Railway Ticket Price Optimization and Dashboard Reporting using Linear Regression with Python, Spark and Plotly (January 2022)

Ufuk Altan

ABSTRACT As the technology improves it becomes more crucial in the transportation industry to determine a way to make better decisions. In terms of pricing, regression or time series forecasting models can be performed. However, when price optimization is considered, it is important to measure the price elasticity of the demand for each product. In transportation industry, demand is constantly altering. It is a big challenge to manage products, track different KPIs and make decisions simultaneously. Therefore, a big data tool like apache spark and a reporting technology like plotly can provide long term solutions. This article explains a data processing progress for spanish railway data, a price optimization model using spark and the plotly library of python for reporting through a dashboard.

INDEX TERMS ***Apache Spark, Plotly, Price Optimization, Linear Regression, Python***

I. INTRODUCTION

Many companies in different industries aggregate data in the purpose of analyzing their customer demography, product features and tracking different metrics. In transportation industry, there is always a constant data flow. Tickets that are bought could be flexible, fare base tickets that are in economy class or also could be standard tickets that belong to the business class. If a data source or a public API is available, competitor data could be utilized as independent variables in a regression model. Furthermore, departure time of a train may affect the price and demand of tickets. There are so many variables to considered, however, this article is going to evaluate the variables in the spanish railway dataset that is found on kaggle.

Insert Date: Includes information of date and time that train tickets are published for sale.

Origin: Provides information of the departure location of the train.

Destination: Provides information of the arrival location of the train.

Start Date: Provides information of the time the train is going to depart.

End Date: Provides information of the time the train is going to arrive.

Train Class: The class of the train represents the different extra services that are provided to the passengers. These could be more comfortable seats, food or even entertainment.

Train Type: There are different types of train available in spanish railways which might indicate the features of being a faster than other trains or even luxury.

Fare: Fare bases are different type of tickets that come with different benefits. For example, standard ticket is just a permission to board the train or a flexible ticket gives an extra permission to change seats, time or even the arrival location of the train.

Price: The purchase price that is required to own a place on the train.

In order to perform price optimization, it is required to have variables that have negative relationship with each other. Price elasticity of the demand is a well-known concept in all industries. As the price increases for a specific product, it is possible to observe that demand is decreasing. The challenge is to find the optimum point for price that maximizes revenue for the company which also known as dynamic pricing.

Later a dashboard is required to track different metrics. Demand by hour or pricing for different destinations and how these prices affect the entire revenue management system. Spanish railway data includes many external variables like train type, train class and fare bases; it also includes the decision variable, which is price, however, it does not include the target variable which demand. In order to perform price

optimization and to build a comprehensive dashboard demand data must be generated even though it is for experimentation purposes.

II. PREPROCESS AND DATA GENERATION

Before generating demand data for price optimization, it is critical to establish a strategy for it. Data generation is whether done for lack of data or for experimentation purposes. In this case, it is an experimentation to observe how price optimization can be done in train ticket pricing.

Building an optimization model requires to build a forecasting model first. Using the linear regression model, it is possible to use these variables to predict the demand and later using an optimization algorithm can be used to determine the price that is going to maximize the revenue.

In such case all these independent variables can be utilized to generate the demand data. Determining importance weights for each categorical variable and a negative correlation between price and demand would ensure the data source for price optimization. Therefore, first step would be to preprocess the data.

First, insert date and end date are removed from the dataset since insert date is same for most of the data and the end date variable can be considered the duplicate of the start date variable. Later, origin and destination variables are merged into a single column representing where train is departed from and where it is going so that it could easily be utilized as a categorical variable.

Using spark functions to analyze the data, it can be observed that 12% of the price data is missing. The consensus is that 1% of the data is affordable to be dropped. Therefore, the missing values inside the data has to be filled before analysis.

Independent variables of the data are assumed to affect the change on the price numbers. These variables can also be utilized when filling in the missing data instead of just filling the null values with the average price value.

Four different dataframes are prepared using the “groupby” function of spark. Train type, train class, fare and from to variables are grouped and the mean value price column is calculated for each dataframe. These four dataframes are used to fill in the price column of the data until there are no null values left on the main dataframe.

Using spark functionalities start date column is turned into timestamp format. Later, variables like month and hour are extracted from the start date column to be presented in the dashboard.

Some metrics that will be presented on the dashboard could be demand by hour, demand by month, average price by each train type or class. Spark functions could be used to process the data, however, plotly does not support spark dataframes. Therefore, during the dashboard part data format must be list or pandas dataframe.

