

# **Store Sales - Time Series Forecasting (Kaggle)**

Fall 2021 CMPE 257 Project Report (Group 4), SJSU

Amol Sadasivan, Atanu Ghosh, Janaarthana Harri Palanisamy, Prabjyot Obhi, and Sabrina Dhaliwal

## **Introduction**

For this Kaggle competition, we were given data regarding store sales in Ecuador and supplementary information which could provide insight on whether people were purchasing items such as the price of oil, holidays, etc. The data was provided from Corporación Favorita, a large Ecuadorian-based grocery retailer.

The data provider is most definitely reputable as the Favorita Corporation and its commercial, industrial and real estate subsidiaries have a strong presence throughout the country. Its different lines of business and formats allows them to adapt their products, services and experiences offering to the local realities. Internationally, the Corporation's subsidiaries have activities in six countries in the region, in addition to the ones in Ecuador.

We'll build a model that more accurately predicts the unit sales for thousands of items sold at different Favorita stores. This has a real world impact for grocery stores as more accurate forecasting can decrease food waste related to overstocking and improve customer satisfaction.

## **Literature Review/Related Work**

It is difficult to find literature that relates exactly to a project statement, so we had to look for more generalized papers with similar problem statements that retained key components. The piece of literature that would fit our project exactly would be something along the lines of "Grocery Store Sales Predictive Model," and it is easy to understand why this does not exist—it's far too specific.

There were a handful of papers that were focused on predicting stock prices and the changes that they would face given their history. Schöneburg's (1990) intent was to use neural networks as a way to accurately model stock prices and their changes. Based on his results, he anticipated that neural networks would be able to successfully be applied to semi-chaotic time series.

Similarly, Ariyo et al. (2014) wanted to focus their research on predicting the values of varying stock prices. However, they utilized the autoregressive integrated moving average model (ARIMA). The data that they used was collected from the New York Stock Exchange and the

Nigerian Stock Exchange. Ariyo et al. (2014), similar to Schöneburg's (1990), believe that the model can genuinely compete with the existing infrastructure that predicts the prices of stock.

Initially we thought these papers would be a good guide to utilize for our project, but we quickly realized they primarily focused on two models that would be used for, as Schöneburg(1990) stated, “semi-chaotic” time series. The data being used in our project and their research are inherently different. Stock prices are extremely volatile throughout the day, week, and/or month, whereas grocery store sales are much more stable. On an extremely high level, people most likely buy groceries once a week or once every two weeks whereas the prices of stock vary drastically on an intraday level.

This led us to find literature that was not focused on stock prices due to the volatility, and we found Nguyen's (2018) work regarding the impact of housing characteristics on the price of the house. The data that she used was composed of information from Zillow, Redfin, and Trulia. Housing data is much more similar to the data in our project than the previous two papers that analyzed the stock market. Her work was also less focused on creating and analyzing the results of one model in particular, whereas Schöneburg (1990) and Ariyo et. al (2014) were focused on Neural Networks and ARIMA, respectively.

## Data Exploration:

The training data is composed of features **store\_nbr**, **family**, **onpromotion**, and the target feature--**sales**.

For this particular file the **store\_nbr** identifies the stores at which the products are sold, the **family** feature indicates whether the type of product sold was a family product, the **onpromotion** field give the total number of items in a product family that were being promoted at a store on a given date, and **sales** gives the total amount of sales for a product family at a particular grocery store on a given date.

The test dataset has the same featureset as the training dataset, and each feature in the testing dataset represents the same thing as in the training dataset; however, this is where we will need to be predicting the sales values.

The supplementary datasets that were given to us were **stores.csv**, **oil.csv**, and **holidays\_events.csv**. The stores file included store metadata such as **city**, **state**, **type**, and **cluster**. In the dataset, the cluster feature is simply just a grouping of similar stores. **Oil.csv** contains the daily oil price, and this was important as Ecuador is an oil-dependent country and its economical health is highly vulnerable to shocks in oil prices. Finally, the **holiday\_events.csv** lists the holidays and events that take place along with their metadata.

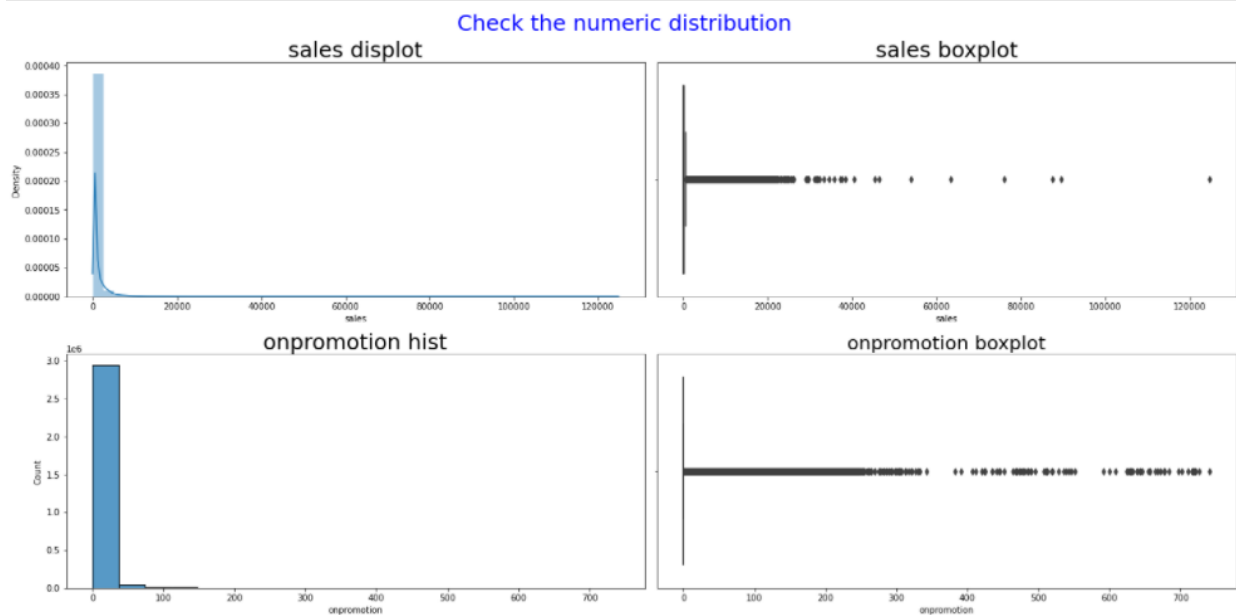
Below are images regarding each dataset:

