Capstone Project

Machine Learning Engineer Nanodegree

Fred
July 22th, 2016

Definition

Project Overview

This project is originated from the Kaggle competition 'Rossmann Store Sales' in 2015, which can be found in https://www.kaggle.com/c/rossmann-store-sales. The following introduction is extracted from the competition home page:

"Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

In their first Kaggle competition, Rossmann is challenging you to predict 6 weeks of daily sales for 1,115 stores located across Germany. Reliable sales forecasts enable store managers to create effective staff schedules that increase productivity and motivation. By helping Rossmann create a robust prediction model, you will help store managers stay focused on what's most important to them: their customers and their teams!"

This project is chosen in this report mainly because of its practical significance. Sales prediction based on given sets of features is important for decision makers in real life. The problem is also challenging as the features contain missing data and a mixture of categorical and continuous features.

The project partially uses the dataset from the competition. The original competition has 3 sets of data: a training set with labels, and test set with only features, and a supplementary information set about the Rossmann stores. The test set is used to predict sales for the competition and contains feature-only data for the 6 weeks. The training set contains the historical sales data for 1115 Rossmann stores. The supplementary set contains information about the stores, such as their distance to competitors and promotion information. In this

project, only the labeled training set and the supplementary information set are used. In contrast to forecast future sales as in the competition, this project aims to predict sales of some store for some day based on given features, such as the number of customers for the day, whether the day is a holiday, the day of week, the store type, the competitor distance, and promotion information, etc.

Problem Statement

We have the following problem statement based on the introduction in project overview.

Problem:

The problem we are trying to solve is: given a set of features about a list stores over a period of time, predict their sales.

Strategy:

- Data exploration and preprocessing: the data will be examined to determine if there are any missing values and select the useful features as input to modeling. Statistics about the data will also be obtained. The data will be split into training, validation, and testing subsets.

- Develop baseline performance benchmark: select dummy baseline method that the trained model should be benchmarked against.

- Model selection, training, optimization, and evaluation: select candidate models, use training subset to train the model, and use validation subset to tune and optimize the model, and finally evaluate the model using the testing subset.

Anticipated solution:

The problem is a regression problem. From the description above, the input data contains categorical, discrete and continuous features, and therefore we can either try feature engineering techniques like one hot encoding, or we can use models that can handle all three types of features. One promising candidate is tree-based algorithms, which can handle both categorical and continuous features.

Metrics

Predicting sales figure is a regression problem, and the metric used will be root mean square error (RMSE), which is the square root of MSE that has very nice mathematical properties. Besides, it is more sensitive to large errors than MAE, and is a desired property in that we prefer solutions that do not deviate significantly from ground truth. The use of RMSE is preferred over MSE in that the error has the same unit as the predictions.

Analysis

Data Exploration

The data used in this project are train.csv, and store.csv. Description of the features can also be found from Kaggle competition home page.

The train.csv contains 1017209 historical sales data from Rossmann stores, and it has the following features:

Store: Id of the store, 1 to 1115.
DayOfWeek: 1 to 7
Date: as we are not interested in forecasting future sales, this field is not really useful.
Sales: the turnover for any given day. This is what we should predict.

For the entire dataset, the statistics are:

count	1.017209e+06
mean	5.773819e+03
std	3.849926e+03
min	0.000000e+00
25%	3.727000e+03
50%	5.744000e+03
75%	7.856000e+03
max	4.155100e+04

The min sales is 0 as some entries are recorded when the store is closed with no sales.
We are not really interested in these entries and they will be removed from the dataset.

Customers: number of customers on a given day.

For the entire dataset, the statistics are:

count	1.017209e+06
mean	6.331459e+02
std	4.644117e+02
min	0.000000e+00
25%	4.050000e+02
50%	6.090000e+02
75%	8.370000e+02
max	7.388000e+03

Again the 0 customers entries are due to stores not open.

