Increase the Efficiency of Capturing a Pokemon: Predict Rare Pokemons' Showing up Based on Historical Data

Wei Li CSI5155

***Abstract*— Pokemon Go is an AR mobile game where players are interested in catching rare Pokemons in the game. Predict'em All consists of 292,061 historical Pokemons' appearing data including some of the most rare types. Based on the relevant information in the data set, we conducted feature engineering and trained machine learning models that predict rare pokemons' appear. The best result is about 0.5 recall rate and 0.1 precison on the rare classes. We compared the pros and cons of different classification models, different sampling methods, learning models in highly imbalanced data set and whether the number of classes in classification tasks can impact models ability to predict minority classes. The Github repository to this project can be found in PROJECT REPO.**

I. Introduction

A. Overview

This report investigate the effect of different machine learning approaches in predicting the Pokemon Go Predict'em All data set. Predict'em All is a imblanced data set, where minority classes takes up less than 5% data points in the data set. We tested 6 supervised machine learning algorithms and an anormaly detection algorithm on the data set, and got minority classes' recall rate of at most 56%.

B. Predict'em All

Pokemon Go is an augmented reality mobile game where users can catch Pokemons that they encounter in real-world and train them to fight with other peoples'. There are in total 151 Pokemons in the game. Some of the Pokemons appears frequently and are easy to be caught, while others do not. Intuitively, since Pokemons have their unique habitats and living habits just like animals, their appearing may be relevant to the environment in the real world, like local time, weather, nearby building, close to water or not, etc. Predict'em All tracked 292061 appearing in different part of the world in several consecutive days, and gives 208 features about the time, location and environment. The 'PokemonId', which is a unique Pokemons identifier of the appearing pokemon, is what we are interested in.

C. Objective

Just like players of all collecting games, players of Pokemon Go expect to catch rare Pokemons. In the project, we categorized the 151 class 'PokemonId' into 5 rarity classes. They are {Common, Uncommon, Rare, Very Rare, Super Rare}. Given relevant information, we want to find a good classifier that can predict the rarity class, i.e tell us Pokemons from which rarity class are more likely to show up. This information can give gamers a guideline when they want to catch rare Pokemons. Moreover, since players are interested in rare Pokemons, we focused more on the 'Very Rare' and 'Super Rare' classes. Our objective is to find classifiers that can successfully predict the appearing of Pokemons from the two rare classes. The skewness of the number of appearing in different rarity classes make the task very hard for simple classifier designed for balanced data set.

D. Experiment

We did three experiments in section III. In the first part, we trained 6 models, a kernalized SVM, a decision tree, a 4-nearest neighbor, a rule learner, a random forest and a AdaBoost classifier on the data set. We used 10-fold cross validation to estimate the average score of the 6 models. The scores we are interested in is recall, precision and f1 score of rarity class '4' and '5', the confusion matrices and classification accuracy (which is not as important as the other scores). We also compared the score of two different sampling approach, the first one is under-sampling data set into a smaller data

set without applying SMOTE, and second one is under-sampling class '1', '2', and '3' and using SMOTE to over sample class '4', '5', which gives a bigger training set, and also induces some bias.

In the second part, we merged data points in rarity class '1', '2', '3' into class '0', and merged data points in rarity class '4', '5' into class '1', and used the balanced sampling method to retrain the 6 models, to see if reducing the problem from 5-classes classification to binary classification can make the prediction of minor class easier.

Finally, in the binarized data set, we tried an unsupervised anomaly detection algorithm one-class SVM to see if it performs differently to algorithms in the second part.

II. Case Study

In this section, we explore the Predict'em All data set and conduct feature engineering from scratch. The original data has 296,021 data points, 207 features and 151 classes to be predicted. After the feature engineering, it can be seen that many of the features are redundant, and the transformed data set consists of 271,050 data points and 58 features, with 5 classed to be predicted.

A. The Data Set

There are 207 features and 151 predicting classes in the data set. We use ⟨feature number⟩ to denote the feature's number in the original data set. These features can be categorize into several subsets. The original features is given in Table 1.

B. features about Pokemon

Firstly, Figure 1 gives a bar chart of the <0> PokemonId. The distribution of the appearing number of Pokemons is sparse and skew. Of the 151 Pokemon types, only few of them appeared for more than 10,000 times, while most of them appeared for just a few hundreds times. It is also worth mention that some of the PokemonId never appeared in the data set, although there are 296,021 data points. For such a distribution, it is hard to predict the individual PokemonId directly. So we considered replacing the PokemonId with a low-dimensional rarity classes.

Rare Pokemon List gives a 5 level rarity classes for Pokemons based on the game's data. In the list, 10 Pokemons are 'common', 40 Pokemons are 'uncommon', 80 Pokemons are 'rare', 10 Pokemons are 'very rare' and 10 Pokemons are 'super rare'. Based on the information, we converted the 'pokemonID' to 'rarity', which is a 5 class ordinal attribute range from 1 to 5 (1 stands for 'common', 5 stands for 'super rare'). After that, feature ⟨0⟩ and ⟨207⟩ is not needed.

The bar chart of 'rarity' is plotted in Figure 2, while still very skew, the problem became much more accessible.

