A1: End-Term Assessment

Rhea Kapoor

Masters in Business Analytics

Hult International Business School

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

Prof. Thomas Kurnicki

July 29, 2023

Table of Contents

Question 1

You've been invited to an interview at a medium size (approx. 30 employees) startup called Earn-it-up. The role that you're interviewing for is a Data Systems Manager. The company's main product is a web-app (written in Java Script) that allows the end user to a get cash advance on their paychecks.

If you were given this opportunity, your main MBO (objective) for the next 12 months would be to hire a team of data scientist/analysts and implement systems and processes to analyze transactional data from the web-app. The hiring manager (the CTO of Earn-it-up) has asked you what resources you would need to start growing an analytics team. They were most interested to learn your point of view regarding data storage systems and what your recommendation is for this given situation.

Answer 1:

To implement resilient systems and streamline processes for efficient analysis of transactional data generated by our web-app, several key resources are essential, with particular focus on the data storage system.

First and foremost, understanding the diverse data types generated by the web-app is critical for designing an appropriate data storage system. This includes financial records, user interactions, timestamps, and user profile information. Additionally, the presence of semi-structured data, like user feedback and logs, must be considered to select the most suitable data storage approach.

Considering the dynamic nature of our web-app and the potential growth in user data, **I recommend adopting a NoSQL database, specifically MongoDB**, for Earn-it-up's data storage system. NoSQL databases excel in handling various data types, including semi-structured and unstructured data, while offering scalability to efficiently accommodate the increasing volume of transactional data. MongoDB's document-oriented model allows related data to be stored together, simplifying complex queries, and enhancing the overall analytics process.

Furthermore, ensuring compatibility with our web-app's technology stack is crucial. MongoDB, being a popular NoSQL database, has robust support for JavaScript and provides various client libraries that seamlessly integrate with our web-app. This compatibility will enable smooth data interactions between the application and the database, enhancing user experience and overall efficiency.

In addition to the data storage system, equipping our data scientists and analysts with the right tools is essential for meaningful data analysis. Providing licenses or subscriptions to industry-standard data analytics tools, such as Tableau or Power BI, empowers the team to visualize and derive valuable insights from the transactional data efficiently.

Lastly, given the sensitive nature of the financial data involved in our main product, implementing robust data security measures is of utmost importance. Allocating resources to ensure data encryption, access controls, and regular security audits will protect user information and maintain compliance with data protection regulations.

Question 2A

Using MongoDB Compass (or alternatively the online Mongo Atlas):

1. How many **Pokémon** (how many documents) are Grass or Water (in Type_1) and have Attack greater than 65? What insight does it bring?

Answer 2A.1:

In total, there are **104 Pokémon** that meet the criteria of having Type_1 as Grass or Water and an attack greater than 65.

Insights:

Popular Types: The fact that there are 104 Pokémon from just two types (Grass and Water) suggests that these types may be relatively popular among trainers. Players might prefer Pokémon with these elemental attributes due to their effectiveness in battles or for strategic reasons.

Competitive Battling: These Pokémon also seem to have high Attack values and players may use these Pokémon frequently in competitive battles. Their popularity may be attributed to their strengths, abilities, and overall effectiveness in battling opponents.

Potential Team Combinations: With 104 Pokémon to choose from, trainers have various options for building competitive teams with Grass and Water types. This variety allows players to experiment with different team compositions, adding depth and complexity to the game.

2. What is the average Defense of all the **Pokémon** that have Attack greater than 30 and are not Legendary? Explain how you can interpret the results.

