Super Market Order Analytics based on Shopping Trends

INFO 6105 Data Science Engineering Methods and Tools

Spring 2020

Project Report

Team Members

1. Priyanka Bandekar (NUID: 001055485)
2. Prasham Shah (NUID: 001057833)
3. Prasanth Thota (NUID: 001844237)

Contents

1. Introduction 2

2. Background Research 2

3. Project Objectives 3

4. Methodology 4

5. Analysis and Results 5

6. Conclusion 14

7. References 15

8. Acknowledgement 16

1. Introduction

Analysis of a customer purchase trend has often been an important sector for various supermarkets. It is an integral analysis from a vendor's perspective as it governs the action behind tough decisions that contribute to the overall profitability for a business. Predicting the shopping capabilities for a customer, for many years, has been a prime area of application in the field of data mining.

The performance and accuracy of the predictions usually revolves around the historical data, decisions or business actions taken at some point and the future trends in the business. This was the driving factor of our project wherein we have made use of historical data and predicted the outcomes based on that data.

We have incorporated different methods to perform the predictive analysis and also tried to improve the accuracies of these prediction methods. Various python libraries like pandas, numpy and sklearn have been used in this project.

2. Background

Our project titled 'Super Marker Order Analytics based on Shopping Trends' focuses on extracting crucial trends from the input data that can be used to predict what the customer is most likely to purchase in his next order based on the purchases made by him in the previous orders. For the purpose of this project, we have made use of 'Instacart Market Basket Analysis' dataset that is freely available on Kaggle.

Instacart is a grocery ordering and delivery app that aims to make it easy to fill the refrigerator and pantry with one's personal favorites and staples when you need them. After selecting products through the Instacart app, personal shoppers review their order and do the in-store shopping and delivery for one.

The 'Instacart Market Basket Analysis' was a Kaggle competition where the users were instructed to predict the next purchase item by a customer based on his previous purchase trends.

Besides predicting the future purchase trends for a customer, we have enabled the use of 'Word2Vec' to cluster the items based on some features. In order to explore the area of data analytics we have used various ML techniques and Tensor Flow capabilities to find out clear solutions about the behavior of a customer.

The data available on Kaggle provides a detailed description of the customer's purchases over a period of time. The dataset used in this project is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we have been provided between 4 and 100 of their orders, with the sequence of products purchased in each order. We have also been provided with the week and hour of day the order was placed, and a relative measure of time between orders.

The dataset consists of 6 csv files that have been described below:

1. aisles.csv: This file contains two columns – aisle_id and aisle. It contains data regarding the name of the aisle along with a unique id associated with each aisle

2. departments.csv: This file contains two columns – department_id and department. It contains data concerning the name of the department and an associated unique id for each department of a product

3. order_products_prior.csv: This file contains four columns – order_id, product_id, add_to_cart_order and reordered. 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

4. order_products_train.csv: It has a schema that is similar to the order_products_prior. This file includes the dataset that will be used to train the test dataset that is explained below.

5. orders.csv: This file contains seven columns – order_id, user_id, eval_set, order_number, order_dow, order_hour_of_day and days_since_prior_order. This file tells us to which set (prior, train, test) an order belongs. We are predicting reordered items only for the test set orders and 'order_dow' is the day of week.

6. products.csv: This file contains four columns – product_id, product_name, aisle_id and department_id. It includes data regarding the products sold by the business.

3. Objectives

The dataset provided on Kaggle is around 500 MB that is quite big. We aim to identify hidden patterns in the data, explore any useful relationships between the data, find out any inconsistencies in the data, cleaning the data and using the consistent data to provide intuitive information from the analysis. The main goal of this project is to predict whether a customer will reorder a particular product based on his previous orders.

We have made use of following stages in our project:

1) Exploratory Data Analysis

This segment of the project can help us to identify the useful aspects of data such as a frequently purchased product by a customer or the busiest day of operation for a store. This information is substantial and can help the store to make definitive and profitable business plans.

2) Logistic Regression

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. It also measures the relationship between the categorical response variable and one or more predictor variables by estimating probabilities.

3) Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. It acts as a meta estimator that fits several decision tree classifiers on various examples of the data set and uses averaging to improve the predictive accuracy and control over-fitting.

4) Gradient Boost Method

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

5) Apriori Algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database.

6) K Nearest Neighbours

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. The algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

4. Methodology

During a search for project topic ideas, we came across a dataset on Kaggle that was based on predicting shopping trends for Instacart, that is an online grocery store. We read through the description of the dataset that was provided and found out that we could do useful exploratory data analysis with it. We thus decided to go forward with this topic.

While exploring the data, we came across a lot of hidden but useful information that could contribute for efficient business movements and thereby lead to profitability. We were able to conclude intuitive outcomes like the busiest day of the week for the store, most purchased products by a customer and so on.

We also analysed some important features from each table and combined them together to perform some analysis and figure out more information about the data. We further tried to develop more problem statements and find out the answers by analysis of the data. We performed feature engineering to implement Word2Vec from the gensim model. This model helped us in clustering products and this is of use help in the recommendation systems.

Apart from exploratory data analytics we tried to apply Random Forests which did not prove to be very accurate because the decision trees did not provide any meaning output. The model was getting over fitted and this is not what was desired.

We also made use of K Nearest Neighbours algorithm but found out that predicting capability of the model dependent on a carefully chosen k-value. So we defined a for loop to compute the appropriate k-value. Using this approach significantly improved our model's predicting capability.

Next we also performed Logistic Regression but this was also accurate to some extent but was not the best approach sense there was no Boolean type data in the data set. Although there was no meaningful correlation between the columns, so we could not find out much information out of it.

We also made use of Apriori analysis to find out the associative features of the products that were purchased by the customer.

We then made use of gradient boosting to predict the products that would be most likely be bought by a customer on the next order. The data was randomly partitioned into train and test with 20 : 80 ratio. We made use of dummy variables to add more features in order to improve the prediction accuracy.