**Train:**

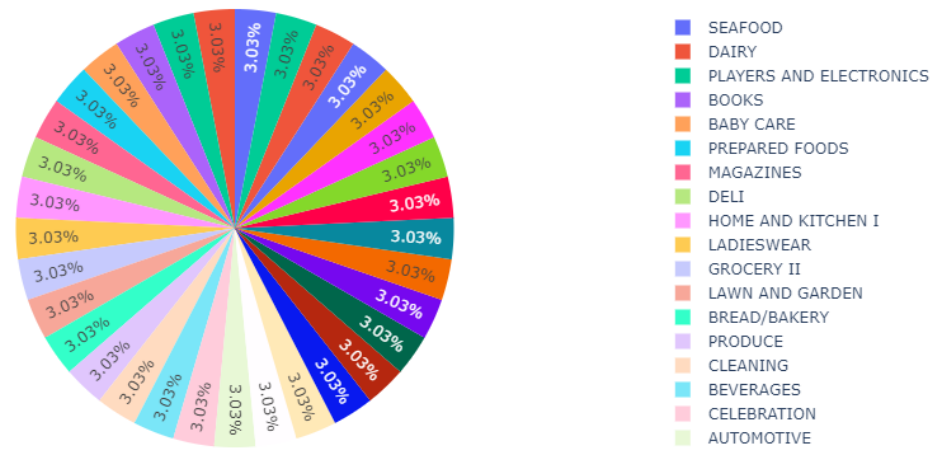
---

Shape : (3000888, 6)

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	id	int64	0	3000888	0	1	2
1	date	object	0	1684	2013-01-01	2013-01-01	2013-01-01
2	store_nbr	int64	0	54	1	1	1
3	family	object	0	33	AUTOMOTIVE	BABY CARE	BEAUTY
4	sales	float64	0	379610	0.0	0.0	0.0
5	onpromotion	int64	0	362	0	0	0



- No missing values
- We could see there are outliers in sales column
- Date column seems to be in object type, need to parse the date to datetime format for time series models.



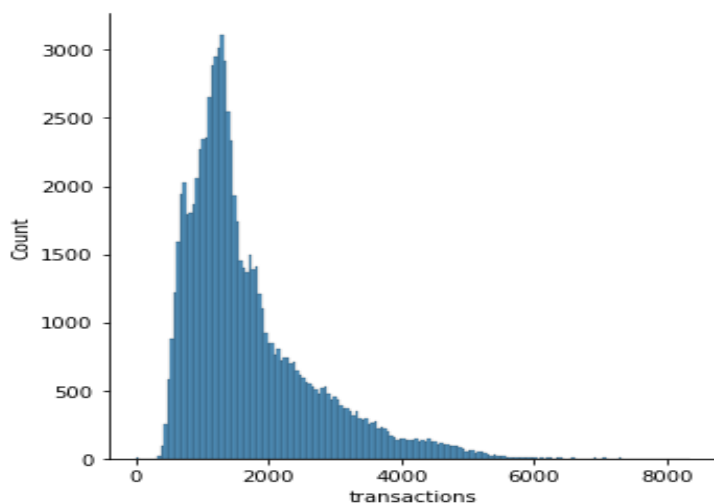
- Percentage of components in the family columns are the same. The dataset seems to be balanced.

## Transactions:

Shape : (83488, 3)

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	date	object	0	1682	2013-01-01	2013-01-02	2013-01-02
1	store_nbr	int64	0	54	25	1	2
2	transactions	int64	0	4993	770	2111	2358

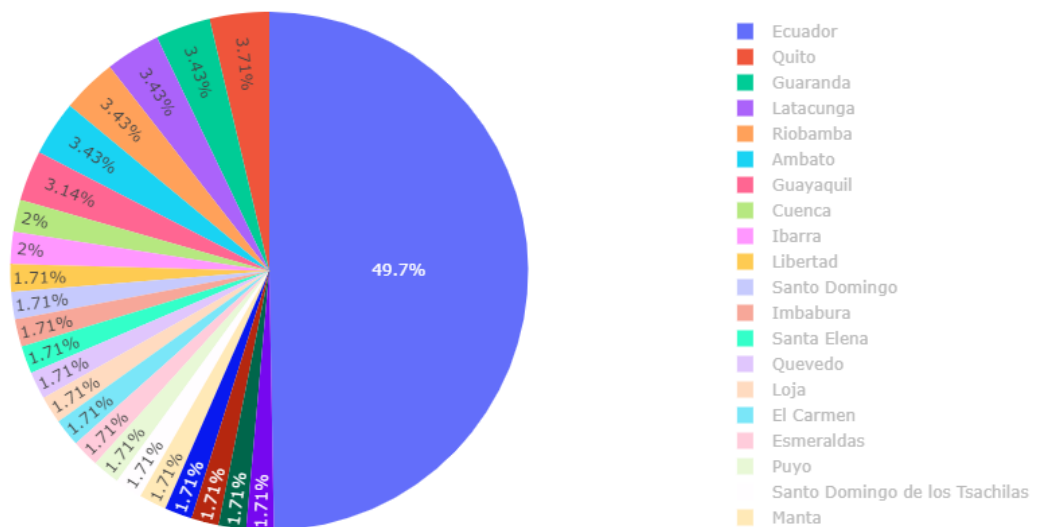
<seaborn.axisgrid.FacetGrid at 0x19429cf8790>



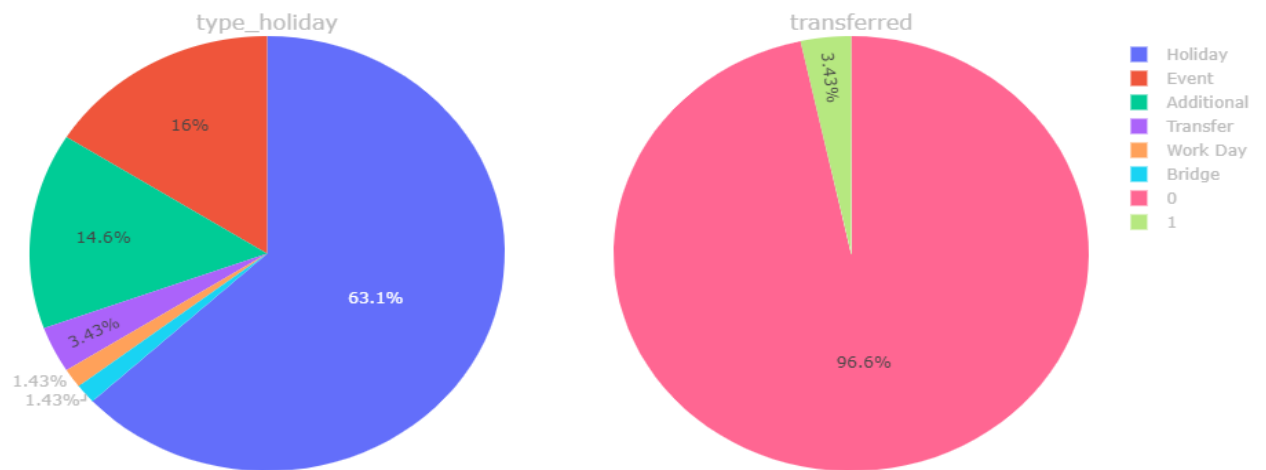
## Holidays:

Shape : (350, 6)

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	date	object	0	312	2012-03-02	2012-04-01	2012-04-12
1	type	object	0	6	Holiday	Holiday	Holiday
2	locale	object	0	3	Local	Regional	Local
3	locale_name	object	0	24	Manta	Cotopaxi	Cuenca
4	description	object	0	103	Fundacion de Manta	Provincializacion de Cotopaxi	Fundacion de Cuenca
5	transferred	bool	0	2	False	False	False



- No missing values
- It is imbalance in transactions when Ecuador took up nearly a half.

