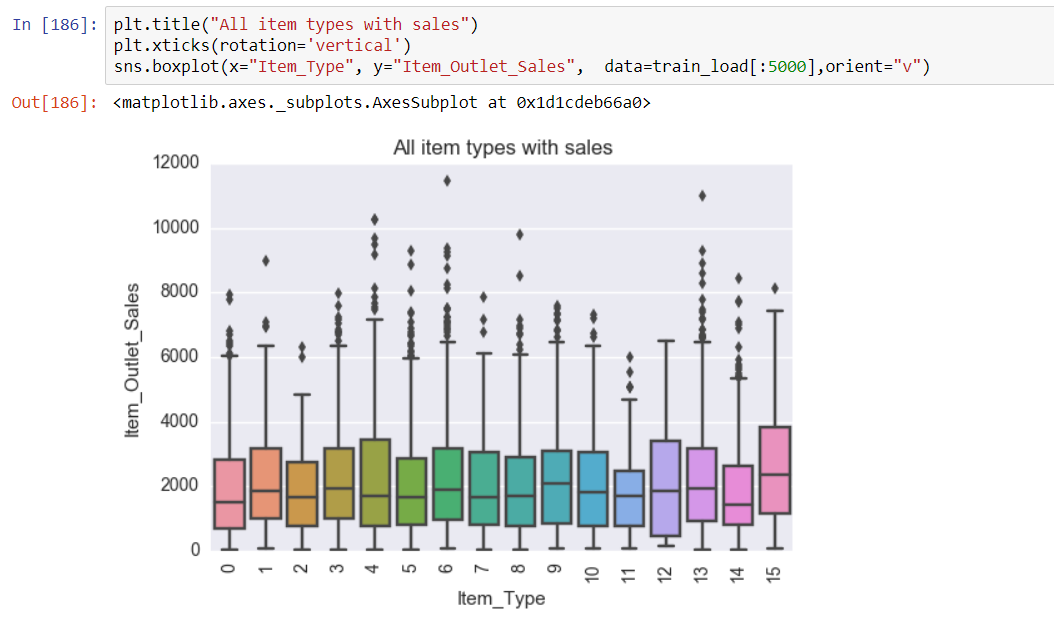
**Analysis 2:** Sum of sales based on different categories of Item\_Type

