**ML1819 Research Assignment 2**

Team 02

Task 101

Rajesh Burla - 18306485

Akashdeep Singh Lamba - 18305063

Shanmukha Sai Ram Pavan - 18305688

Each student picked up one library and implemented the three selected algorithms end-to-end in the assigned library, while constantly communicating progress over a Slack channel. We each maintained shared git repos to share code and collectively work. All three of us then compared our results and then discussed the contents of this document. We realized we were supposed to work in a single repo so we migrated our code to a shared repo in a group with write access to only the three of us. This also means the contributor graph isn’t a perfect reflection of our activity. One of us is less comfortable with git so he has fewer commits to his name.

Word count: ####

<https://github.com/akashdeep-singh/ML1819--task-101--team-02>

<https://github.com/akashdeep-singh/ML1819--task-101--team-02/graphs/contributors>

###screenshot###

Survey of popular Machine Learning Libraries

A comparative analysis of Tensorflow, Sklearn and Weka by implementation of standard algorithms

Akashdeep Singh Lamba  
MSc Computer Science - Intelligent Systems (2018-19)  
 Trinity College Dublin  
 Dublin, Ireland  
 [lambaa@tcd.ie](mailto:lambaa@tcd.ie)

Rajesh Burla  
MSc Computer Science - Intelligent Systems (2018-19)  
 Trinity College Dublin  
 Dublin, Ireland  
[burlar@tcd.ie](mailto:burlar@tcd.ie)

Shanmukha Sai Ram Pavan   
MSc Computer Science - Intelligent Systems (2018-19)  
Trinity College Dublin  
 Dublin, Ireland  
 [parvarths@tcd.ie](mailto:parvarths@tcd.ie)

ABSTRACT

This report presents a comparative analysis of linear regression, SVM classification and kNN across 3 machine learning algorithms.

1 INTRODUCTION

The growing interest in machine learning as led to increase in popularity of open source libraries, and each has a different selling point, viz. training speed, accuracy, ease-of-use, flexibility, platform support or GPU support.

It’s worth asking whether all libraries perform similarly under approximately same conditions or not, and this paper attempts to address this question.

2 RELATED WORK

Inspection of the experiments of Bhuvan M Shashidhara et al. in [1] the results shows that Scikit-Learn is best fit for data in comparison with Weka and Apache Spark frameworks.

Scrutinizing results of the Google Brain team [2] reveals that TensorFlow is a flexible dataflow representation that enables power users to achieve excellent performance and scalability.

3 METHODOLOGY

In order to answer the research question, three Machine learning algorithms and frameworks were chosen to be applied to two pre-processed datasets and resultant metrics such RMSE and accuracy were compared.

3.1 Dataset

*i) Google Play Store Apps*

This is web-scraped data of 10k Play Store apps. This dataset contains all the details of the applications on Google Play. There are 13 features that describe an individual app. [3]

*ii) Car Insurance Cold Calls*

This data consists of 4000 training data points and 1000 test data points of car insurance cold call results. This dataset requires very little cleaning, and is used to teach entry level data mining at TUM. [9]

3.2 Data Pre-processing

*i) Google Play Store Apps*

The dataset in the original form obtained from Kaggle wasn’t fit for direct use. In order to make it suitable for the experiment, sparse columns were eliminated. Further, string columns *Category* and *Genres* were encoded as numbers.

We also added a new column called *Rated 4.4 or more* derived from the column *Rating* for use with SVM. *Rated 4.4 or more* was defined as 1 if the *Rating* column is equal to or greater than 4.4, and -1 otherwise. The resultant classes were nearly equally distributed.

*ii) Car Insurance Cold Calls*

The car insurance dataset requires less processing as there are few empty fields. Outliers were removed by visualizing the *Balance* column. *Job* and *Education* columns were forward filled, while *Communication* and *Outcome* were replaced with “none”.

Scaling was left to implementation specific code since different implementations treat data differently.

