



CASE INTERVIEW

DATA ANALYST

by Dr. Ahmed Eshwihdi

[TASK 1 – IDENTIFYING USE CASES BASED ON AVAILABLE DATA 3](#_Toc4562)

[Data Analysis 3](#_Toc21557)

[Data Description and Exploratory Visualisations 3](#_Toc24248)

[Numerical features overview 4](#_Toc3866)

[Correlation 5](#_Toc32613)

[Monthly Revenue: 6](#_Toc5793)

[Monthly Active Customers 8](#_Toc4929)

[Monthly Order Count 9](#_Toc9815)

[New Customer Ratio 9](#_Toc8804)

[Monthly Retention Rate 10](#_Toc24005)

[Cohort Based Retention Rate 11](#_Toc28343)

[TASK 2 – PRIORITISING USE CASES 12](#_Toc15760)

[What is segmentation? 12](#_Toc11050)

[Benefits and drawbacks of segmentation 13](#_Toc30298)

[Customer Segmentation Methods 13](#_Toc12671)

[Cluster Analysis 13](#_Toc24545)

[Pre-processing Pipeline 14](#_Toc21366)

[Normalize the data 14](#_Toc25079)

[Select the optimal number of clusters 14](#_Toc2660)

[Silhouette 14](#_Toc9791)

[The Elbow Method with the Sum of Squared Errors (SSE) 15](#_Toc8314)

[Interpreting Customer Segments 15](#_Toc24791)

[Visualizing Clusters 16](#_Toc2372)

[Find the best-selling item by segment 17](#_Toc12123)

[Lifetime Value Prediction 18](#_Toc2862)

[TASK 3 – IMPLEMENTING A TOP PRIORITY USE CASE 21](#_Toc18653)

[Multi-Classification Model 22](#_Toc10761)

[Apply hyper-parameter tuning to best modes 25](#_Toc20169)

[Normalizing the data 26](#_Toc746)

[Saving Machine Learning Model : Serialization & Deserialization 29](#_Toc19925)

[Exploratory Data Analysis EDA Concluding Remarks 31](#_Toc3035)

# TASK 1 – IDENTIFYING USE CASES BASED ON AVAILABLE DATA

## Data Analysis

The code was performed in a Jupyter notebook using Python 3.x and several Python packages for structuring, processing, analysing, and visualizing the data.

## Data Description and Exploratory Visualisations

First, we import the datasets, join them together and make of a copy of the source file for this analysis. The dataset contains 1,004,841 rows and 15 columns, after including all columns from the other datasets. Based on this data we can select the North star Metric as Monthly Revenue.

Main features:

InvoiceDate', 'CustomerID', 'CountryID', 'ProductID', 'Price',

'Quantity', 'Amount', 'Country\_Code', 'Country', 'Customer\_Code', 'Customer\_Name'

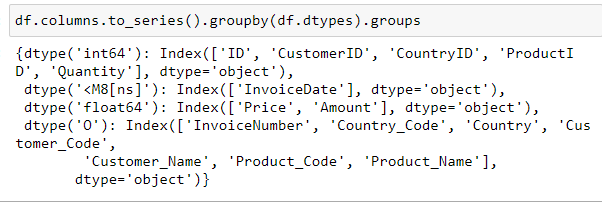
Index(['ID', 'InvoiceNumber', 'InvoiceDate', 'CustomerID', 'CountryID', 'ProductID', 'Price',

'Quantity', 'Amount', 'Country\_Code', 'Country', 'Customer\_Code', 'Customer\_Name',

'Product\_Code', 'Product\_Name'], dtype='object')

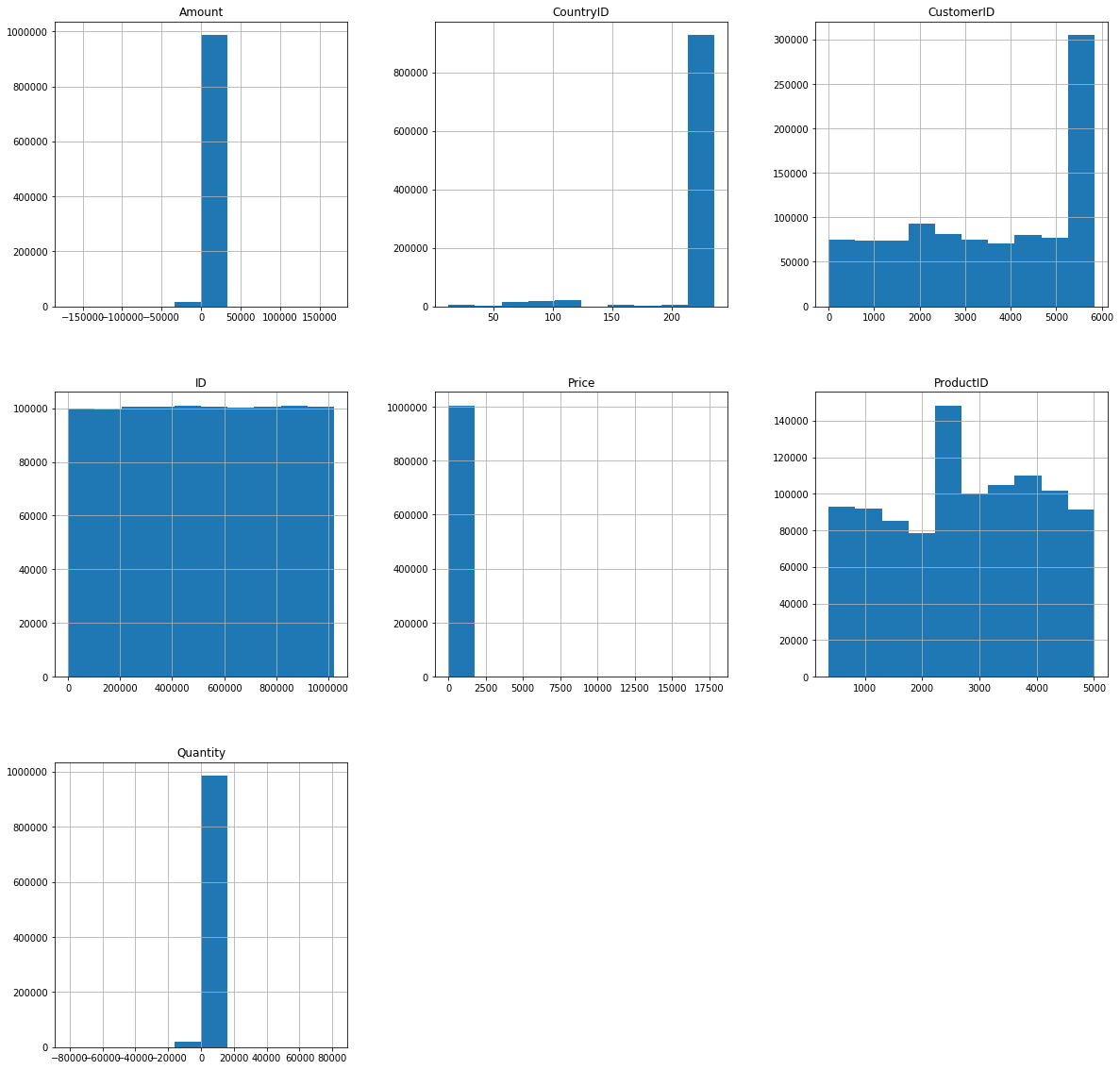
The dataset contains several numerical and categorical columns providing various information on orders and customers’ details.

