Online Sales Prediction

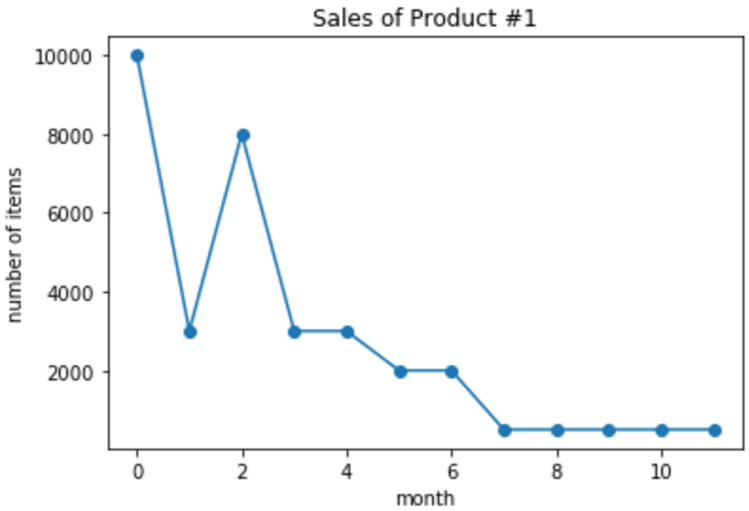
Nov 2017

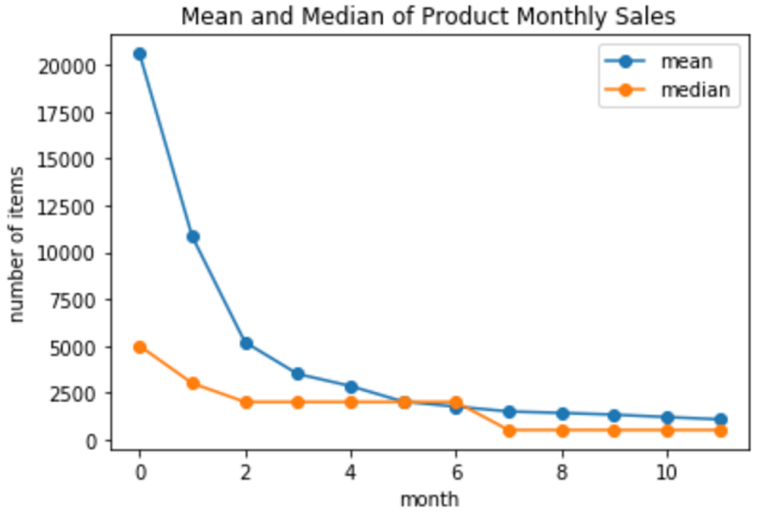
Introduction

1. **Project object**
2. **Understand the data**
3. **Feature engineering**
4. **Model Selection and Parameter Tuning**
5. **Conclusions and Future works**

Project Object

Using product and corresponding campaign information to predict the sales in the first 12 months after each product has been released.





Understand Data

Data source:

"TrainingDataset.csv": training data, data with known sales.

"TestDataset.csv": testing data, data with unknown sales which need to predict.

(Since the leaderboard of this competition does not allow submission anymore, instead of using "TestDataset.csv", I split "TrainingDataset.csv" to training and test sets. In this case I can evaluate the performance.)

Data type:

Targets:

12 months sales – 12 quantitative variables

Features:

513 categorical variables, 31 quantitative variables, 2 date variables.

Evaluation Metric

The evaluation metric is RMSLE (root mean square logarithmic error) across all 12 prediction columns.

