

**TCDS - Final Project**

**Forecasting purchases of items**

**per day and store**

**Data Science**

**FAVORITA** **Project Protocol**

**Authors:**

**Ariel Shafir & Tzvika Senderov**

**MAR 2019**

# **Introduction**

Grocery stores have always faced the challenge of forecasting procurement and sales of consumable products. High forecasting will result in excess inventory being thrown into the trash. If the prediction is low, it will result in a loss of income mainly in products that are sold a lot (Popular).

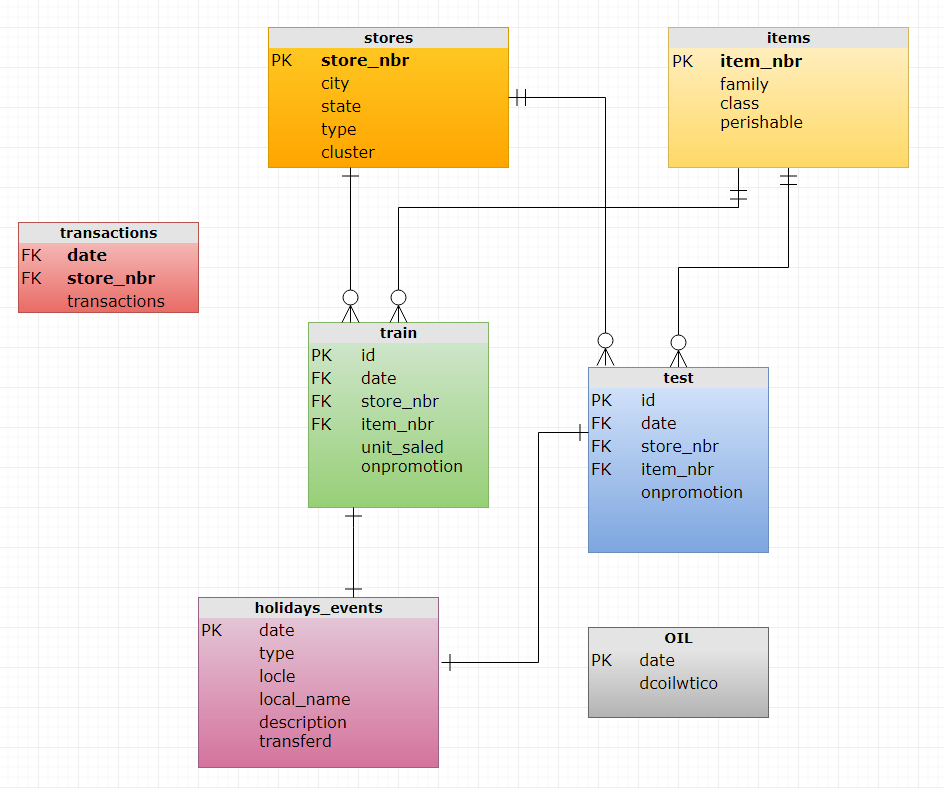
[Corporación Favorita](http://www.corporacionfavorita.com/) is a large Ecuadorian-based grocery retailer.

They operate hundreds of supermarkets, with over 200,000 different products on their shelves tackles this challenge on a daily basis.

The difficulty is greater when it comes to new products and seasonal products, which should be prepared accordingly according to the period of the year, which sometimes also takes into consideration unique storage conditions,

We want to see how using ML machine model will predict the quantity of products sold at a daily level in each store.

The data source originally from a Kaggle challenge. The Dataset has been provided by **Favorita corporation**.



# **Methodology (project design)**

**Data**

The data source was taken from Kaggle Competition – *"Corporación Favorita Grocery Sales Forecasting,* the original challenge was: *can you accurately predict sales for a large grocery chain?"*

These are the csv files that was provided:

### train.csv

Training data, which includes the target unit\_sales by date, store\_nbr, and item\_nbr and a unique id to label rows onpromotion column tells whether that item\_nbr was on promotion for a specified date and store\_nbr   Negative values of unit\_sales represent returns of that particular item.