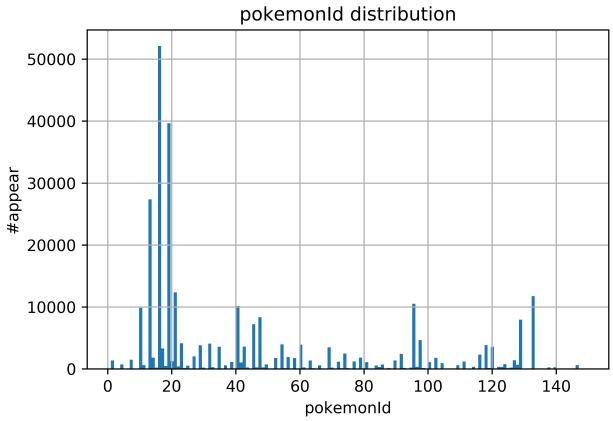

Fig. 1. Distribution of PokemonId

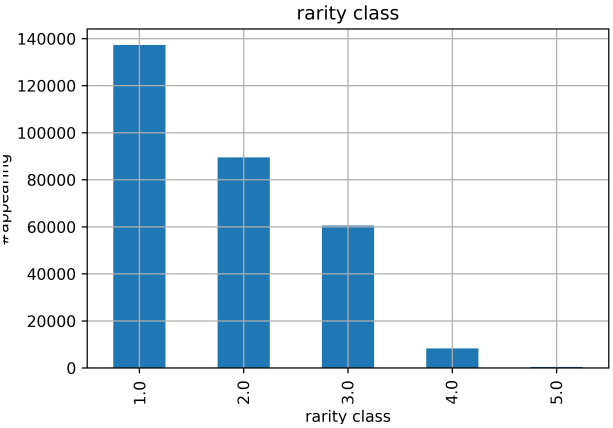

Fig. 2. Distribution of rarity class

Features 'cooc_1' to 'cooc_151' are 151 one-hot encoded features about whether a particular

Pokemon type appeared before the current sighting. Since we converted pokemonId into 5 rarity classes, these features can also be converted to 5 one-hot encoded features about whether Pokemon from other rarity classes appeared previously.

C. features about location

Feature ⟨6⟩ to ⟨12⟩ are about Googles S2 geographic Library, which is a geographic coordinate that is very similar to latitude and longitude. Since these features have similar information, we kept only latitude and longitude.

It was observed that data points in the same 'city' have very similar latitude and longitude, so the feature 'city' is redundant to the latitude and longitude.

For feature ⟨23⟩ 'continent', Figure 3 is a data visualization over a map about the number of data points in different locations. In the map, most of the sighting were recorded in America and Europe, although the data sparsely distributes in 98 different cities and 11 continents(and oceans) in the world. Figure 4, on the other hand, shows that America and Europe have most of the data points in the data set.

Since most of the players comes from America, Europe and Asia (over 97%), and we found that the distribution of the 5 rarity classes are the nearly identical regardless the continents (see Figure 5 to Figure 7), we decided to keep only data points from America, Europe and Asia.

We also discovered that although America, Europe and Asia have similar rarity classes distribution, Europe has the highest rate of 'super rare' (Europe: 2.8×10^{-3} ,America: 7.84×10^{-5}, Asia:7.12×10^{-5}), and Asia has the highest rate of 'very rare' (Asia:0.045, Europe:0.030, America:0.027). Although America has the highest number of data points, it has the lowest rate of rare Pokemons.

Finally, features ⟨29⟩ to ⟨36⟩ is about the local sunrise/sunset time. The feature can be calculated from latitude and longitude, and another feature ⟨13⟩'appearedTimeOfDaynight' gives similar information, so we decided to drop it.

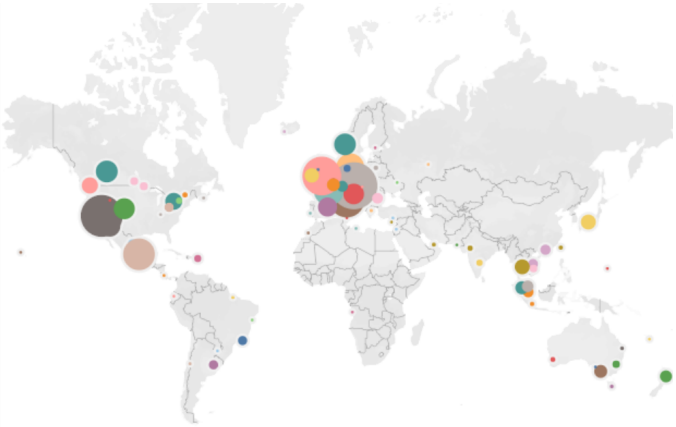

Fig. 3. Geographical Distribution of data

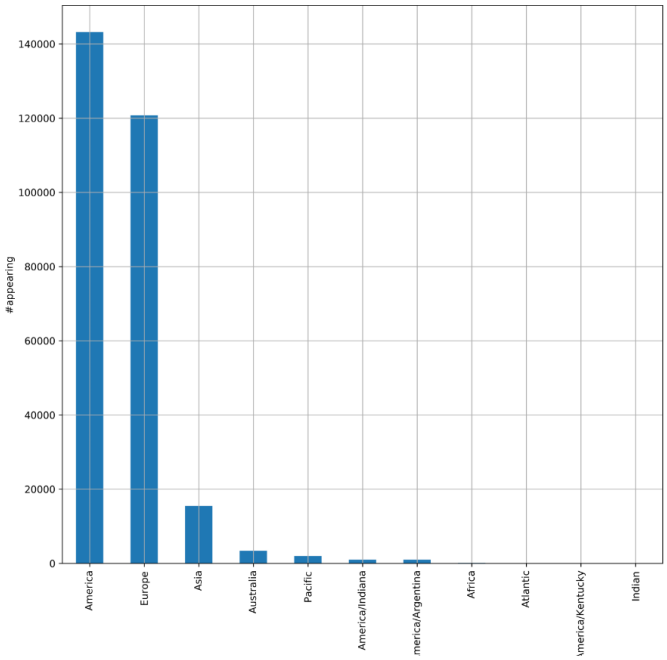

Fig. 4. sample by continent

D. features about time

It observed that all the data was collected in 6 consecutive days from August 2, 2016 to August 8, 2016. As a result, features 'appearedMonth' and 'appearedYear' all have same value on all data points. Also, in such a short observation time, it can be safely assumed that there is no concept drift in the data set (it is unlikely that the game operation company changed their scheme), and the

