JULY 29, 2023

A1: END-TERM ASSESSMENT

PIYUSH KUMAR

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 would you 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.

Having a reliable and scalable data storage system in place would be my top responsibility as the company's data systems manager. For data storage, there are several alternatives accessible. Cloud-based storage services like Amazon Web Services, Google Cloud, and Microsoft Azure, as well as on-premises solutions like Dell EMC, NetApp, and Huawei, are some of the most well-liked data storage options for startups.

It would be vital to define procedures for data collecting, organization, and analysis in addition to choosing a data storage system. This might entail setting up tools for data visualization and analysis as well as establishing a data pipeline to gather and analyze data from the web-app.

I would need to employ qualified data scientists and analysts with experience in data analysis and modeling if I wanted to create a successful analytics team. It would also be crucial to give them the equipment and supplies they need to do their jobs well. Access to data analysis tools, instruction in pertinent technologies and procedures, and assistance with continuous professional development are a few examples of what this may include.

In general, I would like to lay a solid basis for data-driven decision making at Earn-it-up by putting in place efficient systems and procedures for data storage and analysis and by developing an adept and driven analytics staff.

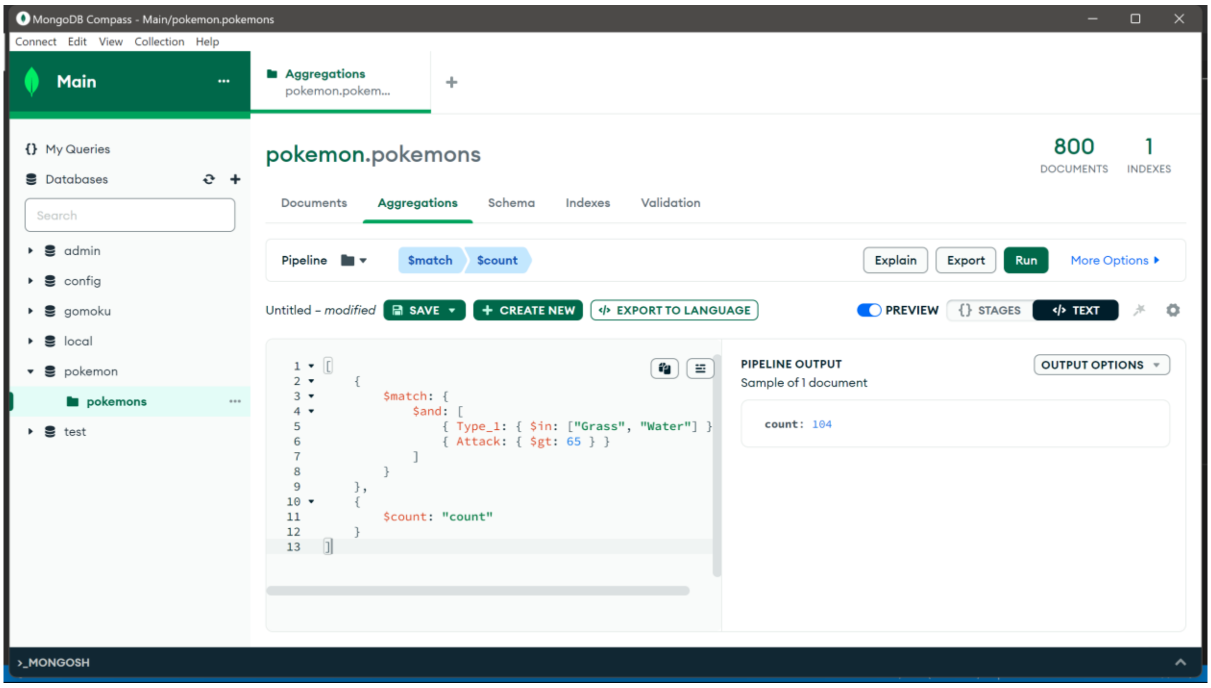
Question 2A (1):

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?