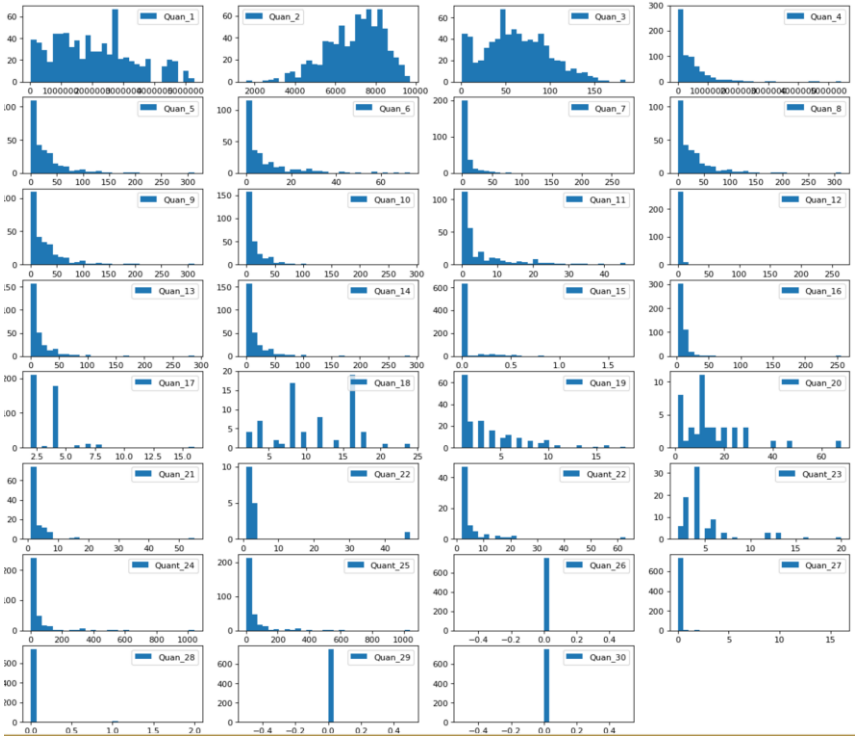
The RMSLE is calculated as

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

- ϵ is the RMSLE value (score)
- n is the total sales in the (public/private) data set
- p_i is your prediction
- a_i is the actual sales for i
- $\log(x)$ is the natural logarithim of x

Feature Engineering



Quantitative features:

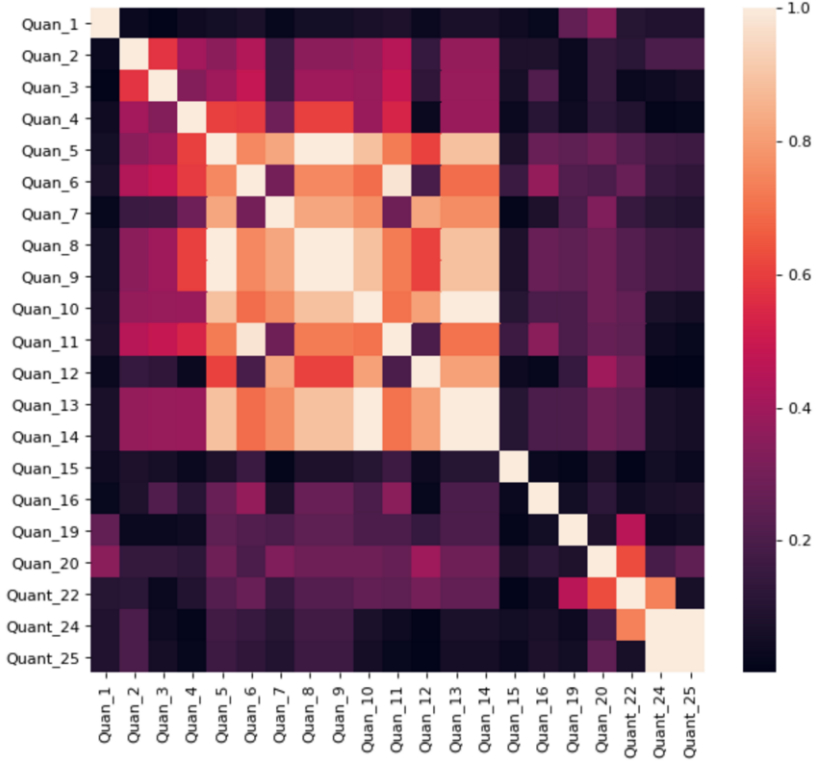
Due to the branching methods of tree models, we need to make sure all the quantitative features follow Gaussian distribution or uniform distribution.

According to the distributions on the left, some features follow long-tail distributions. They can be easily transformed to Gaussian distributions by applying "**log(x+1)**" on them.

Moreover, some quantitative variables have very limited numeric values, such as "**Quan_17**". These features should be treated as categorical features.

At last, some features are invariance, such as "**Quan_29**" and "**Quan_30**". They should be removed.

Feature Engineering



Correlations between quantitative features:

Some features are highly correlated, like “Quan_13” and “Quan_14”.

If we use some linear models, such as linear regression, we need to remove the highly correlated features before modeling.

If we use tree models, such as decision tree and random forest, we can leave these correlated features in the feature set.

Feature Engineering

Categorical features:

Too many categorical features and many of them are purely NULL.
Invariant categorical features should be removed.
Then apply one hot encoder on categorical features.

