Georgia Institute of Technology

Project 03

8/08/19

TEAM REPOKÉMON

Machine Learning & Visual Recognition

DEFINITION

POKÉMON IS THE HIGHEST-GROSSING MEDIA FRANCHISE OF ALL-TIME

Right now, it is estimated that Pokémon has earned $61.1 billion USD since its creation in 1996. This total includes everything from its nearly $50 billion retail sales and behemoth mobile presence thanks to Pokémon Go. In second place, Star Wars falls far behind with a still-hefty revenue total of $42.9 billion. Currently, merchandise sales make up the bulk of its money, but its box office totals aren't anything to sneeze at. Checking out the top media franchises, you will see a lot of familiar names. Guys like Mario, Batman, and James Bond can be spotted with ease — but you won't be able to escape anime as you go down the line. In fact, Japan has a pretty solid grip on the franchise-centric list.

FINANCIAL BREAKDOWN BY CHANNEL

- Licensed merchandise – $61.1 billion
- Video games – $17.138 billion
- Card game – $10.853 billion
- Box office – $1.857 billion
- Manga sales – $1.46 billion
- Home entertainment – $863 million

REFERENCE POINT

Pokémon was first introduced in February 27, 1996 for the Nintendo Game Boy system and played a key role in the survival of Nintendo as the introduction of the popular PlayStation game console was soon followed by the release of Microsoft's XBOX.

Most recently Apple reported that the 2016 introduction of "Pokémon Go" was downloaded more times in its first week of release than any other app to date. Since then, the free-to-play game has shown impressive staying power, reaching a total of 800 million downloads by May 2018 and inspiring the next generation of AR-based toys and games.

HYPOTHESIS

POKÉMON ATTRIBUTES IMPACT ON POPULARITY

PHYSICAL/SOCIAL/MOBILE/LOCAL, LOYALTY ACROSS ALL CHANNELS

Despite the success of similar and far more immersive games from lesser known brands, Pokémon Go burst into pop-culture by merging augmented reality technology with the much-adored Pokémon world. The strategy of (re)capturing new and old fans through a highly innovative brand extension has been successful, illustrated by the total distance walked in real life by its players through the game being further than the distance from Earth to Pluto. With the release of further AR gaming extensions from colossal brands already underway (see Star Wars' Find The Force), how can we explain the success of Pokémon Go as an innovative gaming brand extension?

Could there be an influence on the popularity of different Pokémon characters based upon the perceived pleasingness of their color, faces and shapes versus relying only upon their aggregate performance in comparison to each other? As the most successful media franchise of all time, it's worth a look at the attributes, visual recognition and financial impact of the characters within their respective categories.

TEAM

TEAM

MEGAN

IVAN

LISA

SID

MAURICIO

DAN


PERFORMING A DEEP DIVE
OF EXISTING DATA USING
MACHINE LEARNING
RePokémon

TECHNOLOGY UTILIZED

SEABORN:
Python's Statistical Data Visualization Library

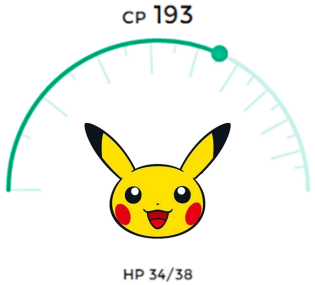
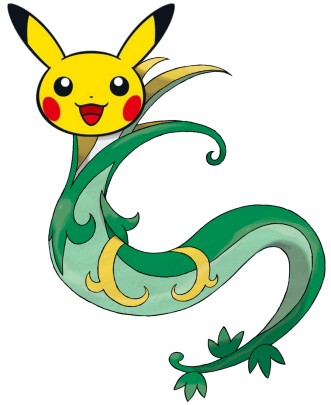


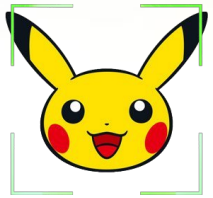
TABLEAU:
Data visualization software



SCIKIT-LEARN
is a free software machine learning library for Python

PYTORCH
A rich ecosystem of tools and libraries

IMAGEAI
State-of-the-art recognition and detection AI



RESNET
is introducing a so-called “identity shortcut connection” that skips one or more layers

