Predict the Winner Pokemon

Team 5 : Tech Geeks

Harshini Gadige (015351232)

Andrew Kehl (010586069)

Aneshaa Kasula (014558427)

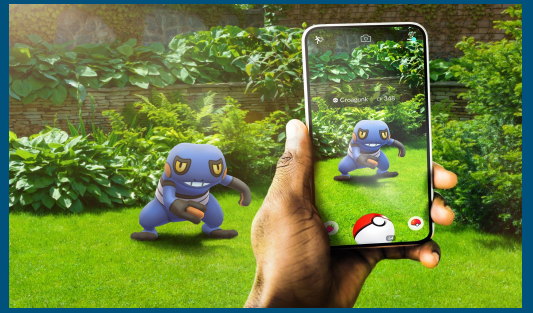
Viritha Vanama (015356991)

Introduction

Pokémon - Meaning Pocket Monsters in Japanese

- Lunatic anime television series in 90’s
- Created by Satoshi Tajiri in 1996
- Highest-grossing media franchise of all time
- Over 20 seasons and 1000 episodes
- In *Pokémon*, humans, known as Pokémon Trainers, catch and train Pokémon to battle other Pokémon for sport.

Motivation and Objective



- Pokémon GO - Augmented Reality mobile game in 2016. Blockbuster successful mobile application developed by Niantic Labs, a former division of Google.
- Most used and profitable mobile apps
- Best selling media franchise of all time
- Winning the battles in the game is more fun and can increase the interest among players which ultimately results in profits for the company
- Machine learning has the potential to predict the winner Pokémon in a battle.

Survey

Literature Survey

- “Predicting the winning side of DotA2”

  -Kuangyan Song, Tianyi Zhang, Chao Ma

- “Predicting the outcome of NFL games using machine learning”

  - Babak Hamadani

Technical Survey

Models selection through k-fold cross validation - Random Forest, SVM, Logistic regression, Decision Tree, AdaBoost

Hyper parameter tuning using RandomSearchCV

Evaluation metrics : Accuracy scores, Classification report, Confusion matrix, ROC and AUC curves

Model Complexity : Cross validation error Vs Training error

Technical Requirements

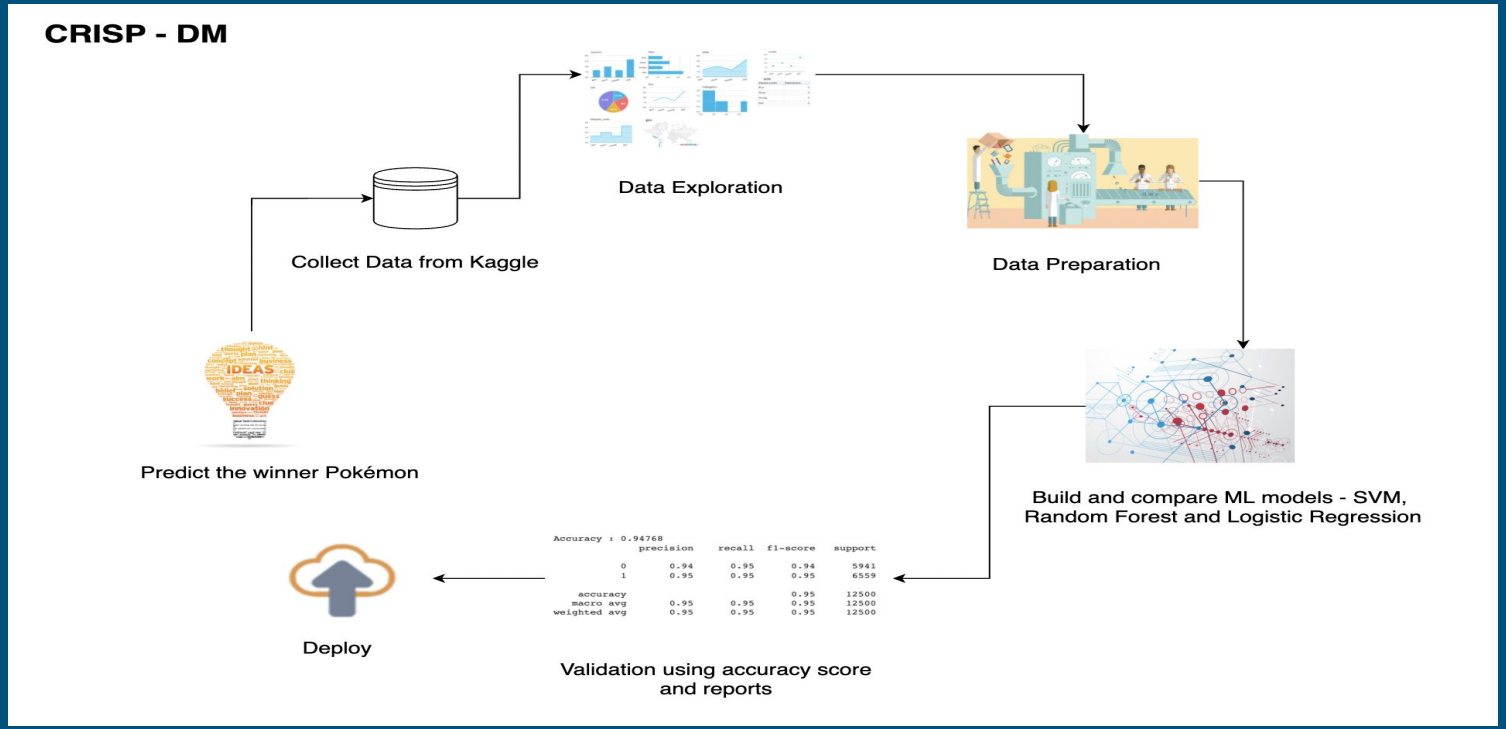
Tools used

- Environment - Google Colab
- Language : Python3
- Libraries : Pandas, Seaborn, Matplotlib, Numpy, Scipy, Scikit learn

