Data and Feature Engineering Challenges in Machine Learning

Kendall E. Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: kendall.nygard@ndsu.edu

Mostofa Ahsan, Aakanksha Rastogi, Rashmi Satyal
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: {mostofa.ahsan, aakanksha.rastogi, rashmi.satyal}@ndsu.edu

Abstract— **The process of developing of a machine learning application presents multiple challenges relating to data and their features. Based on experiences with several applied studies carried out using machine learning methodologies, we report on addressing challenges in the collection, quantity, distribution, quality, sampling, of and relevancy of data. We also address feature engineering and selection issues, including approaches to identifying, combining, and eliminating attributes and features that are not needed or of low significance. We include insight into overfitting and underfitting training data. Example applications include classification, anti-autonomy and trust modeling and analytics for self-driving cars and intrusion detection systems aimed at detecting malicious activity.**

Keywords- Machine Learning, Data Management, Feature Engineering, Self-Driving Cars, Intrusion Detection.

I. INTRODUCTION

Machine Learning (ML) is a rapidly emerging area of artificial intelligence. Many types of applications have been successfully developed and new successes are regularly reported. The famous Turing award winner Jim Gray referred to data science as a "fourth paradigm," taking a rightful place among empirical, theoretical, and computational sciences [1].

Often viewed as interdisciplinary, data science involves mathematics, statistics and computer science as well as other related areas. In many applications, the availability of large and relevant datasets, and the methods of data science provide the lifeblood of machine learning problem-solving approaches. Analyses and decision support in nearly every area of human endeavor today are related to machine learning.

The example machine learning studies that we describe are in the areas of self-driving cars and intrusion detection [2], [33], [36], [37].

In the case of the self-driving car study, there was availability of multi-attribute data about specific collisions. The data contained a host of features and attributes concerning the vehicle itself, the damage incurred, roadway conditions, etc. The objective of the study was to build a classification model that could translate the detailed data into collision predictions and to drive an anti-autonomy trust model. There were important and difficult choices made related to scale and balance within the available dataset, and in feature engineering. A linear sequential supervised learning machine learning model was employed.

The intrusion detection study used supervised learning techniques to build a model for identifying outside threats initiated by malicious actors who wish to breach or compromise a system. Among other datasets, the study examined the famous dataset that originated in the KDD (Knowledge Discovery and Data Mining) competition and was later modified to form the now publicly available NSL (Network Security Laboratory) KDD dataset [6], [7].

The rest of the paper is structured as follows. In Section II, we describe supervised machine learning with illustrations of the flow of a machine learning model and data splits for cross validation. In Section III, we present a self-driving cars example illustrating an implementation of a linear sequential supervised learning artificial neural network model utilizing multiple pre-processed complex attributes. In Section IV, we present the intrusion detection example, explaining how a machine learning model can be tuned to predict and identify attacks. In Section V, we describe data management and feature engineering issues that are ubiquitous in machine learning practice. This section also includes several categorical encoding techniques for preprocessing data for a machine learning algorithm. Finally, we conclude our work in Section VI.

II. SUPERVISED MACHINE LEARNING

We restrict our discussion to supervised learning models. Fundamentally, such models are well-suited to address applications for which there are available datasets whose data instances have a known classification label or target. The initial task is to computationally train the machine learning model to accept the data instances as input and to produce the correct target as output. Once trained, the model is available to accept new data and predict their target classification. The model is successful if it has high values of performance measures such as percentage of accuracy in correctly classifying the new data instances, called the ability to generalize. There are multiple issues surrounding the characteristics of the available data, the classes into which they fall, their attributes and features, and the learning models charged with producing the predictions. Concerning baseball, Coach Yogi Berra famously said, "It's tough to make

 ISBN: 978-1-61208-990-4

predictions, especially about the future." This aphorism is equally true in machine learning [45].

Figure 1 illustrates the general flow of a machine learning technique. Several tasks are included. The overall task of the DEVELOP phase shown at the top is to produce a Final Model that is fully specified, trained and feeds into the PREDICT phase shown below the dotted line, where it is available for generalization use on new data. Starting from the top, the data is shown as partitioned into splits for Training, Validation and Test. The full data is divided into the Training Split and the Test Split. A good way to perform training is to withhold a portion of the data while training is done. It is viewed as a mistake to train and test a machine learning model on the same data. So doing would result in the model memorizing all of the data/target pairs, resulting in the model perfectly knowing all of the answers, leaving no ability to generalize. The result is known as *overfitting*. For validation purposes, the Training Split is typically divided into pieces called folds. Called k-fold cross validation, Figure 2 illustrates basic logic for splitting the data. In this example, k=5 so there are five equal parts. This corresponds to the Validation Split and Model Tuning blocks in Figure 1. In Figure 2, shown in bold italics on the diagonal, there is a designated fold in each row that is specified for testing, with the other four used for training. The key idea is to find the best set of meta-level parameters for a model being developed. All of the major machine learning models have parameters. For example, an ML that utilizes an Artificial Neural Network (ANN) in some way, will be parameterized with settings like Learning Rate (governs weight adjustments), topology (number of hidden layers, nodes within layers, and interconnectedness), and activation functions. Other ML methods, like a Support Vector Machine (SVM) or Logistic Regression also have parameters. When a model is trained on the folds, a performance metric, such as classification accuracy, can be calculated on the testing fold. After all of the fold splits are evaluated in this way, an average is calculated, which yields a score for the parameter settings. Various optimization methods can be employed to explore the parameter space in a quest to identify the best settings. Viewed more generally, the Model Tuning block can also be viewed as exploring various types of models in a quest to not only optimize the use of one type of model, but to also choose among competing models.