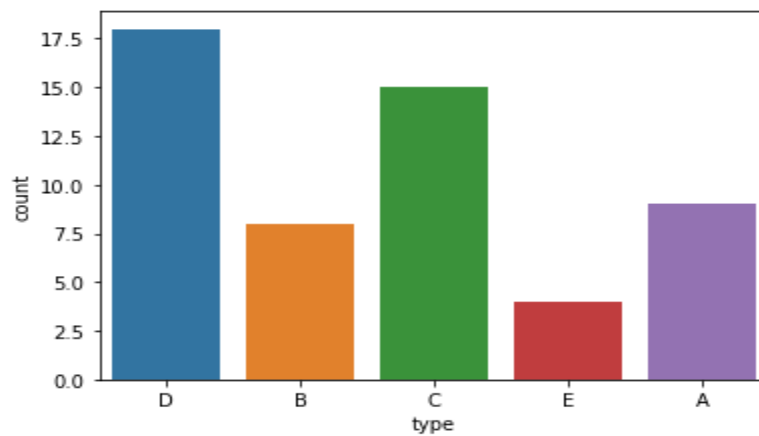


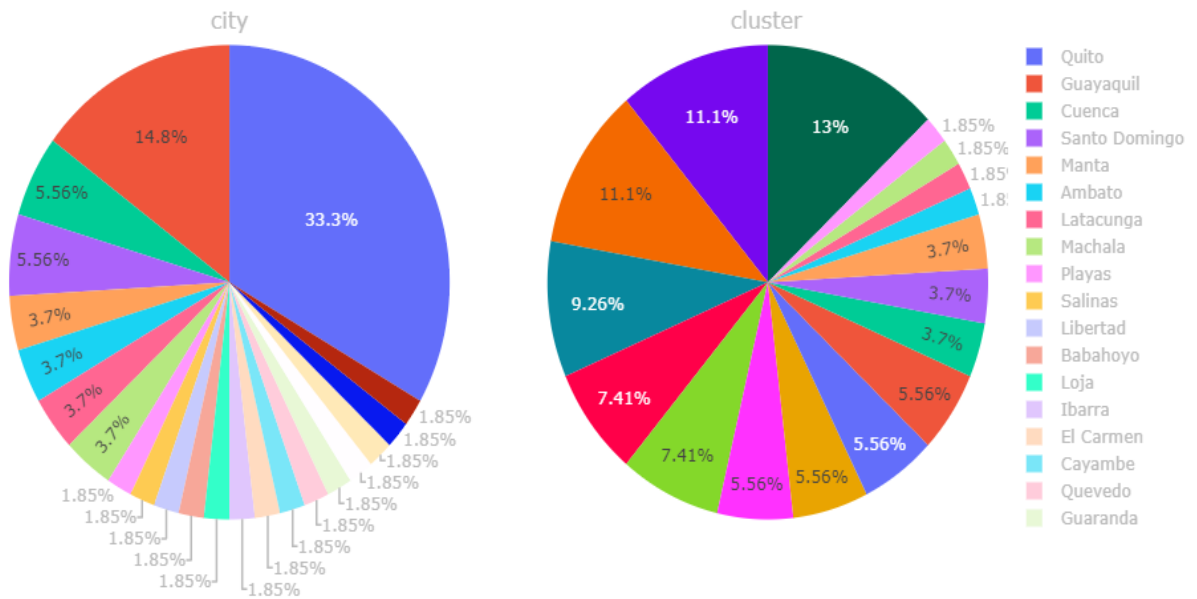
## Distribution of holiday types

Stores:

Shape : (54, 5)

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	store_nbr	int64	0	54	1	2	3
1	city	object	0	22	Quito	Quito	Quito
2	state	object	0	16	Pichincha	Pichincha	Pichincha
3	type	object	0	5	D	D	D
4	cluster	int64	0	17	13	13	8





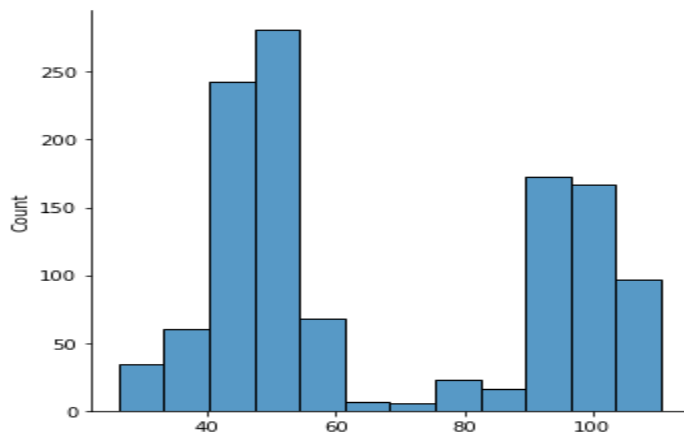
## Distribution of stores

## Oil:

Shape : (1218, 2)

