Time Series Analysis using Machine Learning

Matt Brown

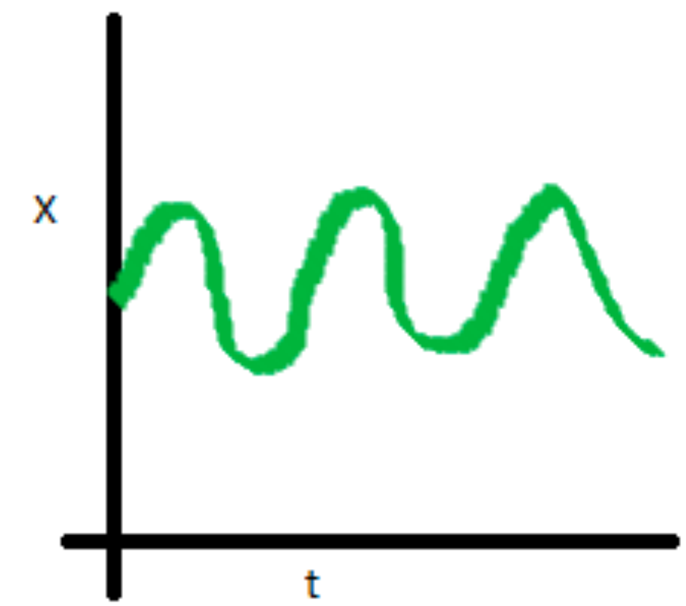
2/13/2018

Agenda

- **Time Series Analysis Overview**
- **Machine Learning for Time Series Forecasting**
- **Walkthrough Example from Kaggle**
  - Baseline Prediction and Initial Model
  - Data Selection
  - Data Cleansing
  - Feature Engineering
  - Modeling
  - Feature Selection
- **Conclusion**

Time Series Analysis

A time series is a set of observations in sequential order, which are generally at successive equally spaced points in time.



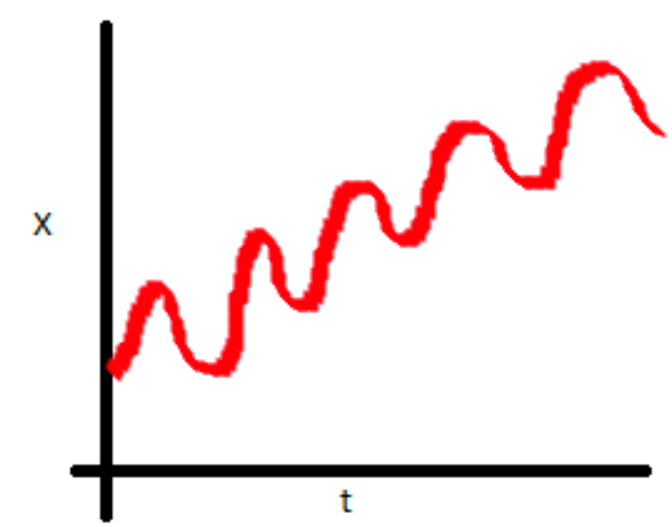
Stationary series

Examples

- Retail Sales
- Financial Data (e.g. stock prices and treasury rates)
- Service Queues (E.g. How many people will arrive at a restaurant?)
- Economic Statistics (e.g. unemployment rates)
- Health Data

Overview

- **Trend:** Long-term change in the mean of the data.
- **Detrend:** Removing trends to show the possible cyclical patterns in the data.
- **Seasonal:** Periodic fluctuation (E.g. increased holiday sales).
- Time series forecasts can be short or long-term. Long-term prediction complexity increases based on the horizon of the predictive period.
- Machine learning can be adapted to develop time series models.



Non-Stationary series

Source: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

Machine Learning for Time Series Forecasting

Overview

- Time series forecasting can be framed as a supervised machine learning problem.
- Supervised learning requires input variables (x) and an output variable (y) and uses an algorithm to learn the mapping function from the input to the output.

$$y = f(x)$$

- The sliding window method can be used to frame a time series problem as a machine learning problem.

```
      date unit_sales
     <chr>      <dbl>
2017-08-01  1.3862944
2017-08-02  0.6931472
2017-08-03  1.0986123
2017-08-04  1.3862944
2017-08-05  0.0000000
2017-08-06  0.0000000
2017-08-07  0.0000000
```

- After applying the sliding window method, supervised machine learning algorithm can be applied to the data set.

https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

Machine Learning for Time Series Forecasting

Overview

- Time series forecasting can be framed as a supervised machine learning problem.
- Supervised learning requires input variables (x) and an output variable (y) and uses an algorithm to learn the mapping function from the input to the output.

$$y = f(x)$$

- The sliding window method can be used to frame a time series problem as a machine learning problem.

