Capstone report meshari aldossari

Machine Learning Engineer Nanodegree January 25th, 2022

Definition

Project overview:

Sales forecasting is one of the first known uses in machine learning and it helped in machine learning acceptance in the market as its importance is clear from a business standpoint being able to predict the sales for the future would help especially in the logistic related problems being able to predict sales should have a noticeable help in them as logistic can be prepared ahead of time and be ready to use.

not only is Sales forecasting important in the logistic department, but also in the finance department since it can help in knowing your approximate income from these sales which would lead to better financial related decisions.

Many companies reward employees for performance against forecast (or more precisely, targets which are derived from forecasts), using an asymmetric system where exceeding the forecast results in positive rewards whereas falling short results in a mixture of punishments and withholding of rewards [1].

problem statement:

the goal is to use time-series forecasting to forecast store sales on data from Corporación Favorita, a large Ecuadorian-based grocery retailer.

Current subjective forecasting methods for retail have little data to back them up and are unlikely to be automated. The problem becomes even more complex as retailers add new locations with unique needs, new products, ever-transitioning seasonal tastes, and unpredictable product marketing.

metrics:

since this is a regression problem the metric used is Root Mean Squared Logarithmic Error.

The Squared Logarithmic Error is calculated as:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i+1)-\log(\hat{y}_i+1))^2}$$

where:

- n is the total number of instances.
- $\hat{y}_i$ is the predicted value of the target for instance (i).
- y_i is the actual value of the target for instance (i).
- log is the natural logarithm.

Analysis

Data exploration:

the dataset consists of six files:

- train.csv: The training data, containing time series of the following features:
  - store_nbr: a categorical feature with 54 unique values for each store.
  - Family: the family of the sold product with 33 unique values.
  - Onpromotion: how much of each family in each store was in promotion.
  - the target sales: the sales of each family in each store.
- test.csv: The test data, having the same features as the training data but without the target feature which is sales.
- sample_submission.csv: A sample submission file in the correct format with two columns which is:
  ID, sales.
- stores.csv: Store metadata, including of the stores:
  city, state, type, and cluster.
- oil.csv: Daily oil price. which Includes values during both the train and test data timeframes. (Ecuador is an oil-dependent country and its economic health is highly vulnerable to shocks in oil prices.)
- holidays_events.csv: Holidays and Events, with metadata about them:
  type, locale, locale_name, description, transferred.

Exploratory Visualization:

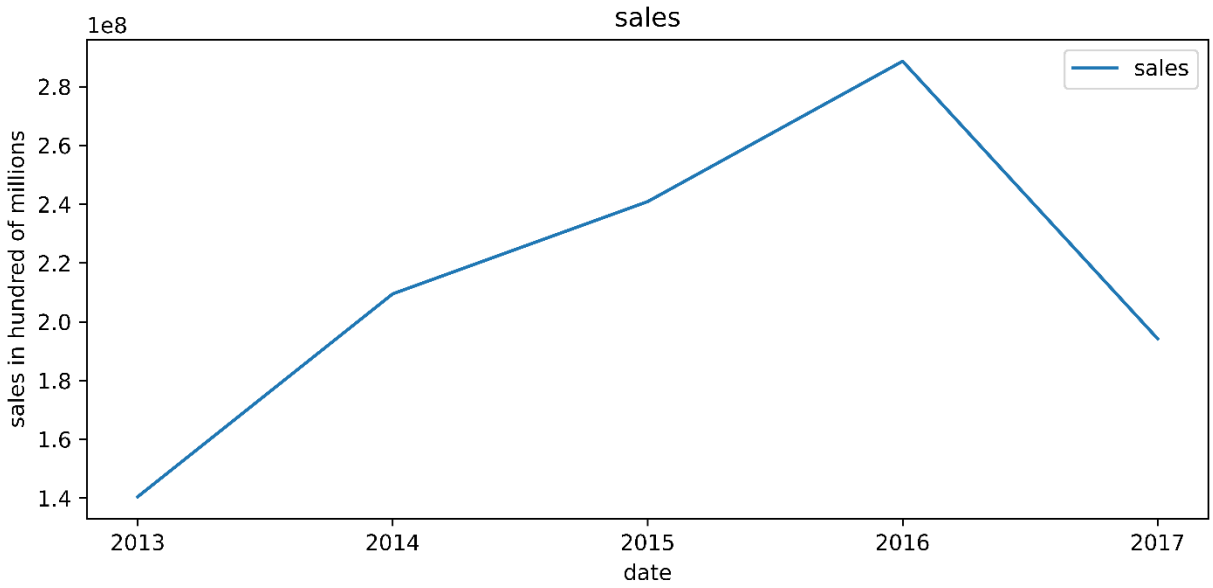


Figure 1: yearly sales

From the figure above we can see that the sales follow a clear upward trend from 2013 to late 2015, which then turned to downward trend for the remaining available training data, with this it's important to make sure that the training data is cut to start from at least 2016 or even later on to not include unnecessary noise of the older trend of the sales.

