ML1819 Research Assignment 2

Team 02
Task 101

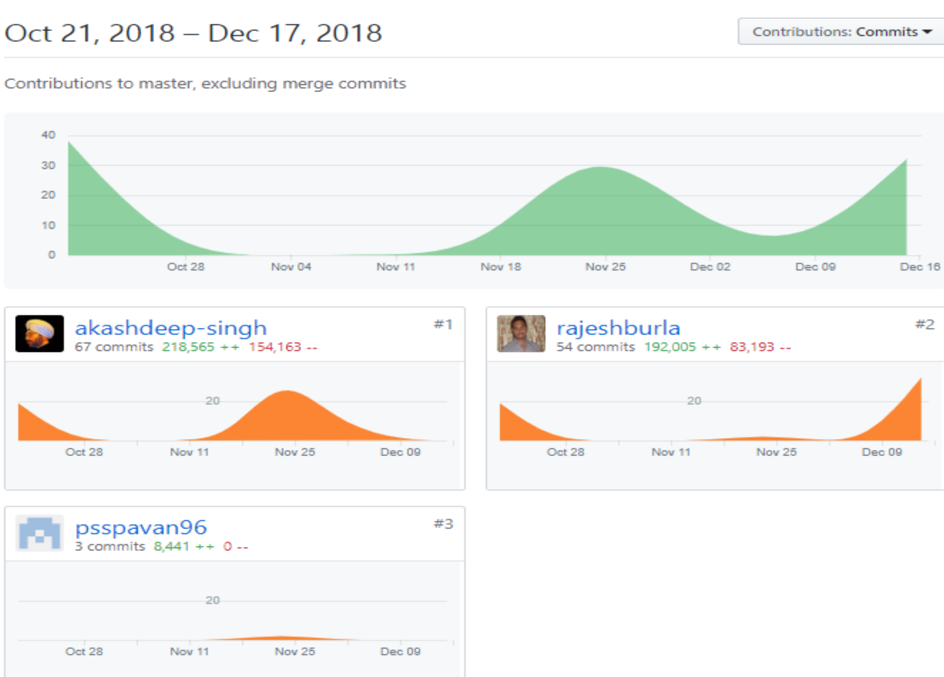
Rajesh Burla - 18306485
Akashdeep Singh Lamba - 18305063
Shanmukha Sai Ram Pavan - 18305688

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: ~1450

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

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



Survey of popular Machine Learning Libraries

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

Akashdeep Singh Lamba
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
lambaa@tcd.ie

Rajesh Burla
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
burlar@tcd.ie

Shanmukha Sai Ram Pavan
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
parvarths@tcd.ie

ABSTRACT

This report presents a comparative analysis of linear regression, SVM classification and k-Nearest Neighbor classification across 3 machine learning algorithms.

1 INTRODUCTION

The growing interest in machine learning as led to increase in popularity of open source implementations. Each implementation has a different selling point, viz. training speed, accuracy, ease-of-use, flexibility, platform support or GPU support. Some of these are available as pluggable libraries, while some ship as fully-featured frameworks.

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

In the rest of this document, the words library and framework are used interchangeably, although there are technical differences between the terms which are not relevant in the context of this research.

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 3 three frameworks were chosen to be applied to two pre-processed datasets and resultant metrics such RMSE and accuracy were compared. The particulars of the methodology are as follows:

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 supervised model-based 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*. Fig. 1 shows a scatterplot between these two features.

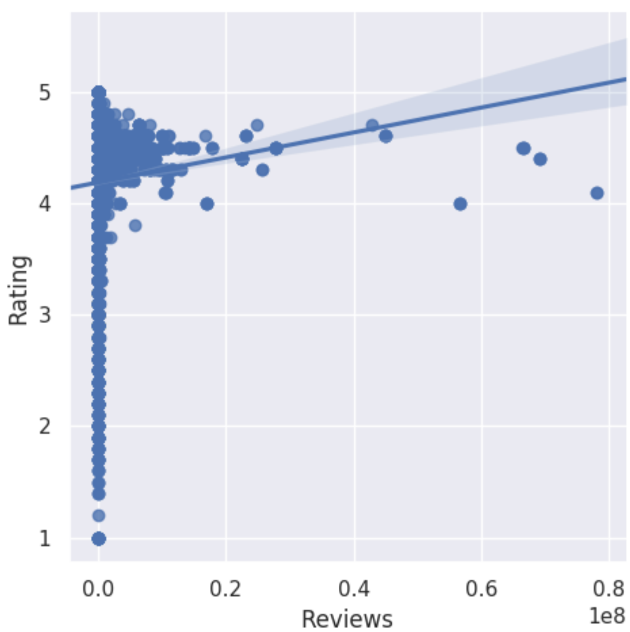


Figure 1: Plot of Rating v/s Reviews

For SVM, *Reviews* was used to predict *Rated 4.4 or more.* Fig. 2 shows a category plot between these two features.

