WOLF: Machine Learning Workflow Management Framework

Sohaib Kiani, Pranav Bahl, Xiaoli Li, Casey Sader and Jun Huan
Faculty of Electrical Engineering and Computer Science
The University of Kansas
Lawrence KS, USA
sohaib.kiani, pranav_bahl, xiaoli.li, c455s614, jhuan @ku.edu

***Abstract*—Applying machine learning techniques is a highly iterative process. The process from idea to code and then to experiment requires several thousands of iterations to find optimum hyper-parameters. Also, it is hard to find best machine learning techniques for a given dataset. WOLF framework has been designed to simultaneously automate the process of best algorithm selection and simplify the search of optimum hyper-parameters. It can be useful to both who are novice in machine learning and just want to find best algorithm for their dataset and also to those who are expert in the field and want to compare their new features or algorithm with state of the art techniques. By incorporating the WOLF framework in their designs, it is easier for novices to apply machine learning techniques on their dataset. With a wide range of evaluation metrics provided, it also helps data scientists to develop better intuition towards machine learning techniques and speed up the process of algorithm development. The main feature of WOLF framework is that user can easily integrate new algorithms at any stage of the machine learning pipeline.**

I. INTRODUCTION

Machine learning has been quite successful in various fields like health, finance, education, sports, computer vision, speech recognition etc. which has encouraged demand for machine learning systems amongst novices in the field. Open source libraries [1] [2] have been built to help novices use machine learning algorithms. By definition, machine learning automates the task of learning in terms of rule induction, classification, regression etc. to draw knowledgeable insights and to forecast an event before it actually takes place. Despite this, the task of selecting the best algorithm(s) for a specific dataset is not automated [4]. With the rapidly growing pool of available machine learning algorithms, it becomes difficult for novices as well as researchers to choose the best algorithm specific to their given dataset. The main steps of any machine learning system consists of preprocessing to prepare data, followed by feature selection/extraction to remove noise or redundant information from data, and then finally applying and evaluating machine learning techniques to gather useful information about the data [3].

Automation is the fuel that drives WOLF. Automating time-consuming and repeatable tasks are the defining characteristics of the project. The rising scope of Artificial Intelligence (AI) and machine learning increases the need for automation to simplify the process and hence help researchers and data scientists to dig deeper into the problem and understanding it well rather than spending time in tweaking the algorithms and the parameters involved. The positive correlation of growing intelligence and the complexity of solutions has shifted the trend from Artificial Intelligence (AI) to Automated Intelligence, a paradigm on which WOLF is based.

WOLF has been built to have impact on a wider audience than traditionally targeted by the machine learning community. The automation of machine learning pipeline helps people with different levels of expertise and requirements, helping novices to identify the best possible combination of algorithms without even having in-depth knowledge of the inner workings of the algorithms. At the other end of the spectrum, WOLF serves as a useful supporting tool for seasoned researchers and businesses to figure out the best resulting hyper-parameters, helping them reduce the time-to-market. The project is developed with the idea behind rising interest of data scientists towards Data Science competitions, as the machine learning automation has become a necessity as competitors spend lot of time in model construction and evaluation.

Although the project so far supports the classification problems but there are a handful of classification algorithms consisting of base classifiers and ensemble methods as well that are not currently supported. WOLF framework provides the leverage of adding proprietary algorithms, making it more customizable since the scope of growing algorithms in the system is unrestricted. The process of adding an algorithm to the pipeline is pretty intuitive and straight forward. Also, the idea of not stealing the freedom of model evaluation from the user has been taken care of in the project; the result file consists of various widely used metric calculations giving user better understanding and control through easy comparison of pipeline combinations, therefore helping the user make decisions with high confidence.

In this paper architecture of WOLF framework will be explained. The usability of the WOLF framework will be demonstrated through experiments on various types of datasets. Moreover, the results generated by WOLF will be used to support the following hypothesis:

1. Comparison between Deep and Shallow Models: Per-

formance comparison between deep and shallow networks explains whether using deep networks is always beneficial or not?

2. Comparison between Machine Learning Algorithms: Performance comparison between different Machine Learning algorithms supported by WOLF demonstrate if they perform differently for a given dataset?

