

**Corporación Favorita Grocery Sales**

**Forecasting**

**DS Project Protocol**

*Author: Tal Beinart*

# Introduction

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

They operate hundreds of supermarkets, with over 200,000 different products on their shelves.

Brick-and-mortar grocery stores are always in a delicate dance with purchasing and sales forecasting. Predict a little over, and grocers are stuck with overstocked, perishable goods. Guess a little under, and popular items quickly sell out, leaving money on the table and customers fuming.

The problem becomes more complex as retailers add new locations with unique needs, new products, ever transitioning seasonal tastes, and unpredictable product marketing

The purpose of this project is to build a model that more accurately forecasts monthly product sales within a year of a particular food family (seafood).

# Methodology (Project design)

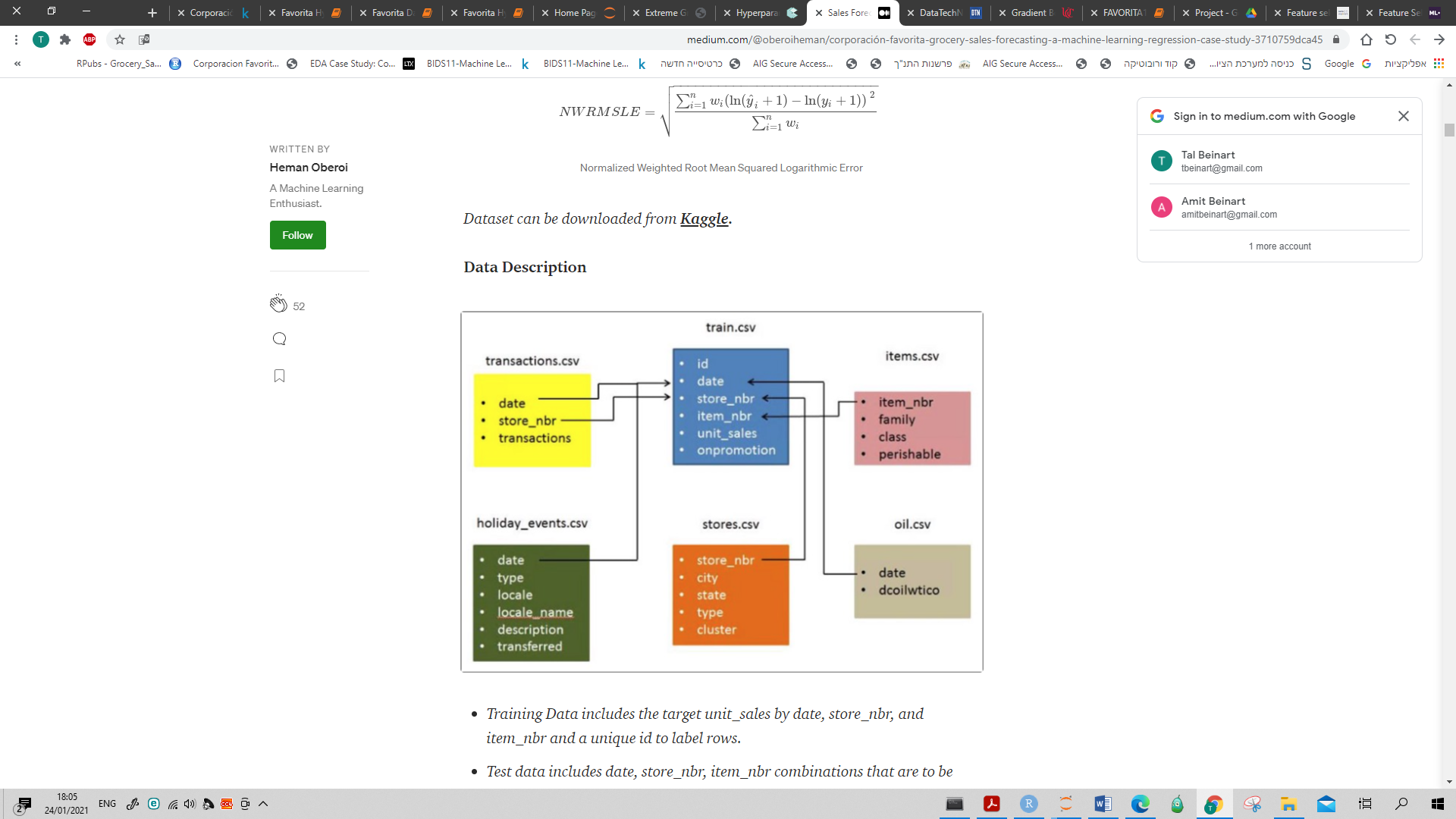
## Data

***Objective*** *-* ***Predicting the Seafood monthly unit sales for the next year****.*

The data is available in Kaggle platform. It contains one main data file (train) that includes dates, store and item information, whether that item was being promoted, as well as the unit sales.

Additional files include supplementary information that may be useful in building the model.

Data description:



* Training Data includes the target variable unit\_sales by date, store\_nbr, and item\_nbr and a unique id to label rows.
* Store metadata includes city, state, type, and cluster (cluster is a grouping of similar stores)
* Item metadata includes family, class, and perishable.
* Transactions Data includes the count of sales transactions for each date, store\_nbr combination. Only included for the training data time frame.
* Oil Data includes daily oil price. Includes values during both the train and test data time frame.
* Events Data includes Holidays and Events, with metadata.

Additional information

A magnitude 7.8 earthquake struck Ecuador on April 16, 2016. People rallied in relief efforts donating water and other first need products which greatly affected supermarket sales for several weeks after the earthquake.

The given sales data is from Jan 2013 to 2017 August 15th.

The predicting model is not TS based, and since the distribution of the Seafood category is pretty stable in terms of # sales over the years we'll use the hole data.

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Seafood #Sales | Seafood Distinct Items Count | Seafood Items |
| 2013 | 373,317 | 8 | 44,830 |
| 2014 | 421,129 | 8 | 51,972 |
| 2015 | 464,062 | 8 | 59,143 |
| 2016 | 447,132 | 8 | 62,678 |
| 2017 | 265,932 | 8 | 39,272 |

# Exploratory Data Analysis

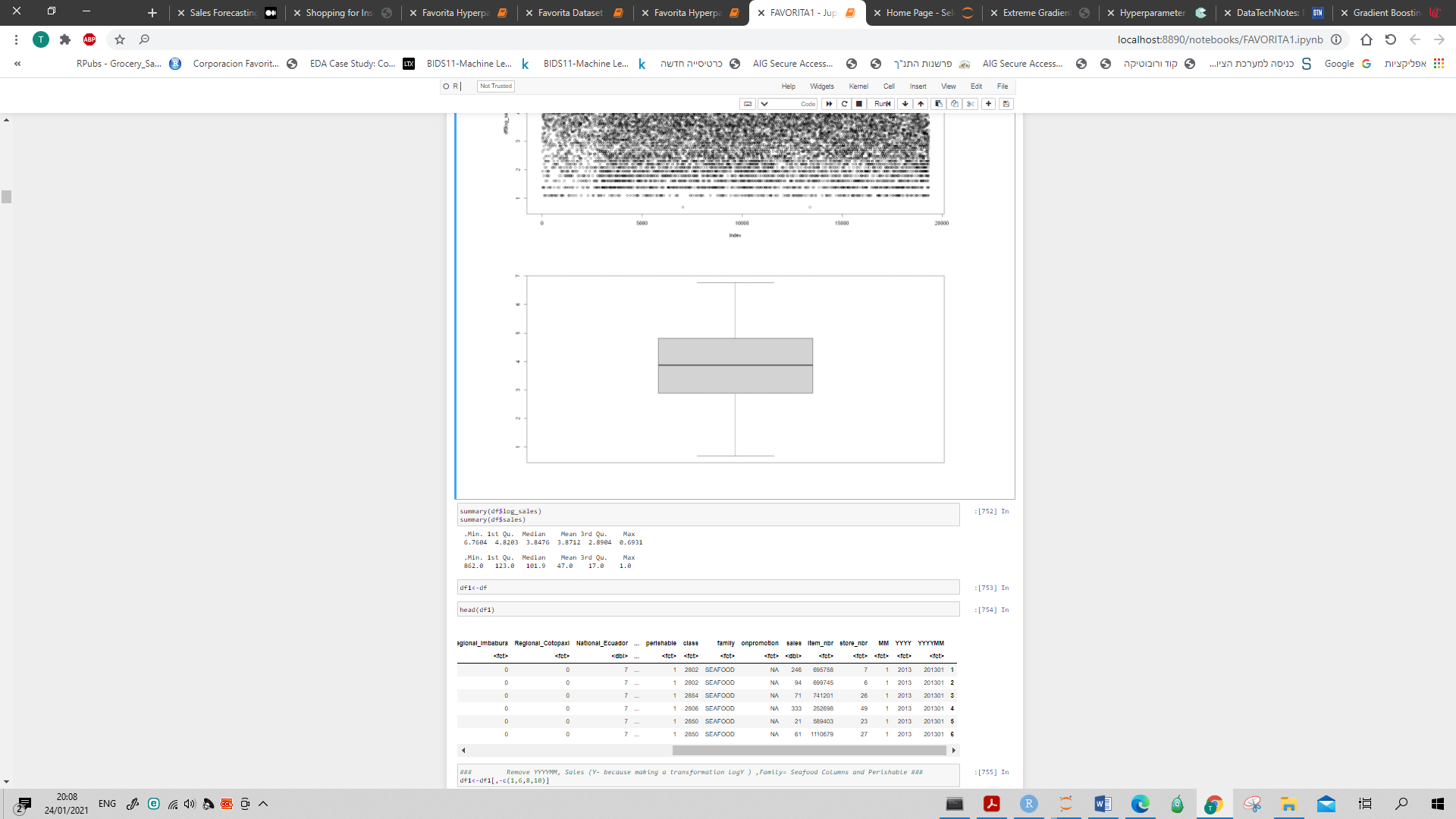
Steps:

1. Look at each table by itself.
2. Creating a flat file using SQL

* Oil table - create YYYYMM variable and add monthly average / max / min oil price.
* Transaction table - group by YYYYMM and Store\_nbr.
* Holidays table – create YYYYMM variable and for each location count its holidays
* Add to the train file YYYYMM variable.
* Create a flat file by left join the sub tables with the train file, selecting the Seafood family.

1. EDA by R studio

* **Data dimension - 42 X 19,336**
* Dealing with NA's and variables type.
* Data summary attached in the appendix.
* Check missing (8,843).
* Plot the label (Sales) and make a transformation in order to normalize its distribution (appendix).



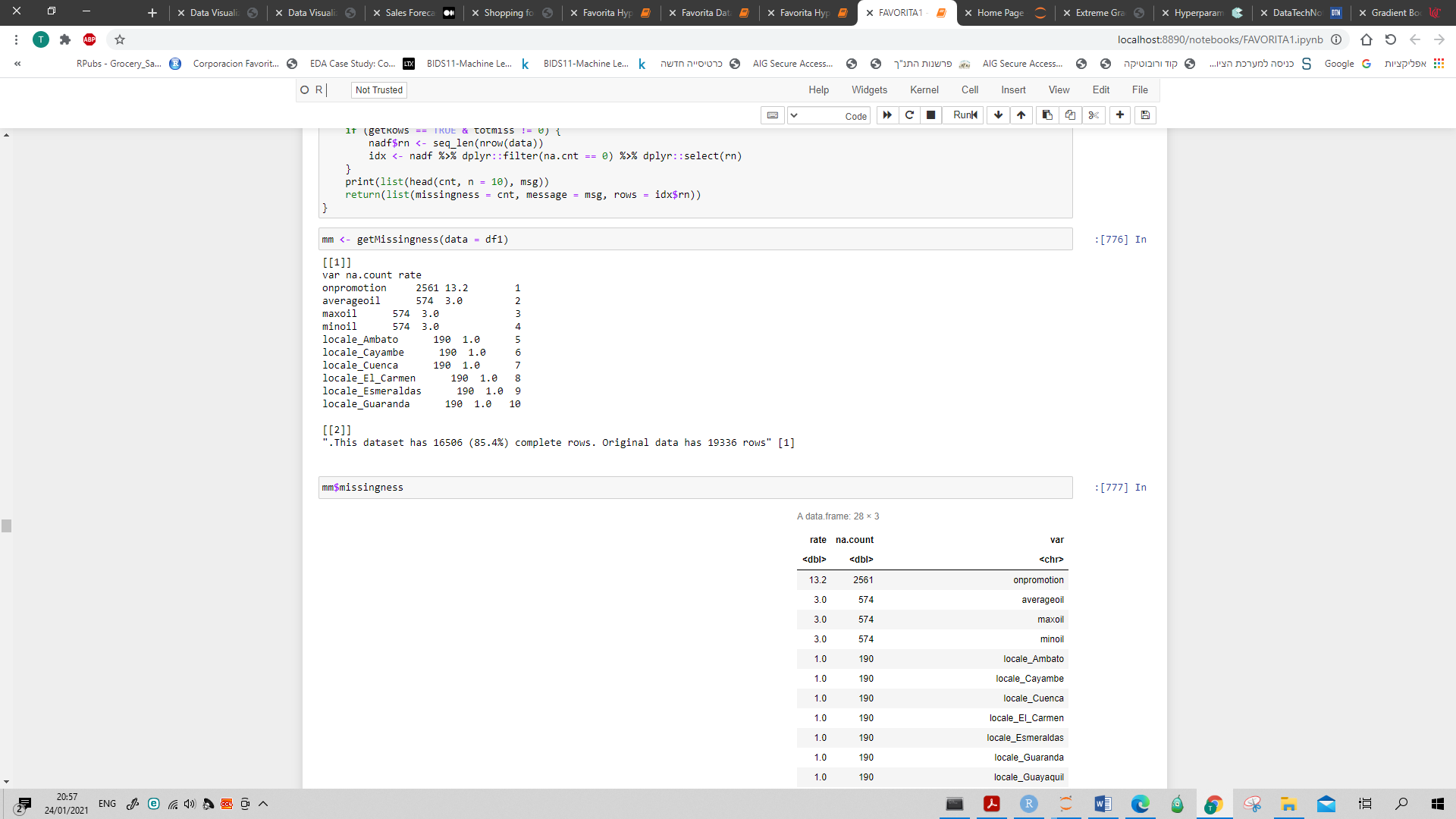
* Looking at the data using Table1 and Explore data functions in order to see variables distribution, missing and outliers (appendix):

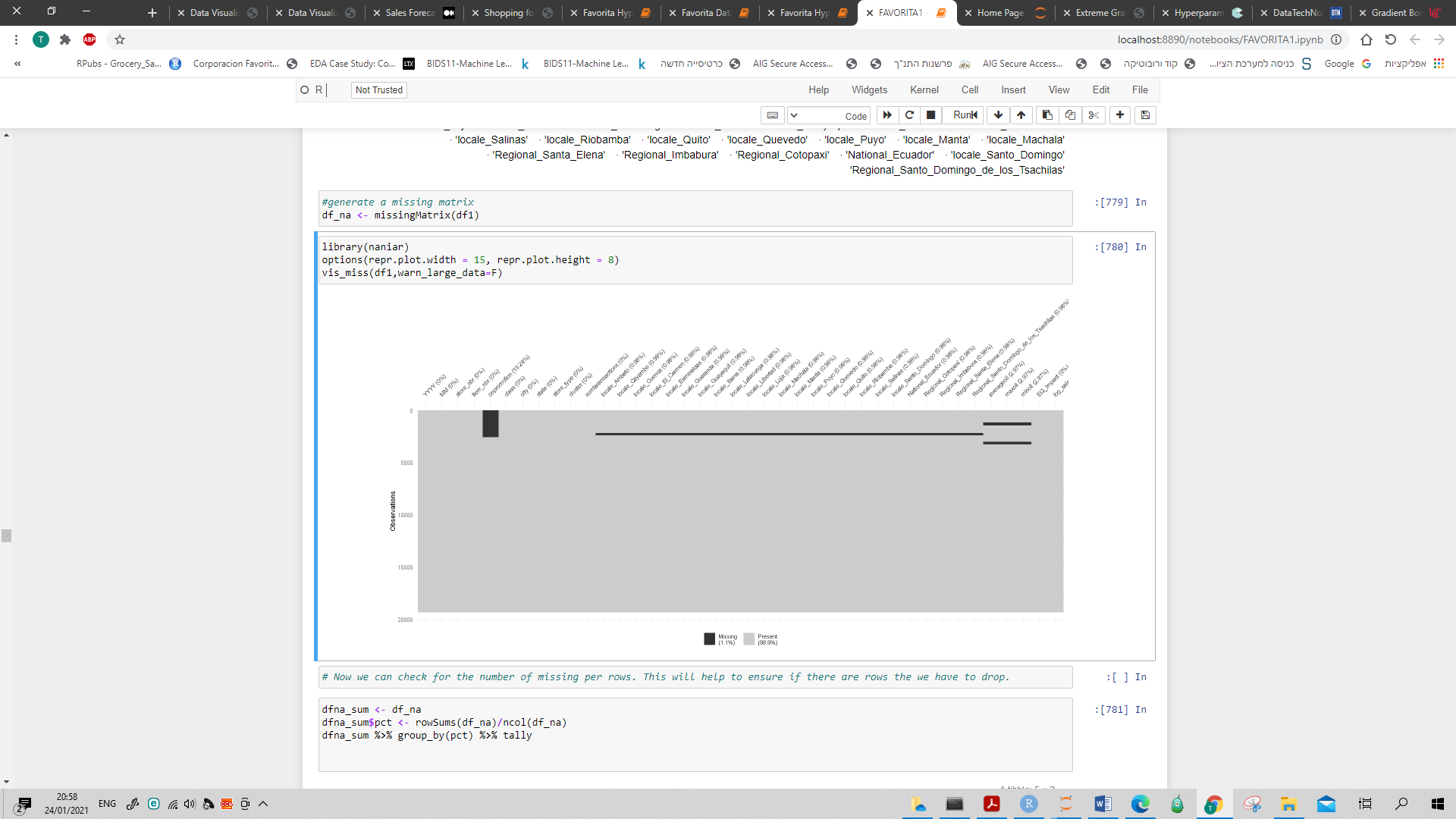
Highlights:

* No outliers
* Missing in Onpromotion variable (2,561 (13.2%)). Since it is a categorical variable we'll add a level : {0=False , 1=True , 2 =Null}.
* In variables that reflects the local holidays there 190 Missing (~1%).