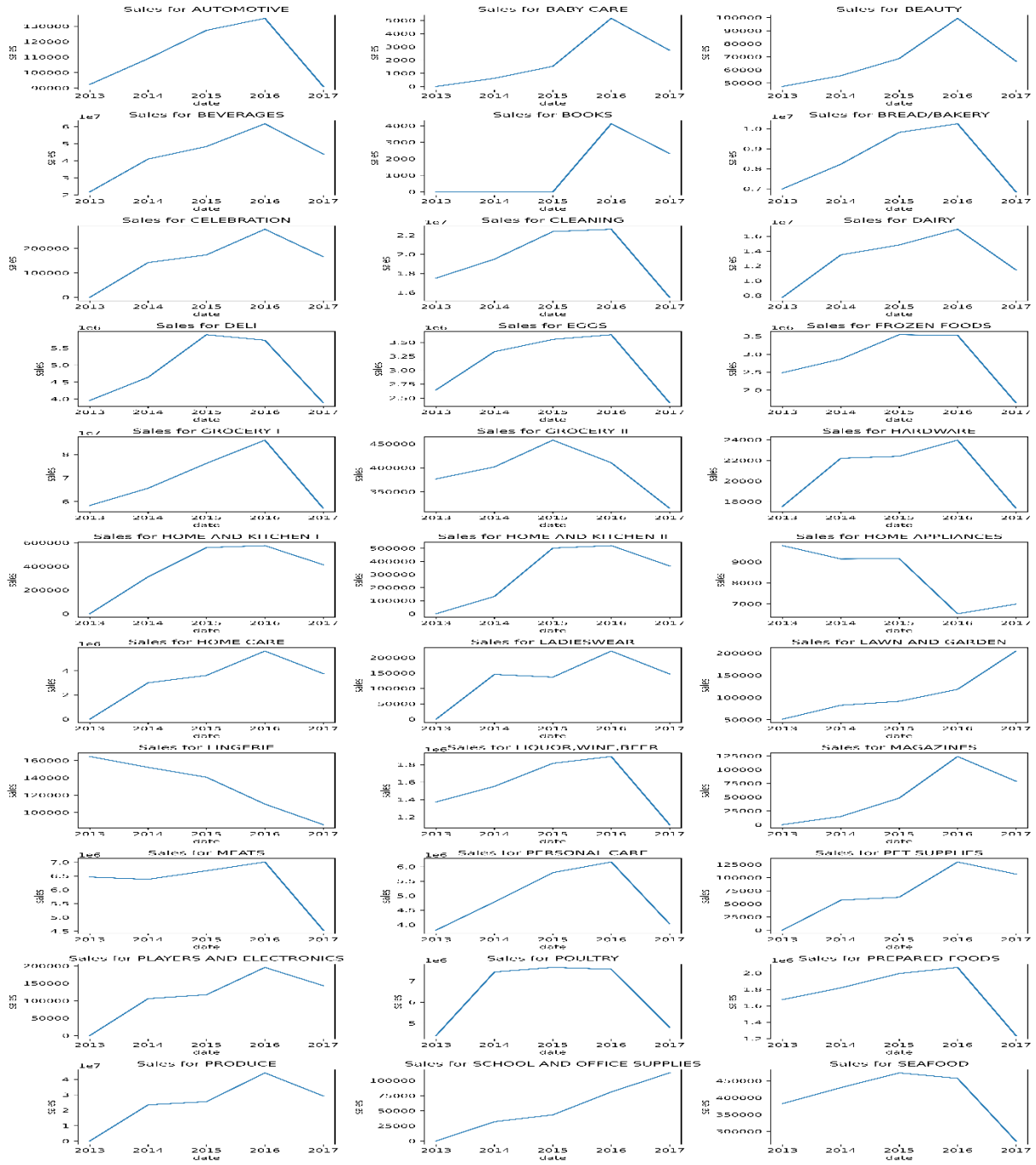


Figure 2: yearly sale by family

From the above figure of yearly average sale of each family it's clear that each family follow a different trend some are nearly complete opposite trends e.g. automotive, and home appliances.