RePokémon

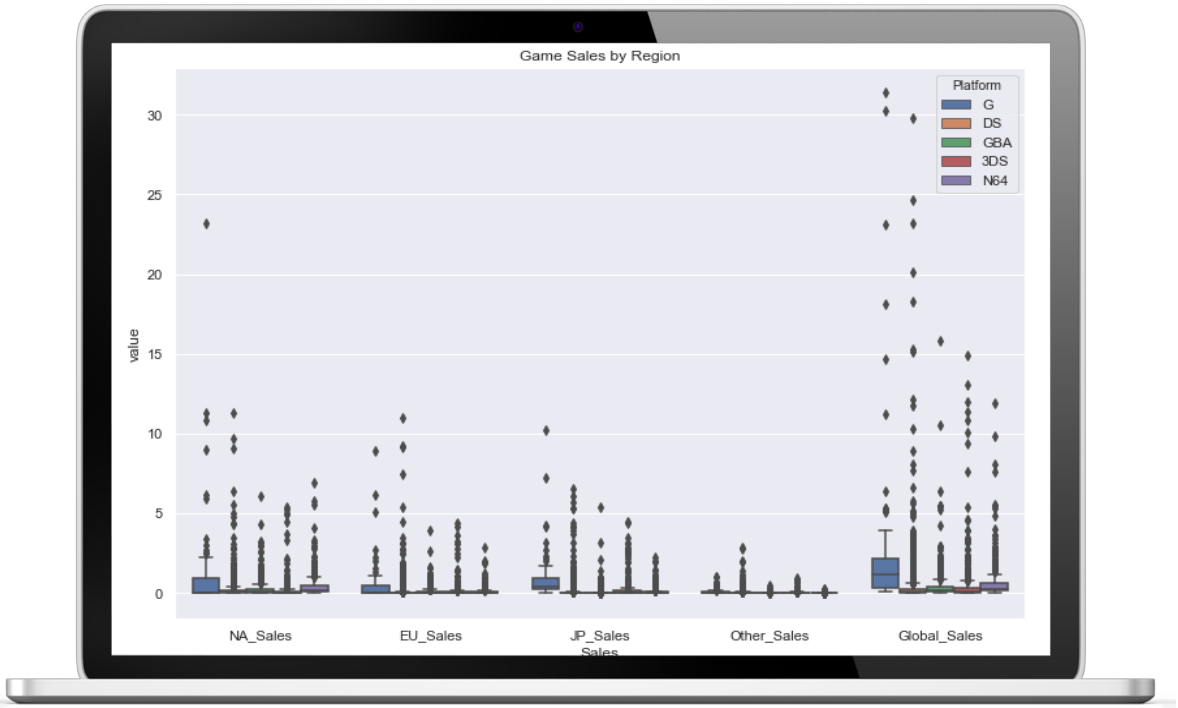

FINANCIAL IMPACT

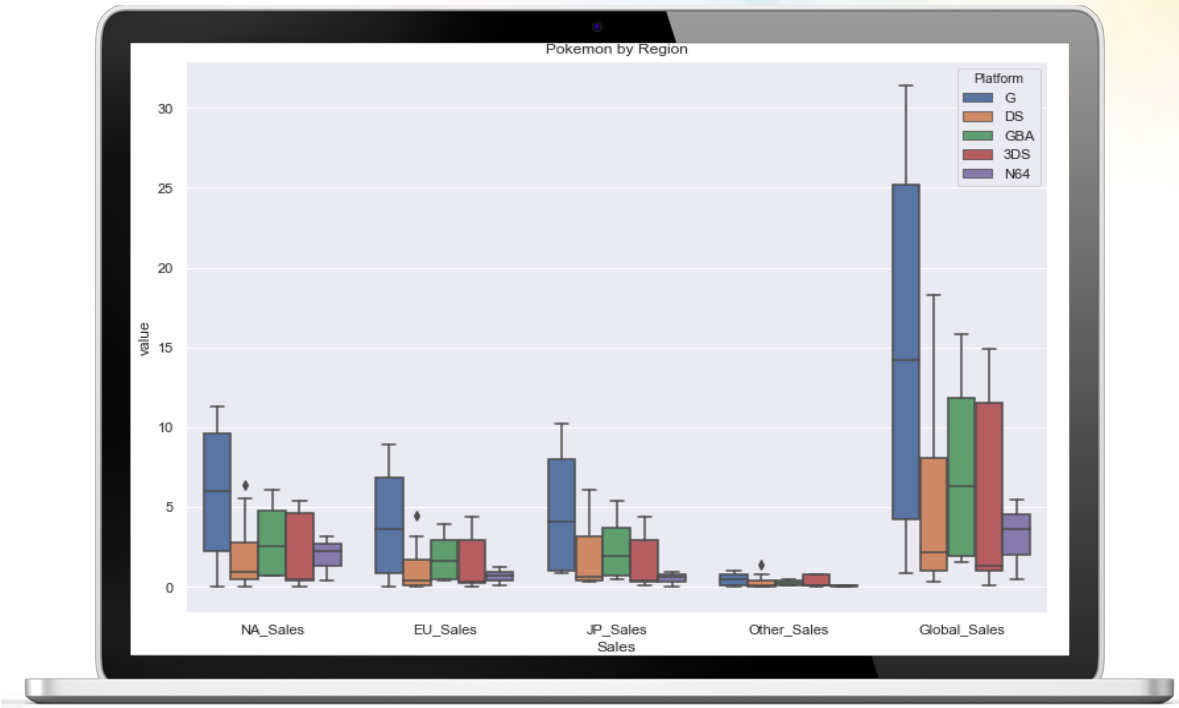
SEABORN

POKÉMON SALES DATA

RePokémon

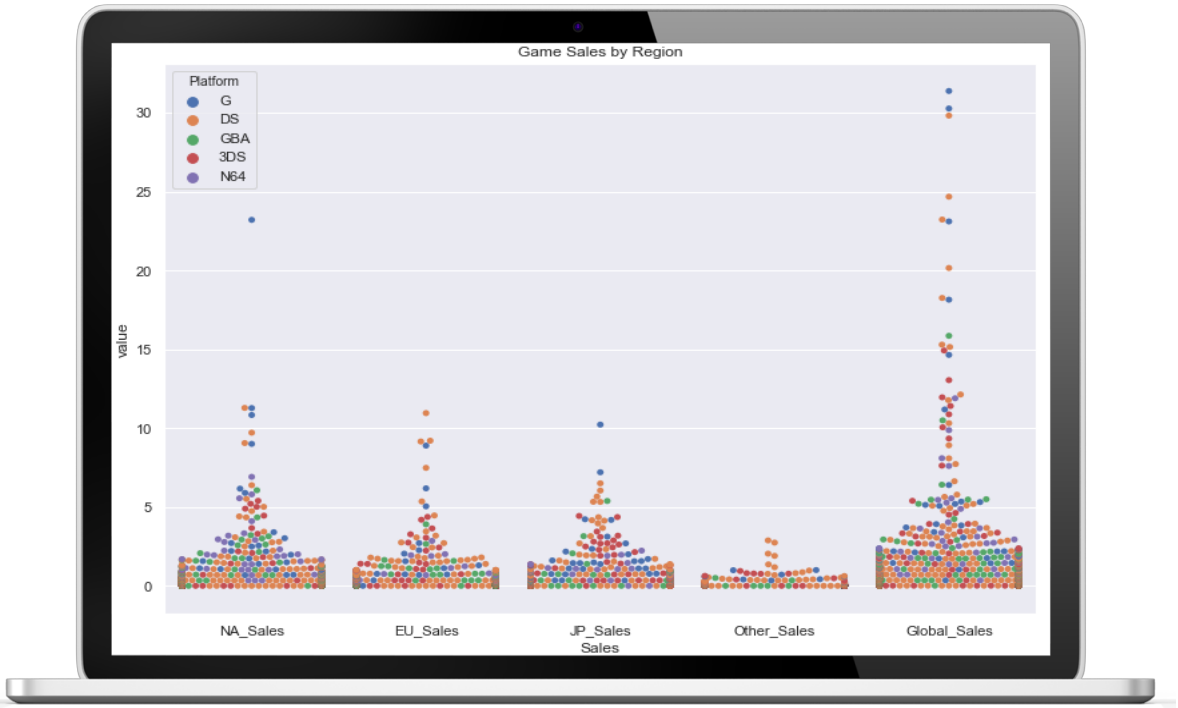
SEABORN

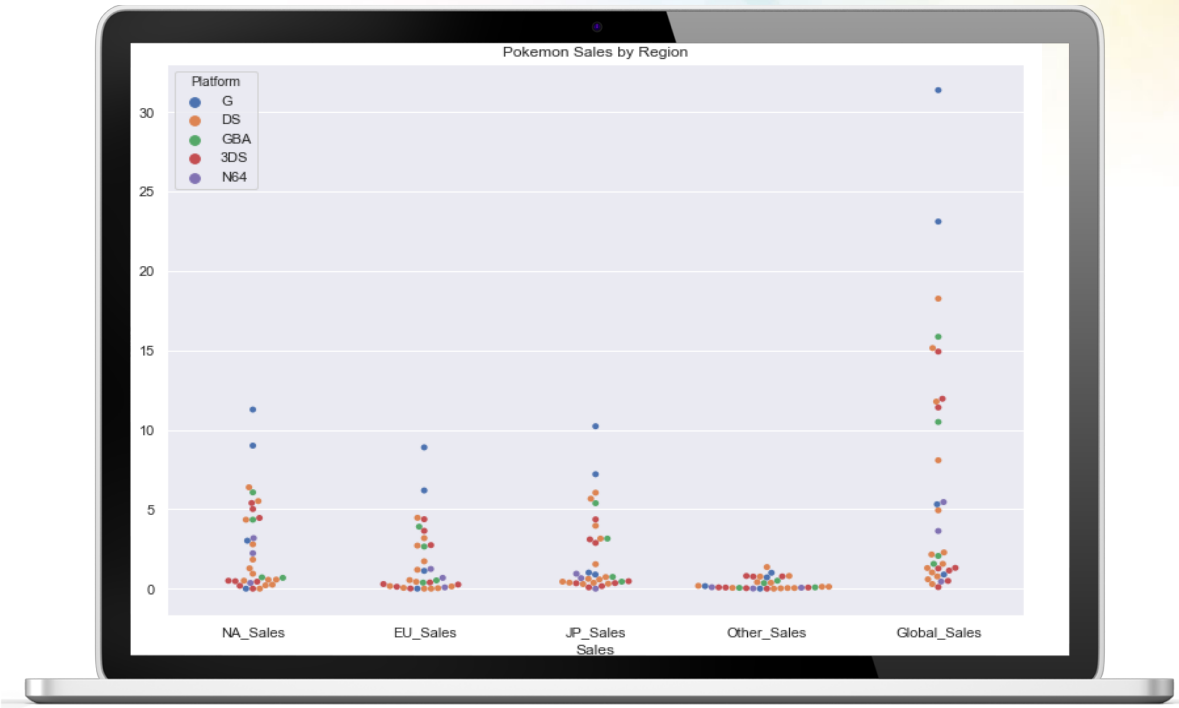




RePokémon

SEABORN





SEABORN TUTORIAL: https://elitedatascience.com/python-seaborn-tutorial