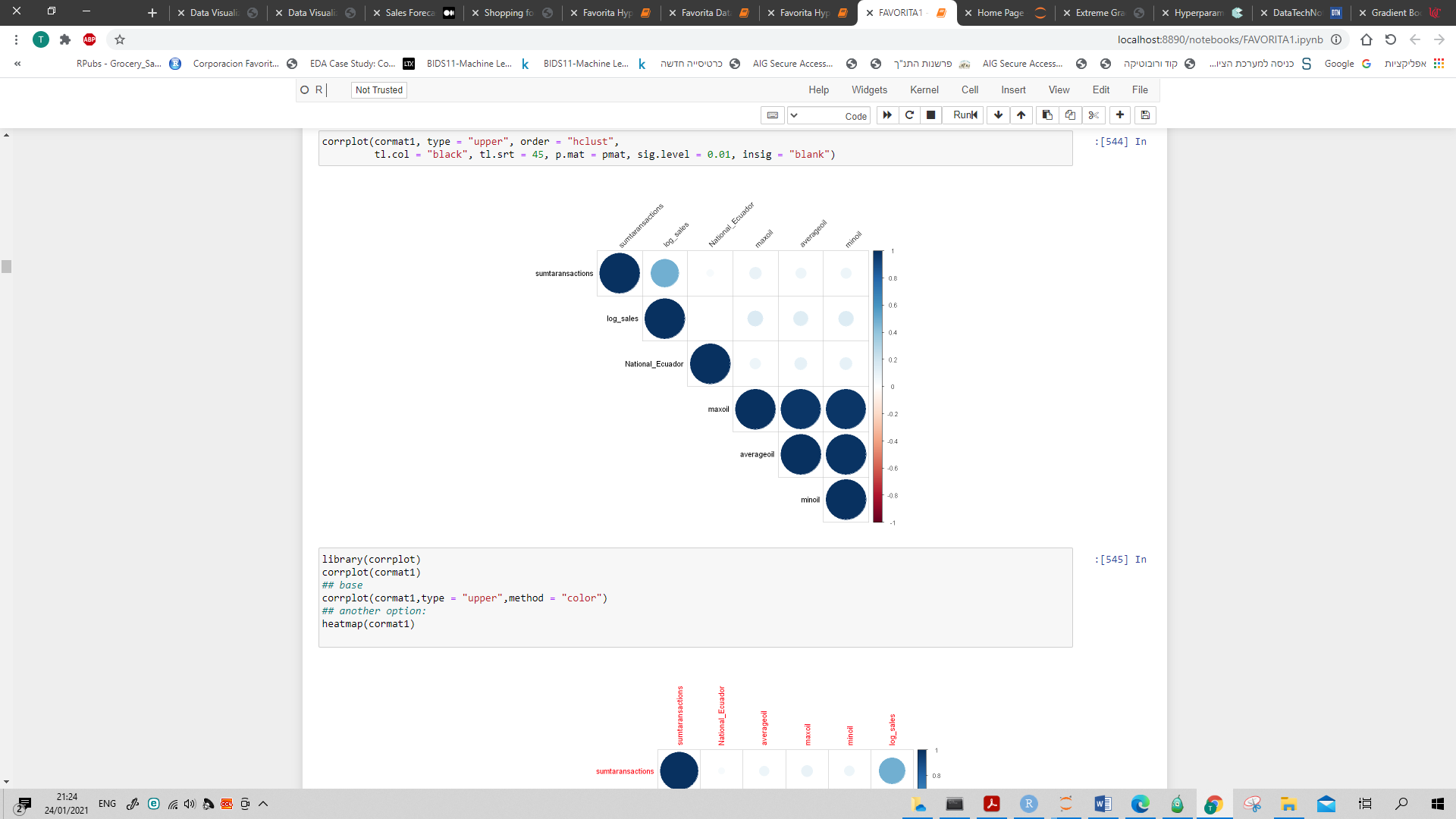
These 190 rows have a very high percent of missingness (over 60%), thus we will drop it.

* 3% missings in the oil variables. We checked that no variable can explain the presence of missing values on any of the missing variables, thus we can assume that the missing mechanism is at least MAR. Since those missings stands at 3% only it will be removed.





* Looking at the label variable vs. the factor variables.
* Different dist. Between stores, item\_nbr, onpromotion, class, cities, states, store\_type and cluster.
* Most local holidays variable factor are not differ regarding log\_sales except locale\_Esmeraldes and locale\_Manta.
* Find Correlation between numeric variables (spearman) and factor variables (Cramer.V).



* Feature Engineering / Impact Coding / Data Extraction / Data Transformation
* For explanatory factor variables with many levels we transformed and replace the origin variable by its mix (percent). For example, we replaced the 54 Store\_nbr by the percent of each store. Same method was used for City (22 levels), State (16) and Cluster (17) variables.
* One-hot encoding was used in order to treat Item\_nbr, Onpromotion, Class and Store\_type variables.
* For each origin categorical variable that we used one-hot-encoding on, we'll reduce one dummy that have the less frequent "1".
* Add indicator variable EQ\_impact that reflects the earthquake in Ecuador on April 2016.
* Check missing again.

**The data dimension - 62 x 18,572**

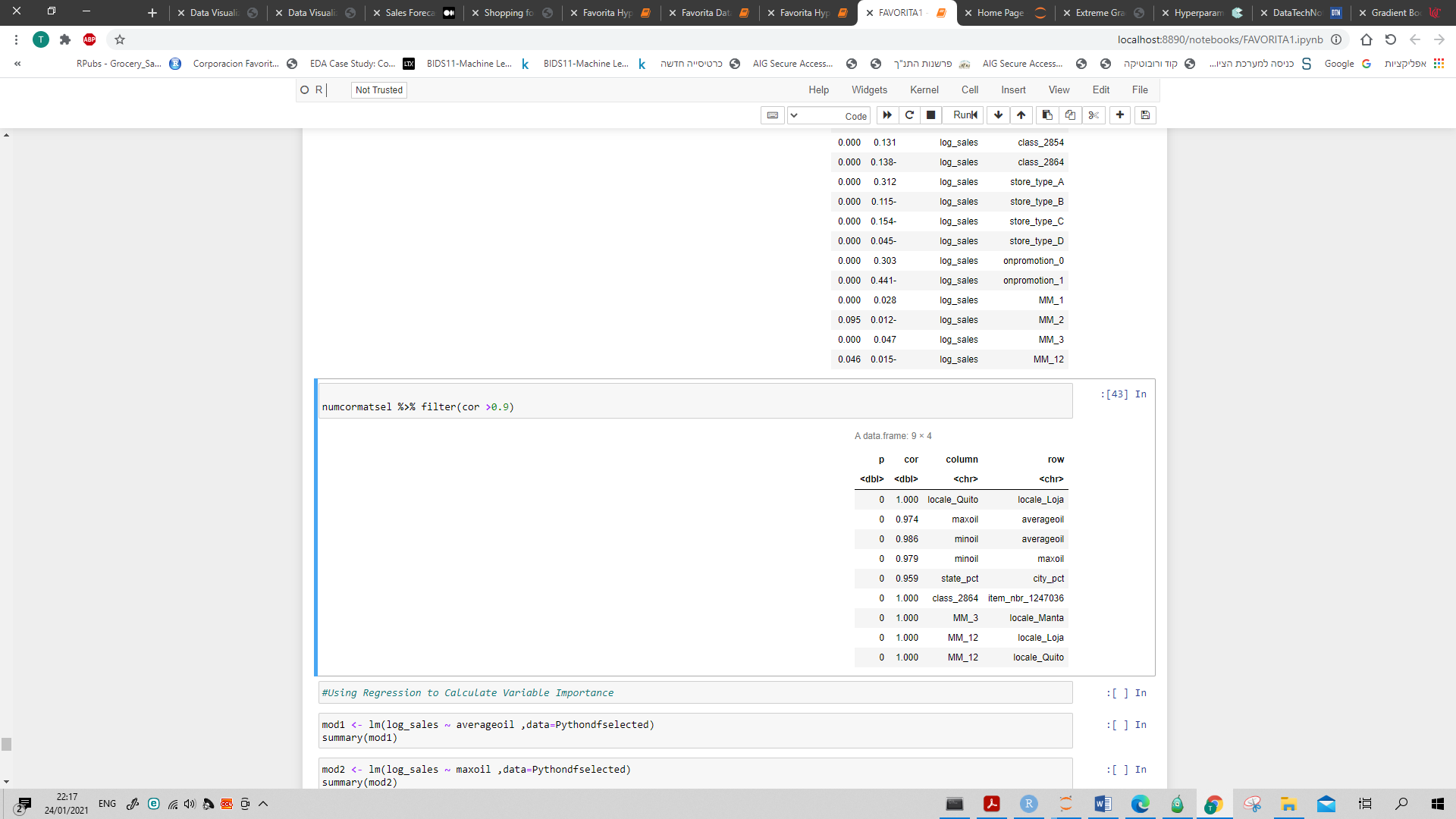
* Feature selection and voting (Python) –

Since our label is continuous Table1 in Python is not running we'll first have the multivariate analysis, make the voting procedure and then procced with univariate analysis and correlations to the label variable using R.

Using Lasso, RandomForest, GradientBoost and SVM end by voting for total\_count >1 which reflects 34 selected variables.



Correlation > 0.9 between the selected variables:



Using Regression to calculate variable importance and removing Average\_oil and Min\_oil variables.

Moreover City and State has high correlation, after checking the impact on log\_sales by simple lm regression we'll reduce State\_pct.

**The final data dimension - 32 x 18,572**

Data retrieval protocol attached in the Appendix.

## Models

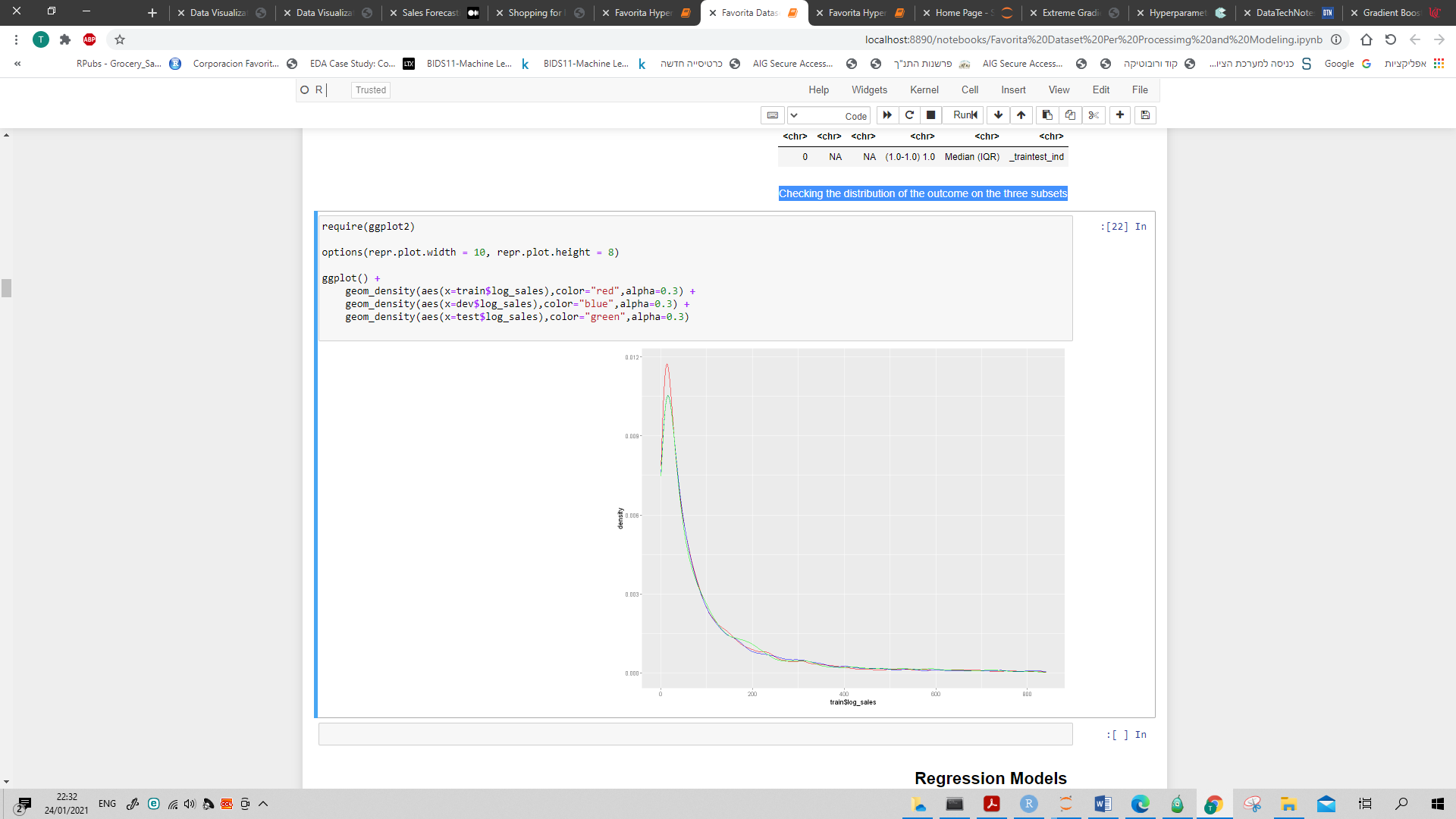
* Preparing the data for modeling – use Table 1 and divide the data into 3 perfectly balanced datasets:

Test (20%) : 32 x 3,715 (green)

Dev (16%) : 32 x 2,972 (blue)

Train (64%) : 32 x 11,885 (red)

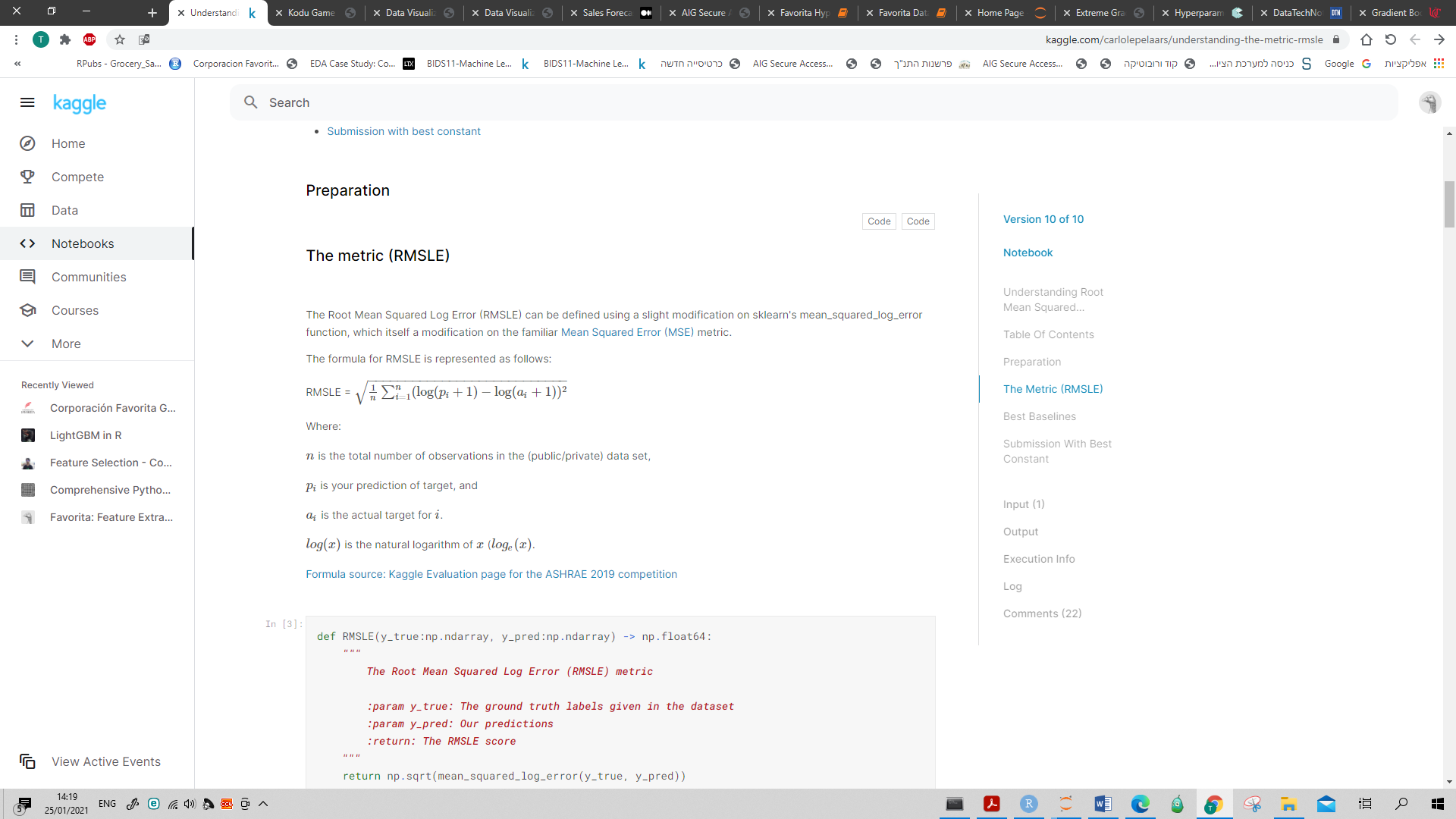
Checking the distribution of the outcome on the three subsets.