This would make it hard for a model to fit all those different categories which would require a complex model for example the benchmark model linear regression wouldn't be able to accurately fit this data, unlike more complex models e.g. XGboost regressor.

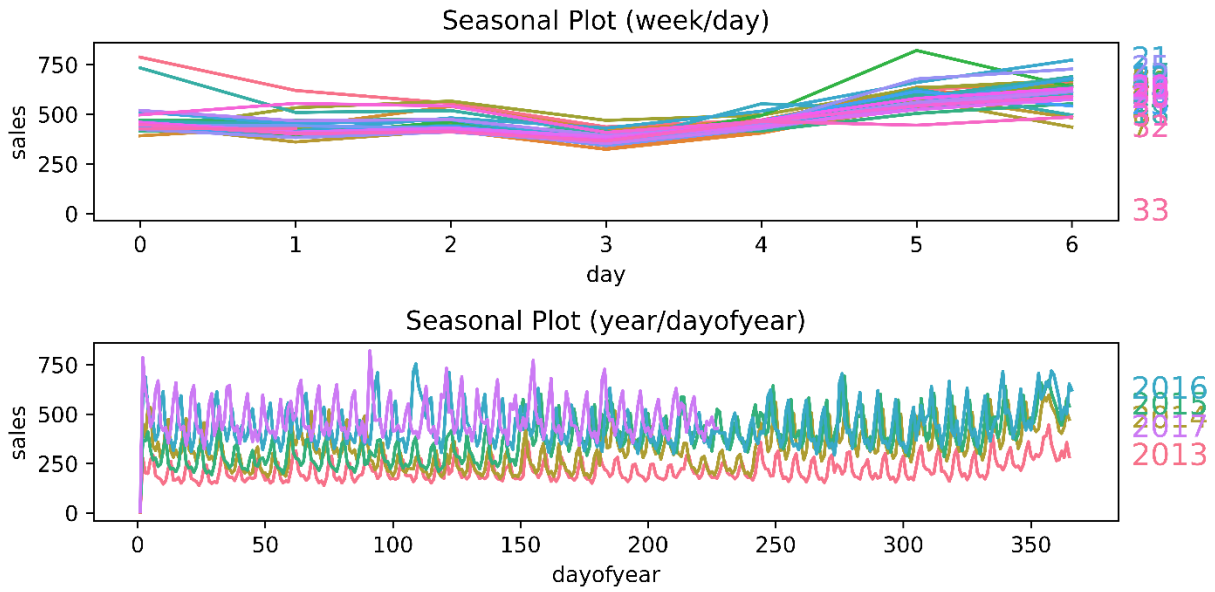


Figure 3: seasonality

For the last figure plots both the weekly seasonality and the yearly seasonality of the sales.

The weekly seasonality is very strong as it's shown by the graph most the weeks has decrease of sales at the start of the week which becomes an increase at the middle of the week.

There is also a strong yearly seasonality which is mostly comprised of fluctuating weekly and monthly seasonality over the years.

Algorithms and Techniques:

First, to capture seasonality and trend statsmodels Deterministic Process was used which

Create a trend variable with different specified order e.g., 2 will include both linear and quadratic terms, also possibly create a seasonal dummies to capture seasonality.

And combined with Deterministic Process is another statsmodels algorithm Calendar Fourier which is when used with Deterministic Process as additional terms can help better capture the seasonality.

Many different regression algorithms were used, so the details of each one of them wouldn't be mentioned but a brief summary of each one would be provided.

1- LinearRegression: fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation [2].
2- Lasso: Linear Model trained with L1 prior as regularizer (aka the Lasso) [2].

3- Ridge: Linear least squares with l2 regularization [2].
4- ElasticNet: Linear regression with combined L1 and L2 priors as regularizer [2].
5- SVR: is the same as support vector machine but is used for regression problems.
6- DecisionTreeRegressor: a Decision Tree algorithm for regression problems.
7- RandomForestRegressor: a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [2].
8- ExtraTreesRegressor: a meta estimator that fits several randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [2].
9- KNeighborsRegressor: The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set [2].
10- XGBRegressor: XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm.

benchmark:

the benchmark model is the simple model Linear Regression.

