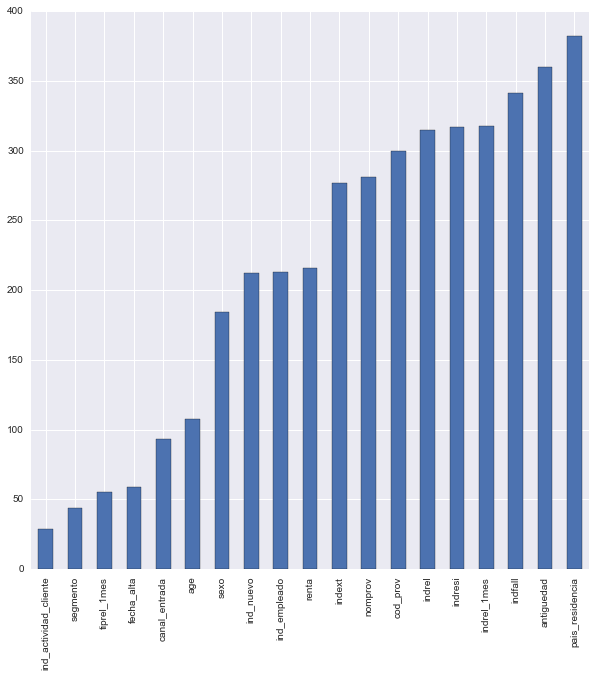
Project write-up

**Decision tree Engine**

After initial planning and theory research behind the workings of decision tree engines, it was deemed necessary to stage the data and explore it’s content, with the aim of finding the most important and relevant features to implement into our recommendation engine.

This involved data engineering and cleansing, specifically imputing along numerous columns in the data set, followed by the use of sklean.selectKbest library to determine the most influential data features on what the customer is likely to buy. This library involves looking at the target variables present, and the correlation between those and the features provided, to give a score of the most influential ones.

The results of this analysis are shown in figure 1.

**Fig (1): a graph representing the importance of the features on the total number of products bought by customers, in a negative scale, meaning the lowest valued features are the most influential ones.**

Once the most important features were found, work underwent on producing a simple working decision tree model, with the plan of using the predicted probabilities for products as a measure for our recommender engine.

While the engine managed to produce relatively accurate prediction based on the training set provided (~85% accuracy), returning the probabilities of predictions and formatting them into an accepted format deemed to be too inefficient for the task at hand, as more work would go into this formatting than actually producing accurate results. Once the code was completed (alterations meant automatic cleansing and engineering of both training and testing data sets), work shifted towards a more suitable engine.

Engine V2 is based around a publicly available script provided on Kaggle by author SRK.

Initial work involved importing the script and relevant programs needed to run the recommender engine (XGBoost is a machine-learning algorithm, which is notoriously difficult to install on windows).

The engine in its default configuration returns a score of 0.026 on Kaggle, which is good enough for a top 1000 place finish on the leader boards.

Work from then on focussed on modifying the engine; include tuning parameters and adding new useful features, like Lag products, which track how the user product ownership changed over a number of months.

One of the important things noticed during the project is the fact that the task is to predict the behaviour of product purchase in June, a month special in Spain, as it is when taxes are due.

In order to address this operation, heavier weightings were imposed onto the June figures from the previous year in order to capture seasonality, rather than trend, which seemed to have a positive effect on results.

Final Model had a score of 0.030, achieving a 170th place in the leader boards, a top 10% finish.