Poker Hand Recognition

Poker Hand Rank Recognition Using Machine Learning Models

Ömer Faruk Merey
Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey
o.merey@etu.edu.tr

Ömer Davarcı
Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey
o.davarci@etu.edu.tr

***Abstract*—Imbalanced class distribution is a common issue in machine learning, particularly in applications with binary classes. In recent years, various approaches have been proposed to address this problem in real-world tasks such as credit card fraud detection and oil spill detection. However, less attention has been given to addressing class imbalance in datasets with more than two classes. When classifying imbalanced multi-class datasets, models often misclassify minority classes as majority classes, leading to low accuracy. While some methods have been proposed for handling imbalanced multi-class data, there is still a lack of research on extreme cases of class imbalance. In this study, we examine a range of methods for addressing imbalanced data, using the Poker Hands dataset from the UCI Machine Learning Repository as a starting point. However, we find that this dataset is not suitable for improving the performance of the selected methods. As a result, we create a new poker hands dataset with a more balanced class distribution, including more instances of rare classes and fewer instances of common classes. Our goal is to use this new dataset to improve the effectiveness of the methods we have chosen for this project.**

***Index Terms*—machine learning, deep learning, multi-class classification, imbalanced data, poker**

I. Introduction

There has been significant research into finding more effective ways to handle imbalanced data in machine learning [1], particularly in binary classification tasks such as medical diagnosis [2] and credit card fraud detection [3]. However, multi-class imbalanced data classification is generally more challenging than binary imbalanced learning.

One example of imbalanced data in a practical setting is the game of poker, where players are dealt 5 cards from a deck of 52 unique cards and ranked into one of 10 distinct hand categories. The probability of each hand type is skewed, with the most common being ”nothing” or a single pair and the least common being a royal flush. In fact, the probability of being dealt a royal flush is only 0.000154%, or one royal flush for every 649,740 hands dealt. This extreme class imbalance presents a challenge for accurately classifying poker hands.

The focus of our work is on the imbalanced probabilities of poker hands and the challenge of accurately classifying them. While it is relatively easy to create a system that can rank poker hands using simple boolean expressions, the real challenge lies in creating a system that can recognize hands that only appear a few times out of millions of samples. This extreme class imbalance presents a unique challenge for machine learning algorithms.

In this study, we compare the performance of XGBoost, CatBoost and Random Forest Classifier, Multi-layer Perceptron Classifier (MLP) from the sci-kit learning kit on a multi-class classification task using poker hand data. This dataset is particularly interesting due to its extreme class imbalance. We also propose a new poker hand dataset with more balanced class distribution and more representative of real-world scenarios that is generated using SMOTE and Weighted SMOTE (WSMOTE). Using our generated data set we also create new columns to help our models with the results. Finally, we compare our models to our generated data sets and raw data sets to test our work.

II. Related Work

A. Related Work for Multi-class imbalanced Data

To address the issue of imbalance data classification, various methods and techniques have been proposed. These can be grouped such as data collection, data augmentation, sample and class weights, validation metrics, hyperparameter tuning. These are the main topics we worked on our project.

1) Data Augmentation: Data augmentation techniques for imbalanced data can be grouped into three main categories: oversampling, undersampling, and hybrid. These approaches aim to adjust the class distribution of the dataset in order to address imbalances. Oversampling involves increasing the number of samples in the minority classes by generating new samples, such as through random duplication. This approach aims to balance the dataset by bringing the minority class up to the same level as the majority class. On the other hand, undersampling involves randomly removing samples from the majority class to bring it down to the level of the minority class. This can be done in order to balance the dataset and make it more representative of the overall population. The hybrid technique combines oversampling and undersampling by applying oversampling to one or more minority classes and undersampling to one or more majority classes. This approach aims to balance the dataset while also preserving the overall distribution of the data. These methods may have been shown to improve performance in classification tasks, they do have some dramatic drawbacks. Using undersampling, may cause losing a potential of useful data. Using oversampling however may cause the model to overfit the data. To reduce the probability of these issues we mentioned, we use Synthetic Minority Oversampling Technique (SMOTE) [4] to deal with imbalanced datasets. To generate synthetic samples, SMOTE uses a nearest neighbors algorithm to identify the k-nearest neighbors of each minority class sample. It then generates new synthetic samples by randomly selecting one of the k-nearest neighbors and combining it with the original sample in a way that lies on the line connecting the two samples. This process is repeated for each minority class sample, resulting in a larger dataset with a balanced distribution of classes.