Methodology

Data Preprocessing:

Some initial preprocessing was done when loading the data e.g., setting the index, dealing with missing values, dropping unnecessary columns.

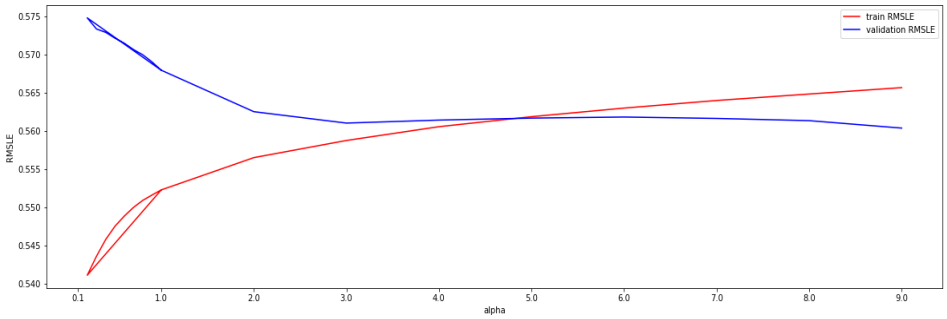
But most of the data preprocessing was done in the feature engineering part since it's a time series problem which needs special futures, the steps followed is the following:

1- Using statsmodels Deterministic Process, and Calendar Fourier, mentioned above in the Algorithms and Techniques section, to create seasonality futures.
2- Merging oil prices with seasonality futures and assigning the average oil price to dates with no recorded oil prices.
3- Merging holidays tokenized description with the result of the previous step and filling dates with no holidays with 0.
4- Splitting the data into training and validation with validation data being the last 15 days of the original training data so from 2017-08-01 to 2017-08-15

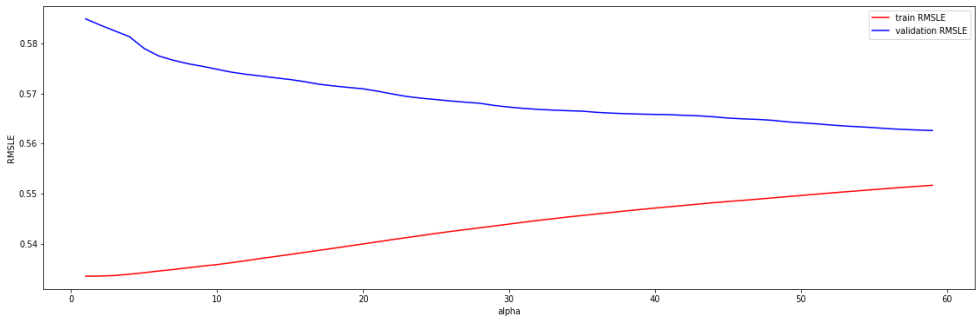
Refinement:

For the refinement of the mentioned algorithms some haven't been optimized (Linear Regression, SVR), the other 8 algorithms were optimized by tuning their parameters:

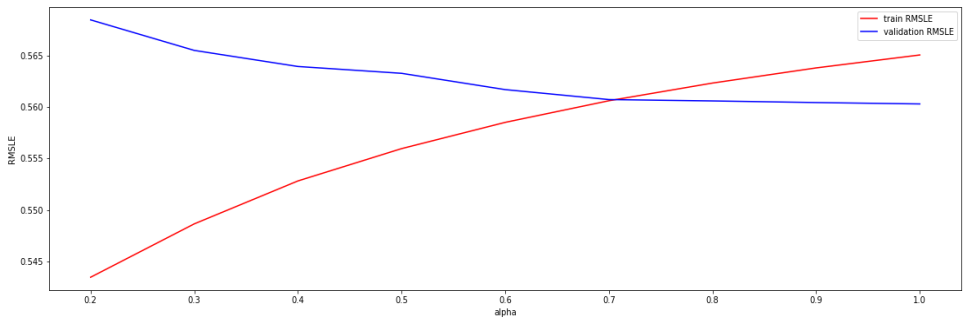
1- Lasso: the most important parameter which should impact the result of the model is alpha, which determine the l1 term, and from the graph below it's clear that an alpha of 3 is the most optimal value.



