Rossmann Sales Analysis

Zhijie Ren, Peiyu Wang

Abstract - In real life, we often find that our machine learning algorithm model does not run as ideal as we want. There are many reasons which may cause this. As rigorous data scientists, we need to try different methods and rule out the wrong guesses. We used some of the methods in our model and analysis and we want to share our experience by taking this dataset as an example. We hope we can help people who are facing the same problem as us.

I. DATASET INTRODUCTION

The datasets we chose for the final project are Rossmann Store Sales data from Kaggle. They are historical sales data for 1115 Rossmann Stores. They consist of 3 datasets, store.csv, train.csv and test.csv.

Store.csv contains information about the 1115 Rossmann Stores including Store Number, StoreType, Assortment, CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2, Promo2SinceWeek, Promo2SinceYear and PromoInterval.

Train.csv and test.csv both contain information about the sales data for the 1115 Rossmann Stores, including Store number, DayOfWeek, Date, Sales, Customers, Open, Promo, StateHoliday and SchoolHoliday.

II. DATA PREPROCESSING

By printing out the unique values in the three datasets, we can see that there are a lot of missing values in CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2, Promo2SinceWeek, Promo2SinceYear, PromoInterval and Open (in test.csv), there is also a mix of data types in some columns, therefore it’s not easy to directly fit models to the dataset. In order to fit models, we have to preprocess data first.

First of all, we want to explore the cause of missing values in order to decide if we want to drop the missing values or replace the missing values with some other values. We looked into where the majority of missing values are- Promo2SinceWeek, Promo2SinceYear and Promo2Interval. By checking if where these values are missing also has a Promo2 value that is not 0, we found out that the values in these 3 features are missing because the stores were not participating in the promotion, or because the corresponding values of feature Promo are 0. It is the same with features CompetitionDistance, CompetitionSinceMonth and CompetitionOpenSinceYear. In this case, we decided to replace the NaN values with 0 instead of dropping them. By plotting the Sales data by time. We can see that there are very strong seasonal fluctuations in the historical Sales data. By plotting the Sales data of June and July versus the Sales data of August and September, we can also observe the similarity between Sales data between June and July and Sales data between August and September. Taking into account that the Sales data we are trying to predict are between August and September 2015, we decided to use the Sales data between June and July 2015 as our testing data.

Because we want to predict the Sales data in the testing set, we assume that those stores are open during the time period of prediction, under this circumstance, we thought it would be more reasonable to replace the missing data in Open column in test.csv with 1 instead of 0. We double checked to make sure that there are no more missing values, and then we concatenated the store.csv and train.csv as our training data and the store.csv and test.csv as our prediction set. After it was done, as we mentioned above, we took out the Sales data between June and July from the training set to be our testing set for model testing. Also, due to the nature of this task, we only kept the data that have an "Open" value of 1 which means that the store was open (if the store was closed, they didn't yield sales, thus, they are not meaningful for this model), and have a positive Sales value (we want to predict the Sales, so only data that have a Sale value that is greater than 0 are useful to this task)

III. MODEL SELECTION AND MODEL FITTING

The goal is to predict the sales of Rossmann Stores, which is a continuous variable. Thus, fitting a Linear Regression Model seems more like the right choice. We first transformed the columns that have a data type other than int by encoding them. Then we fitted a Linear Regression Model to the training dataset and used the model to predict the testing dataset.

As we can see, the Coefficient of Determination (or R^2) is even negative in this Linear Regression Model, which in theory is not even possible. This means that the chosen model does not follow the trend of data, so it fits worse than a horizontal line. Therefore, in order to fit a better model, we have to look more closely into the data and possibly transform data further before fitting models.

IV. IMPROVEMENT

We have thought about multiple reasons why when directly fitting a Linear Regression Model, the model actually fits worse than a horizontal line. One of the reasons that we could think of is that there might be too many features. Thus, when fitting models to all the features, the model was overfitted and therefore, did not generalize. In order to solve this problem, we used Covariance Matrix generated to check if there is strong correlation between any two features, and we also tried to keep only some of the features and fit the model again. We also used PCA to try reducing the dimensionality of the model while keeping most of the information. After that, we did some feature engineering and target engineering. Lastly, we used Random Forest to find out the ten most important features and fitted Logistic Regression Model, Naïve Bayes Model and Linear SVC Model to the ten most important features and tested them for prediction accuracy.

A. Use Smaller Sized Dataset

As mentioned above, we suspected the model was not doing well because of overfitting. Therefore, we tried only using a smaller set of data for training

We shuffled the training set and only kept the first 100,000 data for model training which hopefully would give us a whole picture of the dataset. However, the model fitted was doing even worse than the original Linear Regression Model. Therefore, we concluded that in order to have better model training we should still use all the training data for better outcome.

B. Keep Only Some Features

By printing out the Covariance Matrix, we found that there was not strong correlation indicated by the Covariance Matrix between any two of the features. Therefore, we tried keeping some of the features with intuition to see if the fitted model was able to predict more accurately.