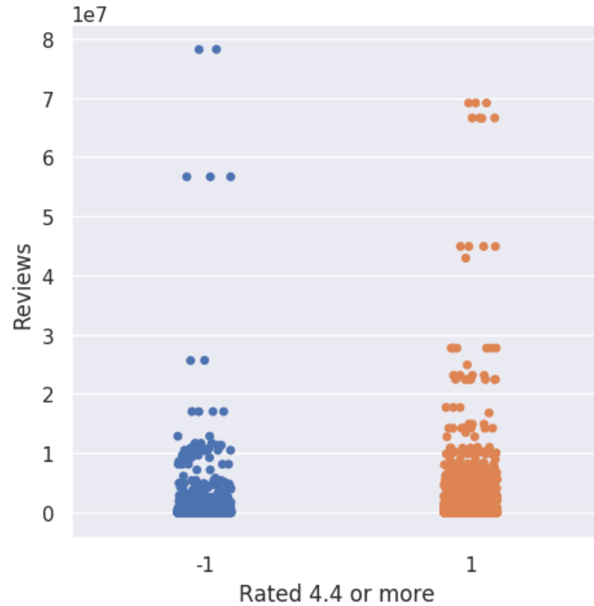


Figure 2: Plot of Review v/s Rated 4.4 or more

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

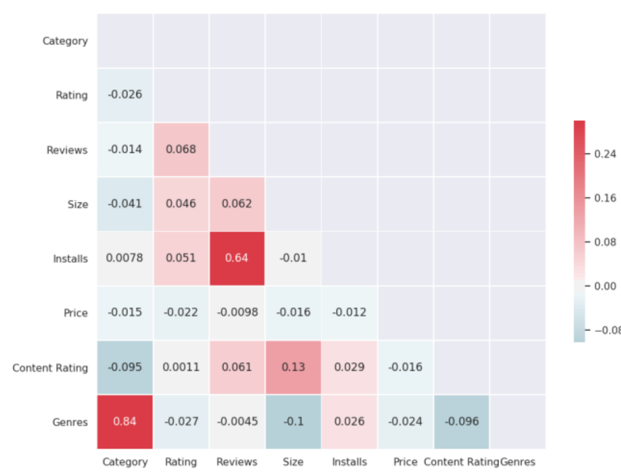


Figure 3: Heatmap of the Google play dataset

ii) Car Insurance Cold Calls

For Linear Regression, the feature *NoOfContacts* was used to predict *LastContactDay*. Fig. 4 shows a scatterplot between these two features.

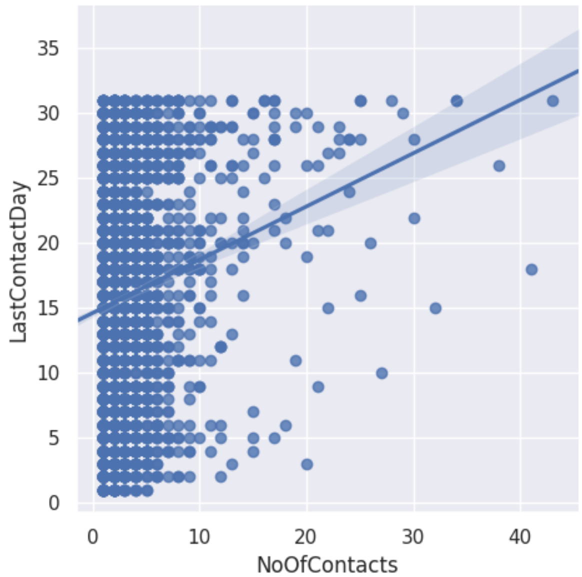


Figure 4: Plot of LastContactDay v/s NoOfContacts

For SVM, Balance was used to predict CarLoan. Fig. 5 shows a category plot between these two features.

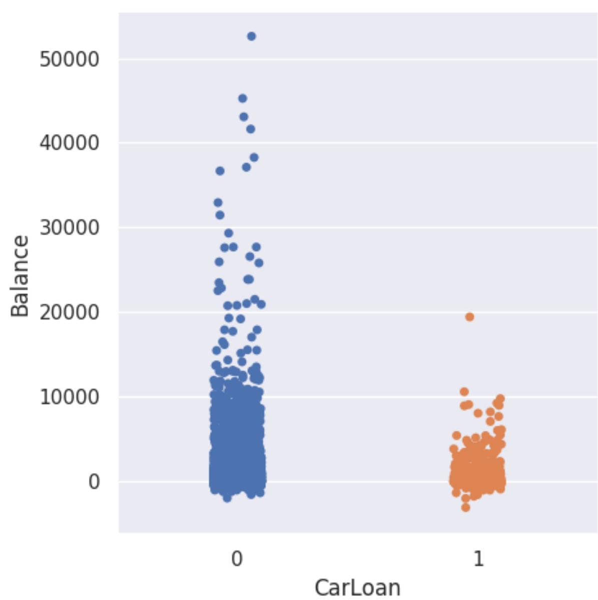


Figure 5: Plot of Balance v/s CarLoan

For kNN, *Age, Balance, NoOfContacts, CallDurationMinutes* were used to predict *Marital* Figure 6 shows the correlation between all the 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.

