Data Science technical test

October 2021

General guidelines

This project will assess your capacity to solve problems on your own, and to present a fully working solution to a data science problem mirroring the ones we are working on daily at Wiremind.

At Wiremind, we leverage machine learning to deduce the willingness-to-pay of train travelers. The models are trained on structured, cleaned and preprocessed datasets that are extracted from raw, unformatted data sources such as SQL tables. The goal of this project is to train such a model based on a clean anonymized dataset extracted and preprocessed for you. The modeling, training and validation will be handled by your code. The data and their underlying trends are close to real customer data.

The programming language is **python**. You are free to organize your code the way you want, and to use any tool or framework you feel comfortable with (*e.g.* TensorFlow, SKLearn, Pandas, PyTorch, jupyter notebooks, any visualisation library you like).

The main evaluation criteria for this test are:

- The **depth of your analysis** of both the data and your model's performance
- The **performance** and **architecture** of your model
- The overall **quality, clarity** and **design** of your codebase

Dataset specifications

With this instruction notice, you have been given a zip folder comprising two CSV files:

- A **test** file containing the data to be used for your validation
- A **train** file containing the data to be used for your training

Those CSV files have been compressed using the **lz4** format. You can refer to the end of this document for a complete definition of each field contained in the dataset.

The dataset can be seen as multiple independent time series representing the daily sales for each offered train - departure date - origin - destination of a customer: for each tuple ("departure_date", "od_origin_time", "origin_station_name",

"destination_station_name"), you have the daily sales of the last 90 days before departure.

Feel free to combine or transform the available set of features of the dataset as you see fit, but remember that your model should be able to return a predicted demand for any day_x and price value in the future.

We already cleaned the dataset for you so there should be very little to do on this side as we want you to focus on model training and data analysis.

Objectives

Data analysis

Although the dataset at your disposal has been pre-processed, it remains subject to the kinks and quirks that come with real-world data sources and may impact the accuracy of predictive models. For this reason, you are expected to conduct an exploratory analysis of the dataset, using any method you see fit, statistical or otherwise. While the exact nature of such analysis is left up to your judgement, particular attention should be given to the **distribution of key numerical features**. Any insight gleaned from your analysis is expected to inform your choice of model architecture, and of any additional data processing step you may choose to perform.

Model training

You are tasked with writing a clear pipeline to train and validate the daily unconstrained demand (column "demand") given the offered price (column "price") and day before departure (column "sale_day_x"). We define the unconstrained demand as the amount of people ready to pay P to buy a ticket for A-B for a given sale_day_x (which is the number of days remaining before departure). Thus your **model** (or composition of models) **should be a function containing at least the following elements** (**day_x**: int, **price**: float).

The choice of the model is left to your appreciation. You are - obviously - not expected to implement anything from scratch, and may use any framework you prefer (*e.g.* XGBoost, LightGBM, SKLearn, PyTorch, TensorFlow).

Model validation

The evaluation step should present a table split per origin/destination, with the **average absolute and relative error for the cumulative demand from a day_x up to the day of departure**. The day_x to be used for this validation are: -90, -60, -30, -20, -15, -10, -7, -6, -5, -3, -2, -1. This means that for instance if a train has sold 200 tickets for origin-destination “CPE-AG” from -90 to 0, we want to compare the sum of the predicted values for each day-x of your model for this origin-destination against the cumulative observed bookings (200). To do this, you can assume you will have all required features for all remaining day_x (including "price").

The choice of **target** and **loss functions** is left up to you. Indeed, one of your missions once in the team will be to define accurate metrics that mimic the business needs of our clients, and anticipate the possible dysfunctionalities of deployed models. Keep in mind that the goal of these models will be to optimize **revenue** (price × demand).

In addition, you are expected to perform a **thorough analysis of your model’s performances and error profiles**. In doing so, particular care should be given to identifying conditions under which the model performs especially well or poorly.

Results presentation

We expect you to send us a link to a **private repository** (github, gitlab) before the deadline. Given the time you have, you may not be able to implement all of your ideas. We suggest you focus on the most important ones, and write a readme.md with eventual follow ups. Your package should at least contain a quickstart.md file allowing us to easily run your pipeline with fresh data, along with a dockerfile to launch it.

If your code receives positive reviews from our team, you will be invited to come present your results on site. This presentation will allow you to discuss your technical decisions and any problems you may have encountered.

If you have any questions feel free to reach out.

Good luck!

Fields definition

Dataset_type: Indicates whether this line should be used during training or testing

Sale_day_x: Number of days remaining before departure. For instance, -2 means that there are 2 days remaining before departure. This also means that the other feature for this line relates to the day-x = -2. If you train leaves on the 2021-02-05, this would mean the rest of the line relates to the sale date of 2021-02-03.

Price: Offered price during this sale_day_x

Demand: Unconstrained demand observed for this sale_day_x and this price

Departure_date: Departure date for the train

Origin_current_public_holiday: 1 if departure date in origin city was a public holiday, 1 otherwise

Origin_current_school_holiday: 1 if departure date in origin city was a school holiday, 1 otherwise

Origin_days_to_next_public_holiday: Number of days between departure date and the next/current public holiday date in origin city (if negative, it means the date is in a holiday period)

Origin_days_to_next_school_holiday: Number of days between departure date and the next/current school holiday date in origin city (if negative, it means the date is in a holiday period)

Destination_current_public_holiday: 1 if arrival date in destination city was a public holiday, 1 otherwise

Destination_current_school_holiday: 1 if arrival date in destination city was a school holiday, 1 otherwise

Destination_days_to_next_public_holiday: Number of days between arrival date and the next/current public holiday date in destination city (if negative, it means the date is in a holiday period)

Destination_days_to_next_school_holiday: Number of days between arrival date and the next/current school holiday date in destination city (if negative, it means the date is in a holiday period)

Origin_station_name: Name of the station of origin

Destination_station_name: Name of the station of destination

Od_origin_time: (hour of departure) * 60 + (minutes of departure). For instance, if you depart at 16:08, this field will have 16*60 + 8 = 968

Od_destination_time: (hour of arrival) * 60 + (minutes of arrival). For instance, if you arrive at 16:08, this field will have 16*60 + 8 = 968

Od_number_of_similar_X_hours: Number of trains offering the same origine-destination in a 12 hours interval around the departure time ("-1" if data is missing). X = 2, 4, 12 in this dataset.

Od_origin_month: Month of departure

Od_origin_week: Week of departure

Od_origin_weekday: Day of week of departure

Od_origin_year: Year of departure

Od_travel_time_minutes: Time to reach destination, expressed in minutes.

Sale_date: Date of sale for this line. Is equals to (departure_date + sale_day_x)

Sale_day: Day of sale date (from 1 to 31)

Sale_month: Month of sale date

Sale_week: Week of sale date

Sale_weekday: Weekday of sale date

Sale_year: Year of sale date