**stores.csv**

Store metadata, including city, state, type, and cluster where   cluster is a grouping of similar stores.

### items.csv

Item metadata, including family, class, and perishable -: Items marked as perishable have a score weight of 1.25.

### oil.csv

Daily oil price. Includes values during both the train *and* test data timeframe. (Ecuador is an oil-dependent country and it's economical health is highly vulnerable to shocks in oil prices.)

### holidays\_events.csv

Holidays, Bridge and Events

From Google we took information about the population <https://www.worldatlas.com/articles/biggest-cities-in-ecuador.html>

We build a function that calculate the payment day and payment day +1

We decided to split the data and avoid in advanced some date we had reason to assume it will influence the data. We avoided 4 months of activity from the earthquake and the beginning of the data which a field is missing until Apr 2014. Finally, we decided to divide the Train file to three-hole years- Train, Dev and Test as describe bellow and then we joined the date from the other tables as we will detail later, for the EDA faze.

At the beginning we assumed that it will be good and will provide a satisfying prediction it we will choose a 3 hole years which includes all parameters so we decided to act as follow.

The **Train** dataset includes activity of one year. Dates between 1/4/2014 and 31/3/2015

The **Dev** dataset includes activity of one year. Dates between 1/4/2015 until31/3 /2016

The **Test** dataset includes activity of one year. Dates between 1/7/2016 until 31/7/2017

Each dataset includes a hole year with all Holidays and events.

After we encountered many difficulties as a result of a very large amount of data, which made it difficult for us to run and perform various processes that include detection of outliers as well as running problems of models very slowly to the crash of the computer. And finally, also very low-quality forecasts, we decided to change direction and choose to make a forecast only on one store number 44.

Originally the complete dataset contains about 125,000,000 records. After we decided to choose a single year and choose 250 items, we stayed with about 3 million records per year. After many difficulties in running models and identifying outliers we reduced to 15 items and then we stayed with approximately 280,000 records each year.

After we decided to change direction and take only one year of data with 100 items sold best and to make the definition of the TRAIN sets, DEV TEST according to the methodology studied.

# **EDA- Exploratory data analysis**

**Clear outcome variable definition:**

How many units were sold of each item per day in **store number 44**.

**Some general EDA of all the data:**

The data contains information about

* Number of units were sold of each item per store and day
* Daily oil price
* Table with holydays and events in Ecuador (national, state and city) actual dates it took place.
* Information of items like category, did it was on promotion? and is it a perishable item?
* Information about the stores like type of store cluster of stores

The original dataset includes 4100 items and about 125,000,000 records. In our datasets we decided to forecast only 250 most sailing items. we partitioned the data by years, so each year data set has about 3 million rows.

We know that in April 2014 there was a major earthquake in Ecuador. This information is very important because it certainly affects consumer behavior at the time and in the subsequent period. Using this data or using unwise use may bias the forecast.

**Assumptions and decisions:**

* We assumed that there is a seasonableness in the behavior of customers and items being sold during the year that is why we decided to pick a hole year.
* The onpromotion field has about 16% pf missing in the hole dataset. the data before 01/04/2014 does not include a value in onpromotion so we preferred to choose a hole year that includes this value as our data set.
* In Ecuador there is a payday twice a month at the 15th and at end of month, we assume its influence is major, so we add parameters which indicates payday, payday -1, payday -2, payday +1 and payday +2 days.
* We add holydays indication by national, state and city to check for influence.
* oil prices influence very much at Ecuador economy, so we add the daily price as a parameter as well as prices 15 and 30 days earlier and the ratio of change between.

At the beginning we thought to make some analyzing in SQL Server but finally we just built the basic row before uploading to python.

We built a view that concatenate the data from all given files. we also calculated the pay day dates in SQL and the seasons.

To get values on the same range we took the percentage out total of sales

We see in this plot that most of categories are in the same range of value except two.

 we see that beverages category seems to have a lot of variance along the year.

# **Variable engineering**

After we uploaded the data to python, we made some format transformations like changing dates from object to date format and oil price to float.