feature number	feature name	description
features about Pokemon		
<0>	pokemonId	the identifier of a pokemon
<207>	class	pokemonId of the appearing, same value as <0>
<56>to<206>	cooc 1 to cooc 151	co-occurrence with any other pokemon (pokemonId) within 100m in last 24 hours
features about location		
<1><2>	latitude, longitude	coordinates of a sighting
<6>to<12>	cellId 90m-to 5850m	geographic position projected on a S2 Cell
<22>	city	the city of a sighting
<23>	continent	the continent of a sighting
<29>to<36>	sunriseMinutesMidnight-sunsetMinutesBefore	local time of sunrise/sunset
features about time		
<3>	appearedLocalTime	time of a sighting in format yyyy-mm-dd'T'hh-mm-ss.ms
<13>	appearedTimeOfDay	night, evening, afternoon, morning
<14><15>	appearedHour/appearedMinute	local hour/minute of a sighting
<16>	appearedDayOfWeek	week day of a sighting
<17>to<19>	appearedDay/appearedMonth/appearedYear	day/month/year of a sighting
features about surrounding		
<20>	terrainType	terrain where pokemon appeared (GLCF Modis Land Cover)
<21>	closeToWater	did pokemon appear close (100m or less) to water (Boolean)
<37>	population density	the population density per square km of a sighting
<38>to<41>	urban-rural	how urban is location where pokemon appeared
<42>, <43>to<48>	gymDistanceKm, gymIn100m to 5000m	how far is the nearest gym, is there a gym in 100/200/etc meters
<49>, <50>to<55>	pokestopDistanceKm , pokestopIn100m to 5000m	how far is the nearest pokestop, is there a pokestop in 100/200/etc meters
features about weather		
<23>	weather	weather type during a sighting
<28>	weatherIcon	compact representation of the weather(same as weather, less information)
<24>	temperature	temperature in celsius at the location of a sighting
<25>	windSpeed	speed of the wind in km/h at the location of a sighting
<26>	windBearing	wind direction
<27>	pressure	atmospheric pressure at the location of a sighting

TABLE I
FEATURES OFPREDICTEMALL

period of the data is not long enough to show any periodicity. As a result, 'appearedDay' is also not useful.

'appearedLocalTime' contain the same information as 'appearedHour' and 'appearedMinute', so we kept only the latter two columns.

An important common characteristic of 'appeared-Hour', 'appearedMinute', 'appearedDayOfWeek', and "longitude' in the previous section is that they are all cyclical and can be seen as continuous. For example, although 12:00 in 24hr time format is far away from 0:00, 23:29 is very close to 0:00; Although E0(0) in longitude is far away from W180(-180), E180(+179) is very close to W180(-180). To let the modeld understand that these features are cyclical, we normalize each of the features into range $[-\pi, \pi]$, and applied sin and cos transformation to the normalized feature, generated 2 new features (sin and cos). After this transformation, the models are able to understand that while the features' sides are far away from the middle, the two sides are close.

E. features about surrounding

We kept all the feature in this category. Features 'closeToWater', 'urban-rural', 'gymIn100m' to 'gymIn5000m', 'pokestopIn100m' to 'pokestopIn5000m' are all encoded as one-hot vectors.

Also, it was discovered that in 'gymDistanceKm' and 'pokestopDistanceKm' (i.e. how far is the nearest gym/pokestop), most of the data points

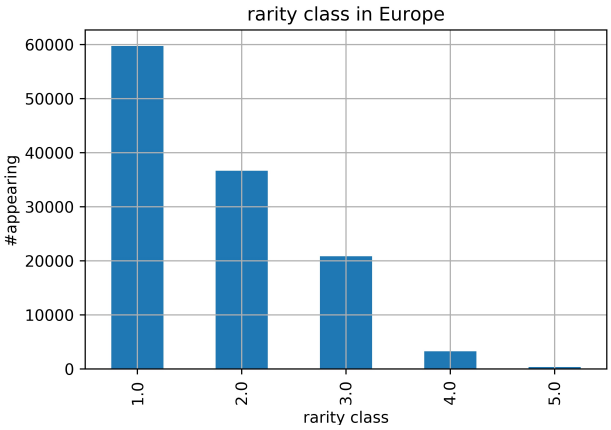

Fig. 5. Rarity in Europe

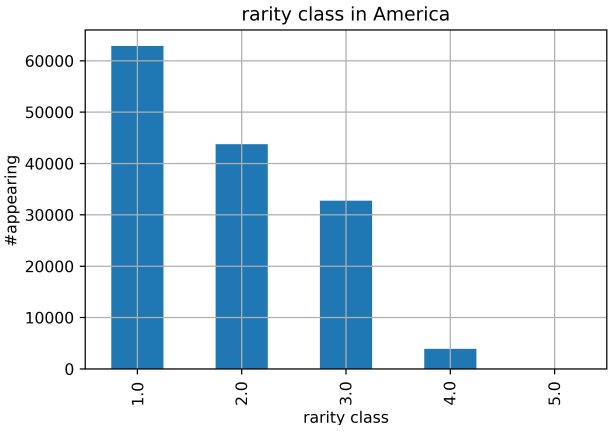

Fig. 6. Rarity in America

fall into the range [0,6], although the biggest distance can be up to 375km. We dropped all the data points that were out of the [0,6] range to make the normalization easier.

