Capstone Project

Predicting sales of Rossman stores

Machine Learning Engineer Nanodegree

Chirag Jhamb

16 August 2016

I. Definition

Problem Overview:

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.

The goal of this project is create a model that predicts 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, managers can stay focused on what's most important to them: their customers and their teams, instead on worrying about profits.

For this probelm, Rossman has provided datasets "train.csv" and "stores.csv" containing sales information on daily basis and information of various stores, respectively. There is also a test set

containing features similar to that of “train.csv” but without the feature “Sales”.

Since all the features are given along with result, this is a Supervised learning problem.

Note: The actual training dataset contains 1017209 rows. Since the data was too large, only 75% of the data was used. The method used can be seen in the file “create_files.py”.

Problem Statement:

Primary goal: Given historical sales data for 1,115 Rossmann stores. The task is to forecast the "Sales" column for the test set. The data can be found here.

For achieving the primary goal, training data will be divided into two parts, first half will contain 75% of the data and the other half that will to test the accuracy of predictions made by the model designed and trained over the first set. For the creation of model, several regression algorithms will be used such as decisiontree, gradientboost etc.. The model having the highest accuracy and the lowest cost(time consumption) will be chosen and will be trained over the entire dataset and then that model will be used to make predictions over the “test.csv” file.

The solution must have precise sale values that each store will achieve in the future.

Metrics

By now, we know this is a regression problem. For testing the accuracy of each model we create a test set by dividing the training data.

Then we use root mean squared prediction error to find out the accuracy of our predictions. The root-mean-square error (RMSE), or RMSD is a frequently used measure of the differences between values (test and train values) predicted by a model or an estimator

and the values actually observed. The RMSE represents the sample standard deviation of the differences between predicted values and actual values. Hence 0 implies a perfect score.

Mathematically, $\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}$.

II. Analysis

Data Exploration

The data is present in the “inputs” folder of the repository. There are three files- “train.csv” containing the records about the sales that is to be used as training data, “store.csv” containing more information about store and the features affecting the sales, which can be merged with the training data and “test.csv” containing records of each day but not the sales, and predicting sales is the goal of this project.

These are all the features given in the dataset along with a short description of each-

•**Id** - an Id that represents a (Store, Date) duple within the test set

•**Store** - a unique Id for each store
•**Sales** - the turnover for any given day (this is what you are predicting)
•**Customers** - the number of customers on a given day
•**Open** - an indicator for whether the store was open: 0 = closed, 1 = open
•**StateHoliday** - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
•**SchoolHoliday** - indicates if the (Store, Date) was affected by the closure of public schools
•**DayOfWeek** – the day of the week
•**StoreType** - differentiates between 4 different store models: a, b, c, d
•**Assortment** - describes an assortment level: a = basic, b = extra, c = extended
•**CompetitionDistance** - distance in meters to the nearest competitor store
•**CompetitionOpenSince[Month/Year]** - gives the approximate year and month of the time the nearest competitor was opened
•**Promo** - indicates whether a store is running a promo on that day

·**Promo2** - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating
·**Promo2Since[Year/Week]** - describes the year and calendar week when the store started participating in Promo2
·**PromoInterval** - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

We can expect outliers to be in features “Stores” and “Customers” of the training set as the rest of the values are categorical or in form of dates.