In multiple places of the ML process, there is a need to evaluate the quality of the predictions using a metric. The empirical accuracy of a method is simply the percentage of the predictions made that are correct. Other metrics are available. More details are provided later in this paper.

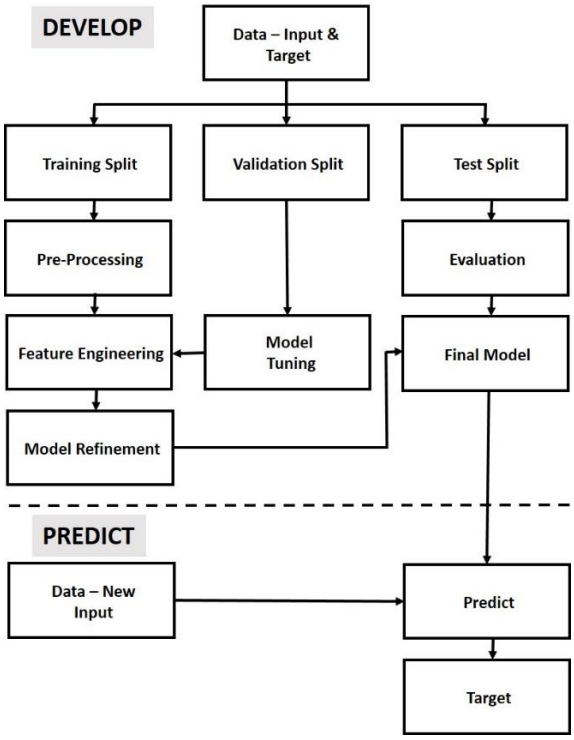

Figure 1. Flow of a Machine Learning Model [2]

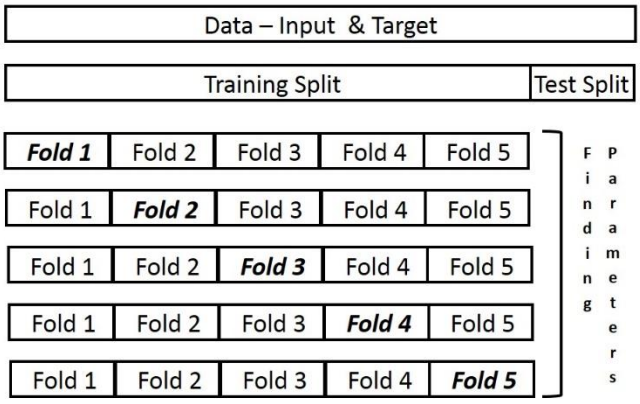

Figure 2. Data Splits for Cross Validation

Raw data is rarely available in a form that is suitable for direct use by an ML model. Pre-processing of data is typically necessary to deal with null or missing values, outliers, transforming or reconciling numeric and categorical values, rescaling, standardizing, etc. We expand on the data-centric issues for the example applications reported in this article.

Feature Engineering also appears in Figure 1. Features are those characteristics that are present in the data that are potentially useful in predicting a target outcome. It is often effective in ML to modify or combine features in some way to produce a new feature that can improve the prediction accuracy of the method. Called Feature Engineering, the operations that can be carried include things like mathematically transforming a single feature or applying a

 ISBN: 978-1-61208-990-4

functional calculation on multiple features. Feature Selection refers to reducing the number of features employed by the model while retaining acceptable results. Reducing the features needed can ease the data collection task and reduce the computational load of running the model. Feature Selection typically follows Feature Engineering. We provide details related to the examples discussed in this paper. Unsupervised Learning is different than supervised learning in many ways. Some of the most known algorithms are, k-means clustering, hierarchical clustering, principal component analysis, and apriori algorithm [44]. The need for unsupervised models is increasing in the cybersecurity domain since attacks are being modified every day [46].

III. SELF-DRIVING CARS EXAMPLE

For an application to self-driving cars, we used available data to study anti-autonomy traits and factors responsible for collisions and diminished trust [2], [38]. Data availability was a challenge since jurisdictions of different states, federal traffic agencies and motor vehicle departments often do not make their data publicly available. Data used in this study was submitted by the manufacturers of autonomous vehicles to the California Department of Motor Vehicles for collisions that occurred with other cars, pedestrians, bicyclists, and other objects during their test drives on roads and freeways in the state. All data applied to collisions that occurred while the cars were being driven in autonomous driving mode. The collisions occurred between October 2014 and March 2020 [2], [38]. The attributes of this dataset are listed in Table I below. All attribute names are feature type categorical and data type object.

TABLE I. COLLISION DATA ATTRIBUTES [2]