2) Data Collection: Class weights refer to the relative importance assigned to each class in the dataset. When working with imbalanced datasets, class weights can be used to adjust the model's decision boundary in order to account for the unequal distribution of classes. For example, in a dataset with a severe class imbalance, the model may be more likely to predict the majority class simply because it occurs more frequently in the training data. By assigning higher class weights to the minority class, the model can be encouraged to pay more attention to the minority class and give it more consideration when making predictions. This can help improve the model's overall performance on the minority class, and can be particularly useful when working with imbalanced datasets where the minority class is the one of interest.

3) Validation Metrics: Validation metrics can be important in evaluating the model's performance. Standard metrics such as accuracy, precision, and recall may not be sufficient. For example, according to Li et al. [6], accuracy is not a good indicator for multi-classification on imbalanced data. A metric that doesn't take class-imbalance into account, e.g. gives equal weight to all classes regardless of their dominance, can provide more "real" or accurate results. Scikit-learn's Classification Reports have one such metric. The F1 score combines the results from both precision and recall, and the Classification Report includes a F1 macro-average metric. [5] Li et al. [6] suggests using Geometric Mean (GM) for imbalanced datasets.

4) Hyperparameter Tuning: It is important to consider the impact of hyperparameters on the model's ability to handle the imbalanced class distribution. Some hyperparameters that may be particularly important to consider when working with imbalanced datasets include class weights, loss functions and sampling techniques. Truic and Leordeanu [7] reports that using weights yielded good results with imbalanced data

B. Related Work for Poker Hand Recognition

This data set was first published by Robert Cattral and Franz Oppacher from Carleton University in 2007. This data set is relatively older than some other data sets so there are quite a few amounts of work done.

1) Poker Hand Classification: An attempt is made to solve poker hand classification problem using different learning paradigms and architectures of artificial neural network: multi-layer feed-forward Backpropagation (supervised) and self-organizing map (un-supervised). Poker data set is touted to be a difficult dataset for classification algorithms. Experimental results are presented to demonstrate the performance of the proposed system. The paper also aims to suggest about training algorithms and training parameters that must be chosen in order to solve poker hand classification problem using neural network model. As neural networks are the most convenient tools for handling complicated data sets with real values, one of the most important objectives of the paper is to explain how a neural network can also be used successfully for classification kind of problems involving categorical attributes. The proposed model succeeded in classification of poker hands with 94% classification accuracy. [8]

2) Poker Hand Induction: In this paper, they compare six well-established decision tree algorithms, two types of neural networks and three different machine learning algorithms provided by the sci-kit learning kit for multi-class classification. They summarise the top score we were able to achieve with each of these methods. In addition, they also provide a new poker hand data set that contains more samples and that is accurately depicting the real world. Lastly, they give our recommendations for future work of using extreme imbalance data sets. [9]

III. Method and Experimental Results

To train a classifier for predicting classes from imbalanced data, several key steps must be taken. First, we gather poker hand data and create a dataset that represents the entire population of combinations (unordered). Next, we select samples for training, validation, and testing that accurately reflect the population. The selected classifiers are then set up for multi-class classification. Finally, the classifiers are subjected to various methods for addressing imbalanced data, and hyperparameter tuning is performed with the use of data augmentation techniques.