Date features:

Two dates are included in the dataset.
"Date_1": date that major advertising campaign began and the product launched.
"Date_2": date that product was announced and a pre-release advertising campaign began.
Create 5 new date related new features:

1) Time difference between "Date_1" and "Date_2"
2) Years and months of "Date_1" and "Date_2". This information should capture seasonality.

Target variables:

Since the evaluation metric is RMSLE, I transfer all the sales to log(x+1). The model can directly fit to log values.

Missing data:

Missing data in categorical variables is filled with "nan".
Missing data in quantitative variables is filled with median.

Note: All the feature engineering is conducted on the integration of "trainingDataset.csv" and "testDataset.csv"

Model Selection

Multi-model approach: creates a model for each month, 12 models in this case.

Single model approach: creates a single model for all 12 months, one model in this case.

The multi-model approach may yield more accurate results than the single model approach. But it requires much longer time to tuning the hyper-parameters of each model separately.

I adopt the single model approach for this project due to the tight deadline.

If the single model approach is adopted, the month should be transferred to a categorical feature. In this case, each row represents a product's sale in one month.

Caution:

Data leakage issue is very likely to happen here.

We should split the data to training, validation, and testing sets on product level. So 12 months sales of a product can only appear in one set. Otherwise, we will very likely to use this product's first month sales in the training set to predict this product's second month sales in the validating set. It will give you a fake boost in prediction accuracy when you check the validation accuracy.

Model Selection and Parameter Tuning

When in doubt, use xgboost.

-- Avito Winner's Interview, Owen Zhang [1].

I use XGBoost Regressor for this project.

Four important parameters for XGBoost:

"n_estimators": number of gradient boot trees.
If the number of trees is too large, the model is very likely to overfit to the training set.

"learning_rate": if the learning rate is too large, the model may stuck in local minima;
if the learning rate is too small, the model may not find the minima after a long time.

"subsample": the proportion of rows used to build the model.

"colsample_bytree": the proportion of the columns used to build the model.

"max_depth": max depth of each individual tree.

Purpose of Parameter Tuning:

Gradient boost approaches can very easily overfit the training set.

We need to find the balance between overfitting and underfitting.

[1] http://blog.kaggle.com/2015/08/26/avito-winners-interview-1st-place-owen-zhang/

Parameter Tuning

1. **Find a rough "n_estimators" with a comparable large "learning_rate".**
   I use the "early stopping mechanism" of XGBoost. By using this method, XGBoost will stop adding more trees when the accuracy in the validation set stops increasing (prevent overfitting). The reason of choosing a large learning rate at first is to increase the fitting speed of the model and decrease the tuning time. (learning_rate=0.1, n_estimators=223)

2. **Based on determined "n_estimator",**
   use k-fold cross-validation to find the optimal values of the other parameters
   ("subsample", "max_depth", "colsample_bytree").
   This step is very time consumed. But the advantage of the large learning rate determined in previous step is shown here. The cross validation takes a much shorter time than using a small learning rate.

3. **After the other hyper-parameters have been determined, change "learning_rate" to a small value (0.01 or even 0.001) and use it to find the optimal "n_estimators" before overfitting.**
   In terms of the accuracy, a small learning rate with a large number of trees is always better than a large learning rate.

4. **Retrain the model with the optimal hyper-parameters on the whole training set.**

Model Performance

A random forest model is also built as the benchmark for comparisons.

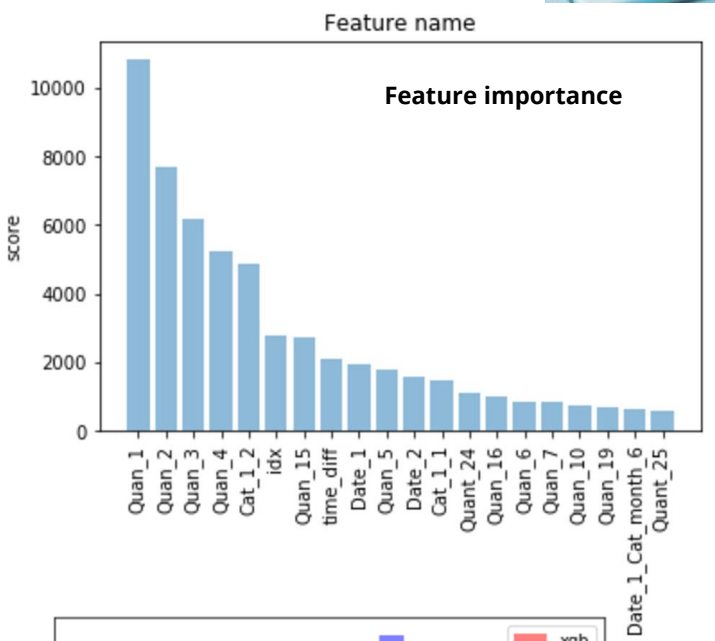
The reason to use RF as the benchmark:

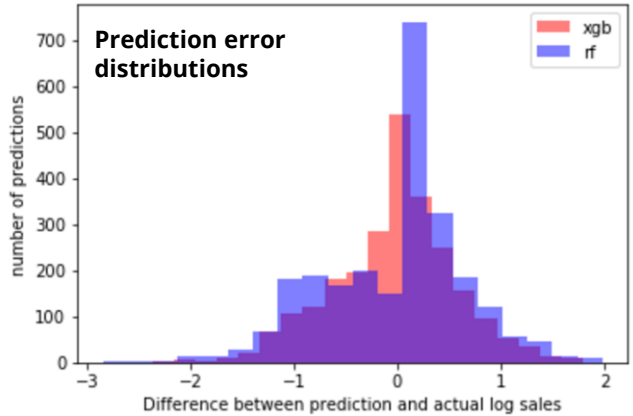
1. Don't require complex hyper-parameters tuning.
According to RF's characteristics, no cross-validation is required. RF doesn't overfit to the training set and can achieve a fair good results without any parameter tuning.

2. RF is an ensemble model, it has better performances than many meta-models.

RMSLE of Random Forest in our test set: 0.6843. No.128 (top 35%) in the leaderboard.

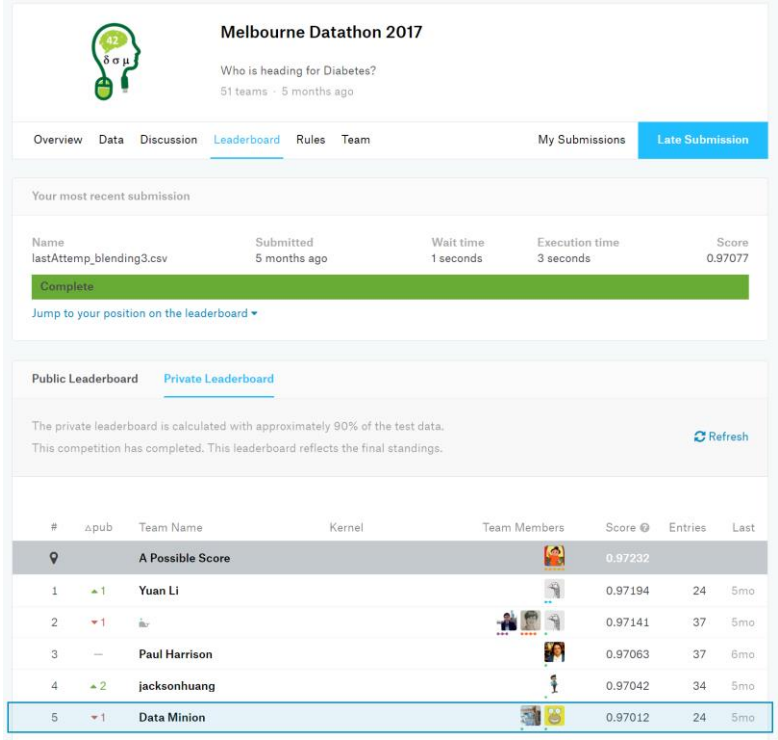
RMSLE of XGBoost in our test set: 0.5935. No.26 (top 7%) in the leaderboard.





Conclusion and Future Works

1. According to my experience, if the basic feature engineering has been done correctly, xgboost should yield fairly good results on most of the predictive projects. (We used basic feature engineering + XGBoost to achieve Top 5 performance in the Melbourne Datathon this year.)
2. Further feature engineering and blending may give you another up to 10% boost.
3. If I have more time, I will grid search on the combinations of features instead of grid search one by one.
4. New features can be created by combining different features. Since the feature names are blinded, it's not possible to come up with some new features by using the knowledge in marketing. This process is very time consumed.



Thank you.

Failed Methods

1. **Build the model to fit sales directly without transferring sales to log(x+1).**

2. **Split training and testing sets on monthly sales level instead of product level. The score of testing result is much higher than the Top 1 in the leaderboard due to the data leakage issue.**