Finally, the data set did not give a clear description to the feature ⟨20⟩ terrainType. The feature contains natural integers range from 0 to 16. We did not know whether the numbers stand for nominal categories, ordinal ranking or can be continuous. To deal with the problem, we calculated the rate of 'super rare' class and the rate of 'super rare and very rare' classes for each terrainType in Table II. The result shows that some terrainTypes have higher SR rate and R&SR rate than the others, and the order of terrainType does not affect the order SR and R&SR rate. So for one thing terrainTypes may be an important feature to be considered, for

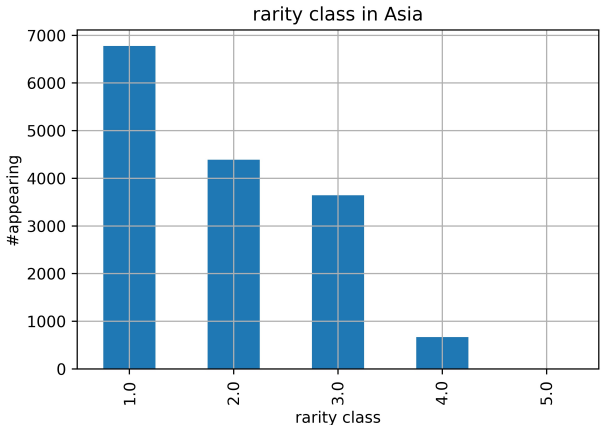

Fig. 7. Rarity in Asia

another thing, the order of terrainType does not affect the SR and R&SR rate. As a result, it is appropriate to use one-hot encoding to represent the feature.

F. features about weather

Features 'weather' and 'weatherIcon' contains the same information, and 'weatherIcon' is a simplified version of 'weather'. we kept only 'weather' in the features set. Feature 'weather' contains 26 nominal weather types, but some of them are overlapped (e.g. MostlyCloudy, Breezy, and BreezyandMostlyCloudy). We converted 'weather' into 5 ordinal features, 'Rain', 'Cloud', 'Clear', 'Wind', and 'Humid'. Ordinal number in these features represent the level of the particular weather type (e.g. In the Rain feature: 'No information or No rain' = 0, 'Drizzle' = 1, 'LightRain' = 2, 'Rain' = 3, 'HeavyRain' = 4).

All remaining features are all numeric features, and were kept in our features set.

G. Missing Values and Normalizing

There is a few missing values in the data set in the 'appearedDayOfWeek' and 'gymDistanceKm' features. The number of missing samples are small, so we dropped these data points.

By using one-hot encoding, we turned all the feature into numeric datatype for numeric based

terrainType	2	1	5	10	12	13	14
VR&SR rate	0.038627	0.024965	0.019385	0.012256	0.023231	0.031732	0.023819
SR rate	0.002861	0.002553	0.002311	0.002288	0.002277	0.001143	0.000857
terrainType	0	7	8	4	11	9	16
VR&SR rate	0.07663	0.006916	0.039244	0.003604	0.017544	0	0.014151
SR rate	0.000653	0.000401	0.00027	0	0	0	0

TABLE II

RARE AND SUPER RARE RATE IN DIFFERENT TERRAINTYPE

models like SVM and KNN. We also used MIN-MAX normalization to normalize big-value features into range [0, 1], which prevents distance based models like KNN from inducing too much bias.

The feature engineering gave us a data set of 58 features and 271050 data points, with 5 rarity classes to be predicted. We computed covariance matrix for all features and calculated SVD of the covariance matrix. We sorted the singular values can plot it according to the ranking. The dropping of singular values is not very steep (Figure 5), which indicates that although there is some linear redundancy, the problem is not very severe. Since the linear correlation between features in not a big problem, we decided to leave the data set as it is without applying PCA. The resulting feature set is given in Table III.

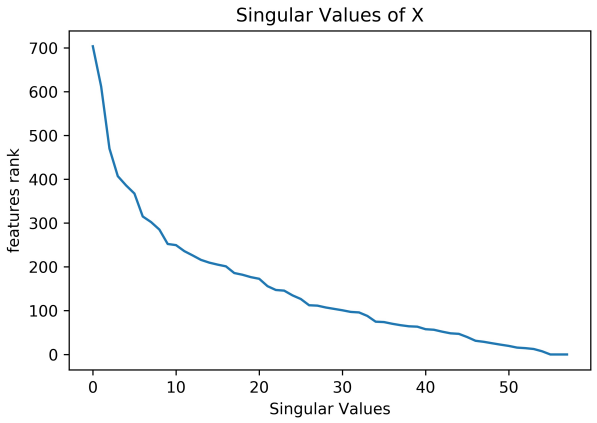

Fig. 8. Singular value of features

III. EXPERIMENTAL SETUP AND EXPERIMENT RESULT

Recall that the data set is very imbalanced. We decided to split the experiment into three part. In the first part, we used sampling methods to rebalance the data set and trained:

- Linear classifier
  - SVM
- Tree-based
  - Decision Tree
- Distance-based
  - KNN
- Rule-based
- Ensemble
  - AdaBoost
  - Random Forest

Then, in the second experiment, instead of predicting 5 rarity classes, we merged the classes '1', '2', '3' into class '0' and merged classes '4' and '5' into class '1', making the task a binary classification (*says 'common' and 'rare'). We retrained the 6 models on the binarized problem, and examined whether a binarizing problem can make the prediction of minority class easier.

Finally, in the third part, we still used the binarized data, but used an anomaly detection method, one-class SVM, to predict the appearing of rarely occurring class.