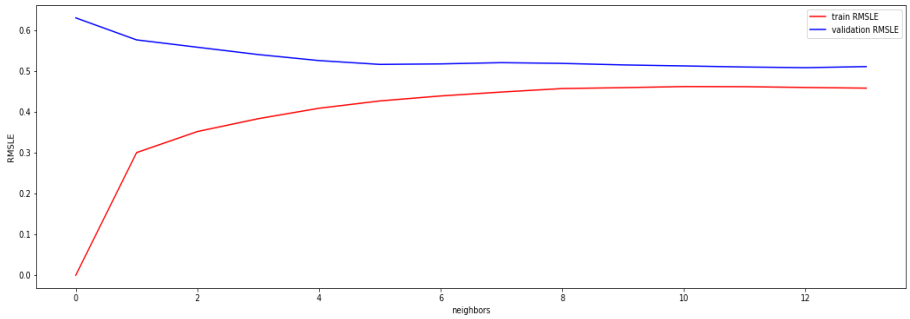
2- Ridge: like the lasso alpha is the parameter to pay the most attention to, alpha value of 50 seems to be optimal enough.



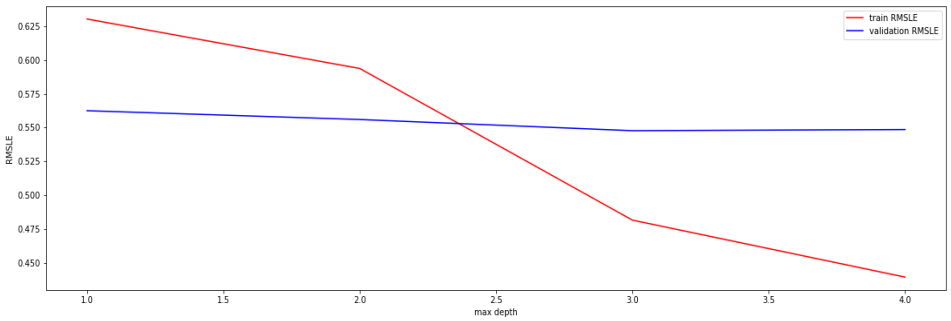
3- ElasticNet: combining both l1 and l2 terms, from the figure below we can see that 0.6 is the most optimal value for this model alpha.



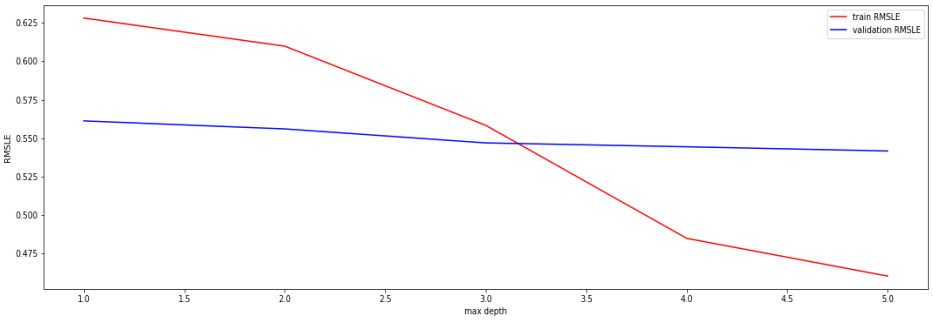
4- KNeighborsRegressor: the parameter to be tuned is n_neighbors, and from the figure below 10 is good value for this parameter.



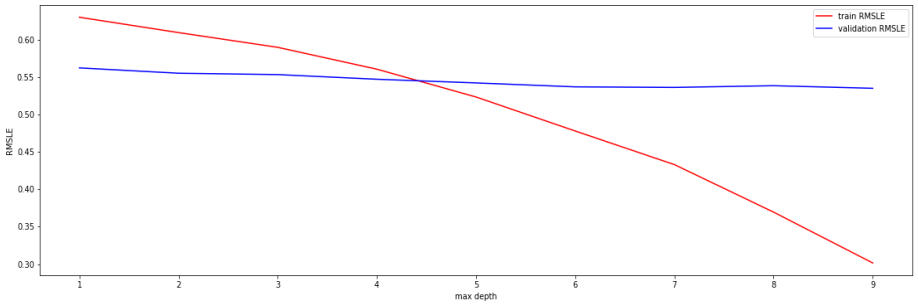
5- DecisionTreeRegressor: all tree-based model in this project was overfitted to the training data and so to limit this overfitting one of the most used parameter to limit tree-based models overfitting is max_depth, which from the figure below 3 is the most optimal value.