After all the categorical variables are preprocessed, it is time to generate the demand data. First thing to consider is the negative relationship between the price and demand variable. To establish such thing price data is divided into sub-ranges using a custom function called “generate_intervals“. This function gets price values that are among specific ranges and matches them with the exact opposite index.

For example, values of the price column ranges from 15-213 euros and there are 40 intervals that are created. Intervals like (15-20), (20-26), (26-31), …, (208-213). Using the custom function more or less intervals could be created to increase or decrease the variability inside the data.

To remind the purpose, a negative relationship must be generated between the price and the demand. Thereof, price values that are in the range of 208-213 will be appended to exactly opposite index which is 15-20. However, in such case there would a negative correlation that is very close to perfect which would be unrealistic.

In real world, demand has a negative relationship with the price but when products are more qualified or somewhat incentive, people may be willing to price. When buying train tickets people might consider buying a more expensive ticket that provides a seat in the business class. It is important to consider the quality of the demanded product when applying a price optimization model.

Another challenge is to decide how to use these categorical variables to create variability on the demand data. These variables are significant causes of prices. How they affect the price would be also how they affect the demand. Average price for each categorical variable could be taken and considered as a parameter when shaping the demand data.

Best way to evaluate average prices would be to use standardization. In machine learning, standardization is used to evaluate numerical variables that have a different range of values. In this case, average price for each category is standardized and turned to values between .75 and .95. These values later will be multiplied with the price values and appended into the opposite indexes.

At the end of the process, pearson correlation coeffiecient between the price and the demand is calculated as -0.438 with a p-value below .05 and spearman correlation

coefficient is calculate to be -0.271 with a p-value below .05. These correlation values would be a realistic relationship between the demand and the price.

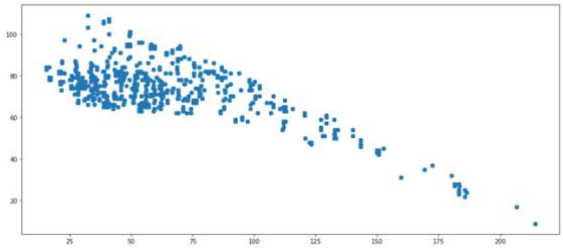

Figure 2.1

Later, demand and price data can be multiplied to revenue data as shown on the figure 2.2. When it comes to price optimization, external variables and the decision variable can be both used to forecast the demand in a linear regression model or forecast the revenue in a non-linear regression model.

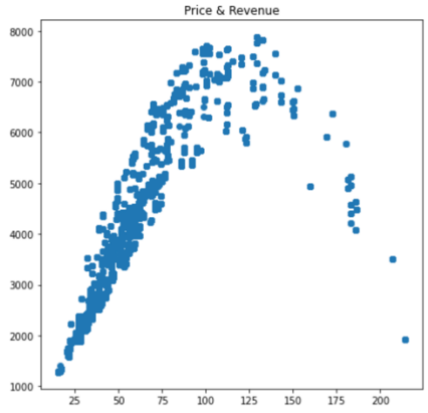

Figure 2.2

In this case, demand variable will be used and later forecasted values will be multiplied with the price to maximize the revenue.

III. COORDINATES

With the generated demand data, dataset now consists so much variety. However, in order to build a map chart, coordinates for each destination is required.

Longtitude and latitude of each destination is written into a python list and turned into a spark dataframe afterwards.

destination	lat	long
BARCELONA	41.3874	2.1686
MADRID	40.4168	-3.70379
VALENCIA	39.469906	-0.376288
PONFERRADA	42.55	-6.59083
SEVILLA	37.3891	-5.984459

Table 3.1

Another important thing is demand and revenue generated in every destination should be matched with their coordinates. So to speak; latitude, longtitude, demand and revenue values should all be place into the same dataframe to create a map chart on the dashboard.

from	to	Demand	Revenue	lat	long
MADRID	VALENCIA	27909907	1.689549504516309E9	40.4168	-3.70379
MADRID	PONFERRADA	4795547	2.973105366986997E8	40.4168	-3.70379
MADRID	BARCELONA	33964253	2.153120697412605E9	40.4168	-3.70379
MADRID	SEVILLA	25592784	1.5989933762504072E9	40.4168	-3.70379
BARCELONA	MADRID	30326098	1.950985811632306E9	41.3874	2.1686
PONFERRADA	MADRID	5621490	3.4295422125269973E8	42.55	-6.59083
SEVILLA	MADRID	24956660	1.537146057425108E9	37.3891	-5.984459
VALENCIA	MADRID	23998350	1.4316202612796075E9	39.469906	-0.376288

Table 3.2

IV. PRICE OPTIMIZATION

Before the demand forecasting and price optimization, there is one last preprocessing step remains which is one hot encoding. Using the StringIndexer and OneHotEncoder classes of spark, categorical variables can be indexed and evaluated as numerical data. Later, VectorAssembler class is used assemble variables in order to perform linear regression.