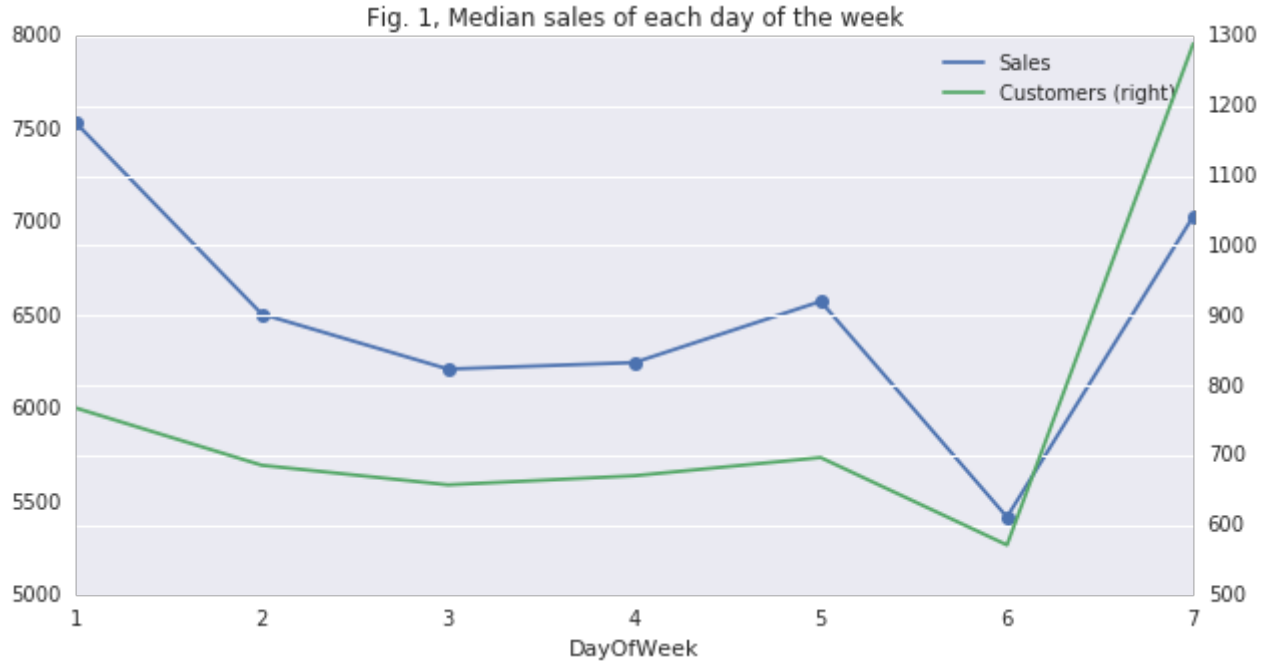
An interesting observation is that there are a lot of dates that need to be handled in the data. This could help us in getting a timeline view.

On some exploration, there were certain NaN values in columns such “Open”, “Promo2” etc. where the values should be either 1 or 0. In case of promotions it is safe to assume there were no promotions at all which is why the column hasn’t been filled. In case of features such as “Open” we can simply check if there has been any sales that day. If there has been no sales then it is safe to assume that the store was closed replace NaN with zero.

Exploratory Visualization:

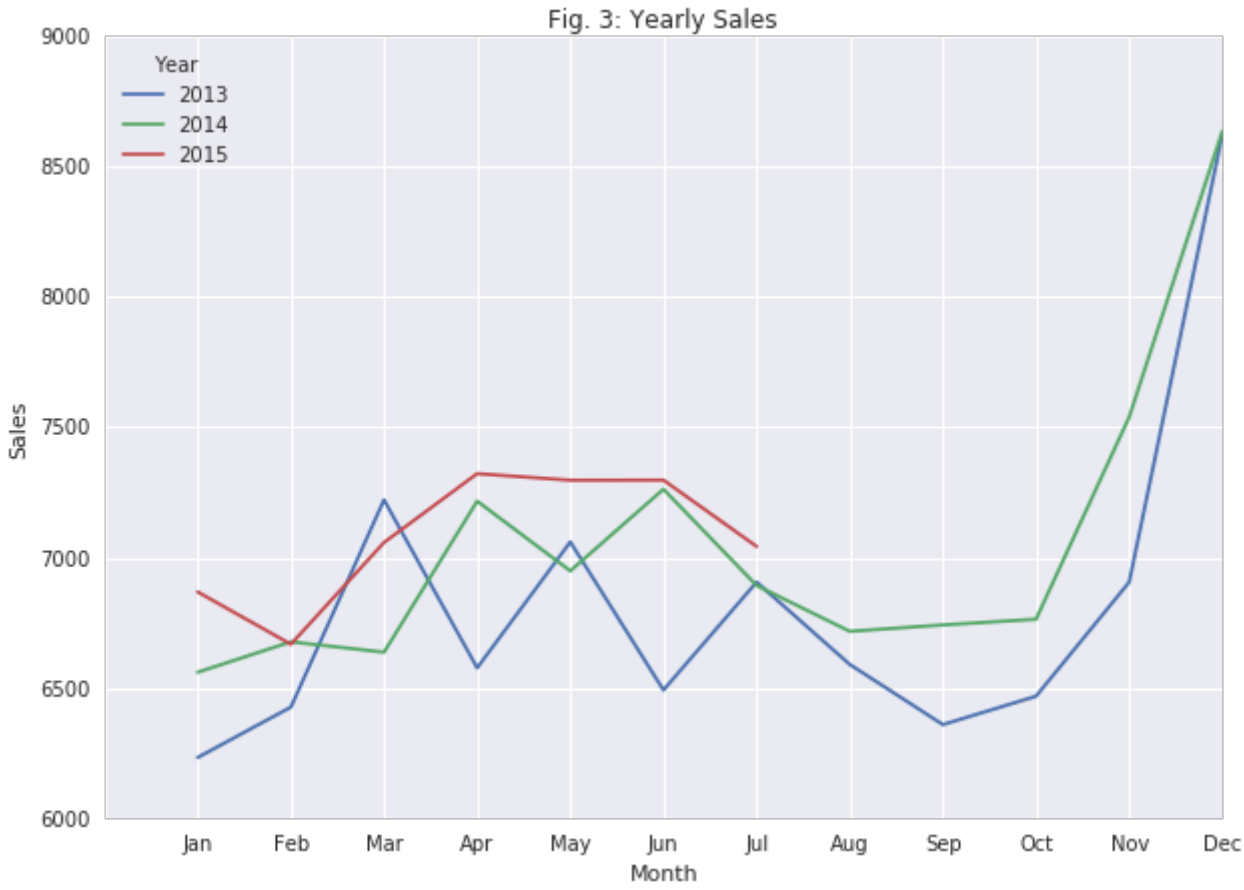
This part basically explores all the features by visualising the data and we use our intuition to predict the correlation between it.