Attribute Type	Attribute Names
Autonomous vehicle details	Manufacturer Name, Business Name, Vehicle Year, Vehicle Make, Vehicle Model, Vehicle was (stopped in traffic/moving)
Accident Details	Date of Accident, Time of Accident
Involved in Autonomous vehicle accident	Involved in Autonomous Vehicle Accident (Pedestrian/Bicyclist/Other), Number of vehicles involved with Autonomous Vehicle
Autonomous vehicle damage	Vehicle Damage, Damaged Area
Details of other vehicle involved in accident	Vehicle 2 Year, Vehicle 2 Make, Vehicle 2 Model, Vehicle 2 was (stopped in traffic/moving)
Involved in Other vehicle accident	Involved in Vehicle 2 Accident Pedestrian, Involved in Vehicle 2 Accident Bicyclist, Involved in Vehicle 2 Accident Other, Number of vehicles involved with Vehicle 2
Injuries	Injured, Injured Driver, Injured Passenger, Injured Bicyclist
Vehicle driving mode	Vehicle Driving Mode
Weather conditions for both vehicles	Clear, Cloudy, Raining, Snowing, Fog/Visibility, Other, Wind
Lighting conditions for both vehicles	Daylight, Dusk-Dawn, Dark Street Lights, Dark-No Street Lights, Dark-Street Lights Not Functioning
Roadway surface for both vehicles	Dry, Wet, Snowy-Icy, Slippery/Muddy/Oily/etc., Holes-Deep-Rut, Loose Material on Roadway, Obstruction on Roadway, Construction/Repair Zone, Reduced Roadway Width, Flooded, Other, No Unusual Conditions
Preceding Movement of Autonomous Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing Into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Preceding Movement of Other Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing Into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Type of Collision	Head On, Side Swipe, Rear End, Broadside, Hit Object, Overturned, Vehicle/Pedestrian, Other
Other	CVC Sections Violated Cited, Vision Obscurement, Inattention, Stop and Go Traffic, Entering/Leaving Ramp, Previous Collision, Unfamiliar With Road, Defective WEH Equip Cited, Uninvolved Vehicle, Other, None Apparent, Runaway Vehicle

This data was extracted from PDF files and converted into CSV format. Data cleaning was a significant effort, and included pre-processing steps for augmenting, labeling, and classifying the data [2], [38]. The core purpose of the study was to associate conditions into a level of trust that people had in a self-driving car. Anti-autonomy refers to decisions and actions taken by a self-driving car that are in some way inappropriate in terms of increasing risk, diminishing safety, or lowering trust. The values of attributes in the data that describe conditions and circumstances that are present when a collision occurs provide a handle to model a mapping between data and trust level. After pre-processing the data, a linear sequential supervised learning ANN model called NoTrust was devised, validated, and tested to classify the target data, using the basic approach illustrated in Figure 1.

The model used the libraries provided by Keras with the Tensorflow backend [39] -[41]. Python was used for programming since it integrates with Keras to access the

neural network Application Programming Interface. The deep learning libraries of Keras support fast prototyping, modularity, and smooth computation.

There are multiple challenges concerning data, features, and metrics in applying a ML methodology to the self-driving car application. First, there were only 256 collision reports available, which is arguably a small number to use in a ML method. Also, in the context of alternative target value possibilities, the data is somewhat unbalanced in that the number of samples across the distinct classes differs. Section V describes methods, such as oversampling, to deal with unbalanced data. Second, there are 140 attributes, a large number relative to sample size, as shown in Table I. Thus, the possible permutations and combinations that could be evaluated in the ANN model is explosive. Fortunately, with initial analyses of the data and evaluation runs, it was possible to identify a subset of attributes and features which revealed that they were mandatory to include. The five attributes shown below form the mandatory set.

- Vehicle driving mode = autonomous
- Vehicle damage = moderate and major
- First vehicle involved = Pedestrians/Bicycle/Other
- Second vehicle involved = Bicycle/Other
- Injuries sustained = Pedestrians/Bicyclists/Others

While keeping the model simple and still retaining accuracy, the mandatory feature set performed well in making trust and do not trust predictions for autonomous vehicles. However, when anti-autonomous traits of the self-driving car itself were incorporated into the model it became apparent that more attributes had to be utilized. These include vehicle driving mode, type of collision, weather conditions, roadway surface conditions and injuries sustained by pedestrian/bicyclists/others. In addition to the linear sequential ANN, evaluation of Recurrent Neural Networks (RNN) models with Long Short-Term Memory (LSTM) were available for possible comparison purposes. When the additional attributes are included, along with measures of severity of damages sustained by vehicle, the imbalance of the data increased. More specifically, the larger number of predictors added more noise, redundancies, increased overfitting, and decreased the quality of the predictions. A related study by Meiri and Zahavi [3] used simulated annealing to search the attribute space.

Combinatorial problems often have issues related to model accuracy, performance, and optimizer bias. Also, the model solutions offered by machine learning include approximation errors which is further exacerbated by the fact that the input configurations to the ML model can be significantly different between training and validation [4]. This can be solved by two approaches – active learning and passive learning. Active learning involves updating the model itself to assure a convergence between training and validation curves in turn improving model accuracy and optimization bias. Passive learning involves the training set providing a uniform and sufficient coverage of the search space [4]. In a similar context, Charikar et al. [5] defined and studied combinatorial feature selection problems and presented a theoretical framework which provided algorithms on approximation and hardness results of these combinatorial problems [5].

IV. INTRUSION DETECTION EXAMPLE

In today's world of connected devices, security of the network is of critical importance. Unauthorized access and malicious activities are a great threat to confidentially, integrity, and availability that form the information security triad. The role of an Intrusion Detection System (IDS) is to detect abnormalities caused by an unauthorized reach into the network and send alerts. An IDS is an element of support for a wall of defense between cyber-attacks. Supervised ML techniques in an IDS can provide high detection accuracy, particularly against known types of attacks.