A. First Experiment: rebalancing data

1) Sampling: Because of the imbalance of the rarity classes distribution (there are 125094 class '1', 82425 class '2', 55546 class '3', 7641 class '4' and 344 class '5'), also because of the spiking

features about Pokemon		
<0>	rarity	<ordinal>how rare the appearing pokemon is
<37>to<41>	cooc_rarity_1 to 5	<boolean>co-occurrence with any other pokemon in other rarity class (rarity) within 100m in last 24 hours
features about location		
<1>	latitude	<quantative>latitude of a sighting
<30><31>	'sin_longitude', 'cos_longitude'	<quantative>longitude of a sighting
<41>	cont_America	<boolean>whether the sighting is in America
<42>	cont_Europe	<boolean>whether the sighting is in Europe
<43>	cont_Asia	<boolean>whether the sighting is in Asia
features about time		
<28><29>	sin_secOfday, cos_secOfday	<quantative>appear seconds after 0 o'clock
<26><27>	sin_weekday, 'cos_weekday	<quantative>week day of a sighting
features about surrounding		
<44>to<57>	terrain_0 to 16	<boolean>whether the pokemon appear in terrainType_i (GLCF Modis Land Cover)
<2>	closeToWater	<boolean>did pokemon appear close (100m or less) to water (Boolean)
<7>	population density	<quantative>the population density per square km of a sighting
<8>to<11>	urban-rural	<boolean>how urban is location where pokemon appeared
<12>, <13>to<18>	gymDistanceKm, gymIn100m to 5000m	<quantative>how far is the nearest gym, <boolean> is there a gym in 100/200/etc meters
<19>, <20>to<25>	pokestopDistanceKm , pokestopIn100m to 5000m	<quantative>how far is the nearest pokestop, <boolean>is there a pokestop in 100/200/etc meters
features about weather		
<32>	Rain	<ordinal>the level of Rain weather
<33>	Cloud	<ordinal>the level of Cloud weather
<34>	Clear	<ordinal>the level of Clear weather
<35>	Wind	<ordinal>the level of Wind weather
<36>	Humid	<ordinal>the level of Humid weather
<3>	temperature	<quantative>temperature in celsius at the location of a sighting
<4>	windSpeed	<quantative>speed of the wind in km/h at the location of a sighting
<5>	windBearing	<quantative>wind direction
<6>	pressure	<quantative>atmospheric pressure at the location of a sighting

TABLE III

computation complexity of some classifiers like kernal SVM when the size of training set grows, we used sampling method.

We considered under-sampling and balanced sampling. Concisely, by under-sampling, we reduced the class '1', '2', '3', '4' to 2500 samples, and kept the class '5' without change. This can give us a data set of about 10300 data points, much balanced without inducing any bias. However, the data set may be too small. So we also considered balanced sampling. We randomly reduced class '1', '2', '3' to 10000 samples, and used SMOTE to oversample class '4' to 10000 samples and oversample class '5' to 5000 samples. Using oversampling can give a bigger data set (45000 data points), but it may also induce to some to bias to classifier about class '4' and class '5'.

In the experiment, we used 10-fold cross validation to evaluate each models' generalization ability. To minimize the bias induced by sampling, we sampled train set (oversample or undersample) only after the partition of train set and validation set in each fold was made. We tried to used SMOTE to oversample rarity class '5' before the partition of train set and test set, and it made class '5' 's recall in test set close to 1 in almost all models, which is unlikely to be true. The reason is that SMOTE created the same bias in both train set and test set,

and the bias is easy to be learnt by most models. Since the same bias also exist in test set, these models can also do well in test set.

We did a toy example to demonstrate the effect of use SMOTE before and after split. We used 10-fold cross validation over a DecisionTree classifier, and calculate the 5 classes' average recall, precision and f-measure on the validation set. The first run was on the whole data without sampling; the second run was undersample class '1', '2', '3' to 10,000 and oversample class '4' to 10,000 and '5' to 5,000 ***before*** splitting test set and train set; the third run use the same sampling strategy, but do it ***after*** the splitting.

Table IV shows the result. Using SMOTE on class '5' before splitting training set and test set increase the model's recall and precision significantly. But when SMOTE was applied after splitting, the recall and precision was much smaller. We also saw the effect of rebalancing data. Comparing to the model that trained on the original data set, the recall of class '5' increase from 0.008 to 0.0053, and precision decrease from 0.008 to 0.005 in the decision tree that trained on rebalanced data set.

2) evaluation criteria: The criteria used to evaluate the performance of models is multi-classes recall, precision and f1 measure. We also computed confusion matrices for classifiers as confusion matrices given a straightforward visualization of the distribution of prediction results. More concisely, the criteria that we mostly interested in is the recall of rarity class '4' and class '5'. Since players interested in rare pokemons, they want a classifier that give them the most accurate warning when rare pokemons have high chance to shows up. However, as there is a trade-off between recall and precision, we also do not want precision to be too low. In the experiment, we used the recall, precision and f1 measure obtained from Decision Tree train over all data without sampling as our baseline (the first-lines of TABLE IV), and tried to improve the result.

3) Construction of SVM: The first model we used is a SVM algorithm. We used a 'rbf' kernal, and used OnevsAll for multiclass classification. We firstly trained it over under sampled train set, and then trained it over balanced sampled train set. The overall accuracy is 0.3528 for the first sampling method and 0.3705 for the second sampling method.

Figure 9 and 10 shows the average confusion matrix and scores of the two sampling method on SVM. It shows that by over sample class 5

under sample				
5673.6	2619.2	1541.1	2675.5	0
2797.3	1983.5	1369.8	2091.9	0
1452.2	1020.1	1439	1643.3	0
152.9	90.5	52.4	468.3	0
16.2	8.5	1.2	8.5	0
balanced sample				
5872.3	2379.9	1809	1847.1	601.1
2776.6	1967.2	1649.9	1462.8	386
1410.7	997.1	1805.7	1150	191.1
155.8	89.7	98	393.8	26.8
14.1	9.3	1.6	5.3	4.1

Fig. 9. Confusion matrix for different sample method on SVM

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.5018	0.0469	0.5622	0.5741	0.4534	0.4694
2	0.2840	0.0239	0.3468	0.3619	0.2406	0.2386
3	0.2888	0.0325	0.3269	0.3368	0.2591	0.3251
4	0.1226	0.0516	0.0681	0.0811	0.6129	0.5156
5	0.0000	0.0119	0.0000	0.0034	0.0000	0.1193

Fig. 10. scores for different sample method on SVM

to 5000, the classifier performed much better in class '5'. The recall of class 5 increased from 0 to 0.119, while the precision and recall of other majority classes did not change a lot. In conclusion, using SMOTE to over sample class '5' helps increase the recall rate on class '5'.