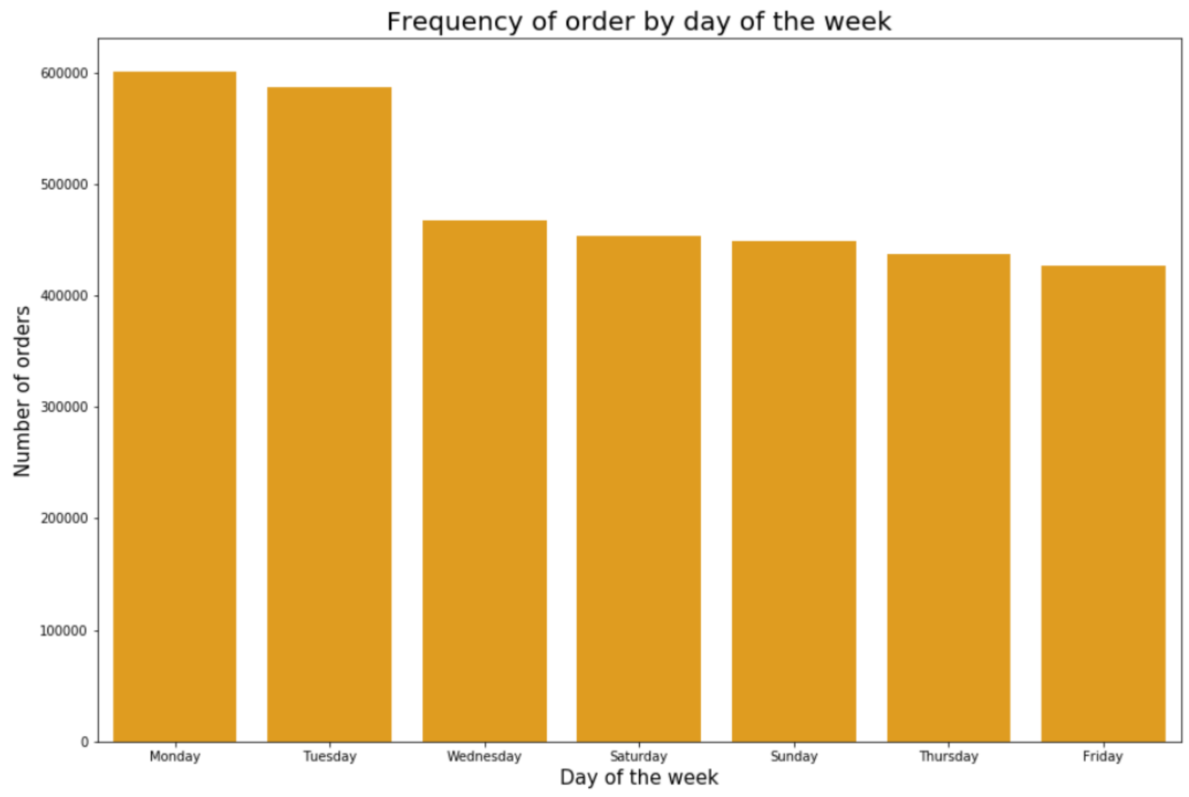
5. Analysis and Results

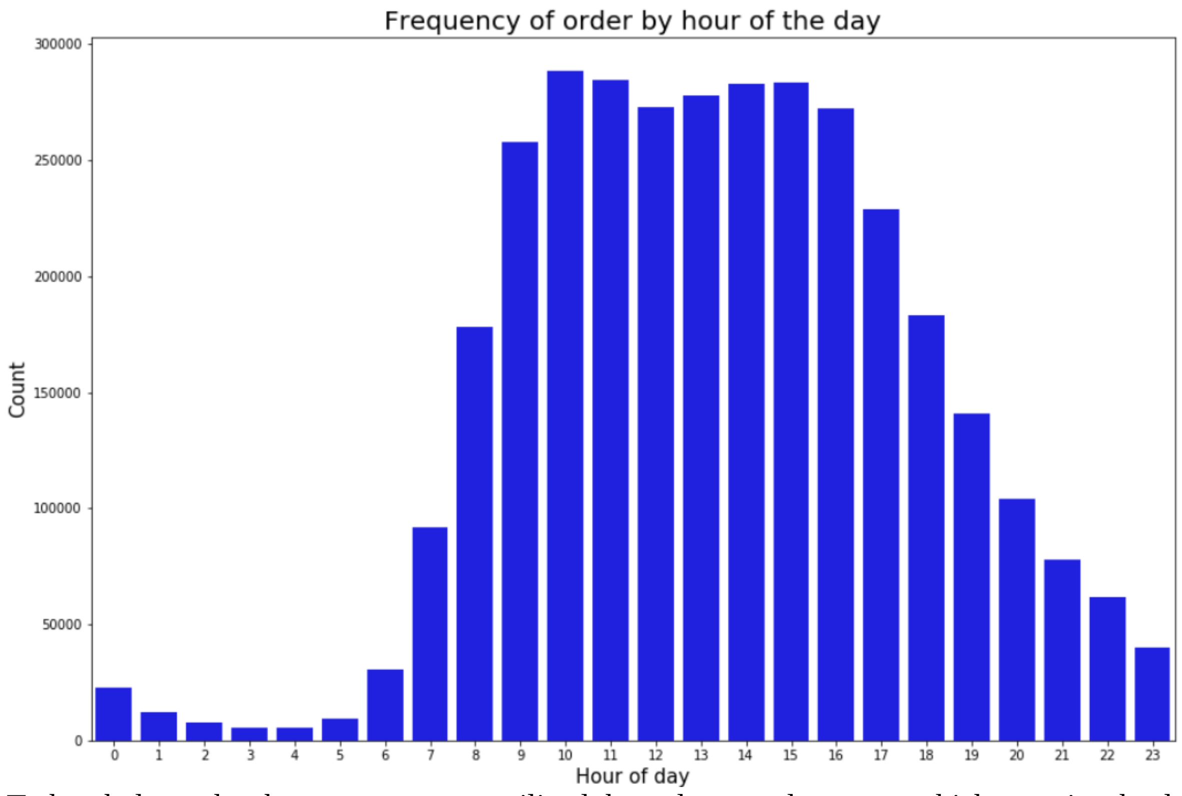
Exploratory Data Analysis:

After performing the Exploratory Data Analysis, we found out that we were able to extract some important trends and features related to the various progressions related to the trends in the customer orders.

Frequency of Orders based on day of the week and hour of the day:

After doing the data analysis we were able to find out that customers were ordering groceries during the weekend and their groceries were being delivered during the start of the week. That is sensible as people are usually busy during the weekdays.