Let’s break down the columns by their type (i.e. int64, float64, object):

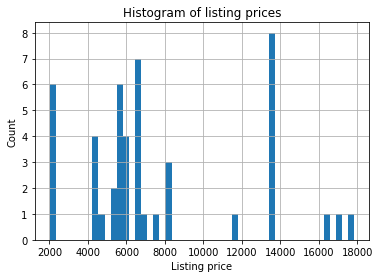
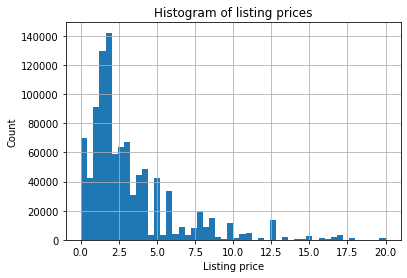


## Numerical features overview

A few observations can be made based on the information and histograms for numerical features

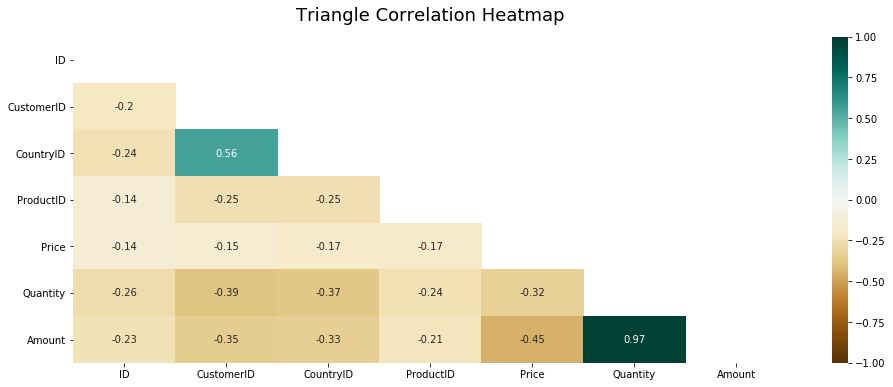
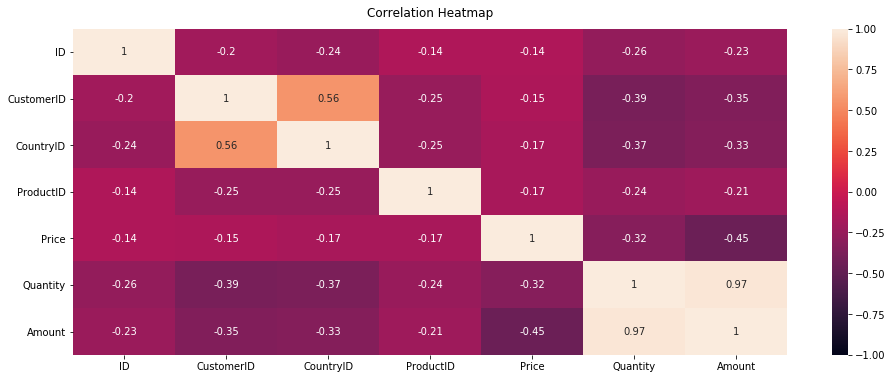


Clearly most of the orders come from the United Kingdom. If we focus on Price Histogram, we can see that most of the purchased products’ price is less than 5 :



## Correlation

Let’s take a look at some of most significant correlations. It is worth remembering that correlation coefficients only measure linear correlations.



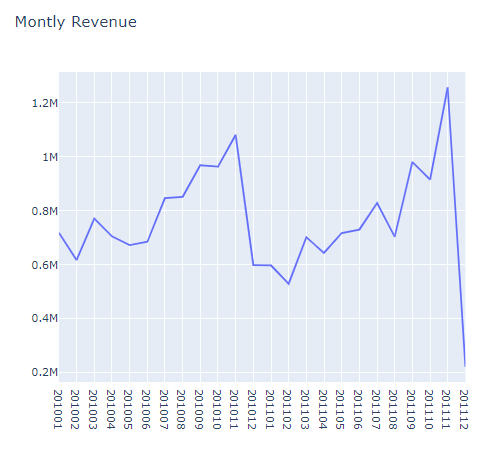
As shown above, CountryID, CustomerID, Amount and Quantity are positively correlated to Attrition; while ProductID and Price are negatively correlated to Attrition.