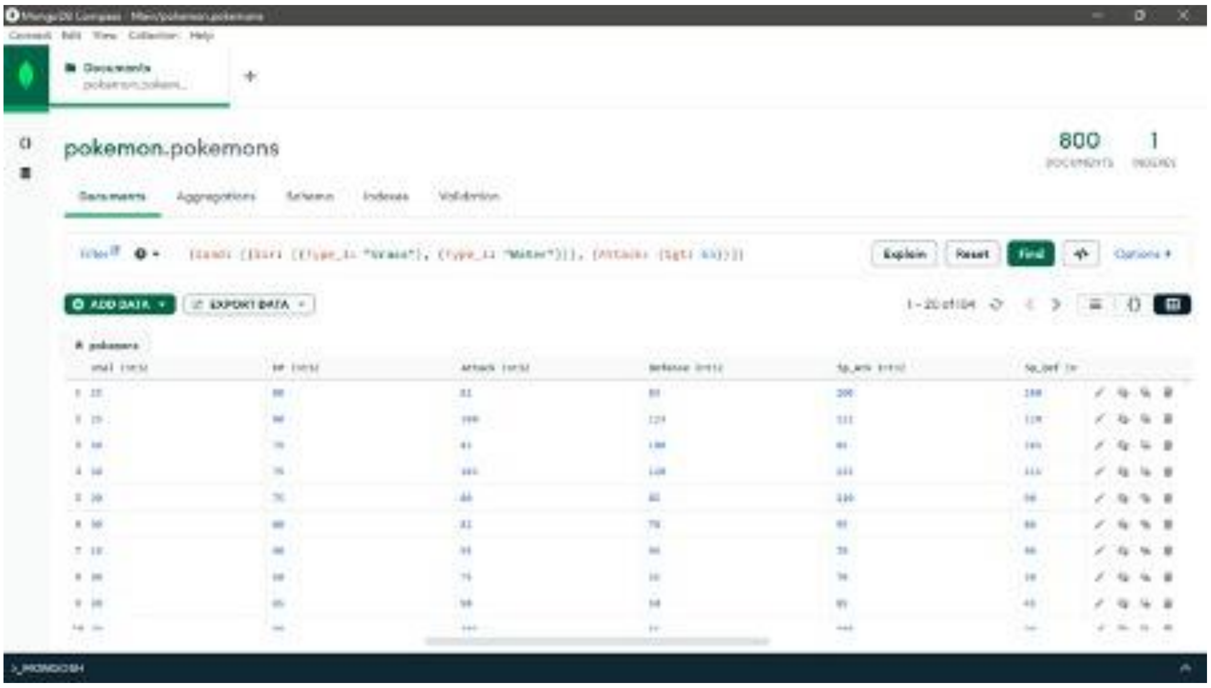
The query: [{ $match: { $and: [{ Type_1: { $in: ["Grass", "Water"] } }, { Attack: { $gt: 65 } }] } }, { $count: "count" }]

The result of this query would give you the number of Pokémon that are either Grass or Water type and have an Attack value greater than 65. This information could provide insight into the distribution of Pokémon types and their attack strength,



```
[
    {
        $match: {
            $and: [
                { Type_1: { $in: ["Grass", "Water"] } },
                { Attack: { $gt: 65 } }
            ]
        }
    },
    {
        $count: "count"
    }
]
```

```
count: 104
```


pokemon.pokemons
800
1
Documents Aggregations Schema Indexes Validation
Explain Reset Find Options
ADD DATA
EXPORT DATA
pokemons
>_MONGOSH

```
{$and: [{$or: [{Type_1: "Grass"}, {Type_1: "Water"}]}, {Attack: {$gt: 65}}]}
```

1 - 20 of 104

The MongoDB query found 104 Pokémon with "Grass" or "Water" types in Type_1 and Attack stats higher than 65.