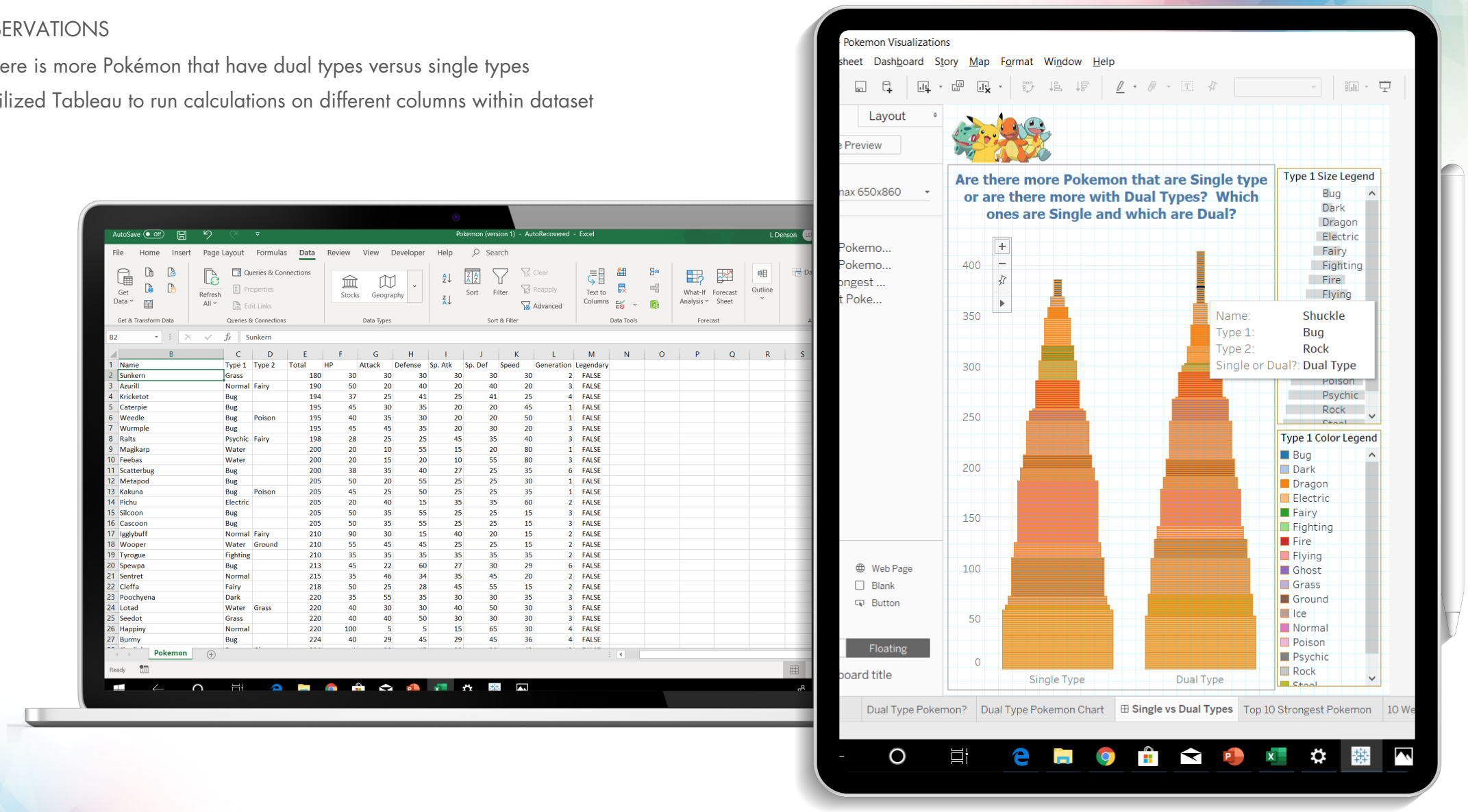
RePokémon


COMPARISON
OF ATTRIBUTES
Grass
Fire
Fighting
Flying
Flying
Psychic
RePokémon

TABLEAU

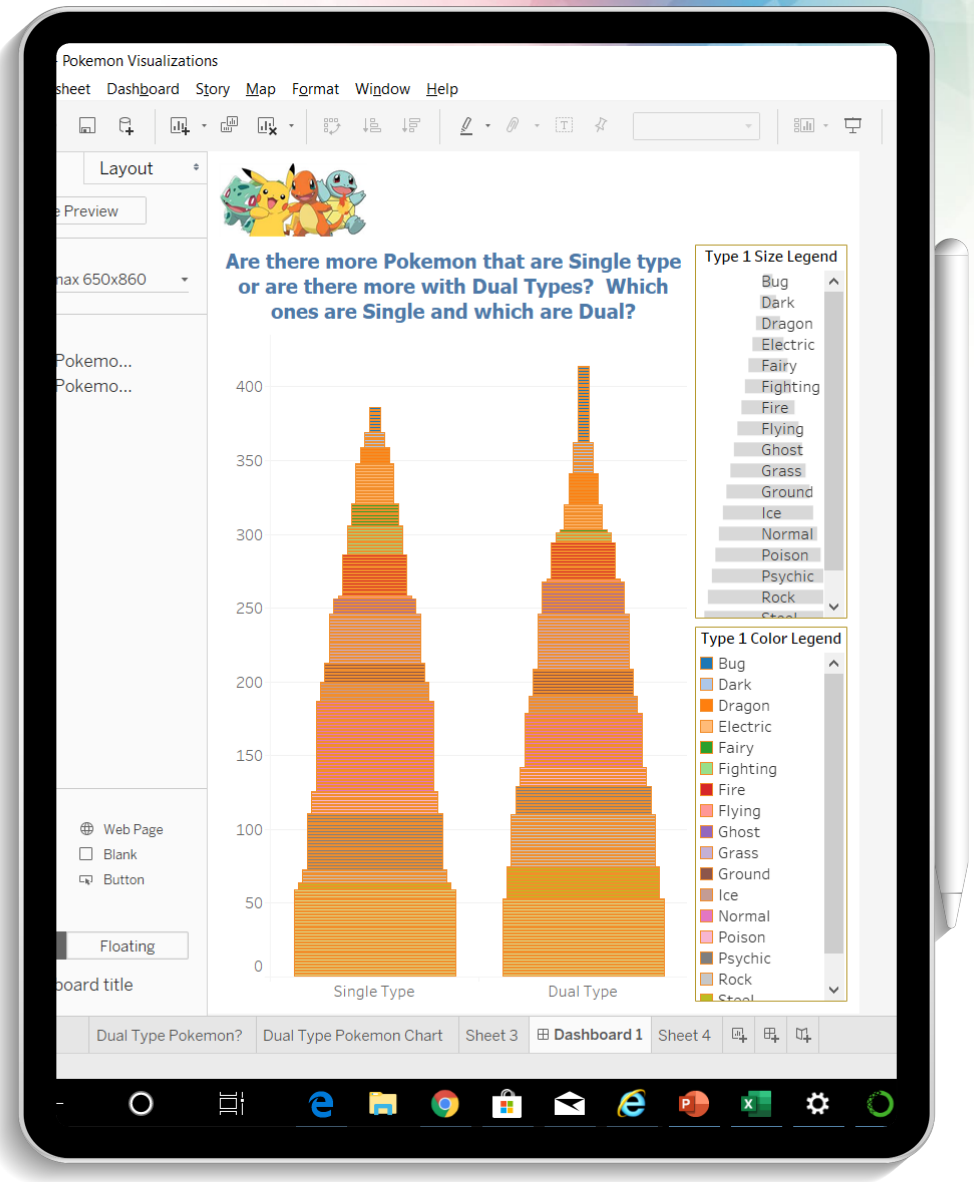
OBSERVATIONS

- There is more Pokémon that have dual types versus single types
- Utilized Tableau to run calculations on different columns within dataset