## Monthly Revenue:

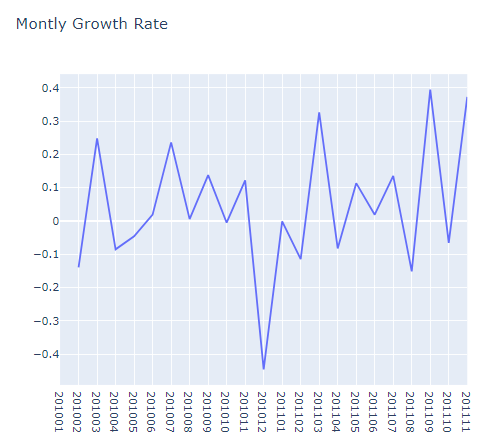
With all these features, we can build our North Star Metric equation:

Revenue = Active Customer Count \* Order Count \* Average Revenue per Order

Next step, visualization. A line graph would be sufficient:



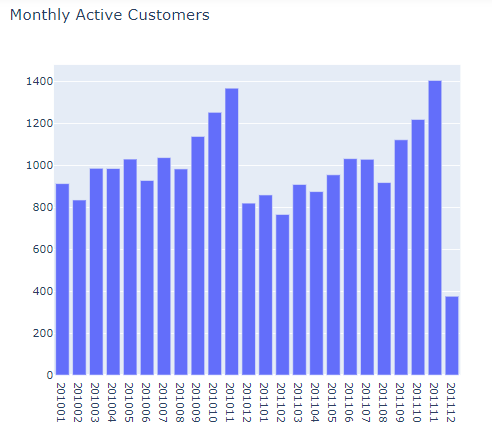
This clearly shows our revenue is growing especially Feb 2011 onwards (and our data in December is incomplete). Let’s figure out what is our Monthly Revenue Growth Rate:



Everything looks good, we saw 36.5% growth previous month (December is excluded in the code since it hasn’t been completed yet). But we need to identify what exactly happened on Dec 2010. Was it due to less active customers or our customers did less orders? Maybe they just started to buy cheaper products?

## Monthly Active Customers

To see the details Monthly Active Customers, we will follow the steps we exactly did for Monthly Revenue. Starting from this part, we will be focusing on UK data only (which has the most records). We can get the monthly active customers by counting unique CustomerIDs. Code snippet and the output are as follows:

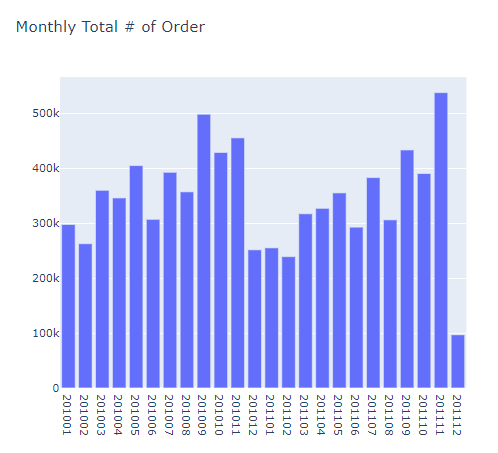


In Dec 2010, Monthly Active Customer number dropped to 820 from 1,368 (-40.1%).

We will see the same trend for number of orders as well.

## Monthly Order Count

We will apply the same code by using Quantity field:



As we expected, Order Count is also declined in Dec 2010 (455k to 251k, -44.8%)

We know that Active Customer Count directly affected Order Count decrease. At the end, we should definitely check our Average Revenue per Order as well.

We observed slow-down in every metric affecting our North Star (Monthly Revenue).

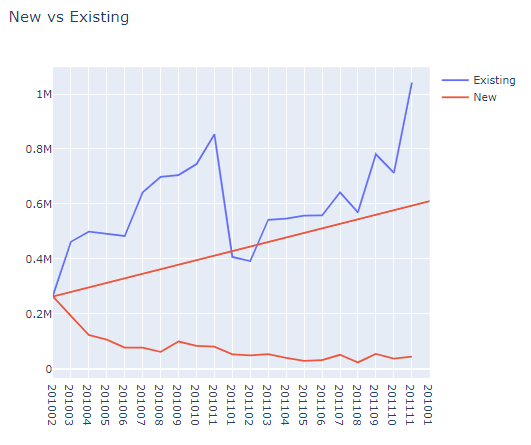
We have looked at our major metrics. Of course there are many more and it varies across industries. Let’s continue investigating some other important metrics:

* New Customer Ratio: a good indicator of if we are losing our existing customers or unable to attract new ones
* Retention Rate: King of the metrics. Indicates how many customers we retain over specific time window. We will be showing results for monthly retention rate and cohort based retention rate.

## New Customer Ratio

First we should define what is a new customer. In our dataset, we can assume a new customer is whoever did his/her first purchase in the time window we defined. We will do it monthly for this case.

We will be using .min() function to find our first purchase date for each customer and define new customers based on that.



Existing customers are showing a positive trend and tell us that our customer base is growing but new customers have a slight negative trend.

## Monthly Retention Rate

Retention rate should be monitored very closely because it indicates how sticky is your service and how well your product fits the market. For making Monthly Retention Rate visualized, we need to calculate how many customers retained from previous month.

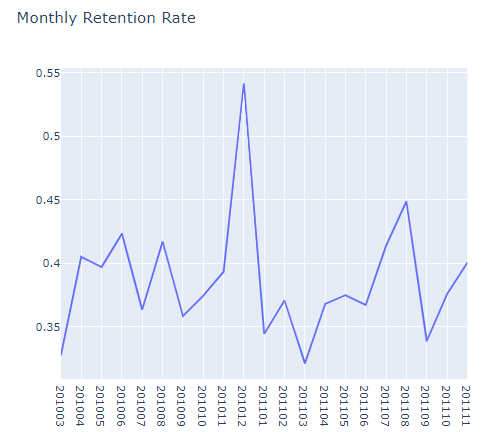
Monthly Retention Rate = Retained Customers From Prev. Month/Active Customers Total

We will be using crosstab() function of pandas.

First, we create a dataframe that shows total monthly revenue for each customer.

In the Retention table, we put active customers on each month as 1, stands for active.