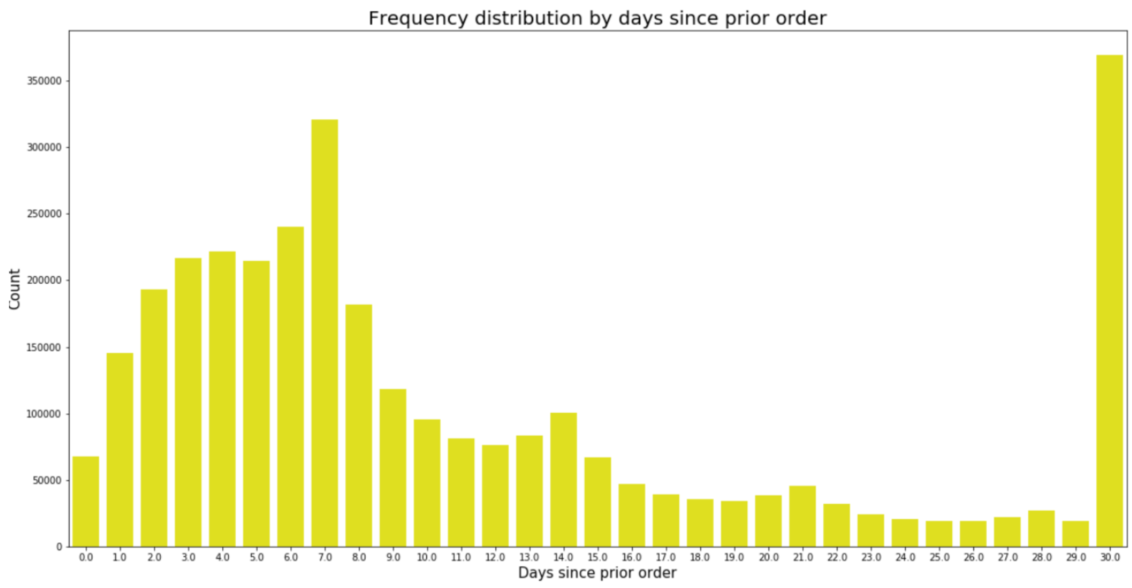




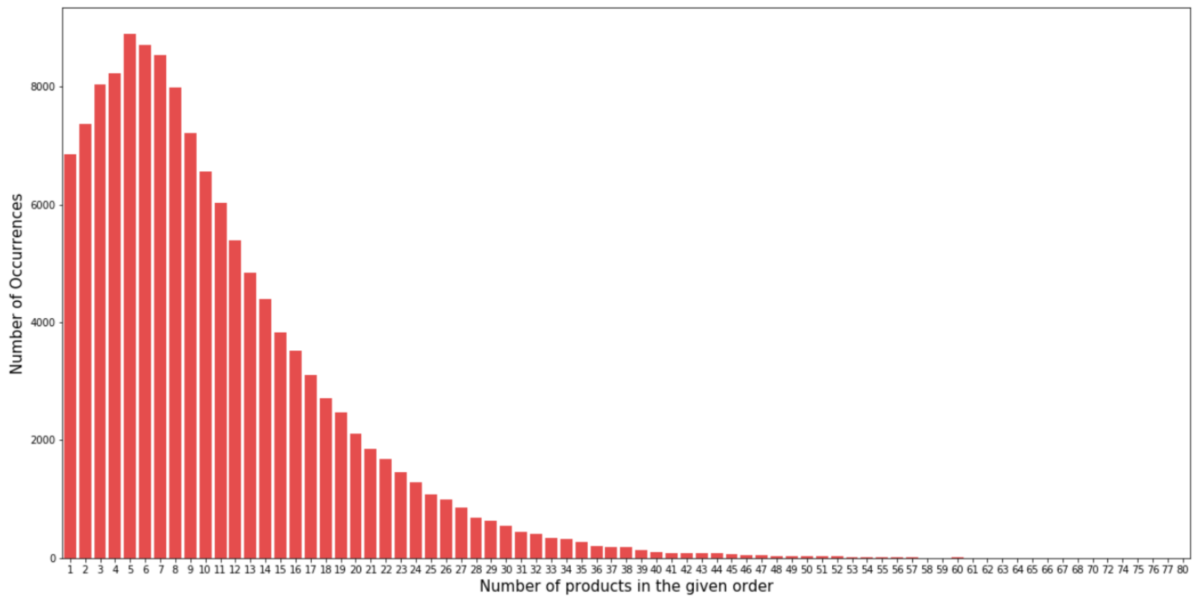
To break down the above patterns, we utilized the orders.csv document which contains the data about the order id and the user id, for example related with the customer. Additionally, there is information about day of the week and, likewise hour of the day on which an order was placed. We were additionally ready to examine that the evening was the busiest time. As should be obvious, according to the bar plot over the hour of 10:00 A.M to 3:00 P.M saw the maximum number of days. In this way, utilizing both the charts, we can presume that afternoon hours during the weekend would be the busiest.

Frequency of re-ordering

We investigated the recurrence of re- ordering of various things, and we discovered that the things are being re-requested at a frequency of 1 week and one month. We have broken down this pattern likewise by extracting data from 'orders.csv' file.