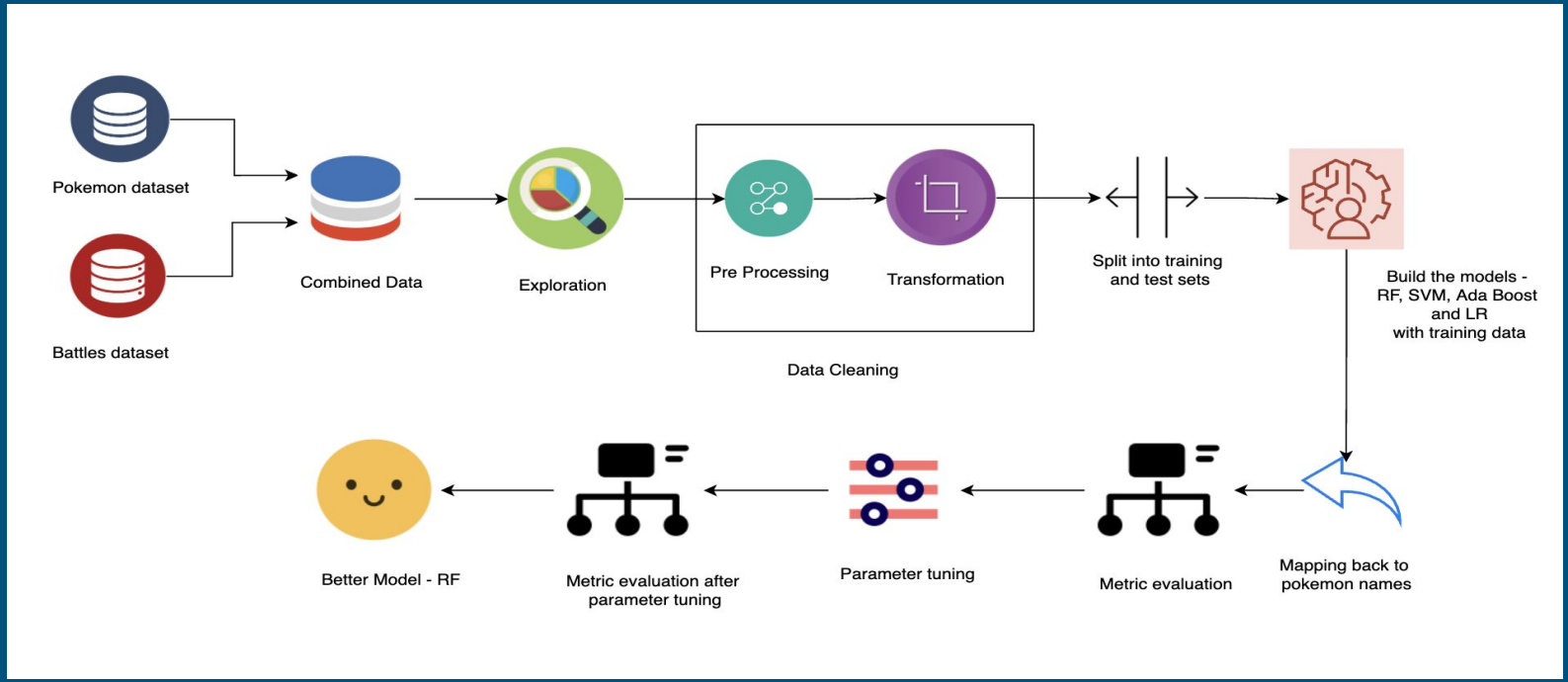
Audience

- General public interested in playing the game
- Company Management

Project Plan - Architecture Diagram



Flow Diagram



Dataset

Pokemon.csv

pokemon.csv (40.45 kB)

Detail | Compact | Column | 12 of 12 columns

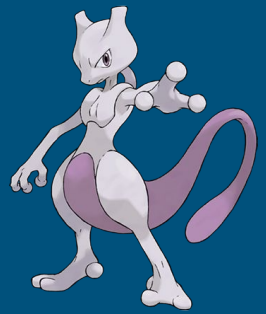
# 1	Bulbasaur	Grass	Poison	# 45	# 49
	Name of pokemon	1st attack type	2nd attack type	Hitpoints	Attack force
1 – 800	800 unique values	Water 14%, Normal 12%, Other (590) 74%	[null] 48%, Flying 12%, Other (317) 40%	1 – 255	5 – 190
1	Bulbasaur	Grass	Poison	45	49
2	Ivysaur	Grass	Poison	60	62
3	Venusaur	Grass	Poison	80	82
4	Mega Venusaur	Grass	Poison	80	100
5	Charmander	Fire		39	52
6	Charmeleon	Fire		58	64
7	Charizard	Fire	Flying	78	84
8	Mega Charizard X	Fire	Dragon	78	130
9	Mega Charizard Y	Fire	Flying	78	104
10	Squirtle	Water		44	48