The NSL-KDD is an update and improvement to the KDD'99 dataset that that was developed for the KDD Cup competition in 1999 [6]. These datasets are publicly available and are very widely used for IDS experiments. The data is primarily internet traffic consisting of 43 features per record, of which the last two are *class* (attack or normal) and *score* (severity of traffic input) [7]. The *class* column provides information on whether the record is considered normal or is a member of one of four attack classes - Denial of Service (DoS), Probe, Remote-to-Local (R2L) or User-to-Root (U2R). There are14 attack types under these 4 classes: Apache2, Smurf, Neptune, Back, Teardrop, Pod, Land, Mailbomb, Processtable, UDPstorm, WarezClient, Guess_Password, WarezMaster, Imap, Ftp_write, Named, Multihop, Phf, Spy, Sendmail, SmpGetAttack, AnmpGuess, Worm, Xsnoop, Xlock, Buffer_Overflow, Httptuned, Rootkit, LoadModule, Perl, Xterm, Ps, SQLattack, Satan, Saint, Ipsweep, Portsweep, Nmap, Mscan [42], [43]. A mixture of categorical (nominal), binary and numeric variables are in the feature set. Each record has basic, content-related, time-related, and host-based features [8]. The attributes of this dataset are listed in Table II.

TABLE II. NSL-KDD DATASET ATTRIBUTES [8]

Attribute Type	Attribute Names
Basic	Duration, Protocol_type, Service, Flag, Src_bytes, Dst_bytes, Land, Wrong_fragment, Urgent
Content related	Hot, Num_failed_logins, Logged_in, Num_compromised, Root_shell, Su_attempted, Num_root, Num_file_creations, Num_shells, Num_access_files, Num_outbound_cmds, Is_hot_login, Is_guest_login
Time related	Count, Srv_count, Serror_rate, Srv_serror_rate, Rerror_rate, Srv_rerror_rate, Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate
Host based traffic	Dst_host_count, Dst_host_srv_count, Dst_host_same_srv_rate, Dst_host_diff_srv_rate, Dst_host_same_src_port_rate, Dst_host_srv_diff_host_rate, Dst_host_serror_rate,

	Dst_host_srv_serror_rate, Dst_host_rerror_rate, Dst_host_srv_rerror_rate

The study also used the UNSW-NB15 dataset. This dataset has 49 features categorized into 6 groups: basic, flow, time, content, labelled and additional generated features [9]. There are 9 attack types: fuzzers, analysis, back-doors, DoS, exploits, generic, reconnaissance, shell code and worms [10]. This dataset has a mixture of categorical, binary, and numerical datatypes. The attributes of this dataset are listed in Table III below.

TABLE III. UNSW-NB15 DATASET ATTRIBUTES [15]

Attribute Type	Attribute Names
Basic	state, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, service, sload, dload, spkts, dpkts
Flow	srcip, sport, dstip, dsport, proto
Content	swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len
Time	sjit, djit, stime, ltime, sintpkt, dintpkt, tcprtt, synack, ackdat
Additional generated (general purpose)	is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd
Additional generated (connection)	ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm
Labelled	attack_cat, attack_cat

The target attribute either identifies records as ‘normal’ or ‘attack’ or distinguishes the record as a particular attack type. Depending on the desired goal of an intrusion detection system, the machine learning model is tuned to identify a particular attack, which is a challenge in itself. It is thus essential to understand the requirement thoroughly and preprocess input data accordingly.

As an illustration of evaluation metrics, at a high level in the IDS study, for each input vector we have exactly one of the following outcomes:

TP = True Positive = Correct predication that the input vector is an Attack

TN = True Negative = Correct prediction that the input vector is not an Attack

FN = False Negative = Incorrect prediction the input vector is not an Attack

FP = False Positive = Incorrect prediction the input vector is an Attack

The most widely reported metric is Basic Accuracy of the model, which simply reports the proportion of attack reports that are correct.

$$\text{Accuracy} = (TP + TN)/(TP = TN = FP + FN)$$

The Basic Accuracy is notoriously deceptive when the classes are unbalanced, as in the case of intrusion detection studies, where most input vectors are not attacks. False reports are of interest. This gives rise to the need for metrics such as Precision and Recall, which can be calculated from information in the confusion matrix given below.

	Prediction		
Actual		Attack	Not an Attack
	Attack	TP	FN
	Not an Attack	FP	TN

$$\text{Precision} = TP/(TP + FP)$$

$$\text{Recall} = TP/(TP + FN)$$

Precision measures the proportion of the vectors reported by the IDS as attacks that are real attacks. Recall measures the proportion of the vectors that are real attacks and do get reported as such. This means that when Recall is high the IDS does not misclassify many true attacks.

In Intrusion Detection applications, false negatives can be very deadly, which favors high Recall. However, dealing with false positives also has a cost. Unfortunately, experiments that improve Precision typically reduce Recall. The reverse is also often true. For this reason, the harmonic mean of the two, called the F1 score is often calculated.

$$F1 = (2*(\text{Precision} * \text{Recall}))/(\text{Precision} + \text{Recall}).$$

The effect of the F1 score, which falls between 0 and 1, is to punish extreme values.

V. METHODOLOGIES FOR ADDRESSING DATA AND FEATURE ENGINEERING ISSUES

We provide detailed descriptions data management and feature engineering issues that are pervasive in ML practice and were of importance in our applied studies.

A. Data Management

Class imbalance in a dataset means that the relative numbers of instances within the classes vary significantly in number [16]. The magnitude of the discrepancies will also vary. Class imbalance is common in most important data domains, including detection of things like fraudulent activities, anomalies, oil spills, and in medical diagnoses. The imbalance of classes occurs in both binary class and multi-class classification [17]. In binary classes, the smaller and larger cardinality classes are called minority and majority classes respectively [16], [18]. Class imbalance can influence the training process of ML techniques and lead to ignoring the minority class entirely. We discuss some of the approaches to treat class imbalances. Figures 3 – 9 illustrate the results of applying each technique.