One day lag

date	unit_sales	lag_1
<chr>	<dbl>	<dbl>
2017-08-01	1.3862944	NA
2017-08-02	0.6931472	1.3862944
2017-08-03	1.0986123	0.6931472
2017-08-04	1.3862944	1.0986123
2017-08-05	0.0000000	1.3862944
2017-08-06	0.0000000	0.0000000
2017-08-07	0.0000000	0.0000000

- After applying the sliding window method, supervised machine learning algorithm can be applied to the data set.

https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

Problem Definition

Overview: We will use a retail sales data set from Kaggle to demonstrate how machine learning can be used for time series forecasting.

Description: Corporación Favorita, a large Ecuadorian-based grocery retailer that operates hundreds of supermarkets, with over 200,000 different products on their shelves, has challenged the Kaggle community to build a model that more accurately forecasts product sales.

Training Data: Sales data from March 1, 2012 - August 15th, 2017.

```
##           id       date store_nbr item_nbr unit_sales onpromotion
## 1: 123926072 2017-08-01         1   103520          3       FALSE
## 2: 123926073 2017-08-01         1   103665          4       FALSE
## 3: 123926074 2017-08-01         1   105574          8       FALSE
## 4: 123926075 2017-08-01         1   105575         14       FALSE
## 5: 123926076 2017-08-01         1   105693          1       FALSE
## 6: 123926077 2017-08-01         1   105737          1       FALSE
```

Goal: Predict sales for each day from August 16th - August 31st 2017 for a given product at a given store.

Baseline and Initial Model

A baseline is a reference point from which to measure the accuracy of a model.

Baseline

- Start each project by generating a baseline prediction. Examples:
  - **Classification:** Predict the most common occurrence for all cases.
  - **Regression:** Predict the mean or median for all cases.
- A baseline prediction should be simple to generate.
- Choose a accuracy score to measure the accuracy of the predictions (E.g. Root Mean Square Error).

```
RMSE = function(m, o){
  sqrt(mean((m - o)^2))
}
```

Initial Model

- Build a minimal model using a subset of the data.
  - Decreases data processing waiting time.
  - Easier to identify errors.
  - Simple to add / modify the sample data after a working has been built.
- Compare the initial model to the baseline.

https://machinelearningmastery.com/how-to-get-baseline-results-and-why-they-matter/

Data Selection

Data selection is choosing the sample of data to use in building a model. The quality of data chosen will have a great impact on the accuracy of the model.

Overview

- Avoid *selection bias, c*hoose data that accurately represents the future case you are predicting.
- Choose the ‘best’ data for whatever you are predicting.
- Different subsets of time series data may require different features (e.g. only using data from a specific month will negate the need to have a *month* feature).
- Consider not only the current problem but also how the model could be used in the future.

Example

- The data set for this competition was too large to load into memory.
- Tested the accuracy of the model on various subsets of the data.
- Used only data from the summer months since the competition wanted only predictions for the month of August.

“More data beats better algorithms…better data beats more data”

-Daniel Tunkelang, Chief Search Evangelist at Twiggle, Ex-Head of Search Quality at LinkedIn

Data Cleansing Tips

Data cleansing is the process of detecting and correcting inaccurate or corrupt entries in a dataset with the end goal of creating a data set which is consistent with other similar data sets in the system

Overview

- Do not clean just for the sake of it, identify the specific problem you are analyzing. What data do you need for that?
- Always use a scripting language to clean. All steps should be repeatable.
- Always be suspect of the data. Even though the first few rows are clean there may be grave errors as you analyze further.
- Explicitly handle NA values. If there are only a few then you can just remove those rows otherwise you may need to implement an imputation technique.