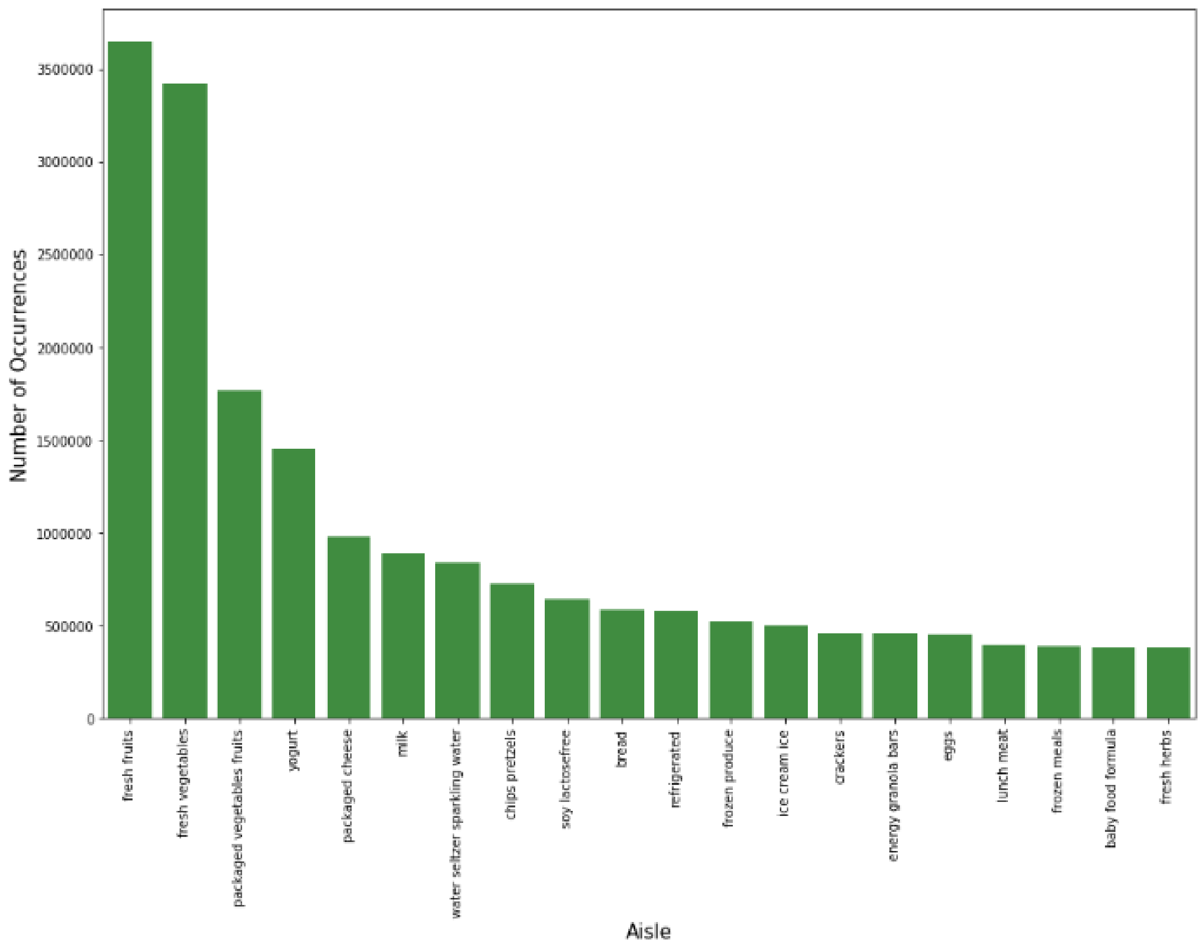


Number of Products Ordered in each order:



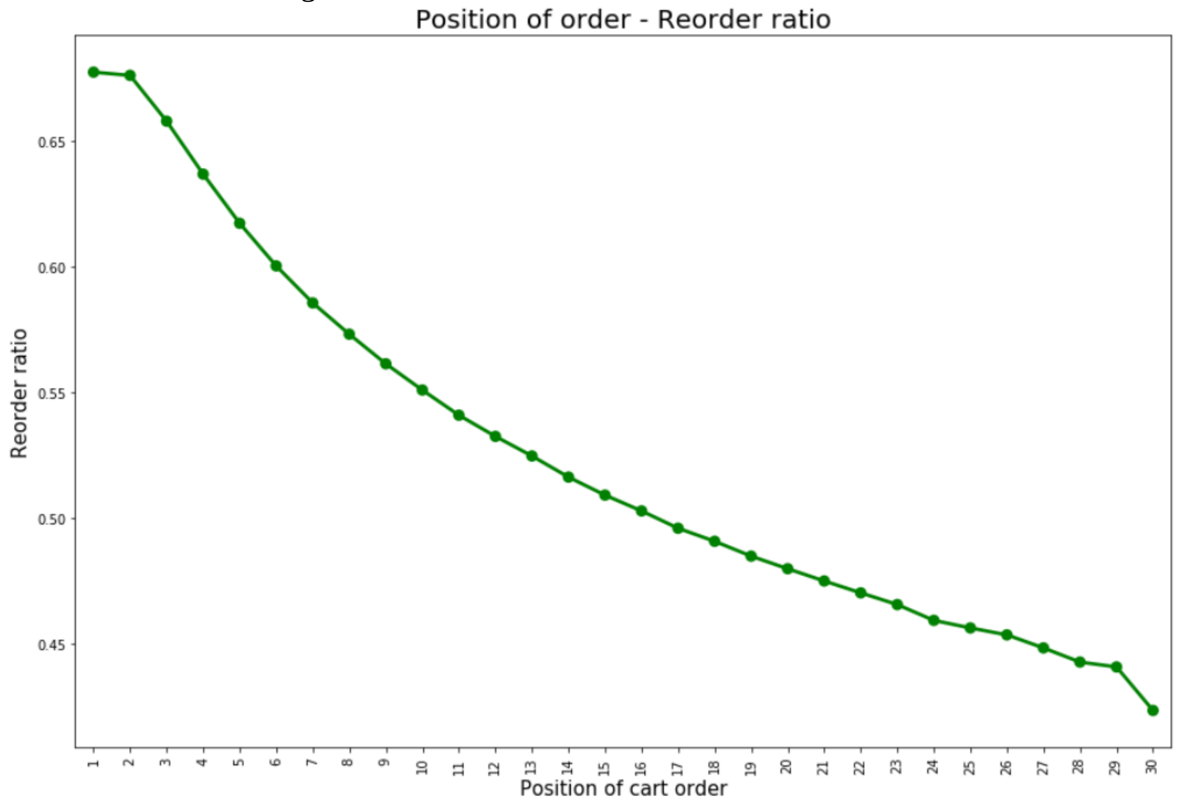
Products with highest order frequency:

According to our research fruits and vegetables are the most ordered products on the app.



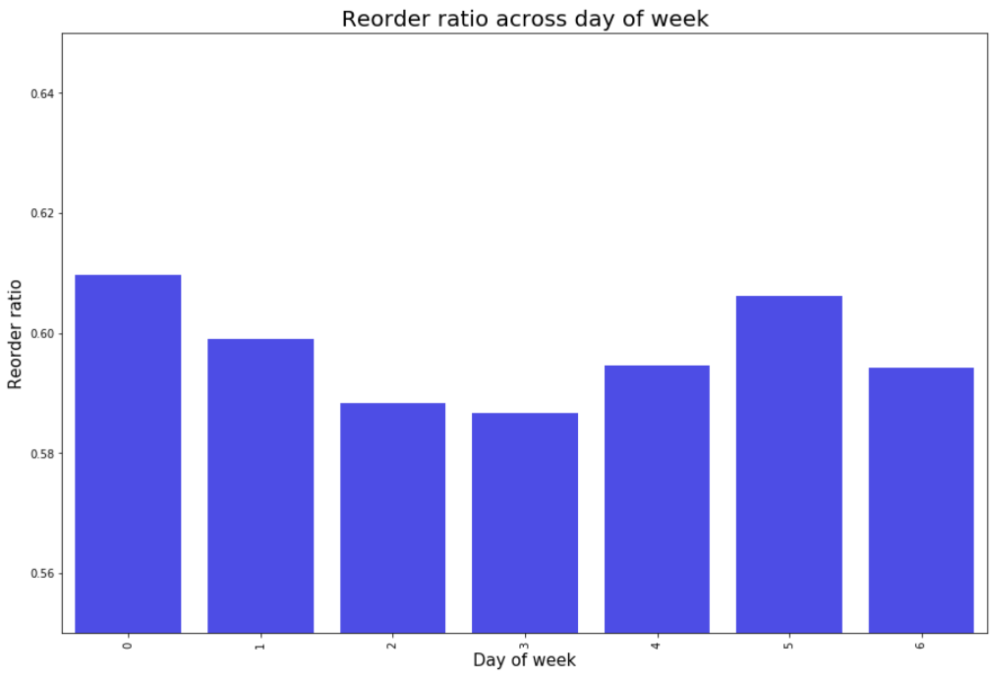
Position of re-ordered products in the cart:

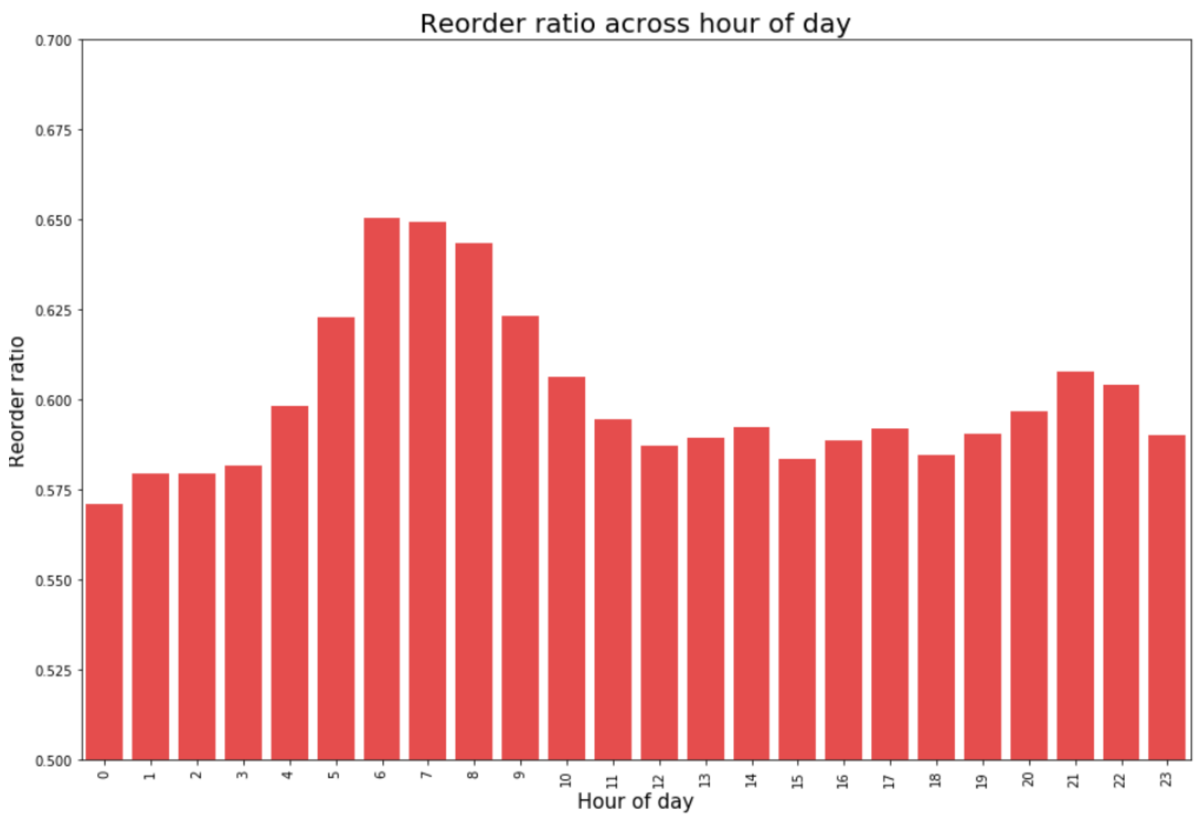
We have analysed that customers generally put the re-order items first in the shopping cart and then order the remaining items.



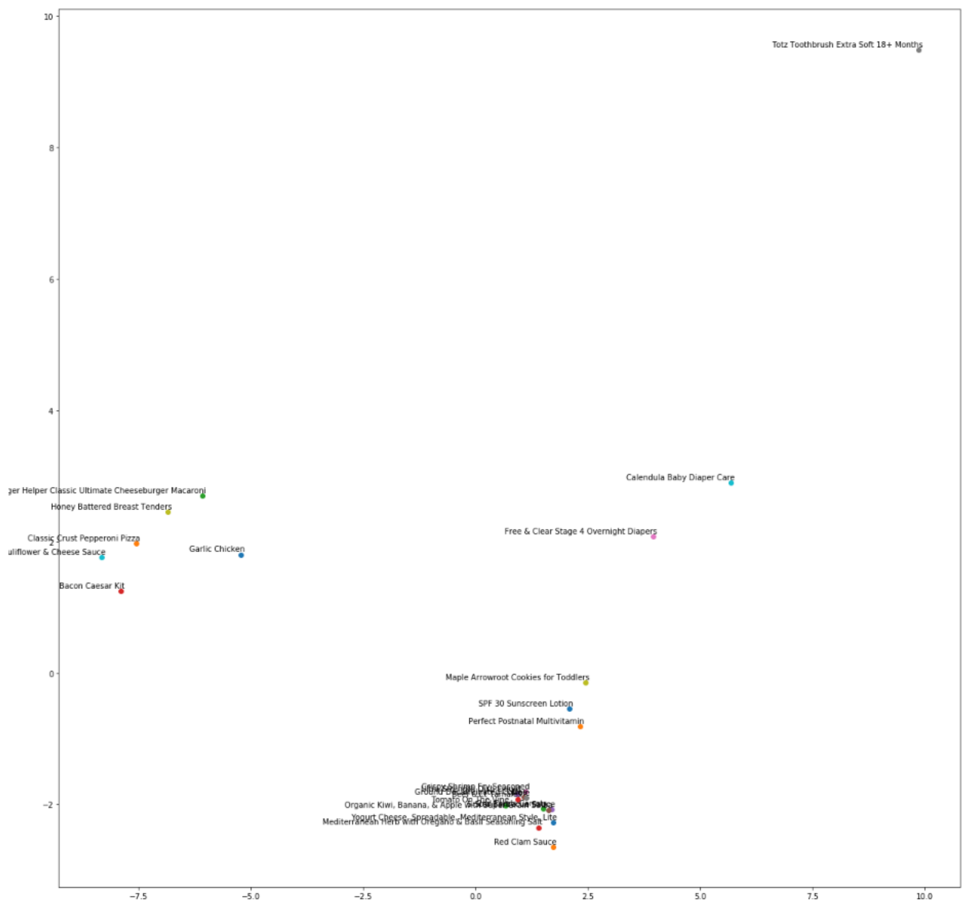
Re-order ration across day of the week and hour of the day:

We concluded that mostly customers tend to re-order during the early (7-8) hours during the weekend.