We start by exploring the median of the sales and customers that stores get each day of the week and view it as a timeline for the week-



That is a very interesting plot. Let's assume day-1 is Monday and day-7 is Sunday. Throughout the week until Friday, less customers are ringing in more sales. On Saturday, both the number of sales and customers drop to the lowest and spike again on Sunday. However, only on Sunday there is an opposite behaviour i.e. more customers are ringing in less sales, in other words there is more customer traffic but less sales. There is only one conclusion from this plot, the customers who walk into the store during the week are serious customers who will most likely make a sale whereas customers walking in on Sunday appear to be

distracted/disinterested/window-shopping! One area of improvement would be to look at converting the customer numbers to sales on the weekend, may be targeted promotions, etc.

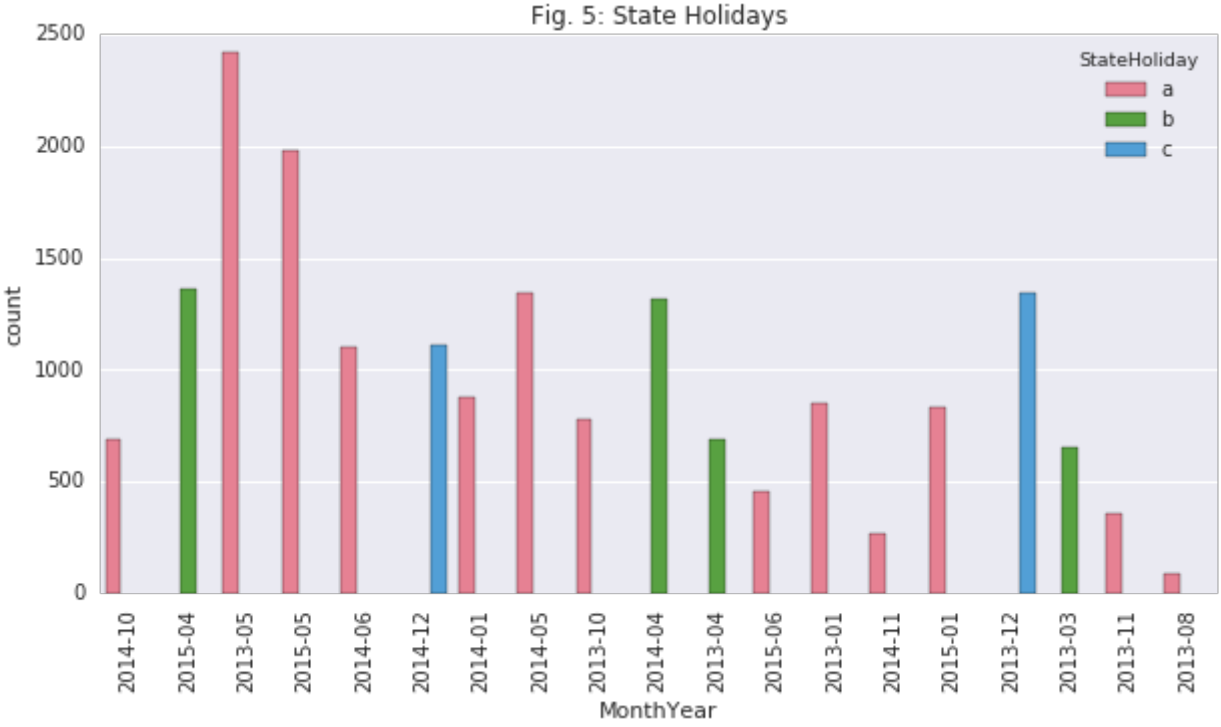


Now lets look at the performance of the stores over the entire timeline, from the starting months present in training data to the last-

The figures 4a and 4b measure performance of stores and share the same x-axis. First and formost thing that is obvious from the plot is that sales and customers are highly correlated. Therefore, majority of customers walking through the store are contributing to sales. However, starting in year 2015, customer numbers are diverging slightly away from sales with less customers contributing to more sales. Customer growth is not evident. Sales and customer numbers appears to spike just before Christmas and fall back down again during the new year. If more customers could be enticed into the

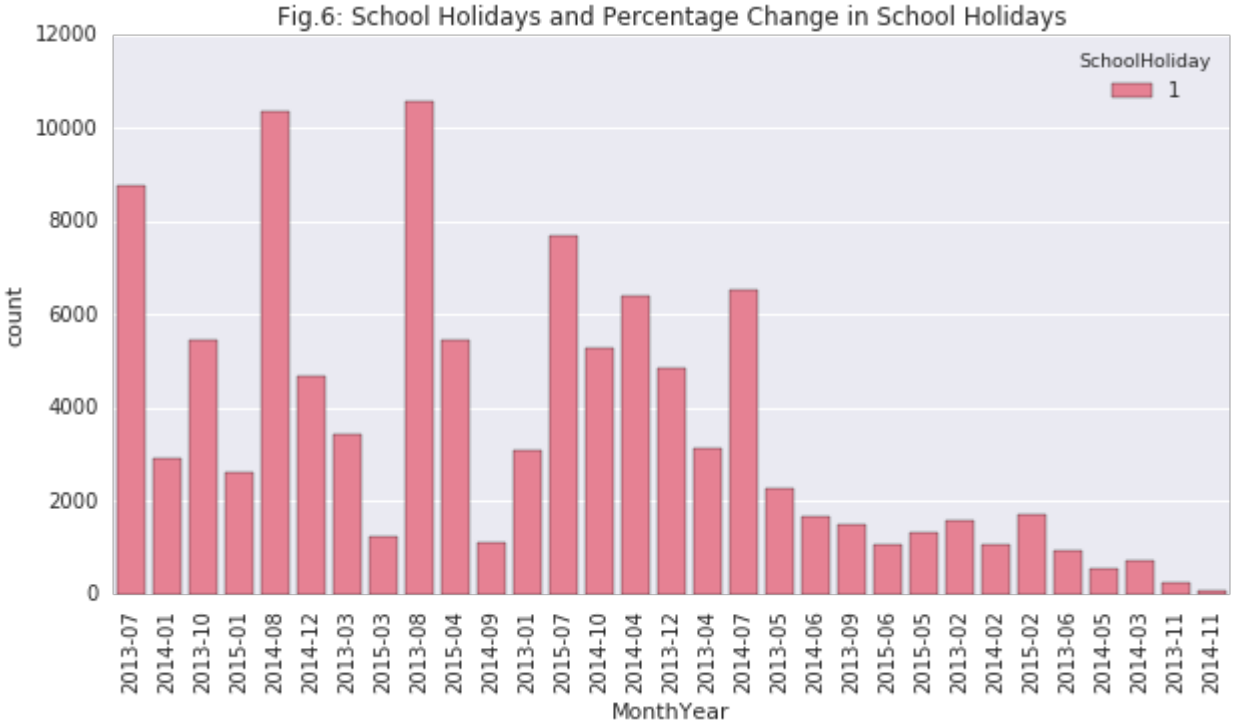
store, better sales could be acheived. Let's plot the sales and customer data with promos, stateholiday and schoolholiday to visualise their behaviour for those days.

