**The problem**

Santander want to build a recommendation engine that will predict which products their existing customers will use in the next month based on their past behaviour and that of similar customers.

The purpose is so that Santander can better meet the individual needs of all customers and ensure their satisfaction.

**Recommendation Models**

Once the planning and initial set-up was in place, the focus was on researching different types of machine learning approaches, recommendation models and algorithms, and choosing the best.

* Collaborative filtering algorithms are more suitable than Content based algorithms, as the number of products were not exhaustive enough to provide a concrete suggestion.
* Supervised learning was the obvious choice as the response (correct) was given.

**The algorithm**

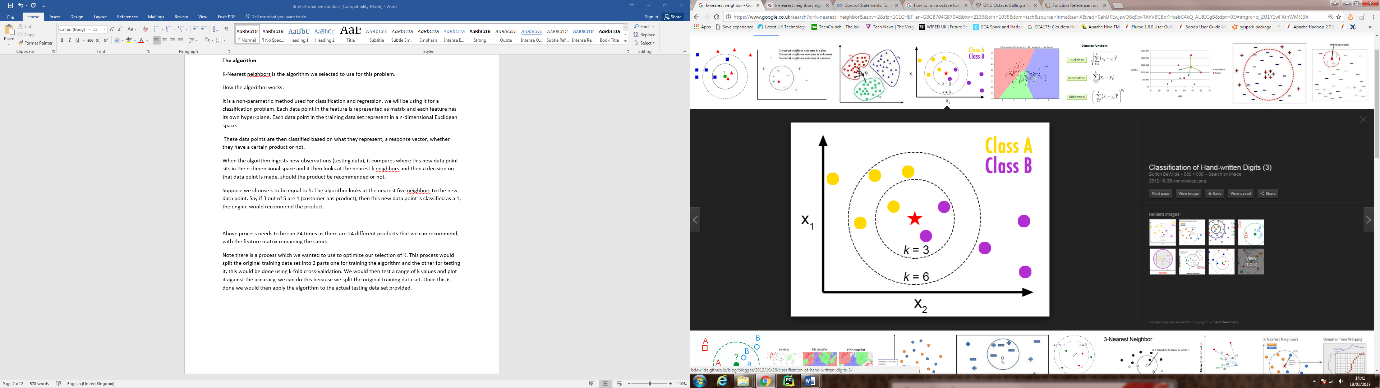
K-Nearest neighbors is the algorithm we selected to use for this problem.

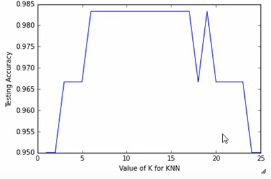
How the algorithm works:

It is a non-parametric method used for classification and regression, we will be using it for a classification problem. Each data point in the feature is represented in a matrix and each feature has its own hyper-plane.

These data points are then classified based on what they represent, a response vector, whether a certain product should be recommended or not.

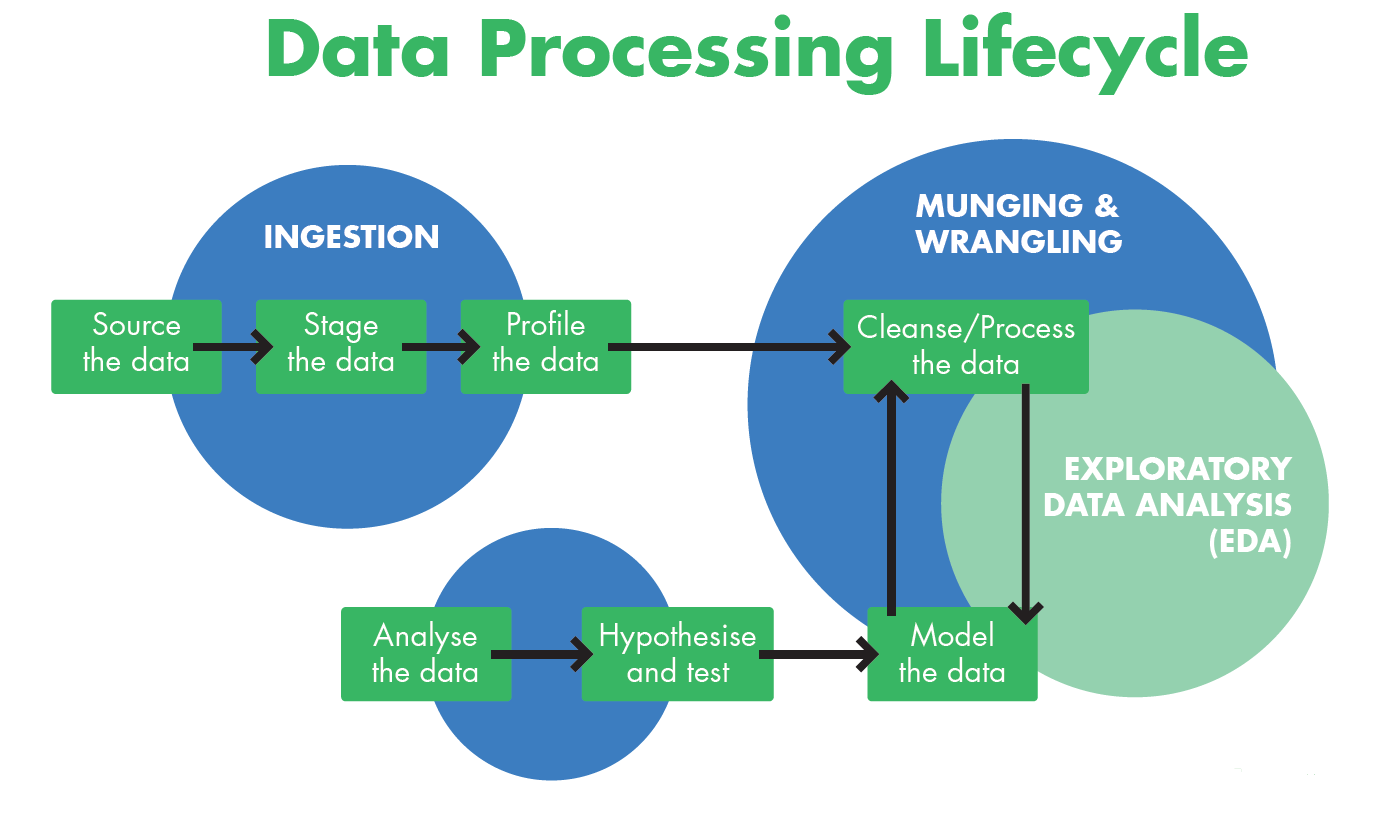
When the algorithm ingests new observations (testing data), it compares where this new data point sits in the n-dimensional space and it then looks at the nearest k neighbors and then a decision on that data point is made, should the product be recommended or not.