Open: binary indicator on whether the store is open (=1) or not.
Promo: indicates whether a store is running a promotion on that day

StateHoliday: indicates a state holiday: a = public holiday, b = Easter holiday, c = Christmas
SchoolHoliday: indicates if the (Store, Date) was affected by the closure of public schools

The first five rows from train.csv are sampled as follows:

Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	\
1	5	2015-07-31	5263	555	1	1	0	
2	5	2015-07-31	6064	625	1	1	0	
3	5	2015-07-31	8314	821	1	1	0	
4	5	2015-07-31	13995	1498	1	1	0	
5	5	2015-07-31	4822	559	1	1	0	

SchoolHoliday
1
1
1
1
1

Note that we are only going to use the entries when open = 1, as closed stores should always give 0 sales. Part of the train.csv will be reserved as test subset, and the remaining entries are used for training and validation/optimization.

The store.csv contains 1115 entries, each has supplementary information about the corresponding stores, and has the following features:

Store: store Id
StoreType: 4 different store models: a, b, c, d
Assortment: assortment level: a = basic, b = extra, c = extended
CompetitionDistance: distance in meters to the nearest competitor store
CompetitionOpenSinceMonth: approximate month of when the competitor opened
CompetitionOpenSinceYear: approximate year when the competitor opened
Promo2: 0 = store is not participating, 1 = store is participating
Promo2SinceWeek: calendar week when the store started participating in Promo2
Promo2SinceYear: year when the store started participating in Promo2
PromoInterval: describes the consecutive intervals Promo2 is started

For each store, we also add two more features, namely the mean sales and median sales for the store. The mean and median are calculated only based on open stores.

Mean Sales

count	1115.000000
mean	6934.208449

std 2383.911051
min 2703.736573
25% 5322.299969
50% 6589.948470
75% 7964.200644
max 21757.483418

Median Sales
count 1115.000000
mean 6674.630942
std 2314.513962
min 2293.000000
25% 5106.250000
50% 6332.000000
75% 7622.750000
max 21872.000000

Accounting only for open stores, the sales statistics are:
count 844392.000000
mean 6955.514291
std 3104.214680
min 0.000000
25% 4859.000000
50% 6369.000000
75% 8360.000000
max 41551.000000

The above three statistics match well.

The first five rows are sampled from store.csv as follows, which shows the data contains textual features, and features with missing values.

Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	\
1	c	a	1270	9	
2	a	a	570	11	
3	a	a	14130	12	
4	c	c	620	9	
5	a	a	29910	4	