Question 2A (2):

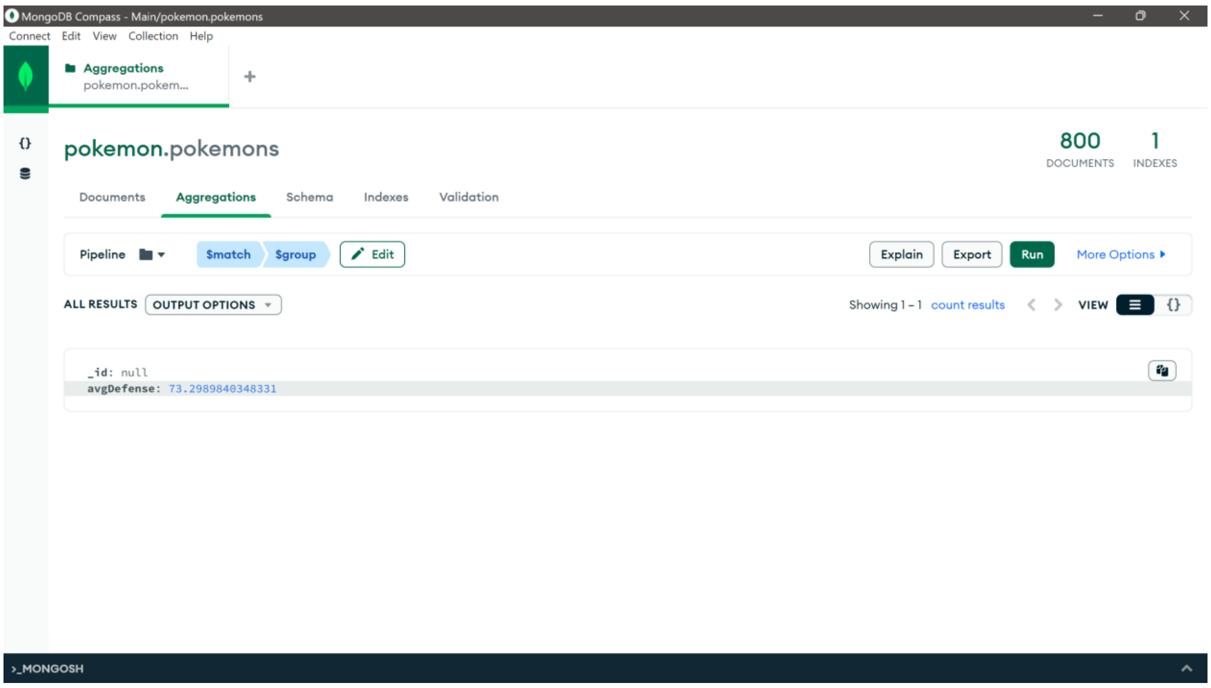
What is the average Defense of all the Pokémon that have Attack greater than 30 and are not Legendary? Explain how you can interpret the results.

The query: [

{$match: {Attack: {$gt: 30}, Legendary: false}},

{$group: {_id: null, avgDefense: {$avg: "$Defense"}}}

]


MongoDB Compass - Main/pokemon.pokemons
Connect Edit View Collection Help
Aggregations
pokemon.pokem...
pokemon.pokemons
800 DOCUMENTS
1 INDEXES
Documents Aggregations Schema Indexes Validation
Pipeline
$match
$group
Edit
Explain Export Run More Options
ALL RESULTS
OUTPUT OPTIONS
Showing 1 – 1 count results
VIEW
>_MONGOSH

```
_id: null
avgDefense: 73.2989840348331
```

