Teste-B2W

Felipe Almeida

Process

- The first thing I started doing after receiving the assignment was to start reading about time-series forecasting, because I had never done time-series forecasting before and I wanted to make sure I made no silly mistakes.

- I read many articles and watched many videos about time-series forecasting, models such as AR, MA, ARMA, ARIMA, etc.

- I also looked at many python packages commonly used to deal specifically with time-series problems, such as pyFlux.

Process

- The next thing I felt I needed to do was to do thorough exploratory data analysis (EDA) on both datasets

- link to notebook

- I found out useful things and quirks about the datasets, about how skewed or concentrated the data were. This helped me get a better understanding of the task at hand.

  - One of the first things I felt I needed to do was to create an auxiliary column in the dataset, namely UNIT_PRICE, because this is what the model should take as parameter (since the task was to find a mapping y=f(x) such that **y** is the quantity of items sold and **b** is the target unit price for that item

Process

- Once I started writing code to start modelling, I found out that the problem was not really a time-series forecasting problem as I had first thought.

- Generally in time series, you generally have one variable of interest, sampled at different points in time.

  - This is **not** what I was supposed to do for this task.

- The problem would be correctly modelled as a time-series if the samples contained *just* the number of items sold (QUANTITY) and a timestamp

- In other words, I could model this problem as a time-series if there were no UNIT_PRICE feature added to the model.

Process

- Once I saw that I would have trouble modelling the problem as a time series, I then started thinking about more common Machine Learning approaches, such as regular regression and classification.

Approach 1: Simple Linear/Polynomial Regression

- I've chosen to do a very simple Linear regression as the first approach because a) it's very simple and b) it's good to try out simple models first because they may be surprisingly good while still being fast and cheap

- link to notebook

- Result: the model did not look very suitable to the job (not even when adding extra degrees of freedom)

Approach 2: Simple Classification

- Although the problem is best seen a *regression* problem (i.e. predict QTY_ORDER given a UNIT_PRICE) I thought it would be useful to use a neural network with softmax output here.
- The reason was twofold:
  - neural networks are nonlinear by default (and trying linear models didn't work, as per the previous approach)
  - Even if we frame the problem as a classification problem, we can still get *soft* (nearly continuous) answers if we discretize the targets (see the notebook) because the softmax function naturally outputs probabilities for *all* target categories.
- link to notebook
- Result: much more understandable and expected results here.

Conclusion

- What I would do if I had more time:
  - Further explore the relationship between our vs competitor prices, e.g.:
    - Do we get more sales if we systematically charge less (per product) than all of our competitors?
  - Instead of using just (quantity,unit_price) features for each input row, we could extract many more feature from external data, such as
    - Price difference with respect to C1,C2,C…,Cn
    - Pay type (one-hot-encoded)

Conclusion

- What I would do if I had more time:
  - Further explore the relationship between our vs competitor prices, e.g.:
    - Do we get more sales if we systematically charge less (per product) than all of our competitors?
  - Instead of using just (quantity,unit_price) features for each input row, we could extract many more feature from external data, such as
    - Price difference with respect to C1,C2,C…,Cn
    - Pay type (one-hot-encoded)

Using extra features raises the question: will we have those data at prediction time?

(the problem description described a function of the following form: QTY = F(PROD_ID,UNIT_PRICE)

Extra: Show Git Knowledge

- This whole project was created using Git, as can be seen on the project link: https://github.com/queirozfcom/teste-b2w

- I've been using Git and other kinds of Code Versioning Systems for years; I'm quite comfortable with it.

Extra: Show SQL Knowledge

- I've used SQL for years, in multiple RDBMSs such as PostgreSQL, mySQL and Oracle; I'm quite comfortable with it.