Since the outcome is a continuous variable (log\_sales) will use regression techniques for predicting.

* Two metrics chosen for comparing the models (since outliers are absent):

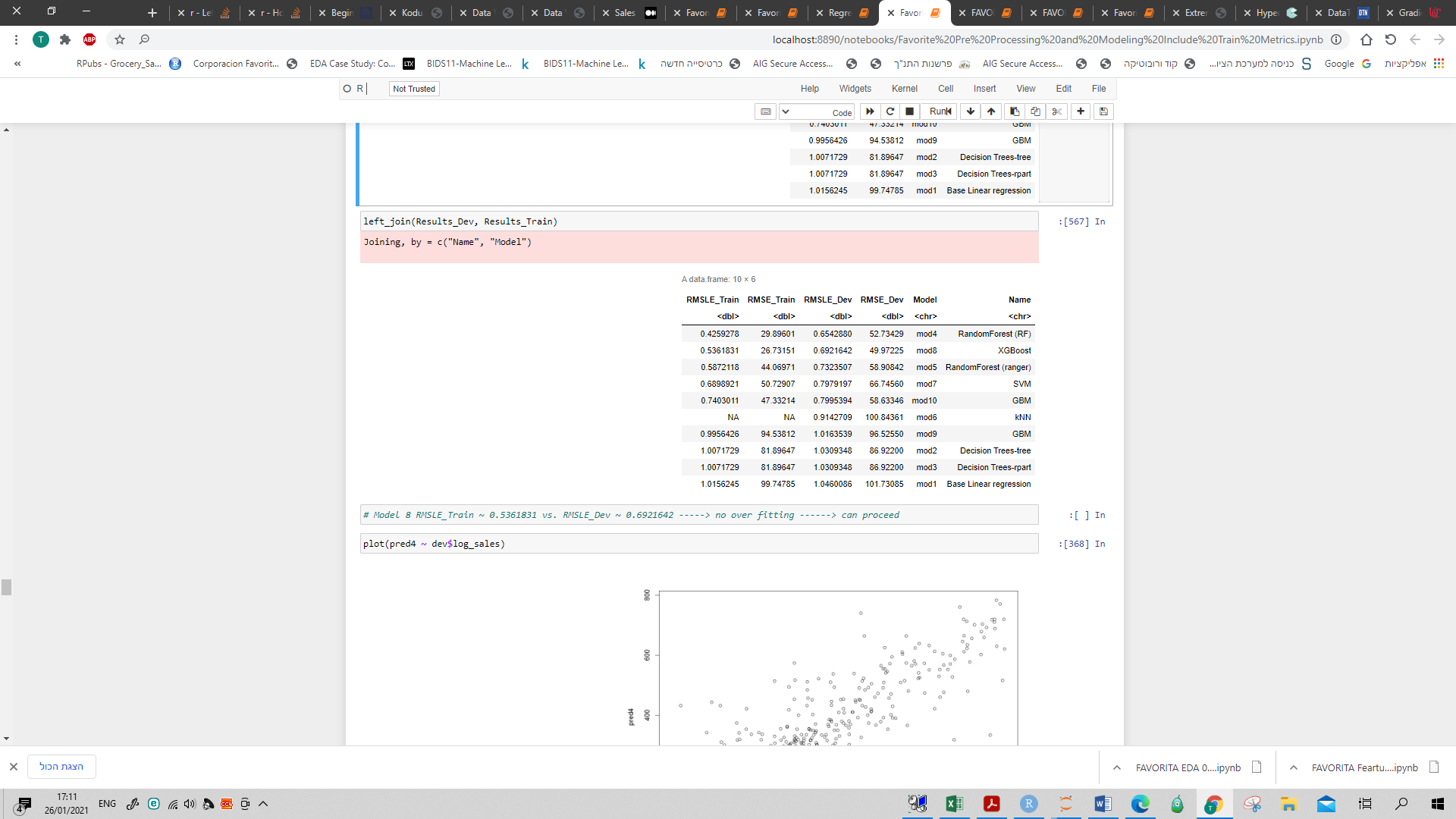




* We run ten different models.

Attached are the ordered metric results (Dev vs. Train results):

Seems that no over fitting occurs.



* Choose the best model by checking the metrics.

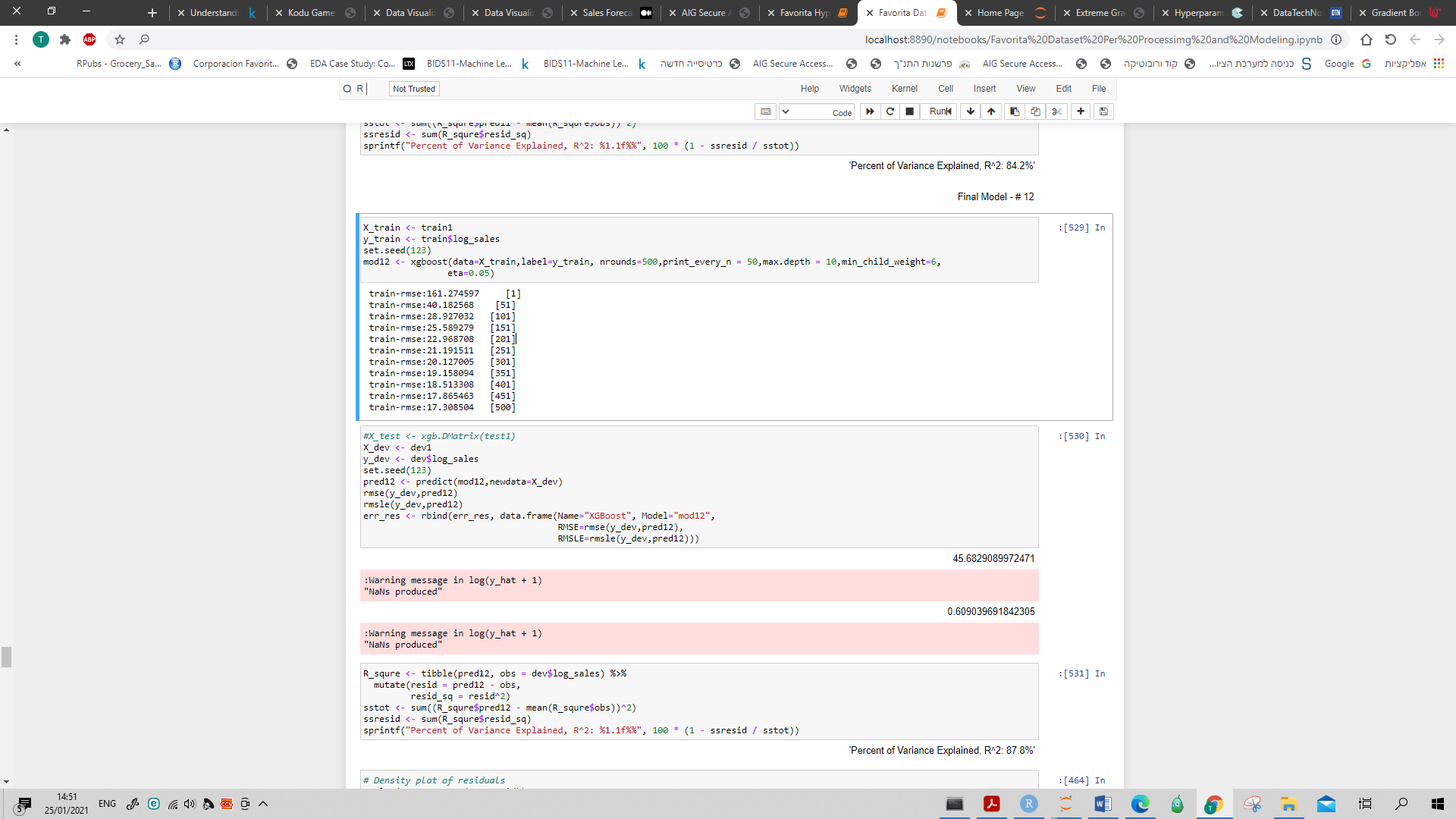
In terms of RMSLE the two top models are RandomForest(RF) and XGBoost.

In terms of RMSE the top model is XGBoost, thus the chosen model would be XGBoost.

The plot graph of the predictive vs log\_sales:



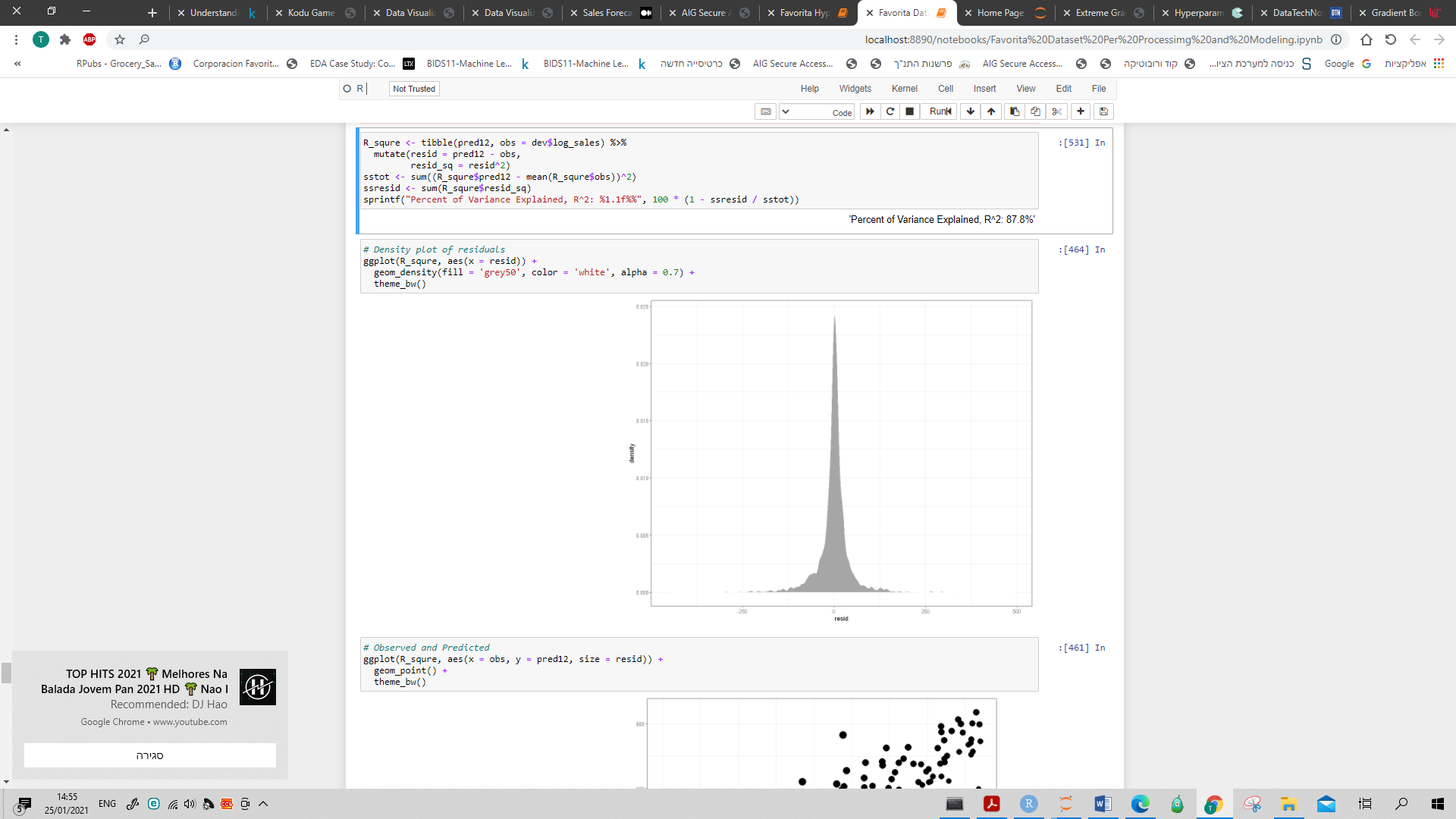
* Now we'll try to "manually" tune and improve the metrics by adding some parameters (model #12):



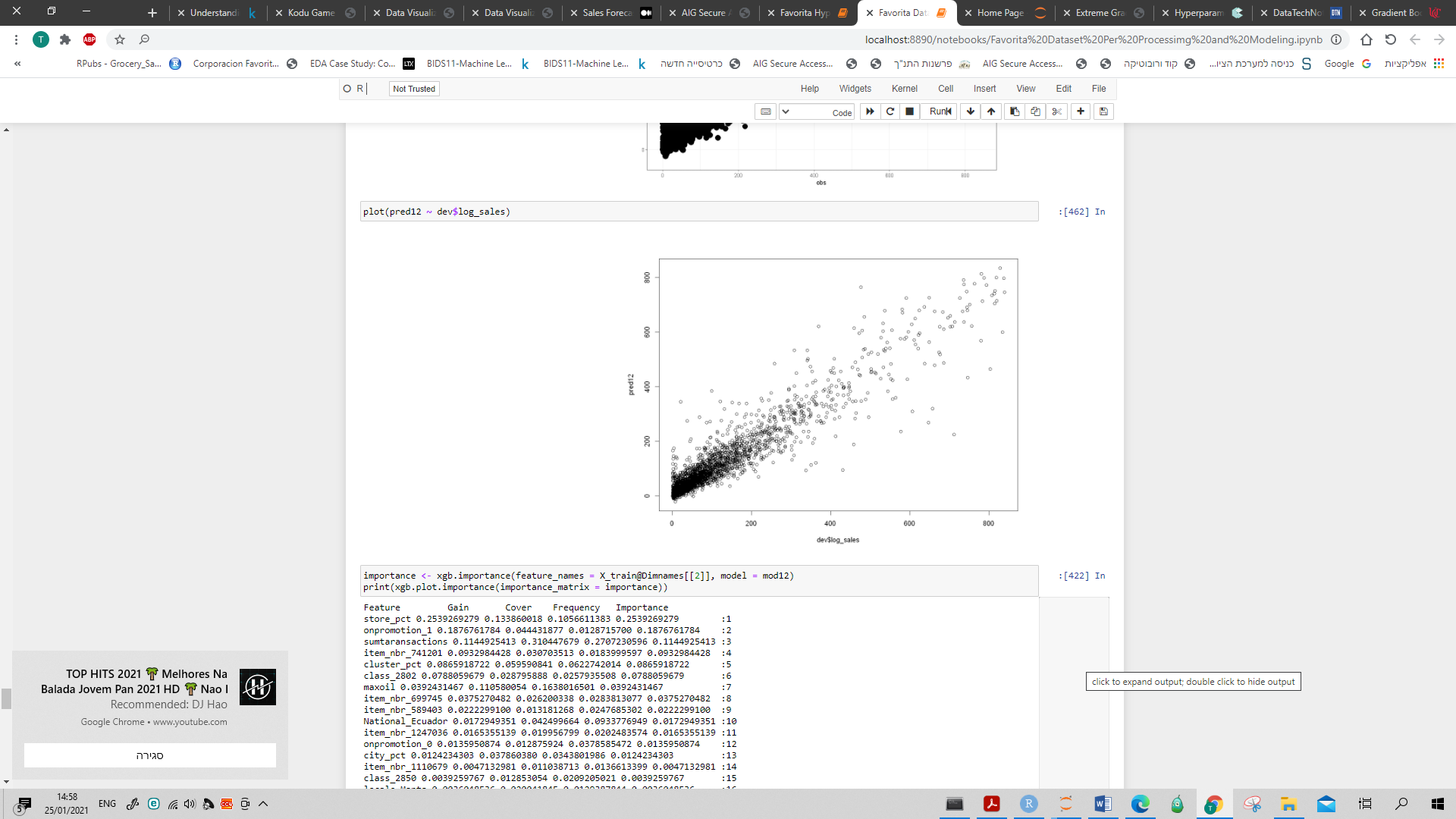
Results in RMSE ~ 45.6829 and RMSLE ~ 0.60903 (reflects an improvement of 12%).