3. Fine-Tuning Hyper-parameters: Is there any benefit of doing fine tuning of hyper-parameters?

This paper has been organized as follows: Section II gives brief overview about similar frameworks developed in the past. Section III highlights main features of the WOLF framework. Section IV and V, provides detailed descriptions about architectural and database management aspects of WOLF. In section VI, the usability of the WOLF framework has been demonstrated by generating results for various data sets. Also, through experimental results above mentioned hypothesis regarding machine learning techniques has been described in section VII. Section VIII highlight some of the future work in order to make WOLF framework more useful.

II. Related Work

In recent years, various platforms and techniques have been developed for easing cumbersome tasks in a machine learning (ML) workflow, such as data pre-processing, feature extraction and selection, and model selection. They can be categorized into two categories, ML workflow management and ML workflow discovery.

A. ML Workflow Management

Platforms and techniques in this category provide tools for creating, modifying, executing or sharing ML workflows. The FBLearner Flow system from Facebook was designed to be capable of easily reusing algorithms, scaling to run thousands of simultaneous custom experiments, and managing experiments with ease. KNIME [6] enables easy visual assembly and interactive execution of a data pipeline through customizable and extentionable nodes. The Portable Format for Analytics is an emerging standard for statistical models and data transformation engines [7]. PFA combines portability across systems with algorithmic flexibility: models, pre-processing, and post-processing are all represented as functions that can be arbitrarily composed, chained, or built into more complex workflows. Kepler [8] provides an graphical interface for creating a “scientific workflow” an executable representation of the steps required to generate results.

B. ML Workflow Discovery

The techniques in the second category is more related to WOLF in the sense that they aim to discover an optimal ML workflow for a ML task. TPOT is a Python automated machine learning tool that optimizes machine learning workflows using genetic programming [9]. The scalability of TPOT may be problematic since its process of finding optimal workflow were not designed for distributed computation. To addressing the issue of scalability in machine learning, Tim et al. proposed MLBase framework [10] consisting of three components, ML Optimizer, MLI, and MLLib. Through ML Optimizer, an optimal learning plan can be selected for a ML task, such as classification, specified using a declarative language. Note that ML Optimizer is still under development and MLLib is specifically designed for Spark. It requires non-negligible efforts to implement an Spark compatible algorithm to achieve distributed computation. DataRobot is a proprietary data science system with software that covers the tasks that a data scientist typically performs. It is designed to automate the task of data cleaning, visualization, model construction, model evaluation, and making predictions.

III. Features of WOLF Framework

The objectives of WOLF framework are quite similar to other related projects mentioned in previous section [4] [9] . However, WOLF implementation is much simpler and there are certain features of WOLF systems that makes it a more attractive automated machine learning platform to use for a wide variety of users. Following are the main features of WOLF platform that differentiate it from similar existing frameworks:

A. Standard Input format:

In order to make preparation of Input Data simple, Wolf requires input dataset should be converted into ”*.arrf” format, which is the most common input file format among other similar platform like Auto-Weka [4].

B. Integration of new algorithms:

To ease the task of developing new algorithms at any stage in the machine learning pipeline(preprocessing,feature selection, machine learning algorithms) WOLF provides the user with a hands-on tool to test and compare newly designed algorithms with state of the art techniques without worrying about other stages of machine learning pipeline. Consequently, it speeds up the development phase of algorithms and saves lots of time and effort in redesigning other stages of the machine learning pipeline.

C. Search Optimum Hyper-parameters:

It is possible to search the hyper-parameter space for the best cross validation score through WOLF. WOLF also enables users to search optimum hyper-parameters for selected machine learning algorithms. User s can provide the WOLF platform with a range of possible hyper-parameter values for which WOLF performs Grid search. Together with optimum hyper-parameter search for each algorithm, WOLF provides the user with the algorithm that performs best for given dataset using cross validation scores.