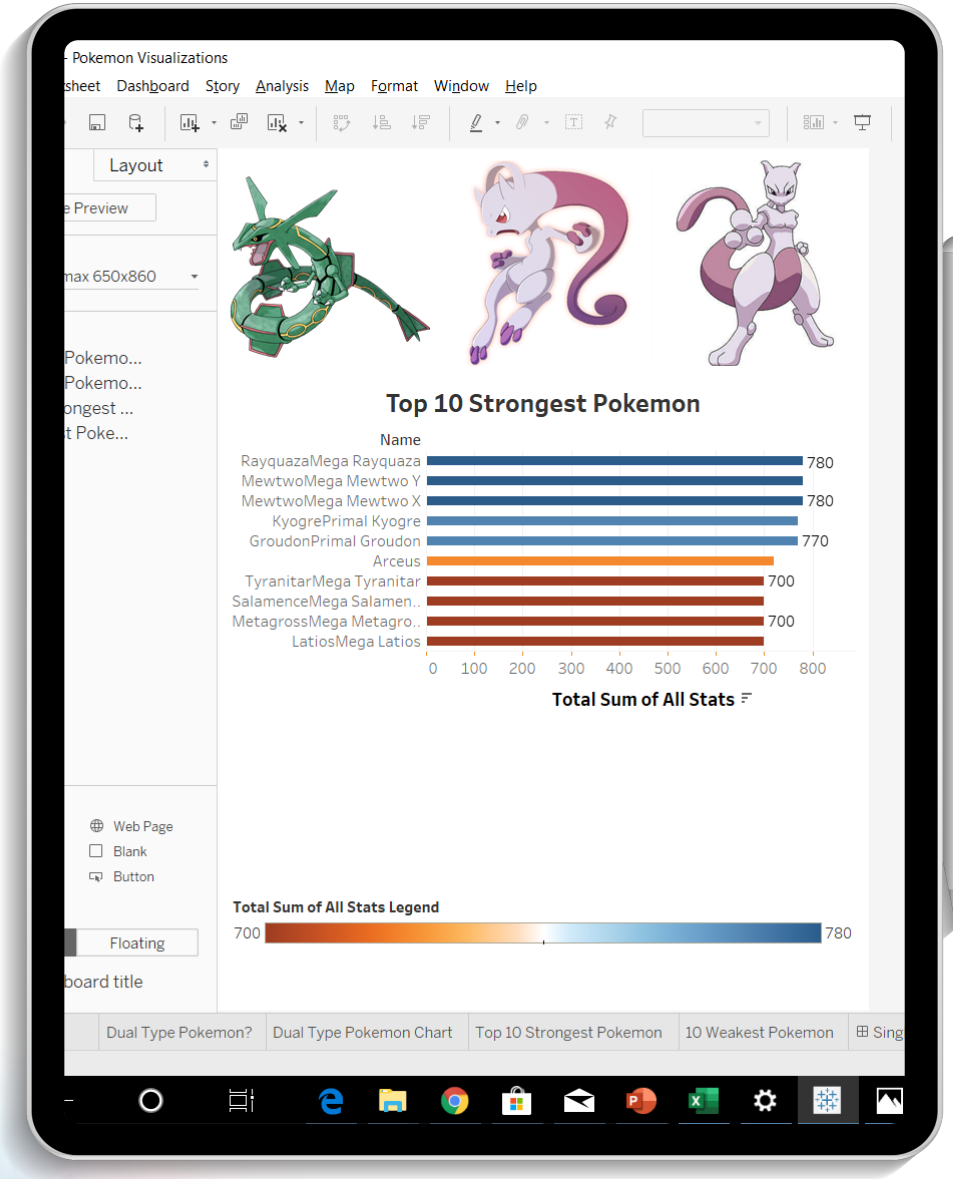
RePokémon

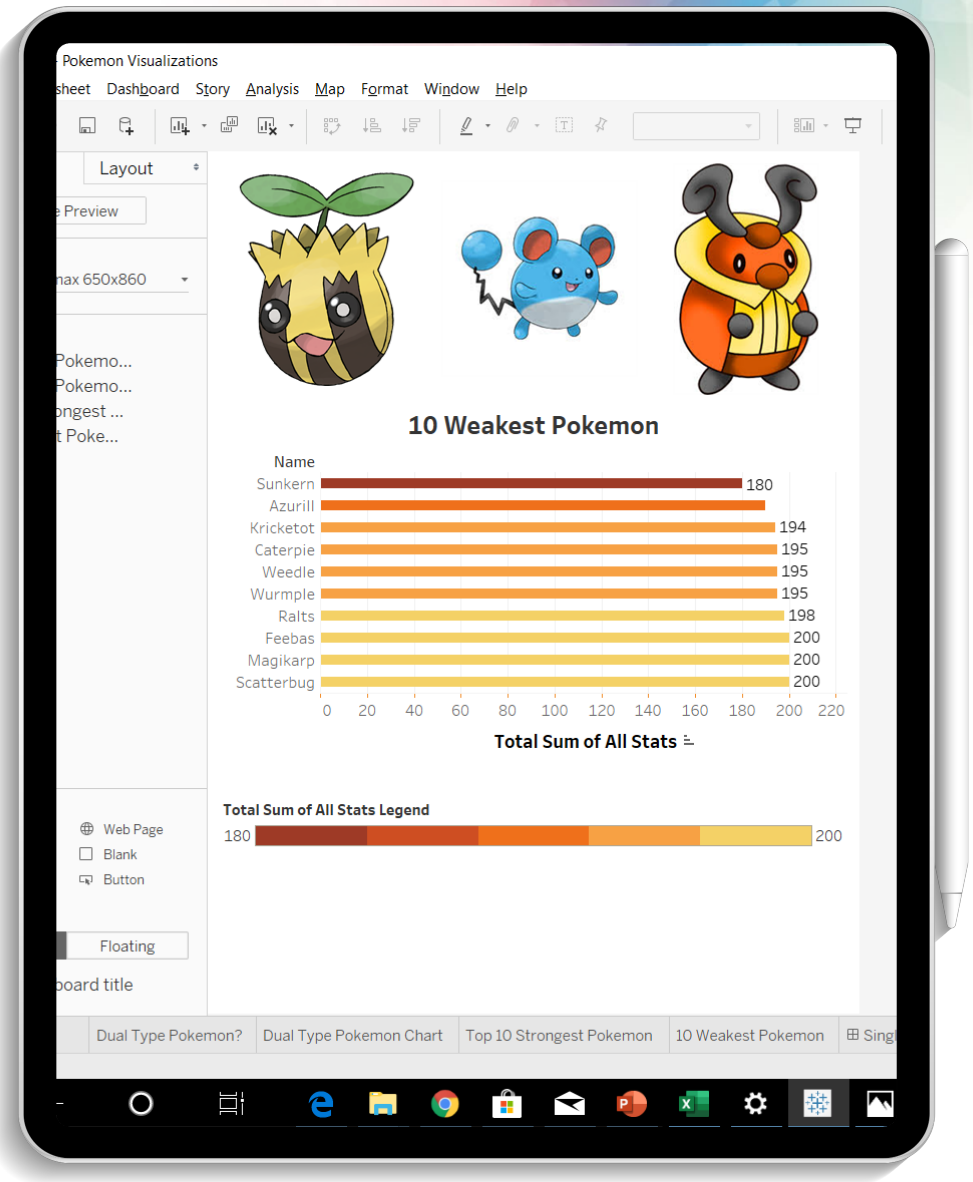
TABLEAU



RePokémon

TABLEAU





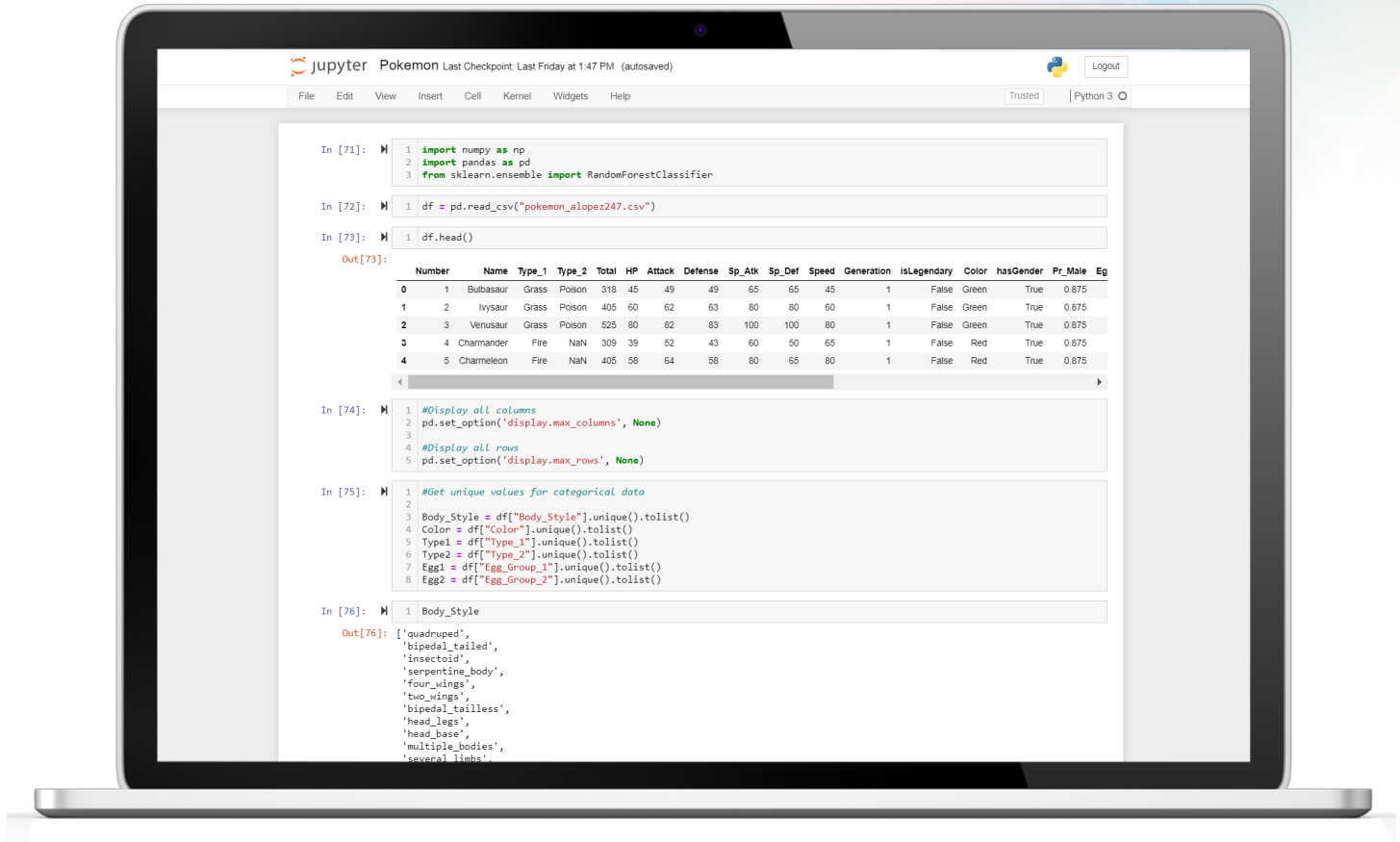
RePokémon

MACHINE LEARNING

SCIKIT-LEARN

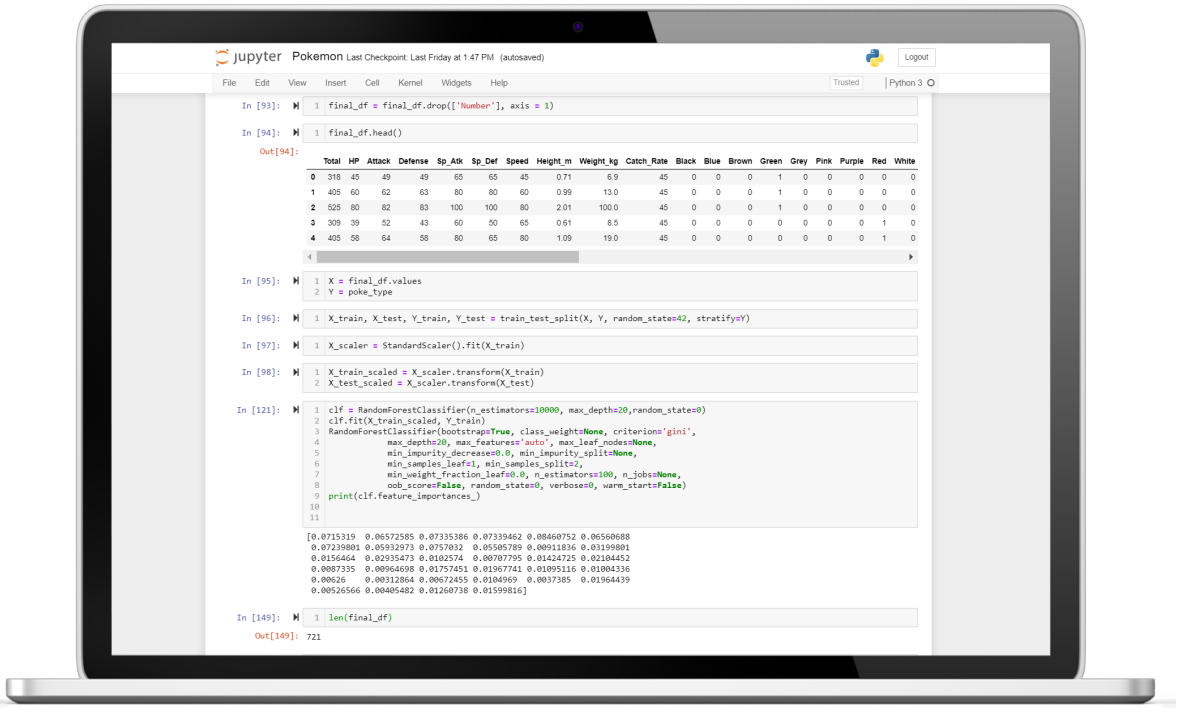
SK LEARN is the library utilized within Python

