A1: End-Term Assessment

HULT International Business School

Data Mining with Mysql, Nosql, Hadoop, Spark, and Hive - DAT-5312 - SFO1

Roger Lopez Benet

7/30/2023

Contents

Question 1:

In order to build a team of data analysts and scientists, with the objective to analyze data of the Earn-it-up transactions, there are some key resources that must be obtained. The new hires of this data science team would have to meet the following criteria. They should be fluent in SQL, and NoSQL languages that can process JSON like documents such as MongoDB.

The mentioned qualifications must be met by the team because a web-app like this, needs to process both structured and non-structured data types. Although both types of data must be used, structured data types are still the most used. This is the case because transactional data tends to be in tabular form, since most fields need to be stored with a very solid structure. Examples of these would be demographics, employment information, payment history, and transactional records in general. But at the same time, unstructured data such as user-generated content, like in the case of customer service tickets, web-app logs, sessions, and impressions.

Although that can be a good approach, it’s also good to point out that using MongoDB makes the whole process much smoother and boosts efficiency, since with MongoDB, documents are stacked on top of each other saving time and complexities of joining them in SQL as well as in case there are some kind transactions that are only made a group of people, and not others, these fields can be blank and not be considered as null, making mining this data much easier.

In the case of unstructured data Earn-it-up cannot use the widely used SQL, instead, it could use NoSQL or DFS, the problem with DFS is too expensive and would burn out our funding quite fast, making it not ideal for our mid-size startup. Instead, we should use NoSQL like MongoDB, which gives us flexibility and scalability, as well as offering all this at a reasonable performance/cost.

Therefore, a combination of both SQL and MongoDB would be the way to go. This allows us to store structured and non-structured data, using the industry standard, SQL which almost everyone in the field knows, which makes it easier to find people with these skills and since it’s a commodity salaries for this skill should be lower too. Besides this, MongoDB should also be a target for us because it uses a more flexible, easy-to-scale infrastructure.

Question 2A:

To reach solid conclusions these findings would have to be contrasted with other data, but with the data we have, the following can be interpreted.

Sub-question #1:

The total number of Pokemons that meet the specified criteria (Type_1 = Grass or Water, and Attack > 65) is **48**. This means that out of the total of 800 documents, only 6% of the Pokemons meet these criteria.

The most obvious insight from this is the fact that out of a total of 800 documents/Pokemons, only 48 meet the criteria, meaning it’s very rare to have a Pokemon above 65 score in Attack. Because of its rareness, their value also should increase, so if these Pokemons were to be collected and traded, the ones with these characteristics, having Attack as the main factor, 65 and above in Attack score, it could be considered a good card.

Sub-question #2:

The average Defense for those documents where Attack is greater than 30, and are not Legendary, is **73.30**.

This can mean that non-legendary Pokemons, for the only fact of not being legendary, tend to have lower Attack and Defense, in this case averaging 73.30. This could reaffirm that the 'Legendary' variable is important since it drives some variables up as we'll also see in later in the paper.

If we were to compare this results with the opposite scenario, the average Attack for the ones with 30+ Defense is 77, which is slightly superior. Therefore, even though this difference is not too large, on average non-legendary Pokemons focus more on Attack than Defense.

Getting back to the question, since the weakest non-legendary Pokemons have been excluded, having a quite low Defense score of 73.30, might also mean that since these are supposedly the most powerful amongst the non-legendary, Pokemons with Attack below 30, the average of the rest should be quite lower, if the necessary correlation matrix were to be done and had similar results to this hypothesis, it would prove a positive correlation between Attack and Defense, meaning when Attack increase Defense does too and the other way around.

Therefore, based on these findings, what can be assured is that Legendary must be quite important for other variables such as Attack and Defense to increase too, and non-legendary pokemons tend to have low scores, making them weaker.

Question 2B:

Question 2A.1 Code (Question 1):

```
[
  {
    $match: {
      Type_1: {
        $in: ["Grass", "Water"],
      },
      $or: [
        {
          Type_2: {
            $exists: false,
          },
        },
        {
          Type_2: {
            $in: ["Grass", "Water"],
          },
        },
      ],
    },
  },
  {
    $match: {
      Attack: {
        $gt: 65,
      },
    },
  },
  {
    $count: "total_count",
  },
]
```

Question 2A.2 Code (Question 2):

```
[
  {
    $match: {
      Attack: {
        $gt: 30,
      },
      Legendary: false,
    },
  },
  {
    $group: {
      _id: null,
      avg_defense: {
        $avg: "$Defense",
      },
    },
  },
]
```

Question 3A:

- Coefficients:

First of all the coefficients tell us that one unit increase of our features ('Sp_Atk', 'Attack', 'Sp_Def', 'Defense') results in an increase of 0.08773497… in the predicted feature or (x variable), in this case ‘HP’.

The same happens with the rest of the features. Since their outputs follow the same order they were input in the model, it’s quite easy to read. But for example, the last coefficient is negative, meaning that every one unit increase in ‘Defense’, decreases the predicted ‘HP’ by 0.06425439…

- Intercept:

Even though this is not a feasible scenario, the intercept is the ‘HP’ value when all other features (considered in our model) are equal to zero. We cannot get business insights from it since this is a very unlikely scenario.