Suppose we have two features x1, x2 and the yellow and purple coloured dots represent the different classes (response vector). If a new data point is inserted, namely the red star. The value of k determines how many neighbors we will consider in order to classify this new data point. If k is 3 then we if clear that the new data point should be classified as purple (the majority of the 3 neighbors is purple), a similar process is used to make a decision when k is 6.

 Above process needs to be run 24 times as there are 24 different products that we can recommend, with the feature matrix remaining the same.

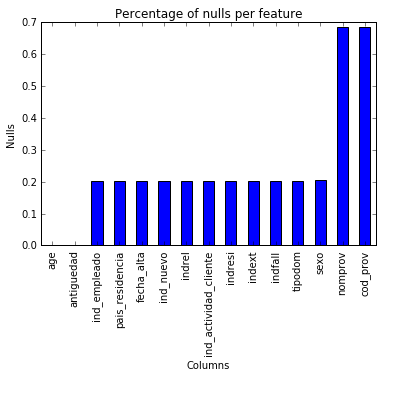
Note there is a process which we wanted to use to optimize our selection of K. This process would split the original training data set into 2 parts one for training the algorithm and the other for testing it, this would be done using k-fold cross-validation. We would then test a range of k values and plot it against the accuracy (figure on the right). Once this is done we would then apply the algorithm to the actual testing data set provided.

**Data Preparation**



Choosing the right tools:

* Apache Spark - Initial choice
  + The volume of data volume did not require a distributed environment.
* IPython
  + Easily deal with large volume of data.
  + One tool does it all - Data engineering, data visualisation and data analytics.
  + Easy to debug.
* Python libraries
  + Pandas – Data manipulation.
  + Scikit-Learn – Machine Learning library, train and test models in a speedy manner

 Data Munging and Wrangling:

* Appropriate feature selection – we used the columns which had the least amount of nulls
* Reduce duplications – use the latest activity of the customer
* Impute nulls and outliers
* KNN works at an optimum level when the feature matrix consists of numbers between 0-1

**Future improvements**

We didn’t get the opportunity to apply the algorithm, so that is one things we would definitely like to implement.

There was one interesting way we found of how to select the data for our feature set, if we had more time this is something we want to implement. We believe this would make our recommendation engine more accurate and versatile.

Our current model is based on entering the response vector which is n by 1 vector. The idea and hope was to then get the responses of all the 24 results and combine them together and give a set of products that should be recommended to the customer. To further this model, the use of Support Vector Machine, would have allowed us to have multiple classification on all the products, so instead of having a n by 1 vector, the response would have been a n by m vector, where m in the number of products (24).

Further to the above, the idea of weighting the products is something we wanted to look into. The idea is that when a data point is represented in the n-dimensional hyper plane, the proximity of how close the data point is within a cluster will be given a weight, the closer it is to the centre the higher the weighting.

Another idea we had was to understand the historical relationship between purchasing patterns and the customers, we had this data but initially we ignored it, for simplicity and only used the most-up-date data. In future we would like to look into these data to avoid recommending product that the customer cancelled in the past.