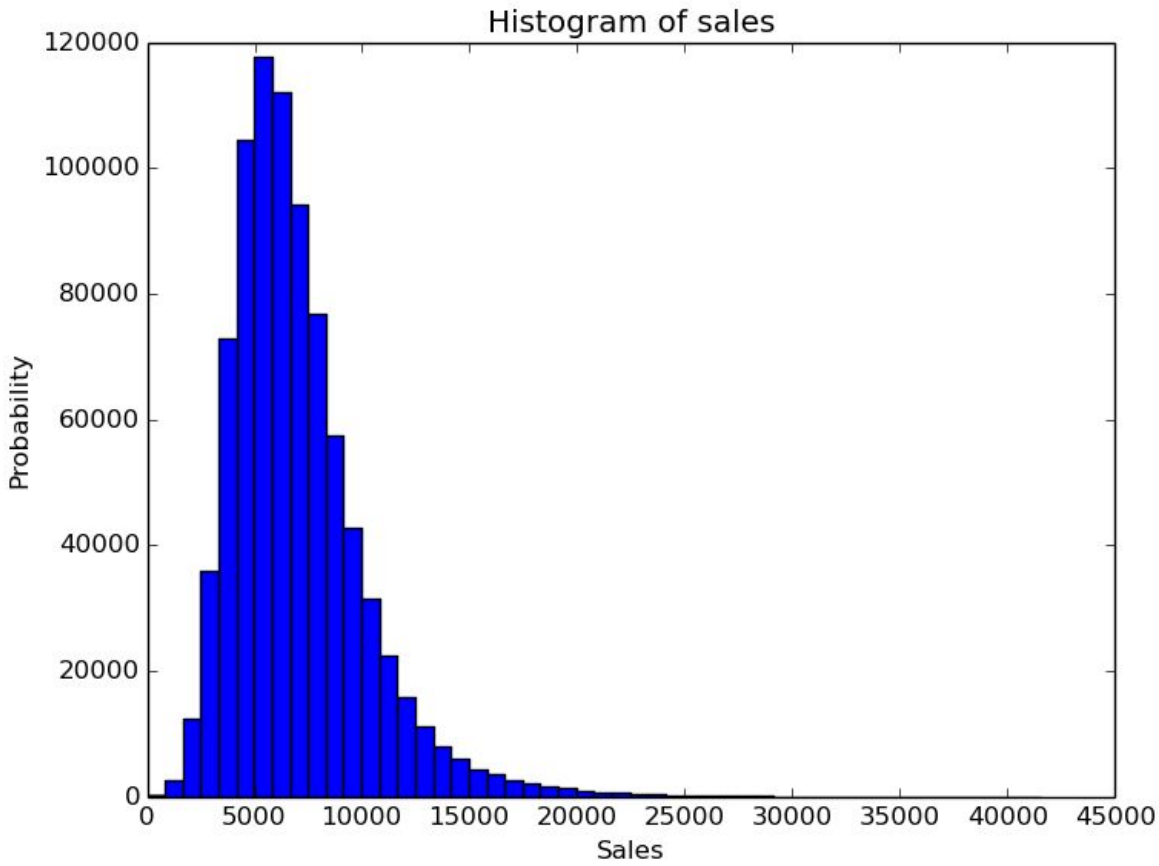
CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	\
2008	0	NaN	NaN	
2007	1	13	2010	
2006	1	14	2011	
2009	0	NaN	NaN	
2015	0	NaN	NaN	

PromoInterval
NaN

Jan,Apr,Jul,Oct
Jan,Apr,Jul,Oct
NaN
NaN

Exploratory Visualization

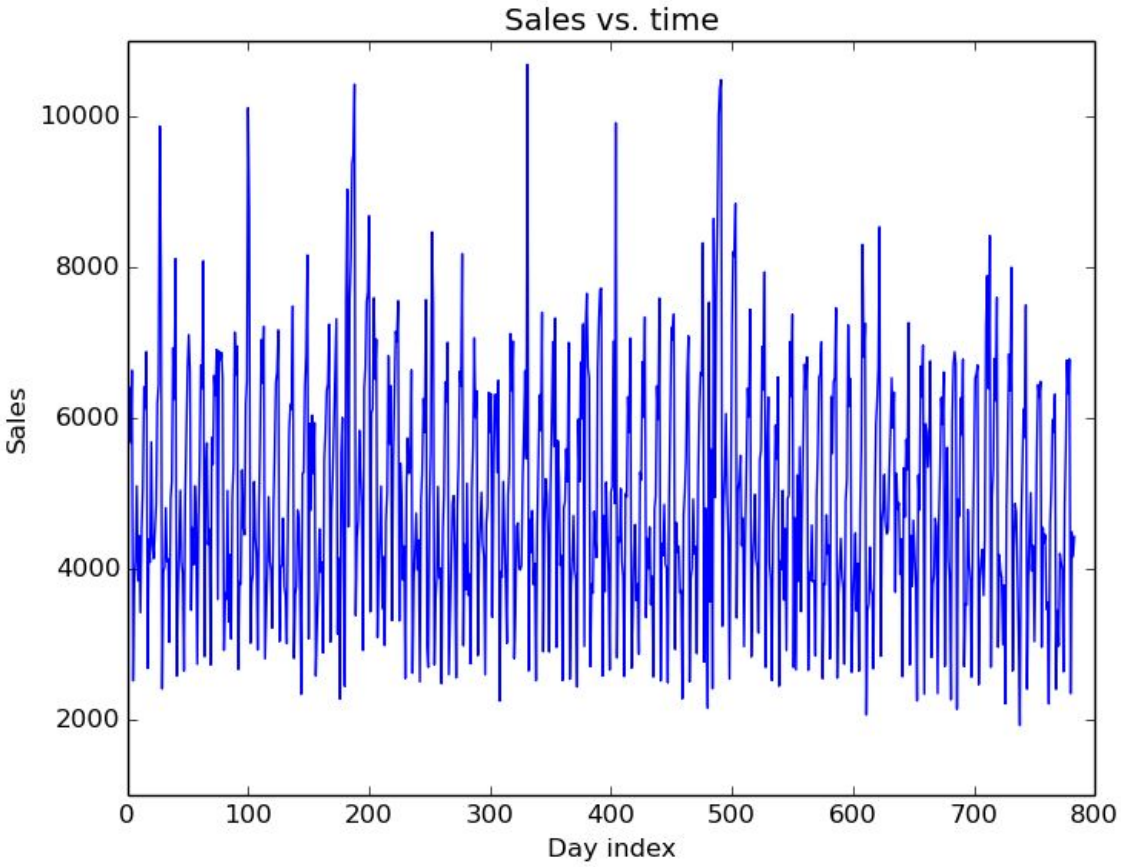
First we look at the distribution of sales.