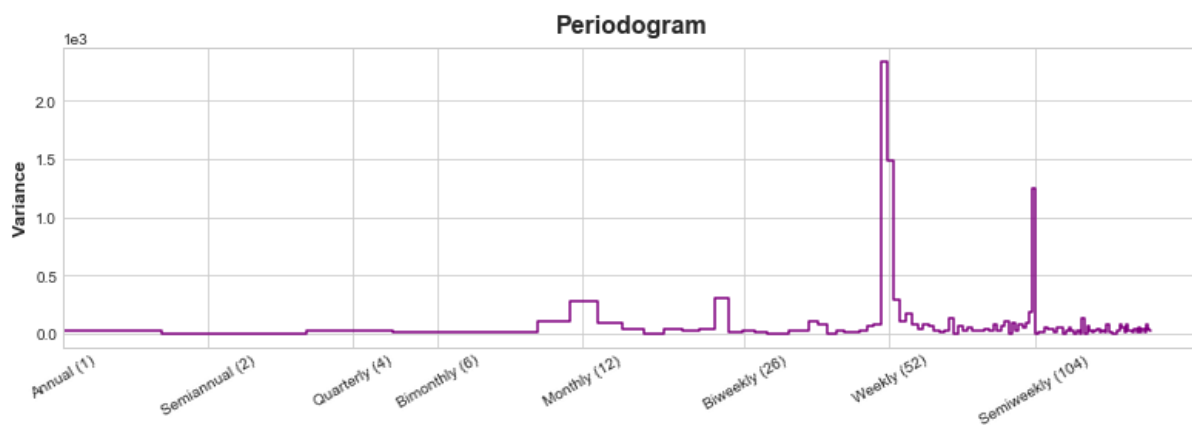
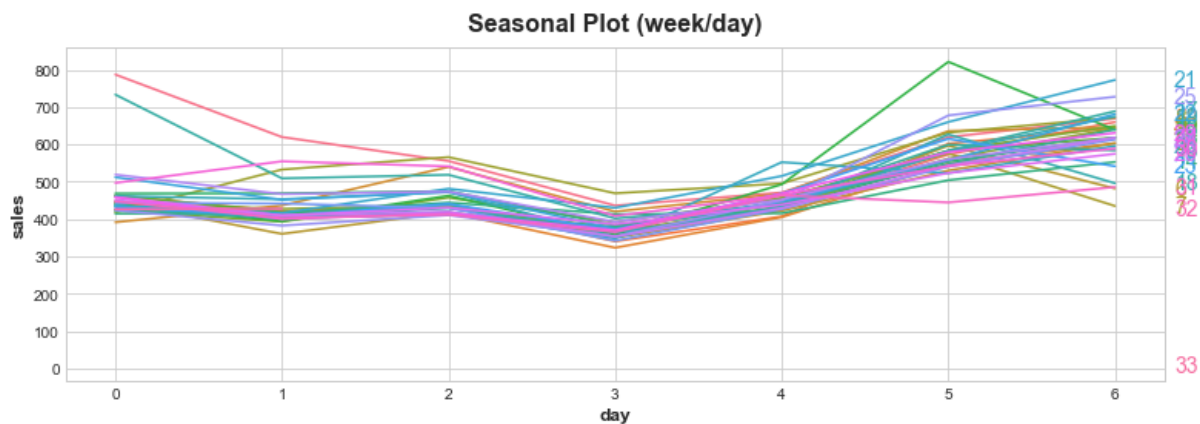
	Feature	DataType	null count	unique count	First value	Second value	Third value
0	date	object	0	1218	2013-01-01	2013-01-02	2013-01-03
1	dcoilwtico	float64	43	998	NaN	93.14	92.97

<seaborn.axisgrid.FacetGrid at 0x1942a583d30>

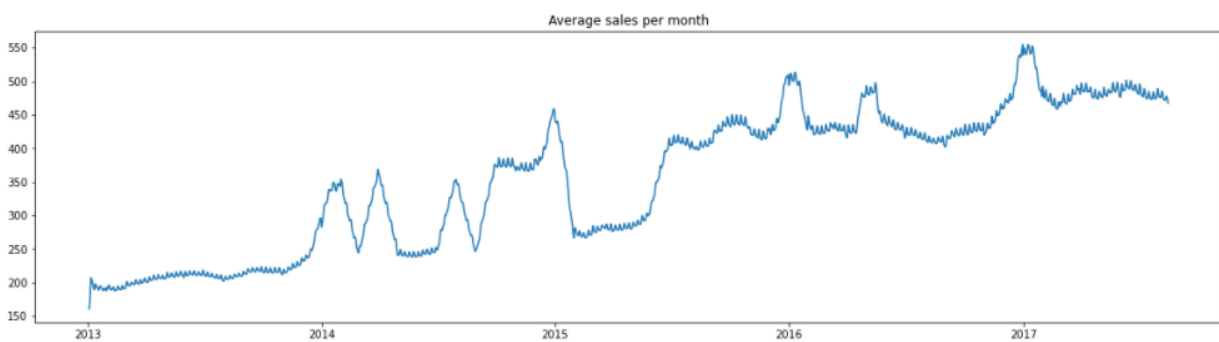


- "dcoilwtico" has null values.

After Merge and preprocessing:

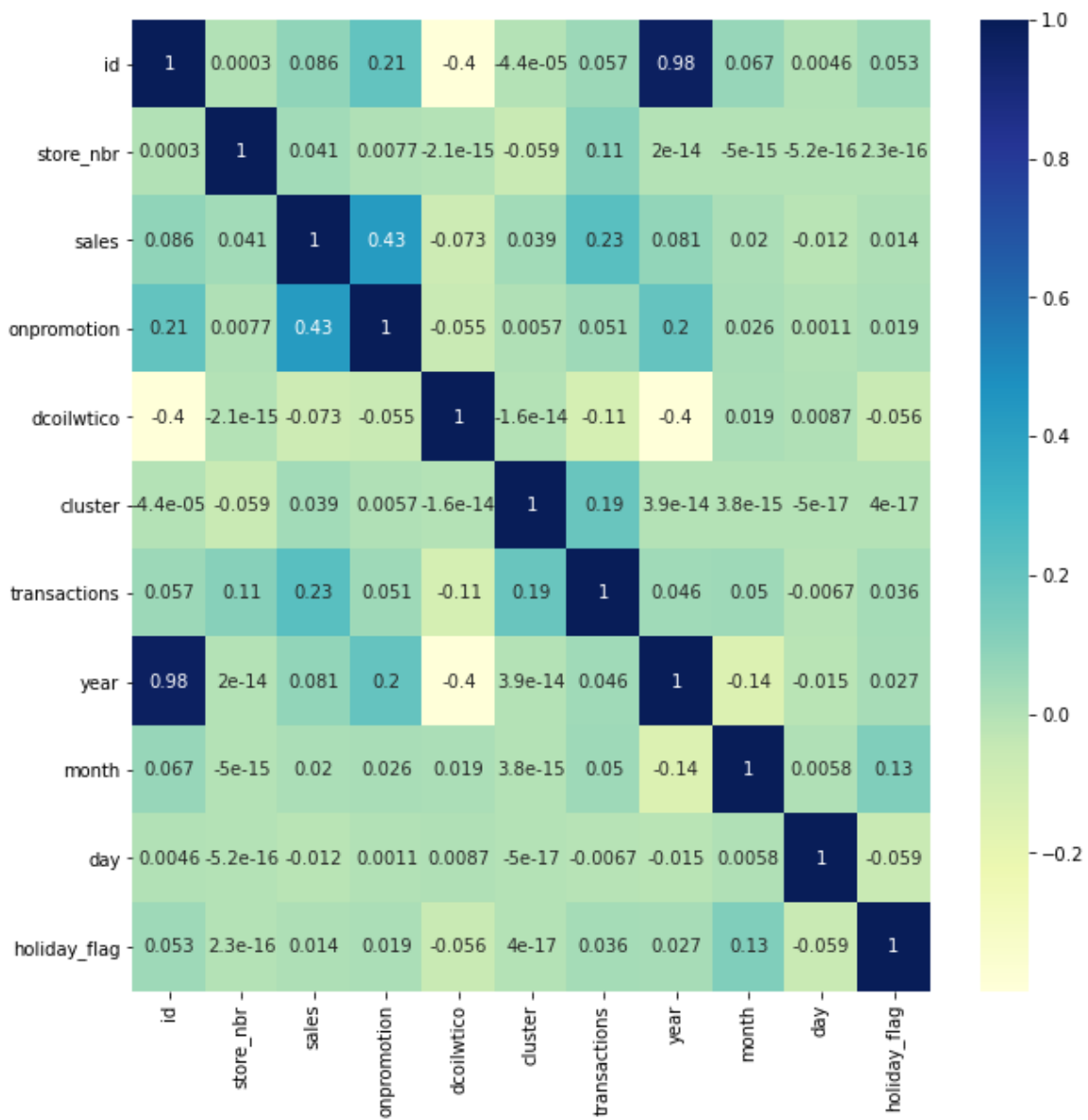
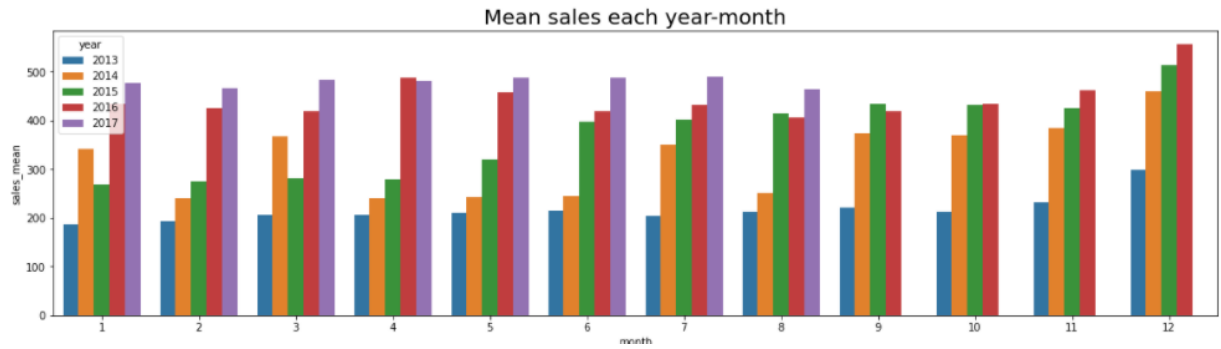


- Both the seasonal plot and the periodogram suggest a strong weekly seasonality.
- From the periodogram, it appears there may be some monthly and biweekly components as well.
- In fact, the notes to the Store Sales dataset say wages in the public sector are paid out biweekly, on the 15th and last day of the month -- a possible origin for these season



- Upward trend on avg sales.





- Key positive correlations: id-year, sales-onpromotion

## Data Preprocessing:

Shape : (3054348, 19)

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	id	int64	0	3000888	0	1	2
1	date	object	0	1684	2013-01-01	2013-01-01	2013-01-01
2	store_nbr	int64	0	54	1	1	1
3	family	object	0	33	AUTOMOTIVE	BABY CARE	BEAUTY
4	sales	float64	0	379610	0.0	0.0	0.0
5	onpromotion	int64	0	362	0	0	0
6	dcoilwtico	float64	878526	995	0.0	0.0	0.0
7	type_x	object	2551824	6	Holiday	Holiday	Holiday
8	locale	object	2551824	3	National	National	National
9	locale_name	object	2551824	24	Ecuador	Ecuador	Ecuador
10	description	object	2551824	101	Primer dia del ano	Primer dia del ano	Primer dia del ano
11	transferred	float64	2551824	2	0.0	0.0	0.0
12	city	object	0	22	Quito	Quito	Quito
13	type_y	object	0	5	D	D	D
14	cluster	int64	0	17	13	13	13
15	transactions	float64	249117	4993	NaN	NaN	NaN
16	year	int64	0	5	2013	2013	2013
17	month	int64	0	12	1	1	1
18	day	int64	0	31	1	1	1

- After merging the datasets we have null values.
- Using fillna with 0 to fill numeric columns.
- There were several holiday events per year, so encoding 1 if occurs holiday else 0

	Feature	DataType	null count	unique count	First value	Second value	Third value
0	id	int64	0	3000888	0	1	2
1	date	object	0	1684	2013-01-01	2013-01-01	2013-01-01
2	store_nbr	int64	0	54	1	1	1
3	family	object	0	33	AUTOMOTIVE	BABY CARE	BEAUTY
4	sales	float64	0	379610	0.0	0.0	0.0
5	onpromotion	int64	0	362	0	0	0
6	dcoilwtico	float64	0	995	0.0	0.0	0.0
7	locale	object	2551824	3	National	National	National
8	description	object	2551824	101	Primer dia del ano	Primer dia del ano	Primer dia del ano
9	city	object	0	22	Quito	Quito	Quito
10	stores_type	object	0	5	D	D	D
11	cluster	int64	0	17	13	13	13
12	transactions	float64	0	4994	0.0	0.0	0.0
13	year	int64	0	5	2013	2013	2013
14	month	int64	0	12	1	1	1
15	day	int64	0	31	1	1	1
16	holiday_flag	int64	0	2	1	1	1

- Null value counts after fillna.

Transforming the data for model predictions:

1. Creation of calendar from 2013 to 2017 August 31st:

We created an empty dataframe named calendar with an index ranging from Jan 2013 till Aug 2017 for preprocessing and merging various datasets into one.

- a. Moving Average for Oil

Here we created a moving average for the oil dataset and took a rolling mean for the period of seven days and stored it as a different column value `df_oil['ma_oil']`.

Then we merged the oil dataset into our calendar dataset and used the `ffill` method to fill null values. 'ffill' stands for 'forward fill' and will propagate the last valid observation forward.

```
calendar['ma_oil'].fillna(method='ffill', inplace=True)
```

- b. Creating a variable day of the week

```
calendar['dofw'] = calendar.index.dayofweek
```

Here we created a new column with values corresponding to the day of week (i.e 0-6)

- c. Workday Event data

First we loaded the holiday event dataset into a dataframe and merged its columns with our calendar dataset. Next we created a new column for workday and classified the values greater than 4 in the dofsw (day of week) column as false and the rest as true.

```
calendar['wd'] = True  
calendar.loc[calendar.dofw > 4, 'wd'] = False
```

## Problem Formulation/Model Selection

Determining what the problem statement is asking you to predict is crucial in determining what type of model needs to be built. The problem statement along with the feature types tend to direct developers to certain models while simultaneously steering developers away from other models. In our scenario, the project was based around predicting a numerical value, so we knew we were not going to use any clustering or classification models.