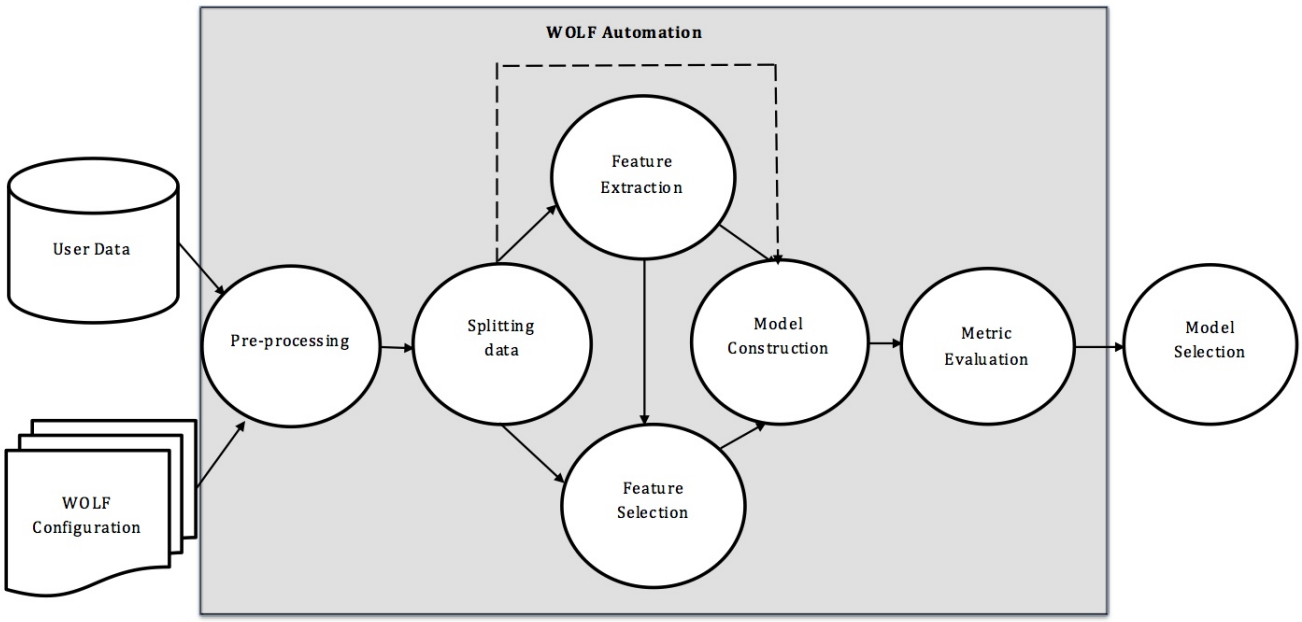

Figure 1. WOLF Framework Transcations

D. Built to Support wide range of tasks:

WOLF has been designed to cover a wide range of machine learning tasks, which include binary and multi-class classification. Different metric evaluation criteria have been integrated into WOLF to develop better intuition of machine learning models for a given dataset.

IV. WOLF Architecture

WOLF works in transactions to figure out the best pipeline for the user's data. A transaction is nothing but a common machine learning task performed to build the best solution for a data problem. Some of those tasks include data pre-processing, cross validation, feature engineering, model construction and evaluation. Each transaction provides choice of various algorithms. The WOLF platform runs all transactions based on the configuration provided by the user, and selects a pipeline consisting of best performing algorithm for each transaction individually. Each transaction in the project is isolated but is dependent on the output generated by its prior task. WOLFs workflow is shown in Figure 1.

A typical WOLF pipeline consists of multiple transactions, not all transactions are mandatory. The dotted line in Figure 1 is to show alternate paths or the transactions that can skipped in a pipeline. Each transaction performs a meaningful task.

The WOLF transactions are dependent on the user input configuration. To keep the consistency across all user configurations, the project follows a protocol which defines the structure and keywords used for creating a configuration file.

Following are the list of transactions in machine learning pipeline adopted in WOLF framework:

A. Preprocessing:

There are some common data preprocessing operations which are performed on the data file provided by user. This step includes operations like getting rid of NaN (Not a Number) values or replacing them with user specified values. Label encoding of categorical features and scaling of numerical features is also performed in the preprocessing step.

B. Splitting data:

The splitting data transaction replicates the task of cross validation. After pre-processing of user input data it is then split into multiple pairs of train and test files. The number of train and test data file combinations generated depends on the values provided by the user for the number of folds and repetitions parameter under split data configuration. The data file is always randomized before splitting which leads to distinct pairs of train and test files.

C. Feature Extraction:

The data extraction is mostly meant for dimensionality reduction of data set or for creating new features by combining multiple features which may be more meaningful to model construction algorithms. The train and test file combinations created by splitting data transactions are treated as input for data extraction phase. Because not all the data sets requires data extraction or the user may want to take control of

the feature extraction step, this transaction has been made optional.

