Manuscript accepted for IEEE 12th Annual Computing and Communications Workshop and Conference (CCWC), Jan 2022

Challenges Faced by Industries and their Potential Solutions in Deploying Machine Learning Applications

Raj Mani Shukla*† and John Cartlidge*
*Department of Computer Science, University of Bristol, UK
†Claritum Limited, Bath, UK
{raj.shukla,john.cartlidge}@bristol.ac.uk

***Abstract*—Across all sectors, organizations attempt to make efficiency savings and performance improvements by incorporating machine learning (ML) into commercial application services. However, in comparison to traditional software applications, design, deployment, and maintenance of ML applications is more complicated. In particular, ML introduces new challenges of data availability, concept drift, scalability, and technical debt. In this paper, we introduce some of the practical challenges that arise when deploying ML applications, and describe potential solutions. Our analysis is based on experience designing and deploying a commercial spend classification service.**

***Index Terms*—Spend analysis, ML service, Application deployment**

I. Introduction

Technology corporations continue to explore ways to best leverage the potential of artificial intelligence (AI) and machine learning (ML) in application software. AI/ML-based applications typically use algorithms to parse data, learn from it, and make predictions; i.e. the system learns from data without relying on rule-based programming. "Big tech" companies such as Google, Facebook, and IBM, have prioritised AI/ML development for many years. More recently, many startups and small-to-medium enterprises (SMEs) are investing in the adoption of AI/ML to optimize performance and functionality. McKinsey estimate the AI/ML-based market at USD 62.35 billion in 2020, and predict annual growth of 40% until 2030, with 50% of companies that embrace AI/ML over the next five to seven years having the potential to double their cash flow [1].

While commercial opportunity, aligned with availability of data, affordable data storage, and growth of processing infrastructure has spurred the growth of ML-based Software as a Service (SaaS) products, developers face challenges building reliable ML-based applications. Normally, the process of building an ML application begins with developing prototype models, where a variety of algorithms and methods are explored. In this step, the accuracy metrics for the prototype are important indicators for the feasibility of the final product. Subsequently, the prototype model is further developed and deployed for use in the live system using software engineering techniques and tools. For the deployed system, the success metrics are binary: does the application succeed or fail in improving human-level performance. Thus, although the prototype model may have high accuracy, the deployed application may not be useful for the end-user. There are various reasons why an ML system may fail to deliver during deployment, even if the prototype model is accurate. These factors include, but are not limited to, differences between test data and live data, low latency, varying data streams, and scalability. Additionally, deployed systems also require continuous monitoring and maintenance, which incurs high technical debt. It is estimated that these costs are significantly higher for ML systems than traditional software applications [2].

Furthermore, ML applications introduce new complexities: ML-based solutions are probabilistic rather than deterministic in nature; output is a function of ML model and data; and developers need to keep track of different versions of the model, hyperparameters, and accuracy metrics, which becomes more challenging as data size grows. While traditional software is expected to be deterministic, ML software will provide different results as the data profile inevitably changes. Testing ML systems is another difficulty, as data and its schemas must be tested along with the software code. Due to these differences, traditional DevOps practices for software engineering, such as continuous integration (CI) and continuous development (CD), may fail when used directly for ML application deployment. In addition, the system requires continuous monitoring and tracking to enable high performance and quality user experience.

In this paper, we discuss some of the potential challenges and their solutions in deploying a real-world application. As a case study, we provide details of a commercial ML-based spend classification application that was deployed for Claritum, a UK-based SME. We discuss research challenges faced while developing the

978-1-6654-8303-2/22/$31.00 ©2022 IEEE

TABLE I
SAMPLE PRINT SPEND DESCRIPTIONS, WITH FIVE CATEGORY LABELS PROVIDED BY EXPERT ANNOTATOR.