Answer 2A.2:

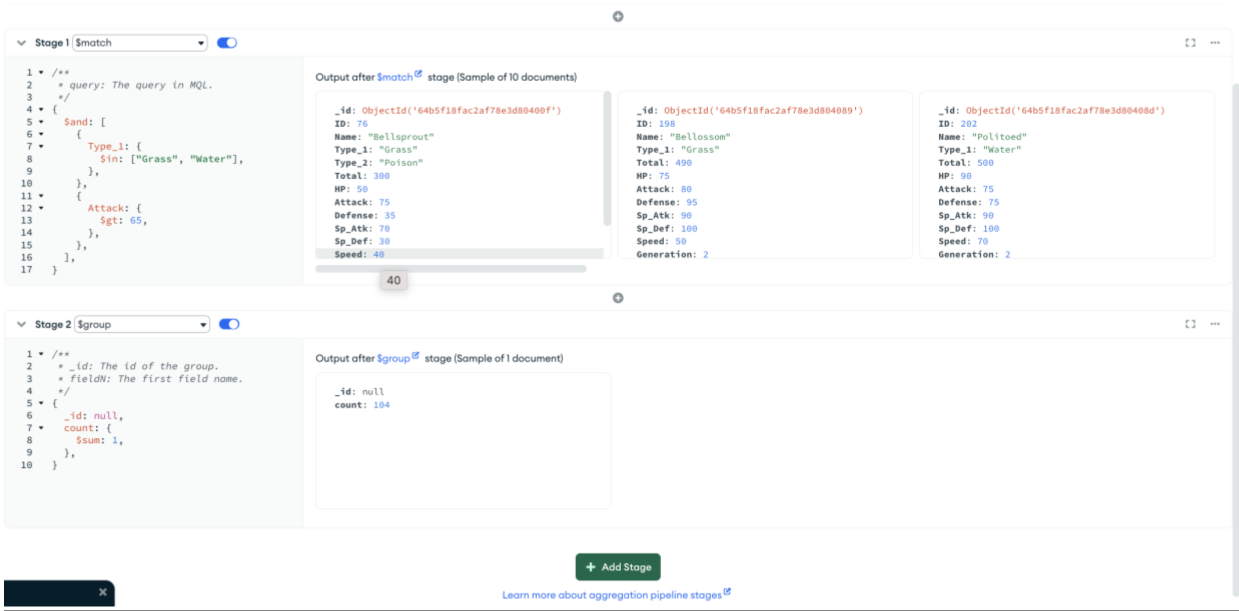
The Pokémon meeting the specified criteria in the question exhibit an **average defense of 73.3**. This indicates that this group of Pokémon (those with Attack greater than 30 and not Legendary) generally possess higher Defense attributes, making them more capable of enduring incoming attacks in battles. Game developers can utilize this average Defense value of 73.3 to evaluate the balance of Defense attributes across different Pokémon types, ensuring competitive and engaging battles with a diverse range of viable Pokémon choices. Players can leverage this information to strategically construct teams that complement each other's strengths and weaknesses effectively.

Question 2B

Please paste any code that you have designed for Question 2A and explain which sub-question does it belong to. You will receive 5 points for correct code for Question 2A.1 and another 5 points for correct code for Question 2A.2.

Answer 2B:

Code for 2A.1



```
/**
 * query: The query in MQL.
 */
{
  $and: [
    {
      Type_1: {
        $in: ["Grass", "Water"],
      },
    },
    {
      Attack: {
        $gt: 65,
      },
    },
  ],
}
```

Output after $match stage (Sample of 10 documents)

```
_id: ObjectId('64b5f18fac2af78e3d80400f')
ID: 76
Name: "Bellsprout"
Type_1: "Grass"
Type_2: "Poison"
Total: 300
HP: 50
Attack: 75
Defense: 35
Sp_Atk: 70
Sp_Def: 30
Speed: 40
```

```
_id: ObjectId('64b5f18fac2af78e3d804089')
ID: 198
Name: "Bellossom"
Type_1: "Grass"
Total: 490
HP: 75
Attack: 80
Defense: 95
Sp_Atk: 90
Sp_Def: 100
Speed: 50
Generation: 2
```