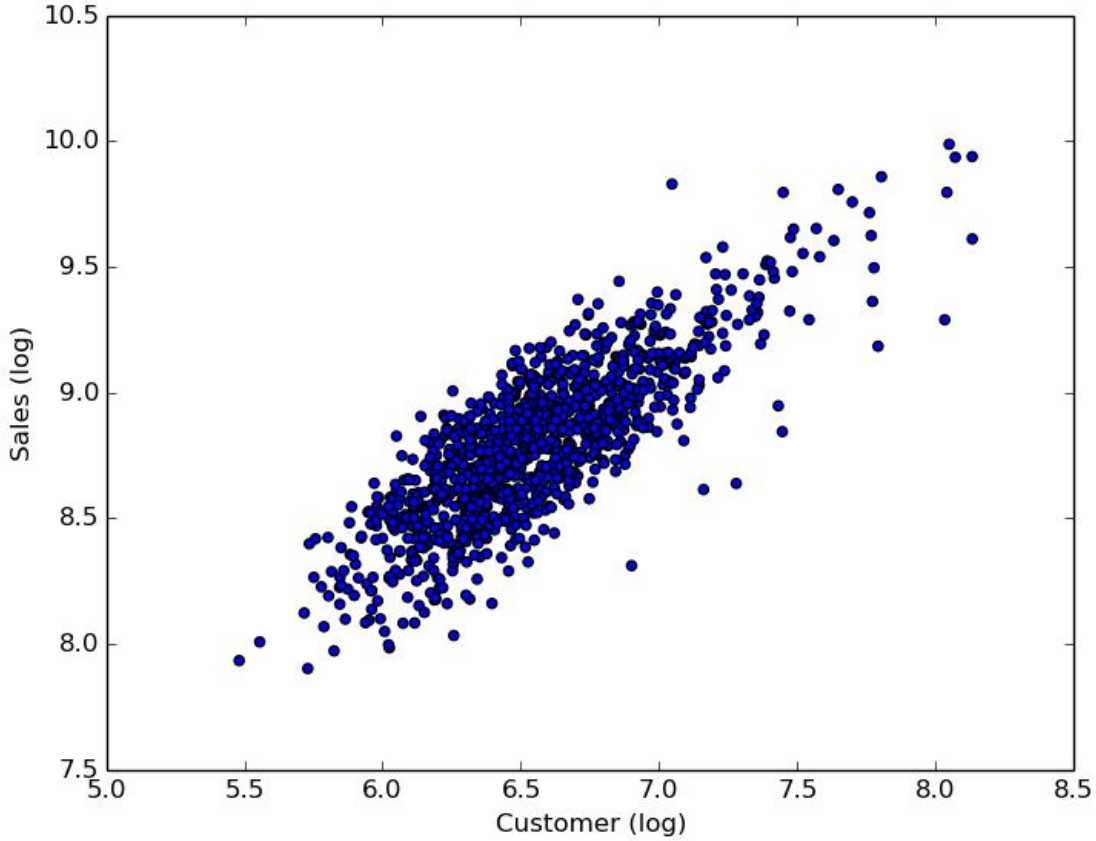
The histogram plot above shows that the distribution of sales is almost balanced, except a slightly heavy tail to the right. This is largely because the left end is lower bounded by zero, as it is impossible to have negative sales. The right hand side, on the other hand, is unbounded.