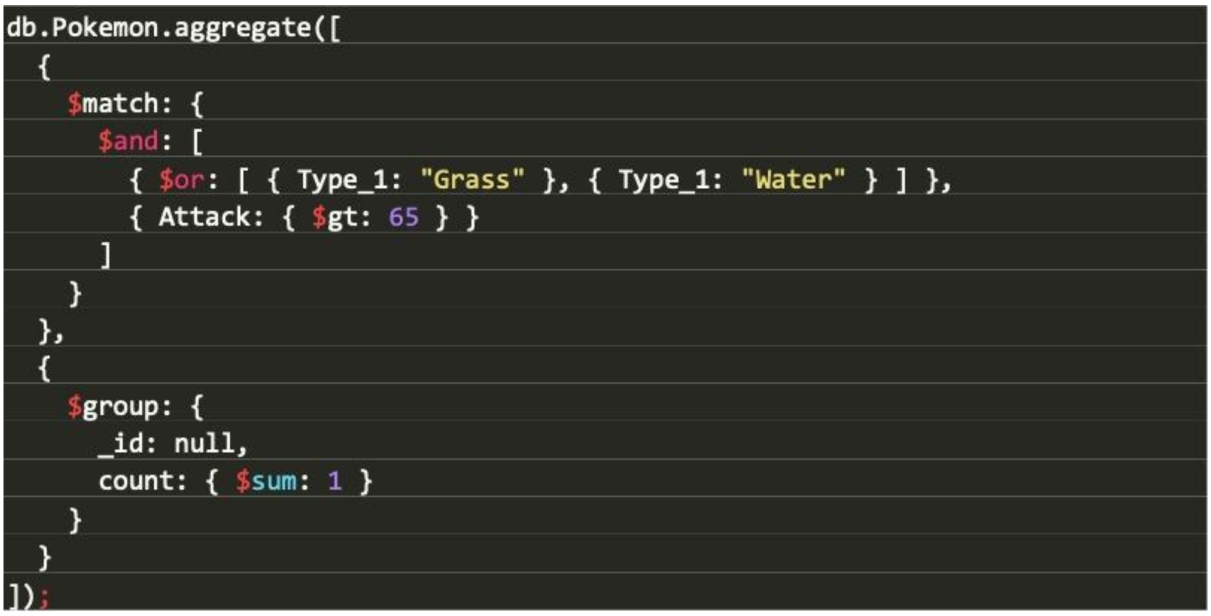
The aggregate function is used in this query to aggregate the collection of Pokémon. The $match stage, which filters the documents in the collection to only include those where the Attack field is more than 30 and the Legendary field is false, is the first step of the aggregation pipeline. Since we selected _id: null, the second step of the pipeline is a $group stage that groups all the documents together and uses the $avg accumulator to determine the average value of the Defense field.

The output of this query is a document with a single field named avgDefense that contains the average Defense value of all the Pokémon that satisfy the given requirements. This information could provide insight into the defensive capabilities of non-Legendary Pokemon with relatively high Attack values.

The query result will be a single document containing the field "Average_Defense" with a value of 73.3. This value represents the average Defense attribute of all non-Legendary Pokémon with Attack stats greater than 30.