Specification	Project	Item	Size	Finishing	Stock
2 A2 posters one-sided 4-colour 3 A4 posters one-sided 4-colour	POS	Poster	A4	Stock supplied	Board
Printed black to face only manilla 110gsm Self seal non window pocket Boxed in 250's	Direct mail	Envelope	C4	Overprint	Paper
Super hit basic extreme branding 2883 - black ink. 4 colour process printed on barrel	Commercial	Promo	A4	Laminated	Paper

TABLE II
PRINT-SPEND CATEGORIES LABELED FOR TRAINING.

Category	Instances
Project	Commercial, Direct Mail, Logistic, POS, Stationary
Item	Booklet, Carriage, Envelope, Fulfillment, Inkjet, Label, Leaflet, Letterhead, NCR, Poster, Promo
Size	A1, A2, A3, A4, A5, C4, Double, Simplex
Finishing	Continuous, Laminated, Overprint, Stock supplied
Stock	Board, Card, Paper, PVC

spend classifier, and provide potential solutions that we adopted, or could be adopted, for similar applications. Some challenges are specific to the spend classification service (SCS) developed for Claritum, while others are more general in nature.

The rest of this paper is organized as follows. A description of the SCS platform is introduced in Section II. Section III discusses the practical challenges and their solutions in deploying ML applications. Finally, conclusions are presented in Section IV.

II. SPEND CLASSIFICATION PLATFORM

This section describes the development of a commercial Spend Classification Service (SCS). The initial focus of the service is on print spend, however the final application will generalise to any spend category. Many large corporations have significant spend on printed materials to fulfill their daily activities. For example, a typical supermarket chain will spend tens or hundreds of millions of dollars every year on printed items such as posters, labels, letters, and stickers. Much of this spend data is uncategorized.[1] Generally, spend data is stored in a company's procurement system and takes the form of unstructured free-text specifications. Understanding and extracting relevant information from this data requires the skills of spend-analysts with knowledge of the sector (i.e., print) that is being analyzed.

Table I presents a sample of print spend specifications. The spend-analyst has categorised the data into five levels, which are most useful for a spend management exercise. These categories are (i) project, i.e., the general spend area, (ii) item, (iii) size, (iv) finishing, and (v) stock, i.e., the material of manufacture. A summary of category instances are presented in Table II.

Having knowledge of categorized spend data enables procurement managers to choose suitable suppliers and negotiate on price. Traditionally, spend data is categorized by spend-analysts; human experts with detailed knowledge of the sector. However, the process is laborious and time consuming. For instance, it took two working days (16 hours) for a spend-analyst to annotate 800 rows of the sample data presented in Table I. A large organization is likely to have millions of rows of spend data each year. This makes a spend-analysis exercise costly. However, as such an exercise usually results in savings of at least 10% of total spend, the likely rewards easily compensate for this outlay. Claritum aim to develop an ML-based classifier to automatically categorize spend data, which will lead to significant efficiency gains and cost savings; and will enable smaller companies to benefit from better spend management.

A view of Claritum's SCS interface, currently in Beta release, is presented in Figure 1. The user approves a row classification by selecting "thumbs up". Approved rows are then used to re-train the models in the background to improve the system performance. In the back-end, different models, such as SVM, Naive Bayes, and Random Forest are compared. As the platform is designed to serve multiple customers, a base model is made available and distributed to each end-user. Users then further train personal instances of the model using their own data and classifications, as shown in Figure 2. Thus, one public model is shared with all customers to generate private models for each. The public model is incrementally improved as the data is accrued from different sources.

[1]One major supermarket chain reported that up to half of all print spend invoices may be uncategorized (personal communication).