The R^2 (percent of variance explained) stands at 87.8%.

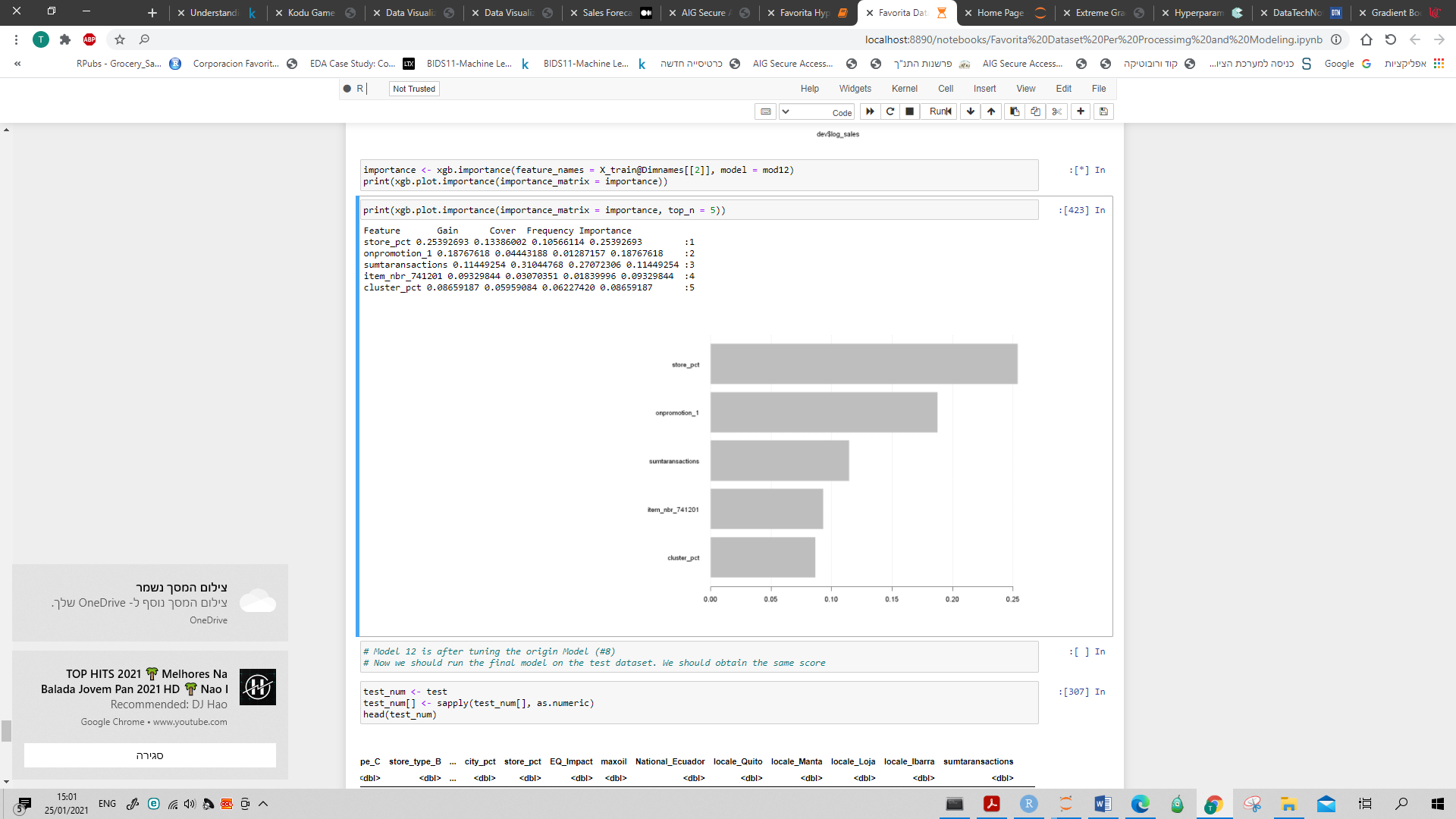
Attached is the density plot of the residuals:



Attached is the plot of the predictive vs. log\_sales:



Attached is the top 5 feature importance graph:



* Next step should be hyperparameter and fine tuning of the XGBoost model (an auto process).

But first we'll consider the manual model, as if it was determined as the perfect model, and take a look on the metrics results when running the model on the Test subset.

Results:

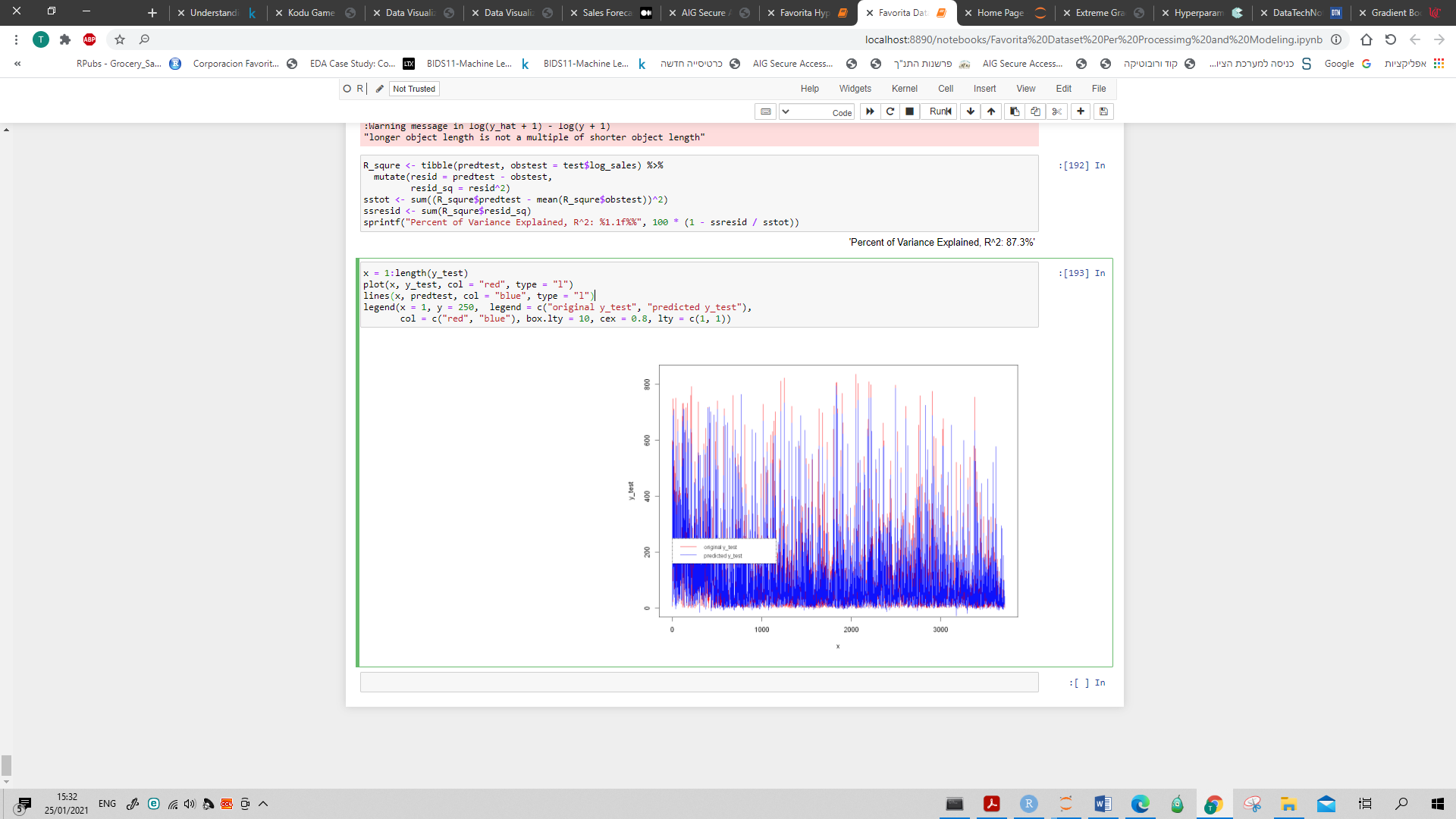
RMSE ~ 46.6116

RMSLE ~ 0.58489

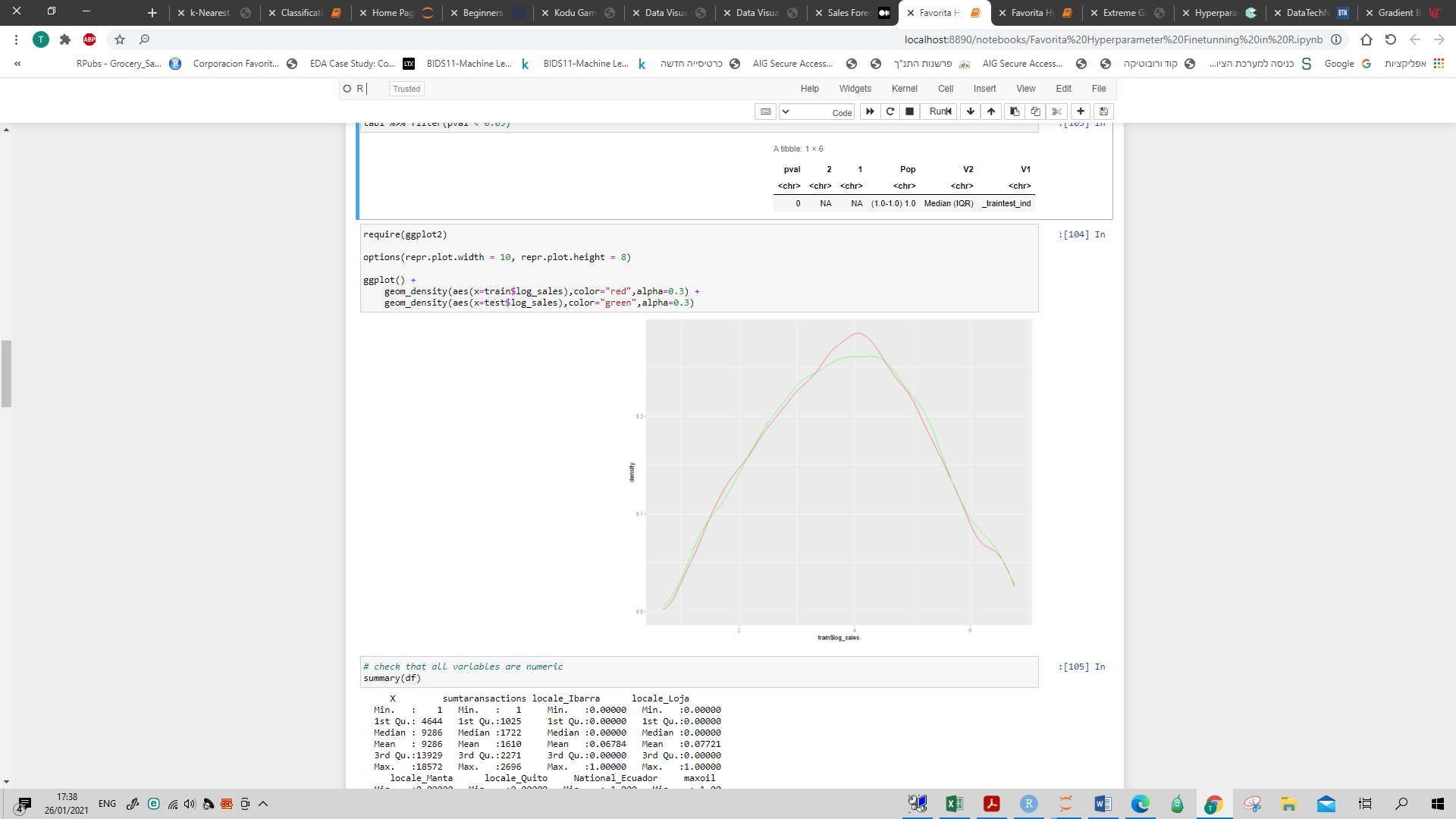
R^2 ~ 87.3%

Seems that there's no over fitting and that the model results in the same value level between the train and test.

Plot of the log\_sales (red) vs. the prediction (blue) on the test dataset:



* XGBoost Hyperparameter and fine tuning
* Dividing the data to Train (80%) and Test (20%) subsets by Table1 (Mechkar) in order to get balanced subsets:



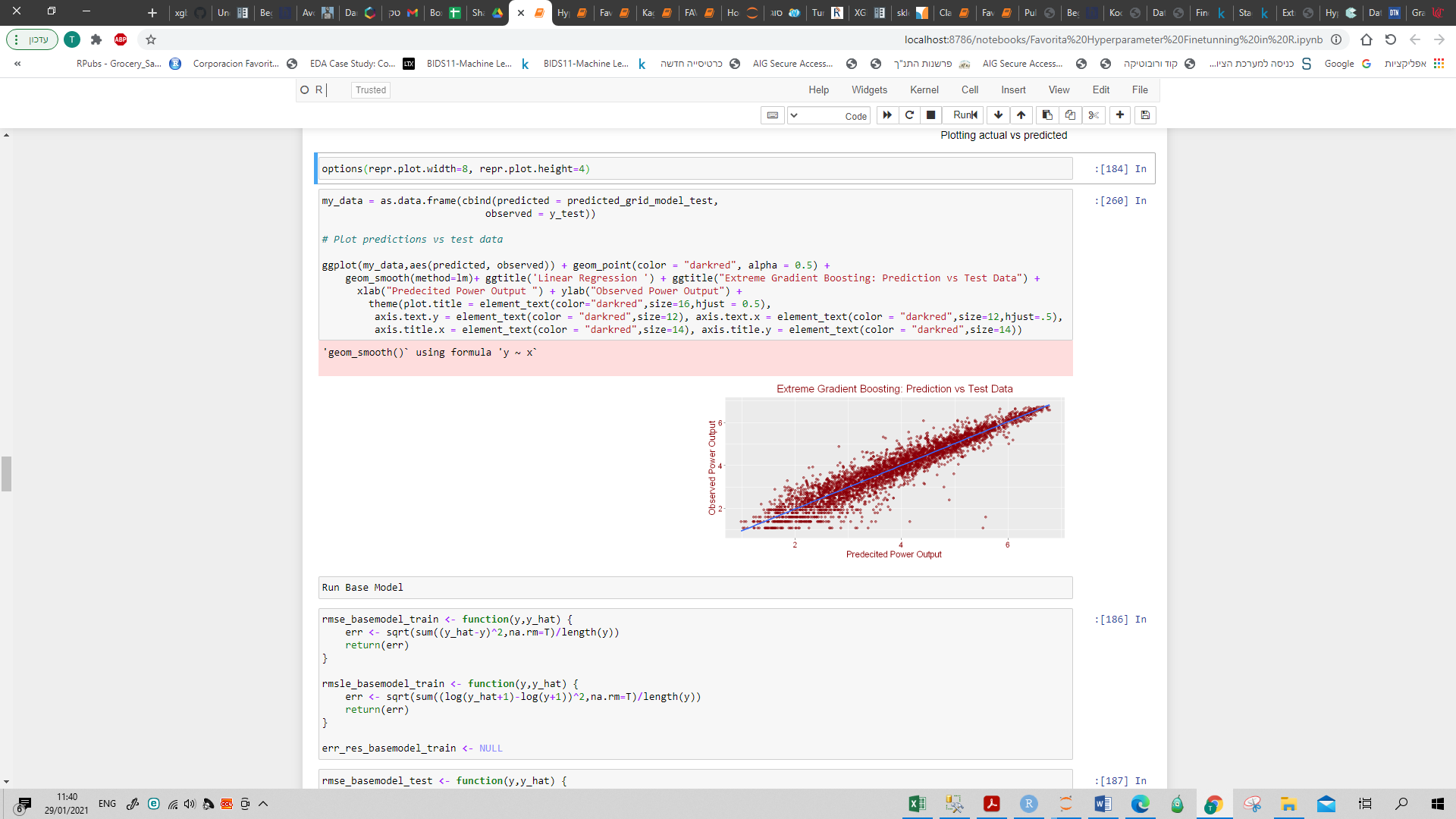
* Set a grid space to search for the best hyperparameters (appendix).

The best parameters out of that grid are:

* nrounds =500
* max\_depth = 15
* colsample\_bytree = 0.7
* eta = 0.05
* gamma=0
* min\_child\_weight = 8
* subsample = 1

The RMSLE (on test subset) of the best grid ~ 0.117 and the R^2 ~ 88%.

* Plot the Obsereved vs. Predicted



* We'll make a comparison between the base model and the best parametrs from the grid model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Date Set | RMSE | RMSLE | R^2 |
| Base Model | Test | 0.469 | 0.119 | 87.4% |
| Train | 0.234 | 0.062 |  |
| Grid Model | Test | 0.447 | 0.115 | 88.5% |
| Train | 0.156 | 0.041 |  |

Since the grid model was built on train-test dataset (revised), will check the results of the base model that was chosen in previous section again (on the revised datasets).