Random oversampling of a minority class. In this approach data instances in minority classes are duplicated at random to induce a balance of membership between classes. Due to randomness of the oversampling, the method is naïve in that

it makes no assumptions about the classes and their members [19], [20]. Since exact copies of some data instances are included in training, there is a risk of overfitting. Classifier accuracy may also be influenced, and computational effort may be increased.

Random undersampling of a majority class. This approach discards data instances from majority classes to induce balancing [21]. As in the case of the oversampling method, the discarded data is chosen randomly and naively. The method applies to both binary and multiclass data. The approach can make it difficult to distinguish boundaries between classes, with an inimical impact on performance measures [22].

Synthetic Minority Oversampling Technique (SMOTE). This technique was introduced in 2002 to address the shortcomings of the oversampling and undersampling approaches [23], [24]. The technique generates synthetic data by calculating feature space similarities between minority class data instances. The K-nearest neighbors of each data instance in a minority class are calculated, then randomly selected one by one. The method then calculates linear interpolations among the data and uses them to create synthetic data instances.

Borderline SMOTE. The SMOTE approach encounters issues when minority class data instances occur in the vicinity of majority class data instances, creating undesirable bridges. The Borderline SMOTE variation addresses this drawback by oversampling only minority class instances near the borderline. Data points are called border points if they are incident to both minority and majority classes and called noise points otherwise [25]. Border points are then utilized to balance the data between classes.

K-Means SMOTE. This technique generates minority class samples in safe and crucial borders of input spaces and thus assists performance in classification. The method begins by clustering the dataset using the K-means procedure, then selects the clusters that have higher numbers of minority samples [26]. Additional synthetic samples are then assigned to clusters where minority class samples are sparsely distributed. No noise points are generated.

SVM SMOTE. A variation of Borderline-SMOTE, the method finds misclassification points. The borderline points are approximated and classified with a Support Vector Machine (SVM) classifier [27]. Synthetic data points are created randomly around the lines joining each minority class support vector with its neighbors.

Adaptive Synthetic Sampling – ADASYN. A limitation of Borderline SMOTE is that is utilizes only synthetic points generated from extreme observations and the borderline instances and neglects the rest of the points in minority classes. This issue is addressed by ADASYN by creating the synthetic data using the density of the existing data [28]. The ratio of synthetically generated data is created in inverse relation to the minority class density. In this way, a less dense area creates more synthetic data.

The Churn Modeling Data from Kaggle was applied to the methods [29]. Figure 3 shows the distribution of the data in the original classes, followed by the outcomes of the alternative methods in Figures 4 to 9.

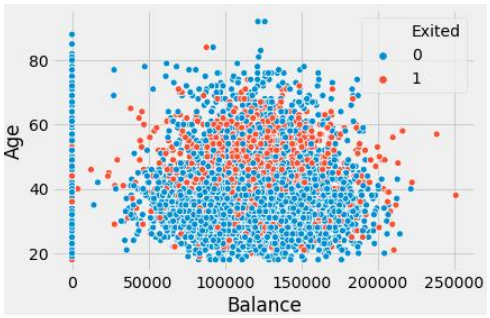

Figure 3. Original Class Imbalance Illustration

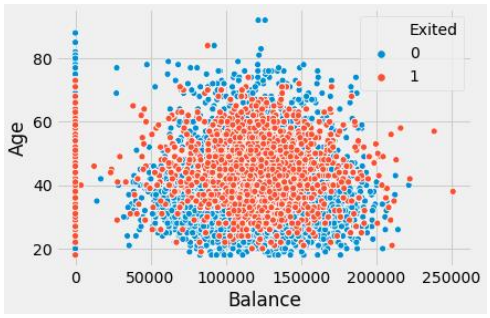

Figure 4. Outcome of Random Oversampling

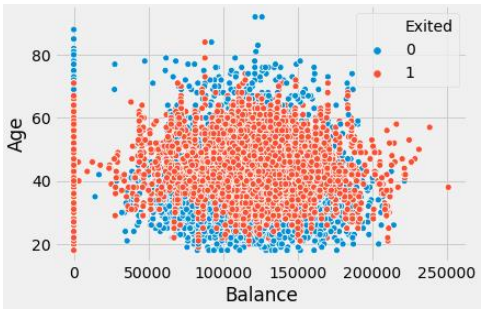

Figure 5. Outcome of SMOTE

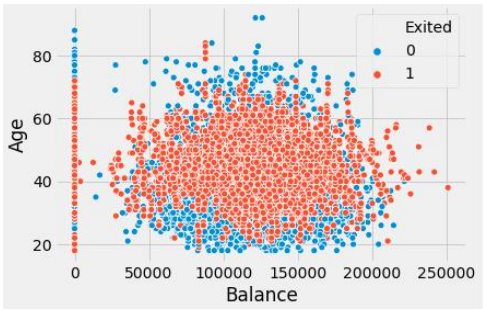

Figure 6. Outcome of Borderline-SMOTE

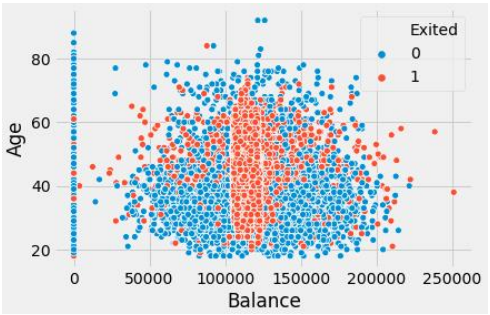

Figure 7. Outcome of K-Means SMOTE.