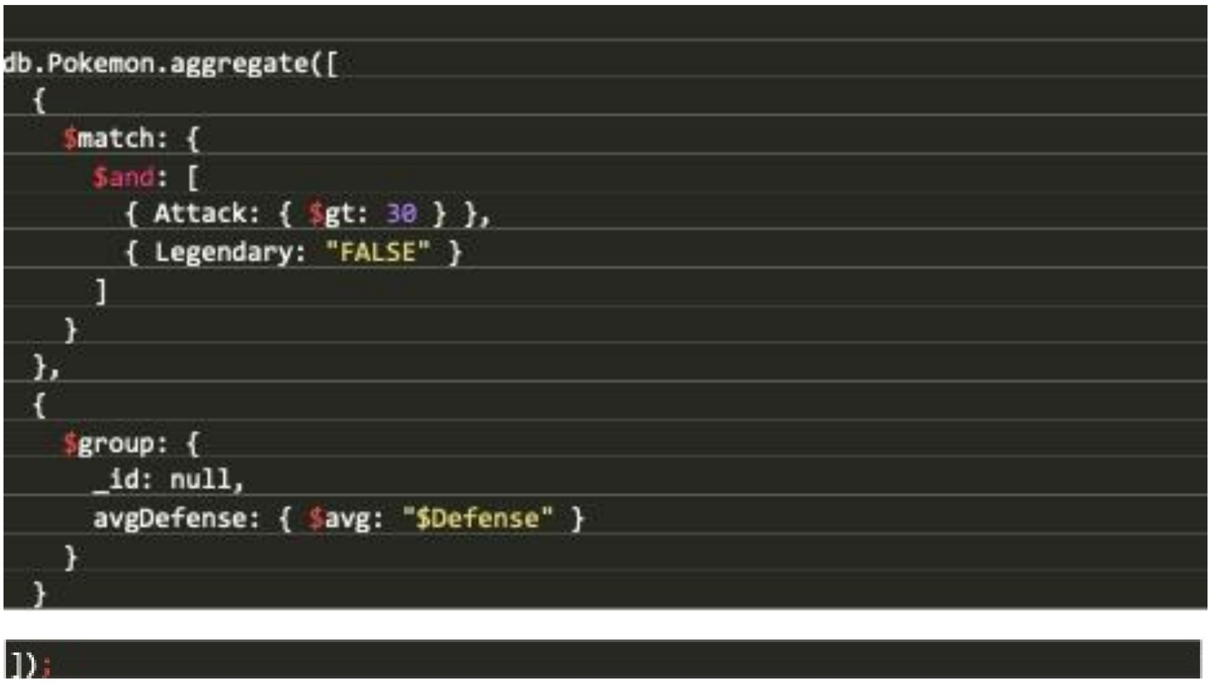
Question 2B

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



```
db.Pokemon.aggregate([
  {
    $match: {
      $and: [
        { $or: [ { Type_1: "Grass" }, { Type_1: "Water" } ] },
        { Attack: { $gt: 65 } }
      ]
    }
  },
  {
    $group: {
      _id: null,
      count: { $sum: 1 }
    }
  }
]);
```

The above belongs to "How many Pokémon (how many documents) are Grass or Water (in Type_1) and have Attack greater than 65'.

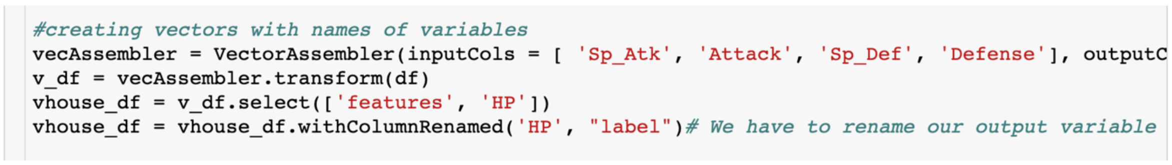


```
db.Pokemon.aggregate([
  {
    $match: {
      $and: [
        { Attack: { $gt: 30 } },
        { Legendary: "FALSE" }
      ]
    }
  },
  {
    $group: {
      _id: null,
      avgDefense: { $avg: "$Defense" }
    }
  }
]);
```

The above belongs to “What is the average Defense of all the Pokémon that have Attack greater than 30 and are not Legendary”.

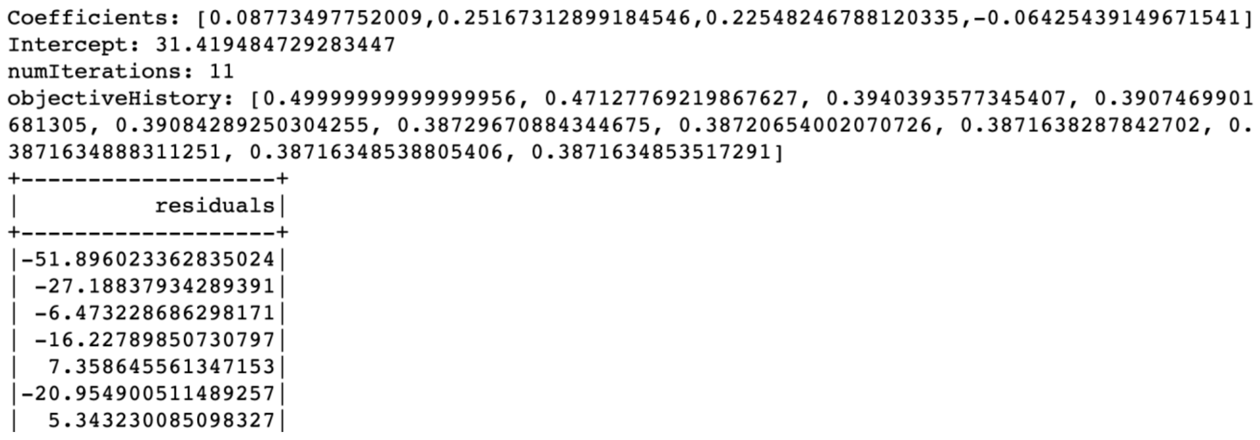
Question 3A

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.