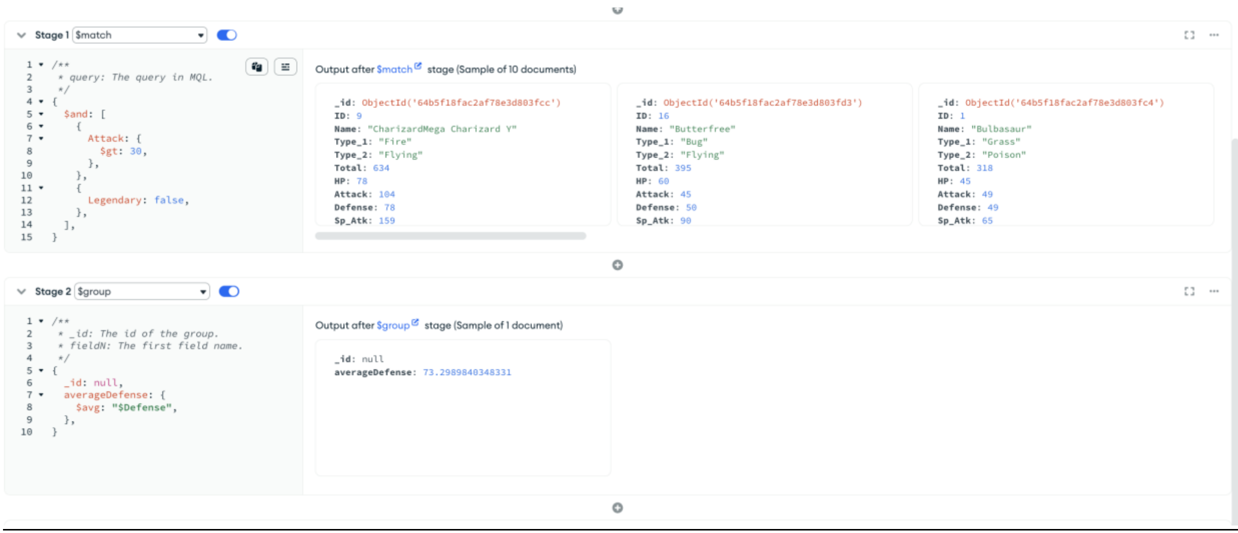
```
_id: ObjectId('64b5f18fac2af78e3d80408d')
ID: 202
Name: "Politoed"
Type_1: "Water"
Total: 500
HP: 90
Attack: 75
Defense: 75
Sp_Atk: 90
Sp_Def: 100
Speed: 70
Generation: 2
```

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: null,
  count: {
    $sum: 1,
  },
}
```

Output after $group stage (Sample of 1 document)

```
_id: null
count: 104
```

Code for 2A.2



Stage 1 $match

```
/**
 * query: The query in MQL.
 */
{
  $and: [
    {
      Attack: {
        $gt: 30,
      },
    },
    {
      Legendary: false,
    },
  ],
}
```

Output after $match stage (Sample of 10 documents)

```
_id: ObjectId('64b5f18fac2af78e3d803fcc')
ID: 9
Name: "CharizardMega Charizard Y"
Type_1: "Fire"
Type_2: "Flying"
Total: 634
HP: 78
Attack: 104
Defense: 78
Sp_Atk: 159
```