- Collecting data from UCI. [10]
- Creating Train, Test and Validation sets
- Preprocessing and Feature Engineering (Data Augmentation).
- Using SMOTE and WSMOTE
- Model Implementation and Training

TABLE I
RANKS AND THE PROBABILITIES

Class	Poker Hands	Probability
0	156,304,800	0.5011773940345370
1	131,788,800	0.4225690276110440
2	14,826,240	0.0475390156062425
3	6,589,440	0.0211284513805522
4	1,224,000	0.0039246467817896
5	612,960	0.0019654015452335
6	449,280	0.0014405762304922
7	74,880	0.0002400960384154
8	4,320	0.0000138516945240
9	480	0.0000015390771693

TABLE II
RANKS AND THE PROBABILITIES

Predictors	Description	Values
S1	Suit of card #1	Ordinal
C1	Rank of card #1	Numerical
S2	Suit of card #2	Ordinal
C2	Rank of card #2	Numerical
S3	Suit of card #3	Ordinal
C3	Rank of card #3	Numerical
S4	Suit of card #4	Ordinal
C4	Rank of card #4	Numerical
S5	Suit of card #5	Ordinal
C5	Rank of card #5	Numerical
CLASS	Poker Hand	{0,1,2...9}

A. Data Collecting

In the dataset, there are 311,875,200 unique poker hand samples. Table 1 shows dataset information and statistics by each class. For data structure, each card in a given poker hand is denoted by a suit-value that can be between 1 and 4, and a rank-value that can be between 1 and 13. Each poker hand is assigned a CLASS (rank) label between 0 and 9 which indicates how good the hand is and what kind of hand it is. Table 2 lists each predictor and a few examples. Table 3 lists each class and their description.

If we were to create every possibility in the Poker game we had to face a 7.5 GB large .csv file and train it to our models. Which for a problem like this is very inefficient and redundant.

TABLE III
CLASS DESCRIPTION

Class	Description
0	Nothing in hand; not a recognised poker hand
1	One pair; one pair of equal ranks within five cards
2	Two pairs; two pairs of equal ranks within five cards
3	Three of a kind; three equal ranks within five cards
4	Straight; five cards, sequentially ranked with no gaps
5	Flush; five cards with the same suit
6	Full house; pair + different rank three of a kind
7	Four of a kind; four equal ranks within five cards
8	Straight flush; straight + flush
9	Royal flush; Ace, King, Queen, Jack, Ten + flush

B. Creating Train, Test and Validation Sets

We used the train and test sets that was found in the UCI for Poker Hand Data Set. There were 25,009 samples in the train data set while 999,999 samples in the test data set. Since it is a large data set in the overall of train and test sets, our test data set is bigger than our train data set. The distribution of classes between the train and the test set was relatively same so we didn't have to exchange any data for the rare classes. Using GridSearchCV, we decided to use 5 fold for our data so our raw data's validation set was approximately 20,000 samples while our augmented data set using SMOTE and WSMOTE's validation set was bigger.

C. Preprocessing and Feature Engineering

Our data was very unstructed in some ways, for example, straights should be in such order that the card differences should be in an ascending order but all of them was not in such way. After accomplishing some small work of data science to our data sets we trained a simple decision tree model received a really bad score on our classes. Not only on our test data but also on our train data. After analyzing the results of our simple model, we came to the conclusion that our features are highly uncorrelated due to lack of information. Changing the order of cards were not making any significant improvement. So we tried to generate new features for our data set. For pairs, sets and quads we created 10 new features representing the count of cardinals and the count of suits which gave us a significant improvement on deciding pairs, sets, quads and flushes. For straight rank, we generated 4 more features to keep track of different cardinalities in a hand to see if a straight is possible. Using these 4 new features we created a 4 more feature to make use of predicting the most rare combination, royal flush. Finally, we added a new feature to keep track of unique suit amount to help us check if there is a flush condition or not. Our values were varying around 0 to 13 giving the chance to not use normalization since it the values are in a narrow range. Also using normalization did not have an expressive improvement.