We also tried to over sample the '5' class to 10000 data points, but resulting recall of SVM was smaller than 0.119. Oversampling more

	class	1	2	3	4	5
recall	no sample	0.574	0.374	0.334	0.329	0.008
	sample before split	0.353	0.315	0.341	0.592	0.875
	sample after split	0.362	0.315	0.342	0.506	0.053
precision	no sample	0.582	0.372	0.328	0.328	0.008
	sample before split	0.358	0.318	0.343	0.577	0.863
	sample after split	0.553	0.337	0.263	0.107	0.005
f1 measure	no sample	0.578	0.373	0.331	0.329	0.008
	sample before split	0.355	0.316	0.342	0.584	0.869
	sample after split	0.438	0.326	0.297	0.176	0.009

TABLE IV

COMPARING SAMPLE BEFORE AND AFTER SPLIT

minorities may not necessarily helps to predict minorities.

It is also discovered that training kernal SVM took the longest time among the 6 models.

4) Construction of Decision Tree: We then trained decision tree models over the same under sampled and balanced sampled data set. The impurity was measured by Gini impurity. The accuracy is 0.3056 for the first sampling method and 0.4305 for the second sampling method.

under sample				
3913.3	3289.6	2818	2043	445.5
2153.6	2317.3	2094.1	1406.3	271.2
1234.5	1424.9	1694.2	1067.4	133.6
119.1	122.4	146	358	18.6
9.6	10.2	6.2	5.8	2.6
balanced sample				
6510.7	4681.9	840	422.2	54.6
3092.3	3985.6	776.1	350.2	38.3
1805.9	2438.8	866.6	428.3	15
163.2	207.7	82.7	307.4	3.1
14.2	16.4	1.8	1.2	0.8

Fig. 11. Confusion matrix for different sample method on decision tree

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.392488	0.5404	0.5267	0.5619	0.3129	0.5204
2	0.300786	0.4072	0.3235	0.3518	0.2811	0.4835
3	0.275177	0.2134	0.2507	0.3375	0.3050	0.1560
4	0.126998	0.2705	0.0735	0.2039	0.4684	0.4025
5	0.005969	0.0109	0.0031	0.0072	0.0722	0.0234

Fig. 12. scores for different sample method on decision tree

Figure 11 and 12 shows the average confusion matrix and scores of the two sampling method on decision tree. The scores and confusion matrices shows that although decision tree performed better in majority classes(and thus the overall accuracy was higher), its ability to predict minority classes '4' and '5' was weak compare to SVM. However, its fast training speed makes it still a good model to be considered.

5) Construction of 4-nearest neighbor: We constructed 4-nearest neighbor model over the two sampled train sets. Before the experiment, we tested the algorithm on the same train set and test set to determine the best k (number of neighbors). We tested k range from 2 to 8, and the result shows that the performance of k (in classifying minority classes) in range 4 to 8 is close, but k = 4 is a little better. Since k=4 also requires the least computation, we chose the 4-NN model. Since

we had no prior knowledge to the features, we used the default 2-minkowski diantance(Euclidean distance). It turned out that 4-NN is the second slowest algorithm among the 6 candidate algorithms. Because of the large sample set and large feature space, the model spent most of its time calculating the distance of points in validation set and points in train set.

The accuracy of the two sampling method is 0.344 and 0.347 respectively. The following Figure gives the result of 4-NN.

under sample				
5293.5	3256.9	2072.7	1784	102.3
2888.6	2351.3	1647.3	1296.5	58.8
1657.3	1430.2	1355	1080.6	31.5
167.1	139.7	129.5	323.6	4.2
14.8	9.7	4.4	5.1	0.4
balanced sample				
5305.2	3032.6	1825.5	1735.6	610.5
2747.4	2337.5	1517.8	1275.3	364.5
1527.2	1369.6	1386.3	1099.3	172.2
135	118	114.6	377.7	18.8
11.5	10.1	4.3	4.7	3.8

Fig. 13. Confusion matrix for different sample method on 4-NN

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.469886	0.4772	0.5282	0.5455	0.4232	0.4241
2	0.304733	0.3094	0.3271	0.3404	0.2853	0.2836
3	0.251764	0.2665	0.2601	0.2860	0.2440	0.2496
4	0.123179	0.1437	0.0721	0.0841	0.4234	0.4946
5	0.003442	0.0063	0.0020	0.0032	0.0117	0.1100

Fig. 14. scores for different sample method on 4-NN

Again, balanced sampling gives a better result on the minority classes. Also, 4-nearest neighbor's recall on rarity class '4' and '5' is very close to that of SVM. Since 4 nearest neighbor can be trained a little faster than SVM, it may be a better tool to be considered over SVM.

6) Construction of Rule Learner: We warped a rule classifier with a OnevsAll method for multi-classes classification. However, the result is very weird.

under sample				
125	198.7	337.6	305	11543.1
45.8	165.6	450.6	287.7	7292.8
20.9	72.4	636	282.1	4543.2
1.7	4.3	7.5	83.8	666.8
0.9	0.6	0.5	0.7	31.7
balanced sample				
17.3	22.3	155.9	307.2	12006.7
7	23.5	227.7	359.2	7625.1
4.7	11.8	373.2	359.6	4805.3
0.2	0.9	2.6	110.7	649.7
0	0.1	0.3	0.9	33.1

Fig. 15. Confusion matrix for different sample method on rule-learning

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.018966	0.0027	0.2535	0.0592	0.0100	0.0014
2	0.037156	0.0056	0.2923	0.1215	0.0200	0.0029
3	0.178949	0.1163	0.4562	0.5010	0.1145	0.0673
4	0.073069	0.1156	0.0692	0.1057	0.1089	0.1451
5	0.002628	0.0026	0.0013	0.0013	0.9188	0.9630

Fig. 16. scores for different sample method on rule-learning

As shown in figure 15 and 16, the rule classifier classified most of the data points to the class '5', the most rarely occurred class. As a result, the classifier made the recall of class '5' close to 1 and made its precision close to 0. The accuracy of the classifier is also close to 0.