```
_id: ObjectId('64b5f18fac2af78e3d803fd3')
ID: 16
Name: "Butterfree"
Type_1: "Bug"
Type_2: "Flying"
Total: 395
HP: 60
Attack: 45
Defense: 50
Sp_Atk: 90
```

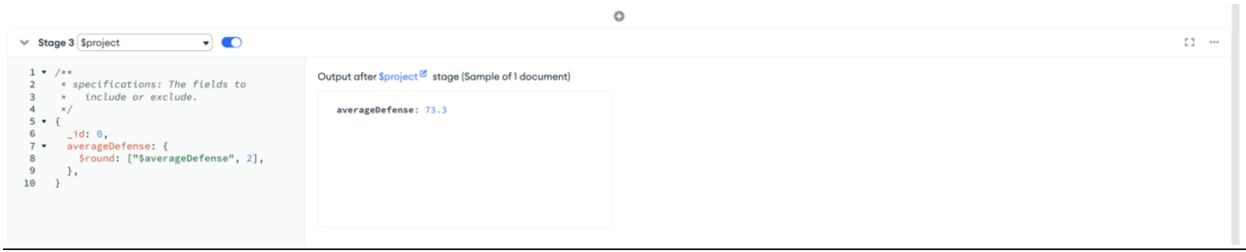
```
_id: ObjectId('64b5f18fac2af78e3d803fc4')
ID: 1
Name: "Bulbasaur"
Type_1: "Grass"
Type_2: "Poison"
Total: 318
HP: 45
Attack: 49
Defense: 49
Sp_Atk: 65
```

Stage 2 $group

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: null,
  averageDefense: {
    $avg: "$Defense",
  },
}
```

Output after $group stage (Sample of 1 document)

```
_id: null
averageDefense: 73.2989840348331
```



Stage 3 $project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  _id: 0,
  averageDefense: {
    $round: ["$averageDefense", 2],
  },
}
```

Output after $project stage (Sample of 1 document)

```
averageDefense: 73.3
```

Question 3A

The following piece of code is from a linear regression PySpark pipeline:

```
#creating vectors with names of variables
vecAssembler = VectorAssembler(inputCols = [ 'Sp_Atk', 'Attack', 'Sp_Def', 'Defense'], outputC
v_df = vecAssembler.transform(df)
vhouse_df = v_df.select(['features', 'HP'])
vhouse_df = vhouse_df.withColumnRenamed('HP', "label")# We have to rename our output variable
```

The entire PySpark code will result in the following output:

```
Coefficients: [0.08773497752009,0.25167312899184546,0.22548246788120335,-0.06425439149671541]
Intercept: 31.419484729283447
numIterations: 11
objectiveHistory: [0.49999999999999956, 0.47127769219867627, 0.3940393577345407, 0.3907469901
681305, 0.39084289250304255, 0.38729670884344675, 0.38720654002070726, 0.3871638287842702, 0.
3871634888311251, 0.38716348538805406, 0.3871634853517291]
+-------------------+
|          residuals|
+-------------------+
|-51.896023362835024|
| -27.18837934289391|
| -6.473228686298171|
| -16.22789850730797|
|  7.358645561347153|
|-20.954900511489257|
|  5.343230085098327|
```

Please interpret the output. What is the business insight from the output? Please be as specific as possible.

Answer 3A:

The coefficients represent the weights assigned to each input feature (Sp Atk, Attack, Sp_Def, Defense) in the linear regression model. These coefficients indicate the impact of each input feature on the predicted output (HP in this case). Positive coefficients (e.g., 'Sp Atk' and 'Attack') imply that an increase in the corresponding feature will result in a higher predicted HP, while a negative coefficient (e.g., 'Defense') suggests that an increase in 'Defense' will lead to a decrease in the predicted HP. The interpretations of each feature are as follows:

- **'Sp Atk'**: An increase of one unit in 'Sp Atk' will lead to an increase of approximately 0.0877 in the predicted HP value. This suggests that Pokémon with higher Special Attack tend to have slightly higher HP values.
- **'Attack'**: An increase of one unit in 'Attack' will result in an increase of approximately 0.2517 in the predicted HP value. This indicates that Pokémon with higher Attack attributes tend to have significantly higher HP values.
- **'Sp_Def'**: An increase of one unit in 'Sp_Def' will cause an increase of approximately 0.2255 in the predicted HP value. This implies that Pokémon with higher Special Defense tend to have slightly higher HP values.
- **'Defense'**: An increase of one unit in 'Defense' will lead to a decrease of approximately -0.0643 in the predicted HP value. This means that Pokémon with higher Defense attributes may have slightly lower HP values.

Business Insights

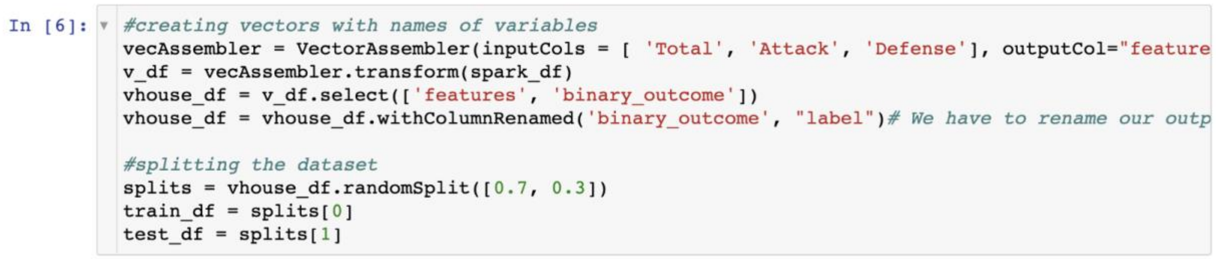
Understanding the correlation between HP and attributes like Sp Atk, Attack, Sp_Def, and Defense can help trainers and players identify the optimal attribute balance for Pokémon. They can prioritize attributes that positively impact HP, leading to more powerful and durable Pokémon in battles.

By focusing on attributes that have the most significant impact on HP, trainers can specialize Pokémon for specific battle roles. For example, Pokémon with high Attack can be used for dealing heavy damage, while those with high Defense can act as tanks to absorb hits.

Trainers can leverage the insights to strategically build teams with Pokémon that possess a balance of offensive (Sp Atk, Attack) and defensive (Sp_Def, Defense) attributes. This approach can increase the team's overall survivability and battle performance.

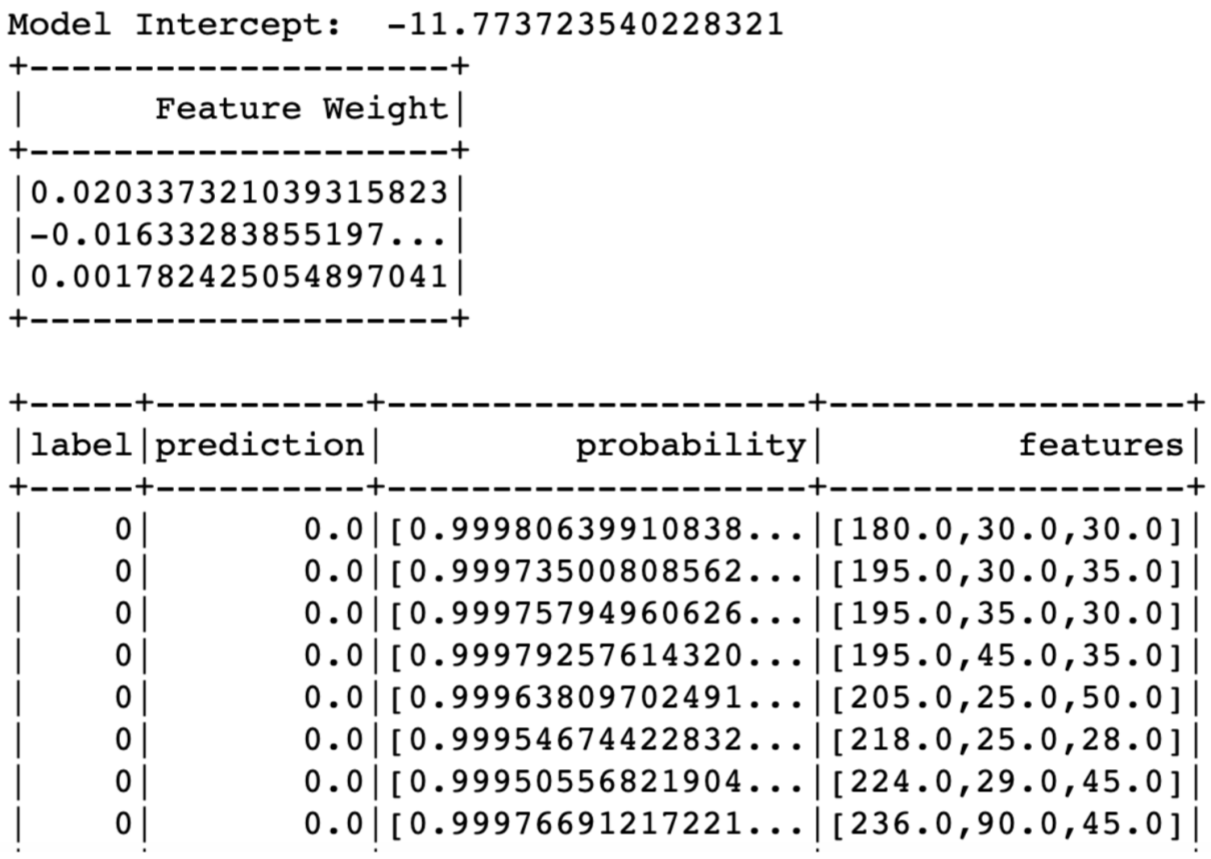
Question 3B

The following piece of code is from a logistic regression PySpark pipeline.