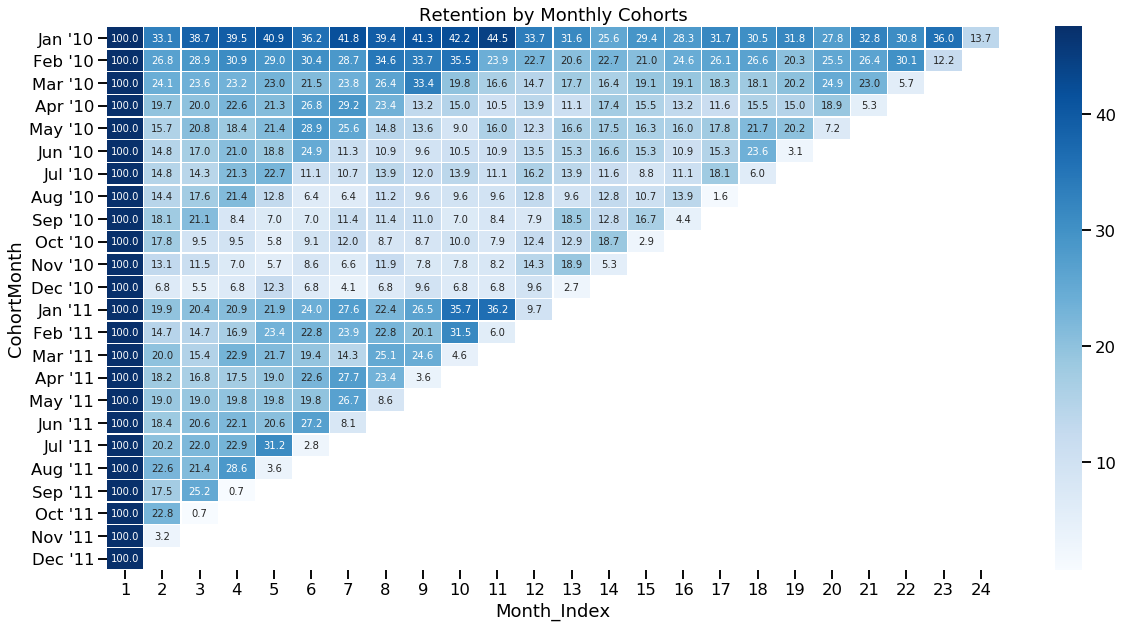
We calculate Retained Customer Count from previous month and Total Customer Count.



Monthly Retention Rate jumped significantly from Oct 2011 to Dec 2011 and went back to previous levels afterwards.

## Cohort Based Retention Rate

There is another way of measuring Retention Rate which allows us to see Retention Rate for each cohort. Cohorts are determined as first purchase year-month of the customers. We will be measuring what percentage of the customers retained after their first purchase in each month. This view will help us to see how recent and old cohorts differ regarding retention rate and if recent changes in customer experience affected new customer’s retention or not.



We can see that first month retention rate became better recently and in 2010, 13.7% of our customers retain with us.

# TASK 2 – PRIORITISING USE CASES

The use cases that can be identified using the data are:

* **Customer Segmentation**
* **Customer Lifetime Value Prediction (CLV)**
* Predicting Next Purchase Day
* Predicting Sales

However, we will cover two use cases: Customer Segmentation and Customer Lifetime Value Prediction

Customer Segmentation

Consumers are inundated with information; more information than ever before. Our brains have learned to ignore or otherwise become confused due to the enormous amounts of information we consume daily. This has further complicated the field of marketing, and now businesses must leverage analytic to better understand their customers, and how to attract them.

## What is segmentation?

Segmentation, either market or customer segmentation, has become a staple in the modern marketer’s toolbox. Market segmentation is the process of grouping consumers based on meaningful similarities. Segments are typically identified by geographic, demographic, psychographic, or behavioral characteristics.

Segmentation is used to inform several parts of a business, including product development, marketing campaigns, direct marketing, customer retention, and process optimization.

Put simply, segmentation allows you to better understand your customers.

## Benefits and drawbacks of segmentation

Customer segmentation is more than just a way for businesses to optimize their marketing campaigns. Customer segmentation is a two-way street.

Both consumers and companies benefit.

*Consumers benefit because:*

1. They feel like companies have their best interest in mind.
2. Content and products address and fulfill their needs.

*Companies benefit because they can:*

1. Optimize their marketing spend
2. Increase customer lifetime value (CLV)
3. Improve customer service and customer experience
4. Implement optimal marketing channel selection for their each segment
5. Improve product features and offerings
6. Identify and cater to most profitable customers

Here are the disadvantages of clustering:

1. Customer groups created may not be easily interpretable.
2. If data is not based on consumer behaviour (such as products or services purchased), it may not be clear how to use the clusters that are found.

As you can see, one downside of clustering is that it may find groups that don’t seem to make a lot of sense on the surface. Often this can be fixed by using a better suited clustering algorithm.

## Customer Segmentation Methods

There are numerous methods to perform segmentation, data requirements, and purpose. The following methods are some of the most broadly used, but this is not an exhaustive list.

## Cluster Analysis

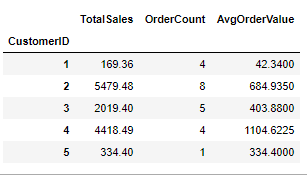
Clustering is an unsupervised learning technique which is exploratory in nature and does not have a defined target or output. Clustering is often done to undercover hidden patterns within a dataset or for real-world uses such as market segmentation. Cluster analysis is a method of grouping, or clustering, consumers based on their similarities.

There are 2 primary types of cluster analysis leveraged in market segmentation: hierarchical cluster analysis, and partitioning. For now, we’re going to discuss a partitioning cluster method called k-means. The main differences between Hierarchical and Partition Clustering are that each cluster starts as individual clusters or singletons.

The k-mean clustering algorithm is a frequently used algorithm for drawing insights into the formations and separations within data. In marketing, it is often used to build customer segments and understand the behaviours of these different segments.

## Pre-processing Pipeline

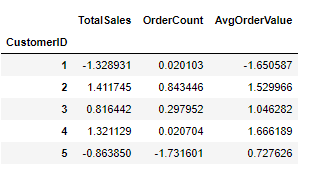
In this section, we undertake data pre-processing steps to prepare the datasets for Machine Learning algorithm implementation. Now let’s transform the data so that each record represents a single customer’s purchase history.



We now have a DataFrame with total sales, order count, and average order value for each customer.

## Normalize the data

Clustering algorithms like K-means are sensitive to the scales of the data used, so we’ll want to normalize the data.



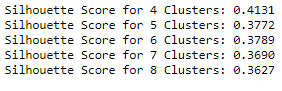
Our data is scaled between -2 and 2. Now let’s get to clustering.

## Select the optimal number of clusters

we’re ready to run cluster analysis. But first, we need to figure out how many clusters we want to use. There are several approaches to selecting the number of clusters to use, but we are going to cover two in this article: (1) silhouette coefficient, and (2) the elbow method.

### Silhouette

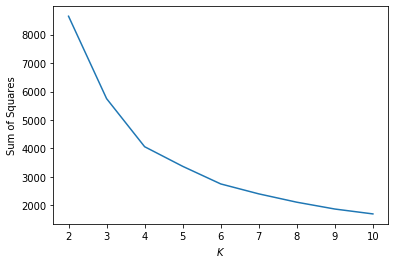
By implementing Silhouette coefficient to the code we can find the ideal number of clusters.