Fig. 5: State Holidays



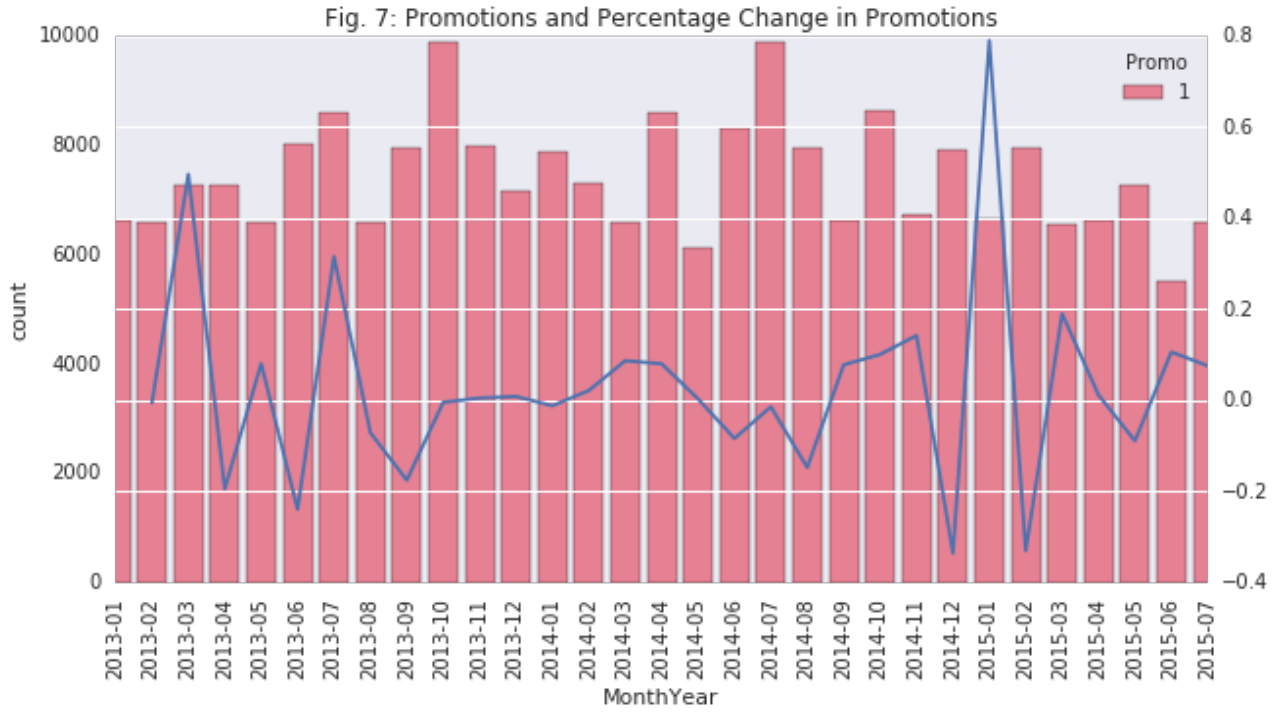
State Holidays (Fig. 5)

I think there is something amiss about the "StateHoliday" data. From August 2014, there appears to be no StateHoliday at all, which is highly unlikely, most likely it is the case of "missing data". That can be fixed by filling the values using previous years StateHolidays, as StateHolidays rarely change.



School Holidays (Fig. 6)

SchoolHolidays appears to be seasonal, with July-August being the highest, may be July-August is the summer vacation time. Interesting observation is that there are minimal/least school holidays in November i.e. before Christmas vacation. Another observation is the decreasing trend of school holidays reaching a minimum point just before the vacation starts. On the other hand, after the vacation when the school restarts there is a increasing trend of school holidays with an abrupt drop after which the decreasing trend takes on. This abrupt drop might be the school restart after a mid-term break.



Promotions (Fig. 7)

When we look at the number of promotions being held, it appears that every month there were promotions at various stores. An interesting observation is that there is a massive drop in sales/customers in January, 2014 and there were no promotions subsequent to these drop in numbers which clearly suggests that the management were unable to predict this drop in numbers. If they had predicted the drop they would have organised more promotions to lure in the customers. This is evident from the January, 2015 , where they have organised more promotions in January, 2015 learning from their mistakes from 2014. Subsequently, there appears to be more sales, albeit small change compared to Jan-Feb 2014, with same number of customers. They have organised more promotions during School holidays but it has not had much positive impact on sales or customer numbers. Again, another interesting observation is sales always has an increasing uptrend during the days when the schools are in session.