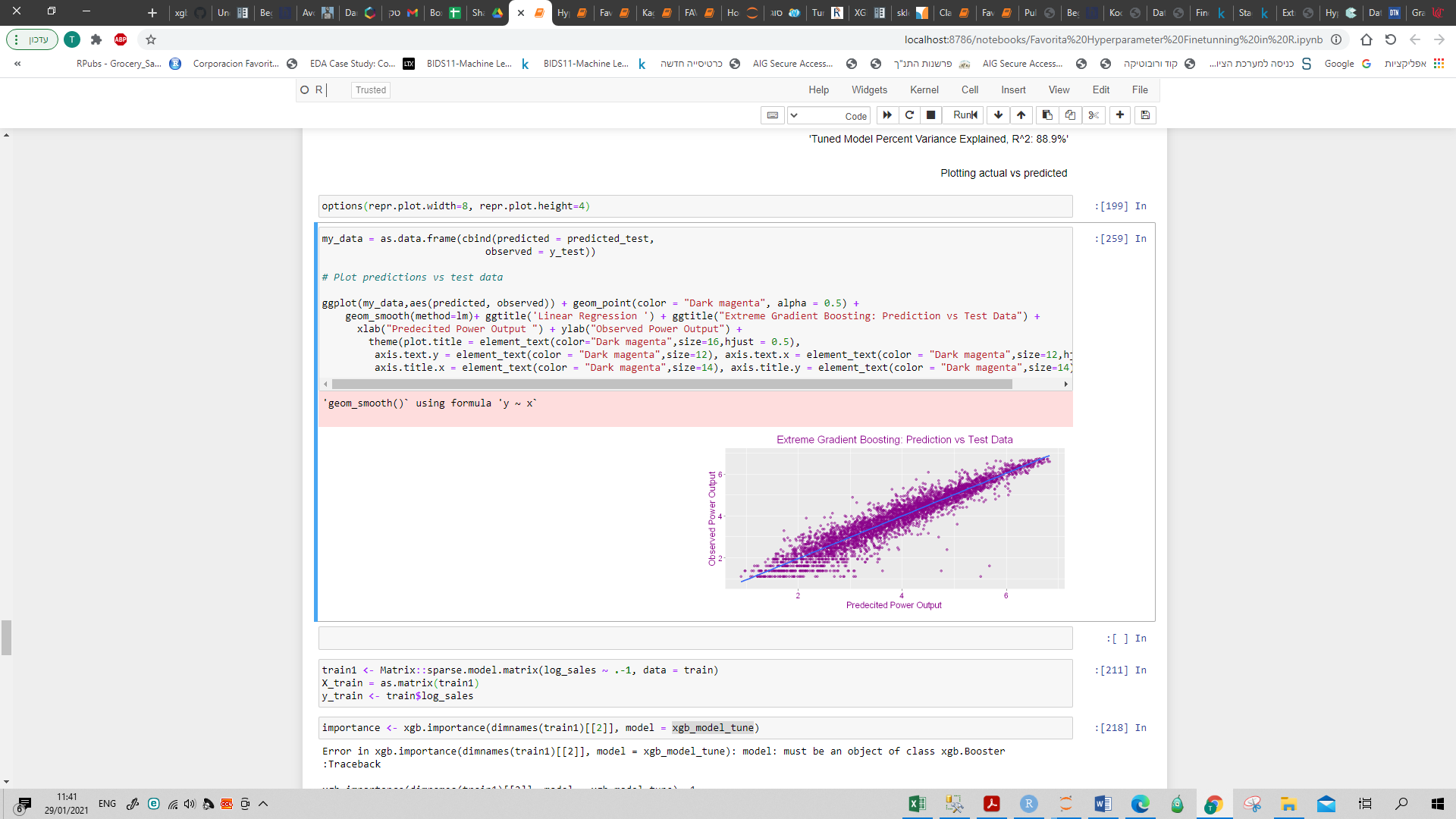
As can be seen there's no over fitting and the RMSLE of the grid model is better than the one of the base model.

* We'll fine tune the grid model by narrow the vector of each parameter around the best parameter from the previous step. The best parameters out of that tuned grid are:
* nrounds = 550
* max\_depth = 10
* colsample\_bytree = 0.6
* eta = 0.06
* gamma= 0
* min\_child\_weight = 8
* subsample = 1
* Check tunes grid results vs. the best grid results and the base model:

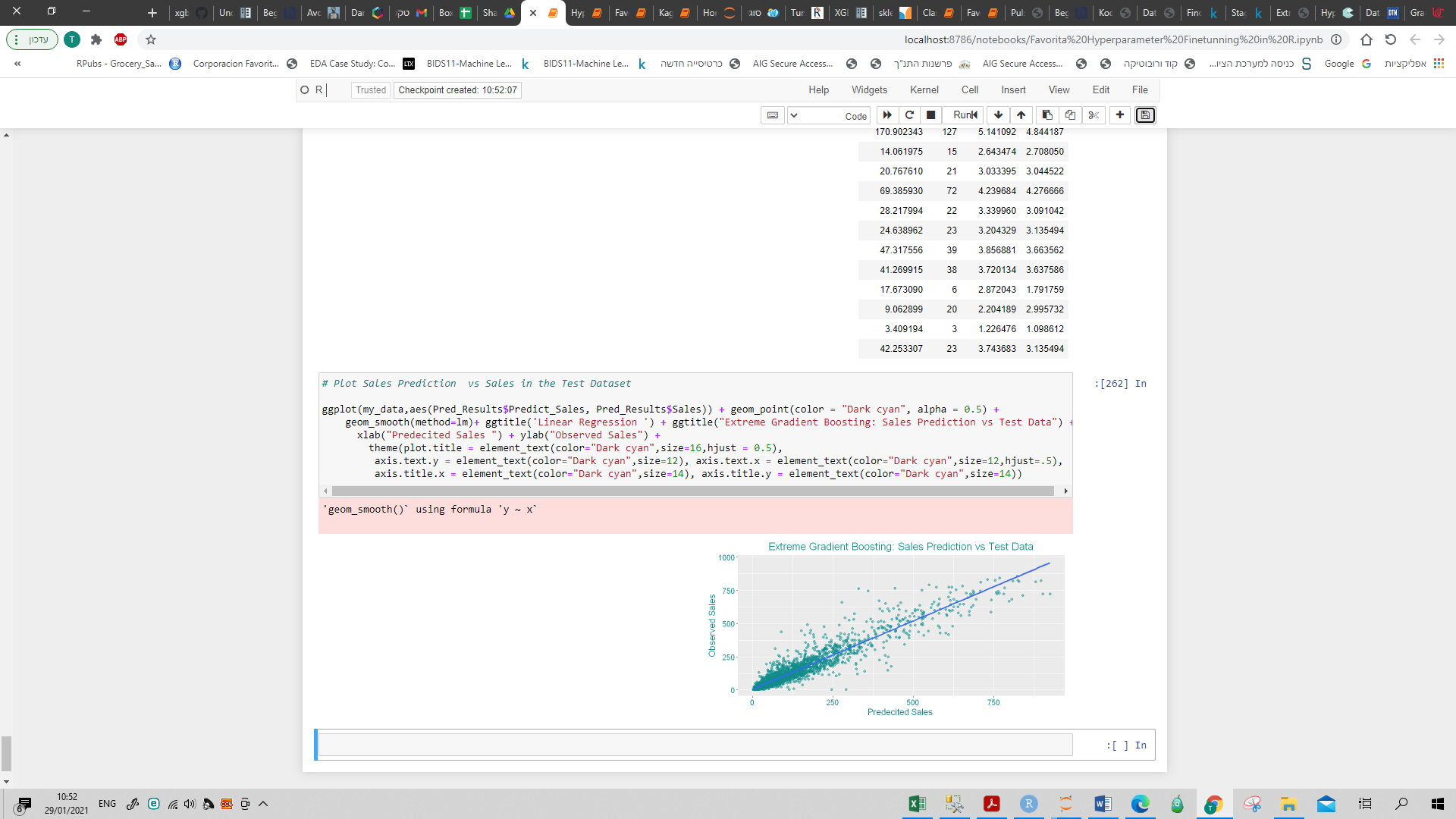
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Date Set | RMSE | RMSLE | R^2 |
| Base Model | Test | 0.469 | 0.119 | 87.4% |
| Train | 0.234 | 0.062 |  |
| Grid Model | Test | 0.447 | 0.115 | 88.5% |
| Train | 0.156 | 0.041 |  |
| Tuned Model | Test | 0.439 | 0.112 | 88.9% |
| Train | 0.225 | 0.059 |  |

The tuned model (final model) results are better than the base model and the grid model.

Its RMSLE on the test dataset stands at 0.059, an improvement of 2.4% compared to the grid model. The R^2 stands at 88.9%.



The final step is looking at the results; predictive sales values (by taking the exponential due to our earlier transformation) vs. the sales column in the test dataset.

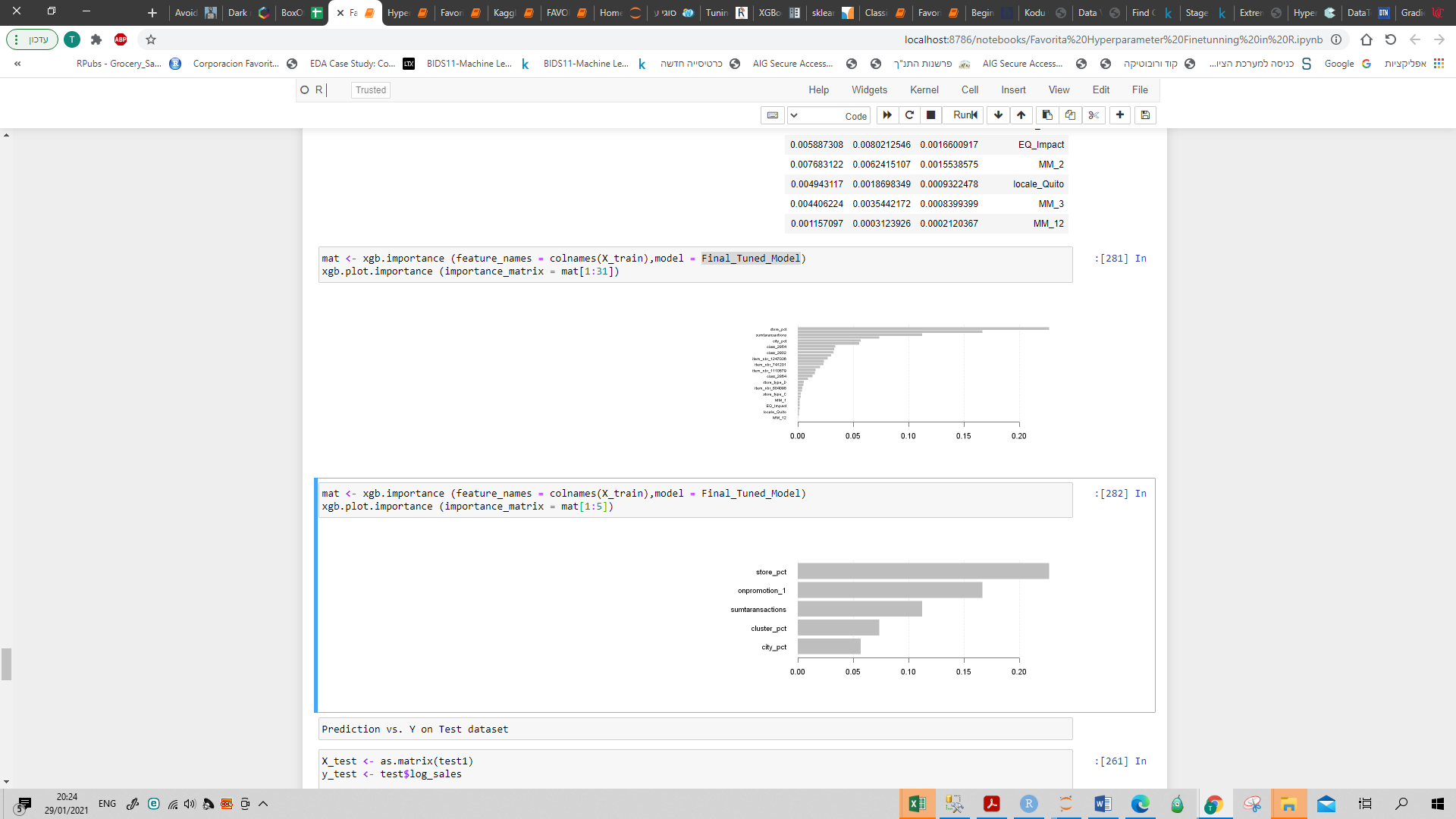


Seems that the model predicts well for unit sales < 250 which represent 90% out of monthly sales data (the error is lower in that range). Yet, most of the high level sales are around the diagonal line, which represent perfect fit.

One should remember that the less the data the more the variability increase, so the final model seems pretty good.

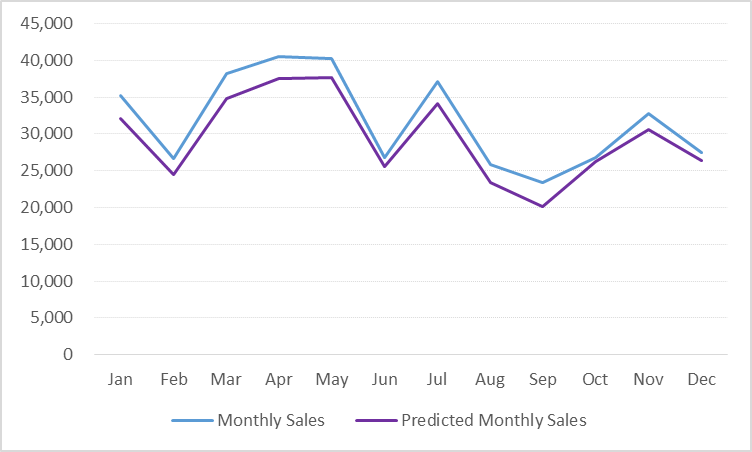
|  |  |  |  |
| --- | --- | --- | --- |
| Sales Category | Count | Mix | RMSE |
| 0-250 | 3,312 | 89% | 28.4 |
| 250-750 | 383 | 10% | 100.7 |
| 750+ | 20 | 1% | 168.3 |
| Total Sales | 3,715 | 100% | 43.8 |

The top 5 features importance (a bit different from the base model):



\* Full list in the appendix

Monthly Sales Prediction

Model based prediction on monthly sales (test dataset):

|  |  |  |
| --- | --- | --- |
| **Month** | **Monthly Sales** | **Predicted**  **Monthly Sales** |
| Jan | 35,223 | 32,079 |
| Feb | 26,600 | 24,541 |
| Mar | 38,168 | 34,868 |
| Apr | 40,550 | 37,570 |
| May | 40,233 | 37,720 |
| Jun | 26,822 | 25,511 |
| Jul | 37,175 | 34,195 |
| Aug | 25,822 | 23,467 |
| Sep | 23,431 | 20,091 |
| Oct | 26,854 | 26,188 |
| Nov | 32,834 | 30,559 |
| Dec | 27,479 | 26,447 |
| **Total** | **381,191** | **353,236** |

The trend over the year is predicted pretty well. The average variance is around 7%, increased due to Sep, which has the highest variance ~ 14%.

# Results and Conclusions

The origin Favorita data contains over 125M row.

In this project we decided to focus on the Seafood family and predict its monthly sales over a year.

The initial Seafood dataset contains around 19K rows with 41 independent variables.

The label (unit sales) distribution over the years was stable.

No outliers were found and the missing data was handled by data reduction or adding category level of NA's.

EDA process and feature selection results with file of 32 variables and 18,572 rows.

In the model selection part we divided the data into 3 balanced datasets:

Train (64% of the data), Dev (16%) and Test (20%) run 10 different models.

The best model, in terms of RMSLE, was XGBoost. No overfitting was found.

For "self regulation" we "manually" improved the model by adding parameters to the model.

That model was set to be the base model and was checked on the Test dataset. Results were good.

In the model fine-tuning we divided the data into Train (80%) and Test (20%).

We set a grid for random search with XGBoost hyperparametets run the model and the predict on Train and Test.

We run the base model on the Train and Test datasets.

We fine tuned the grid and run the tuned model and the predict on Train and Test.

Comparing the 3 models results we set the tuned model as the final model which represent the best metric, lowest RMSLE ~ 0.112 and the best R^2 ~ 88.9%.