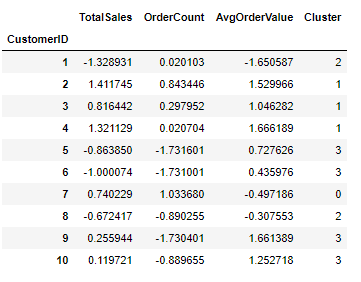
Cluster 4 had the highest silhouette coefficient, indicating 4 would be the best number of clusters. But we’re going to double-check that with the elbow method.

### The Elbow Method with the Sum of Squared Errors (SSE)

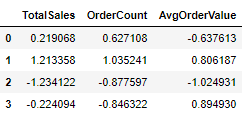
Based on the graph above, it looks like K=4, or 4 clusters is the optimal number of clusters for this analysis. Now let’s interpret the customer segments provided by these clusters.



## Interpreting Customer Segments

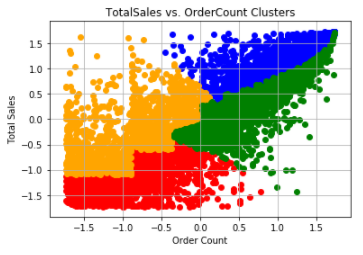


Now let’s group the cluster metrics and see what we can gather from the normalized data for each cluster.

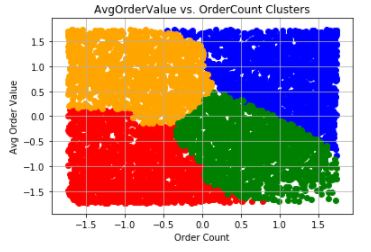


## Visualizing Clusters

For this next piece, we are going to visualize the clusters by putting the different columns on the x and y-axes. Let’s see what we get.



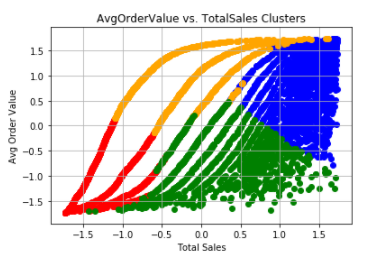
The customers in red have low total sales and low order count, meaning they are all-around low-value customers. On the other hand, the customers in blue have high total sales and high order counts, indicating they are the highest value customers.



In this plot, we’re looking at the average order value vs the order count. Once again, the customers in red are the lowest value customers and the customers in blue are the highest value customers.

We could look at this in another way. We could look at the customers in the orange cluster and attempt to find ways to increase their order count with email reminders or SMS push notifications targeted based on some other identifying factors. Maybe we could email them a discount if they return within 30 days. Better yet, we can offer a delayed coupon (to be used in a specific time period) upon checkout.

Likewise, with customers in the green segment, you might want to try some cross-selling and up-selling techniques at the cart. Maybe a quick pop-up with an offer, based on market basket analysis (see the market basket analysis section below).



In this plot, we have the average order value versus total sales clusters. This plot further substantiates the previous 2 plots in identifying the blue cluster as the highest value customers, red as the lowest value customers, and the orange and green as high opportunity customers.

From a growth perspective, we’d focus my attention on the orange and green clusters. We’d attempt to better understand each cluster and their granular behaviour on-site in order to identify which cluster to focus on first and inform the first few rounds of experiments.

Customers in Red Cluster %26.65

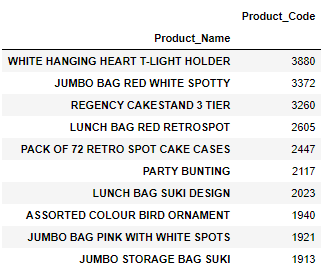
Customers in Blue Cluster %23.96

Customers in Green Cluster %26.28

Customers in Orange Cluster %23.11

## Find the best-selling item by segment

We know that we have 4 segments and know how much they spend per purchase, their total spending, and their number of orders. The next thing we can do that will help us better understand the customer segments is to identify which items are the best-selling within each segment.



Based on this information, we now know that the WHITE HANGING HEART T-LIGHT HOLDER is the best-selling item for our highest-value cluster. With that information in hand, we can make recommendations of **Other Items You Might Like** to customers within this segment.

Customer segmentation can have an incredible impact on a business when done well. We could potentially increase the sales by 50%, if we target the customers in green and orange clusters.

Customer Lifetime Value Prediction (CLV)

In the previous case, we segmented our customers and found out who are the best ones. Now it’s time to measure one of the most important metric we should closely track: Customer Lifetime Value.

Companies invest in customers (acquisition costs, off-line ads, promotions, discounts & etc.) to generate revenue and be profitable. Naturally, these actions make some customers super valuable in terms of lifetime value but there are always some customers who pull down the profitability. We need to identify these behaviour patterns, segment customers and act accordingly.

Calculating Lifetime Value is the easy part. First we need to select a time window. It can be anything like 3, 6, 12, 24 months. By the equation below, we can have Lifetime Value for each customer in that specific time window: *Lifetime Value =Total Gross Revenue -Total Cost*

This equation now gives us the historical lifetime value. If we see some customers having very high negative lifetime value historically, it could be too late to take an action. At this point, we need to predict the future with machine learning. We are going to build a simple machine learning model that predicts our customers lifetime value.

## Lifetime Value Prediction

Let’s identify our steps:

* Define an appropriate time frame for Customer Lifetime Value calculation
* Identify the features we are going to use to predict future and create them
* Calculate lifetime value (LTV) for training the machine learning model
* Build and run the machine learning model
* Check if the model is useful

Deciding the time frame really depends on the industry, business model, strategy and more. For some industries, 1 year is a very long period while for the others it is very short. In our case, we will use with 6 months.

RFM scores for each customer ID, which we calculated in the previous case, are the perfect candidates for feature set. To implement it correctly, we need to split our dataset. We will take 3 months of data, calculate RFM and use it for predicting next 6 months. So we need to create two dataframes first and append RFM scores to them.