#	Specification	finishing	item	size	project	stock	Approve?
1	'Transparent self-adhesive vinyl printed in colour and then over printed in white measuring 2400mm x 1800mm Transparent self-adhesive vinyl printed in colour and then over printed in white measuring 2400mm x 2625mm '	Laminated 62%	Poster 100%	A4 60%	POS 51%	Paper 35%	
2	'Ref - Superhit 2883 Pens (Black) Black Ink Barrel Print - 2 colours (yellow / white) Clip Print - 1 colour (yellow)'	Laminated 54%	Promo 100%	A4 64%	Commercial 35%	Paper 96%	
3	'White body Plastic Printed 2 colour logo to 1-position Packed to suit Delivery to Glasgow '	Overprint 41%	Promo 100%	A4 74%	Commercial 72%	Paper 89%	
4	'Sapporo Keyring Engraved in 1-position Product code: - 16608M Packed to suit Delivery to Glasgow Prices reqd By Thursday 9am please '	Stock supplied 47%	Promo 100%	A4 57%	Commercial 32%	Paper 43%	
5	'Plain white laser labels with permanent adhesive Size: 104 x 148.50 Squoare Cut Code: LL04NSE'	Laminated 40%	Label 92%	A6 100%	POS 93%	Paper 69%	
6	'Size: 85mm x 55mm, 2pp 4:4 on 350gsm uncoated Trimmed to size Delivery is to one UK address Delivery is to 1 UK address Supplier to make one artwork amend- change to date in T&C\'s Please advise your best lead time from approval **Price and lead tim	Laminated 39%	Poster 30%	A4 30%	POS 56%	Board 90%	
7	'Size: 5 1/2 deep x 9 1/2"	Continuous 89%	NCR 95%	A4 34%	Commercial 92%	Paper 88%	
8	'Size: 114mm deep x 232mm Printed 2 colours to face, one to flap on white 90gsm Standard opaque Window size: 40mm deep x 93mm Position: 22mm up from base, 23mm in from left Gummed wallet '	Overprint 85%	Envelope 69%	A4 47%	Direct Mail 76%	Paper 92%	
9	'Reference: 6000-26C1-F Dims: 44.0 x 2.0 x 520.0 Outer: Brown Kraft Plugs: 2 x 0.44'	Stock supplied 58%	Leaflet 29%	A4 41%	Logistics 34%	Paper 53%	

Fig. 1. User interface of *BRIGHT Spend Analytics.AI* SaaS. Specifications are automatically classified and presented with "traffic light" coloring to indicate confidence score. Cells with incorrect classifications can be manually corrected, which are then given a confidence score of 100% (e.g., row 1 item "poster"). Users approve a row by selecting "thumbs up".

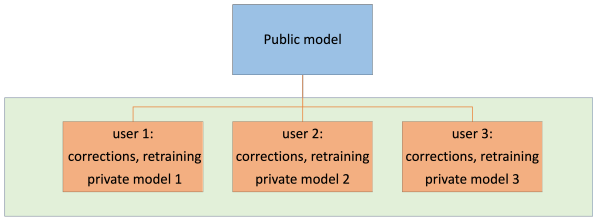

Fig. 2. Retraining process for the platform.

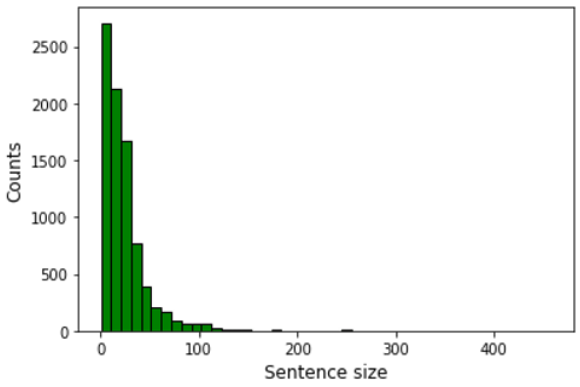

Fig. 3. Specification length distribution.

There exists many challenges and practical issues that need to be solved in categorizing such data and deploying the ML-application. In the next section, we discuss a selection of the main challenge. Some are specific to spend classification, while others are applicable to any ML application service.

III. Challenges related to employing ML in spend analysis application

A. Expert data labeling

Data labeling is imperative for training supervised ML applications. Initial data labelling is performed manually by domain experts, and may be performed in parallel by multiple experts. The task of manual data labeling is often delegated to external expert consultants, and is generally not performed by the ML application developers. Thus, different experts and different companies are likely to label similar data in different ways. Therefore, it is important that expert classifiers are involved in the development process from an early stage.