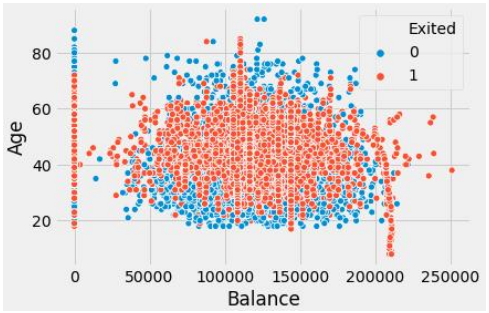

Figure 8. Outcome of SVM SMOTE

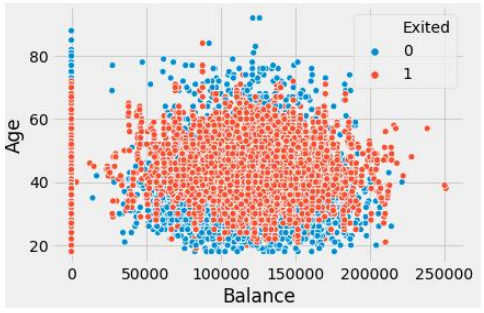

Figure 9. Outcome of ADASYN

B. Data Management and Feature Engineering

There are multiple methods for feature engineering on categorical data. The inputs to ML algorithms must be numeric, but many applications have categorical data. In our work with ML methods for self-driving car collisions there are examples of ordinal categorical data, such as rating of severity of damages or a weather condition. The intrusion detection work examples include counts of file access attempts, session duration, or error rates. There are also examples of nominal data, such as a type of vehicle in our self-driving car work, or whether or not a flag is set in the intrusion detection work. Various encoding methods are used to convert the variables into a useful numerical representation [9]. Choosing an appropriate encoding scheme is an essential part of data preprocessing for a ML algorithm. Some of the categorical encoding techniques are described below.

a) One-hot encoding

This method converts an attribute with N possible categories into N distinct features. In the NSL-KDD dataset, the *protocol type* attribute has 3 possible values - Internet Message Control Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). One-hot encoding converts this attribute into three feature columns as shown in Figure 10. It follows a 0/1 representation to indicate presence (indicated by 1) or absence (indicated by 0) of a value.

protocol_type
icmp
udp
tcp
tcp
udp

⇒

icmp	udp	tcp
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0

Figure 10. One-hot encoding used in Protocol Type attribute

b) Dummy coding

Like one-hot encoding, dummy coding converts an attribute with N values to a feature set of N-1 values. The converted set of binary variables are called dummy variables. Figure 11 illustrates one-hot ending and dummy coding applied to the same set of categorical records.

Column	Code
A	100
B	010
C	001

One- Hot Coding

Column	Code
A	10
B	01
C	00

Dummy Code

Figure 11. One-hot and dummy encoding used in the same dataset [11]

c) Effect coding

While one-hot encoding and dummy coding use only 0 and 1 to encode categorical variables, effect coding sets values that sum to zero in the new feature set. As a result, negative values may also be generated in the encoded feature

set. Effect coding is a preferred choice when there is an interaction of categorical variables in a dataset as it can provide reasonable estimates of main effect and of the interaction [12].

d) Hash encoding

Hash encoding is appropriately used for categorical variables that have a large number of possible values. The method uses a hash function to map categorical values into numbers. Commonly used hash functions include Message Digest functions MD2, MD4, MD5 and Secure Hash Algorithms SHA0, SHA1, SHA2 and SHA3. The MD5 hash function is used by default [11]. Hash encoding returns a variable map with smaller dimension than other encoding schemes, such as one-hot encoding or dummy coding. Figure 12 below shows the hash-encoding process:

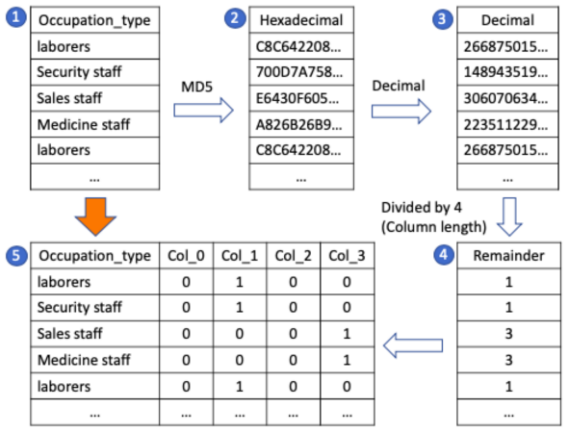

Figure 12. Hash Encoding Process [13]

e) BaseN encoding

BaseN encoding converts categorical variables into a consistently encoded feature set using a selected base, such as base 2 for binary encoding. The base or radix is the available number that can be used in different combinations to represent all values in a numbering system. A BaseN encoder encodes categorical values into arrays of their base-n representation.

f) Target encoding

Target encoding (also known as mean encoding) replaces a variable with a mean of the target value for that variable. Figure 13 provides an illustration. When the values of a categorical variable are already of a high volume, target encoding provides an advantage over other methods as it does not add extra dimensions to the dataset.

	feature	feature_label	feature_mean	target
0	Moscow	1	0.4	0
1	Moscow	1	0.4	1
2	Moscow	1	0.4	1
3	Moscow	1	0.4	0
4	Moscow	1	0.4	0
5	Tver	2	0.8	1
6	Tver	2	0.8	1
7	Tver	2	0.8	1
8	Tver	2	0.8	0

Figure 13. Target Encoding [14]

g) Label or ordinal encoding scheme

Ordinal categorical variables require that the order of the variable be preserved. For example, a road surface when a collision occurs might be categorized into dry, somewhat wet, or very wet so that the 3 values have an order that might provide additional insights. Ordinal encoding scheme aims at preserving this order when mapping values to numeric form. The method simply assigns each label a number (for example dry=1, somewhat wet=2 and very wet=3).