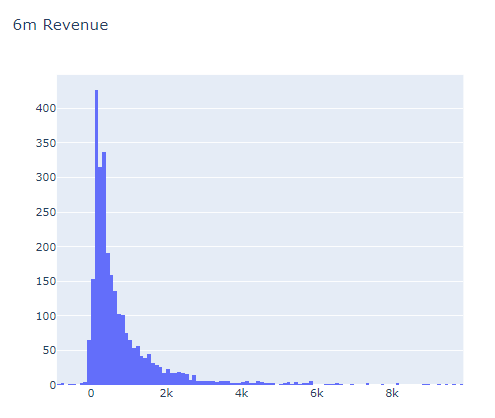
* how recently a customer has purchased (recency)
* how often they purchase (frequency)
* how much the customer spends (monetary)

RFM helps to identify customers who are more likely to respond to promotions by segmenting them into various categories.

Since our feature set is ready, let’s calculate 6 months LTV for each customer which we are going to use for training our model.

There is no cost specified in the dataset. That’s why Revenue becomes our LTV directly.

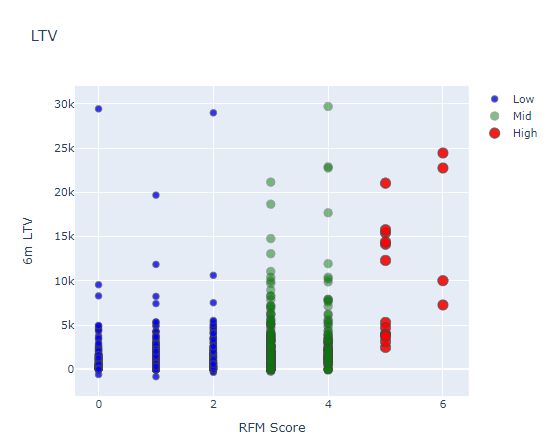
This code snippet calculates the LTV and plot its histogram:



Histogram clearly shows we have customers with negative LTV. We have some outliers too. Filtering out the outliers makes sense to have a proper machine learning model.

Next step. We will merge our 3 months and 6 months dataframes to see correlations between LTV and the feature set we have.

The code below merges our feature set and LTV data and plots LTV vs overall RFM score:



Positive correlation is quite visible here. High RFM score means high LTV.

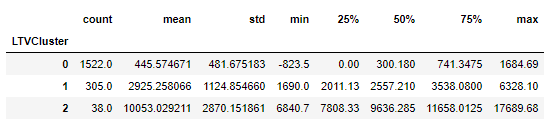
Before building the machine learning model, we need to identify what is the type of this machine learning problem. LTV itself is a regression problem. But here, we want LTV segments. Because it makes it more actionable and easy to communicate with other people. By applying K-means clustering, we can identify our existing LTV groups and build segments on top of it.

Considering business part of this analysis, we need to treat customers differently based on their predicted LTV. For this example, we will apply clustering and have 3 segments (number of segments really depends on your business dynamics and goals) in this case we chose 3 clusters from the previous case:

* Low LTV
* Mid LTV
* High LTV

We are going to apply K-means clustering to decide segments and observe their characteristics:

We have finished LTV clustering and here are the characteristics of each clusters:



2 is the best with average 10.5k LTV whereas 0 is the worst with 445.

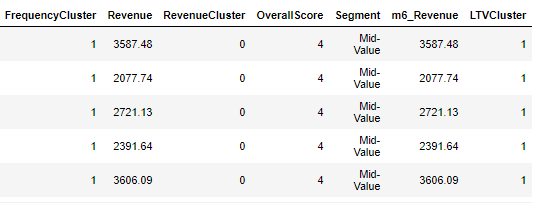
# TASK 3 – IMPLEMENTING A TOP PRIORITY USE CASE

There are few more step before training any machine learning model.

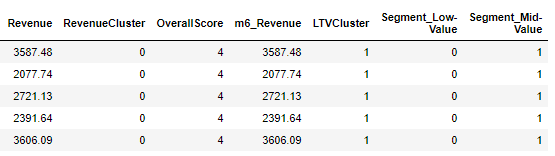
Need to do some feature engineering. We should convert categorical columns to numerical columns.

* We will check the correlation of features against our label, LTV clusters.
* We will split our feature set and label (LTV) as X and y. We use X to predict y.
* Will create Training and Test dataset. Training set will be used for building the machine learning model. We will apply our model to Test set to see its real performance.

Let’s start with the first line. get\_dummies() method converts categorical columns to 0–1 notations. See what it exactly does with the example:



The above table shows our dataset before get\_dummies(). We have one categorical column which is Segment. The table below shows what happens after applying get\_dummies():



Segment column is gone but we have new numerical ones which represent it. We have converted it to 3 different columns with 0 and 1 and made it usable for our machine learning model.

## Multi-Classification Model

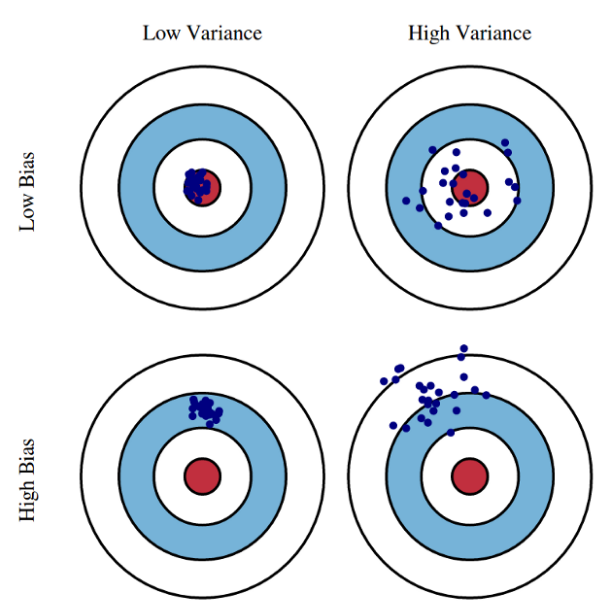
Since we have the training and test sets we can build our models. We will run the list of models below on our training and test sets:

* Logistic Regression (LR)
* Gaussian Naive Bayes (NB )
* Random Forest Classifier (RF)
* Support Vector Classifier (SVC)
* DecisionTreeClassifier (Dtree)
* XGBoost Classifier (XGB)
* K Neighbors Classifier (KNN)

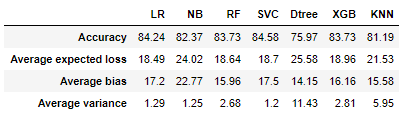
We will include Bias-Variance Tradeoff results in our tables to help us to:

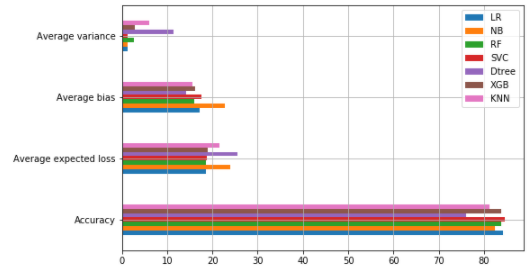
* Avoid over-fitting and under-fitting problems
* Have consistence in prediction

The Mean Squared Error (MSE) of a statistical model can be expressed as the sum of the squared bias of its predictions, the variance of those predictions, and the variance of some error term. The image below to be particularly good at illustrating what the two terms mean.