An interesting observation can be made when sale is plotted against the day index: there is a periodic pattern in sales, as shown in the figure below for the sales against day index for store 2. For some days the sales are significantly higher than the sales on others. This pattern may be explained by the regular shopping behavior that people buy more on

weekends for the entire week's supply, highlighting the importance of week of the day feature in sales prediction.



It is expected that some features are correlated with the sales. As an example, the average sales per store is plotted against the average number of customers for the store.



As expected, the mean sales increase with increasing number of customers, and there is a clear linear relationship between the log of number customers and the log of mean sales.

Algorithms and Techniques

From data exploration, the input data contains categorical, discrete, and continuous types, and therefore tree-based algorithms are suitable. The decision tree regressor is selected to be the algorithm used. Besides the advantage that decision tree requires little data preparation to handle both numerical and categorical data, decision tree can be interpreted easily.

Decision tree works by splitting into branches based on the results from a series of tests over the attributes. The series of tests may be designed by finding the feature that maximize the information gain in each split. Each branch of the tree represents one outcome of the

test, and data points yielding the same test results belong to the same leaf. When used for regression, the data points on the same leaf are averaged to yield the prediction.

Some of the default parameters that are of interest for the decision tree regressor are:

max_depth=None,
min_samples_split=2,
min_samples_leaf=1,

Besides decision tree as the basic form of tree algorithms, there are other enhanced versions such as random forest, but due to resource limitation, these advanced algorithms are not used in this report.

Benchmark

The per-store mean and median sales are used as benchmark result. The can be obtained from the original data by first grouping all entries for a store and then calculating the mean and median.

Methodology

Data Preprocessing

The preprocessing step is handled by preprocess.py.

The two csv files are merged using store Id as key. Then the merged data is filtered such that closed stores are removed. Initially the merged data has 20 columns (without Sales label column) and 844392 rows, summarized as follows:

```
Store                       844392 non-null int64
DayOfWeek                   844392 non-null int64
Date                        844392 non-null object
Sales                       844392 non-null int64
Customers                   844392 non-null int64
Open                        844392 non-null int64
Promo                       844392 non-null int64
StateHoliday                844392 non-null object
SchoolHoliday               844392 non-null int64
MeanSales                   844392 non-null float64
MedianSales                 844392 non-null float64
StoreType                   844392 non-null object
Assortment                  844392 non-null object
CompetitionDistance         842206 non-null float64
CompetitionOpenSinceMonth   575773 non-null float64
CompetitionOpenSinceYear    575773 non-null float64
Promo2                      844392 non-null int64
Promo2SinceWeek             421085 non-null float64
Promo2SinceYear             421085 non-null float64
PromoInterval               421085 non-null object
```

Note that some features ('CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval') has more than 30% of missing values, and in this project these features are dropped, as they are not critical features like number of customers, and can be removed to simplify the model.

In addition, the ‘Open’ feature is not useful any more, as all the filtered data has Opne = 1. The ‘Date’ and ‘Store’ features are also removed.

The textual features ('StoreType', 'Assortment', and 'StateHoliday') are also converted to numerical values, representing categorical index.

The missing values in the remaining dataset (namely, CompetitionDistance) are filled with value of -99, which represent a special class that does not have meaningful data.

After the above processing, the data is summarized as follows:

```
DayOfWeek            844392 non-null int64
Customers            844392 non-null int64
Promo                844392 non-null int64
StateHoliday         844392 non-null int64
SchoolHoliday        844392 non-null int64
MeanSales            844392 non-null float64
MedianSales          844392 non-null float64
StoreType            844392 non-null int64
Assortment           844392 non-null int64
CompetitionDistance  844392 non-null float64
Promo2               844392 non-null int64
```

No scaling is performed over the data, as decision tree does not require such processing.