When it came down to selecting models to which algorithms we wanted to create, we had to choose the regressor versions of the model. The most basic model that we started with was a simple Linear Regression Model. We chose to use this because it was one of the more basic models that would allow us to start the project quickly.

We also chose to use **xgboost Regressor, RandomForest Regressor, Ridge Regression, and svm regressor** so we would be able to see how each model would do when compared to one another. We needed to use the regressor, similar to how we used xgboostregressor in lab two, as the data is numerical. This also indicated to us that we were not going to be able to use accuracy, F1 score, precision, or recall as a means of quantifying how well our model performed as those are metrics used for classification.

When we were creating the SVM Regression model, we faced an issue regarding the time it would take to complete the training process. The notebook was left alone for three hours, and the process had not completed. Shortly after, the kernel had restarted, so we simply scrapped that model.

## Data Modeling

### 1. Data Splitting:

- a. The data from the train.csv and test.csv was loaded into a dataframe and each of the column values were assigned a data type

```
dtype={'store_nbr': 'category', 'family': 'category', 'sales':  
'float32'}
```

- b. The training data was divided into X and y for model fitting. The y dataframe was first pivoted using the unstack() function for the columns 'store\_nbr', 'family'.

```
y = df_train.unstack(['store_nbr',  
'family']).loc[start_date:end_date]
```

The unstack function pivots a level of the (possibly hierarchical) row index to the column axis, producing a reshaped dataframe with a new innermost level of column labels.

- c. For X First, we initialize a deterministic process with index=y.index, constant=False, a linear time trend and a fourier component, . The in\_sample method returns the full set of values that match the index.

```
dp = DeterministicProcess(index=y.index, constant=False, order=1, seas  
onal=False, additional_terms=[fourier], drop=True)  
  
X = dp.in_sample()
```

Deterministic processes simplify creating deterministic sequences with time trends or seasonal patterns. They also provide methods to simplify generating deterministic terms for out-of-sample forecasting.

Next the values of columns from the calendar dataframe were copied into X.

### 2. Applying Ridge Regression:

A ridge regression model was designed with alpha= 0.5 (This alpha values was selected after testing various alpha values ranging from 0.1 to 0.9 )

```
Ridge(fit_intercept=True, solver='auto', alpha=0.5, normalize=True)
```

This model was trained using the X,y dataframe and yielded an RMSE of 0.434

### 3. Applying Random Forest Regression

A Random Forest Regression model was designed with `n_estimators = 300`.

```
RandomForestRegressor(n_estimators = 300, n_jobs=-1, random_state=1)
```

This model on training yielded an RMSE of 0.445

### 4. Custom Regression Function

After seeing the results using both of the above regression models a custom regression model was created using a combination of Ridge Regression and Random Forest Regression models.

The Ridge model yielded better rmse for all columns except for 'SCHOOL AND OFFICE SUPPLIES'. Thus for this column alone a Random Forest Regression model was selected for training.

Joblib library's parallel function was used to create a model pipeline. This model yielded a slightly better result of RMSE: 0.42.

We have used this model for our final submission. As of today our submission ranks at 60th out of the 837 submissions

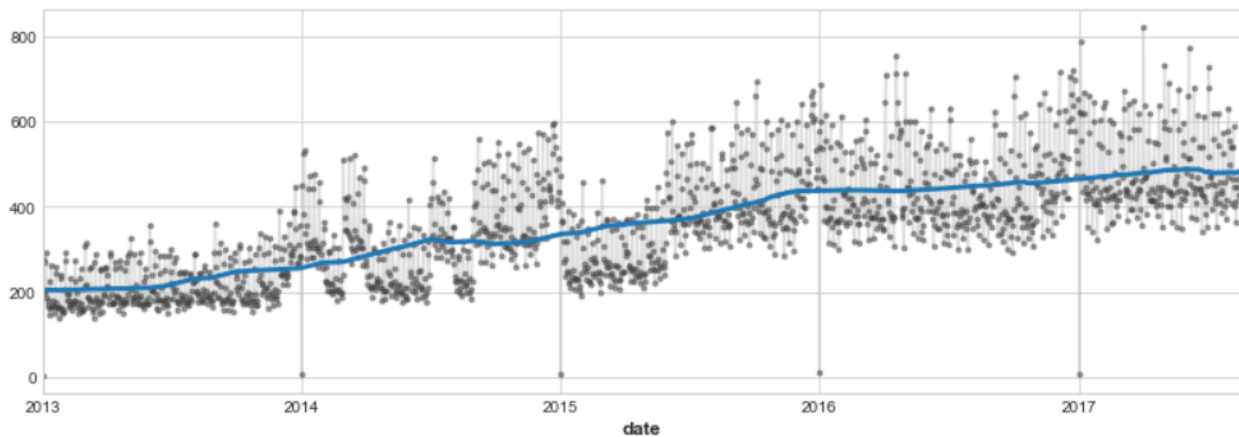
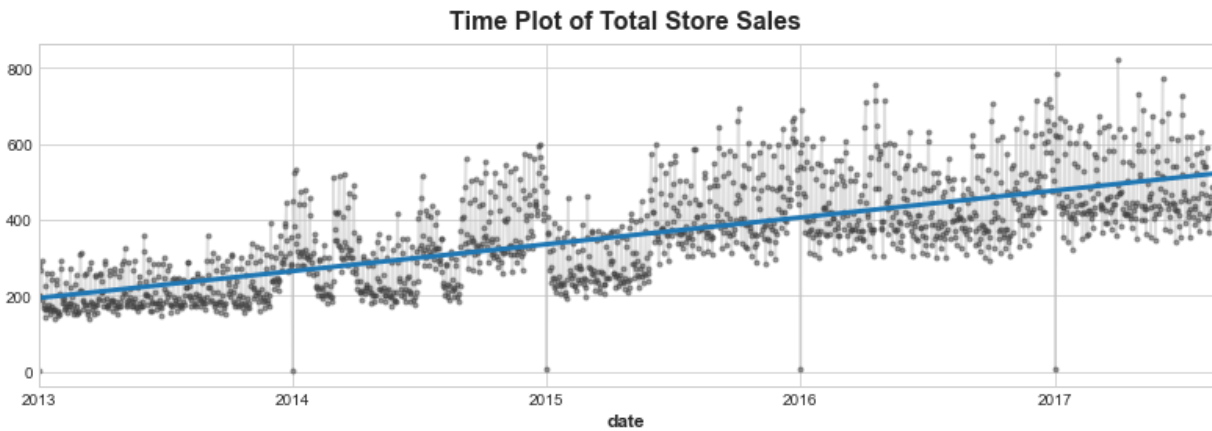
## Evaluation, Result Analysis, and Visualizations

Regression predictive modelling is the task of approximating a mapping function from input variables to a continuous output variable. Regression is different from classification, which involves predicting a category or class label. Accuracy is a measure for classification, not regression. The skill and performance of the regression model must be reported as an error in those predictions. If you are predicting a numeric value like a height or dollar amount, you don't want to know the model predicted the value exactly; instead, we want to know how close the predictions were to the expected values. Error addresses this and summarizes on average how close predictions are to their expected values.