**Approach:**



**Analysis 3:** Box plot view for all sales items

**Approach:**



**Part 2: Building and Evaluating Models**

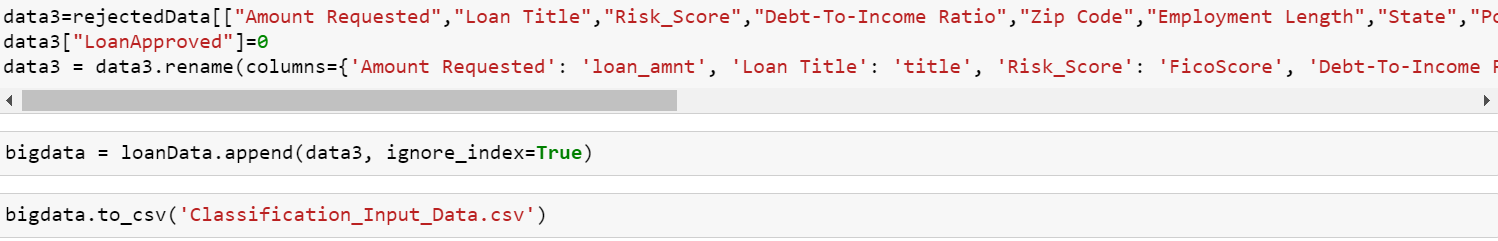
**Classification**

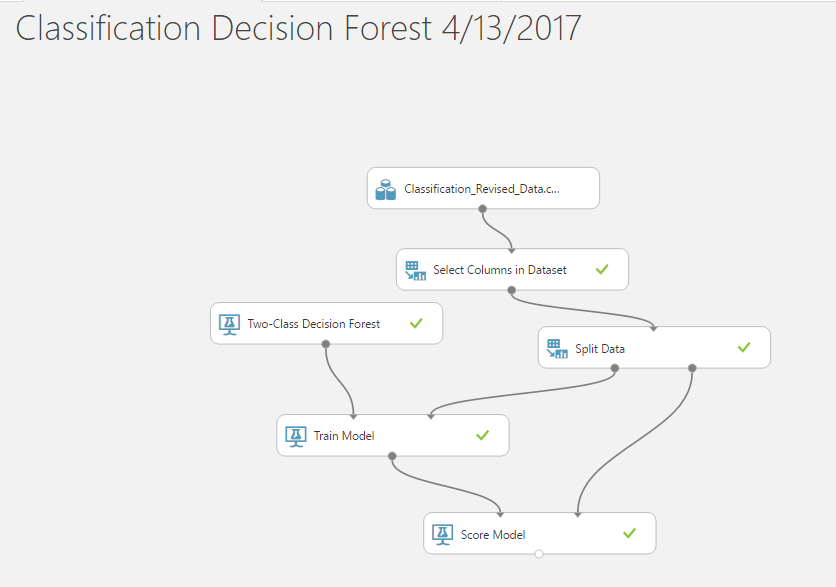
Our first step in classification is to obtain a dataset which has accepted and rejected observations.

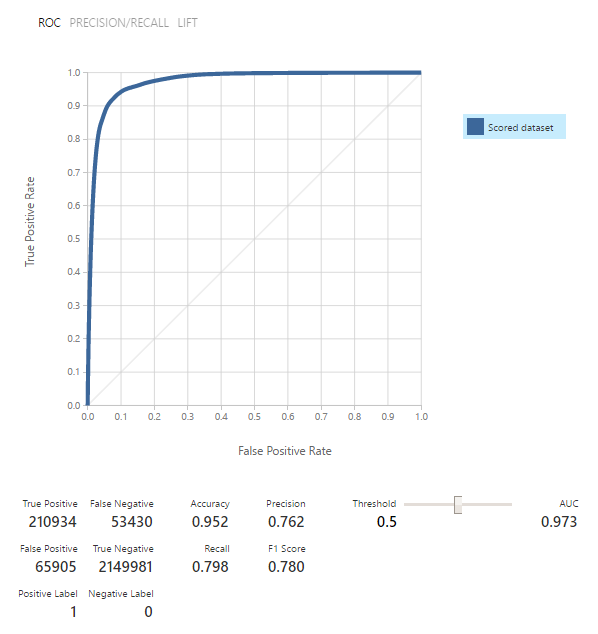
To get this combined data we are concatenating loan accepted data with the rejected data based on common columns in both the files. The selected columns are as follows:

|  |  |
| --- | --- |
| **Accepted Loan Data** | **Rejected Loan Data** |
| Loan\_amnt | Amount Requested |
| title | Loan Title |
| FicoScore (derived attribute) | Risk\_Score |
| dti | Debt-To-Income Ratio |
| zip\_code | Zip Code |
| emp\_length | Employment Length |
| addr\_state | State |
| policy\_code | Policy Code |
| LoanApproved | LoanApproved (derived attribute) |

Below is the code snippet for the same.







Based on the combined file of rejected and accepted records, we trained our model based on few variables, whether to give loan to a person or not. We took the common columns between both the rejected and accepted files, combined it together and moved the data with policy code 2 into the rejected file, as they will not be getting the loan henceforth.

Among the 8 columns, we figured out few columns which are most relevant and gives us higher accuracy of output based on backward selection. We then trained those 8 columns and predicted whether a person should be given loan or not.

We implemented various classification algorithms like Logistic Regression, Random Forest, KNN, Neural Network using iPython Notebook.

Among these, we figured out the best model based on the Accuracy rate we obtained.

Random Forest – 95.2%

SVN – 92.8 %

Logistic Regression – 92.8%

Neural Network – 94.9%

As per the accuracy rates, Random Forest has the highest accuracy and hence that is our best model for Classification. We then implemented Random Forest on Microsoft Azure Machine Learning and the output is as shown in the above image.