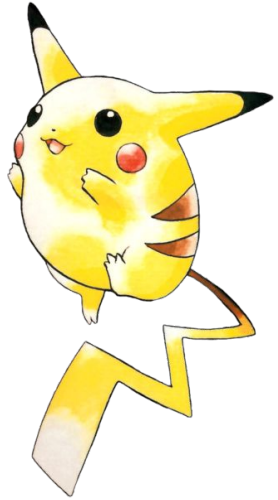
Pokémon



By: Briant Anaya , Ramon Delgadillo, Raymond Lo, Kenny Vu

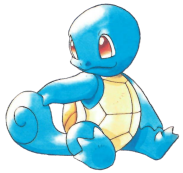
What's The Problem?

- We are using a dataset that consists over hundreds of Pokemon from the older generation along with newer generation pokemon! The generations go up to Scarlet and Violet. Based on the stats given, can we determine if a pokemon has evolved and if so, what stage evolution?

- This is a multi-classification problem. There are three classes to consider, which are stage evolution 1, 2, and 3.

- We employ three models, which are Logistic Regression, Random Forest, and Neural Networks.

Features



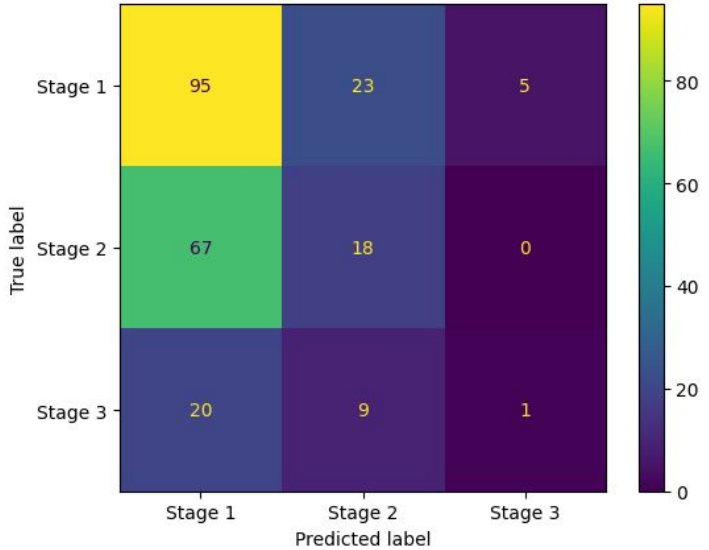
F	G	H	I	J	K	L	M
Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Evolution Stage
318	45	49	49	65	65	45	1
405	60	62	63	80	80	60	2
525	80	82	83	100	100	80	3
625	80	100	123	122	120	80	3
309	39	52	43	60	50	65	1
405	58	64	58	80	65	80	2
534	78	84	78	109	85	100	3
634	78	130	111	130	85	100	3
634	78	104	78	159	115	100	3
314	44	48	65	50	64	43	1
405	59	63	80	65	80	58	2
530	79	83	100	85	105	78	3
630	79	103	120	135	115	78	3
195	45	30	35	20	20	45	1

Logistic Regression

- We create, fit, and predict using our Logistic Reg. model aftering splitting data.
- Results show an accuracy of .47-.48.
- Stage evolution 1 had the least confusion while the rest got confused often.

```
Accuracy: 0.4789915966386555
              precision    recall  f1-score   support

           1       0.52      0.77      0.62       123
           2       0.36      0.21      0.27        85
           3       0.17      0.03      0.06        30

    accuracy                           0.48       238
   macro avg       0.35      0.34      0.32       238
weighted avg       0.42      0.48      0.42       238
```

```python
# Create a Logistic Regression model
logistic_regression_model = LogisticRegression(multi_class = 'ovr', max_iter = 1000)

# Fit the model to the training data
logistic_regression_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = logistic_regression_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print(classification_report(y_test, y_pred))
```



Data from PokemonDB*

Stats Example

Base stats

HP	20
Attack	10
Defense	55
Sp. Atk	15
Sp. Def	20
Speed	80
Total	**200**

Base stats

HP	50
Attack	20
Defense	55
Sp. Atk	25
Sp. Def	25
Speed	30
Total	**205**

Base stats

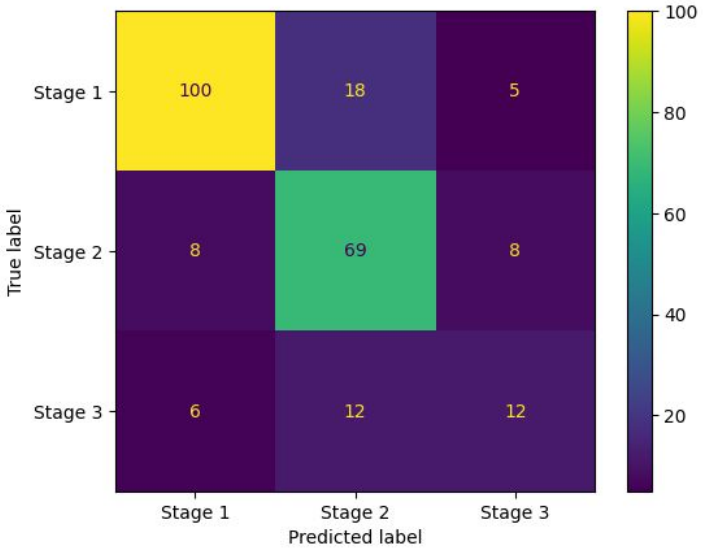
HP	106
Attack	110
Defense	90
Sp. Atk	154
Sp. Def	90
Speed	130
Total	**680**