B. Short text classification

Spend data is often unformatted text containing a description of the items purchased. Therefore, it is prone to errors, spelling mistakes, inconsistencies, and idiosyncrasies. Descriptions also tend to be short. A distribution of description lengths of real-world print spend data are shown in the Figure 3. We see that the mode length is less than 10 words, and only a tiny fraction contain 100 words or more. The classification of short-text is challenging because descriptions often lack semantic meaning; they are sparse and have fewer word co-occurrences; and they do not provide enough information for good similarity measures between different texts. Short texts are also more ambiguous and do not contain as much contextual information as longer paragraphs or documents.

Classification of such text is challenging [3]. Improvements can be made by pre-processing text using regular expressions to extract keywords. Techniques such as Graph Convolution Neural Networks can also be used, however the substantial training times [4] can make these approaches impractical for applications where models are trained frequently per user.

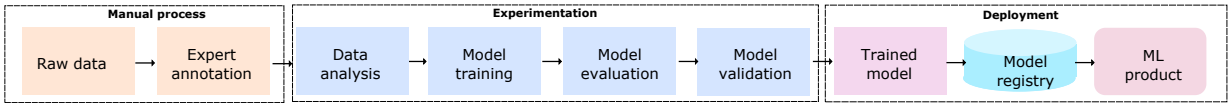

Fig. 4. Machine Learning product development pipeline.

C. Long ML product development cycle

The ML product development lifecycle is long and includes extensive experimentation, data analysis, and validation steps. A typical ML pipeline is presented in the Figure 4. The process begins with data acquisition and expert data annotation, which is a time-consuming manual process. There is then an iterative cycle of experimentation to develop a proof-of-concept application, where various models are trained and tested for performance. Subsequently, the application is deployed using traditional software engineering practices. Once the application is live, users are able to upload and classify their own data. However, as this data may vary significantly from the training data, it is possible for the application to perform unexpectedly. Therefore, continuous on-going monitoring is required.

Due to these difficulties, many small-scale enterprises are reluctant to invest time and money into ML application development. As a result, many ML application proposals are rejected before they leave the drawing board. To mitigate this problem, methods need to be developed to enable quick deployment of ML applications and reduce the overhead of a complex ML development cycle. Some of the practices that can help developers are provided below:

1) Automatization of ML processes: Recently, there has been an introduction of practices to automate steps of the ML development pipeline, including data acquisition, experimentation, validation, deployment, monitoring, and up-gradation. Such practices are called MLOps or AIOps [5] and platforms like MLflow have been developed to streamline the ML lifecycle [6]. MLOps allows the continuous integration and continuous delivery of machine learning products and are found to be beneficial to developers. However, many data scientists still use traditional manual processing to develop applications. Additionally, MLOps practices are immature and do not cover use cases such as expert annotation for data labeling, platform interaction with the consumer, and human-in-the-loop (HITL) in ML cycles. Thus, automatizing steps of the ML process requires these additional cases to be adopted as well.

D. Agile Software Engineering Practices

Many application developers and software engineers adopt agile practices to enable better project delivery, with rapid updates based on end-user feedback [7]. Often, ML application development teams use agile software engineering practices to develop and deploy applications. However, agile development practices need to be revisited for ML applications, which differ considerably from traditional software applications. Previously, we have described AgileML, an agile methodology for deploying a spend classification application [8]. AgileML offers benefits of rapid deployment, and enables users to begin using the ML application at an early stage of development.

E. Personalized retraining

An ML product is often used by many customers, each with varying requirements, including patterns, classes, and hierarchy. Requirements for each individual customer can also vary over time. To address this issue, we suggest developing a global model that can then be further trained locally by each user. To develop and train the two models together, the following approaches can be combined and automated:

1) Transfer learning: It is possible to generate a general "base" model and then utilize transfer learning to adapt the model for each specific user [9]. Transfer learning has been used for many application domains.

2) User control: Allow users to define labels before training the model. Thus, initially, the user interface may request sample training data, and then based on that model training, testing, and selection is automated.

3) Model matching: Once a user defines their dataset and labels, the platform may check existing models in the system. This may be done using correlation testing between different text-based datasets. If there is a significant match above a threshold, the platform may suggest the possible hierarchy of labels.