Algorithms and Techniques

It is obvious from the above section of exploration that Sales cannot be predicted just by using customer data or just the promotions. The sales are affected by each and every attribute. In order to make the prediction of sales, first we will get rid of the outliers in the Customers feature and the Sales feature as there might be exceptional days when the numbers are too high or too low. Such days will affect the precision of the model.

After dealing with outliers, we can start preprocessing the data. This includes getting rid of Null values, label encoding features like StoreType, StateHoliday etc.. As discussed earlier conversion of date is going to be important for prediction which is also proved in the visualisations as sales change a lot with dates, months and days.

We can also process Competition's details as it would definitely affect the number of customers which would indeed affect the sales. Promotions might help in getting the customers back, which is why we need to encode all the Promo columns as well.

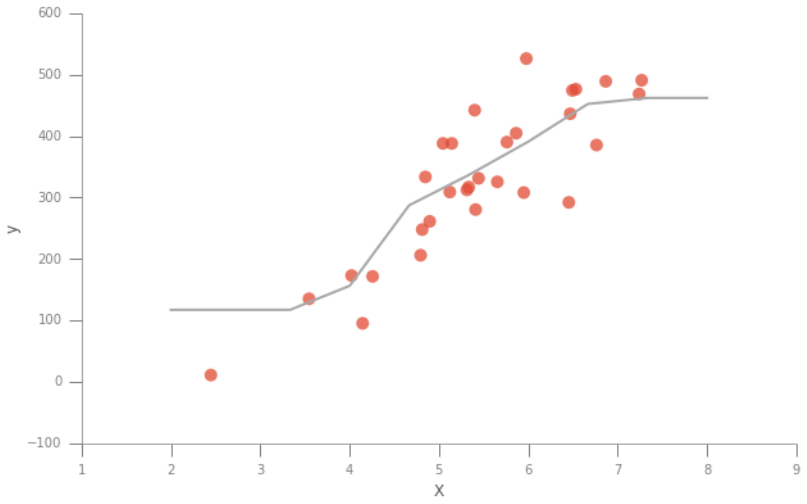
Once we have preprocessed the data, we can split it using cross_validation.train_test_split method. This method randomly shuffles the data and return two sets, training and testing. The size of the test can be defined.

The training set then is used to train three models-

1. DecisionTree regression- The goal of this model is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. DecisionTree takes features and work out a way to get to the target variable using if-then-else decision rules on the input features.

2. Kneighbour regression- The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. K neighbour regression implements learning based on the k-nearest

neighbors of each query point, where k is an integer value specified by the user. Example of regression made by Kneighbour-



3. GradientBoost regressor- Gradient Boosted Regression is a generalization of boosting to arbitrary differentiable loss functions. Gradient boosting produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrarydifferentiable loss function.

These models have been chosen for this problem because of the following properties-

- Given the features affecting the sales, the data can be easily split into deciding chaterstics using if-then-else decision rules on input features. This requires data prepration which can be done easily, so we do not need to worry about the main shortcoming here. Though decision tree can be unstable, it is very good at handling categorical data as well as numerical. Hence the multiple types of features in our dataset won't be a problem. We can find out if it's unstable for our data only by looking at the test score of the model and then decide if it can be used or not.

- Our dataset has a large number of datapoints. Kneighbours uses brute force to perform fast computation which helps us in reducing the cost of the model. On the other hand, it is good only in case of continous values which is not the case for some of the input features. We can find out how much that affects the accuracy of the model by the test scores.
- GradientBoost is a slow model, hence the cost of the model is going to be high. On the other hand, it uses a different approach by optimization of an arbitrarydifferentiable loss function. If we are unable to get a good score by any other model we might be able to get a good score through this approach.

Benchmark

Given that we are going to be experimenting over 3 models, I expect at lease one model to have a good RMSE score, note that in case of RMSE the lower the better with 0 being the perfect score.

To have a good idea about the Sales the store managers need to have certain amount of belief in the accuracy of the model. Certain rate of error can be accepted but if the error is almost as much as one third then the model is no good. With experience the store managers would be able to predict the with that much error rate. If the models are not predicting at least one third of the data correctly then the data requires more processing and the selected models need to be optimized or changed. Hence the benchmark of RMSE should be 0.33.

III. Methodology

Data Processing-

First we start by handling the outliers. Detecting outliers in the data is extremely important in the data preprocessing step of any analysis. The presence of outliers can often skew results which take into consideration these data points. There are many "rules of thumb" for what constitutes an outlier in a dataset. Here, we will use Tukey's Method for identfying outliers: An *outlier step* is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal.