Features

‘#’ - Denoting ID of pokemon

Name, Type 1, Type 2, HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, Generation, Legendary

Dataset Cont...



combats.csv

combats.csv (580.59 kB)

Detail | Compact | Column

# First_pokemon	# Second_pokemon	# Winner
1 – 800	1 – 800	1 – 800
266	298	298
702	701	701
191	668	668
237	683	683
151	231	151
657	752	657
192	134	134
73	545	545
220	763	763
302	31	31

Features

First_pokemon, Second_pokemon - Denoting Battle pair of pokemons

Target label : Winner

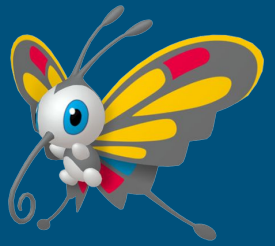
Data Exploration & Preparation

Data Exploration

Data Quality Report: Continuous Features Report

FEATURE NAME	Count	Miss %	Card.	Min	1st Qrt.	Mean	Median	3rd Qrt	Max	Std. Dev.
HP	800	0	94	1	50	69.26	65	80	255	25.53
Attack	800	0	111	5	55	79	75	100	190	32.46
Defense	800	0	103	5	50	73.84	70	90	230	31.18
Sp. Atk	800	0	105	10	49.75	72.82	65	95	194	32.72
Sp. Def	800	0	92	20	50	71.9	70	90	230	27.83
Speed	800	0	108	5	45	68.28	65	90	180	29.06

Data Exploration Continued

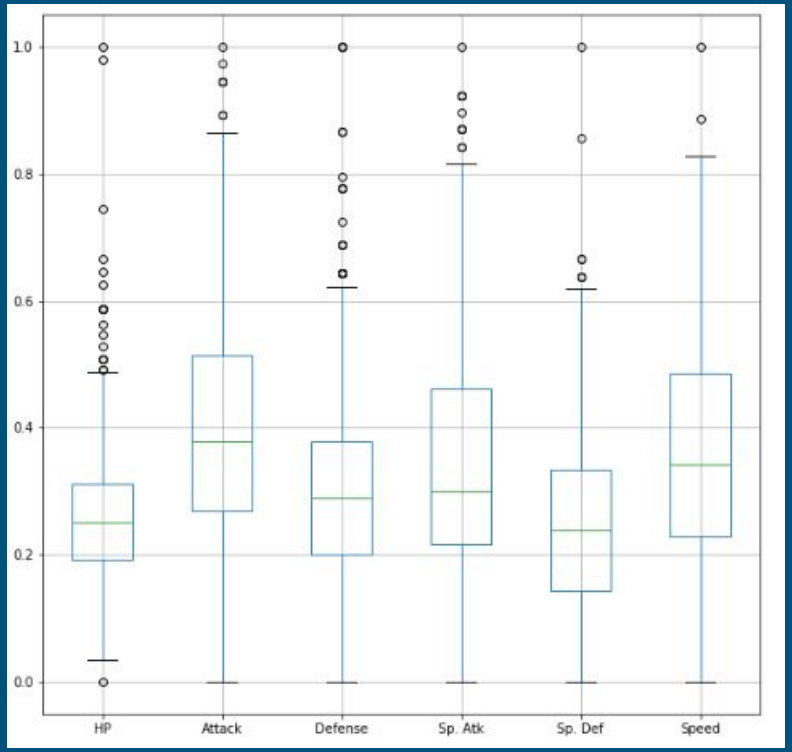


Data Quality Report: Categorical Features

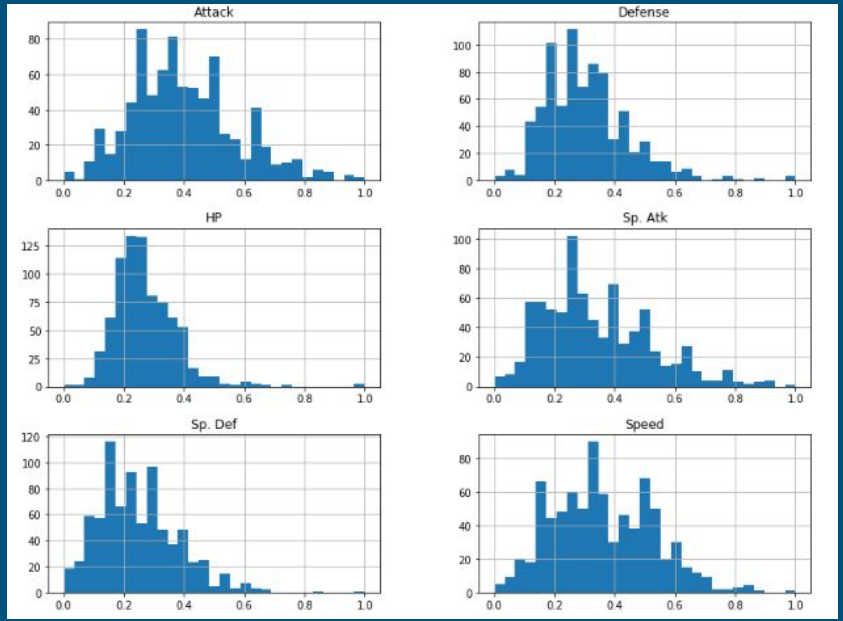
FEATURE NAME	Count	Miss %	Card.	Mode	Mode Freq	Mode %	2nd Mode	2nd Mode Freq	2nd Mode %
#	800	0	800	800	1	0.12	263	1	0.12
Name	799	0.01	799	Sawk	1	0.13	Cobalion	1	0.13
Type 1	800	0	18	Water	112	14	Normal	98	12.25
Type 2	414	3.86	18	Flying	97	23.43	Ground	35	8.45
Generation	800	0	6	1	166	20.75	5	165	20.62
Legendary	800	0	2	FALSE	735	91.88	TRUE	65	8.12