Reorder ratio across hour of day
Reorder ratio
Hour of day
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Word2Vec

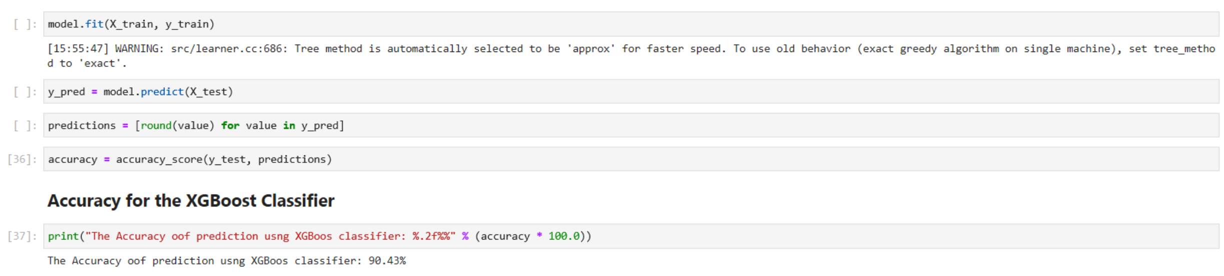

Totz Toothbrush Extra Soft 18+ Months
ger Helper Classic Ultimate Cheeseburger Macaroni
Honey Battered Breast Tenders
Classic Crust Pepperoni Pizza
uliflower & Cheese Sauce
Garlic Chicken
Bacon Caesar Kit
Calendula Baby Diaper Care
Free & Clear Stage 4 Overnight Diapers
Maple Arrowroot Cookies for Toddlers
SPF 30 Sunscreen Lotion
Perfect Postnatal Multivitamin
Crispy Shrimp Fry Seasoned
Tomato On The Vine
Organic Kiwi, Banana, & Apple
Yogurt Cheese, Spreadable, Mediterranean Style, Lite
Mediterranean Herb with Oregano & Basil Seasoning Salt
Red Clam Sauce
10
8
6
4
2
0
−2
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Word2vec is a two-layer neural net that processes content. Its info is a text corpus and its outputs are a set of vectors: highlight vectors for words in the corpus. While Word2vec is certainly not a deep neural system, it transforms content into a numerical structure that deep nets can comprehend. The above figure gives us the data about the relationship of items requested by customers. Comparable things and items are demonstrated to be "close". Their relative implications have been translated to quantifiable distances.

XGBoost:

XGBoost is an optimized distributed gradient boosting library intended to be profoundly proficient, adaptable and compact. It executes Machine Learning Algorithms under the Gradient Boosting structure.

The precision utilizing this came to be 90.43%



```
[ ]: model.fit(X_train, y_train)

     [15:55:47] WARNING: src/learner.cc:686: Tree method is automatically selected to be 'approx' for faster speed. To use old behavior (exact greedy algorithm on single machine), set tree_metho
     d to 'exact'.

[ ]: y_pred = model.predict(X_test)

[ ]: predictions = [round(value) for value in y_pred]

[36]: accuracy = accuracy_score(y_test, predictions)
```

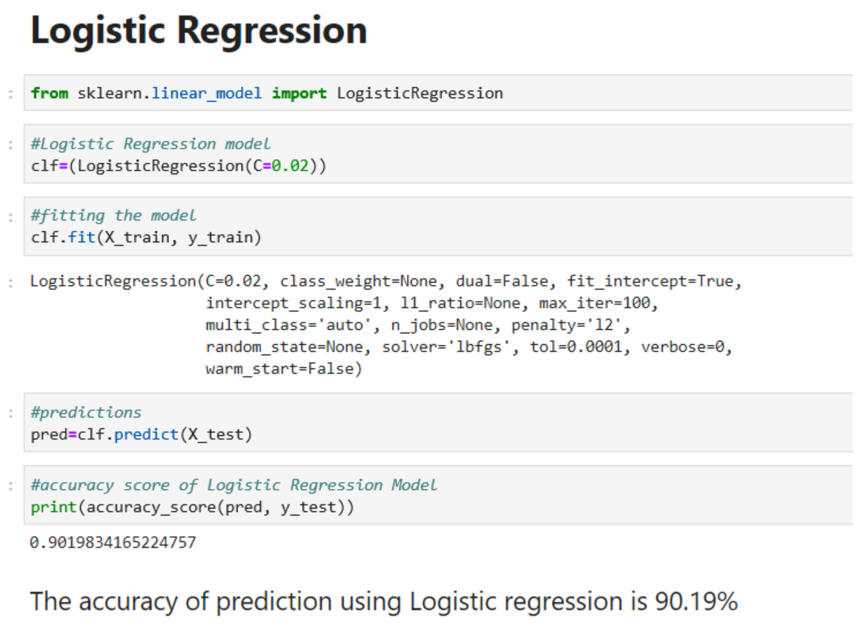
Accuracy for the XGBoost Classifier

```
[37]: print("The Accuracy oof prediction usng XGBoos classifier: %.2f%%" % (accuracy * 100.0))

      The Accuracy oof prediction usng XGBoos classifier: 90.43%
```

Logistic Regression:

Logistic regression is a statistical model that in its fundamental structure a logistic function to model a binary dependent variable which is re-ordering a specific item, albeit a lot increasingly complex augmentations exist.

The exactness of this technique is 90.19%.