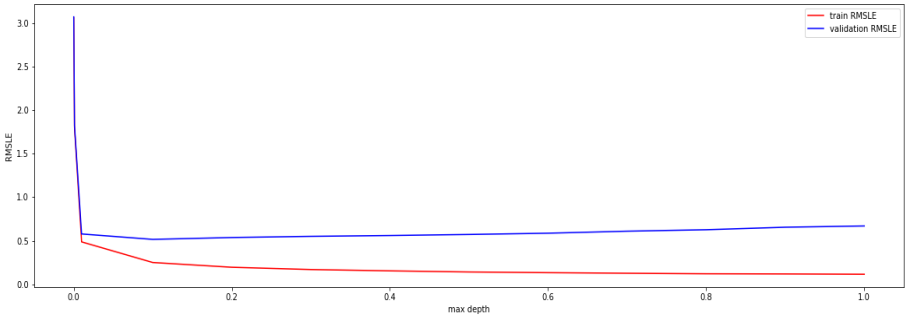
6- RandomForestRegressor: was over fitted as was mentioned above so max_depth was tuned to 4 based on the below figure.



7- ExtraTreesRegressor: the max_depth was tuned to 6 based on the below figure.



8- XGBRegressor: some parameter was manually chosen to limit overfitting, and learning_rate was tuned to 0.1 as clear from the below figure.



Implementation:

The implementation of the algorithms was done after the data preprocessing mentioned above which followed these steps:

1- Trying Linear Regression and calculating the training and validation of its prediction.
2- Implementing the other nine algorithms.
3- Tuning each algorithm to be well fitted.
4- Comparing the models results.
5- Choosing the best 3 models.
6- Testing the best 3 models with the test set.

Results

Justification:

	Training RMSLE	Validation RMSLE	Testing RMSLE
Benchmark model: LinearRegression	**0.28**	**0.51**	**0.45554**
Final model: XGBRegressor	**0.53**	**0.58**	**0.49920**

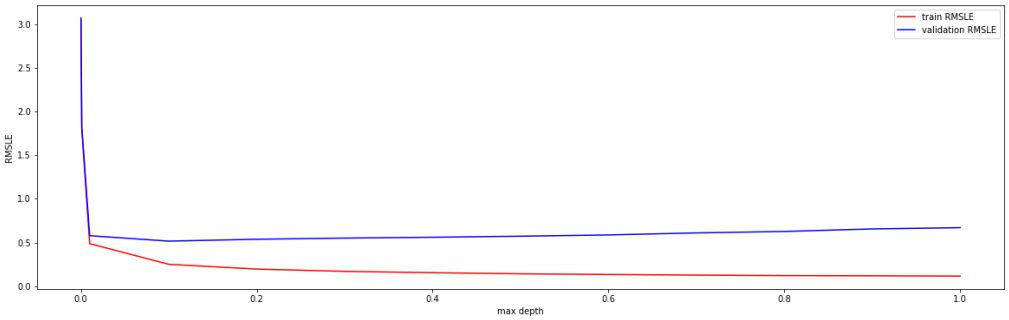
It's quite clear that the final model XGBRegressor has outperformed the benchmark model LinearRegression in all three scores of training, validation, testing.

The final model does good job of accurately forecasting the sales although not the exact amount but a close one that could benefit the stores to at least forecast days for an overall large sale, a specific family large or small sale, and many other ways that this model can be used for.

Model Evaluation and Validation:

The final model XGBRegressor parameters is as following:

- gamma = 5: to minimize overfitting the value was chosen after manually testing a few possible values with 5 performing the best.
- max_depth = 3: to prevent overfitting like all tree-based models this can be realized by limiting the depth of the tree and 3 was the best value to balance the fit of the data.
- n_jobs = -1: this parameter is just to use all available threads of the device when building the model to save time.
- learning_rate = 0.1: this was chosen after manually choosing the above parameter by testing a few possible values saving each value and it's training and validation RMSLE and then plotting them.



the validation process was done twice first with a validation split when building the models, and lastly with the test set as an extra measure of the model robustness.

References:

1. Fildes, Robert, et al. "Researching sales forecasting practice: Commentaries and authors' response on "Conducting a Sales Forecasting Audit" by MA Moon, JT Mentzer & CD Smith." International Journal of Forecasting 19.1 (2003): 27-42.
2. Scikit learn documentation, https://scikit-learn.org/stable/index.html.
3. https://www.kaggle.com/learn/time-series