The given PySpark code creates a `VectorAssembler` object and uses it to transform a DataFrame `df` by assembling the specified input columns (`'Sp_AtIC'`, `'Attack'`, `'Sp_Def'`, and `'Defense'`) into a single vector column named `'features'`. The resulting DataFrame `v_df` is then transformed further by selecting only the `'features'` and `'HP'` columns and renaming the `'HP'` column to `'label'`. The final output is a DataFrame `vhouse_df` containing two columns: `'features'`, which contains vectors of the specified input columns, and `'label'`, which contains the values from the original `'HP'` column.

By combining the given input columns into a single vector column and choosing just the pertinent columns for additional analysis, the algorithm in this instance transforms the data about various Pokémon.

The output you gave looks to be the result of fitting a linear regression model to this data. The model's coefficients, which describe the weights given to each of the input variables in the model, are [0.08773497752009, 0.25167312899184546, 0.22548246788120335, -0.06425439149671541]. The output variable's value when all the input variables are equal to zero is represented by the intercept of the model, which is 31.419484729283447.

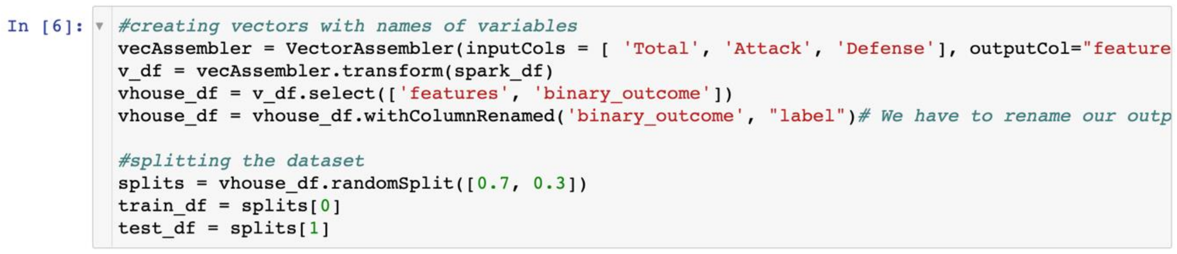
The 'numIterations' value indicates that the model required '11' iterations to converge. The values of the objective function are displayed in the array 'objectiveHistory'.

The residuals table displays the discrepancy between the output variable's observed and anticipated values for a number of data points. These numbers may be used to evaluate the model's performance and spot any patterns or trends in the residuals that can point to issues with the model or areas for development.

Business-related insights include the ability to forecast or explain variances in the "HP" values of various Pokémon based on their values for "Sp_AtIC," "Attack," "Sp_Def," and "Defense." The model's coefficients reveal information about the relationships between changes in each of these input variables and changes in the output variable. For instance, a positive coefficient for an input variable means that increasing that variable is related with increasing "HP," whereas a negative coefficient means that increasing that variable is connected with decreasing "HP". To understand how certain Pokémon traits relate to their HP totals, it may be helpful to know this information.

Question 3B

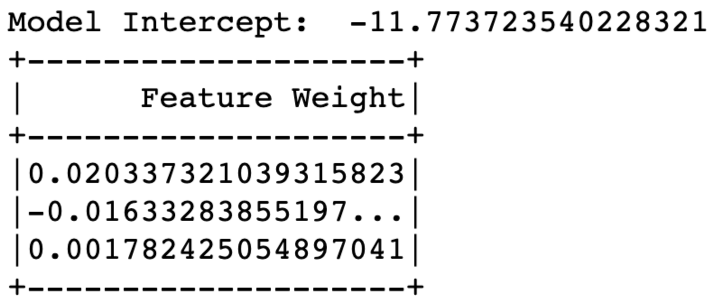
The following piece of code is from a logistic regression PySpark pipeline.