Data Cleansing

Example

- In this problem, the training and test sets have inconsistent formats
- Use the *padr* package to add missing zeros to the training dataset so that its format matches the test dataset.

```
train$date <- as.Date(train$date) #Convert to date data type

train <- pad(train, start_val = as.Date(MinDate)
             , end_val = as.Date(MaxDate), interval = 'day'
             , group = c('store_nbr', 'item_nbr'), break_above = 100000000000
) %>%
  fill_by_value(unit_sales) #Adds zeros to the dataframe. Adds NAs without this argument.
```

Original

```
##          date unit_sales store_nbr item_nbr
## 1: 2017-08-01          3         1   103520
## 2: 2017-08-02          1         1   103520
## 3: 2017-08-03          2         1   103520
## 4: 2017-08-04          3         1   103520
## 5: 2017-08-08          3         1   103520
## 6: 2017-08-10          3         1   103520
## 7: 2017-08-11          1         1   103520
## 8: 2017-08-12          1         1   103520
## 9: 2017-08-13          1         1   103520
```

Updated

```
##           date unit_sales store_nbr item_nbr
##  1: 2017-08-01          3         1   103520
##  2: 2017-08-02          1         1   103520
##  3: 2017-08-03          2         1   103520
##  4: 2017-08-04          3         1   103520
##  5: 2017-08-05          0         1   103520
##  6: 2017-08-06          0         1   103520
##  7: 2017-08-07          0         1   103520
##  8: 2017-08-08          3         1   103520
##  9: 2017-08-09          0         1   103520
## 10: 2017-08-10          3         1   103520
```

Feature Engineering

Feature engineering is the process of using domain knowledge to create new features to improve the accuracy of machine learning models.

Overview

- The quality of features will have the greatest impact on the accuracy of the model.
- Performance of machine learning algorithms decrease with an increased number of correlated input features (I.e. Noise decreases accuracy)
- Learn about the domain. What is important to know about this problem?
- Be creative, it doesn’t hurt to test new features.
- Explicitly specify feature types.

Quotes

Coming up with features is difficult, time-consuming, requires expert knowledge. “Applied machine learning” is basically feature engineering. - Andrew Ng, *Machine Learning and AI via Brain simulations*

…you have to turn your inputs into things the algorithm can understand…
- Shayne Miel, answer to “What is the intuitive explanation of feature engineering in machine learning?”

Feature Engineering for Time Series

Overview

- Machine learning algorithms cannot ‘understand’ dates.
- Temporal features must be explicitly created.
- The features must be of numeric data type.

Example

Date: ‘2017-08-01’

- Here we extract four time features from the ‘date’ column which may illuminate trends and seasonality in the data.

```
#Create and Modify variables
df$month <- as.numeric(substr(df$date, 6, 7))
df$day <- as.numeric(as.factor(weekdays(as.Date(df$date)))) #day of week
df$day_num <- as.numeric(substr(df$date, 9, 10))
df$year <- substr(df$date, 1, 4)
```

Feature Engineering

Overview

- **Sliding window method** can be used to create new temporal features for this problem.
- The *RcppRoll* package can be used to create sliding window variables
- Features we could create here are nearly endless (e.g. rolling mean / median / max / sd / min for any number of days).

Example

```
train <- train %>%
  group_by(store_nbr, item_nbr) %>%
  mutate(lag_1 = lag(unit_sales, 1)
         , avg_3 = lag(roll_meanr(unit_sales, 3), 1)
         )
```

One day lag

date	unit_sales	lag_1	store_nbr	item_nbr
<chr>	<dbl>	<dbl>	<int>	<int>
2017-08-01	1.3862944	NA	1	103520
2017-08-02	0.6931472	1.3862944	1	103520
2017-08-03	1.0986123	0.6931472	1	103520
2017-08-04	1.3862944	1.0986123	1	103520
2017-08-05	0.0000000	1.3862944	1	103520
2017-08-06	0.0000000	0.0000000	1	103520
2017-08-07	0.0000000	0.0000000	1	103520

Previous three day average

unit_sales	avg_3
<dbl>	<dbl>
1.3862944	NA
0.6931472	NA
1.0986123	NA
1.3862944	1.0593513
0.0000000	1.0593513
0.0000000	0.8283022
0.0000000	0.4620981

Feature Engineering

Updated Data Set

- Created seven new features from the date and unit_sales columns.
- These will be suitable to build an initial model.