Data Exploration Continued

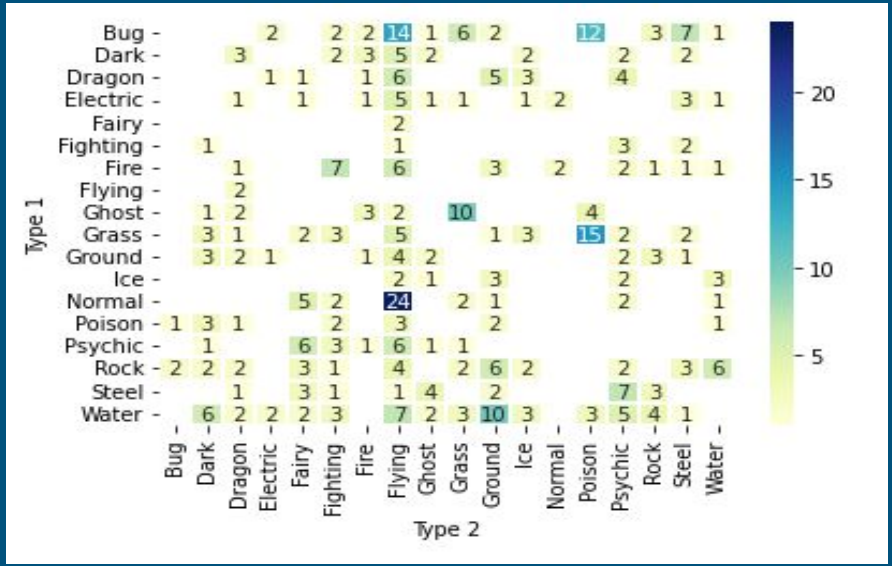
Box Plot

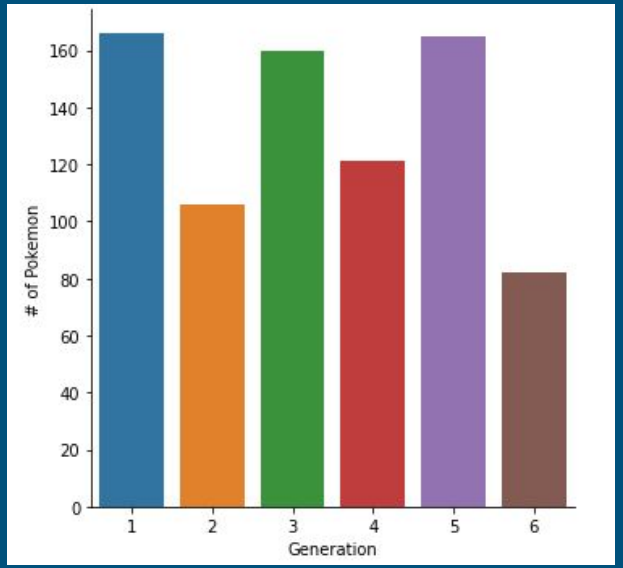


Histograms



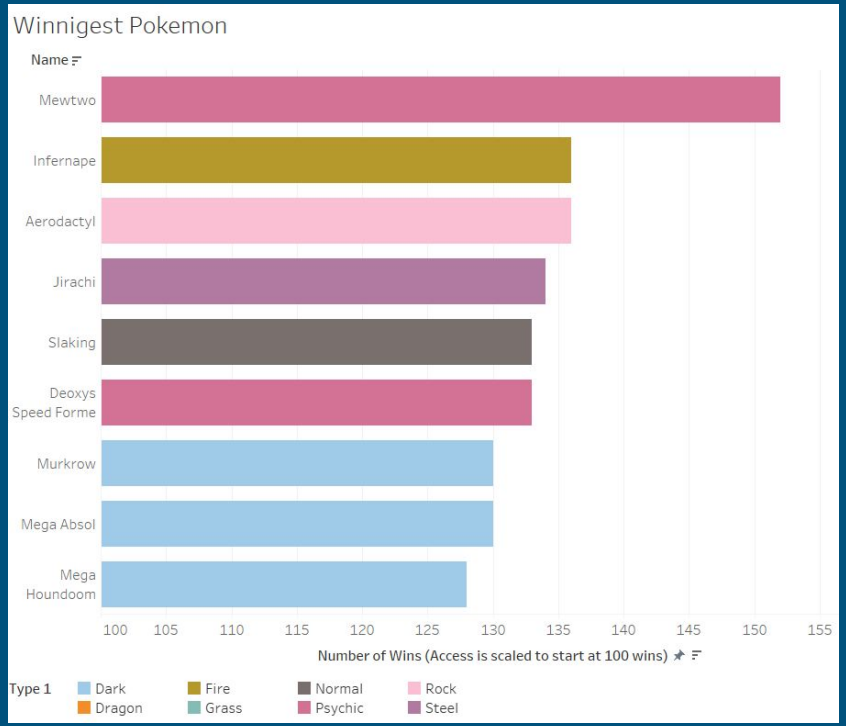
Data Exploration Continued





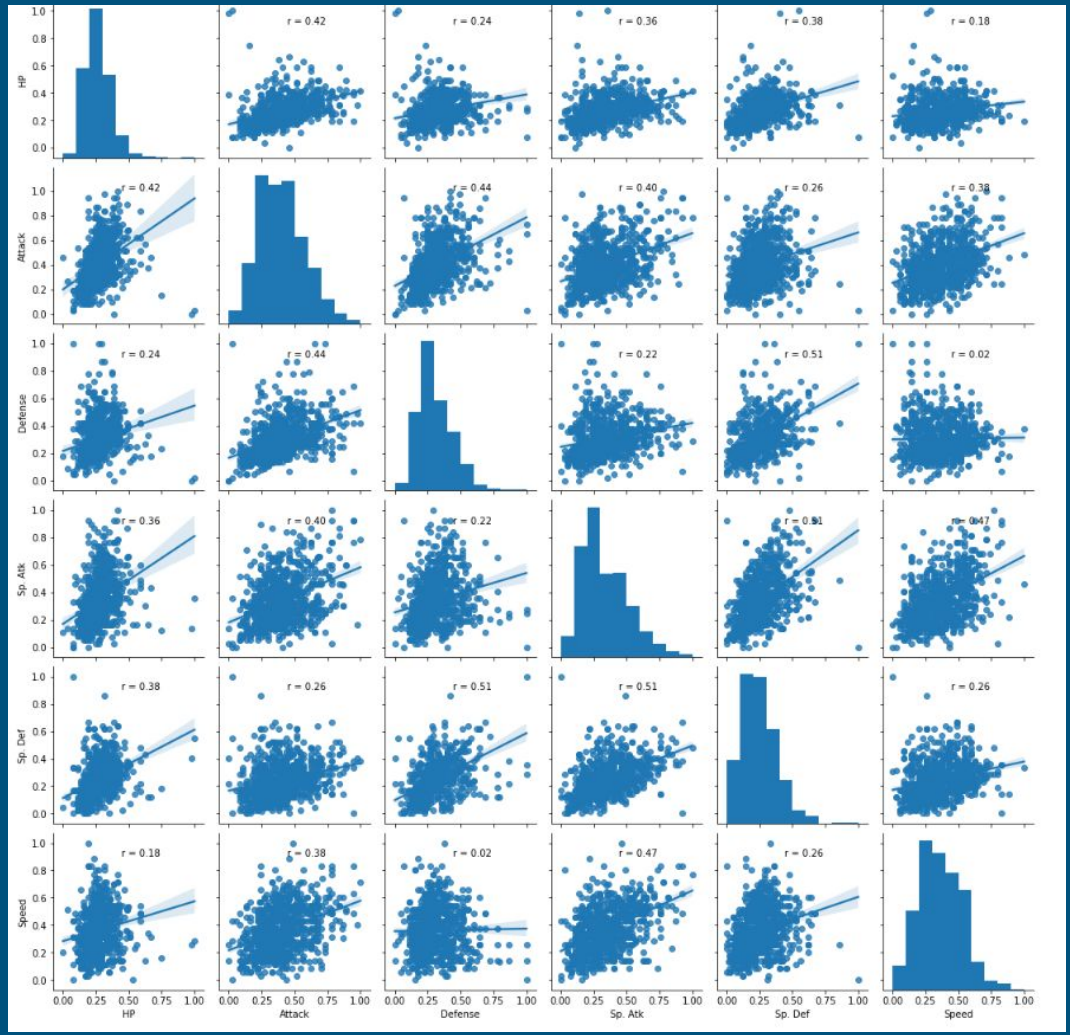
As per the heatmap, “Normal” as Type 1 and “Flying” as Type 2 seems to be the popular combination.

Data Exploration Continued



Name	Type 1	Type 2	Generation	Legendary	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	strength
Mewtwo	Psychic	None	1	True	0.413	0.568	0.378	0.783	0.333	0.714	3.189
Slaking	Normal	None	3	False	0.587	0.838	0.422	0.462	0.214	0.543	3.066
Mega Aerodactyl	Rock	Flying	1	False	0.311	0.703	0.356	0.326	0.357	0.829	2.881
Deoxys Speed Forme	Psychic	None	3	True	0.193	0.486	0.378	0.462	0.333	1.000	2.852
Mega Houndoom	Dark	Fire	2	False	0.291	0.459	0.378	0.707	0.333	0.629	2.797
Jirachi	Steel	Psychic	3	True	0.390	0.514	0.422	0.489	0.381	0.543	2.738
Infernape	Fire	Fighting	4	False	0.295	0.535	0.293	0.511	0.243	0.589	2.466
Aerodactyl	Rock	Flying	1	False	0.311	0.541	0.267	0.272	0.262	0.714	2.366
Murkrow	Dark	Flying	2	False	0.232	0.432	0.164	0.408	0.105	0.491	1.833