However, when we then tested pair-wise binary classification of rarity class '5' and another class using rule learning, the classifier can separate the two classes soundly. We concluded that rule learning is not good for such a multi-classes classification problem. Beside, in the binarized problem in the next experiment, rule classifier also produced reasonable result.

7) Construction of Random Forest: The random forest we used contained 100 weak learners. The

models was also trained on the two differently sampled train sets. The random forest did not do good in predicting minority classes comparing to other classifiers.

under sample				
6518.7	1368.6	1517	3105.1	0
3456.9	969.6	1318.4	2497.6	0
1786.3	478.3	1397	1893	0
202.9	42.5	51.9	466.8	0
20.6	3.9	0.6	9.3	0
balanced sample				
4865.6	46.6	5532.5	1928	136.7
2789.3	55.8	3700	1606.7	90.7
1374.2	40.6	2728.7	1364.8	46.3
179.8	2.1	217.7	357.9	6.6
27.1	0.6	0.5	5.4	0.8

Fig. 17. Confusion matrix for different sample method on Random Forest

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.531974	0.4473	0.5440	0.5268	0.5210	0.3889
2	0.171353	0.0132	0.3400	0.4008	0.1178	0.0068
3	0.282846	0.3077	0.3292	0.2241	0.2515	0.4912
4	0.107496	0.1187	0.0590	0.0680	0.6110	0.4688
5	0	0.0024	0.0000	0.0013	0.0000	0.0222

Fig. 18. scores for different sample method on Random Forest

The random forest is close to decision tree in this problem in that they both had only 0.02 recall rate on class '5', worse than SVM and 4-NN. But the two algorithms also had slight better recall on class '4' comparing to SVM and 4-NN, and they can also be trained much faster than SVM and 4-NN. So depending on the rarity class we are interested in ('4' or '5'), decision tree and random forest may be a better choice.

8) Construction of AdaBoost: The AdaBoost model used decision trees as base estimator. The algorithm was forced to terminated when the number of based estimators grows over 50. The balanced sampling method still outperform the under sampling method. It is also discovered that AdaBoost's recall rate on the class '4' and '5' is close to that of SVM and 4-NN. Since AdaBoost can be trained much faster than the two algorithms, it is a more desirable choice than the other two when training time (or evaluation time) is a big concern.

under sample				
5109.9	2365	1961.7	2848.1	224.7
2622.7	1764.5	1585.2	2124.7	145.4
1385.9	967	1504.7	1598.2	98.8
144.1	93.3	125.9	393.7	7.1
15.3	7.4	1.7	8.9	1.1
balanced sample				
5423.6	2034.8	2028.8	2590.7	431.5
2728.9	1563.2	1673.3	2016.6	260.5
1474.3	804.7	1623.1	1506.8	145.7
153.3	82.2	129.1	382.6	16.9
14.2	7.2	1.1	9.3	2.6

Fig. 19. Confusion matrix for different sample method on AdaBoost

	f		precision		recall	
class	under	balance	under	balance	under	balance
1	0.4681	0.4858	0.5508	0.5543	0.4085	0.4336
2	0.2614	0.2446	0.3407	0.3481	0.2140	0.1896
3	0.2782	0.2905	0.2932	0.3027	0.2707	0.2921
4	0.1020	0.1058	0.0567	0.0592	0.5149	0.5014
5	0.0046	0.0055	0.0025	0.0029	0.0315	0.0833

Fig. 20. scores for different sample method on AdaBoost

9) Conclusion Synthesis:

- We tested two different sampling method, one is under sampling class '1','2','3','4' to 2500, and do nothing to class '5', the other is under sampling class '1','2','3' to 10000, and over sampling class '4' and class '5' to 10000 and 5000 respectively. The result shows that the second sampling almost always better than the first one in terms of the recall rate and precision of minority class. The reason may be that the second sampling method preserved more data from the majority classes, and enable classifiers to learn more information to separate minority

classes and majority classes.

- In the 5-classes classification problem, because of the similarity of the three majority classes (class '1', '2','3'), none of the classifiers got a descent overall accuracy. The best accuracy we got is 43% in decision tree. However, as what we tested in the following experiment, when the problem was turned into a binary classification, the accuracy can go up to 90% while the classifiers can still distinguish minority class from majority class soundly.

- Based on the experiment result above, we categorized the 6 algorithms into 3 categories. Kernalized SVM and 4-NN belongs to the first category: their training (or evaluating) took a long time even in small data set, but they gave the highest recall rate on the minority class '4' and '5' without making the precision going too low. Decision tree, random forest and AdaBoost belong to the second category: they can be trained fast, but their recall on class '5' is not as good as SVM and 4-NN. The decision tree and random forest performed almost identical in this experiment, but AdaBoost had a performance on minority classes that is close to SVM and 4-NN. The rule based classifier belongs to the third category, which perform badly in the multi-classes classification problem.

B. Second experiment: binarizing rarity classes

In this experiment, we shrank the number of classes to be predicted to make the classification easier for classifiers. Concisely, what we did is:

1) merging the '1', '2', and '3' classes into '0' class, and merging the '4' and '5' classes into '1' class. The resulting data set contains 263065 data points in negative class, and 7985 data points in positive class. The positive class accounted for 3% data points in the data set. This is still an imbalanced data set.

2) using the same balanced sampling method as the first part, we under sampled class '0' to 30000 data points and over sampled class '1' to 15000 data points

3) retraining the 6 models in the binarized data set. (using 10 fold cross validation, calculating the averaged recall, precision and F1 measure for each class).