id	date	store_nbr	item_nbr	unit_sales	onpromotion	month	day	day_num	year	lag_1	avg_7	avg_3
124040053	2017-08-02	3	208659	5	FALSE	8	7	2	2017	5	5.428571	4.666667
124147068	2017-08-03	3	208659	8	FALSE	8	5	3	2017	5	4.857143	5.000000
124249525	2017-08-04	3	208659	13	FALSE	8	1	4	2017	8	5.714286	6.000000
124353637	2017-08-05	3	208659	7	FALSE	8	3	5	2017	13	7.000000	8.666667
124464110	2017-08-06	3	208659	11	FALSE	8	4	6	2017	7	6.714286	9.333333
124575321	2017-08-07	3	208659	10	FALSE	8	2	7	2017	11	7.714286	10.333333
124679094	2017-08-08	3	208659	10	FALSE	8	6	8	2017	10	8.428571	9.333333
124780347	2017-08-09	3	208659	6	FALSE	8	7	9	2017	10	9.142857	10.333333

Modeling

Overview

- **Data cleansing and feature engineering are the most important steps to building an accurate model.**
- Use the simplest model. You can often get 90% of the way without a complex solution.
- Ensemble models are generally more accurate than any one algorithm (i.e. ‘Wisdom of the crowd’).
- Don’t build a new model each time you generate a prediction.
- Choose an evaluation metric for your model such as RMSE.

Popular Modeling Tools

- Top participants in this competition generally used gradient boosted trees (packages *Light GBM* and *xgboost*) and neural networks for modeling.
- They used ensemble methods to improve the predictions.
- Python was the most used language.

Modeling and Predicting

Recursive Prediction for Multi-step Forecasting: The recursive strategy involves using the prediction for the prior time step as an input for making a prediction on the following time step.

- The field colored **black** is the target variable and the fields colored **blue** are the features we use to predict the target.
- All of these fields must be updated during each prediction iteration.
- The inputs are predicted values rather than actual historical data.

id	date	store_nbr	item_nbr	unit_sales	onpromotion	month	day	day_num	year	lag_1	avg_7	avg_3
125185374	2017-08-13	1	103520	0.6931472	FALSE	8	4	13	2017	0.6931472	0.5941262	0.9241962
NA	2017-08-14	1	103520	0.0000000	NA	8	2	14	2017	0.6931472	0.6931472	0.6931472
NA	2017-08-15	1	103520	0.0000000	NA	8	6	15	2017	0.0000000	0.6931472	0.4620981
125497043	2017-08-16	1	103520	NA	FALSE	8	7	16	2017	0.0000000	0.4951051	0.2310491
125707697	2017-08-17	1	103520	NA	FALSE	8	5	17	2017	NA	NA	NA
125918351	2017-08-18	1	103520	NA	FALSE	8	1	18	2017	NA	NA	NA
126129005	2017-08-19	1	103520	NA	FALSE	8	3	19	2017	NA	NA	NA
126339659	2017-08-20	1	103520	NA	FALSE	8	4	20	2017	NA	NA	NA
126550313	2017-08-21	1	103520	NA	FALSE	8	2	21	2017	NA	NA	NA
126760967	2017-08-22	1	103520	NA	FALSE	8	6	22	2017	NA	NA	NA

- Complexity increases with the length of the predictive period in long-term forecasts.
- Errors will propagate through the model which can decrease the accuracy of longer-term predictions.

Modeling and Predicting

Recursive Prediction for Multi-step Forecasting: The recursive strategy involves using the prediction for the prior time step as an input for making a prediction on the following time step.

- The field colored **black** is the target variable and the fields colored **blue** are the features we use to predict the target.
- All of these fields must be updated during each prediction iteration.
- The inputs are predicted values rather than actual historical data.

id	date	store_nbr	item_nbr	unit_sales	onpromotion	month	day	day_num	year	lag_1	avg_7	avg_3
125185374	2017-08-13	1	103520	0.6931472	FALSE	8	4	13	2017	0.6931472	0.5941262	0.9241962
NA	2017-08-14	1	103520	0.0000000	NA	8	2	14	2017	0.6931472	0.6931472	0.6931472
NA	2017-08-15	1	103520	0.0000000	NA	8	6	15	2017	0.0000000	0.6931472	0.4620981
125497043	2017-08-16	1	103520	0.4703642	FALSE	8	7	16	2017	0.0000000	0.4951051	0.2310491
125707697	2017-08-17	1	103520	NA	FALSE	8	5	17	2017	0.4703642	0.5623000	0.1567881
125918351	2017-08-18	1	103520	NA	FALSE	8	1	18	2017	NA	NA	NA
126129005	2017-08-19	1	103520	NA	FALSE	8	3	19	2017	NA	NA	NA
126339659	2017-08-20	1	103520	NA	FALSE	8	4	20	2017	NA	NA	NA
126550313	2017-08-21	1	103520	NA	FALSE	8	2	21	2017	NA	NA	NA
126760967	2017-08-22	1	103520	NA	FALSE	8	6	22	2017	NA	NA	NA

- Complexity increases with the length of the predictive period in long-term forecasts.
- Errors will propagate through the model which can decrease the accuracy of longer-term predictions.