Implementation

The data is first preprocessed before actual modeling.

The per-store mean and median sales are calculated during the preprocessing step, where the sales are grouped by store ID and the mean and median for each group are calculated. Two new columns, MeanSales and MedianSales, are created and joined to the dataframe in cal_per_store_sales_summary.

Some features like 'StoreType', 'Assortment', and 'StateHoliday' are textual, and they are converted to numerical values by assigning each class an index in map_txt_to_num.

The data when the store is closed are dropped as the sales are always 0 in this case.

One complication in implementation is some features have missing values. While these feature may still convey useful information, it is hard to handle features with a large number of missing values. In this project, the feature with more than 30% missing values are dropped, and the missing values in the remaining features are set to a dummy flag value.

Subsequent to data preprocessing, the modeling part is handled by model.py.

The input data is split into training and test subset. This is another complication of the project, as the data is in time order. To avoid look ahead bias, the data is not split randomly, but rather,

the first 70% of the data is used for training, and the remaining 30% of the data is used as test subset.

The RMSE is chosen as the performance metric, and it is implemented in performance_metric.Basically, it leverages on the mean_squared_error of sklearn by taking the square root of the MSE.

The main algorithm evaluates the performance of benchmark using mean sales and median sales in cal_baseline. No training is required, as the per-store mean and median sales have been calculated during preprocessing and can be obtained directly from the corresponding columns in the data frame.

The untuned decision tree is first trained in basic_training by fitting the model to the training subset, and the fitted model is used for prediction.

The model is tuned in tuned_training, which searches for the optimum parameter. The model tuning process is described in more details in the subsequent section on refinement.

A fixed random seed is set globally throughout the code.

Refinement

The RMSE for the two baselines are 1949.92804 and 1973.51584679 for the mean and median respectively.

Initially the decision tree algorithm uses the default setting with max_depth set to None (which means the nodes are expanded until all leaves are pure), and the RMSE for untuned decision tree is 1710.94632837. The RMSE for the basic decision tree is over 200 smaller than baseline methods.

The maximum depth of the decision tree is then tuned. The core implementation of model tuning is in grid_search. The grid_search from sklearn is not used, but a proprietary implementation that assigns the first 80% of the data to training and last 20% of the data to validation is used. The parameter to be optimized is max_depth. Admittedly there are other parameters which could also be optimized, but due to resource constraint, only the single parameter is selected for optimization in model tuning. For each max_depth, the basic_training function is used to fit and predict the data with the training and validation subsets, and the RMSE for each max_depth is calculated and recorded based on the validation set. The max_depth that yields the minimum RMSE is selected.

In grid search, the range of max depth searched is from 2 to 20, and the optimum max depth is found to be 10, which yields RMSE of 1219.39947311, a further 500 reduction from the untuned model.