Correlation Matrix



Data Preparation

Feature	Data Quality Issue	Potential Handling Strategies
Name	1 pokemon's name is missing	Get valid name from different dataset
Type 2	Has 386 null values	Null values are valid; replace null with "None"

Data Preparation Continued

Transformation

- Legendary column containing True or False is changed to 0 (False) and 1 (True)
- FeatureHasher is used to convert Type1 and Type2 columns to numericals.
- Mapped combats with pokemons dataset

Split training and testing data into

75% and 25% respectively

	#	Name	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	0	1	2	3	4	0	1	2	3	4
0	1	Bulbasaur	45	49	49	65	65	45	1	0	2.0	0.0	0.0	0.0	-1.0	0.0	-2.0	0.0	2.0	-2.0
1	2	Ivysaur	60	62	63	80	80	60	1	0	2.0	0.0	0.0	0.0	-1.0	0.0	-2.0	0.0	2.0	-2.0
2	3	Venusaur	80	82	83	100	100	80	1	0	2.0	0.0	0.0	0.0	-1.0	0.0	-2.0	0.0	2.0	-2.0
3	4	Mega Venusaur	80	100	123	122	120	80	1	0	2.0	0.0	0.0	0.0	-1.0	0.0	-2.0	0.0	2.0	-2.0
4	5	Charmander	39	52	43	60	50	65	1	0	1.0	-1.0	0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	796	Diancie	50	100	150	100	150	50	6	1	0.0	-1.0	-1.0	1.0	1.0	2.0	-1.0	0.0	-1.0	1.0
796	797	Mega Diancie	50	160	110	160	110	110	6	1	0.0	-1.0	-1.0	1.0	1.0	2.0	-1.0	0.0	-1.0	1.0
797	798	Hoopa Confined	80	110	60	150	130	70	6	1	-1.0	-2.0	-2.0	0.0	0.0	-1.0	0.0	0.0	1.0	-1.0
798	799	Hoopa Unbound	80	160	60	170	130	80	6	1	-1.0	-2.0	-2.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0
799	800	Volcanion	80	110	120	130	90	70	6	1	1.0	-1.0	0.0	-1.0	1.0	2.0	0.0	0.0	0.0	-1.0

800 rows x 20 columns

Feature Hasher

Used sklearn FeatureHasher

Converts strings into scipy.sparse matrices using the hash function - signed 32-bit version of Murmurhash3

n_features parameter represents the number of columns in the output matrix and we have set the parameter value to 4

```
print(pokemon['Type 1'].nunique())
print(pokemon['Type 1'].unique())
print(pokemon['Type 2'].nunique())
print(pokemon['Type 2'].unique())
```

```
18
['Grass' 'Fire' 'Water' 'Bug' 'Normal' 'Poison' 'Electric' 'Ground'
 'Fairy' 'Fighting' 'Psychic' 'Rock' 'Ghost' 'Ice' 'Dragon' 'Dark' 'Steel'
 'Flying']
```

```
f = d4.toarray()

unq = np.unique(f, axis=0)
print("nFeature Shape:", f.shape)
print("nUnique Shape:",unq.shape)
```

```
nFeature Shape: (800, 4)
nUnique Shape: (18, 4)
```

Modeling

K-fold Cross Validation for Model selection

The goal here is to ensure that we calculate the best estimate of how a prediction model will perform when actually being deployed.

Models:

```
('LRG', LogisticRegression(solver='liblinear', max_iter=250)))
('KNB', KNeighborsClassifier()))
('GNB', GaussianNB()))
('RFC', RandomForestClassifier(random_state=42, n_estimators=100)))
('DTC', DecisionTreeClassifier(random_state=42, criterion='entropy', max_depth=5)))
('SVC', LinearSVC(random_state=0,dual=False)))
('ADA', AdaBoostClassifier(random_state=0)))
```

Used most common variant in practise : 10-fold CV

Random Forest has highest mean ROC score

```
[('0.982121', 'RFC'),
 ('0.954552', 'DTC'),
 ('0.933969', 'KNB'),
 ('0.927415', 'LRG'),
 ('0.926798', 'SVC'),
 ('0.917204', 'ADA'),
 ('0.842869', 'GNB')]
```

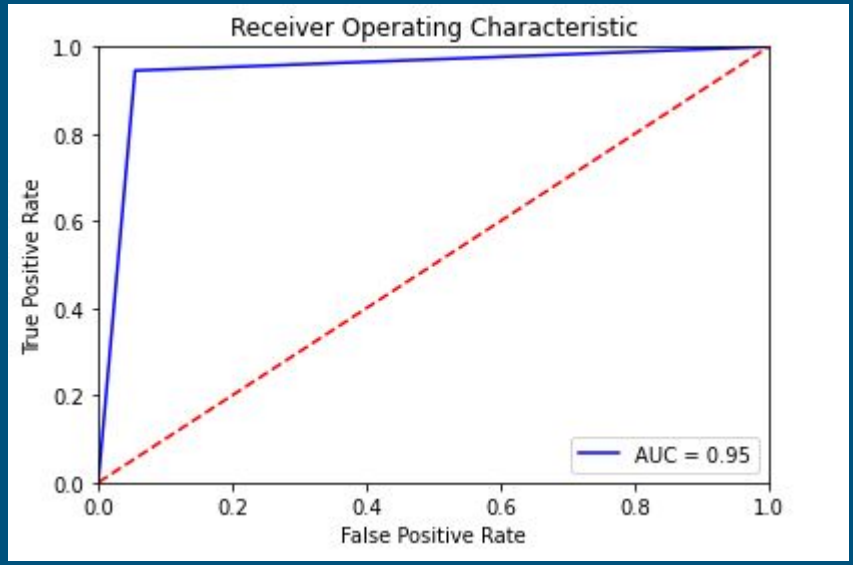
Random Forest

Used RandomForestClassifier of scikit learn library with n_estimators = 100

Accuracy : 0.94768

	precision	recall	f1-score	support
0	0.94	0.95	0.94	5941
1	0.95	0.95	0.95	6559
accuracy			0.95	12500
macro avg	0.95	0.95	0.95	12500
weighted avg	0.95	0.95	0.95	12500