Modeling and Predicting

Recursive Prediction for Multi-step Forecasting: The recursive strategy involves using the prediction for the prior time step as an input for making a prediction on the following time step.

- The field colored **black** is the target variable and the fields colored **blue** are the features we use to predict the target.
- All of these fields must be updated during each prediction iteration.
- The inputs are predicted values rather than actual historical data.

id	date	store_nbr	item_nbr	unit_sales	onpromotion	month	day	day_num	year	lag_1	avg_7	avg_3
125185374	2017-08-13	1	103520	0.6931472	FALSE	8	4	13	2017	0.6931472	0.5941262	0.9241962
NA	2017-08-14	1	103520	0.0000000	NA	8	2	14	2017	0.6931472	0.6931472	0.6931472
NA	2017-08-15	1	103520	0.0000000	NA	8	6	15	2017	0.0000000	0.6931472	0.4620981
125497043	2017-08-16	1	103520	0.4703642	FALSE	8	7	16	2017	0.0000000	0.4951051	0.2310491
125707697	2017-08-17	1	103520	0.5400128	FALSE	8	5	17	2017	0.4703642	0.5623000	0.1567881
125918351	2017-08-18	1	103520	NA	FALSE	8	1	18	2017	0.5400128	0.4414026	0.3367923
126129005	2017-08-19	1	103520	NA	FALSE	8	3	19	2017	NA	NA	NA
126339659	2017-08-20	1	103520	NA	FALSE	8	4	20	2017	NA	NA	NA
126550313	2017-08-21	1	103520	NA	FALSE	8	2	21	2017	NA	NA	NA
126760967	2017-08-22	1	103520	NA	FALSE	8	6	22	2017	NA	NA	NA

- Complexity increases with the length of the predictive period in long-term forecasts.
- Errors will propagate through the model which can decrease the accuracy of longer-term predictions.

Modeling and Predicting

Recursive Prediction for Multi-step Forecasting: The recursive strategy involves using the prediction for the prior time step as an input for making a prediction on the following time step.

- The field colored **black** is the target variable and the fields colored **blue** are the features we use to predict the target.
- All of these fields must be updated during each prediction iteration.
- The inputs are predicted values rather than actual historical data.

id	date	store_nbr	item_nbr	unit_sales	onpromotion	month	day	day_num	year	lag_1	avg_7	avg_3
125185374	2017-08-13	1	103520	0.6931472	FALSE	8	4	13	2017	0.6931472	0.5941262	0.9241962
NA	2017-08-14	1	103520	0.0000000	NA	8	2	14	2017	0.6931472	0.6931472	0.6931472
NA	2017-08-15	1	103520	0.0000000	NA	8	6	15	2017	0.0000000	0.6931472	0.4620981
125497043	2017-08-16	1	103520	0.4703642	FALSE	8	7	16	2017	0.0000000	0.4951051	0.2310491
125707697	2017-08-17	1	103520	0.5400128	FALSE	8	5	17	2017	0.4703642	0.5623000	0.1567881
125918351	2017-08-18	1	103520	0.4711375	FALSE	8	1	18	2017	0.5400128	0.4414026	0.3367923
126129005	2017-08-19	1	103520	NA	FALSE	8	3	19	2017	0.4711375	0.4096870	0.4938382
126339659	2017-08-20	1	103520	NA	FALSE	8	4	20	2017	NA	NA	NA
126550313	2017-08-21	1	103520	NA	FALSE	8	2	21	2017	NA	NA	NA
126760967	2017-08-22	1	103520	NA	FALSE	8	6	22	2017	NA	NA	NA

- Complexity increases with the length of the predictive period in long-term forecasts.
- Errors will propagate through the model which can decrease the accuracy of longer-term predictions.