The method used for dimensionality reduction is Principal Component Analysis (PCA).

D. Feature Selection:

Taking the train and test files pairs generated after data extraction or splitting data transaction (if data extraction is skipped) the operation of feature selection is performed on them. The idea behind this transaction is to get rid of noisy data or the features which are not meaningful to model construction algorithms. The task of feature selection is also optional for the same reasons as feature extraction. After performing feature selection, pairs of train and test files are created but this time the output files consist of meaningful features only.

The algorithm used for this purpose is Support Vector Machine_Recursive Feature Elimination (SVM_RFE). Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and weights are assigned to each one of them. Then, the features whose absolute weights are the smallest are pruned from the current set features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

E. Model Construction:

The input for model construction transaction is train and test file combinations generated by feature selection or feature extraction or the splitting data transaction based on the users configuration if any of the transaction is skipped. This transaction consists of multiple algorithms and each model construction algorithm runs in parallel. The output generated by this transaction consists of result files which contain true values and predictions made for test files. The number of generated files depend on the number of algorithms chosen by the user for experimentation and also the freedom of hyper-parameters as configured by users. So far, the following classifiers have been integrated into the WOLF framework:

1. AdaBoost.
2. Bernoulli Naive Bayes.
3. Gaussian Naive Bayes.
4. Decision Tree.
5. Logistic Regression (LR).
6. Random Forest (RF).
7. C-Support Vector Machine(C-SVM).
8. Linear Support Vector Machine (Linear SVM).
9. Nu Support Vector Machine (Nu SVM).
10. Linear Discriminant Analysis (LDA).
11. Quadratic Discriminant Analysis.
12. Deep Neural Network (DNN).

F. Model Evaluation:

After performing all the important tasks of machine learning, evaluation of results is undertaken. In this step, metric calculation is performed for all possible combinations formed by prior transactions. WOLF consists of metrics like precision, accuracy, area under curve (AUC), Matthews correlation coefficient (MCC), f1-score and many more. The output of this transaction is used for decision making or choosing the best (based on the chosen metrics) predicting combination of algorithms.

G. Model Selection:

This is the moment of truth; the user is provided with a result file consisting of metrics calculated for all possible combinations based on the configuration file. The result file also consists of graph representation of all metrics for each configuration combination which may be further pruned by the user. This gives users the leverage of analyzing the result for appropriate choice of the algorithm which performs best according to their metric selection.

V. WOLF Database Management

In WOLF, the configuration of all the transactions get saved. Also, the metric evaluation results of each configuration get saved into the database. The idea behind integrating database to the project is to make the tracking of each run easier. It also helps to ease the process of result generation after the complete pipeline completion which is then utilized for model selection. As the configuration of each user differs, using relational database could have caused huge waste of space. To conserve space, a non-relational database, MongoDB is used in which only the parameters for a transaction that are specified by users get saved.

The use-case of each collection is:

A. Split Data:

The users configuration for Splitting Data transaction of the project gets saved into this collection. The collections consists of data file information by storing the input file name and also the parameters like number of folds and number of repetitions set by user in the configuration file.

B. Files:

This collection keeps track of the files name that get created after splitting data transaction. It just consists of train file name and test file name for each pair generated. The idea behind this is to keep track of best performing pair if required.

C. Feature Extraction:

The configuration passed for Feature Extraction transaction by user gets saved into this collection. The information stored in the collection consists of, executable (name of algorithm that get executed) and parameters for the chosen executable.

Table I
DATASETS FOR BINARY CLASSIFICATION

Dataset Name	No. of Attributes	No. of samples	Positive / Negative Samples	Percent Positive Samples
Bank note authentication	5	1372	610 / 762	0.445
Blood Transfusion Service Center	5	748	178 / 570	0.237967914
Climate Model Simulation Crashes	19	540	494 / 46	0.914814815
Connectionist Bench (Sonar, Mines vs. Rocks)	61	208	111 / 97	0.533653846
default of credit card clients	24	30000	6636 / 23364	0.2212
Fertility	10	100	12 / 88	0.12
LSVT Voice Rehabilitation	311	126	42 / 84	0.333333333
Pima Indians Diabetes	8	768	268 / 500	0.348958333
Spambase	58	4601	1813 / 2788	0.394044773
Vertebral Column	7	310	210 / 100	0.677419355
Wholesale customers	8	440	142 / 298	0.322727273