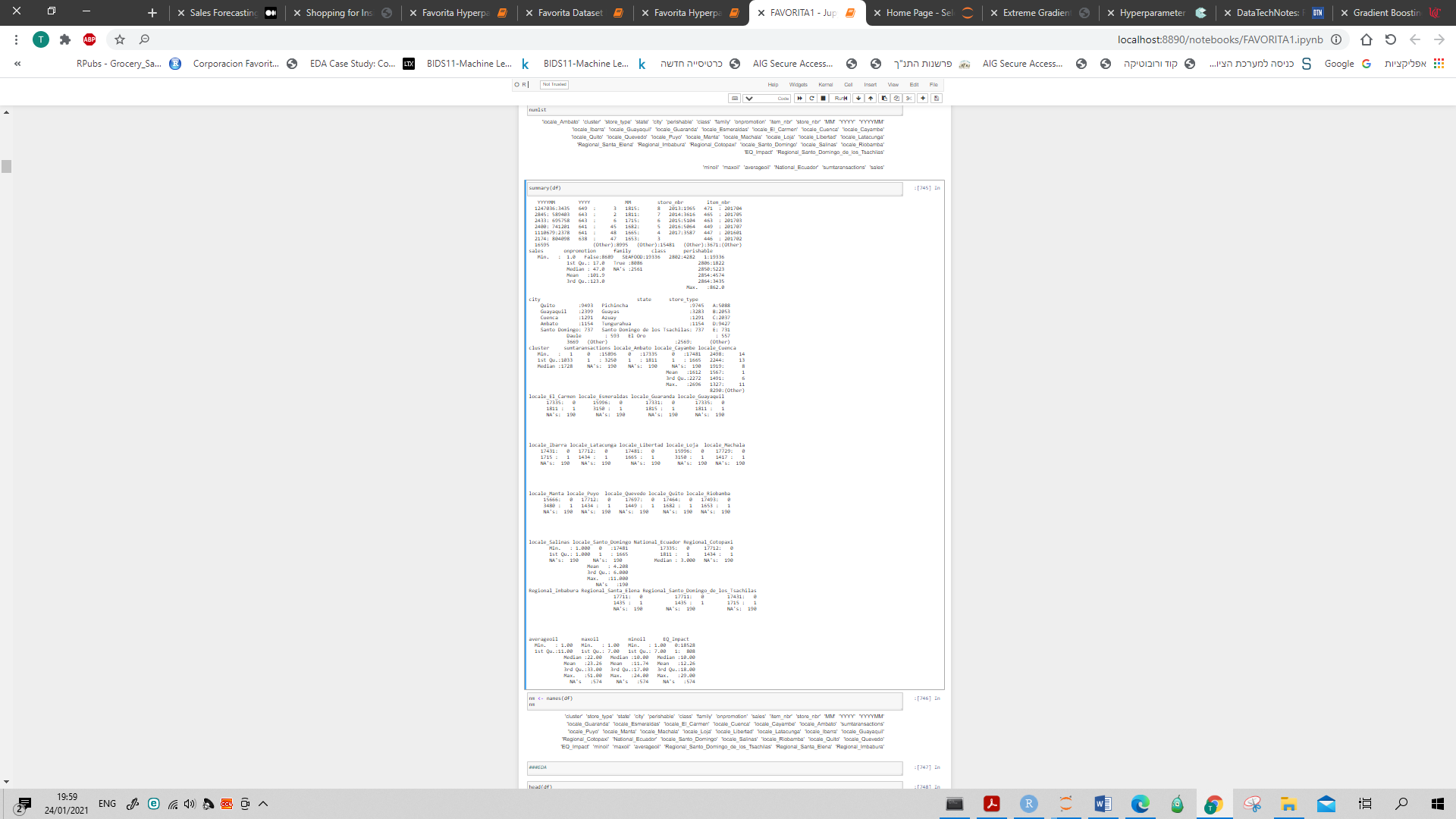
Finally we predict monthly sales based on the Test dataset.

As mentioned above the sales trend over the year was captured well but we still see a 7% variance, meaning that there's still a place to optimize the model.

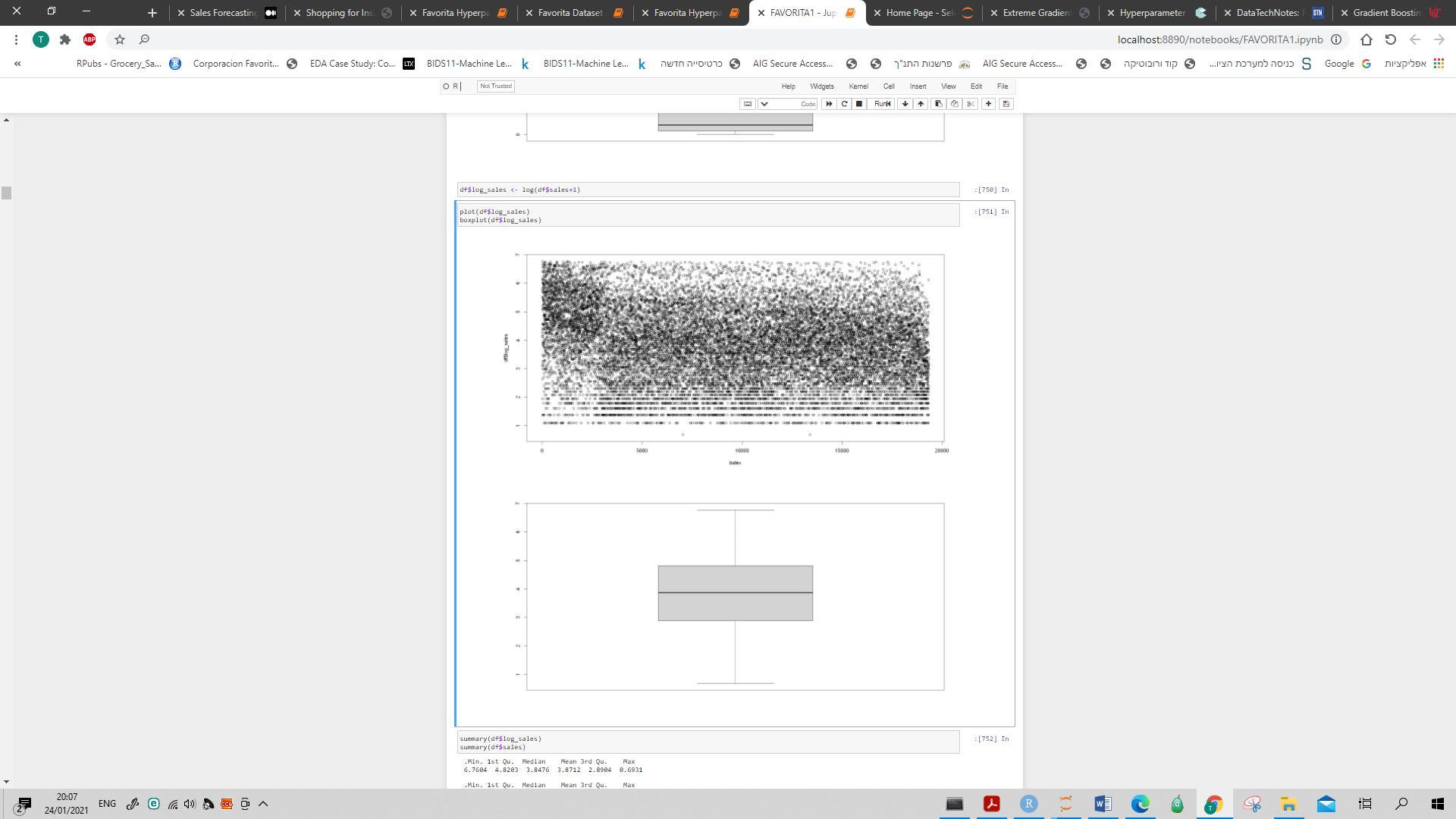
**Appendix**

****[**Data retrieval protocol**](https://docs.google.com/spreadsheets/d/1pYYjgwZ_8PS1Bcmc2kRNHTL0f_rk__GCJALLs1JHPUQ/edit#gid=0)**:**

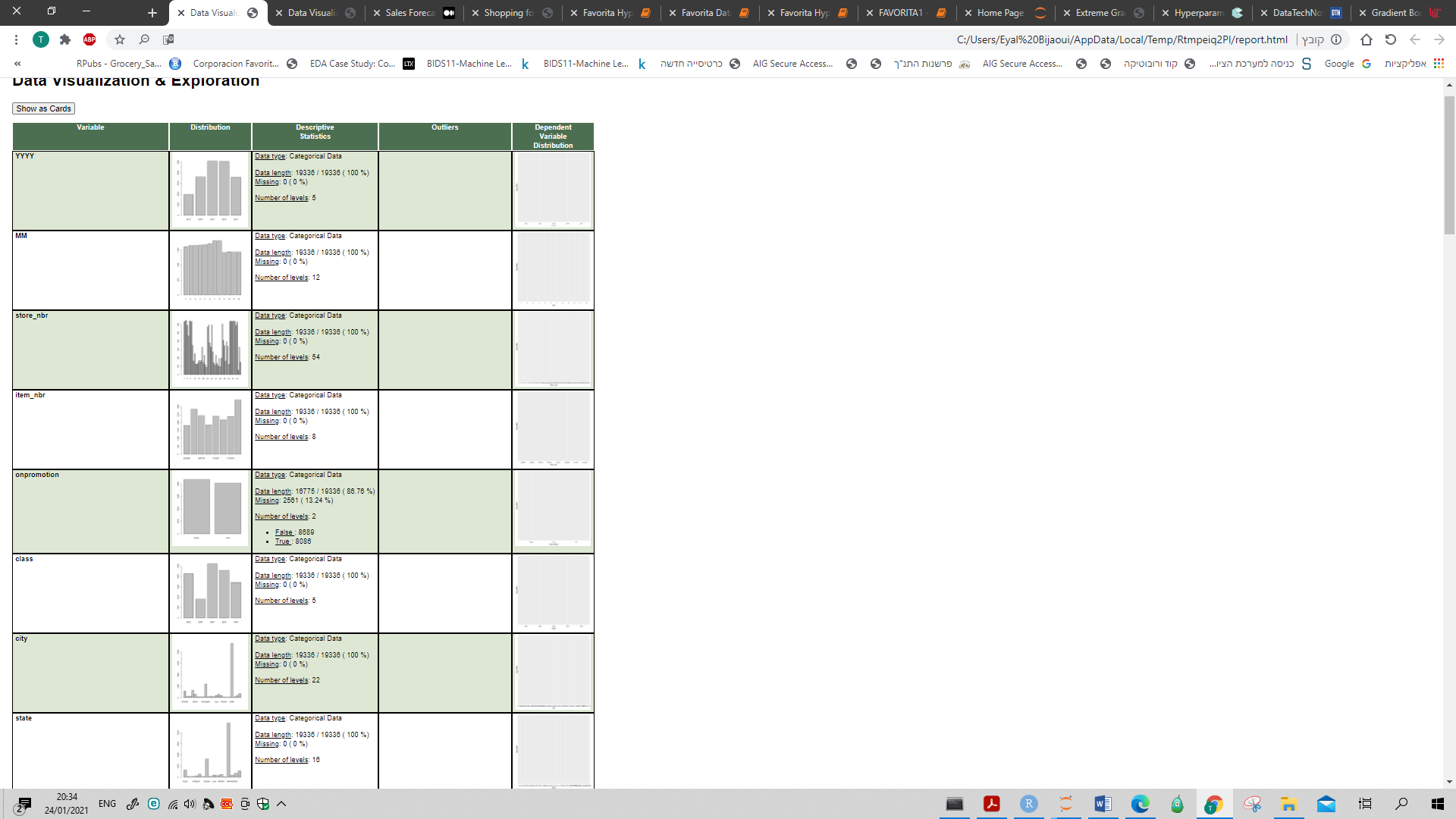
**Summary (data frame):**

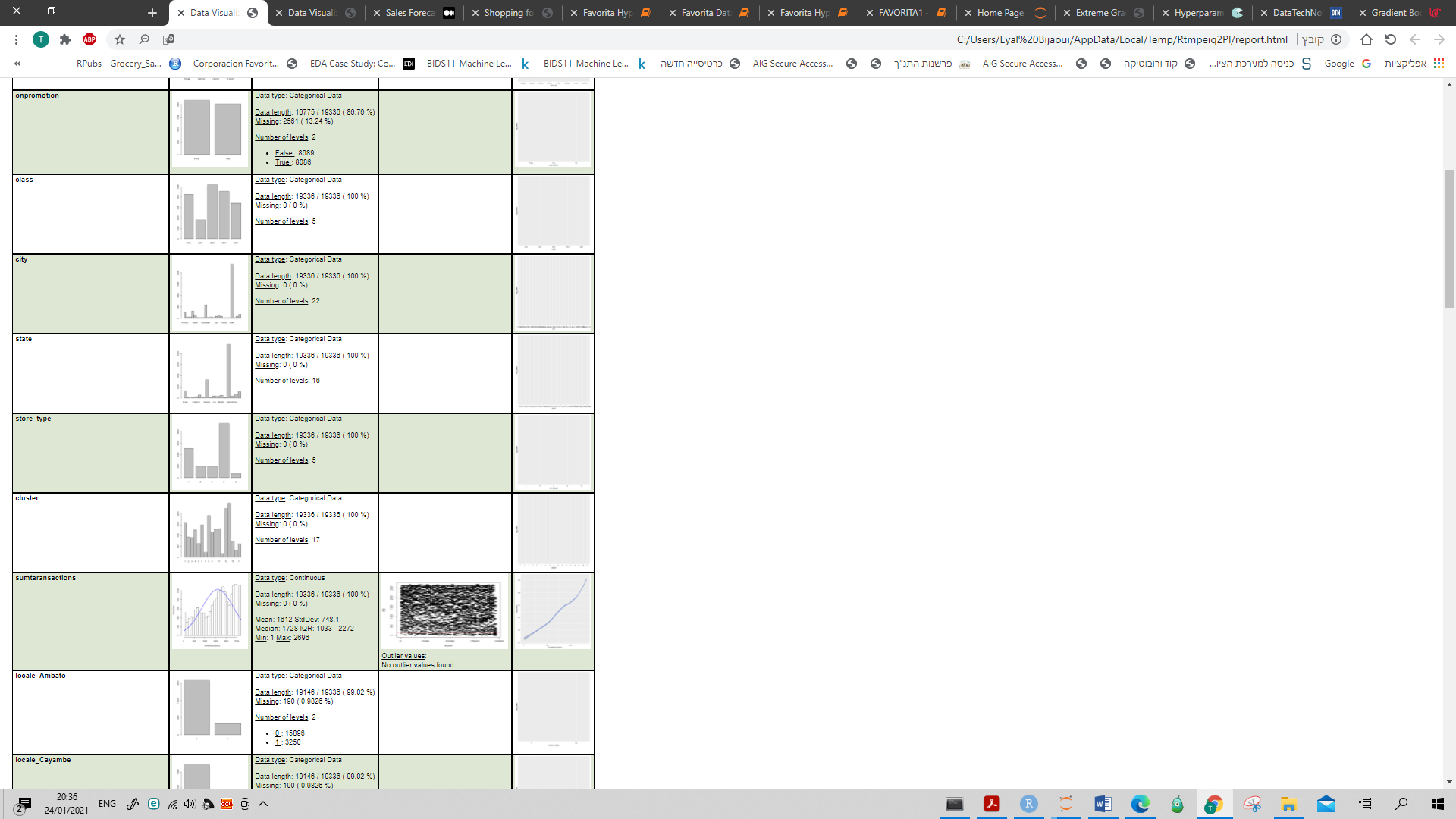


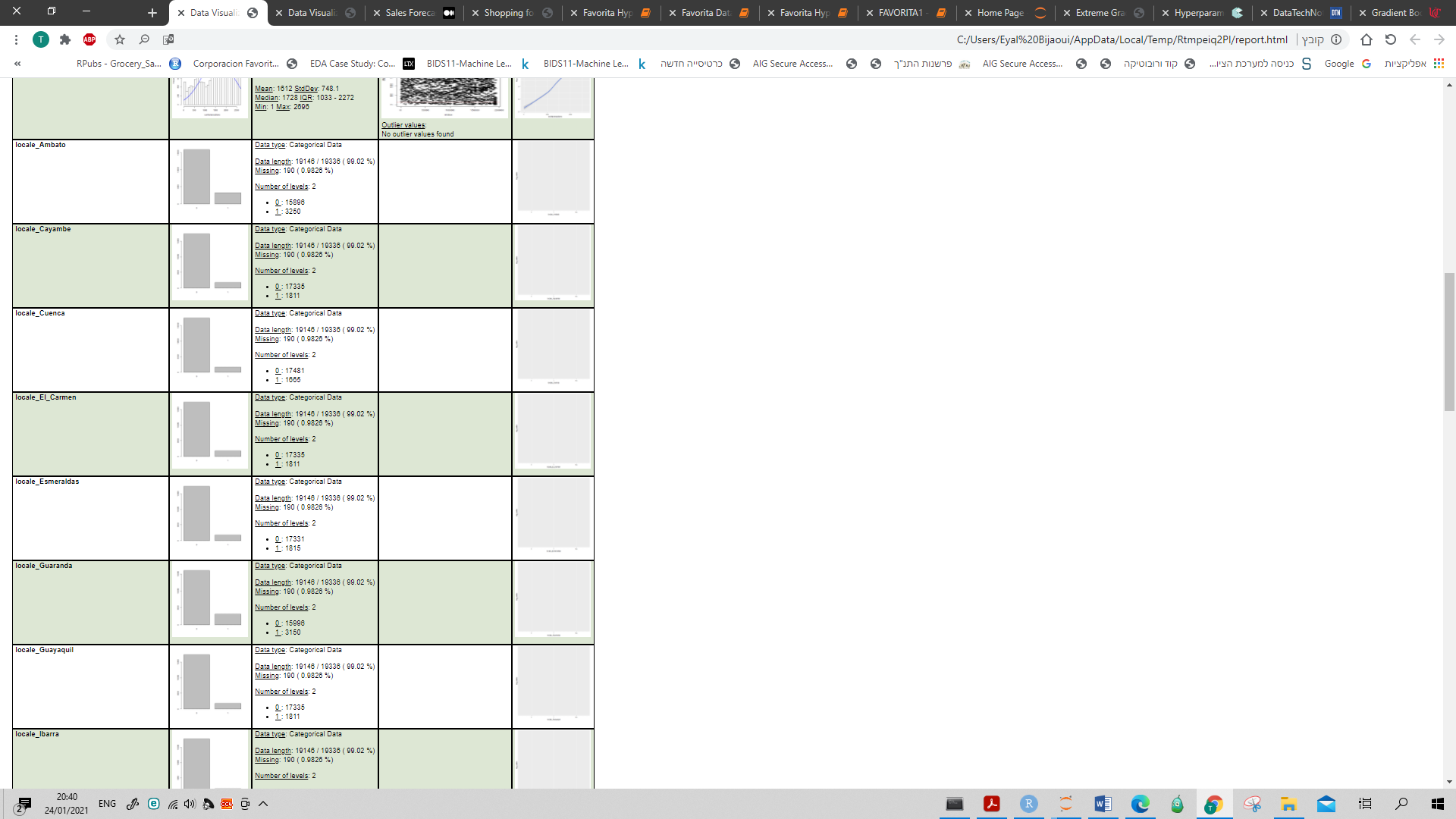
**Transforming the sales count using Log:**