Feature Selection

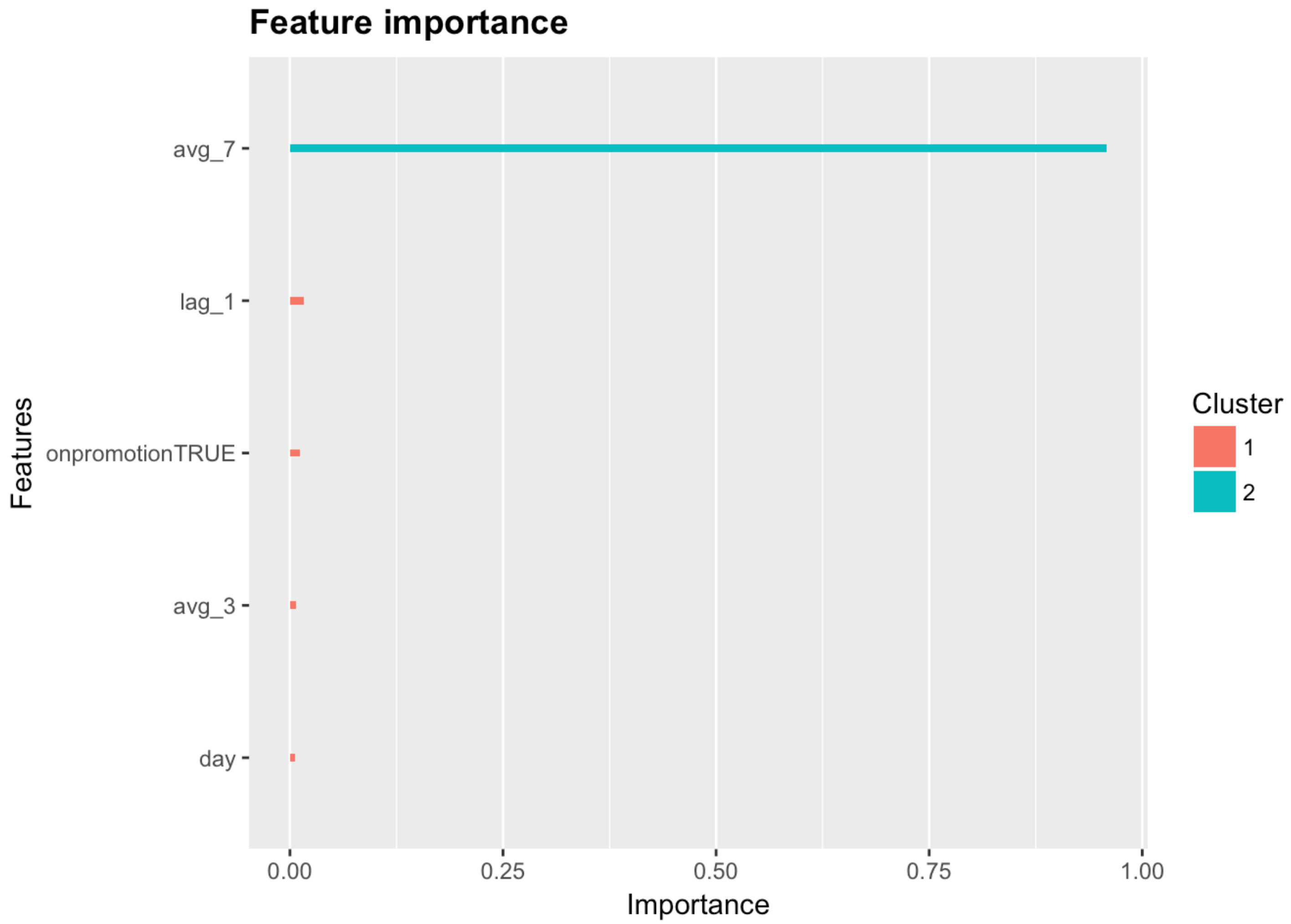
Feature selection is choosing the features to use in the model.

Overview

- Models are both more interpretable and potentially more accurate with fewer features (i.e. less noise).
- Plotting the importance of the features to the model can help you understand feature importance

Example

```
importance <- xgb.importance(feature_names = colnames(trainMatrix), model = model)
xgb.ggplot.importance(importance_matrix = importance)
```



Conclusion

- Try the simplest model first.
- Spend time on feature selection and engineering.
- Always generate a baseline to compare against the model.
- Expert knowledge is important, spend time getting to know the data.
- Stay focus on the end goal of the analysis, don't get lost in optimizing the small details.

Email: matt@RedOakStrategic.com

"It's tough to make predictions, especially about the future"

-Yogi Berra

RED OAK strategic