All categorical parameters we transformed to category and all other to int64.

Later when we will make the dummies variable it will automatically recognize the categories and split it.

At the beginning we thought it will be easier to build the dummies in the SQL server, but it was a mistake.

The **get dummies** function can drop automatically on column so we don’t get in to dummy trap. If means that we need to avoid providing the model 100% of categories of the same field and leave on out which is the complimentary to 100% otherwise the model may be confused. If you want to drop the most common category by yourself, you should not set **drop\_first** as True and delete later.

We considered the type of holydays in our view, there is few types of holydays in Ecuador: local holydays, regional holydays and national holydays. These we built in advanced in SQL.

We calculated with SQL function if the date is a payday, one day after the payday one day before payday and two days before payday.

We calculate the oil price 15 and 30 days before each day and calculated how much (%) the oil price changed compare 15 and 30 days before and today.

Column "**unit\_sales**" is our Y column.

Original data of all stores includes information of 5 store types and cluster of stores, but as we decided to make our prediction on one store only (the biggest store number 44)

All the information about the city, population, state and stores become irrelevant.

# **Missing values**

# **determination of MCAR-MAR-MNAR and treatment**

Description function and df.isna().sum()

Showed about 30% nulls in oil price column.

Most of the null values are **MNAR** missing not at random because it is on weekends,

This variable is a special case (based on time series) and the common imputation is to add the value of the day before. we copied the last known value of the day before which is the same when there is no trade.

Few days had missing values which wasn’t on weekends. some of these days are probably holidays such as DEC 25 and JAN 01 and some maybe **MCAR**. Either way, because it is very few dates the treatment, we chose is the same, we took the last known value.

We extracted the date and oil price columns from our flat file table. Then imported table of oil prices at the month before our data begins and added missing dates manually.

We built a column contains the oil price day before (with shift function) and replaced with the nan values. We ran it three times to replace all nulls.

Then we add columns of the price 15 days ago and 30 days ago and calculated the ratio of change between each one and actual date price and finally merged it with our data.

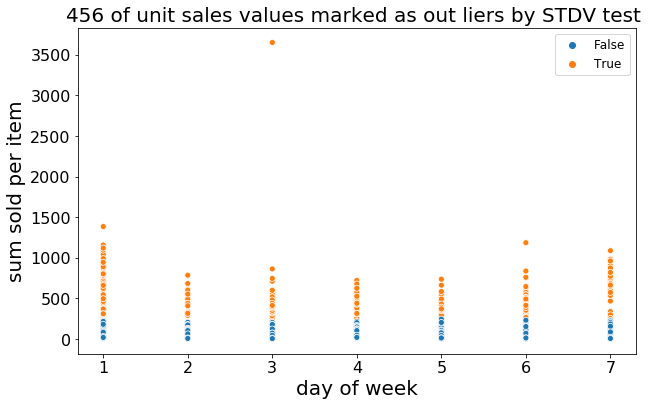
# **Outliers determination and treatment**

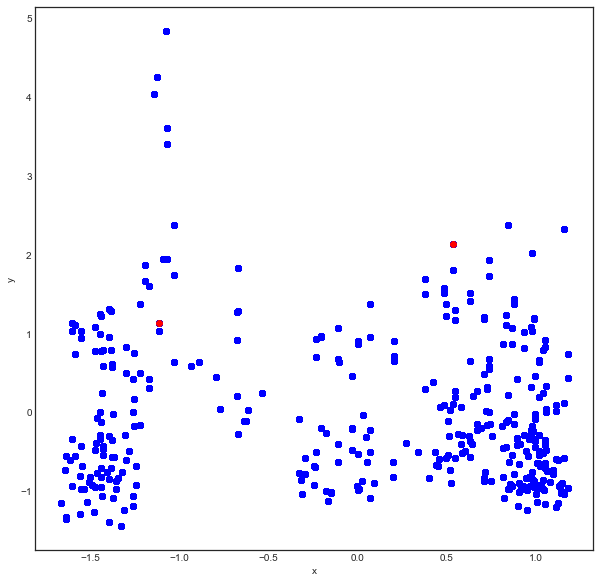
**Univariable outliers detection:**

We ran some functions to detects values that may be outliers.