C. Feature Selection

The complexity of ML models increases with the dimensionality of the dataset. Predictive models often fail to achieve high accuracy because of inadequate analyses directed to feature selection. Selecting the most important and significant features reduces the complexity of the model and can also increase the prediction performance [36], [37]. Multiple approaches are available and effective for reducing the feature set. Prominent ones are described below.

Filter Methods. In our work, we choose feature selection methods that apply to situations with a categorical output, such as whether an input vector is an attack or not. The filter methods eliminate features independently of the ML method used. A univariate feature filter evaluates the importance of single features using univariate statistical tests. Each feature is paired with the target to evaluate statistical significance between them. The analysis of variance or ANOVA F-test is widely used. The F-test calculates the ratio between variance values [30]. The resultant measures of the relative importance of individual features provides a tool for determining features that are unnecessary or of little importance.

Wrapper Methods. The wrapper methods directly evaluate combinations of features by running the ML model restricted to the set of candidate features. Taken to an extreme, all combinations would be evaluated, an impossible task in practice. Thus, various search space approaches are employed. Forward search iteratively adds promising feature vectors one by one to build a feature set. Backward search starts with all features and successively eliminates those that

 ISBN: 978-1-61208-990-4

perform poorly. Many approaches based on optimization techniques are available [31]. The self-driving car work basically follows a forward selection approach based on both advance knowledge about the importance of certain features from analytics on the data itself along with test runs of the ML model. Heavy computational load and possibilities of overfitting are potential drawbacks [33].

Embedded Methods. Embedded methods utilize mathematical information that is available during the training of the model to determine the relative importance of features. In some sense embedded methods mitigate the drawbacks of the filter and wrapper methods but retain their strengths. When implemented carefully, they are not prone to overfitting [34]. The XGBoost technique produces an importance score for each attribute that is used to identify those that can be confidently eliminated [35]. In applications like intrusion detection, a large number of attributes presents a huge computational burden. The embedded methods are highly successful in greatly reducing the features needed in intrusion detection ML work.

VI. CONCLUSION AND FUTURE WORK

Machine learning is now a well-established and effective approach in many domains. The studies on collisions involving self-driving cars and on intrusion detection in networks reported in this paper are examples of complex problem-solving domains with special challenges. These applications have dimensions of importance in inter-related areas of cybersecurity, trust, risk, safety, reliability, autonomy, and anti-autonomy. The data-centric and feature engineering challenges are extensive, but addressable. We describe approaches to addressing these challenges. Results reveal several implications for research needs and frontiers for next steps in research. These include the quest for methods that can be deployed in real-time, automate feature engineering, choose, and extract features dynamically, and simultaneously support multiple performance evaluation metrics.

REFERENCES

[1] T. Hey, S. Tansley, and K. Tolle "The Fourth Paradigm: Data-Intensive Scientific Discovery, " Redmond, Washington: Microsoft Research, 2009.

[2] A. Rastogi, "Trust and Anti-Autonomy Modelling of Autonomous Systems," Order No. 28255688, North Dakota State University, Ann Arbor, 2020.

[3] R. Meiri and J. Zahavi. "Using simulated annealing to optimize the feature selection problem in marketing applications," In *European Journal of Operational Research*, vol. 171, no. 3, pp. 842-858, 2006.

[4] M. Lombardi and M. Milano, "Boosting combinatorial problem modeling with machine learning," In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5472-5478, 2018.

[5] M. Charikar, V. Guruswami, R. Kumar, S. Rajagopalan, and A. Sahai, "Combinatorial feature selection problems," In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 631-640, 2000.

[6] KDD Cup 1999 Data, Kdd.ics.uci.edu [Online]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed: 22 August, 2022].

[7] A Deeper Dive into the NSL-KDD Data Set. Medium [Online]. Available from: https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657 [Accessed: 22 August, 2022].

[8] L. Dhanabal and S.P. Shantharajah "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," In *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446-452, June 2015.

[9] S. Choudharya and N. Kesswani "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," In *Procedia Computer Science*, vol. 167, pp. 1561-1573, 2020.

[10] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole "Benchmarking datasets for Anomaly-based NetworkIntrusion Detection: KDD CUP 99 alternatives," In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), IEEE*, pp. 1-8, 2018.

[11] 8 Categorical Data Encoding Techniques to Boost your Model in Python!, Analytics Vidhya [Online]. Available from: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/ [Accessed: 22 August, 2022].

[12] Introduction to SAS, UCLA: Statistical Consulting Group. [Online]. Available from: https://stats.idre.ucla.edu/sas/modules/sas-learning-moduleintroduction-to-the-features-of-sas/ [Accessed: 22 August, 2022]

[13] A Data Scientist's Toolkit to Encode Categorical Variables to Numeric. [Online]. Available from: https://towardsdatascience.com/a-data-scientists-toolkit-to-encode-categorical-variables-to-numeric-d17ad9fae03f [Accessed: 22 August, 2022]

[14] Improve your classification models using Mean /Target Encoding. [Online]. Available from: https://medium.com/datadriveninvestor/improve-your-classification-models-using-mean-target-encoding-a3d573df31e8 [Accessed: 22 August, 2022]

[15] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," In *2015 military communications and information systems conference (MilCIS), IEEE*, pp. 1-6, 2015.