Table II
RESULTS GENERATED FOR SPAMBASE DATASET BY WOLF

Algorithm Name	Accuracy	f1 score	mcc	Precision	Recall	roc auc
Random Forest	0.940491943	0.922081025	0.875282549	0.952318939	0.893881254	0.932342067
Decision Tree	0.905932828	0.881324856	0.803675827	0.87639301	0.88665373	0.902562047
AdaBoost Classifier	0.938579417	0.921548713	0.871293486	0.927807641	0.915611007	0.934563288
Deep Neural Network	0.723321129	0.538341427	0.405704617	0.776242927	0.421798091	0.670599051
Bernouli Naive Bayes	0.885960569	0.849226568	0.759761205	0.886503696	0.815392448	0.873621528
Linear Support Vector	0.832480662	0.776085134	0.673030914	0.821482253	0.791323227	0.825286975
Logistic Regression	0.927605137	0.906666538	0.847999572	0.921425047	0.892614492	0.921487069
Linear Discriminant Analysis	0.887568983	0.846374519	0.764242867	0.916240821	0.78682252	0.869953218
Gaussian Naive Bayes	0.821346197	0.808553048	0.676682579	0.700090522	0.957035295	0.845072219

D. Feature Selection:

The feature selection collection provides information about the users configuration for Feature Selection transaction. It consists of executable (name of algorithm that get executed) information and also consists of parameters values provided for chosen algorithm.

E. Algorithm:

The information related to the Model Construction transaction gets stored into this collection. The stored information consists of executable (name of user selected algorithm) and parameters for the selected algorithm.

F. Result:

As the name explains this collection contains output generated by Model Evaluation transaction. All the metrics value that get calculated in evaluation step like mcc, f1-score, precision, and recall etc. gets stored into the collection. The result collection provides moment of truth; it is used to query the results of all pipeline combinations executed by user and generate the report for model selection.

All the collections in WOLF database schema except Result has one to many relationship with Result collection. As Result collection contains the metric evaluation result so it makes sense to refer each result document to the configuration it is related to. The Result collection along with metric evaluation output, consists of unique ids for all the other collections like split data, feature extraction and selection, algorithm. The unique ids of other collections are nothing but the foreign keys that point to the particular configuration used for the generated result.

VI. EXPERIMENTS

The main focus of carrying out the following experiments is to present main features of WOLF framework to compare performance of state of the art machine learning techniques with optimum hyper-parameters. As WOLF is an automated machine learning platform, so a user will only provide data in .arff format along with the configuration file. The user can tell which algorithm to execute in each transaction and the range of hyper-parameters for each algorithm in the configuration file.

A. Binary Classification Tasks

In order to demonstrate applicability of WOLF platform, different families of datasets from UCI repository [5] for binary classification tasks were selected. In order to pass those data sets to WOLF, they were first converted into .arff data format.

Table I provides a list of datasets that has been evaluated by WOLF. The number of attributes in Table I corresponds to feature points in each dataset. The number of feature points are not too big, that's why feature selection or extraction transactions were not used. Also, it is useful to note the total number of samples and ratio of positive samples. For

Table III
DEEP NEURAL NETWORKS RESULTS WITH VARIOUS HYPER-PARAMETERS FOR VERTEBRAL DATASET

Accuracy	Learning Rate	Batch Size	No. Neurons in each layer	roc auc
0.829677419	0.001	200	[100,100,100]	.796761905
0.836129032	0.001	100	[100,100,100,100]	0.797333333
0.410322581	0.01	100	[100,100,100,100]	0.506619048
0.61483871	0.01	100	[100,100,100]	0.510380952
0.831290323	0.001	100	[100,100,100]	0.795595238
0.523870968	0.01	50	[100,100,100]	0.507142857
0.63483871	0.01	200	[100,100,100,100]	0.596380952
0.84	0.001	200	[100,100,100,100]	0.810142857
0.329677419	0.01	50	[100,100,100,100]	0.5
0.809032258	0.001	50	[100,100,100]	0.795666667
0.745806452	0.01	200	[100,100,100]	0.668857143
0.816129032	0.001	50	[100,100,100,100]	0.804309524