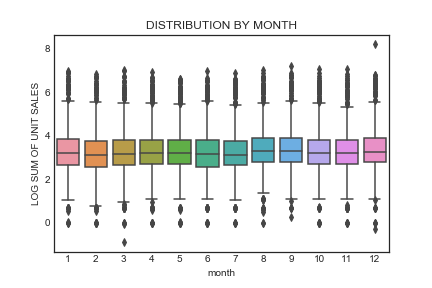
**zscore\_outliers** – determine outliers by using stdev value.

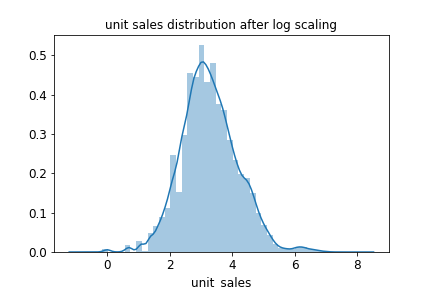
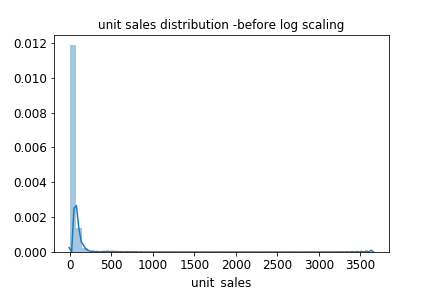
**iqr\_outliers** - Using median and Interquartile range (Boxplot) Which displays exceptions based on deviation of a q75- q25 range multiplied by coefficient.

Here we can see a plot with which implies of about 450 values being outliers.colors implies the distribution of the data and outliers’ values.

**Multivariable outliers detection:** we used *dbscan\_mvoutliers* function which indicated of only 2 outliers as can be shown at the following diagram

In this graph we can see the sales distribution over the month and can see there are suspicious values for outliers.

  
  
  
**after we checked and treated outliers we decided to normalize our Y with log**

**Our Y “unit sales” distribution histograms before scaling and after normalization**

# **Feature Selection**

In order to prepare data for feature selection we hade to take out our keys- **date and store number**

Change all category parameters we want to represent as one hot encoding, we changed to category so the function get dummies will know change it automatically .

get\_dummies(FF\_X, drop\_first = True )

creating column of our Y to a log Y for normalizing data.

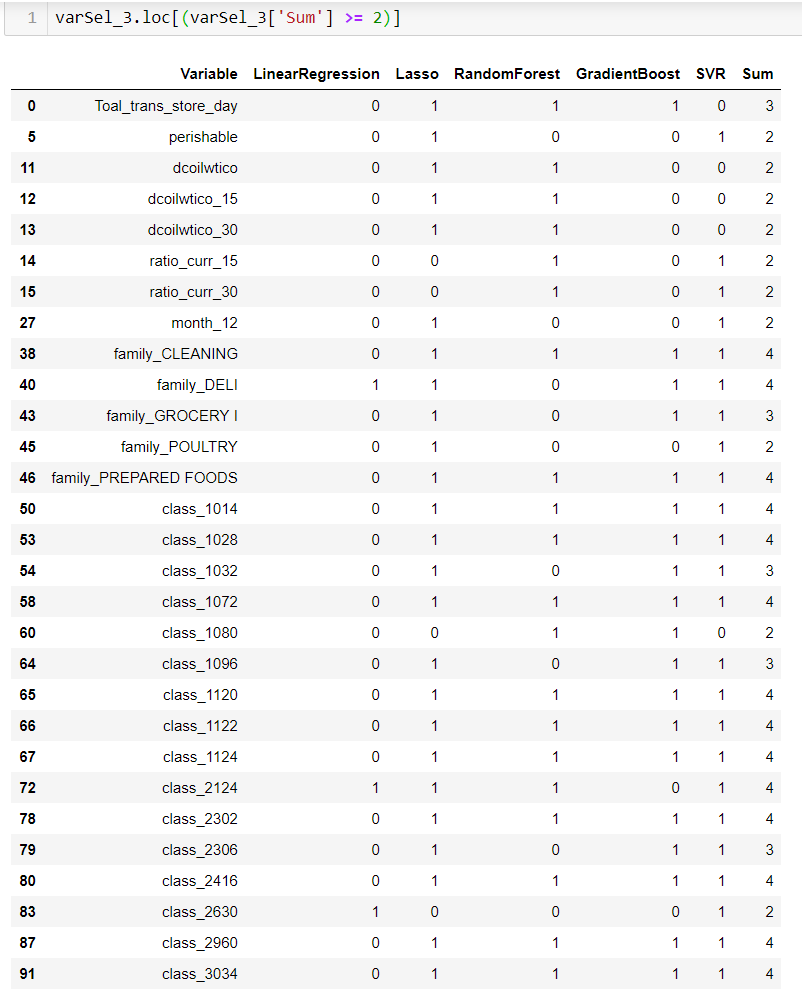
FF\_X['log\_unit\_sales']=np.log(FF\_X['unit\_sales'])