[16] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," InIn *Intelligent data analysis*, vol, 6, no. 5, pp. 429-449, 2002.

[17] R. Gomes, M. Ahsan, and A. Denton, "Random Forest Classifier in SDN Framework for User-Based Indoor Localization," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0537-0542, 2018.

[18] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," In *Int. J. Adv. Soft Comput. its Appl.*, vol. 5, no. 3, 2013.

[19] A. Ghazikhani, H. S. Yazdi, and R. Monsefi, "Class imbalance handling using wrapper-based random oversampling," In *20th Iranian Conference on Electrical Engineering (ICEE2012), IEEE*, pp. 611-616, 2012.

[20] Z. Zheng, Y. Cai, and Y. Li, "Oversampling method for imbalanced classification," In *Computing and Informatics*, vol. 34, no. 5, pp. 1017-1037, 2015.

[21] A. M. Denton, M. Ahsan, D. Franzen, and J. Nowatzki, "Multi-scalar analysis of geospatial agricultural data for sustainability," In *2016 IEEE International Conference on Big Data (Big Data), IEEE*, pp. 2139-2146, 2016.

[22] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special Issue on Learning from Imbalanced Data Sets," In *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1-6, 2004.

[23] M. Ahsan, R. Gomes, and A. Denton, "SMOTE Implementation on Phishing Data to Enhance Cybersecurity," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0531-0536, 2018.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," In *Journal of artificial intelligence research*, vol. 16, pp. 321-357, 2002.

[25] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," In *International conference on intelligent computing*, pp. 878-887. Springer, Berlin, Heidelberg, 2005.

[26] H. Hairani, K. E. Saputro, and S. Fadli, "K-means-SMOTE untuk menangani ketidakseimbangan kelas dalam klasifikasi penyakit diabetes dengan C4.5, SVM, dan naive Bayes," In *J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

English – H. Hairani, K. E. Saputro, and S. Fadli, K-means-SMOTE to trate class imbalance in the classification of diabetes with C4.5, SVM, and Naïve Bayes, *In J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

[27] Y. Tang, Y. Q. Zhang, and N. V. Chawla, "SVMs modeling for highly imbalanced classification," In *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281-288, 2008.

[28] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE*, pp. 1322-1328, 2008.

[29] O. Özdemir, M. Batar, and A. H. Işık , "Churn Analysis with Machine Learning Classification Algorithms in Python," In *The International Conference on Artificial Intelligence and Applied Mathematics in Engineering*, pp. 844-852. Springer, Cham, 2019.

[30] N. O. F. Elssied, O. Ibrahim, and A. H. Osman, "A novel feature selection based on one-way ANOVA F-test for e-mail spam classification," In *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 3, pp. 625-638, 2014.

[31] M. Ahsan, R. Gomes, and A. Denton, "Application of a convolutional neural network using transfer learning for tuberculosis detection," In *2019 IEEE International Conference on Electro Information Technology (EIT), IEEE*, pp. 427-433, 2019.

[32] A. Mustaqeem, S. M. Anwar, M. Majid, and A. R. Khan, "Wrapper method for feature selection to classify cardiac arrhythmia," In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE*, pp. 3656-3659, 2017.

[33] M. Ahsan and K. Nygard, "Convolutional Neural Networks with LSTM for Intrusion Detection," In *The 35th International Conference on Computers and Their Applications (CATA 2018)*, vol. 69, pp. 48-59, pp. 69-79. 2020.

[34] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," In *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 703-715, 2019.

[35] Y. Wang and X. S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," In *arXiv preprint arXiv:1901.08433*, 2019.

[36] M. Chowdhury, J. Tang and K. Nygard, "An artificial immune system heuristic in a smart grid," In *The 28th International Conference on Computers and Their Applications*, 2013.

[37] M. Chowdhury and K. Nygard, "Machine learning within a con resistant trust model," In *The 33rd International Conference on Computers and Their Applications (CATA 2018)*, pp. 9-14, 2018.

[38] A. Rastogi and K. Nygard, "Threat and alert analytics in autonomous vehicles," In *The 35th International Conference on Computers and Their Applications (CATA 2018)*, vol. 69, pp. 48-59, 2020.

[39] Home - Keras Documentation, Keras.io, 2020. [Online]. Available: https://keras.io/. [Accessed: 16 August, 2022].

[40] "TensorFlow", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/. [Accessed: 16 August, 2022].

[41] "tensorflow/tensorflow", GitHub, 2020. [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed: 16 August, 2022].

[42] M. Ahsan, , N. Rifat, M. Chowdhury, and R. Gomes, Detecting Cyber Attacks: A Reinforcement Learning Based Intrusion Detection System. *In 2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 461-466, 2022.

[43] M. Ahsan, K. E. Nygard, R. Gomes, M . M . Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review, " *In Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 527-555, 2022.

[44] N. I. Rifat, "Feature Engineering on the Cybersecurity Dataset for Deployment on Software Defined Network, " 2020.

[45] Y. Berra, "A Quote By Yogi Berra", 2022. [Online]. Available: https://www.goodreads.com/quotes/261863-it-s-tough-to-make-predictions-especially-about-the-future. [Accessed: 22-August-2022].

[46] M. Ahsan, N. Rifat, M. Chowdhury, and R. Gomes, "Intrusion Detection for IoT Network Security with Deep Neural Network, " *In 2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 467-472, 2022.