D. Data Augmentation (SMOTE and WSMOTE)

Empirically, using a training set with different numbers of examples from each class can result in a classifier that is biased towards the majority class. If the test set is also imbalanced in a similar way, the classifier may appear to have a higher accuracy than it actually does. To avoid this issue, it is important to use stratified sampling to ensure that the validation and testing sets accurately reflect the overall population. Making changes to these sets, especially the training set, can increase the risk of drawing incorrect conclusions about the results. We hypothesized that using larger training sets would increase the total training time for machine learning algorithms, especially when dealing with imbalanced datasets that require generating synthetic samples using SMOTE. Therefore, we wanted to investigate whether it is necessary to generate synthetic samples for the frequently represented minority classes. To test this, we developed a modified version of SMOTE called Weighted

TABLE IV
USING WEIGHTED SMOTE

Class	Training Set	Augmented Samples	Total
0	12493	1	12494
1	10599	1	10600
2	1206	10	1216
3	513	24	537
4	93	134	227
5	54	231	285
6	36	347	383
7	6	2048	2090
8	5	2500	2505
9	4	3126	3130

TABLE V
USING DEFAULT SMOTE

Class	Training Set	Augmented Samples	Total
0	12493	0	12493
1	10599	1894	12493
2	1206	11287	12493
3	513	11980	12493
4	93	12400	12493
5	54	12439	12493
6	36	12457	12493
7	6	12487	12493
8	5	12488	12493
9	4	12489	12493

SMOTE (WSMOTE) where we assign different weights to the synthetic samples generated for each class.

$$Samples(nc) = nc + floor((N/nc) * a) \tag{1}$$

We defined a function that takes in the number of samples belonging to a particular class (nc) and the total number of samples in the data set (N), and returns the new number of samples for that class using a parameter a, which is recommended to be between 0 and 1. We chose to use a value of 0.2 for each class. The new sample counts by class and the total number of samples in the augmented training set are shown in Table 4. We tried different values of a for the samples function and the best results were giving with the values from 0.2 to 0.4. Using a big a value was causing redundant sampling over the values smaller than 0.5.

In table 5 however, we tried to use the SMOTE without giving any weights to test the performance of our model on the data set. Augmented samples were 8,422 using the sample amount function that each are assigned to. This version of SMOTE took the majority class sample as a threshold and sampled to that threshold point.

E. Model Implementation and Training

1) XGBoost: Popular and efficient open-source implementation of the gradient boosting algorithm for machine learning. It is widely used in data science and has achieved state-of-the-art results on many machine learning benchmarks. Works by training a series of decision trees in a sequential manner, using optimization techniques to make the training process more efficient and regularization techniques to prevent overfitting. We mostly focused on learning rate, estimators, reg alpha and reg lambda values for this model. See Table 6 for more detail.

TABLE VI
XGBOOST HYPERPARAMETERS

Hyperparameter	Values
learning_rate	0.1
max_depth	10
min_child_weight	1
gamma	0
subsample	1
colsample_bytree	0.3
reg_alpha	0
reg_lambda	1

Learning rate is used to determine the step size that updates the weak learners. We started with default value and decreased it to 0.0001 and test our data set. The results were not significantly different, thus using GridSearchCV we came to the conclusion that the default value of learning rate was the best.

Reg alpha and reg lambda was other important hyperparameters to consider. While alpha was for L1 Regularization term, lambda is for L2 Regularization and these values are add penalty for large weights. As we mentioned above, this is a crucial step for imbalanced class problems. For the alpha value the default value did not change after tuning in the parameters but the lambda value did change to 1 giving 1 penalty resulting in a better results for rare classes.

We also took max depth into the consideration, but it didn't gave us any remarkable results. We decided to use 10 which was the best and reliable value.

We didn't use too much estimators for our XGBoost to reduce the time of training but used colsample by tree hyperparameter to determine the fraction of the features that will be used. This attribute was very helpful after oversampling our data helping us to get better results on rare classes. Raw data was poorly used with this hyperparameter due to lack of some classes.