- Residuals:

This part is quite important since residuals point out if the model over-predicted or under-predicted the HP predicted values compared to the actual HP value. Therefore, the extreme values seen in the output, i.e. -51.89 and -27.18 mean the model underpredicted these values giving them a lower number than what’s supposed to be and the opposite happens with those positive numbers.

- numIteration:

The number of iterations is also known as number of epochs, which is the number of times the model iterates over the model’s weights with the goal of minimizing the loss function and maximizing accuracy or the prediction, in this case, ‘HP’.

- objectiveHistory:

These are records of the difference between the loss function mentioned before and the predicted value for every epoch. Therefore, there should be 11 records since the model was run for 11 epochs.

Question 3A Business Insights:

The take away from these outputs is that when it comes to coefficients, features with higher coefficients such as ‘Attack’ and ‘Sp_Def’ (Special Defense) in general drive HP points up the most, while other features with negative coefficients, which is the case of ‘Defense’, negatively affect HP.

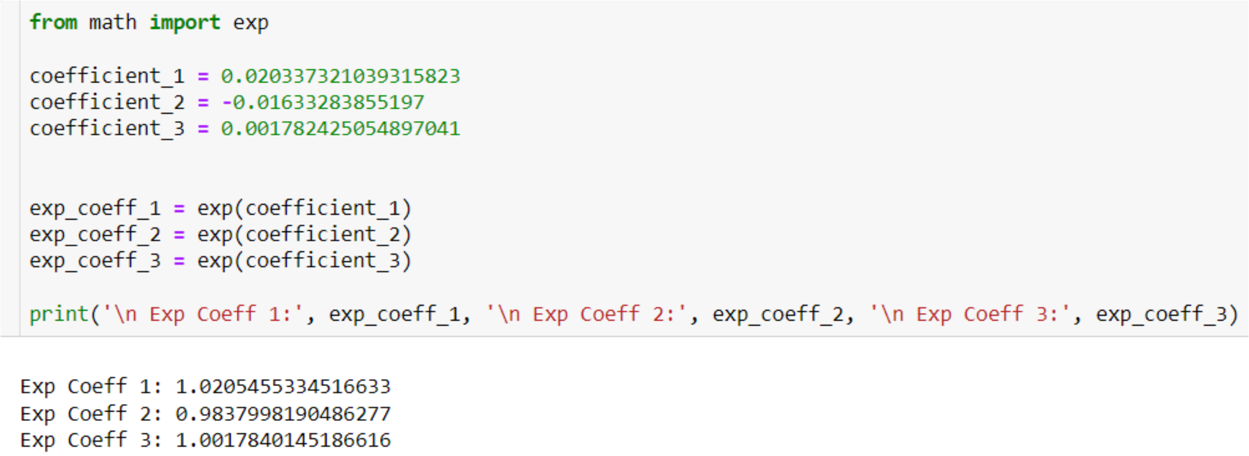
Therefore, based on these findings if higher HP is our goal, it’s more important to have higher 'Sp_Atk' (Special Attack), 'Attack', 'Sp_Def', but lower ‘Defense', since then there is a higher chance that their Attack and Special Defense scores will be higher. But in order for this to be the case, negative coefficients like Defense, should be minimized for the other two to increase.

Now, this can get tricky really fast, because Defense is important for Pokemons too, therefore a maximizing function should be implemented in order to find the right balance between all variables while maximizing HP by also maximizing Attack and Special Defense.

Question 3B:

Sub-question #1:

In order to make the coefficients interpretable, we need to use the exponential function to transform them back to their odds ratio. In order to do this, the following simple step must be done: i.e. exp(coefficient number). They were calculated using PySpark, as seen in the screenshot below:



```
from math import exp

coefficient_1 = 0.020337321039315823
coefficient_2 = -0.01633283855197
coefficient_3 = 0.001782425054897041


exp_coeff_1 = exp(coefficient_1)
exp_coeff_2 = exp(coefficient_2)
exp_coeff_3 = exp(coefficient_3)

print('\n Exp Coeff 1:', exp_coeff_1, '\n Exp Coeff 2:', exp_coeff_2, '\n Exp Coeff 3:', exp_coeff_3)
```

```
 Exp Coeff 1: 1.0205455334516633
 Exp Coeff 2: 0.9837998190486277
 Exp Coeff 3: 1.0017840145186616
```

These are the results of the exponential coefficients:

- Exp Coeff 1: **1.0205455334516633**
- Exp Coeff 2: **0.9837998190486277**
- Exp Coeff 3: **1.0017840145186616**

Sub-question #2:

The first coefficient (1.0205…) means that for every one unit increase of our y variables, in this case 'Total' (total points), the predicted variable or x-variable ('binary_outcome') increases by 1.0205455334516633. The same happens for the second one and third one, where one unit increase in Attack, and Defense, results in an increase of the predicted variable by 0.9837998190486277 and 1.0017840145186616 respectively.

Therefore, it can be inferred that out of these 3 variables the ones that positively affect our predicted variable ('binary_outcome') the most are 'Total' and 'Defense', since they have the highest coefficients. The variable 'Attack' is quite close to the other two, so it could also be taken into account.

Therefore, the higher these three variables are, the more likely these Pokemons are to be 'Legendary', which means these factors should be the main focus for players when choosing their Pokemons. Therefore, even though these are only 3 variables, it's a good indication that in order for players to have Legendary Pokemons, 'Total', 'Defense', and 'Attack' should be maximized.