These practices require that each user's specification is stored separately to avoid data poisoning and allow each user to access models that have been specifically trained for them.

F. Extreme multi-label text classification

Extreme multi-label text classification (XMTC) is the problem where a large number - hundreds of thousands, or millions - of labels are present in the dataset [10]. This is typical for problems such as spend classification where there are millions of potential product lines. Such a large number of labels gives rise to the problems of data sparsity and scalability. In these scenarios, there are likely to be many labels - so called "tail labels" - that have a very small number of positives. XMTC problems are different from binary or multi-class problems as the dependency between labels must be leveraged. Items may also belong to multiple categories.

G. Data acquisition

Often, application developers will have limited data available for developing ML applications. This occurs for various reasons. First, the industry that has data are different from the industries that develop the application. Thus, purchasing data can be costly. Second, data may contain personal or commercially sensitive information, so users will not be prepared to share. Third, data can be poisoned and may not be useful. While unsupervised or semi-supervised methods will ameliorate some of these issues, these approaches may not be suitable for applications requiring high accuracy. Other alternatives include:

1) GANs: Generative adversarial networks (GANs) could be used for generating synthetic data. GANs have been successfully applied in various application domains. However, their application in text data has not been widely studied [11]. Another problem with GAN is that they need more processing time. Thus, using GAN in the pipeline increases the time of processing the text classifier.

2) FSL: ML methods like One/Few Shot learning (FSL) enables training of models from a small training set [12]. However, one of the problems with FSL is that they have comparatively lower accuracy than supervised learning. In such a case, based on the nature of spend data - i.e., short, unstructured text - the parameters related to FSL like distance function and clustering methods need to be further investigated.

H. Mitigating technical debt

The proof-of-concept (POC) developed for a ML application is often performed on sample data. However, once the system or application is deployed online, issues of scalability arise. A POC showing high accuracy graphs and confusion metrics may not be valid when the system is used on a large scale. Traditional software systems are deterministic in nature and maintaining and sustaining the model over a long period of time is not a complex problem. ML systems, on the other hand, are probabilistic in nature as the output is associated with an error level. Due to their non-deterministic nature developing, maintaining, and sustaining the ML system is a complex business. ML systems also have technical debts due to issues like feedback loops and correction cascades. The technical debt is worse in ML systems because deploying the number of tasks needed to develop a new version of the system gets multiplied.

I. Testing/Verification

Traditional software systems are generally static in nature and thus it's relatively easy to perform A/B comparison testing, or analyze the code coverage of the software systems. In comparison, the probabilistic nature of ML-based software makes testing and verification more difficult. Furthermore, the performance of ML-based systems depends upon the data. Besides drift patterns in the data, there may be various unintentional changes that can degrade performance. For example, if a company updates a product code, this may conflict with an older product code that appears in the training data. Testing of such inconsistency is difficult and needs to be included in the ML pipeline. In addition to testing the software application, the data and its schema also require testing. For automated ML facilities, i.e. MLOps, the software itself is divided into several components, responsible for automatic training, testing, and deployment of the application. Therefore, the whole pipeline needs to be automatically tested, which is more difficult than testing individual components.

J. Reproducibility

For an ML service, a seed model is deployed and then retrained when new data is collected. The model training is a repetitive process and the seed model and data may differ significantly across versions of the trained model. The latest versions of the model may have high accuracy metrics of precision, recall, and f-score and handle the more complex problems containing more classes. Also, the models trained on a large incoming data-set handle data pattern variation better. However, as the ML model is probabilistic they may still not produce the same results on specific samples. This is problematic for users that are often not experts in data science as they may see a sample that was correctly classified in an earlier version is being incorrectly classified in a recent model trained on large data and handles the better problem. Thus, reproducibility is a problem for ML applications that need to be handled. A simple solution would be to provide a feedback mechanism to the user about the average accuracy and how the new model is solving a complex problem to raise human-level performance. Another option would be to compare each new sample if it exists in the training data and copy the predictions from training data rather than using the ML model. However, this is a time-consuming process as the training data and incoming data stream to be classified grows.