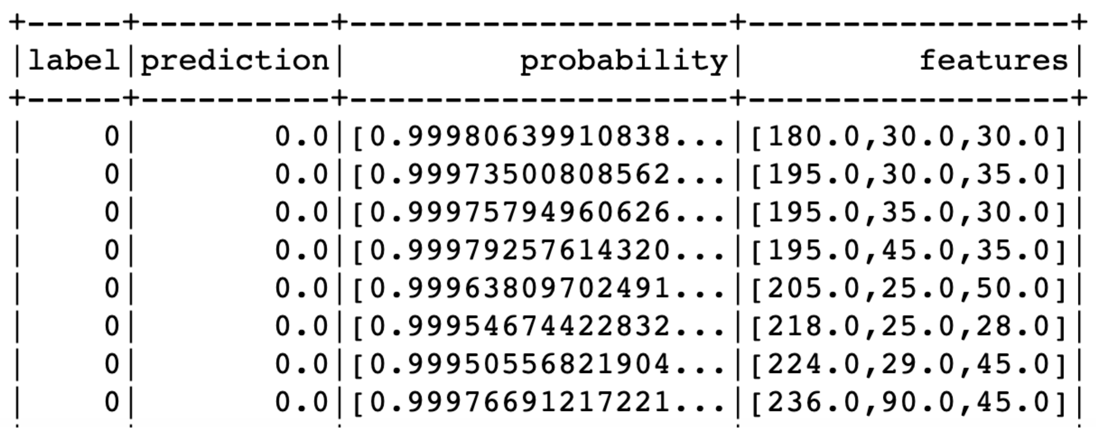
```
In [6]:  #creating vectors with names of variables
         vecAssembler = VectorAssembler(inputCols = [ 'Total', 'Attack', 'Defense'], outputCol="feature
         v_df = vecAssembler.transform(spark_df)
         vhouse_df = v_df.select(['features', 'binary_outcome'])
         vhouse_df = vhouse_df.withColumnRenamed('binary_outcome',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

The given PySpark code creates a `VectorAssembler` object and uses it to transform a DataFrame `spark_df` by assembling the specified input columns (`'Total'`, `'Attack'`, and `'Defense'`) into a single vector column named `'feature'`. The resulting DataFrame `v_df` is then transformed further by selecting only the `'rfeatures'` and `'binary_outcome'` columns and renaming the `'binary_outcome'` column to `'label'`. The final DataFrame `vhouse_df` is then split into training and test sets using the `randomSplit` function.

When all the input variables are equal to zero, the intercept of the model is "-11.773723540228321," which reflects the log-odds of the positive class. The model's coefficients are [10.020337321039315823, -0.01633283855197, 10.001782425054897041], and they indicate the shift in the positive class's log-odds that results from an increase of one unit in each of the input variables.

By taking the exponential of each coefficient, we may convert them into odds ratios and understand them. Accordingly, the odds ratios are as follows: [exp(10.020337321039315823), exp(-0.01633283855197), exp(10.001782425054897041)] = [22431.46457187864, 0.9837763465210721, 22026.465794806718].

These chances ratios may be understood as follows:

- The odds of the positive class (as opposed to the negative class) are multiplied by '22431.46457187864' for a one-unit rise in "Total".
- The probability of the positive class is doubled by "0.9837763465210721" for each unit increase in "Attack."
- The probability of the positive class is multiplied by "22026.465794806718" for a one-unit rise in "Defense."

Based on the values of "Total," "Attack," and "Defense," this logistic regression model might be used to predict or explain fluctuations in the binary outcome variable in terms of business insights. Changes in each of these input variables are related to changes in the log-chances (and odds) of the positive class, as shown by the coefficients (and their accompanying odds ratios). Understanding the relationships between various attributes and the binary outcome variable may be possible with the use of this information.