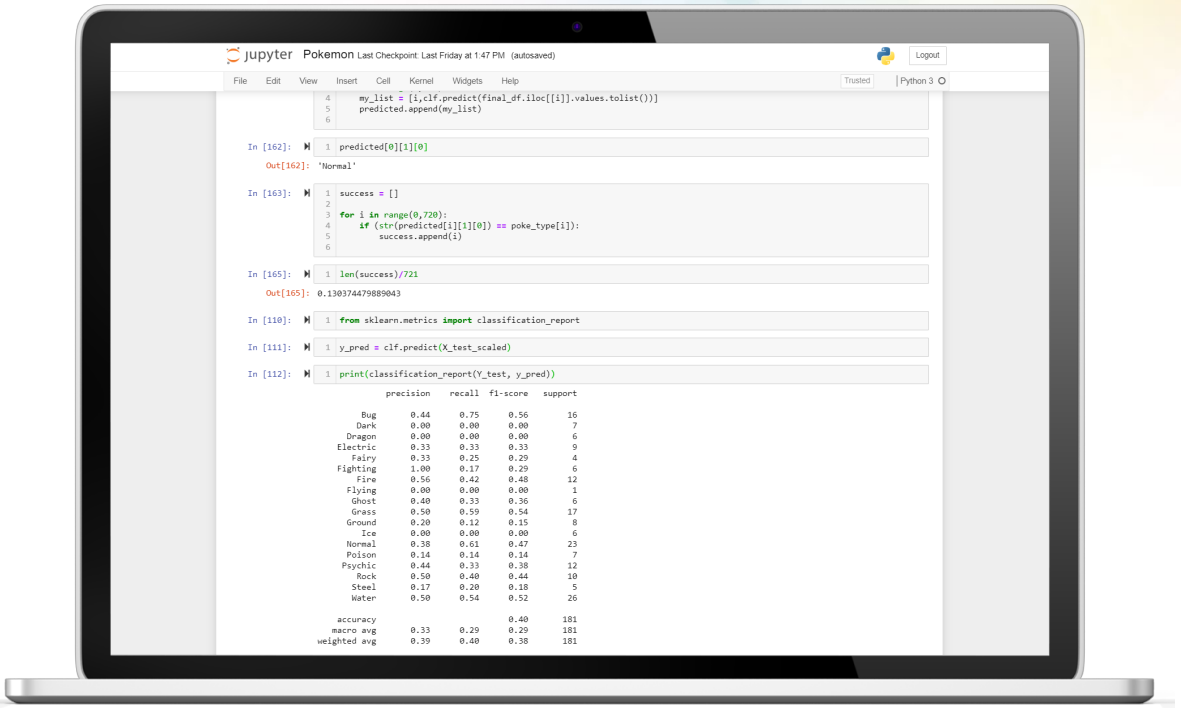
- Random Forrest Classifier
- Pandas Library
- Was not receiving good results on type
- In response, did the prediction to all of them and ran a ‘for-loop’ on all of them
- The percent success on all 721 was 13.04%



RePokémon

SCIKIT-LEARN



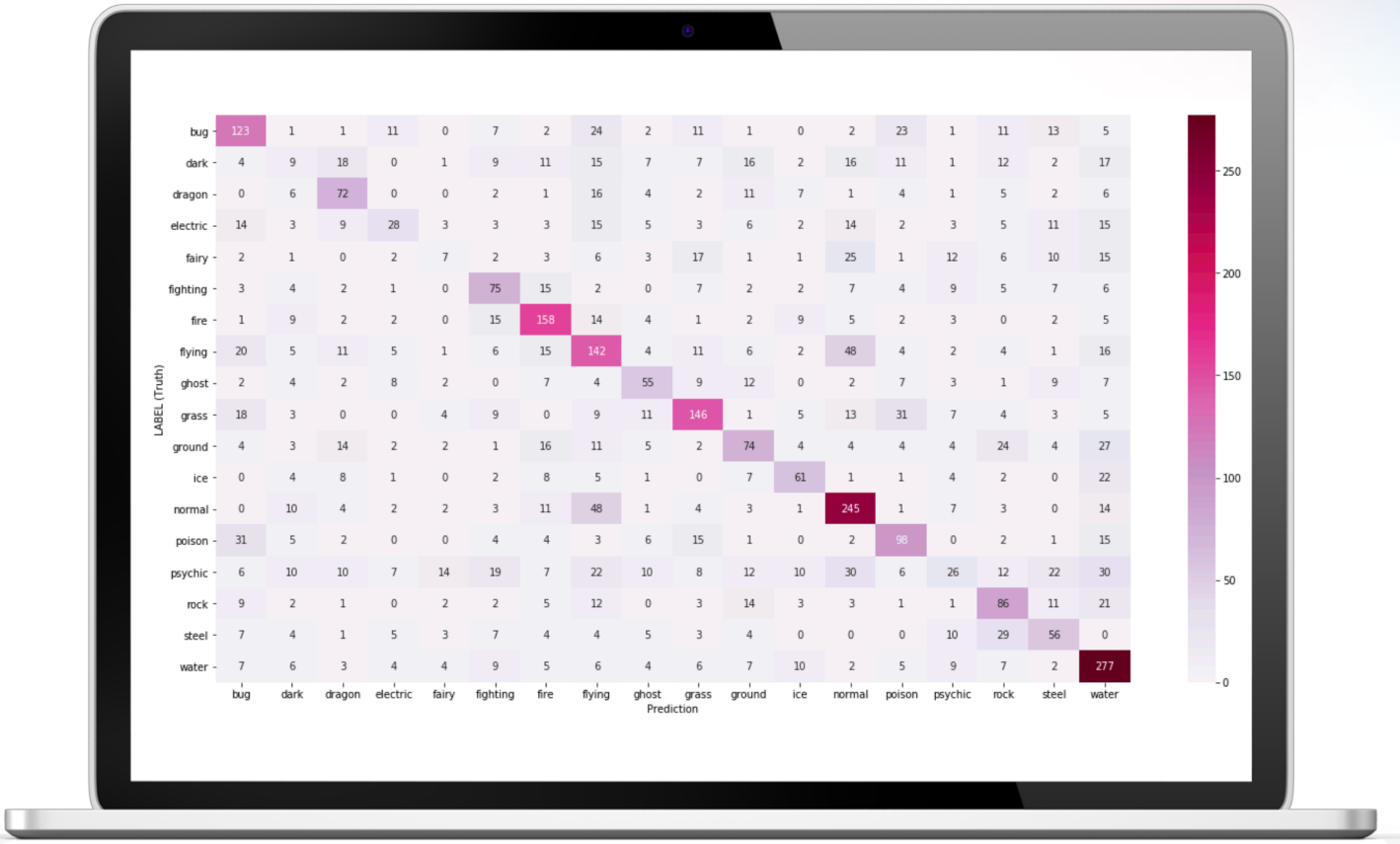


RePokémon


PREDICTION MODELING
WITH PYTORCH
RePokémon

PYTORCH

PREDICTION MODELING



Confusion Matrix

RePokémon

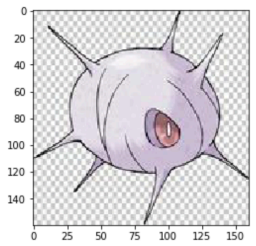
PREDICTION OUTCOMES

PREDICTION MODELING

```
x = 37 # Choose a number between 0 and len(test_data)-1 (3716 --> images in test_data)

image_tensor = test_data[x][0]
x_class = test_data[x][1]
real_class = [item for item in test_class_names if test_class_names[item]== x_class][0]
im = inv_normalize(image_tensor)
plt.imshow(np.transpose(im.numpy(), (1, 2, 0)));
print(f"\nActual Class: {real_class}\n")
```

```
Actual Class: bug
```



```
image_tensor.shape
```

```
torch.Size([3, 160, 160])
```

```
# CNN Model Prediction:
CNNmodel.eval()
with torch.no_grad():
    new_pred = CNNmodel(image_tensor.view(1,3,160,160)).argmax()
print(f'Predicted Class: {new_pred.item()} {class_names[new_pred.item()]}')
```

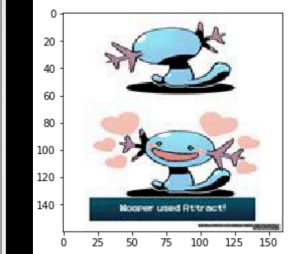
```
Predicted Class: 13 poison
```

Wrong Prediction

```
x = 3716 # Choose a number between 0 and len(test_data)-1 (3716 --> images in test_data)

image_tensor = test_data[x][0]
x_class = test_data[x][1]
real_class = [item for item in test_class_names if test_class_names[item]== x_class][0]
im = inv_normalize(image_tensor)
plt.imshow(np.transpose(im.numpy(), (1, 2, 0)));
print(f"\nActual Class: {real_class}\n")
```

```
Actual Class: water
```




```
image_tensor.shape
```

```
torch.Size([3, 160, 160])
```

```
# CNN Model Prediction:
CNNmodel.eval()
with torch.no_grad():
    new_pred = CNNmodel(image_tensor.view(1,3,160,160)).argmax()
print(f'Predicted Class: {new_pred.item()} {class_names[new_pred.item()]}')
```

```
Predicted Class: 17 water
```

Correct Prediction

```
# Evaluate the saved model against the test set
test_load_all = DataLoader(test_data, batch_size=10000, shuffle=False)

with torch.no_grad():
    correct = 0
    for X_test, y_test in test_load_all:
        y_val = moe_model(X_test)
        predicted = torch.max(y_val,1)[1]
        correct += (predicted == y_test).sum()

print(f'Test accuracy: {correct.item()}/{len(test_data)} = {correct.item()*100/(len(test_data)):7.3f}%')
```

```
Test accuracy: 1738/3717 =  46.758%
```