Applied AdaBoost ensemble technique on Decision Tree which gave accuracy rate of 86% and is less than original random forest (94%)



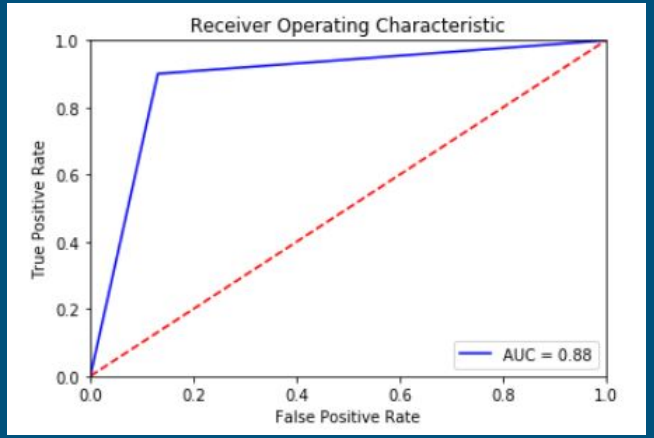
Support Vector Machine (SVM)

Used svm of scikit learn library

SVM's classification by hyperplane is efficient for classification as it can handle high dimensionality features

Data is standardized and is passed to the model which will increase the accuracy

Kernel type : rbf (default)



Accuracy : 0.90608

	precision	recall	f1-score	support
0	0.91	0.90	0.90	5941
1	0.91	0.92	0.91	6559
accuracy			0.91	12500
macro avg	0.91	0.91	0.91	12500
weighted avg	0.91	0.91	0.91	12500

Logistic Regression

Scikit-Learn Logistic Regression Model

Max Iterations = 250

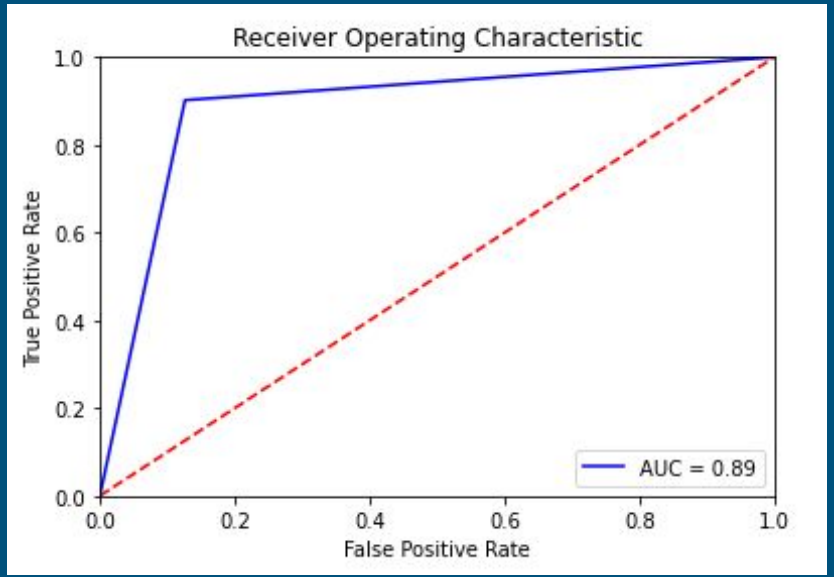
Lib Linear

L2 Penalty

Non-Normalized

Accuracy : 0.88976

	precision	recall	f1-score	support
0	0.89	0.87	0.88	5941
1	0.89	0.90	0.90	6559
accuracy			0.89	12500
macro avg	0.89	0.89	0.89	12500
weighted avg	0.89	0.89	0.89	12500