FF\_X=FF\_X.loc[:, FF\_X.columns != 'unit\_sales']

We created an array to store all of our models’ outcomes and later we will check for features that was suggested as important by more models.

We had 92 features

After models running, we decided to Thake every variable that had suggested by two models. We checked it with 4 ant with 3 and the best outcome of prediction we got with these 29 features.



# **Partition balance**

We partitioned our data to 3 partitions – TRAIN DEV AND TEST.

We will split the data as 10% for dev dataset, 10% for test dataset and the rest 80% as training.

We built a date frame that includes only our selected features and added the Y in order to prepare our data for partitioning.

In order to check the that partitions data has the same distribution, we ran tests until the distributions where the same and **P values of all variables** where up then 0.05.

We built an Automatic loop for finding the right seed that will provide us the **result of 0 variables with P-value smaller than 0.05 so we can repeat it later.**



# **Models**

* Our prediction is how many items will be sold in a day for each item

We have a Regression problem.

# **Deployment of your model**

In each model we used the normalized parameter option of the model.

models we used are:

* Linear Regression Model
* LASSO
* RIDGE
* SVM For Regression- (kernel = 'rbf')
* measurements were used to evaluate the prediction.

We used: R^2

**Selected model**

The best prediction of our Y, we got from **Ridge** model so we will try to treat with the Hyperparameters for fine tuning the model and to get a better prediction.

**Hyperparameters fine- tuning**

For finding the best hyperparameters we ran the ridge model with different values.

hyperparameter fine-tuning by cross-validation using grid-search

we found the best parameters for the model and check the final model with the train, dev and this time also the test. The results are results section.

# **Results**

* The final amount of data used (total, train, test, etc)

The hole dataset included 39,000 rows before splitting data.

Train – 29,520 rows

Dev – 3690 rows

Test – 3689 rows

* The multivariate test showed us only 2 outliers. We deleted the hole record.

The univariate tests where shown between 1 to 10 percent of outliers but we didn’t make any other test which could help us determine if is outliers or not nor treated them in any way.

* We had about 30% missing values of Oil price. as we described before, most of the missing are **MNAR** because it is on weekends, so we copied the last known value. Few more days had missing values which wasn’t on weekends. some of these days are holidays such as DEC 25 and JAN 01 and some maybe **MCAR**. Either way, because it is very few dates the treatment, we chose is the same, we took the last known value.
* The data we picked from the hole dataset is of a hole year between 1/04/2014 and 31/3/2015.
* We built new features of oil price. We used previous values and the ration of difference between two weeks ago and current value.
* Model selected: **L2 regularization – Ridge** – it provides the best outcome
* Hyperparameter fine-tuning- we used deferent **alpha values** and different **solver**

We changed max iterations until we got the best combination.