Figure 21 and 23 gives the experiment result.

RF	ada	svm
0.9705405	0.9004353	0.9039329
rule	tree	4nn
0.8537392	0.844442	0.7906106

Fig. 21. Accuracy after binarize classification target

precision	recall	f score	confusion metrix	
0.97139	0.49966	0.65988	13144.2	13162.3
0.03031	0.5154	0.05725	387.1	411.4

Fig. 22. Scores of one-class learning

The result shows that the classifiers got higher accuracy after binarize and also got higher recall and precision on the minor classes (although sometimes not as good as the score of class '4' before binarization). In the 2 classes classification case, 4-NN got the highest recall, and SVM got the highest precision on the positive class. SVM and AdaBoost got close and more balanced score (in recall and precision). However, the best model is decision tree since it got a very high recall rate and a not very bad precision on positive class comparing to other models. Also decision tree can be trained much faster than models like SVM and 4NN. Also note that random forest cannot predict positive class at all, although it obtained the highest accuracy.

C. Third experiment: using one-class learning

In this section, we implemented one-class SVM to detect class positive (pokemons rarity level '4' and

AdaBoost	precision	recall	f measure	confusion matrix	
	0.978	0.918	0.947	24147.1	2159.4
	0.107	0.325	0.161	539.3	259.2
4NN	precision	recall	f measure	confusion matrix	
	0.983	0.798	0.881	20983.6	5322.9
	0.077	0.559	0.136	352.6	445.9
tree	precision	recall	f measure	confusion matrix	
	0.983	0.854	0.914	22470.4	3836.1
	0.098	0.524	0.166	380.3	418.2
RF	precision	recall	f measure	confusion matrix	
	0.971	1.000	0.985	26306.5	0
	0.000	0.000	0.000	798.5	0
rule	precision	recall	f measure	confusion matrix	
	0.976	0.871	0.920	22900.6	3405.9
	0.066	0.301	0.108	558.5	240
SVM	precision	recall	f measure	confusion matrix	
	0.980	0.920	0.949	24205.5	2101
	0.123	0.370	0.185	502.9	295.6

Fig. 23. Scores after binarize classification target

'5') as anormalies.

Firstly, we binarized the target to be predict, then we used 10-fold cross vaildation. In each fold, we randomly under sampled the train set to 50000 data points (because of the long training time of SVM), and trained one-class SVM on the train set. Finally, we used the trained one-class SVM to predict anormalies on the test set, and compared it with the real labels.

The biggest difference of one-class SVM and algorithms above is that one-class SVM is unsupervised, meaning that it did not utilize the labels we have. This is a huge disadvantage to one-class SVM. Actually, comparing to other supervised classification models, one-class SVM did really bad. The average accuracy is 0.50, Figure 22 gives the averaged confusion matrix, recall, precision and f score.

From the result, it can be seen that one-class SVM's performance is not better than randomly guess the two classes. Although its recall of positive class is comparable to that of other classifiers, its precision of positive class is the smallest one (apart from random forest). In the confusion matrix, about half of the negative samples were misclassified as positive, and about half of the positive samples were misclassified as negative.

The result also indicate that there may not be a clear boundary between common pokemons and rare pokemons in the feature space we have. The one-class learning's bad performance may due to many overlapping in feature space of class positive and class negative.

IV. CONCLUSIONS

We conducted data analysis on the Pokemon Go Predict'em All data set. The data set is very imbalanced, even after we compressed the 151 classes to be predict into 5 classes and 2 classes. We did feature engineering to the data set, built and tested 6 supervised algorithms and one anormaly detection algorithm, we also tried two different

sampling methods. We discovered that 1)In imbalanced data set like this, oversampling methods like SMOTE can help preserve more data points and thus make classification algorithms perform better. 2)The 6 algorithms can learn from the data and can predict minority classes to some extend, but the performance is limited. The one-class SVM did not used the train labels, and its not better than a random guessing model. 3)Generally, SVM and k - nearest neighbor perform good on minority class in imbalanced data, but their training/evaluation cost is very high. AdaBoost's performance is close to the two models, and can be trained fast. Bagging method like random forest perform very bad in imbalanced data set. 4)Binary classification problem is easier than multiclasses classification problem for most of the classifiers.

We also discovered that the features' ability to predict the appearing of Pokemon is not very good. The best result we got is from the decision tree in the binarized classification problem, with minority class recall rate 0.524 and precision 0.098. The features have some ability to predict Pokemons, but they are not the best predictor variables. If subsequent data set can obtain more relevant features, the result might be better.

All in all, although those simple models can only predict the rare Pokemons' appearing in a limited accuracy, they can still serve as a good guideline to players (for example, by feeding in real life information into a light weighted classifier, players can get rare-pokemons-appearing-warning that have up to 50% recall rate).

REFERENCES

[1] SemionKorchevskiy. (2016). Predict'em All. Retrieved from https://www.kaggle.com/semioniy/predictemall

[2] Joe Ramir. (2016). Some Exploratory Data Analysis. Retrieved from https://www.kaggle.com/jraramirez/some-exploratory-data-analysis

[3] pokego. (2019). Rare Pokmon List. Retrieved from http://www.pokego.org/rare-pokemon-list/rare

[4] Ian London. (2017). Encoding cyclical continuous features - 24-hour time. Retrieved from https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

[5] Christian S. Perone. (2015). Googles S2, geometry on the sphere, cells and Hilbert curve. Retrieved from http://blog.christianperone.com/2015/08/googles-s2-geometry-on-the-sphere-cells-and-hilbert-curve/

[6] dingo. (2017). VA-PPT. Retrieved from https://public.tableau.com/profile/dingo2924!/vizhome/VA-PPT/Dashboard1?publish=yes