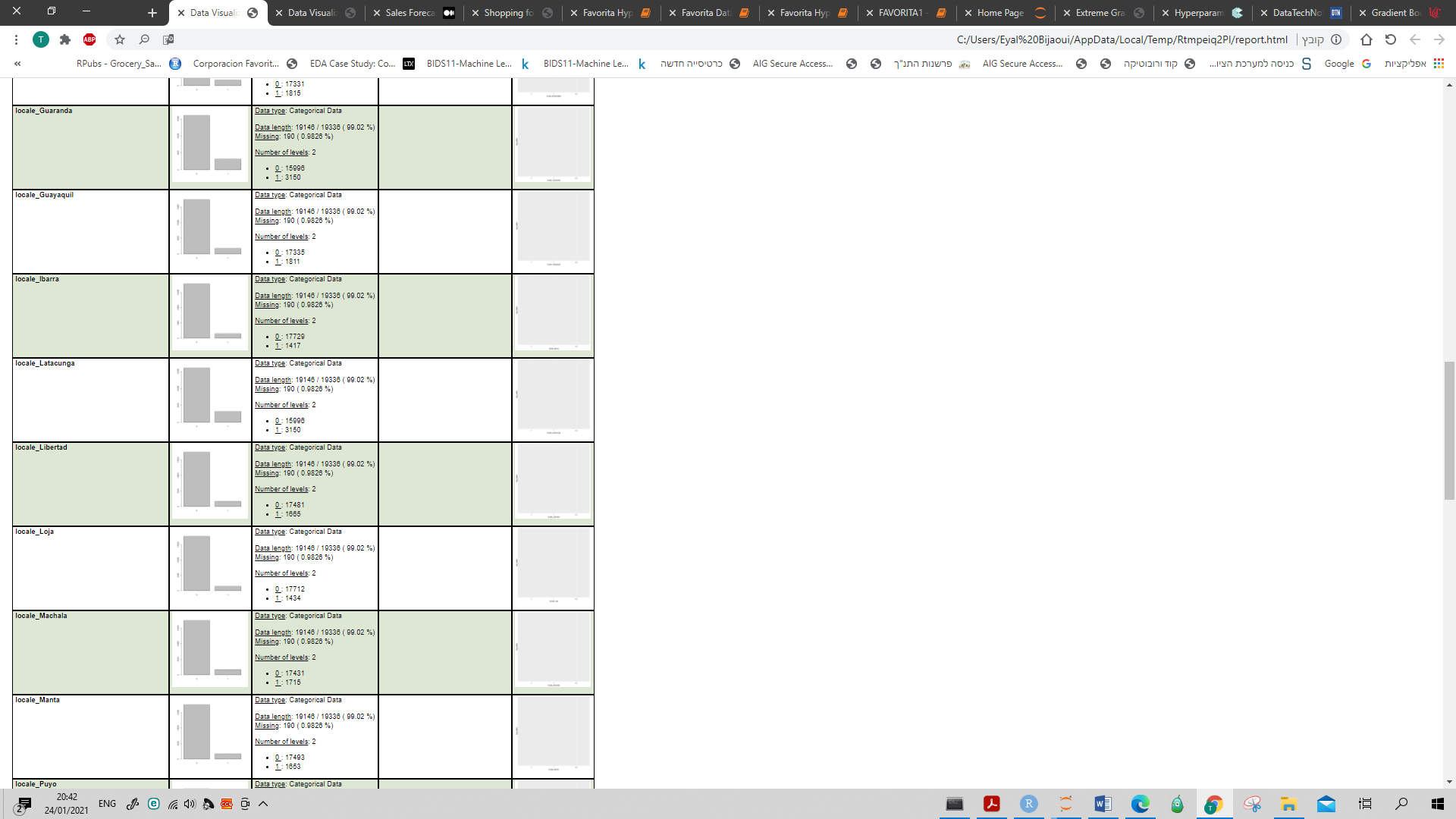


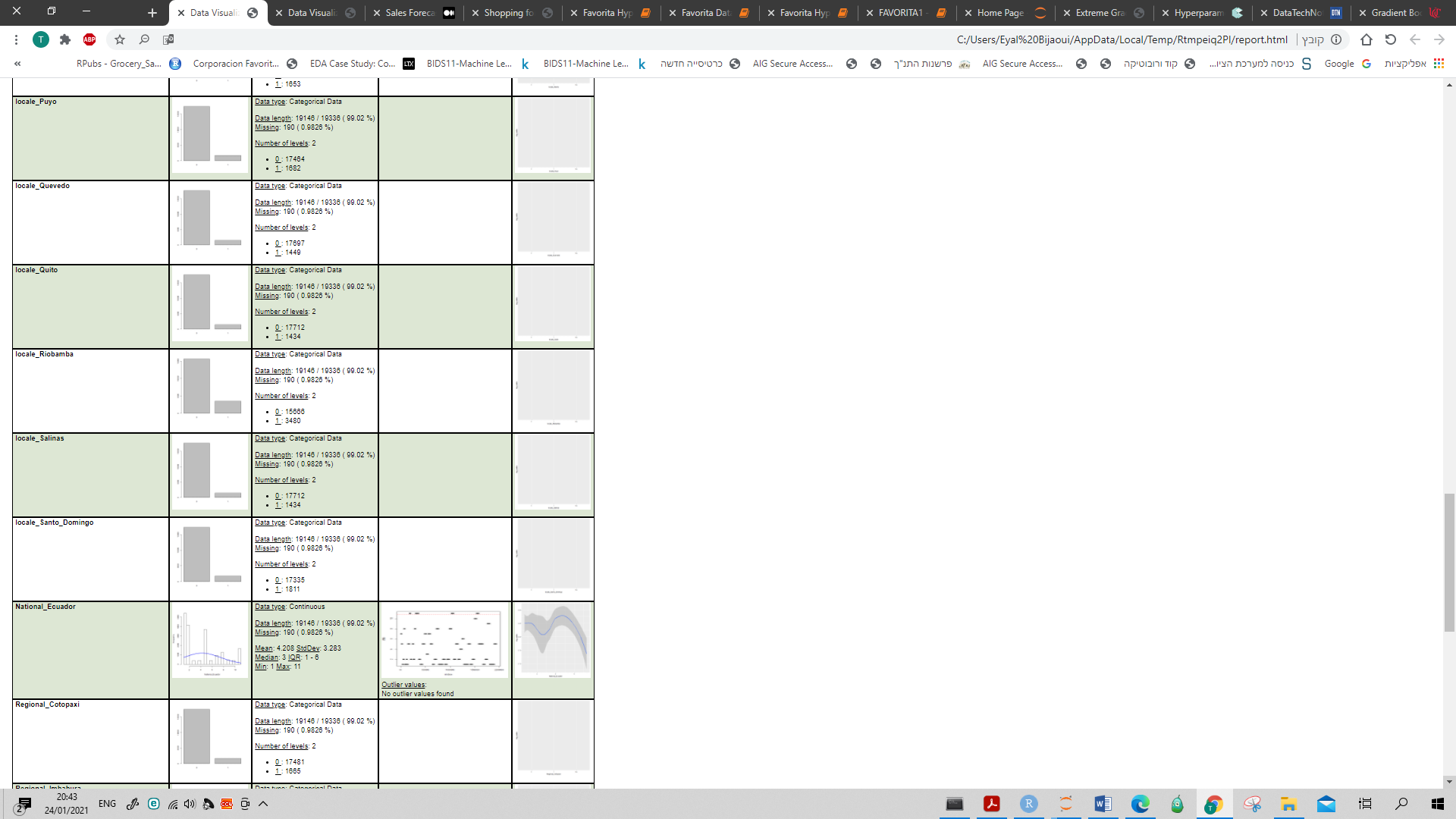
**Data Exploration & Visualization**

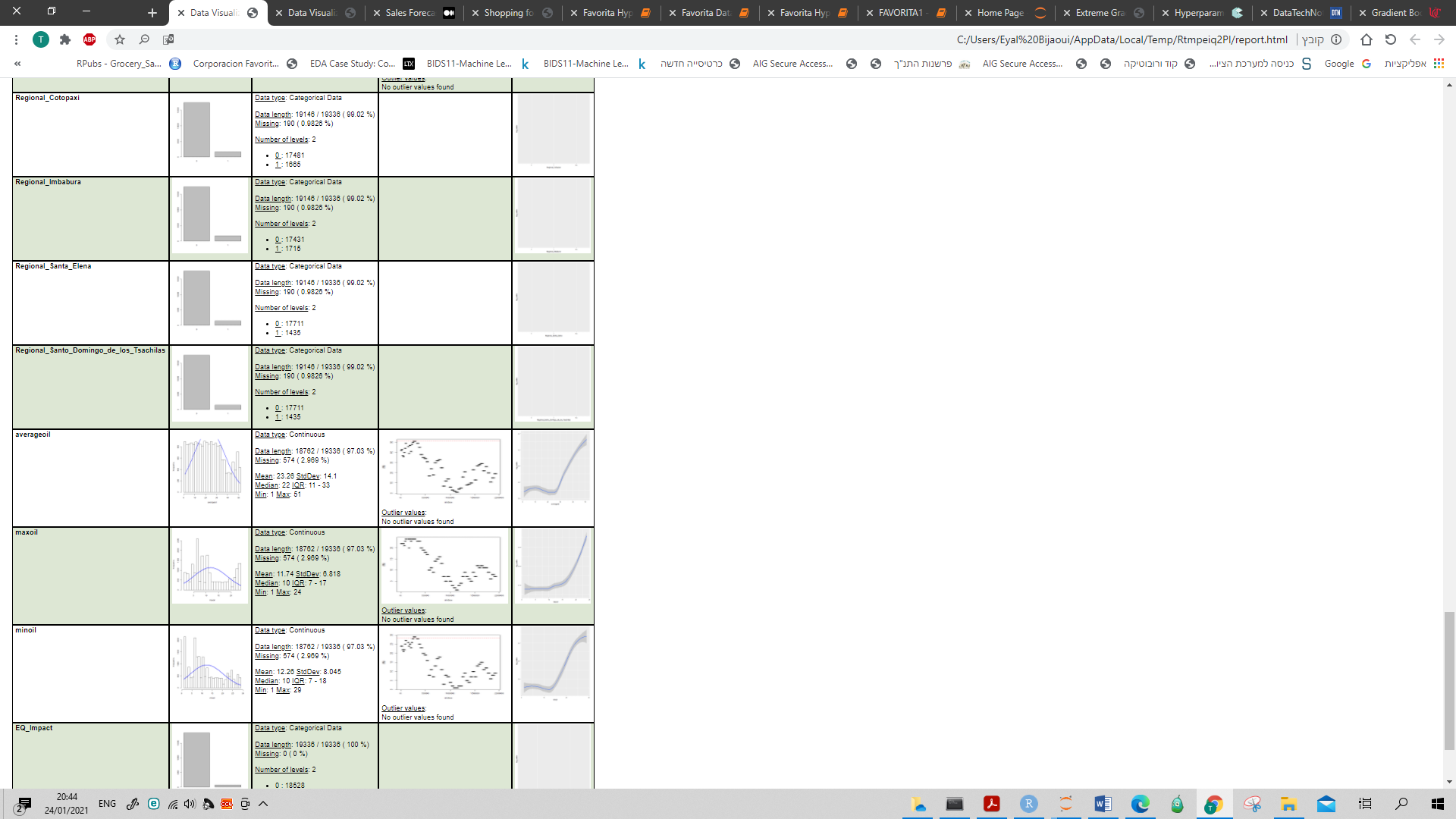






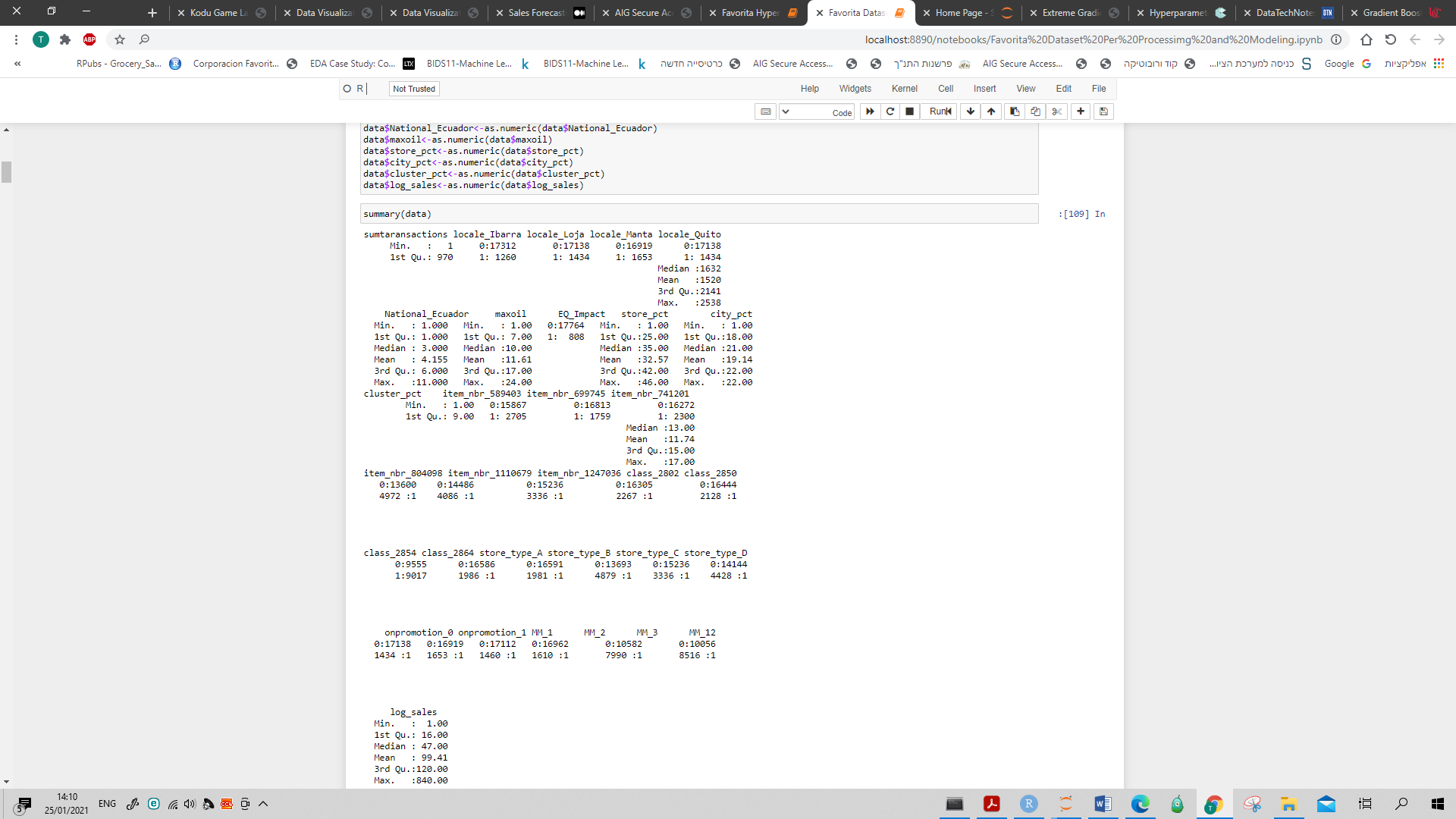




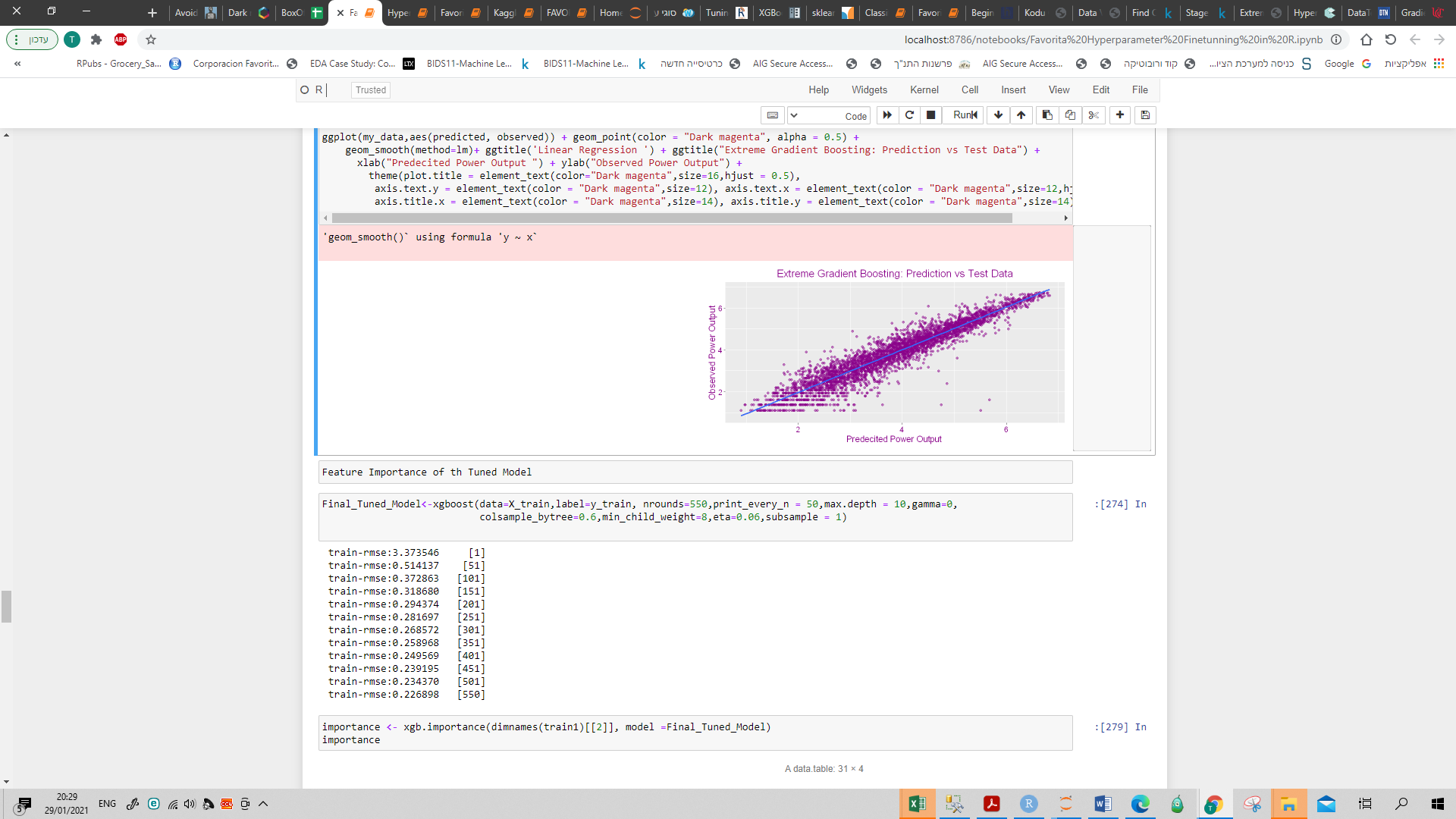


# 

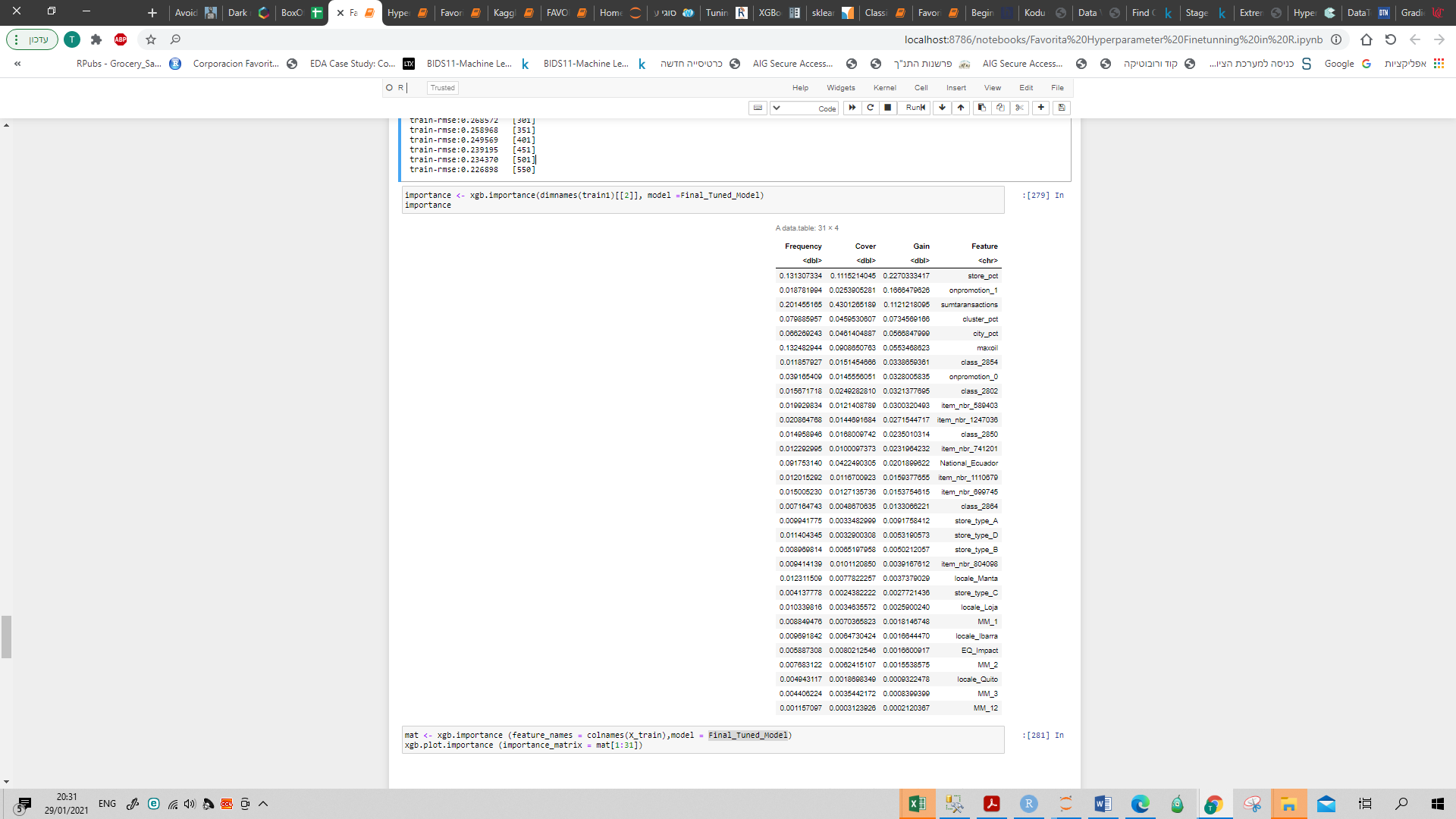
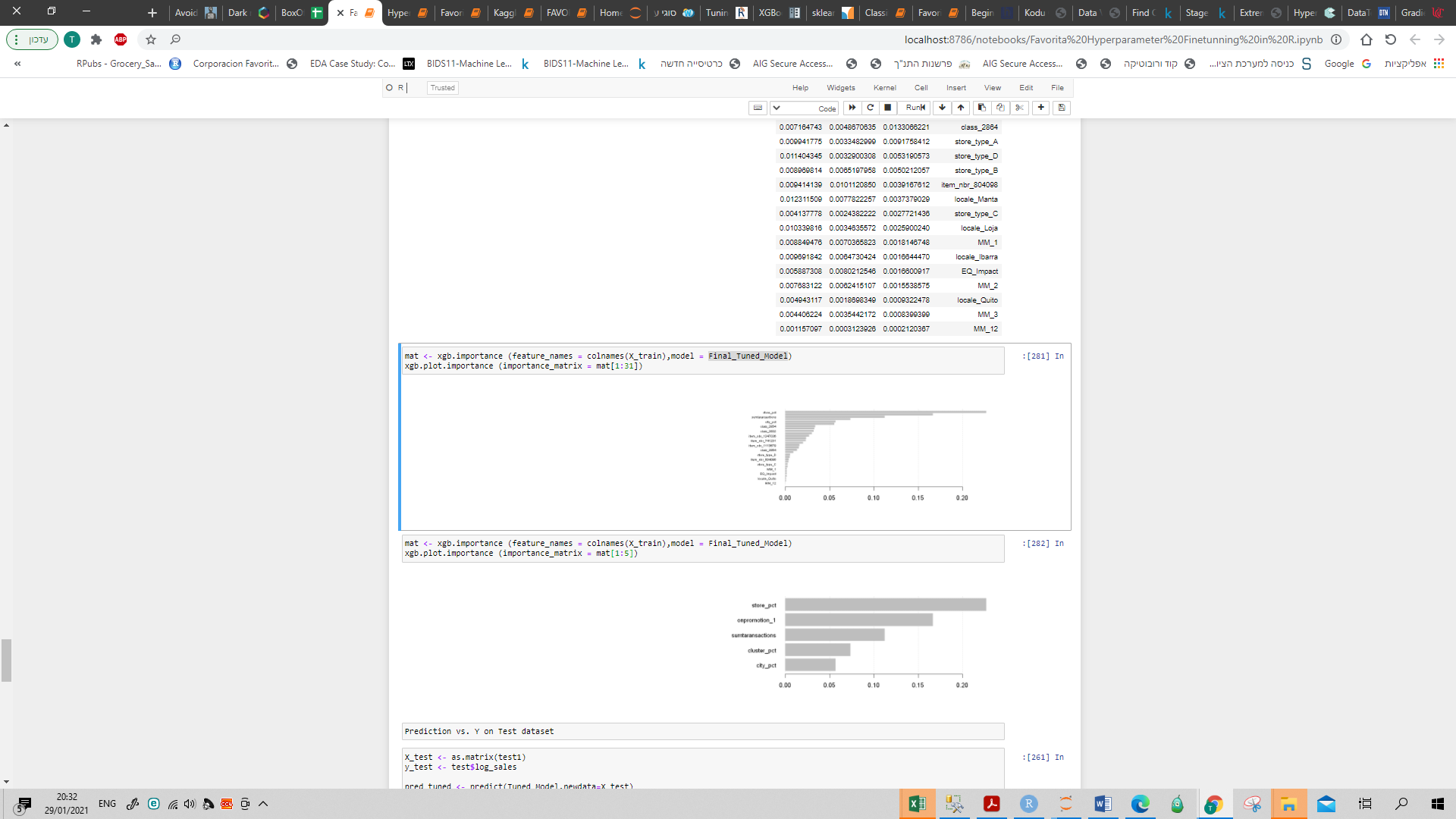
**Data Summary – Selected variables:**



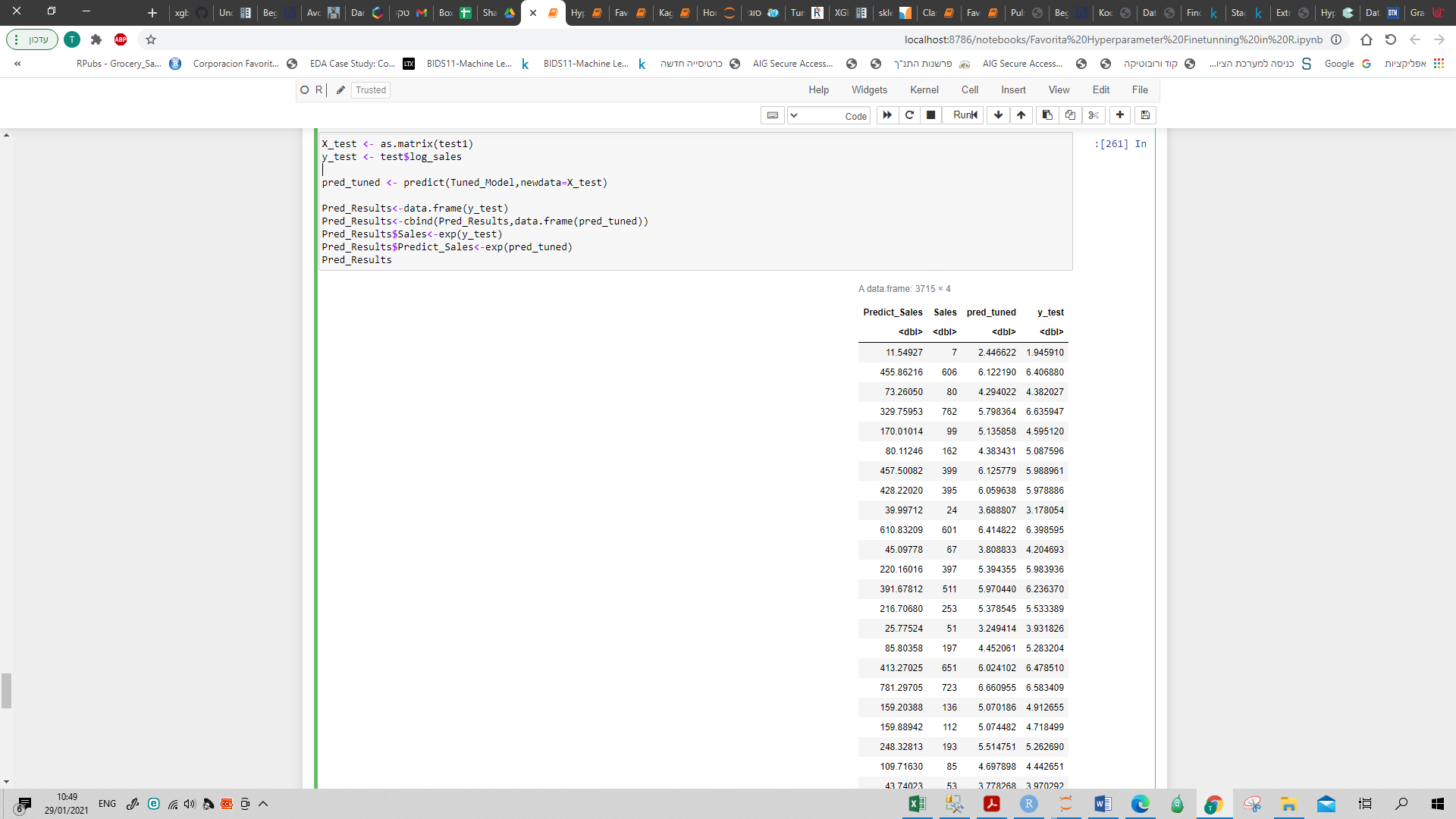
**Final Model:**



**Feature importance – full list:**



**Comparison of the predictive sales vs. sales column in the test dataset - example:**



# **XGBoost parameters for tree booster:**

**nrounds** - controls the maximum number of iterations. For classification, it is similar to the number of trees to grow

**max\_depth** - [default=6], range: [0,∞]

maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.

**colsample\_bytree** -  [default=1].  is the subsample ratio of columns when constructing each tree.

Subsampling occurs once for every tree constructed.

**eta** - [default=0.3, alias: learning\_rate], range: [0,1]

Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights

of new features, and eta shrinks the feature weights to make the boosting process more conservative.

**gamma** -  [default=0, alias: min\_split\_loss], range: [0,∞]

Minimum loss reduction required to make a further partition on a leaf node of the tree.

The larger gamma is, the more conservative the algorithm will be.

**min\_child\_weight** -  [default=1], range: [0,∞]

Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min\_child\_weight, then the building process will give up further partitioning. In linear regression task, this simply corresponds to minimum number of instances needed to be in each node.

The larger min\_child\_weight is, the more conservative the algorithm will be.

**subsample** - [default=1], range: (0,1]

Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting.

Subsampling will occur once in every boosting iteration.

**Working Files**

**Jupyter Notebooks:**

1. EDA- Favotita1.ipynb (R)
2. Feature selection (multivariate) - FAVORITA Fearture Selection.ipynb (Python)
3. Feature selection (univariate) - Favotita1.ipynb (R)
4. Model selection- Favorite Pre Processing and Modeling Include Train Metrics.ipynb (R)
5. Hyperparameters and fine-tuning- Favorita Hyperparameter Finetunning in R.ipynb (R)

**Excel:**

1. Test\_Pred\_Results - further checks
2. Monthly Prediction on the Test\_Pred\_Results\_All