Best result was with alpha = 1e-15 and solver **svd**

**Results from models:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MODEL** | **score (X\_train, y\_train)** | **score (X\_dev, y\_dev)** | **r2\_score(y\_dev,y\_hat)** | score(X\_test, y\_test) | r2\_score(y\_test,y1\_hat) |
| Linear Regression | 0.4208 | 0.425940 | 0.425 |  |  |
| LASSO | 0.339768 | 0.32384 | 0.3238 |  |  |
| RIDG | 0.44022 | 0.42611 | 0.426 | 0.4523 | 0.45234 |
| SVM | 0.1901558 | 0.097531 | 0.0975 |  |  |
|  |  |  |  |  |  |

# **Conclusion**

We have learned a lot along this project.

The work process in general, and feature engineering, design and running models, and more. This is just the beginning, and we intend to deepen and specialize in the field. Making the forecast was very challenging. The forecast we succeeded in producing here was below what we expected, and we believe that the issue of extreme values should be examined and dealt with as well as additional parameters that explain our Y.

maybe we could have tried another approach and group the unit sale per day to 6 groups like 1 to 5 = 1, 5 to 10 = 2… above 40 = 6.

It may also be that the amount of information we used was insufficient. We believe that the more information we have, the better the models will be able to learn.

**TABLE ONE – before on hot encoding**table one for only one store (store\_nbr = 44) and top 100 sale items

|  |  |  |  |
| --- | --- | --- | --- |
| **Variables** | **Categories** | **n** | **Population** |
| Individuals | n | 1 | **36,899** |
| month |  |  |  |
| day\_of\_week |  |  |  |
| season | summer | 0 | 9,100.00 ( 24.70%) |
| season | spring | 1 | 9,199.00 ( 24.90%) |
| season | autumn | 2 | 9,600.00 ( 26.00%) |
| season | winter | 3 | 9,000.00 ( 24.40%) |
| item\_nbr |  |  |  |
| Toal\_trans\_store\_day | Mean (SD) | 1 | 4,493.30 ( 750.80) |
| Class (of item) | 34 |  |  |
| LocalHoliday | 1 | 1 | 200.00 ( 0.50%) |
| RegionalHoliday | 0 | 0 | 36,899.00 ( 100.00%) |
| NationalHoliday | 1 | 1 | 3,700.00 ( 10.00%) |
| unit\_sales | Mean (SD) | 1 | 42.30 ( 67.80) |
| unit\_sales | Median (IQR) | 2 | 25.00 ( 15.00- 46.00) |
| onpromotion | 1 | 1 | 1,227.00 ( 3.30%) |
| dcoilwtico - OIL PRICE | Mean (SD) | 1 | 81.20 ( 22.70) |
| dcoilwtico - OIL PRICE | Median (IQR) | 2 | 91.50 ( 54.20- 101.90) |
| dcoilwtico- OIL PRICE | Missing (%) | 3 | 11,100.00 ( 30.10%) |
| perishable | 1 | 1 | 12,546.00 ( 34.00%) |
| is\_pay\_day\_m2 | 1 | 1 | 2,700.00 ( 7.30%) |
| is\_pay\_day\_m1 | 1 | 1 | 2,400.00 ( 6.50%) |
| is\_pay\_day | 1 | 1 | 2,400.00 ( 6.50%) |
| is\_pay\_day\_p1 | 1 | 1 | 2,400.00 ( 6.50%) |
| is\_pay\_day\_p2 | 1 | 1 | 2,300.00 ( 6.20%) |
|  |  |  |  |
|  |  |  |  |
| **city** | **Quito** | **0** | **36,899.00 ( 100.00%)** |
| **city\_population** | **1,399,814** | **0** | **36,899.00 ( 100.00%)** |
| **state** | **Pichincha** | **0** | **36,899.00 ( 100.00%)** |
| **type** | **A** | **0** | **36,899.00 ( 100.00%)** |
| **cluster** | **5** | **0** | **36,899.00 ( 100.00%)** |

**TABLE 1 Partition balance check**



**Data Retrieval Protocol**

**strikethrough line values means that finally wasn’t use**