3.3 Libraries

**i) Weka**: Weka consists of a GUI and a programmable library to provide data mining and machine learning features. It’s written in Java. It contains robust sequential implementations of many machine learning algorithms. [10]

**ii) Scikit-Learn**: Sklearn this is a Python module consisting of a library of a wide-range of machine learning tools and algorithms, both supervised and unsupervised, and uses in-memory computation for fast processing on medium-scale workloads. [11]

**iii) TensorFlow**: TensorFlow is a flexible Python framework for building fast and complex machine learning models specifically targeted for deep learning and neural networks. TensorFlow receives data in the form of Tensors, which are arrays of dimensions and ranks. It supports distributed execution over GPUs and CPUs. [12]

The selection of these three implementations was done based of popularity as reported by [4] and [5]. Including Weka helped us make sure our study was not limited to Python-based implementations. While this isn’t an exhaustive list of implementations, it’s a good starting point.

3.4 Machine learning Algorithms

Three machine learning algorithms were chosen: Linear Regression, Support Vector Machines (SVM) and k-Nearest Neighbour (kNN) to test Root Mean Square Error (RMSE) and Accuracy on the pre-processed dataset using the selected frameworks.

3.5 Feature selection

*i) Google Play Store Apps*

For Linear Regression, the feature *Reviews* was used to predict *Rating* (Figure.1).



Figure 1: Plot of Rating v/s Reviews

For SVM, *Reviews* was used to predict *Rated 4.4 or more* (Figure 2).



Figure 2: Plot of Review v/s Rated 4.4 or more

For kNN, *Reviews*, *Size*, *Genres* was used to predict *Category*. Figure 3 shows the correlation between the selected features.



Figure 3: Correlation between the selected features of Google play dataset

*ii) Car Insurance Cold Calls*

For Linear Regression, the feature *NoOfContacts* was used to predict *LastContactDay* (Figure 4)



Figure 4: Plot of LastContactDay v/s NoOfContacts

For SVM, *Balance* was used to predict *CarLoan* (Figure 5)



Figure 5. Plot of Balance v/s CarLoan

For kNN, *Age, Balance, NoOfContacts, CallDurationMinutes* was used to predict *Marital* Figure 6 shows the correlation between the selected features



Figure 6: Correlation between the selected features of Car Insurance dataset.

Feature selection was done by manually scanning the datasets and reasoning about which features are well-suited for the given model and features that were skewed or sparse were dropped.

3.6 Cross validation

While Weka provides an in-built option to cross-validation, thereby making splitting of data and configuration of CV parameters internal, sklearn requires to explicit configuration of kFold validation with a split dataset. TensorFlow does not directly deal with any of these aspects. For TensorFlow and sklearn, we used sklearn’s kFold validation with default parameters while using a random-indexed splitting between train data and test data, keeping a ratio of 7:3.

4 RESULTS & DISCUSSION

The experiment was run for 10 different *k*-folds and the final results are sum of the calculated mean and standard deviation.

For linear regression, as seen in Fig. 7a, and Table 1a, RMSE (Root mean squared error) for the GooglePlay dataset is about 0.51 for all the chosen frameworks. Whereas, for CarInsurance dataset, as seen in Fig. 7b and Table 1b, TensorFlow and Weka performed almost same with RMSE of ≈ 0.832 while Sklearn had a RMSE of 0.838. The lower the RMSE is better.

**Table 1a: Mean and Standard Deviations of Linear Regression RMSE for GooglePlay dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sklearn | Tensor Flow | Weka |
| Mean | 0.514077 | 0.515105 | 0.51413 |
| Standard Deviation | 0.01213 | 0.014018 | 0.01232 |

Result for each iteration is shown in: Appendix Table 10a.

Figure 7a: RMSE for Linear Regression of GooglePlayAppStore dataset.

**Table 1b: Mean and Standard Deviations of Linear Regression RMSE for Car Insurance dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sklearn | TF | Weka |
| Mean | 8.386618 | 8.346447 | 8.32825 |
| Standard Deviation | 1.78E-15 | 0.064973 | 0.001376 |

Result for each iteration is shown in: Figure 7b and Appendix Table 10b.