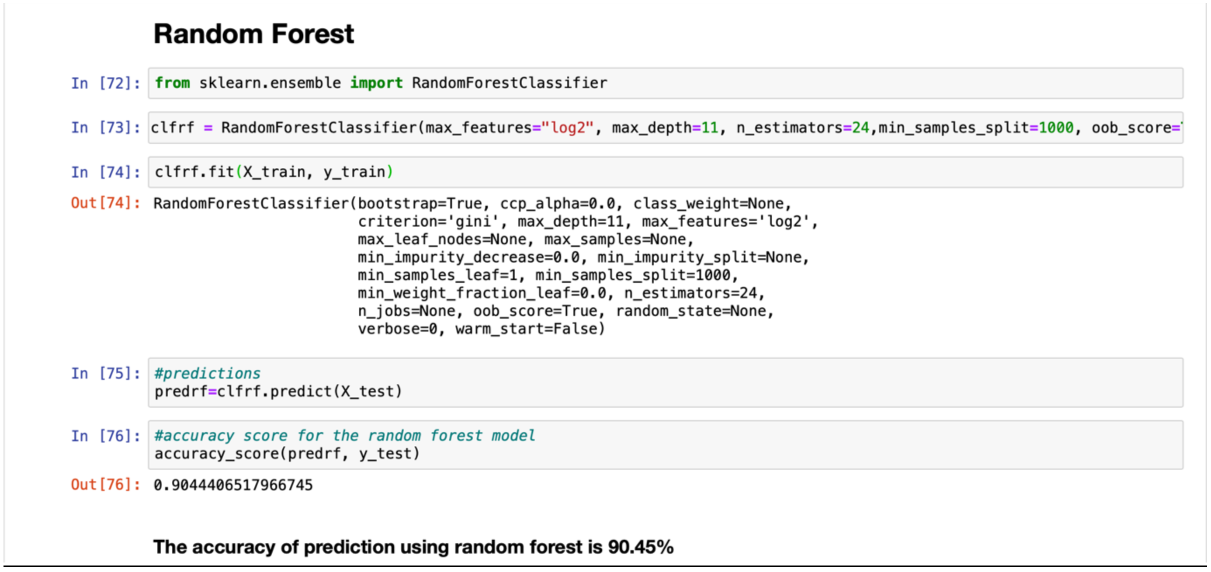
Logistic Regression

```
: from sklearn.linear_model import LogisticRegression

: #Logistic Regression model
  clf=(LogisticRegression(C=0.02))

: #fitting the model
  clf.fit(X_train, y_train)

: LogisticRegression(C=0.02, class_weight=None, dual=False, fit_intercept=True,
                     intercept_scaling=1, l1_ratio=None, max_iter=100,
                     multi_class='auto', n_jobs=None, penalty='l2',
                     random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                     warm_start=False)

: #predictions
  pred=clf.predict(X_test)

: #accuracy score of Logistic Regression Model
  print(accuracy_score(pred, y_test))

  0.9019834165224757
```

The accuracy of prediction using Logistic regression is 90.19%

Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Here this technique has been utilized to group if an item will be re-requested or not.



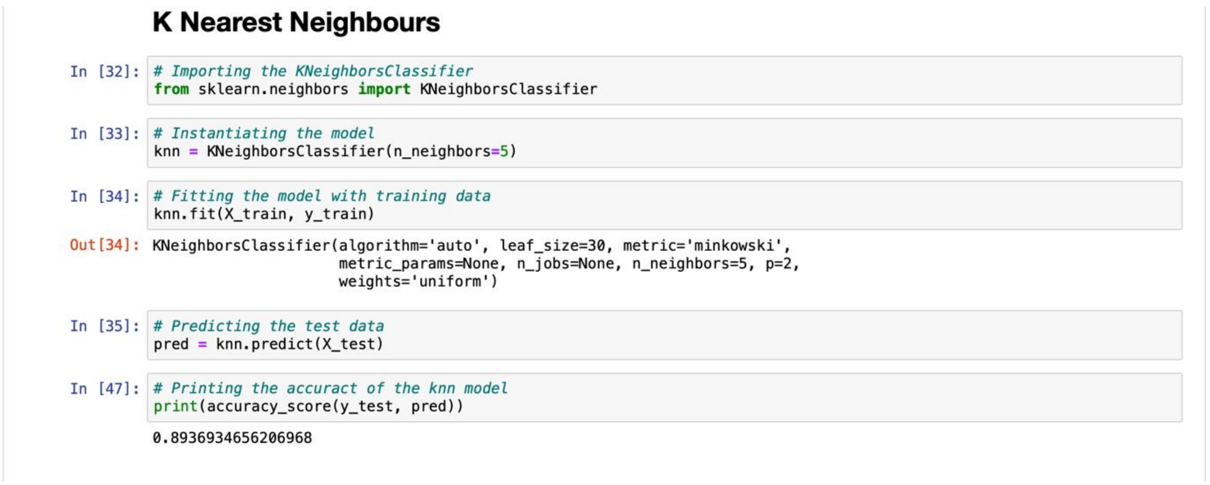
Random Forest

```
In [72]: from sklearn.ensemble import RandomForestClassifier

In [73]: clfrf = RandomForestClassifier(max_features="log2", max_depth=11, n_estimators=24,min_samples_split=1000, oob_score=

In [74]: clfrf.fit(X_train, y_train)

Out[74]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                               criterion='gini', max_depth=11, max_features='log2',
                               max_leaf_nodes=None, max_samples=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=1000,
                               min_weight_fraction_leaf=0.0, n_estimators=24,
                               n_jobs=None, oob_score=True, random_state=None,
                               verbose=0, warm_start=False)

In [75]: #predictions
         predrf=clfrf.predict(X_test)

In [76]: #accuracy score for the random forest model
         accuracy_score(predrf, y_test)

Out[76]: 0.9044406517966745
```

The accuracy of prediction using random forest is 90.45%

K-Nearest Neighbours:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. The algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. Here this technique has been utilized to group if an item will be re-requested or not.



K Nearest Neighbours

```
In [32]: # Importing the KNeighborsClassifier
         from sklearn.neighbors import KNeighborsClassifier

In [33]: # Instantiating the model
         knn = KNeighborsClassifier(n_neighbors=5)

In [34]: # Fitting the model with training data
         knn.fit(X_train, y_train)

Out[34]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                             weights='uniform')

In [35]: # Predicting the test data
         pred = knn.predict(X_test)

In [47]: # Printing the accuract of the knn model
         print(accuracy_score(y_test, pred))

         0.8936934656206968
```

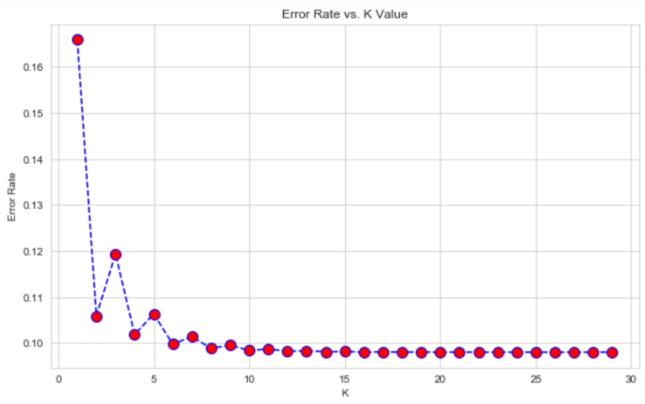
Choosing a K Value

- Let's go ahead and use the elbow method to pick a good K Value!
- Create a for loop that trains various KNN models with different k values, then keep track of the error_rate for each of these models with a list. Refer to the lecture if you are confused on this step.