K. Explainability/Interpretability

One of the major hindrances in deploying ML applications for commercial applications is that they are black-box in nature. For applications like spend analysis, interpretation of the classification may further augment the consumer experience. There exist many open-source explainable ML models like Local Interpretable Model-Agnostic Explanations (LIME) and Shapley Additive Explanations (SHAP) that can be employed to interpret classification results [13]. Employing these techniques can help consumers further accelerate the verification process, as shown in the user interface presented in Figure 1. However, it is difficult to estimate

the accuracy of these interpretations. For ML models, classification errors and confusion matrices can describe model performance, but for LIME and SHAP it's difficult to mathematically validate these explanations. Therefore, it is necessary to include a human-in-the-loop (HITL) process.

L. Improving base models

In the scenario where a base model is distributed to different customers and then the models are privately trained, as shown in Figure 2, a problem that is commonly encountered is that the base model lags behind the privately trained models. Although the base model can be improved by training as new data is gathered, the data source may be different than what the consumers have trained their data on. Thus, the improvement in the base model by the service provider using a dataset that may be different than the consumer has, is not an ideal situation. To mitigate the issue the following approaches could be adopted:

1) Federated machine learning: It is common for customers to be reluctant to share data with the developers, which makes application development difficult. To mitigate this issue, federated learning needs to be revisited for developing an application service [14]. Federated learning may help develop more robust models by combining various local models of the customers into a global model. Federated learning requires interfaces where the local models can be shared to create a global model. However, there are privacy issues as information, such as the size of the data a model is trained on, may be leaked. This needs to be mitigated by adopting different approaches such as automating the training process and reducing developer intervention.

2) Secure information sharing: The companies indeed hesitate to share their data for training due to privacy and security issues. However, ML systems are data-hungry and they need labeled information for training and making a better ML model. Therefore, standards to share the domain data between organizations need to be investigated. Such standards are quite popular in domains like cybersecurity for sharing threat-related information. Various governments encourage organizations to share cyber-threat-related information with each other [15]. Much research has been done to develop models such as CYBEX, to share such information between organizations [15]. Along the same line, the standards and models to share data securely for ML training, while preserving user privacy, can be developed. One of the problems with ML data is that they often are very large as compared to the domains like cybersecurity and thus standards need to be developed accordingly.

IV. Conclusions

This paper has presented a spend analysis application and its architecture. We presented practical issues that were faced during the development of this commercial spend classification service, and discussed approaches that can be adopted to address some of these issues.

Acknowledgment

This work was supported by Innovate UK Knowledge Transfer Partnership between University of Bristol and Claritum Limited (KTP 11952).

References

[1] T. Balakrishna, M. Chui, B. Hall, and N. Henke, "The state of AI in 2020," 2020. [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2020

[2] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, pp. 2503–2511, 2015.

[3] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Journal of Information, MDPI*, vol. 10, no. 4, p. 150, 2019.

[4] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7370–7377.

[5] GoogleCloud, "MLOps: Continuous delivery and automation pipelines in machine learning," 2021. [Online]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[6] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar, "Accelerating the machine learning lifecycle with MLflow," in *USENIX Conference on Operational Machine Learning (OpML)*, May 2019.

[7] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile Software Development Methods: Review and Analysis*. VTT, 2002.

[8] R. M. Shukla and J. Cartlidge, "AgileML: A machine learning project development pipeline incorporating active consumer engagement," in *IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2021.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[10] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 115–124.

[11] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational generative adversarial networks for text generation," in *International conference on learning representations*, 2018.

[12] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[13] X. Man and E. P. Chan, "The best way to select features? comparing MDA, LIME, and SHAP," *The Journal of Financial Data Science*, vol. 3, no. 1, pp. 127–139, 2021.

[14] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[15] D. K. Tosh, "Market based models for cybersecurity information exchange (CYBEX)," Ph.D. dissertation, University of Nevada, Reno, 2016.