The table below summarizes the results of running all the models fallowed by a graph to visualise the results:





From the accuracy in the results table above, we can see that the best performing models are SVC, LR, XGB and RF respectively. The worse performance model is Dtree. However, after checking Bias-Variance Tradeoff for the top models, we can see that LR has the lowest Bias and SVC has the lowest variance. We have included Cross Validation as on of the ways of measuring model stability. It provides the score of the model by selecting different test sets. If the deviation is low, it means the model is stable.

A full report of running all the models below:

LR 0.8476394849785408

precision recall f1-score support

0 0.85 0.99 0.91 376

1 0.73 0.22 0.34 86

2 0.67 0.40 0.50 5

accuracy 0.84 467

macro avg 0.75 0.54 0.58 467

weighted avg 0.83 0.84 0.80 467

Confusion Matrix

[[372 4 0]

[ 66 19 1]

[ 0 3 2]]

NB 0.8133047210300429

precision recall f1-score support

0 0.86 0.94 0.90 376

1 0.51 0.28 0.36 86

2 0.50 0.80 0.62 5

accuracy 0.82 467

macro avg 0.62 0.67 0.62 467

weighted avg 0.79 0.82 0.80 467

Confusion Matrix

[[354 22 0]

[ 58 24 4]

[ 0 1 4]]

RF 0.8433476394849786

precision recall f1-score support

0 0.86 0.97 0.91 376

1 0.65 0.28 0.39 86

2 1.00 0.40 0.57 5

accuracy 0.84 467

macro avg 0.83 0.55 0.62 467

weighted avg 0.82 0.84 0.81 467

Confusion Matrix

[[366 10 0]

[ 62 24 0]

[ 0 3 2]]

SVC 0.8433476394849786

precision recall f1-score support

0 0.83 1.00 0.90 376

1 0.64 0.08 0.14 86

2 1.00 0.40 0.57 5

accuracy 0.82 467

macro avg 0.82 0.49 0.54 467

weighted avg 0.79 0.82 0.76 467

Confusion Matrix

[[375 1 0]

[ 79 7 0]

[ 0 3 2]]

Dtree 0.7467811158798283

precision recall f1-score support

0 0.86 0.88 0.87 376

1 0.36 0.30 0.33 86

2 0.23 0.60 0.33 5

accuracy 0.77 467

macro avg 0.49 0.59 0.51 467

weighted avg 0.76 0.77 0.77 467

Confusion Matrix

[[330 44 2]

[ 52 26 8]

[ 0 2 3]]

XGB 0.8540772532188842

precision recall f1-score support

0 0.86 0.98 0.91 376

1 0.67 0.26 0.37 86

2 0.50 0.60 0.55 5

accuracy 0.84 467

macro avg 0.67 0.61 0.61 467

weighted avg 0.82 0.84 0.81 467

Confusion Matrix

[[367 9 0]

[ 61 22 3]

[ 0 2 3]]

KNN 0.8390557939914163

precision recall f1-score support

0 0.85 0.95 0.90 376

1 0.50 0.22 0.31 86

2 0.60 0.60 0.60 5

accuracy 0.82 467

macro avg 0.65 0.59 0.60 467

weighted avg 0.78 0.82 0.79 467

Confusion Matrix

[[359 17 0]

[ 65 19 2]

[ 0 2 3]]

## Apply hyper-parameter tuning to best modes

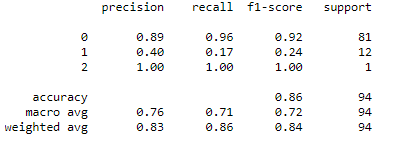
For improving a model further, we’ll do Hyperparameter Tuning. Programmatically, we will find out what are the best parameters for our model to make it provide the best accuracy.

Our score increased from 85% to 86%. It is an improvement.

XGBClassifier has many parameters. For this example, we will select max\_depth and min\_child\_weight.

Accuracy shows 86% on the test set. First we need to check our benchmark. Biggest cluster we have is cluster 0 which is (1522) 81.6% of the total base. If we blindly say, every customer belongs to cluster 0, then our accuracy would be 81.5%.

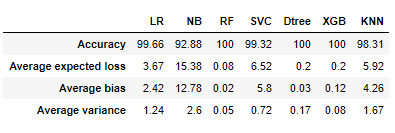
86% vs 81.5% tell us that our machine learning model is a useful one but needs some improvement.



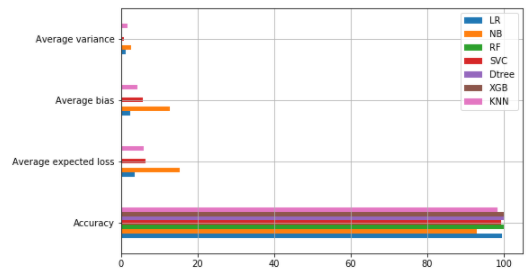
Precision and recall are acceptable for 0. As an example, for cluster 0 (Low LTV), if model tells us this customer belongs to cluster 0, 89 out of 100 will be correct (precision). And the model successfully identifies 96% of actual cluster 0 customers (recall). We need to improve the model for other clusters. For example, we barely detect 17% of Mid LTV

## Normalizing the data

We’ve found that by normalizing the data we increased the accuracy for almost all the models to reach 99%. Classification reports for all the models’ performance below:



We can see from the table above that there are few good performance models, like RF, XGB. These two models were in the top performance models previously. Again we can see that clearly from the graph below:



For more detailed Classification reports, please check the report below:

LR 1.0

Accuracy on training set: 0.99

Accuracy on test set: 0.99

precision recall f1-score support

0 0.99 1.00 1.00 310

1 1.00 1.00 1.00 53

2 1.00 0.80 0.89 10

accuracy 0.99 373

macro avg 1.00 0.93 0.96 373

weighted avg 0.99 0.99 0.99 373

Confusion Matrix

[[310 0 0]