I therefore wrote code which finds out outliers in customers and in sales and then sees which outliers are common in both the features. Then the common outliers are dropped. Here is the result-

```
Number of outliers in sales = 15863
Number of outliers in customers = 22600
Number of common outliers = 11334
Dropping common outliers...
1.90838602133 %  of data dropped
```

As it can be seen, the number of common outliers is 11334. This might seem like a high number but when we compare it the number of data points given to us its only 1.9% of the data. Thus, it is feasible dropping these many points.

As discussed earlier, data requires moderate amount of processing. Only some features having numerical values were used directly- 'Store', 'CompetitionDistance', 'Promo', 'Promo2', 'SchoolHoliday'.

The initial step of processing the data was to fill all the NaN values with zeroes. Here we assume that the column was not filled out because of abcense of that feature. Then to make the processing

faster, I drop the rows where the the store is closed, that is where open is set to zero as we are only interested in training the models for days when the store was open and hence there were any sales. Then features having categorical values 'StoreType', 'Assortment' and 'StateHoliday' have all the values replaced by lables which can be used in models.

After that we move to the dates. The date format given is aribtrary and requires to be worked upon. Thus all the dates are divided into the features- 'DayOfWeek', 'Month', 'Day', 'Year', 'WeekOfYear'. Then process the date of Competition feature which is given either in year or month. We convert all the values into months to have one single unit for comparison. The same step is repeated in case of “PromoOpen” which is given since year, since week etc.. In the end, “IsPromoMonth” mapped with month values and is assigned 0 and 1 depending on the value.

To make model creation a little more faster, all the rows having Sales as 0 is dropped as it is probably an unfilled value which will only affect the model negatively.

Implementation

First, the steps were divided into various functions to calculate the time taken by each function in order to check the cost.

The first function trains the classifier that it takes as an argument. It applies the fit method and reports the time. Then the second function runs the prediction over the training set itself and returns the root mean square error rate. It also returns the time taken to make predictions over the training set. Then the third function makes the prediction over the test set and reports the time and the score.

First, the sales data is converted into log values in order to make predictions easier. Then each of the models mentioned in the earlier section are called and passed as arguments on the functions and the time and score is reported for each.

Once the score is reported, the most efficient model is chosen and the feature importance is calculated. Feature importance tells us which features were the most relevant in making predictions. This can be compared to our analysis while exploring the data.

The feature importance is then represented on the bar graph.

Then the entire dataset is trained over the chosen model and the final predictions are made and saved to the test file.

Refinement

Decisiontree regressor was chosen for regression. Initially, the error rate was high. Hence the Sales data for training the model was converted into logs to decrease the error rate. The root mean squared error value over the test set was 0.1819, which is much

better than what I had expected. Then by applying GridSearchCV the error rate was reduced to 0.164. GridSearchCV exhaustively considers all parametric combinations that are passed in param grid. In this case, *leaf samples* and *sample split* values of the decision tree algorithm was optimised to get the optimum score.

IV. Results

Model Evaluation and Validation

Lets look at the results for each model-

For Decision Tree regressor-

Training a DecisionTreeRegressor using a training set size of 392592. . . Trained model in 6.3615 seconds Made predictions in 0.6946 seconds. mean_squared_error for training set: 0.0000.

Made predictions in 0.1140 seconds.

mean_squared_error for test set: 0.1819.

For Kneighbours-

Training a KNeighborsRegressor using a training set size of 392592. . .
Trained model in 3.6165 seconds
Made predictions in 23.1225 seconds.
mean_squared_error for training set: 0.1927.
Made predictions in 5.8234 seconds.
mean_squared_error for test set: 0.2470.

For GradientBoost regressor-

Training a GradientBoostingRegressor using a training set size of 392592. . .
Trained model in 71.3005 seconds
Made predictions in 1.1283 seconds.
mean_squared_error for training set: 0.3151.
Made predictions in 0.2588 seconds.
mean_squared_error for test set: 0.3181.

In case of Kneighbours, the cost of the model is low as expected, and the error rate is lower than the benchmark value that was decided. This is a good model but the error rate is relatively higher than the DecisionTree model, hence this model is not used for final predictions.

The GradientBoost regressor has extremely high cost for training the model which was expected but also gives the highest error rate. This model does not classify as the optimal model.