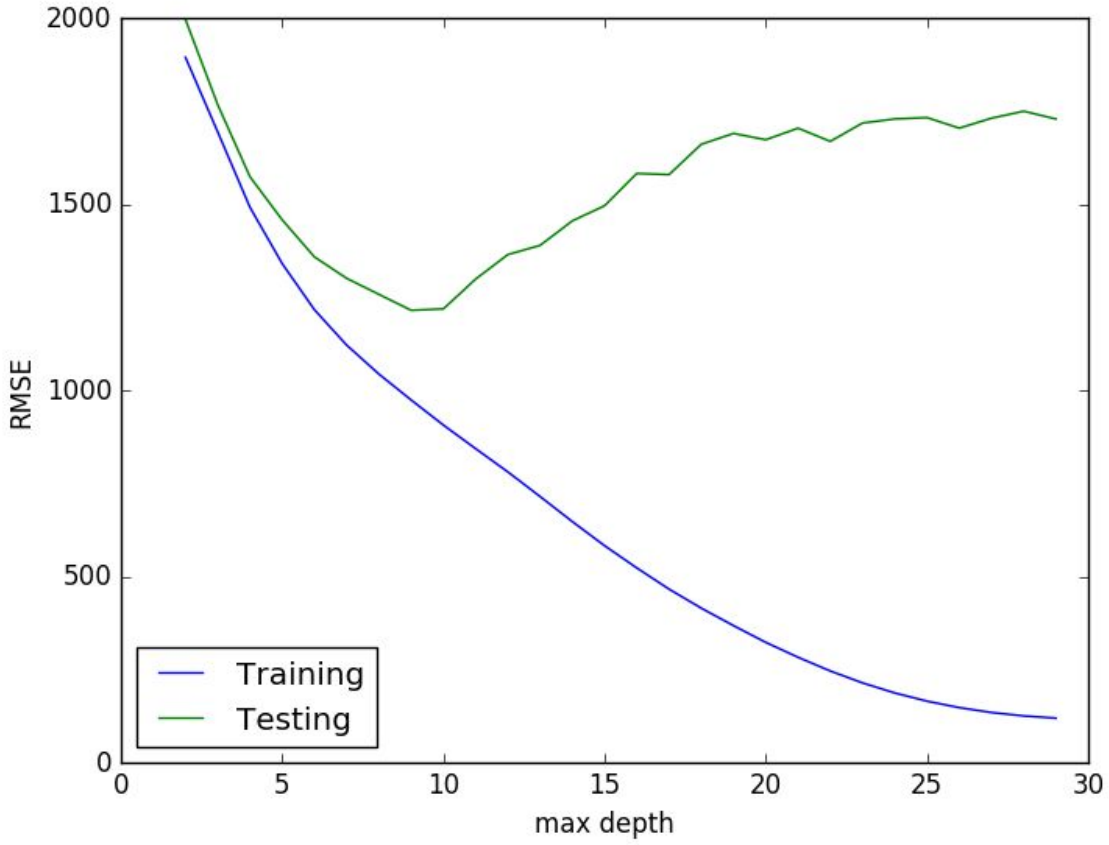
The results are summarized below:

	Baseline (mean)	Baseline (median)	Decision tree (default, max_depth = None)	Decision tree (tuned, max depth = 10)
RMSE	1949.92804	1973.51584679	1710.94632837	1219.39947311

Results

Model Evaluation and Validation

The RMSE for training and testing data at various max_depth values from 2 to 30 are shown below. The turning point where the testing error values is when max_depth is 9, which best addresses the trade-off between overfitting and underfitting, and is close to the max_depth from model tuning, which is 10.



Six entries were selected from the original dataset. The model is then trained by 10 different random splits, and prediction are made based on the 10 trained models over the 6 test data point.

The ground truth for the 6 test data is:

Data index	Sales
1	5263
2	5020
3	4782
4	5011
5	6102
6	4364

For the 10 random splits, the following predicted sales are obtained:

Split no. \index	1	2	3	4	5	6
1	4994	4623	4944	4492	5627	4177
2	4610	4610	4868	4908	5609	4118
3	4663	4663	4256	4492	5509	4106
4	4937	4665	4665	5565	5548	3911
5	4634	4634	4813	4492	5442	4078
6	4822	4681	4516	5047	5512	4129
7	4758	4581	4581	4492	5664	3994
8	4678	4678	4434	4492	5482	4069
9	4685	4685	4478	4492	5806	4169
10	4333	4579	4946	5475	5604	4119

The prediction does not deviate significantly from the ground truth, and the model is considered consistent.