We have taken the three error metrics that are commonly used for evaluating and reporting the performance of a regression model; **rmse**, **mae** and **rmsle**. The evaluation metric in the kaggle competition is Root Mean Squared Logarithmic Error (RMSLE).

## Linear Regression:

We used linear regression algorithm to construct forecasting models since it is widely used and adapts naturally to even complex forecasting tasks. We started with a basic model with time dummies and lag features.



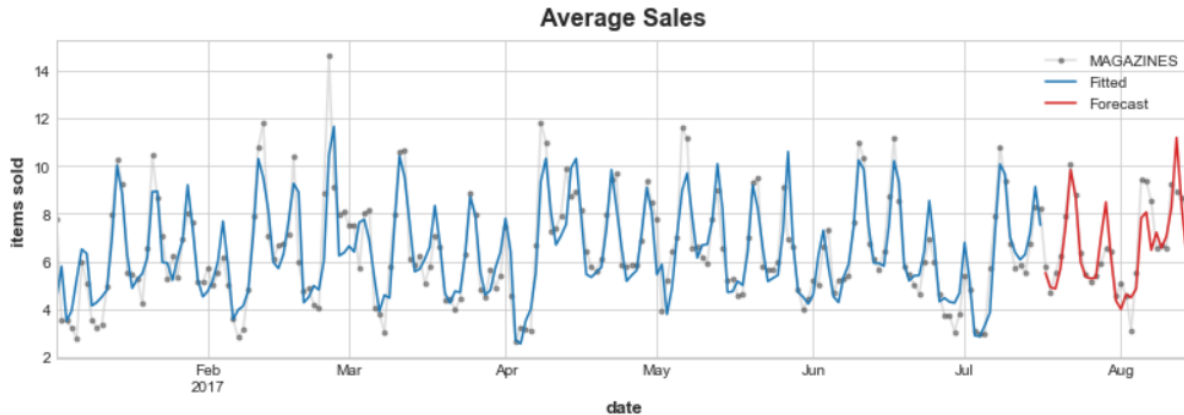
**Moving average plot (trend)**

After exploring some trends and patterns using some basic linear regression models we have used a function from the statsmodels library called `DeterministicProcess`. This function helped us in avoiding some tricky failure cases that can arise with time series and linear regression.

Training RMSLE: 0.14679  
Validation RMSLE: 0.13098

Training RMSE: 0.07828  
Validation RMSE: 0.19057

Training MAE: 0.83372  
Validation MAE: 0.77922



We had a kaggle score of 0.51 (rmsle) for this model. This model includes moving averages and other rolling statistics in their feature sets. So, we thought such features would be useful when used with GBDT (Gradient Boosting algorithms like XGBoost).

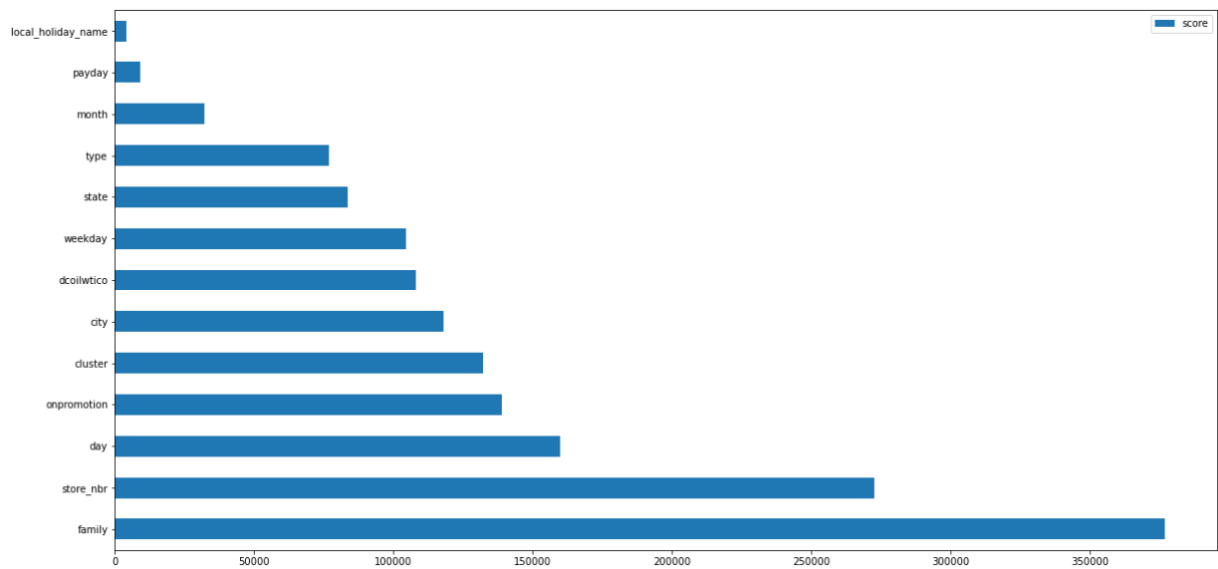
### Ensemble Models:

We tried to implement XGBoost and Random Forest to the dataset. Since the dataset was too large we have used DMatrix in our XGBoost model which was helpful for both memory efficiency and training speed.