DecisionTree regressor is the clear winner, it has the lowest error rate and doesn't take too long to train. Even though training time is higher than Kneighbours, the time taken to make predictions is much lower. The benchmark value of RMSE was 0.33, and this model gave the value of 0.18 which is even better. Thereby making DecisionTree the most optimal model.

After optimisation of DecisionTree through gridsearch, the model's error rate was reduced to 0.16.

Justification

In the previous section, the benchmark value for the error rate was almost 0.33. Thanks to decisionTree regressor the error rate was almost half of the expected value. If the sales can be predicted with an error rate of only 0.16, then it would be very easy for the manager to make necessary changes and see what increases or decreases the sales precisely.

DecisonTree Regression created a model that predicts the value of a Sales by learning simple decision rules inferred from the data features. The error was so low because each feature was applied if-then-else decision rules to predict sales and since the features and numeric as well categorical the model was a perfect fit.

Thus, the entire dataset is trained in the end and Sales of the test set is predicted. It can be said with confidence that this model would have precisely predicted the required values.

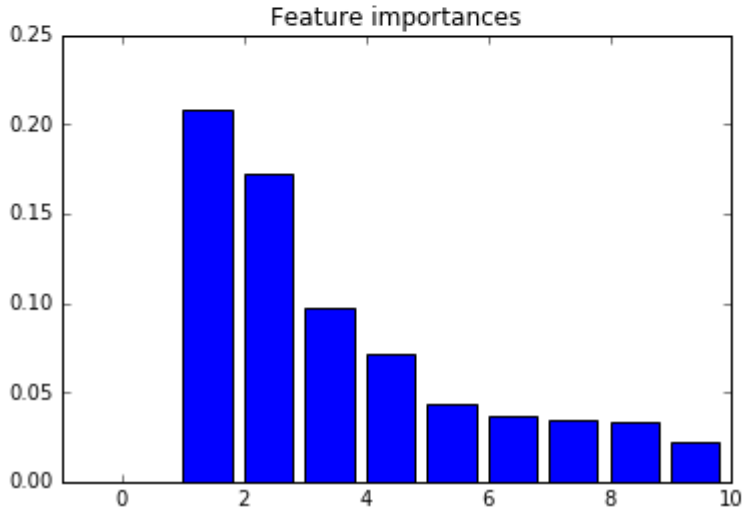
Hence, the task of this project is completed!

V. Conclusion

Free-Form Visualization

Finally, when the optimal model has been trained, we can look at which features were the ones that affected the importance the most and evaluate our prediction made earlier in this project. Lets look at the importances-

```
Feature ranking:
1. feature 1 :DayOfWeek (0.242784)
2. feature 0 :Store (0.207635)
3. feature 2 :Date (0.172682)
4. feature 13 :CompetitionOpenSinceYear (0.097464)
5. feature 8 :SchoolHoliday (0.071592)
6. feature 10 :Assortment (0.042990)
7. feature 5 :Open (0.036529)
8. feature 12 :CompetitionOpenSinceMonth (0.034023)
9. feature 14 :Promo2 (0.033737)
10. feature 6 :Promo (0.022285)
11. feature 3 :Sales (0.012412)
12. feature 9 :StoreType (0.011741)
13. feature 11 :CompetitionDistance (0.006378)
14. feature 4 :Customers (0.004225)
15. feature 15 :Promo2SinceWeek (0.002041)
```



Interestingly, the Store, DayOfWeek and Date were the most explored features which seemed to make the most difference on sales. This prediction was right.

On the other hand, holidays and promo also seemed to make a lot of difference but these features have been given lower ranks.

Reflection

The analysis of the project initially was an iteresting part of the project, as it was able to tell us which feature affects the sales almost as precisely as the feature_importance property of the decisionTree regressor told us. There were difficulties in visualising the data as the data was not preprocessed. There were a lot of NaN values in the data which were degrading the quality of outputs at almost every phase.

I expected final model to be less time consuming but 6-7 seconds of training time is still acceptable. Optimising the model was a challenge as there were indexing errors which were dealt with (as commented in the code).

Now this model can be used for predicting Sales and even if more stores or different businesses come along, if there are similar features then this model can be used for proper predictions.

Improvement

I do believe that the features given were enough for making an optimal precison model. There might certain algorithms such XGBoost which might do a better job of prediction than decisiontree if fully optimised, but it can't be said for sure. If the final solution was used as a benchmark, then it would be tough to beat but there has been better models having ever lower error rates. Thus, there is wide possibility of improvement which is followed by more research on regression models.