```
In [6]: #creating vectors with names of variables
        vecAssembler = VectorAssembler(inputCols = [ 'Total', 'Attack', 'Defense'], outputCol="feature
        v_df = vecAssembler.transform(spark_df)
        vhouse_df = v_df.select(['features', 'binary_outcome'])
        vhouse_df = vhouse_df.withColumnRenamed('binary_outcome', "label")# We have to rename our outp

        #splitting the dataset
        splits = vhouse_df.randomSplit([0.7, 0.3])
        train_df = splits[0]
        test_df = splits[1]
```

The entire PySpark code will produce the following outputs:



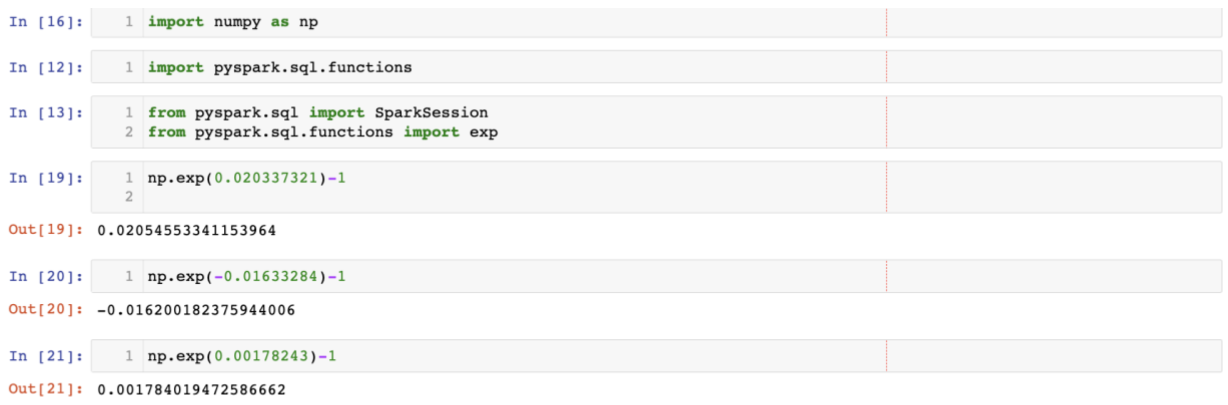
```
Model Intercept:  -11.773723540228321
+--------------------+
|      Feature Weight|
+--------------------+
|0.020337321039315823|
|-0.01633283855197...|
|0.001782425054897041|
+--------------------+