- Let's suppose I loaded the data (sales.csv and comp_prices.csv) into two tables, with the following schemas:

```
CREATE TABLE sales
(
  prod_id character varying,
  date_order date,
  qty_order double precision,
  revenue double precision
)
```

```
CREATE TABLEcomp_prices
(
  prod_id character varying,
  date_extraction date,
  competitor character varying,
  competitor_price double precision,
  pay_type integer
)
```

Extra: Show SQL Knowledge

- For example, if the data (sales and competitor) were stored in an SQL database and I wished to find out the average unit price (revenue / quantity) per PRODUCT for each month, I would write a query like this:

```
select
    to_char(date_trunc('month', s.date_order),'mm') "month",
    s.prod_id,
    round(avg(s.revenue / s.qty_order)::numeric,2) "average_unit_price"
from sales s
group by date_trunc('month',s.date_order), s.prod_id
order by month
```

Extra: Show SQL Knowledge

- For example, if the data (sales and competitor) were stored in an SQL database and I wished to find out the average unit price (revenue / quantity) per PRODUCT for each month, I would write a query like this:

Extract the month

```
select
    to_char(date_trunc('month', s.date_order),'mm') "month",
    s.prod_id,
    round(avg(s.revenue / s.qty_order)::numeric,2) "average_unit_price"
from sales s
group by date_trunc('month',s.date_order), s.prod_id
order by month
```

Only show 2 decimal places

Extra: Show SQL Knowledge

- The result is something like this (only some rows shown)

	month text	prod_id character varying	average_unit_price numeric
1	01	P2	799.85
2	01	P6	1906.68
3	01	P7	816.84
4	02	P2	821.85
5	02	P3	1471.61
6	02	P7	820.64
7	02	P1	1430.19
8	02	P6	1715.64
9	03	P1	1390.98
10	03	P3	1407.58
11	03	P9	564.55
12	03	P7	821.36
13	03	P6	1773.99
14	03	P4	565.12

Extra: Show SQL Knowledge

- Another example: what is the average difference (per day) between the UNIT_PRICE we charge for each product and the prices our competitors on average) charge? This is a slightly more complex query (on the next page)

Extra: Show SQL Knowledge

```sql
SELECT
        b2.day,
        b2.prod_id,
        round(avg(b2.average_unit_price - cp.average_unit_price_competitor)::numeric,2) "our avg price - competitor avg price"

FROM
        (SELECT
                to_char(date_trunc('day', cp.date_extraction),'dd/mm') "day",
                cp.prod_id,
                round(avg(cp.competitor_price)::numeric,2) "average_unit_price_competitor"
        FROM comp_prices cp
        GROUP BY date_trunc('day',cp.date_extraction),cp.prod_id
        ORDER BY day) cp

        JOIN

        (SELECT
                to_char(date_trunc('day', s.date_order),'dd/mm') "day",
                s.prod_id,
                round(avg(s.revenue / s.qty_order)::numeric,2) "average_unit_price"
        FROM sales s
        GROUP BY date_trunc('day',s.date_order), s.prod_id
        ORDER BY day) b2

        ON

        cp.day=b2.day AND
        cp.prod_id=b2.prod_id

group by b2.day,b2.prod_id
order by day, prod_id
```

Extra: Show SQL Knowledge

The result (first rows only) looks something like this:

	day text	prod_id character varying	our avg price - competitor avg price numeric
1	01/01	P2	37.17
2	01/01	P6	-251.98
3	01/01	P7	26.65
4	01/02	P2	33.43
5	01/02	P6	-49.20
6	01/02	P7	-9.89
7	01/03	P2	-9.82
8	01/03	P6	491.25
9	01/03	P7	-35.28
10	01/04	P1	-39.89
11	01/04	P2	-23.52
12	01/04	P3	-52.43

Extra: Show SQL Knowledge

The result (first rows only) looks something like this:

	day text	prod_id character varying	our avg price - competitor avg price numeric
1	01/01	P2	37.17
2	01/01	P6	-251.98
3	01/01	P7	26.65
4	01/02	P2	33.43
5	01/02	P6	-49.20
6	01/02	P7	-9.89
7	01/03	P2	-9.82
8	01/03	P6	491.25
9	01/03	P7	-35.28
10	01/04	P1	-39.89
11	01/04	P2	-23.52
12	01/04	P3	-52.43

On this day, our prices for P2 were, on average, R$ 37,17 higher than our competitors

Extra: Show Clustering Knowledge

- I know about clustering methods (K-means, Gaussian Mixtures, etc) but I'm not sure where they could help here, except for visualization purposes.