Random Forest

- We repeat the same steps as before but using RF instead.
- Results proved promising with a generous .76 or 76% accuracy.
- The confusion matrix shows that stage evolution 1 is the least confusing class, while stage evolution 2 confusion improved drastically compared to other models.

```
Accuracy: 0.7605042016806722
              precision    recall  f1-score   support

           1       0.88      0.81      0.84       123
           2       0.70      0.81      0.75        85
           3       0.48      0.40      0.44        30

    accuracy                           0.76       238
   macro avg       0.68      0.67      0.68       238
weighted avg       0.76      0.76      0.76       238
```

```python
random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train, y_train)

y_pred = random_forest_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print(classification_report(y_test, y_pred))
```



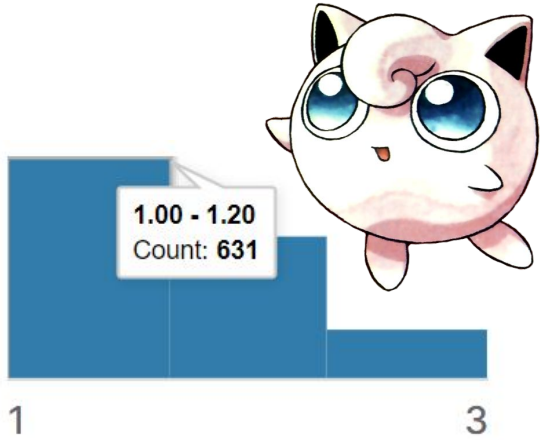
Neural Network

- After creating, fitting, and predicting on our NN model, we get an accuracy of .46 or 46%.
- The confusion matrix shows stage evolution 1 and 2 are moderately being confused for other classes, while stage evolution 3 is the most confused.

```
# Assuming y_train is an array of string labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
# Ensure the encoded labels are integers
y_train_encoded = y_train_encoded.astype(int)

model = Sequential([
    Dense(units = 25, activation="relu"),
    Dense(units = 15, activation="relu"),
    Dense(units = 3, activation="softmax")              ])
model.compile(
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),
    loss = SparseCategoricalCrossentropy()
          )
model.fit(X_train, y_train, epochs = 10)
```

```
30/30 [==============================] - 0s 1ms/step
[[311 179  63]
 [148 123  48]
 [ 49  26   5]]
Accuracy on test data set:  0.46113445378151263
```

Oversampling and undersampling



- We observed that there were more stage evo. 1 $ 2 than the other stages. As an attempt to improve results, we tried to undersample first. Results came back worse with an accuracy of 32% for Logistic Regression . We surmise reducing data on this particular dataset does not help, as expected.

- We tried oversampling, which duplicates data that are stage 3 to match the amount of other stages and the result came out to be better across the models.

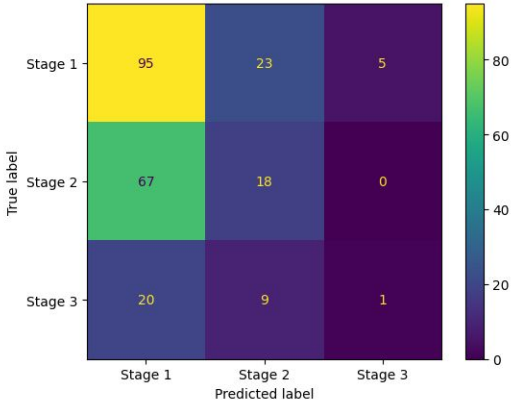
```
2/2 [==============================] - 0s 3ms/step
[[ 1  2  5]
 [ 8  8 14]
 [ 8  3 10]]
Accuracy on test data set:  0.3220338983050847
```

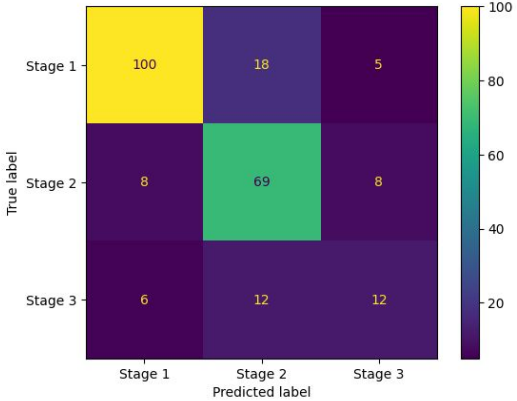
```
12/12 [==============================] - 0s 1ms/step
[[ 83  36  42]
 [  2   2   0]
 [ 49  42 109]]
Accuracy on test data set:  0.5315068493150685
```

Results explained

- Our Random Forest model performed the best with an accuracy of 76%. Accuracy improved further from oversampling and grid search CV, which came out to be 89%.
- We noticed that stage 1 and stage 2 evolution were more accurately predicted compared to the other models.
- Both our Logistic Regression and Neural Network had similar results, however, logistic regression improved with hyper parameter tuning. The confusion matrix shows that only stage evolution 1 was accurately predicted.

```
30/30 [==============================] - 0s 1ms/step
[[311 179  63]
 [148 123  48]
 [ 49  26   5]]
Accuracy on test data set:  0.46113445378151263
```





Results explained: Continued

- There a few explanations for our results. Most likely, the features or stats of a pokemon do not have a strong correlation as to their evolution stage. There exists pokemon that are in their stage evolution 3 but are still weaker or have lower stats than other pokemons that may have not evolved yet or have not reached stage evolution 3.

- Overall, lower or higher stats do not necessarily tell you the evolution stage of a pokemon. That is why feature engineering is not viable as well, since combining features will not result in anything useful.