2) CatBoost: A gradient boosting algorithm that is specifically designed to handle categorical features and missing values in the data. It uses decision trees as weak learners and applies various regularization techniques to prevent overfitting. Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

The main hyperparameters we were considering in CatBoost was depth, learning rate and iterations. We also tried some

TABLE VII
CATBOOST HYPERPARAMETERS

Hyperparameter	Values
learning_rate	0.029
depth	10
l2_leaf_reg	3
border_count	254
bagging_temperature	1
rsm	1

TABLE VIII
MLPCLASSIFIER HYPERPARAMETERS

Hyperparameter	Values
Hidden Layer Sizes	(50, 50, 50)
Learning Rate	constant
Activation	tanh
solver	adam
alpha	0.0001

other hyperparameters such as border count and bagging temperatures but changing the values of there hyperparameters were crucial and we observed overfitting in most of the values. See Table 7 for more detail.

Changing the depth of the weak learners was important in some way because the cause of overfitting. Since our data set was imbalanced we were very careful when changing these parameters. So, we tried different values and the best parameter for the depth was 10. While the default value for CatBoost was 6, it gave us a pretty good improvement not causing any overfitting issues.

The default value for iterations were very high compared to our values we used to tune our model. This can cause very complex models and may cause overfitting. After experimenting, we came to the conclusion to reduce the default iteration value can be a better choice. This decision was also effecting our learning rate tuning which we have to change the values.

3) Neural Network (MLPClassifier): MLPClassifier is a class in the scikitlearn library that implements a multi-layer perceptron (MLP) for classification. An MLP is a type of artificial neural network that is composed of multiple layers of interconnected nodes, or ”neurons.” Each layer receives input from the previous layer and passes it on to the next layer until the final layer, which produces the output.

Our main focus when using neural network is hidden layer sizes, activation function and some parameters such as solver, alpha and the learning rate. See Table 8 for more detail.

For hidden layer sizes we used 1, 2 and 3 hidden layers, to see the effect to our dataset. It was giving good results and reducing the chance of overfitting every time we increase the hidden layer size.

The most important hyperparameters for this model was the activation functions and the learning rate. We used tanh and relu activation functions and for the learning rate we used constant and adaptive rates. We trained models to test out these hyperparameters and see the results. Using tanh gave us a better result because of the more balanced range of outputs. This wide range gives the network more freedom to learn. This freedom that is provided from tanh, helps the network to identify classes easily. On the other hand learning rate was another aspect to tuning, it is stated that the two types of the learning rate should be used and see which on works the best on your problem. So, we used both of the learning rate types and get the best result from the constant type. Adaptive type of learning rate may not be suitable for this problem because of the class imbalance.

We also used the solver hyperparameter. We considered two solvers sgd and adam. The Adam solver is an adaptive optimization algorithm that combines the ideas of stochastic gradient descent (SGD) with momentum and an adaptive learning rate. he SGD solver is a variant of stochastic gradient descent that updates the model weights based on a randomly selected subset of the training data, rather than the entire data set. The Adam solver is very useful for very large data sets which in our case is the best choice. We used both of them unsuprisingly adam solver was better.

4) Random Forest Classifier: Popular machine learning algorithm that can be used for both classification and regression tasks. It is an ensemble method, which means that it combines the predictions of multiple individual models to make a final prediction. There are several parameters that can be adjusted when training a Random Forest classifier, such as the number of base models to include in the ensemble, the maximum depth of each decision tree, and the minimum number of samples required to split a node. Adjusting these parameters can influence the performance of the final model.

We used hyperparameter tuning with random forest classifiers and try to evaluate these classifiers. We used estimators, criterion, max depth and max features. See Table 9 for more details.

In random forest classifier, n estimators mean the number of base models that are in the ensemble. The value of n estimators will typically improve the performance of the model, but at the cost of increased computational time. We used fairly enough amount of estimators for the efficiency and the accuracy of our models. To find the best value we used different validation set using GridSearchCV.

The criterion parameter determines the function that is used to measure the quality of a split in the decision tree.