Figure 7b: RMSE for Linear Regression of CarInsurance dataset

In case of SVM, accuracy as the metric was chosen to report given that SVM results are binary and comparing MSE or RSME doesn’t make sense. Here, accuracy is defined as the number of correctly predicted outcomes over the total number of predictions. Fig. 8a, 8b and Table 2a, 2b, shows that accuracy stood at about 0.57 and 0.87 for the both datasets respectively with all the selected frameworks. The accuracy is significantly equal across the chosen frameworks.

**Table 2a: Mean and Standard Deviations of SVM Accuracy for GooglePlay dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sklearn | TF | Weka |
| Mean | 0.5736 | 0.577825 | 0.573102 |
| Standard Deviation | 0.003904 | 0.009233 | 0.007195 |

Result for each iteration is shown in: Figure 8a and Appendix Table 11a.

Figure 8a: Accuracy for SVM of GooglePlayAppStore dataset

**Table 2b: Mean and Standard Deviations of SVM Accuracy for Car Insurance dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sklearn** | **TF** | **Weka** |
| **Mean** | 0.8696 | 0.871917 | 0.872869 |
| **Standard Deviation** | 0.006545 | 0.007768 | 0.007176 |

Result for each iteration is shown in: Figure 8b and Appendix Table 11b.

Finally, for kNN, we again chose RMSE to represent performance and from Fig. 1c, it’s apparent that there isn’t much difference as RMSE was about 1.8 for sklearn and TensorFlow, while a tad lower at 1.78 for Weka.

The overall low difference in results seems to suggest that the three implementations are more or less consistent with minor differences. It may be noted that the difference in the level of abstraction also effects how the tests are carried out. To test Weka, the algorithms were implemented in Java code and also verified the results with Weka’s own GUI, which hides implementation details. Sklearn exposes an easy API through which we can configure the model and execute it on our data. TensorFlow allows much more flexibility as it provides building blocks for defining the model and executing it on the given primitives.

5 LIMITATIONS & OUTLOOK

While the experiment conducted within the prevailing constraints provides an overview of comparison between the three libraries, it’s by no means a comprehensive analysis. Metrics such as training speed haven’t been considered, and more sophisticated algorithms haven’t been tested. Further, frameworks like Apache Spark’s MLib [7] and Orange [8] should be part of such a study of machine learning implementations, and future work on this project will focus on addressing these shortcomings.

ACKNOWLEDGMENTS

This analysis was conducted as part of the 2018/19 Machine Learning module CS7CS4/CS4404 at Trinity College Dublin [6].

REFERENCES

[1] Bhuvan M Shashidhara et al. “Evaluation of Machine Learning Frameworks on Bank Marketing and Higgs Datasets” IEEE 2015 DOI: 10.1109/ICACCE.2015.31

[2] The team Google Brain’s “TensorFlow: A system for large-scale machine learning” 12th USENIX Symposium on Operating Systems Design and Implementation in 2016.

[3] Lavanya Gupta, "Google Play Store Apps", [Online] https://www.kaggle.com/lava18/google-play-store-apps

[4] Datanyze, "Machine Learning Market Share Report | Competitor Analysis | TensorFlow, scikit-learn, MLlib," [Online] https://www.datanyze.com/market-share/machine-learning

[5] Joeran Beel, "Experience with and Preference of Machine-Learning Libraries: scikit-learn vs. Tensorflow vs. Weka … [What Machine-Learning Students Think/Like/Know/Are…]," [Online] https://www.scss.tcd.ie/joeran.beel/blog/2018/01/28/what-machine-learning-students-think-like-know-are-experience-with-and-preference-for-machine-learning-libraries-scikit-learn-vs-tensorflow-vs-weka/

[6] Joeran Beel and Douglas Leith. Machine Learning (CS7CS4/CS4404). Trinity College Dublin, School of Computer Science and Statistics. 2018.

[7] Xiangrui Meng et al. MLlib: Machine Learning in Apache Spark, Journal of Machine Learning Research 17 (2016) 1-7

[8] Demsar J, Curk et al. Orange: Data Mining Toolbox in Python. Journal of Machine Learning Research 14(Aug):2349−2353., https://orange.biolab.si/citation/

[9] GregKondla, “Car Insurance Cold Calls”, [Online] https://www.kaggle.com/kondla/carinsurance

[10] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[11] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[12] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.