In [50]:
```
# Defining an empty list to append the error rate values
error_rate = []


for i in range(1,30):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [52]:
```
# Creating a plot to display the Error Rate vs K value
plt.figure(figsize=(10,6))
plt.plot(range(1,30),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[52]:
```
Text(0, 0.5, 'Error Rate')
```



- We find that the Error rate remains relatively constant from k=10 to k=30

In [53]:
```
# Instantiating the KNeighboursClassifier with K=20
knn = KNeighborsClassifier(n_neighbors=20)

# Fitting the knn model
knn.fit(X_train,y_train)

# Predicting the test data
pred_i = knn.predict(X_test)
```

- We find that the Error rate remains relatively constant from k=10 to k=30

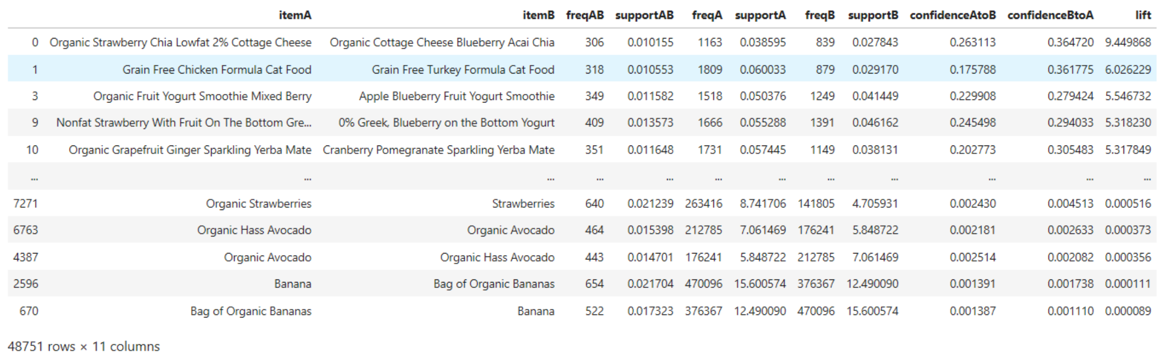
In [53]:
```
# Instantiating the KNeighboursClassifier with K=20
knn = KNeighborsClassifier(n_neighbors=20)

# Fitting the knn model
knn.fit(X_train,y_train)

# Predicting the test data
pred_i = knn.predict(X_test)
```

In [46]:
```
# Printing the accuracy of the model
print(accuracy_score(y_test, pred_i))
```

```
0.9019716166441217
```

The accuracy of prediction using K Nearest Neighbours method is 90.197%

Apriori Algorithm analysis:



	itemA	itemB	freqAB	supportAB	freqA	supportA	freqB	supportB	confidenceAtoB	confidenceBtoA	lift
0	Organic Strawberry Chia Lowfat 2% Cottage Cheese	Organic Cottage Cheese Blueberry Acai Chia	306	0.010155	1163	0.038595	839	0.027843	0.263113	0.364720	9.449868
1	Grain Free Chicken Formula Cat Food	Grain Free Turkey Formula Cat Food	318	0.010553	1809	0.060033	879	0.029170	0.175788	0.361775	6.026229
3	Organic Fruit Yogurt Smoothie Mixed Berry	Apple Blueberry Fruit Yogurt Smoothie	349	0.011582	1518	0.050376	1249	0.041449	0.229908	0.279424	5.546732
9	Nonfat Strawberry With Fruit On The Bottom Gre...	0% Greek, Blueberry on the Bottom Yogurt	409	0.013573	1666	0.055288	1391	0.046162	0.245498	0.294033	5.318230
10	Organic Grapefruit Ginger Sparkling Yerba Mate	Cranberry Pomegranate Sparkling Yerba Mate	351	0.011648	1731	0.057445	1149	0.038131	0.202773	0.305483	5.317849
...	...	...	...	...	...	...	...	...	...	...	...
7271	Organic Strawberries	Strawberries	640	0.021239	263416	8.741706	141805	4.705931	0.002430	0.004513	0.000516
6763	Organic Hass Avocado	Organic Avocado	464	0.015398	212785	7.061469	176241	5.848722	0.002181	0.002633	0.000373
4387	Organic Avocado	Organic Hass Avocado	443	0.014701	176241	5.848722	212785	7.061469	0.002514	0.002082	0.000356
2596	Banana	Bag of Organic Bananas	654	0.021704	470096	15.600574	376367	12.490090	0.001391	0.001738	0.000111
670	Bag of Organic Bananas	Banana	522	0.017323	376367	12.490090	470096	15.600574	0.001387	0.001110	0.000089

48751 rows × 11 columns

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It continues by distinguishing the frequent individual things in the database and extending them to bigger and bigger thing sets as long as those thing sets show up sufficiently regularly in the database. Here we have utilized it to discover associative rules dependent on what items the customers will in general request together.

6. Conclusion

Busiest days of the week: Monday and Tuesday

Busiest Hours of the day: 10 AM to 11 AM

Most popular products: Banana and Organic Banana

Most popular Department: Produce

Most popular aisle: Fresh fruits

Most Re-ordered products: Dairy Eggs

Most Frequently Ordered products pairs: Red Onion and Bag of Organic Bananas

Minimum number of orders per customer is 4 while maximum is 100

Average Frequency of Re-ordering: Weekly and Monthly

Average % of Products which are re-ordered: 59%

Average number of products per order: 5

Best Model for Prediction: Gradient Boost Classifier

Highest accuracy achieved: 90.47%

Products ordered frequently:

- Different types of cottage cheese
- Different types of cat food
- Different types of smoothies
- Different types of yogurt
- Different types of Sparkling Yerba
- Different types of baby food

7. References and Sources

- Rakesh Agrawal and Ramakrishnan Srikant Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowl- edge Discovery and Data Mining, KDD ’16, pages 785–794, New York, NY, USA. ACM.
- https://towardsdatascience.com/a-gentle-introduction-on-market-basket-analysis- association-rules-fa4b986a40ce
- https://en.wikipedia.org/wiki/Apriori_algorithm
- https://en.wikipedia.org/wiki/XGBoost
- https://en.wikipedia.org/wiki/Random_forest
- https://omarito.me/word2vec-product-recommendations/
- https://www.kaggle.com/c/instacart-market-basket-analysis/kernels
- https://en.wikipedia.org/wiki/Word2vec

8. Acknowledgement

We would like to show our gratitude to Professor Handan Liu for guiding us through the semester during these challenging conditions and help us choose a good topic and encourage us during this project. We would also like to acknowledge Instacart for providing their anonymous dataset on Kaggle open for data analysts for practicing the skill of crunching data to get useful information.