Table IV
ACCURACY SCORE FOR ALL BINARY CLASSIFICATION DATASETS

Dataset Name	RF	Linear SVM	DNN	LR	Bernoulli Naive Bayes	LDA	Ada Boost	DT
Bank note auth.	0.9923	0.9889	0.9998	0.9896	0.8419	0.9786	0.996	0.9810
Blood Transfusion	0.6279	0.5323	0.5003	0.5495	0.4993	0.5419	0.6181	0.5852
Climate Sim. Crashes	0.5501	0.7941	0.6581	0.5773	0.5	0.7158	0.7535	0.6527
Sonar, Mines/Rocks	0.8167	0.7692	0.61	0.7507	0.5051	0.7358	0.7954	0.6919
Default of credit card	0.6540	0.5217	0.5	0.4999	0.6731	0.6127	0.6388	0.6081
Fertility	0.5531	0.4960	0.5223	0.4988	0.5	0.4878	0.5375	0.4964
Voice Rehabilitation	0.7831	0.5058	0.6344	0.5529	0.6854	0.7256	0.7916	0.7351
Pima Indians Diabetes	0.7216	0.5597	0.6864	0.7151	0.5035	0.7253	0.7131	0.6412
Spambase	0.9323	0.8252	0.6706	0.9215	0.8736	0.8699	0.9345	0.9025
Vertebral Column	0.8058	0.7261	0.8101	0.8095	0.6438	0.8017	0.7917	0.7592
Wholesale customers	0.9053	0.6939	0.5	0.8765	0.5	0.7748	0.8800	0.8520

a dataset with high percentage of one class of samples, a machine learning model which is biased towards most likely label will give high accuracy. So other evaluation metrics like precision will give better insight in such cases.

Table II provides evaluation metrics generated by WOLF framework for all algorithms that have been applied to Spambase Dataset. There is no dependency among algorithms, so all these algorithms were executed and evaluated in parallel. Evaluation metrics calculated for binary classification are the most popular ones to evaluate algorithm performance. For instance, in some applications it's only desirable to have a high accuracy score, however for critical applications it can also be desirable to have a high precision score. Precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples. It is also worth noting if the machine learning model is predicting randomly or has learned something from training data. For this purpose, (roc_auc) score can give better intuition and model is considered good if the score is close to 1. For Spambase dataset, Random Forest and Ada Boost classifier performed better as compared to other algorithms in terms of accuracy. Table II provides information about which family of algorithms are superior to others for a given dataset. DNN performance is quite poor for this particular application which is evident from roc_auc score which is close to 0.5. The possible reason could be the size of training dataset. In summary, the evaluation metrics for binary classification provided by WOLF framework may help data scientists to develop better intuition of different machine learning techniques.

Table III demonstrates another feature of the WOLF framework, which is to search for optimum hyper-parameters through grid search. However, except for Deep Neural Networks(DNN) grid search is simple and quite efficient to find optimum hyper-parameters. Table III gives details about performance of DNN for Vertebral dataset for a range of hyper-parameters provided by the user in the configuration file. It can be deduced from the table that learning rate plays a vital role in DNN's performance as compare to batch size and depth. This helps data scientists to determine which hyper-parameters plays vital role in performance and give better understanding about DNN for the given problem, which helps in adopting good implementation practices in the future or among research community. More advanced parametric search algorithms, like Bayesian Optimization or active optimization will be integrated into the WOLF framework as part of future work.

Table IV summarizes the results for all datasets mentioned in Table I. It can be seen that the WOLF framework is

Table V
DATASETS FOR MULTICLASS CLASSIFICATION

Dataset Name	No. of Attributes	No. of samples	Classes
Dermatology	35	358	6
Iris	5	150	3
Leaf	16	340	30
Page Blocks Classification	11	5473	5
Wine Quality	12	4898	11

Table VI
ACCURACY SCORE FOR ALL MULTICLASS CLASSIFICATION DATASETS

Algorithm Name	Dermatology	Iris	Leaf	Page Blocks Classification	Wine Quality
Random Forest	0.97	0.929	0.745	0.852	0.412
Linear SVM	0.956	0.934	0.578	0.667	0.0801
DNN	0.613	0.571	0.585	0.119	0.050
Gaussian Naive Bayes	0.843	0.933	0.713	0.495	0.219
LR	0.97	0.93	0.385	0.732	0.210
LDA	0.956	0.964	0.7937	0.669	0.253
AdaBoost	0.48	0.905	0.0853	0.554	0.114
Decision Tree	0.97	0.929	0.745	0.852	0.412

applicable to different families of datasets and automates the process of finding the best machine learning algorithm with optimum hyper-parameters. It can be observed that not a single machine learning model outperform in all datasets. So the user needs an automated platform like WOLF to choose best algorithm for each application. Also, if data scientists develop a new algorithm, it can easily be compared with existing commercial or proprietary algorithms.