TABLE IX
MLPCLASSIFIER HYPERPARAMETERS

Hyperparameter	Values
n_estimators	10
criterion	gini
max_depth	30
max_features	log2

TABLE X
RAW DATA CLASSIFICATION REPORT RANDOM FOREST CLASSIFIER

Class	Precision	Recall	F1-Score	Amount of Samples
0	0.63	0.81	0.71	501208
1	0.57	0.49	0.53	422498
2	0.39	0.00	0.01	47622
3	0.54	0.00	0.01	21121
4	0.25	0.00	0.00	3885
5	1.00	0.00	0.01	1996
6	0.00	0.00	0.00	1424
7	0.00	0.00	0.00	230
8	0.00	0.00	0.00	12
9	0.00	0.00	0.00	3
accuracy			0.61	999999
macro avg	0.34	0.13	0.13	999999
weighted avg	0.59	0.61	0.58	999999

There are two types gini and entropy. We used both to tune this parameter. Gini gave us the best result due to it's efficiency but the accuracy was not very different from entropy.

The last hyperparameters we used for tuning random forest classifier is that max depth and max features. Max depth is for the limit of depth while max features determines the number of features that are randomly selected from the total number of features. Max depth can be specified as a number while max feature can be also be specified as strings. We used log2 max feature and log2 of number of features gave us the best result. Recalling other models we trained once again the max depth gave us the best result for the value of 30.

EXPERIMENTAL RESULTS AND METRICS

Even though, to solve this problem there should not be any need for machine learning algorithms. Our main aim was to solve the imbalance problem and identify the solutions. [9]

In our work we trained models with raw data and also preprocessed and feature engineered data to test the effect of it to imbalance problems. We think that just showing accuracy of our models in very insufficient way to evaluate an imbalanced data set, as stated in. [5]

Our models gave quite good accuracy on our data. It did give a very poor result on raw data. We think giving one example results of a model is enough to comprehend the importance of preprocessing and Feature Engineering. As shown in Table 10, the raw data used with random forest classifier is very unpleasing.

TABLE XI
CATBOOST CLASSIFICATION REPORT ON WSMOTE

Class	Precision	Recall	F1-Score	Amount of Samples
0	1.00	1.00	1.00	501208
1	1.00	1.00	1.00	422498
2	1.00	1.00	1.00	47622
3	1.00	1.00	1.00	21121
4	1.00	1.00	1.00	3885
5	1.00	1.00	1.00	1996
6	1.00	1.00	1.00	1424
7	1.00	1.00	1.00	230
8	0.92	1.00	0.96	12
9	0.33	1.00	0.50	3
accuracy			0.61	999999
macro avg	0.34	0.13	0.13	999999
weighted avg	0.59	0.61	0.58	999999

TABLE XII
CATBOOST CLASSIFICATION REPORT ON SMOTE

Class	Precision	Recall	F1-Score	Amount of Samples
0	1.00	1.00	1.00	501208
1	1.00	1.00	1.00	422498
2	1.00	1.00	1.00	47622
3	1.00	1.00	1.00	21121
4	1.00	1.00	1.00	3885
5	1.00	0.98	0.99	1996
6	0.96	1.00	0.98	1424
7	1.00	0.76	0.86	230
8	0.55	1.00	0.71	12
9	0.13	1.00	0.23	3
accuracy			1.00	999999
macro avg	0.86	0.97	0.88	999999
weighted avg	1.00	1.00	1.00	999999

After evaluating all the model results, we came to the conclusion that the best results were CatBoost. As shown in Table 11. We obtained really remarkable results using CatBoost but some of the rare classes were still not good as others but relatively acceptable due to imabalancity. These results in Table 11 were made with oversampled data using Weighted SMOTE and a help of a saples function.

Using Weighted SMOTE was one way of approaching to the solution for data augmentation. What we did after this was we used SMOTE once again but this time without the weights and sample amount. This type of SMOTE object created samples based on the amount of the majority class. As shown in table 12, this approach was not a good idea because it caused to overfit the data. The reason is that even though there isn't that much of a combination of some rare classes we tried to oversample these data which is unnecessary and has no real life explanation. For example, for class 9 there is at most 480 combinations but if you use SMOTE with this sampling strategy it will create way more than 480 samples and cause the model to overfit. We discovered overfitting when we compared our test and train results on the model.