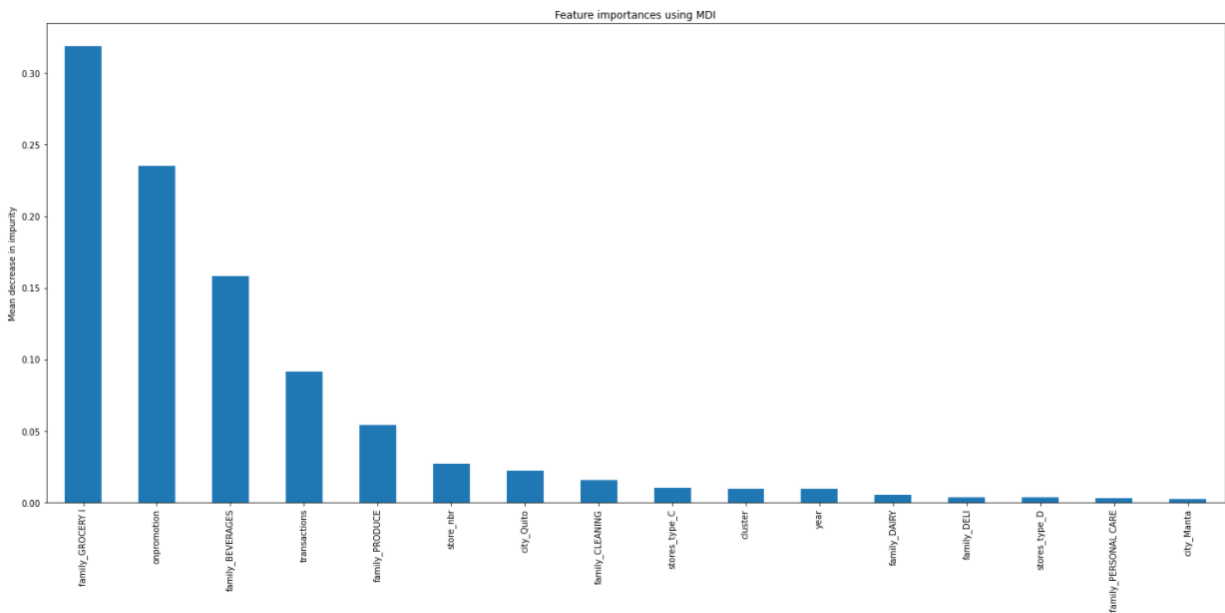
```
[0]      train-rmse:3.98392      eval-rmse:3.98526
[5000]   train-rmse:0.40131      eval-rmse:0.46033
[10000]  train-rmse:0.37629      eval-rmse:0.44340
[15000]  train-rmse:0.36613      eval-rmse:0.43756
[20000]  train-rmse:0.36022      eval-rmse:0.43459
[25000]  train-rmse:0.35617      eval-rmse:0.43271
[30000]  train-rmse:0.35315      eval-rmse:0.43145
[35000]  train-rmse:0.35078      eval-rmse:0.43050
[35606]  train-rmse:0.35053      eval-rmse:0.43043
rmse: 0.430418
mae: 0.301451 , elapsed time: 331.21sec
```



With the validation set we achieved a decent score so when this model was submitted to kaggle, we achieved Kaggle rmsle: 0.45 for XGBoost and 0.44 for Random Forest models which was much better compared to our previous linear regression model. We observed that both the tree models had family as the most important feature.



Feature Importance (XgBoost)



Feature Importance (Random Forest)

## Hybrid Model (Ridge + RandomForest)

In our first model we used linear regression which excels in extrapolating trends and in our second model we used XGBoost and Random Forest which excels at learning interactions, but can't extrapolate trends. The solution we found was to build a hybrid forecaster model that combines complementary learning algorithms and let the strengths of one make up for the weakness of the other.

When Ridge model was implemented on the family feature we found the model's performance was low for School and Office Supplies, whereas Random Forest model did a good job on that. We experimented with different alpha values in which the best result was from alpha=0.5.


family		family	
AUTOMOTIVE	0.259202	AUTOMOTIVE	0.259202
BABY CARE	0.066660	BABY CARE	0.066660
BEAUTY	0.267450	BEAUTY	0.267450
BEVERAGES	0.199187	BEVERAGES	0.199187
BOOKS	0.026701	BOOKS	0.026701
BREAD/BAKERY	0.125449	BREAD/BAKERY	0.125449
CELEBRATION	0.295910	CELEBRATION	0.295910
CLEANING	0.204513	CLEANING	0.204513
DAIRY	0.136196	DAIRY	0.136196
DELI	0.108830	DELI	0.108830
EGGS	0.147672	EGGS	0.147672
FROZEN FOODS	0.145027	FROZEN FOODS	0.145027
GROCERY I	0.210306	GROCERY I	0.210306
GROCERY II	0.347753	GROCERY II	0.347753
HARDWARE	0.273930	HARDWARE	0.273930
HOME AND KITCHEN I	0.259483	HOME AND KITCHEN I	0.259483
HOME AND KITCHEN II	0.219104	HOME AND KITCHEN II	0.219104
HOME APPLIANCES	0.154737	HOME APPLIANCES	0.154737
HOME CARE	0.122103	HOME CARE	0.122103
LADIESWEAR	0.259483	LADIESWEAR	0.259483
LAWN AND GARDEN	0.216374	LAWN AND GARDEN	0.216374
LINGERIE	0.400216	LINGERIE	0.400216
LIQUOR,WINE,BEER	0.612719	LIQUOR,WINE,BEER	0.612719
MAGAZINES	0.254086	MAGAZINES	0.254086
MEATS	0.123141	MEATS	0.123141
PERSONAL CARE	0.137507	PERSONAL CARE	0.137507
PET SUPPLIES	0.210464	PET SUPPLIES	0.210464
PLAYERS AND ELECTRONICS	0.223845	PLAYERS AND ELECTRONICS	0.223845
POULTRY	0.122109	POULTRY	0.122109
PREPARED FOODS	0.126234	PREPARED FOODS	0.126234
PRODUCE	0.184626	PRODUCE	0.184626
SCHOOL AND OFFICE SUPPLIES	<u>1.394414</u>	SCHOOL AND OFFICE SUPPLIES	<u>0.074222</u>
SEAFOOD	0.253164	SEAFOOD	0.253164
dtype: float64		dtype: float64	

Our hybrid model was built with Random Forest for "School and office supplies" feature and Ridge model for predicting other family features which gave us the best Kaggle score of 0.42.

### Models and Kaggle results:

Model	RMSLE
Linear Regression	0.51
XGBoost	0.45
RandomForest	0.44
Ridge	0.43
Hybrid Model (Ridge + Random Forest)	0.42

### Kaggle Standing:

Overview	Data	Code	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions	...	
60	Group 4							0.42398	8	5d

Ranked 60 among 900 participants in the Store Sales Time series forecasting kaggle competition.

## Conclusion/Future Work

Overall, this project was a great learning experience in which we were able to apply what we learned in our course to a real world application. Regarding future work, there could be an extension done where the models that Schöneburg (1990) and Ariyo et. al (2014) could be applied, simply to see how their more complex models fare at predicting sales where there is not as much volatility. The comparison between resources used and the overall accuracy attained with each set of models would be interesting to examine. The simpler models that we used had garnered us a rank in the top 50 of 800 at the time of writing this report, so examining the tradeoffs between resources and real world performance would be the next logical step for this project.