B. Multiclass Classification Tasks

For Mutliclass Classification, few datasets were acquired from UCI repository [5] and the details about each data set are provided in Table V. Results generated by WOLF platform are shown in Table VI.

Again it can be observed selection of suitable algorithm is an important step. For instance for Dermatology dataset, performance of AdaBoost classifier is quite poor as compare to other algorithms. Again, in Leaf dataset it's performance degrades severely.

In summary, WOLF framework was able to find optimum algorithms for all datasets except Wine Quality. But it provides a good insight to the data scientist about the dataset in order improve performance.

VII. HYPOTHESIS

From the results described in previous section, following hypothesis can be postulated:

1: Comparison between Shallow and Deep Models: In this experiment, we try to compare performance of deep models with standard shallow machine learning models to develop better understanding about the applicability of deep models. It is evident that deep models do not perform better when data size is small. Also, if ratio of positive to negative samples is not closer to 0.5, DNN models are more likely to over fit.

2. Fine tuning Hyper-Parameters is essential: This is the essential part of getting good performance from a given machine learning model. It is evident from Table III that model's performance drastically suffer with wrong set of hyper-parameters.

3. No standard Machine Learning Algorithm for all applications: It can be observed from Results shown in Table IV, that no single machine learning algorithm can be applied to all datasets. So there will always be a need to find which algorithm is better suited for a particular application.

VIII. FUTURE WORK

In order to make deep learning more effective for smaller datasets, fine tunning of pre-trained models also known as transfer learning [11] needs to be integrated into WOLF. Moreover, grid search for optimum hyper-parameters for DNN requires lots of resources. To address this issue, better parameter optimization techniques like Bayesian optimization [12] or latest techniques like active optimization [13] and learning to learn [14] need to be integrated into the WOLF framework. Efforts are being made to make the WOLF framework publicly available online, where users can upload the project directly to get the results, or contribute to the project.

REFERENCES

[1] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion, Grisel, Duchesnay E. Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, (2011), 2825-2830.

[2] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rckstie, and J. Schmidhuber. PyBrain. JMLR, 2010.

[3] Sebastian Raschka, Python machine learning : unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics, 2015.

[4] C. Thornton, F. Hutter, H. Hoos, K. Leyton-Brown Auto-weka: combined selection and hyperparameter optimization of classification algorithms Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2013), pp. 847-855.

[5] Lichman M. , UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.

[6] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Ktter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, Bernd Wiswedel. The Konstanz Information Miner (KNIME). Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization. Springer Berlin, Heidelberg (2008)

[7] Jim Pivarski, Collin Bennett, Robert L. Grossman. Deploying Analytics with the Portable Format for Analytics (PFA). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016, Pages 579-588.

[8] Bertram Ludscher et al., Scientific workflow management and the Kepler system: Research Articles. Concurrency and Computation: Practice & Experience - Workflow in Grid Systems archive Volume 18 Issue 10, August 2006, Pages 1039-1065

[9] Randal S. Olson and Jason H. Moore,TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning, JMLR: Workshop and Conference Proceedings 64:6674, 2016.

[10] Tim Kraska, Ameet Talwalkar, John Duchi, Rean Griffith, Michael Franklin, Michael Jordan. MLbase: A Distributed Machine-learning System,6th Biennial Conference on Innovative Data Systems Research (CIDR) 2013.

[11] Qiang Yang, Sinno Jialin Pan, "A Survey on Transfer Learning", IEEE Transactions on Knowledge & Data Engineering, vol. 22, no. , pp. 1345-1359, October 2010.

[12] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012.

[13] Kirthevasan Kandasamy, Jeff Schneider, Barnabas Poczos, "Query Efficient Posterior Estimation in Scientific Experiments via Bayesian Active Learning", Artificial Intelligence Journal (AIJ) 2017.

[14] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, Nando de Freitas, "Learning to learn by gradient descent by gradient descent", 2016.