Justification

The following result was obtained for the baseline, untuned decision tree with default parameters, and decision tree with tuned max_depth:

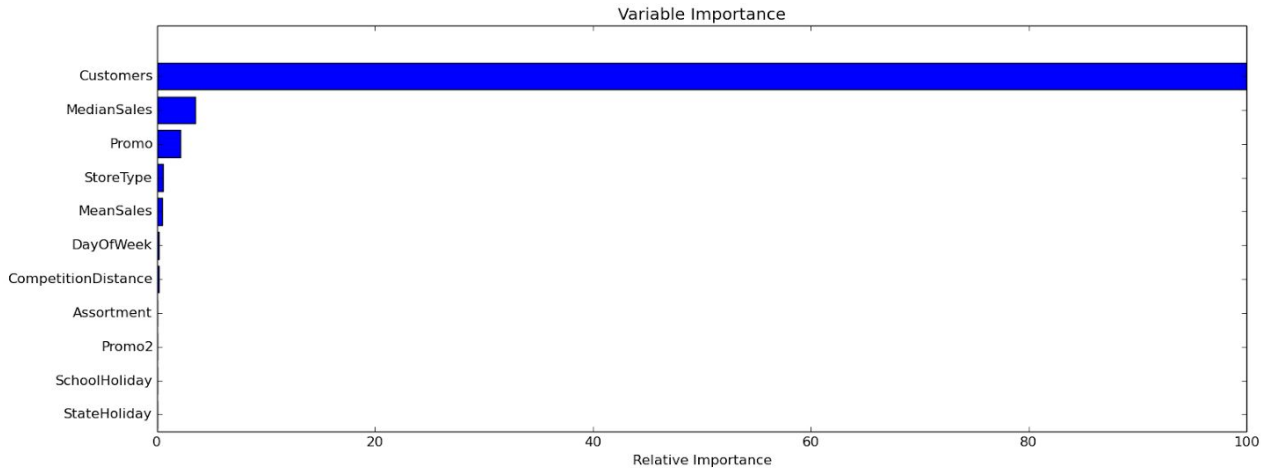
	Baseline (mean)	Baseline (median)	Decision tree (default, max_depth = None)	Decision tree (tuned, max depth = 10)
RMSE	1949.92804	1973.51584679	1710.94632837	1219.39947311

The RMSE for the basic decision tree is about 200 smaller than baseline methods. The tuned model's RMSE is further reduced by about 500 from the basic untuned model.

Conclusion

Free-Form Visualization

The relative importance of the features are plotted below, which shows that the number of customers is the most important feature. Intuitively this makes sense as more customers are likely to yield higher sales. The positive correlation between the number of customers and the sales are also evidenced from the plot earlier. Median sales is the second most important feature, and it is more important than mean sales, perhaps due to its robustness against fluctuation of the sales at both ends. This also implies that median sales could characterize a store. The third most important feature is promo, which is also justifiable from a business perspective.



Reflection

This project aims to predict sales given a set of features. The given dataset contains both numerical and categorical data, which makes tree-based method an ideal fit. Due to resource constraint, decision tree regressor is selected.

The baseline benchmark are the median and mean sales of each store, which are calculated during data preprocessing, and added to the data frame as additional feature columns.

In the model training and evaluation stage, the baseline performance is first calculated, followed by untuned decision tree training and evaluation. The decision tree model is then tuned by optimizing the maximum depth parameter.

Feature engineering may be the most challenging part of the project. The features come from different sources and needs to be combined. There are also missing data in the original features. The approach adopted is to drop features that have significant portion of missing values. For features with only a few missing values, the missing values are replaced by a dummy flag value.

The other interesting aspect of the project is on how training and testing subsets are generated. 30% of the data is reserved as testing set, and model training and optimization are based on the other 70%. The training and testing subsets are split such that the first 70% rows are used for training and the remaining 30% rows are used for testing. Random splitting is not used so as to avoid look ahead bias.

Improvement

It is known that the decision tree easily overfits, and typically ensemble methods may be used to enhance tree-based algorithm performance. However, due to resource limitation, ensemble methods such as random forest is not implemented in this project, but they may be explored for future extensions.

Furthermore, the periodic sales pattern discussed earlier suggests time series based techniques to extract the seasonal factors for sales prediction.

Overall accuracy of the regressor can be improved by techniques such as stacking, which is commonly used in Kaggle competitions, where predictions from multiple regressors are combined (e.g. by linear regression) to give the final prediction. From the result in sensitivity analysis, the predicted price seems to be lower biased. This bias term may also be compensated in the stacking stage.