Before we end this section, we want to show how our preprocessing and feature engineering effected our models by showing the feature importance of features with our CatBoost

TABLE XIII
TOP 10 IMPORTANT FEATURES FOR CATBOOST

Feature Names
diff_4
cnt_s2
diff_2
cnt_s3
cnt_s5
diff_3
cnt_c1
C5
cnt_s4
cnt_c4
cnt_c5
cnt_c3

model. See Table 13 for more details. The features we created are highly important when predicting the classes for a model. This shows that our feature engineering gave us a significant amount of improvement when solving the class imbalance in domain-specific problems.

IV. DISCUSSION

In this project we have explored several methods to overcome class imbalance problem using Poker Hand data set. We discovered that this data set is very imbalance because of the nature of the game.

Other than researches that were done in this data set. We used feature engineering to help our models interpret the data in a more convenient way. Creating extra features and preprocessing really helped our models. This approach really improved our results. Feature work such as sorting some of the rows in our data, creating difference columns increased the accuracy of our models by almost %80.

Using data augmentation was also a good technique. Some researches have been done using smote but we used different variations of SMOTE to understand the effect of it to the Poker Data set. Which showed us the upsides and the downsides of sampling. This paper will give a good idea to people who want to use SMOTE in various ways and expect the outcome.

Although using these methods mentioned and the methods that have not been covered or discovered, we have come to the conclusion that gathering and collecting more data is one of the best way to overcome imbalance data set problem.

In our project, we placed a strong emphasis on testing and runtime efficiency. We conducted more tests on CatBoost than any other boosting method because it can utilize the GPU, which significantly speeds up the training and testing process compared to using a single CPU core. However, it is possible that performing a similar number of tests on the other methods could have produced even better results. We also feel that we could have conducted more tests on Neural Networks, given that previous research on the poker hand dataset has shown a range of different parameters for the number of hidden layers and neurons. Our limited testing of three to five different parameters may not have been sufficient to determine the best performance for poker hand using this method.

V. CONCLUSION

First we gathered our data from UCI [**?**] and tried to collect more data online to overcome the imbalance problem. After gathering enough data we analyzed the data and found some inconsistency as mentioned in the Preprocessing and Feature Engineering section. Based on our analysis, we generated new features and did changes to our data by sorting some rows. Getting the data ready, we also used SMOTE and WSMOTE copying our data. We used sample function for determining how many samples should be sampled for WSMOTE. We trained models with our raw data, oversampled Data and weighted oversampled data. After obtaining the desired versions of our data, we trained models and use hyperparameter tuning to get the best results. Using classification reports on our classes we used different versions of our data that is mentioned above for the models. The results, as expected, was better than raw data and we analyzed best models to see the feature importance. Feature importance was a good metric for our data processing and feature engineering. Seeing enriched results were a good indicator to see our work being successful.

REFERENCES

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Systems with Applications 73 (2017) 220–239.

[2] S. Shilaskar, A. Ghatol, P. Chatur, Medical decision support system for extremely imbalanced datasets, Information Sciences 384 (2017) 205–219.

[3] K. Oh, J.-Y. Jung, B. Kim, Imbalanced classification of manufacturing quality conditions using cost-sensitive decision tree ensembles AU - Kim, Aekyung, International Journal of Computer Integrated Manufacturing 31 (2018) 701–717.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, Journal Of Artificial Intelligence Research (2011).

[5] https://towardsdatascience.com/a-good-machine-learning-classifiers-accuracy-metric-for-the-poker-hand-dataset-44cc3456b66d

[6] F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, Y.-C. Tian, Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets, Information Sciences 422 (2018) 242–256.

[7] C.-O. Truic, A. A. Leordeanu, Classification of an Imbalanced Data Set Using Decision Tree Algorithms, U.P.B. Sci. Bull., Series C 79 (2017).

[8] S. Jabin, "Poker hand classification," 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016, pp. 269-273, doi: 10.1109/CCAA.2016.7813761.

[9] Thomas Angeland, Kim-Andre Engebretsen, Mustafa Numic; Poker Hand Induction: Multi-Class Classification of Extreme Imbalanced Data with Machine Learning, 2019

[10] https://archive.ics.uci.edu/ml/datasets/Poker+Hand