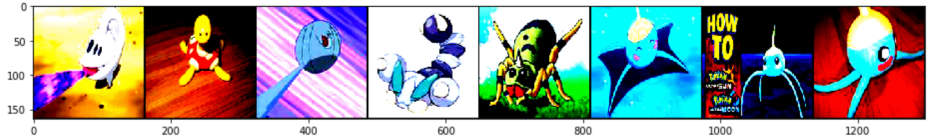
Saved Model Accuracy

RePokémon

EXAMPLES OF MISSES

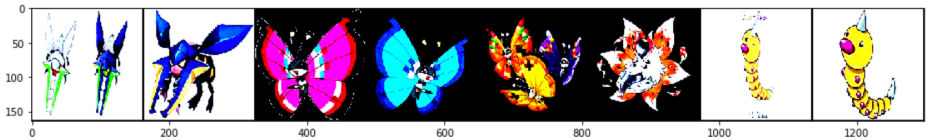
```
Index: [  170   171   172   178   182   183   185   186]
Label: [    0     0     0     0     0     0     0     0]
Class:  bug bug bug bug bug bug bug bug

Guess: [   15     5     2    13     3     9    17     7]
Class:  rock fighting dragon poison electric grass water flying
```



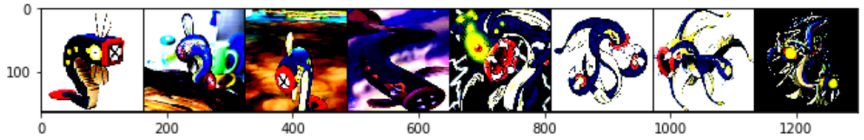
```
Index: [  202   203   204   205   206   208   209   211]
Label: [    0     0     0     0     0     0     0     0]
Class:  bug bug bug bug bug bug bug bug

Guess: [    3     3     7     7     7    16    13    13]
Class:  electric electric flying flying flying steel poison poison
```



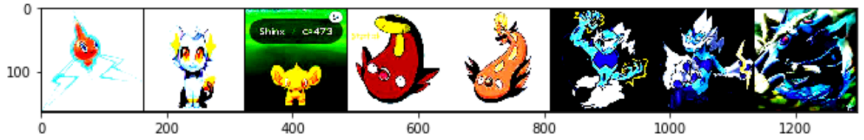
```
Index: [     552      553      554      555      556      557      558      559]
Label: [       3        3        3        3        3        3        3        3]
Class:    electric   electric   electric   electric   electric   electric   electric   electric

Guess: [       7        0       17       12       15       17       16        0]
Class:    flying    bug    water    normal    rock    water    steel    bug
```



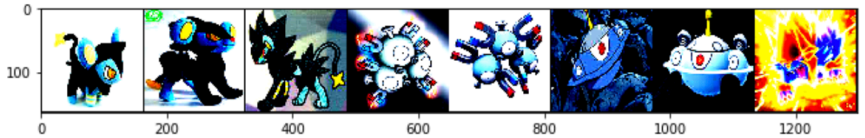
```
Index: [     636      638      640      641      642      643      644      645]
Label: [       3        3        3        3        3        3        3        3]
Class:    electric   electric   electric   electric   electric   electric   electric   electric

Guess: [       8        7        9       10       10        7        7        7]
Class:    ghost    flying    grass    ground    ground    flying    flying    flying
```



```
Index: [     596      597      598      599      600      601      602      604]
Label: [       3        3        3        3        3        3        3        3]
Class:    electric   electric   electric   electric   electric   electric   electric   electric

Guess: [      17        5        5       16       16       16       16        7]
Class:    water    fighting    fighting    steel    steel    steel    steel    flying
```



```
Index: [     617      622      627      629      630      631      634      635]
Label: [       3        3        3        3        3        3        3        3]
Class:    electric   electric   electric   electric   electric   electric   electric   electric

Guess: [       4       12        7        8       10       10       17        8]
Class:    fairy    normal    flying    ghost    ground    ground    water    ghost
```

RePokémon

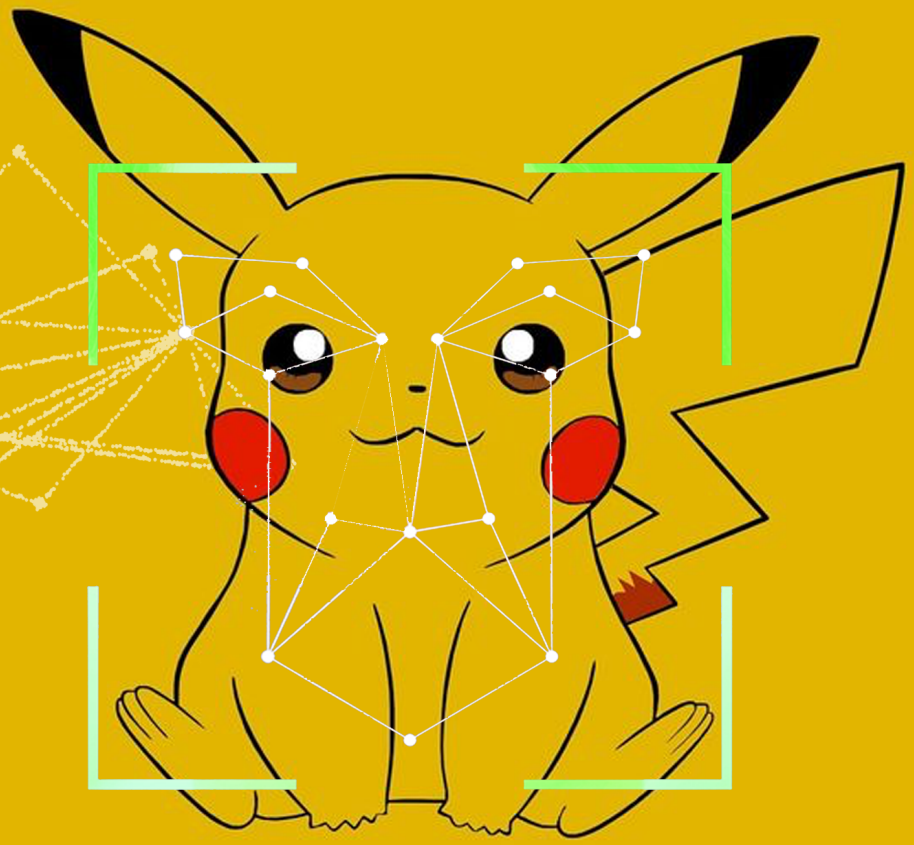

DEEP LEARNING
RePokémon

CLASSIFICATION

VIZUAL RECOGNITION LEARNING

IMAGE AI is the library utilized

- Resnet is an algorithm that the library uses
- Pandas Library
- ResNet was the deep learning algorithm we used
- Pokémon types had to be manually organized by photo, (image file)
- Each type had 300 training images and 50 testing images
- Image AI was the library we used
- Ran 10 epochs
- Exact accuracy of the model was 71.37%



FIVE TYPES OF POKÉMON

- Fire
- Water
- Grass
- Electric
- Ice

IMAGEAI

```
PokeRecognizer.py

PokeModelTrainer.py   PokeRecognizer.py ●

Users ▸ HIERARCHY ▸ ~_GROUP_PROJECT_ASSETS ▸ PROJECT_03 ▸ AE ▸ FinalProjectTest ▸ PokeRecognizer.py ▸ ...

1  from imageai.Prediction.Custom import CustomImagePrediction
2  import os
3
4  execution_path = os.getcwd()
5
6  prediction = CustomImagePrediction()
7  prediction.setModelTypeAsResNet()
8  prediction.setModelPath("C:\\Users\\Admin\\Desktop\\FinalProjectTest\\model_ex-010_acc-0.713693.h5")
9  prediction.setJsonPath("C:\\Users\\Admin\\Desktop\\FinalProjectTest\\model_class.json")
10 prediction.loadModel(num_objects=5)
11
12 predictions, probabilities = prediction.predictImage("C:\\Users\\Admin\\Desktop\\FinalProjectTest\\BulbTest.jpg", result_count=1)
13
14 for eachPrediction, eachProbability in zip(predictions, probabilities):
15     print(eachPrediction , " : " , eachProbability)
16
```

Recognizer

```
PokeModelTrainer.py

PokeModelTrainer.py ×   PokeRecognizer.py

Users ▸ HIERARCHY ▸ ~_GROUP_PROJECT_ASSETS ▸ PROJECT_03 ▸ AE ▸ FinalProjectTest ▸ PokeModelTrainer.py ▸ ...

1  import os
2  from imageai.Prediction.Custom import ModelTraining
3
4  model_trainer = ModelTraining()
5  model_trainer.setModelTypeAsResNet()
6  model_trainer.setDataDirectory('C:\\Users\\Admin\\Desktop\\FinalProject\\pokemon')
7  model_trainer.trainModel(num_objects=5, num_experiments=10, enhance_data=True, batch_size=32, show_network_summary=True)
8

3.7.1 64-bit ('base': conda)   ⊗ 2 ⚠ 0   Ln 8, Col 1   Spaces: 4   UTF-8   CRLF   P
```