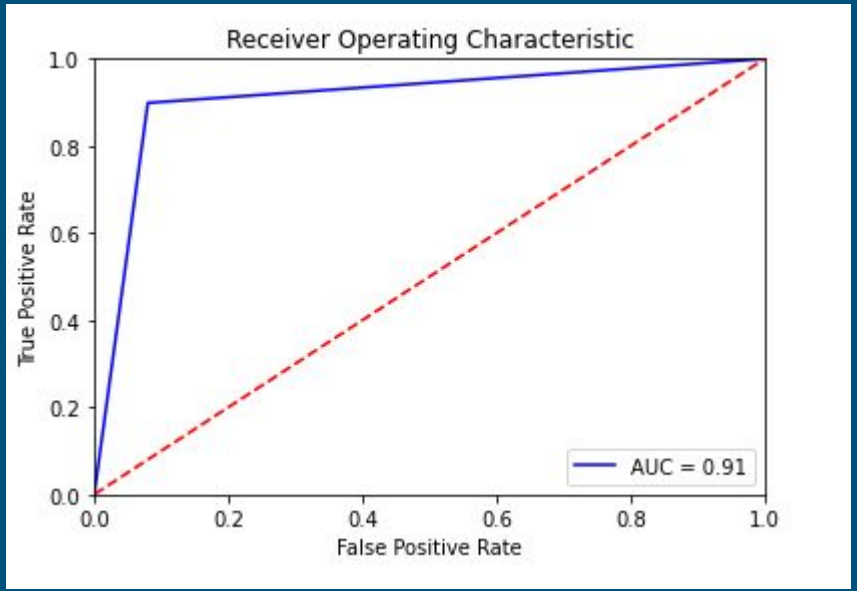
Decision Tree

Supervised ML model where we continuously split the data based on a parameter which have high information gain

Used scikit learn DecisionTreeClassifier module with criterion as ‘entropy’ and max depth of tree = 5

Accuracy : 0.90888

	precision	recall	f1-score	support
0	0.89	0.92	0.91	5941
1	0.93	0.90	0.91	6559
accuracy			0.91	12500
macro avg	0.91	0.91	0.91	12500
weighted avg	0.91	0.91	0.91	12500



Parameter Tuning and Results

Parameter tuning

Used RandomSearchCV to pull the best fit values for parameters

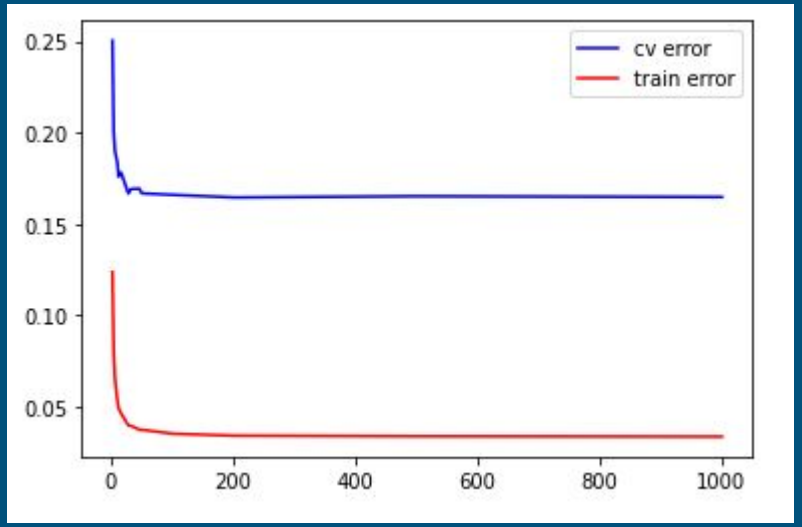
- Random Forest - n_estimators = 400. Percentage of accuracy increased = 0.5
- SVM - Kernel = linear. Percentage of accuracy increased = 0.2
- Ada Boost - n_estimators = 1000. Percentage of accuracy increase = 0.4

Finally we chose RandomForest as it is giving high accuracy with N_estimators = 400

```
Random Forest Classifier
Accuracy:0.951
{'n_estimators': 400,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': None,
 'bootstrap': False}
```

Model complexity Vs Accuracy

Random Forest with parameter : n_estimators



Final results after parameter tuning

Random Forest is the best model that is chosen after parameter tuning by setting n_estimators = 400 which gave accuracy rate of 95.1%

Binary output is mapped back to *Pokémon* names

```
Expected : [1 0 0 ... 1 1 1]
Predicted : [1 0 0 ... 1 1 1]
```

```
Enter First Pokemon: Omastar
Enter Second Pokemon: Shuckle
```

```
Predicted winner : Omastar
```

Total count of mismatch values for Expected Vs Predicted : 645

LIVE DEMO!

References

- https://towardsdatascience.com/increasing-model-reliability-model-selection-cross-validation-1ce0bf506cd
- https://stats.stackexchange.com/questions/52274/how-to-choose-a-predictive-model-after-k-fold-cross-validation
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.FeatureHasher.html
- https://scikit-learn.org/stable/modules/cross_validation.html
- https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://en.wikipedia.org/wiki/Freedman%E2%80%93Diaconis_rule
- https://www.kaggle.com/abcsds/Pokémon
- https://www.kaggle.com/ajisamudra/winner-in-Pokémon-combat-prediction/data?select=combats.csv
- https://en.wikipedia.org/wiki/Pok%C3%A9mon
- https://en.wikipedia.org/wiki/Pok%C3%A9mon_(video_game_series)
- https://www.kaggle.com/terminus7/Pokémon-challenge
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
- https://cs229.stanford.edu/proj2015/249_report.pdf
- https://cs229.stanford.edu/proj2006/BabakHamadani-PredictingNFLGames.pdf
- https://www.kclas.ac.in/wp-content/uploads/2021/01/The-Cricket-Winner-Prediction-With-Application-Of-Machine-Learning-And-Data-Analytics-.pdf
- https://indjst.org/download-article.php?Article_Unique_Id=INDJST12050&Full_Text_Pdf_Download=True
- https://content.iospress.com/download/journal-of-sports-analytics/jsa200436?id=journal-of-sports-analytics%2Fjsa200436
- Kelleher, J. D., Brian Mac Namee, & Aoife D'arcy. (2015). Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies. The MIT Press.

The END