Next is to test the regression model with various metrics. Model has a R2 value of 0.978, a R2 adjusted value of 0.978, RMSE of 1.181 and MAE of 0.826. These results were satisfiying a regression model.

Another test would be the percentage error test. Using the real and forecasted values and calculating percentage difference between them. These percentage errors create a new dataframe, that has a mean of 1.19%, a median of 0.646%, min value of 0.010% and max value of 85%.

Median value can be considered the most valuable metric in this scenario. 85% max value is most likely caused by outliers that could later be removed to improve the performance.

Looking at the all test data of 513.808 rows, 0.36% was above 10% error percentage. Above 80% were only 198 rows and more than 99% were below 10% error threshold.

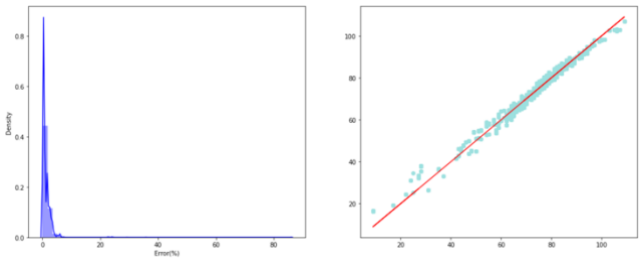

Figure 4.1

To optimize the price, a piece of data can be taken from the sample, forecast demand for different values of price and revenue can be later calculated and maximized.

For example, a train from Barcelona to Madrid, train type of ave-tgv, train class of tourist and a flexible type of fare base. A price range of 15-214 is given and each value of the price is used to forecast the demand and to calculate revenue.

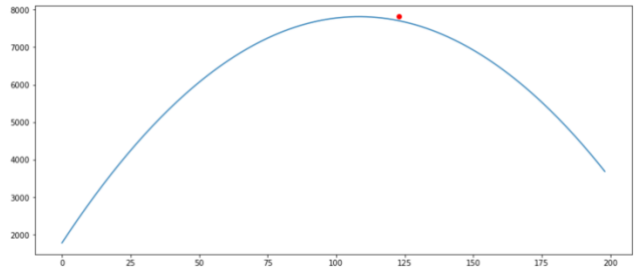

Figure 4.2

In this example, optimum price would be 123 euros with a demand 63.49 and with a maximized revenue of 7810.17.

IV. DATABASE INTEGRATION

After data is prepared and optimization model is built data is placed into PostgreSQL and retrieved using spark in order to build the dashboard.

Providing few configuration details, a JDBC driver that establishes the connection between the spark and the relational database. Interesting fact was Jupyter Notebook could not retrieve the data from the database but Sublime Text actually made it happen, which is also used to create the dashboard using plotly.

IV. PLOTLY

Plotly is a python library that can be used to build interactive dashboard applications that does not require of frontend development technologies like Javascript and HTML.

It is built on Flask and React.js in order to react to the requirements of the business world as fast as possible. This project uses Flask, Plotly and Apache Spark to create an interactive dashboard that could retrieve big data as needed.

V. CONCLUSION

The project started from data preprocessing. Later demand data is generated, price optimization is performed, data is retrieved from the database and presented inside the dashboard.

Transportation industry requires constant monitoring of specific metrics and quick but reasonable decision making. Technologies like Python, Apache Spark, Plotly and PostgreSQL brings competitive advantage to businesses. As a result of these technologies and data an advanced price optimization model can be built and served.

REFERENCES

- The Data Science Course 2021: Complete Data Science Bootcamp, 365 Careers
- Spark and Python for Big Data with Pyspark, Jose Portilla
- Pyspark & AWS: Master Big Data with Pyspark, AI Sciences
- Apache Spark 3 for Data Engineering & Analytics with Python, David Charles
- Create Interactive Dashboards in Python by Plotly and Dash, Mubeen Ali
- Interactive Python Dashboards with Plotly and Dash, Jose Portilla
- Data Engineering using Kafka and Spark Structured Streaming, Durga Viswatnava and Raju Gadiraju