Trainer

RePokémon

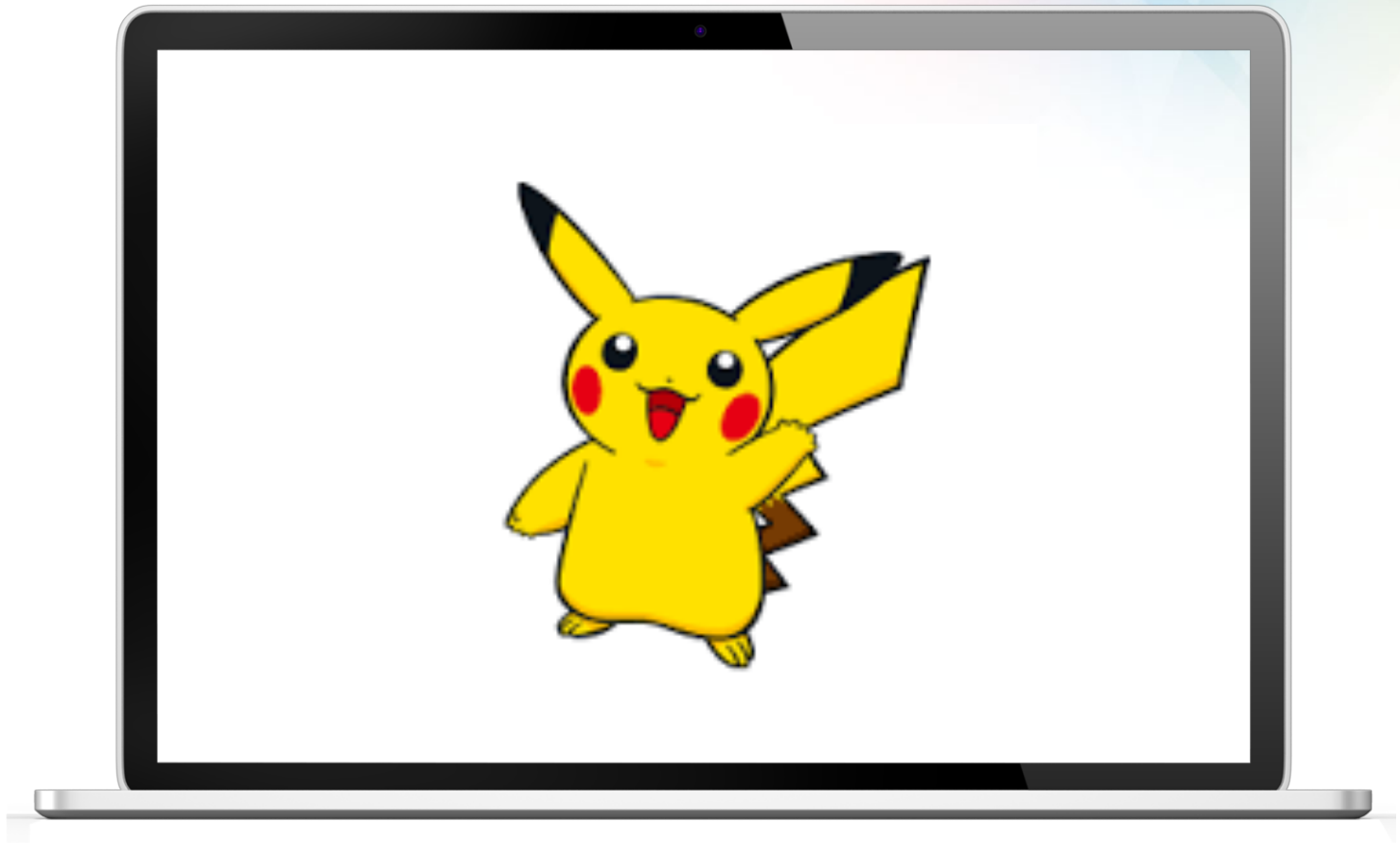
OUTPUT AFTER TEST OF IMAGE

Pikachu Image results:

Electric : 88.83466124534607

Fire : 5.044957250356674

Water : 2.711331844329834



RePokémon

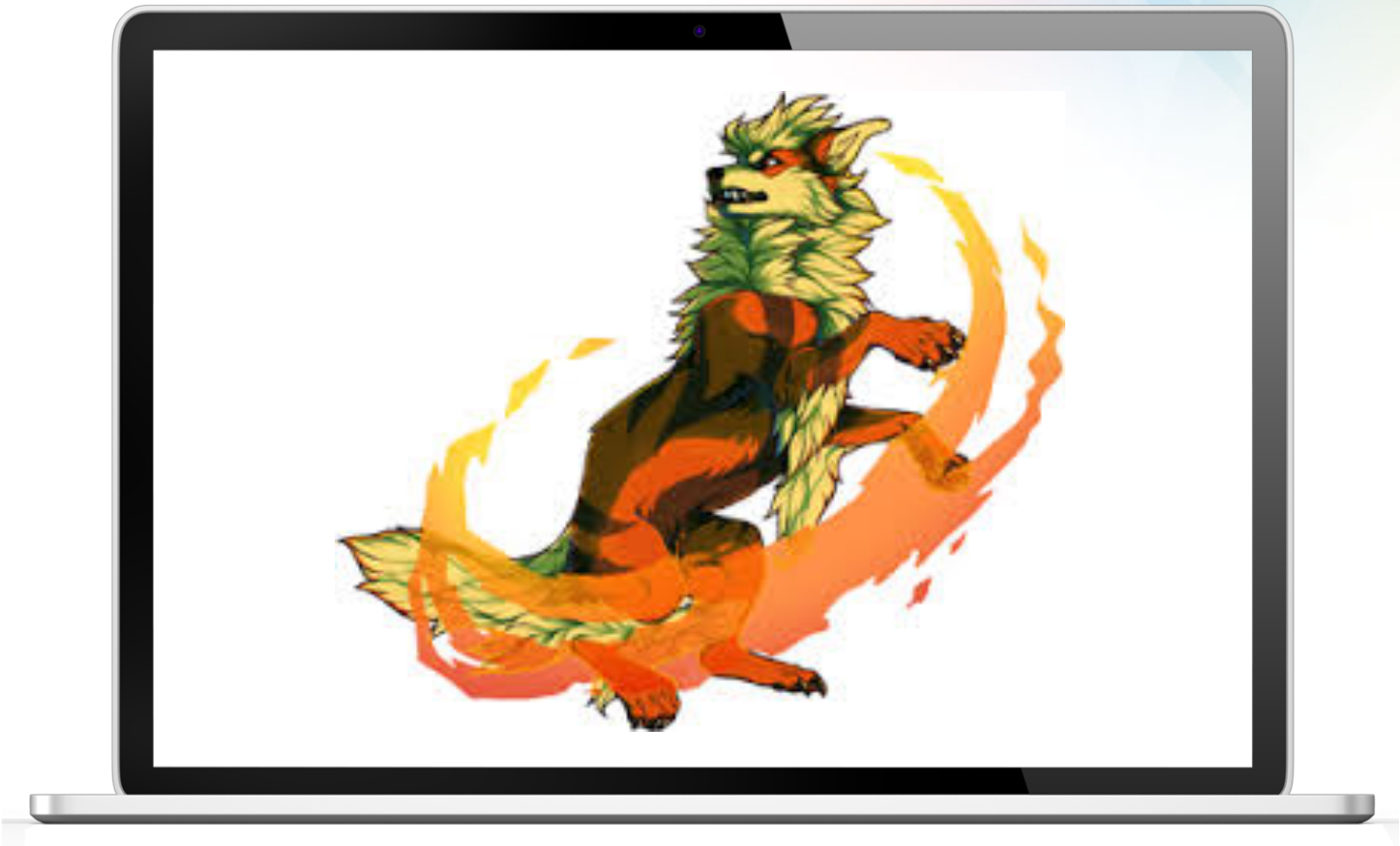
OUTPUT AFTER TEST OF IMAGE

Arcanine test results:

Fire : 65.90073704719543

Electric : 12.308239936828613

Ice : 10.716183483600616



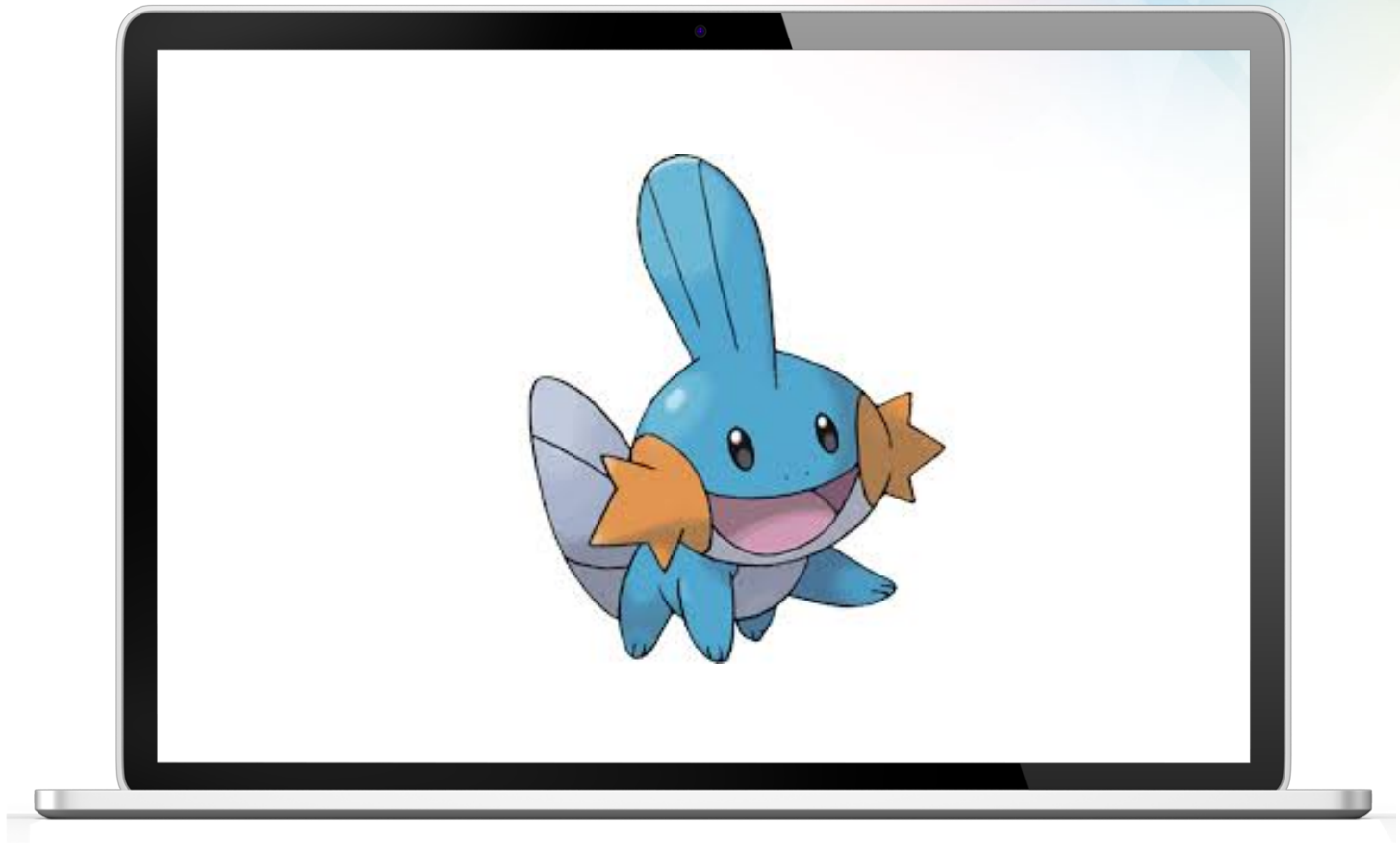
OUTPUT AFTER TEST OF IMAGE

Mudkip test results:

Water : 64.42905068397522

Ice : 25.727957487106323

Electric : 5.9982482343912125



OUTPUT AFTER TEST OF IMAGE

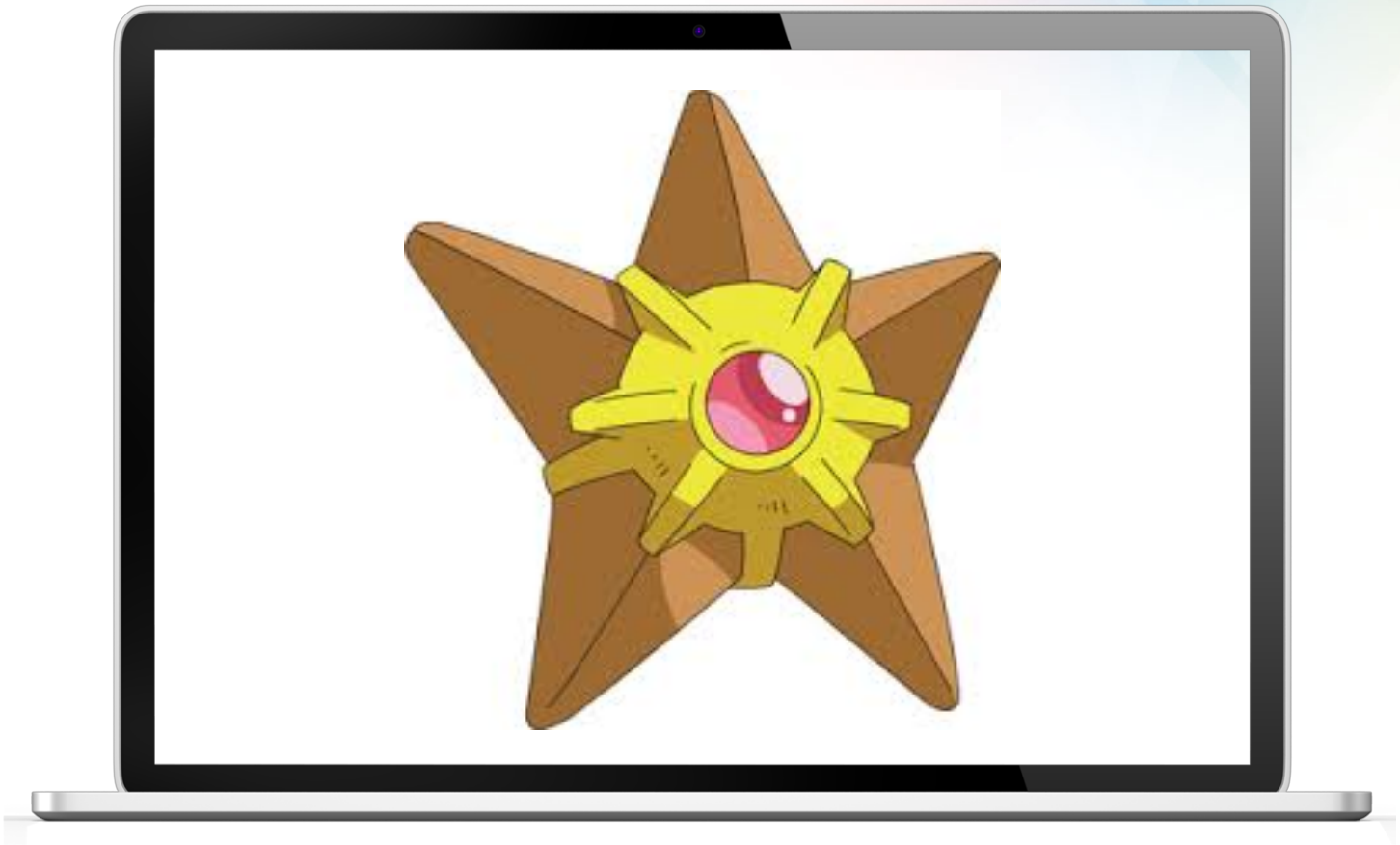
Staryu Test:

Electric : 81.20018243789673

Fire : 10.354503989219666

Ice : 3.5705827176570892

This is incorrect, staryu is water




WITH MORE TIME

WITH MORE TIME

FINANCIAL IMPACT

- Pokémon decline to provide financial data upon written request

COMPARISON OF ATTRIBUTES

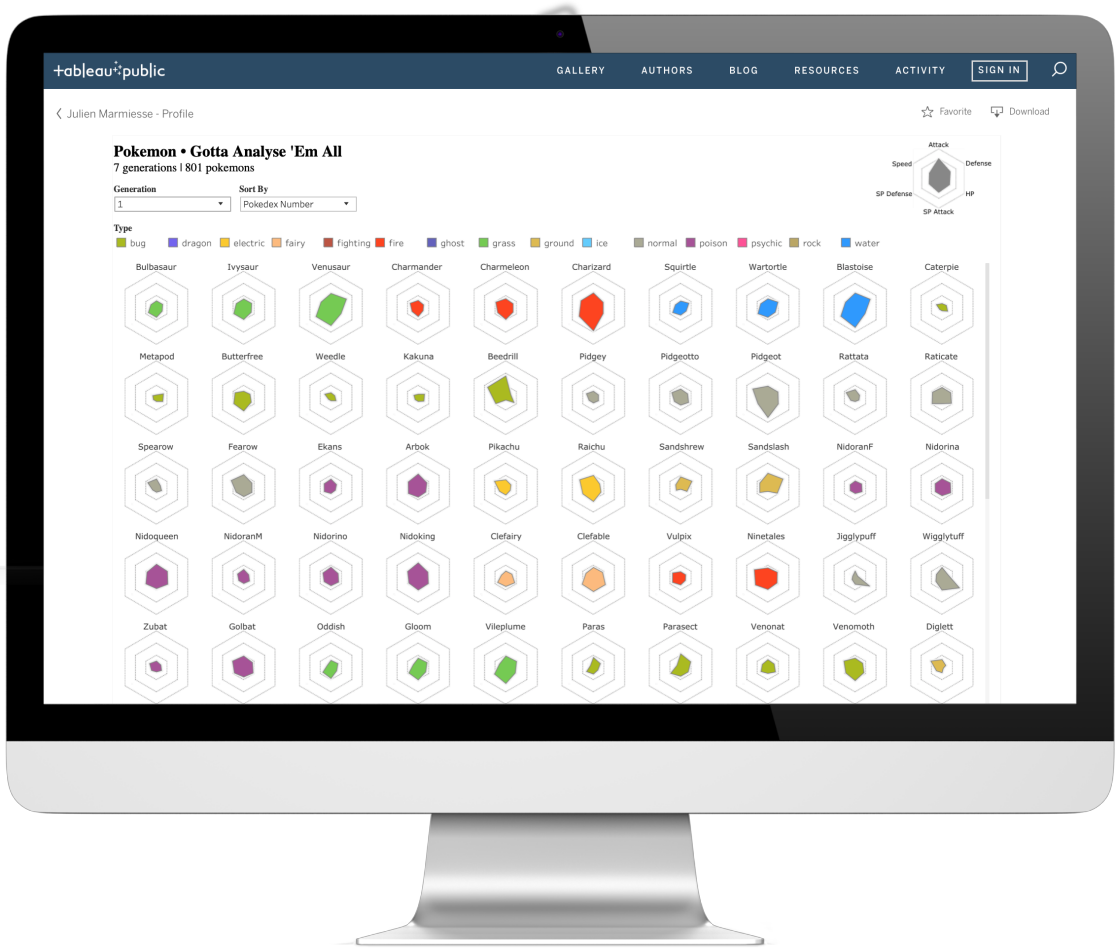
- Would have gone farther into the visualization of character attributes, (see right for example)

MACHINE LEARNING

- Play around with the features more
- Adding features with more datasets

DEEP LEARNING

- Would have spent more time training more images
- Would have gone farther down the path of testing theory surrounding color recognition as primary



Julien Marmiesse – Profile: http://digg.com/2018/pokemon-stats-data-viz

RePokémon

POKÉMON GO SLACK EMOJI PACK

https://github.com/israelvicars/pkmn-go-emoji