In the first set of features we kept, we dropped features Customers, competitionDistance, competitionOpenSinceYear and Promo2SinceWeek and we tried fitting Linear Regression Model again to see how well it did. Not surprisingly, the Linear Regression Model fitted was still not as good as a horizontal line.

In the second set of features we kept, we dropped features Store, DayOfWeek, Date, StoreType, CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear and PromoInterval. We tried fitting the Linear Regression Model again to see if the model did better. And we found out that the model was still not as good as a horizontal line

Moreover, the Coefficient of Determination (R^2) scores given back by the models fitted with only some of the features kept were even lower than the original Coefficient of Determination (R^2) score. After thorough thoughts, we decided to try Principal Component Analysis instead, which would keep most of the information while reducing the dimensionality of the data.

C. PCA For Dimensionality Reduction

By printing out the principal components and the variance they explained, we decided to transform the testing data by keeping the principal components that consist of the 95% cumulative variance explained. Then we used the transformed dataset to fit Linear Regression Model again. The new model still gave back a bad Coefficient of Determination (R^2) score. In this case, we decided not to move forward with dimensionality reduction with PCA

After the above experiments we realized that maybe the problem of fitted model not being able to predict well is not overfitting. Maybe the problem is that the model cannot understand the input features properly. In order to solve this problem, we decided to work on feature engineering before fitting model.

D. Feature Engineering and Target Engineering

By looking more closely into the data, we found that in features StoreType, Assortment and StateHoliday, there is string data type, like ‘a’, ‘b’, ‘c’ and ‘d’. Moreover, in feature StateHoliday, there’s a mixture of number in string format data type, like ‘0’, int data type, like 0 and string data type, like ‘a’, ‘b’, ‘c’. We realized that it might be hard for the model to understand the mixtures properly. There are other features that are hard to interpret as well, like PromoInterval, which is a combination of the abbreviations of Months. For feature Date, it is not even in the datetime data type. Feature CompetitionSinceYear and CompetitionOpenSinceMonth are also hard for the model to interpret because they are not in chronologic order thus not really meaningful.

Therefore, we decided to transform the features accordingly before fitting models again.

After feature engineering, we dropped features Date, Customers, Open, PromoInterval and monthStr before fitting Linear Regression Model again. The Coefficient of Determination (R^2) score given back by the model is 0.21, from which we can see an increase.

Then we had the thought of fitting Logistic Regression Model, Naive Bayes Model and SVC Model to the dataset. However, in order to do so, we had to transform the target (or Sales in this specific case) first. As mentioned above, the dependent variable, Sales data, is a continuous variable. Thus, it is not really suitable for Logistic Regression Model or Naive Bayes Model or SVC Model. Therefore, we decided to apply the philosophy of Logistic Regression to the dataset, by normalizing the Sales data first and then making the normalized Sales data which are greater than 0.5 value 1 and smaller than 0.5 value 0. Then we used the transformed features and target to fit Logistic Regression Model, Naive Bayes Model and SVC Model.

As we can see, the models fitted are doing much better than Linear Regression Model after feature engineering. The use of fitting Logistic regression Model, Naive Bayes Model and SVC Model can be interpreted this way, for a specific store with the predicted Sales value 1, it means that for this specific store, the predicted Sale is above average Sale across all stores. For a specific store with the predicted Sale value 0, it means that for this specific store, the predicted Sale is below average Sale across all stores.

We also fitted Linear Regression Model to the dataset after both feature engineering and target engineering again and not surprisingly, the newly fitted Linear Regression Model is not doing as well as the Linear Regression Model after only feature engineering.

E. Keep the Most Important Features

After fitting Logistic Regression Model, Naïve Bayes Model and Linear SVC Model successfully, we were still not very satisfied with the prediction accuracy of the models fitted. Combining our concern for model overfitting, we decided to use Random Forest to find out the most important features and use only the most important features to fit the models.

We chose the ten most important features as our new features for fitting models, and we fitted Logistic Regression Model, Naïve Bayes Model and Linear SVC Model again and tested them for prediction accuracy. We could see that there was no obvious improvement in models fitted with the only ten most important features. Although Linear SVC Model is performing slightly better. In this case, we decided not to move forward with keeping the ten most important features for model fitting.

V. CONCLUSION

After many experiments with repeated model fitting with different features, we came to the conclusion that, directly fitting models to datasets for good prediction is not always the case. Sometimes, in order to fit a good model and use it for successful prediction, we will need to look closely into the data and transform the data before model fitting. Sometimes, simple encoding does not work well. In this case, we, as data scientists, will need to look more closely into the data and engineer data into data that models can understand or interpret. This is where the experience and good intuition of good data scientists come into play. We believe that, by using this project as an example, we have proved this point. We have also learned a lot in the process of exploring the datasets, applying different models and trying figuring out how to improve the performances of the models. Moving forward, we will remember the lessons we have learned from this project and better ourselves.