[ 0 53 0]

[ 2 0 8]]

NB 0.9195171026156942

Accuracy on training set: 0.90

Accuracy on test set: 0.89

precision recall f1-score support

0 0.99 0.87 0.93 310

1 0.61 0.96 0.75 53

2 0.59 1.00 0.74 10

accuracy 0.89 373

macro avg 0.73 0.95 0.81 373

weighted avg 0.93 0.89 0.90 373

Confusion Matrix

[[271 32 7]

[ 2 51 0]

[ 0 0 10]]

RF 1.0

Accuracy on training set: 1.00

Accuracy on test set: 1.00

precision recall f1-score support

0 1.00 1.00 1.00 310

1 1.00 1.00 1.00 53

2 1.00 1.00 1.00 10

accuracy 1.00 373

macro avg 1.00 1.00 1.00 373

weighted avg 1.00 1.00 1.00 373

Confusion Matrix

[[310 0 0]

[ 0 53 0]

[ 0 0 10]]

SVC 0.9899396378269618

Accuracy on training set: 0.99

Accuracy on test set: 0.98

precision recall f1-score support

0 0.98 1.00 0.99 310

1 1.00 1.00 1.00 53

2 1.00 0.40 0.57 10

accuracy 0.98 373

macro avg 0.99 0.80 0.85 373

weighted avg 0.98 0.98 0.98 373

Confusion Matrix

[[310 0 0]

[ 0 53 0]

[ 6 0 4]]

Dtree 1.0

Accuracy on training set: 1.00

Accuracy on test set: 1.00

precision recall f1-score support

0 1.00 1.00 1.00 310

1 1.00 1.00 1.00 53

2 1.00 1.00 1.00 10

accuracy 1.00 373

macro avg 1.00 1.00 1.00 373

weighted avg 1.00 1.00 1.00 373

Confusion Matrix

[[310 0 0]

[ 0 53 0]

[ 0 0 10]]

XGB 1.0

Accuracy on training set: 1.00

Accuracy on test set: 1.00

precision recall f1-score support

0 1.00 1.00 1.00 310

1 1.00 0.98 0.99 53

2 1.00 1.00 1.00 10

accuracy 1.00 373

macro avg 1.00 0.99 1.00 373

weighted avg 1.00 1.00 1.00 373

Confusion Matrix

[[310 0 0]

[ 1 52 0]

[ 0 0 10]]

KNN 0.9859437751004017

Accuracy on training set: 0.99

Accuracy on test set: 0.98

precision recall f1-score support

0 0.98 1.00 0.99 310

1 1.00 0.96 0.98 53

2 1.00 0.60 0.75 10

accuracy 0.98 373

macro avg 0.99 0.85 0.91 373

weighted avg 0.98 0.98 0.98 373

Confusion Matrix

[[310 0 0]

[ 2 51 0]

[ 4 0 6]]

Now we have a machine learning model which predicts the future LTV segments of our customers. We can easily adapt our actions based on that.

## Saving Machine Learning Model : Serialization & Deserialization

In computer science, in the context of data storage, serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and reconstructed later in the same or another computer environment. In Python, pickling is a standard way to store objects and retrieve them as their original state.

We can save all our machine learning models in a pickle to use/call for production. For example our machine learning models, were pickled for future use below:

[('LR', LogisticRegression(C=1.0, 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)),

('NB', GaussianNB(priors=None, var\_smoothing=1e-09)),

('RF', RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=None, max\_features='auto',

max\_leaf\_nodes=None, max\_samples=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=100,

n\_jobs=None, oob\_score=False, random\_state=None,

verbose=0, warm\_start=False)),

('SVC', SVC(C=1.0, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)),

('Dtree', DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=None, max\_features=None, 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, presort='deprecated',

random\_state=None, splitter='best')),

('XGB', XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bynode=1, colsample\_bytree=1, gamma=0,

learning\_rate=0.1, max\_delta\_step=0, max\_depth=3,

min\_child\_weight=1, missing=None, n\_estimators=100, n\_jobs=1,

nthread=None, objective='multi:softprob', random\_state=0,

reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=None,

silent=None, subsample=1, verbosity=1)),

('KNN', KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform'))]

# Exploratory Data Analysis EDA Concluding Remarks

* The dataset does not feature any missing or erroneous data values, and all features are of the correct data type apart from the “Invoicedate”, which was restored to date.
* The dataset is imbalanced with the majority of observations describing sales in GB.
* About 91% of sales is in GB
* Around 83% of the products’ prices are less than 5 unit price
* We saw 36.5% growth previous month
* Monthly Active Customer number dropped to 820 from 1,368 (-40.1%)
* Order Count is also declined in Dec 2010 (455k to 251, -44.8%)
* The monthly order average increased for Dec (15.8 to 17.8)
* Existing customers are showing a positive trend and tell us that our customer base is growing but new customers have a slight negative trend
* Monthly Retention Rate significantly jumped from Nov 2010 to Dec 2010
* First month retention rate in 2010, was 13.7% of our customers retain with us.
* Silhouette and the Elbow Methods can find four clusters
* Good Value Customers %26.65, Potential Customers %47.07 and Customers at risk %26.28
* we can make recommendations of **Other Items You Might Like** to customers within this segment.
* Positive correlation between RFM score and 6m LTV. High RFM score means high LTV.
* LTV\_Cluster 2 is the best with average 10.5k LTV whereas 0 is the worst cluster with 445.
* We see that 3 months Revenue, Frequency and Monetary RFM scores will be helpful for our machine learning models
* 7 machine learning models are used for this case study: LR, NB, RF, SVC, Dtree, XGB and KNN
* The best performing models in this case are SVC, LR, XGB and RF respectively.
* we can see that LR has the lowest Bias and SVC has the lowest variance.
* Applying hyper-parameter improved the models slightly, by about 1 to 5%
* Normalizing the data improved the models significantly, reaching 100% in some models.
* We can save all our machine learning models in a pickle to use/call it for production.

Possible future actions:

* Adding more features could potentially increase the number of use cases that we do. For example if we have churn field we can build [Churn Prediction](https://towardsdatascience.com/churn-prediction-3a4a36c2129a) model.
* Adding more data would improve the machine learning models