+-----+----------+--------------------+-----------------+
|label|prediction|         probability|         features|
+-----+----------+--------------------+-----------------+
|    0|       0.0|[0.99980639910838...|[180.0,30.0,30.0]|
|    0|       0.0|[0.99973500808562...|[195.0,30.0,35.0]|
|    0|       0.0|[0.99975794960626...|[195.0,35.0,30.0]|
|    0|       0.0|[0.99979257614320...|[195.0,45.0,35.0]|
|    0|       0.0|[0.99963809702491...|[205.0,25.0,50.0]|
|    0|       0.0|[0.99954674422832...|[218.0,25.0,28.0]|
|    0|       0.0|[0.99950556821904...|[224.0,29.0,45.0]|
|    0|       0.0|[0.99976691217221...|[236.0,90.0,45.0]|
```

1. Please transform the coefficients to an interpretable form. (5 points for correct calculations)
2. Interpret your transformed coefficients. What is the business insight? (5 points for correct business inerpretations)

Answer 3B.1:

The coefficients can be transformed into interpretable form in Pyspark using the formula **exp(coeff) - 1**.

Following is the screenshot from Jupyter Notebook transforming the coefficients:



```
In [16]:  import numpy as np

In [12]:  import pyspark.sql.functions

In [13]:  from pyspark.sql import SparkSession
          from pyspark.sql.functions import exp

In [19]:  np.exp(0.020337321)-1

Out[19]: 0.02054553341153964

In [20]:  np.exp(-0.01633284)-1
Out[20]: -0.016200182375944006

In [21]:  np.exp(0.00178243)-1
Out[21]: 0.001784019472586662
```

Thus, the feature weights in transformed state are as follows. These can be used to interpret the impact in odds of success of the business outcome.

Feature	**Transformed Coefficients**	**Percentage Increase in odds of success (rounded)**
Total	0.0205455	2.05%
Attack	-0.0162002	-1.62%
Defense	0.0017840	0.18%

Answer 3B.2:

Interpretation of transformed coefficients

The transformed coefficients give us the percentage change in odds for the positive class (binary outcome) associated with a one-unit increase in each respective attribute. The interpretations are as follows:

Total: The transformed weight of approximately 0.0205 (rounded to four decimal places) for the 'Total' attribute indicates that a one-unit increase in the 'Total' attribute leads to an approximate 2.05% increase in the odds for the positive class occurrence. Pokémon with higher 'Total' stats have a 2.05% higher chance of belonging to the positive class.
Attack: The transformed weight of approximately -0.0162 (rounded to four decimal places) for the 'Attack' attribute indicates that a one-unit increase in the 'Attack' attribute leads to an approximate 1.62% decrease in the odds for the positive class occurrence. Pokémon with higher 'Attack' stats have a 1.62% lower chance of belonging to the positive class.
Defense: The transformed weight of approximately 0.0017 (rounded to four decimal places) for the 'Defense' attribute indicates that a one-unit increase in the 'Defense' attribute does not significantly impact the odds for the positive class occurrence. The 'Defense' attribute does not contribute to an increase or decrease in the likelihood of the positive class outcome.

Business Insights

Balanced Team Building: Trainers need to consider not only the offensive power ('Attack') of their Pokémon but also their overall 'Total' stats. Striking a balance between Attack and other attributes, such as Defense, can lead to a well-rounded team with a higher likelihood of success in the positive class outcome. Diversifying attributes can be key to achieving victory in various battles and scenarios.
Attribute Optimization: To improve the chances of their Pokémon belonging to the positive class, trainers should prioritize optimizing the 'Total' stats. Finding the right balance between Attack and Defense attributes can lead to better performance overall.