3.7 Metrics

For linear regression, RMSE (root mean squared error) is reported since regression is being used to predict real-valued labels. A lower RMSE is better.

For SVM and kNN classifiers, accuracy is reported since results are classes 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. Accuracy ranges between 0 and 1, with values closer to 1 being more accurate and those closer to 0 being very inaccurate.

For binary classification on uniformly distributed datasets, accuracy of around 0.5 can be achieved using a trivial model that always predicts the same class. This implies that a binary classification model with an accuracy close to 0.5 is performing poorly.

We are not particularly concerned with achieving high accuracy or low RMSE, and more concerned with finding how different libraries perform for answering the research question.

4 RESULTS AND DISCUSSION

In order to establish statistical significance and reproducibility, the experiment was executed for each combination of library, algorithm, and dataset 10 times and the final results are summary of the calculated mean and standard deviation. The graphs also show confidence interval estimates at 95% confidence level. The detailed results of all executions are included in Appendix A. Summarized results and the graphs are compared here.

For linear regression, as seen in Fig. 7, and Table 1, RMSE (root mean squared error) for the GooglePlay dataset is about 0.51 for all the chosen frameworks.

On the other hand, for CarInsurance dataset, as can be seen in Fig. 8 and Table 2, TensorFlow and Weka performed almost same with RMSE ~0.832 while Sklearn had a RMSE of 0.838.

Table 1: 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

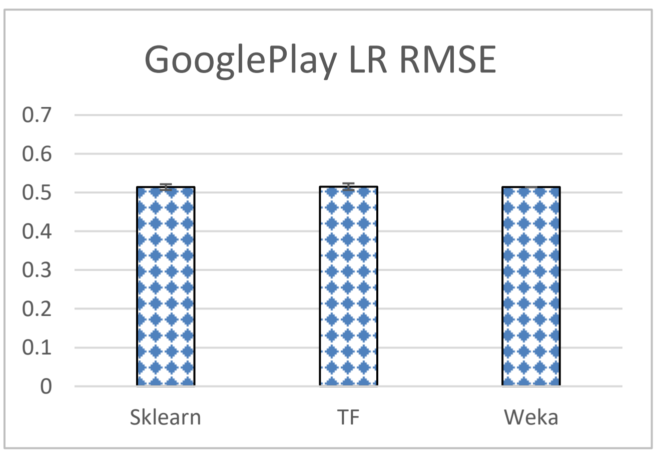


Figure 7: RMSE for Linear Regression of GooglePlay dataset

Table 2: Mean and Standard Deviations of Linear Regression RMSE for Car Insurance dataset

	Sklearn	TF	Weka
Mean	8.307902272	8.346447	8.32825
Standard Deviation	0.071298395	0.064973	0.001376

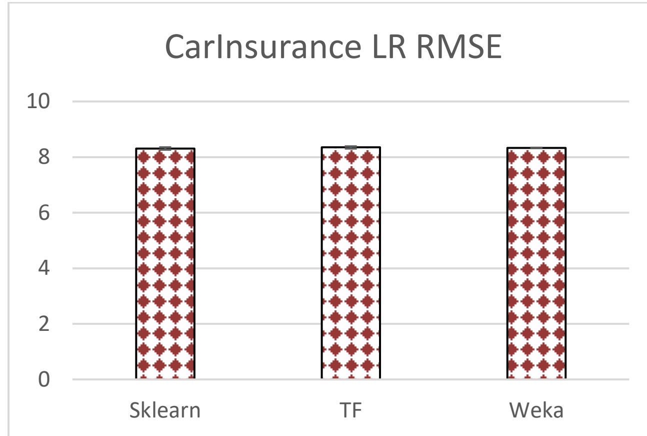


Figure 8: RMSE for Linear Regression of CarInsurance dataset

In case of SVM, Fig. 9, 10 and Table 3, 4, shows that accuracy stood at nearly 0.57 and 0.87 for the Google Play dataset and the Car Insurance dataset, respectively with all the selected frameworks. The accuracy is remarkably similar across the chosen frameworks.

Table 3: 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

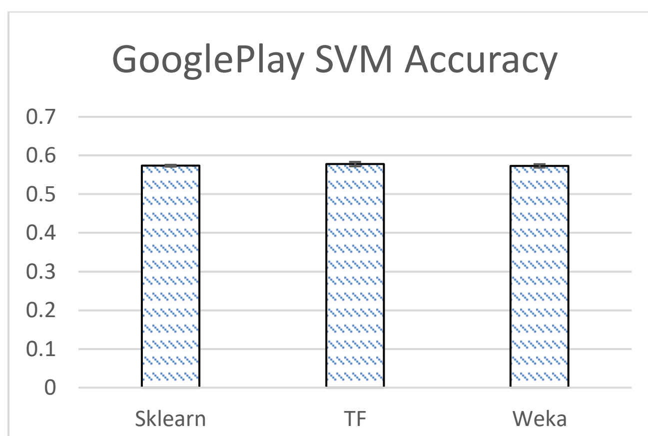


Figure 9: Accuracy for SVM of GooglePlay dataset

Table 4: 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

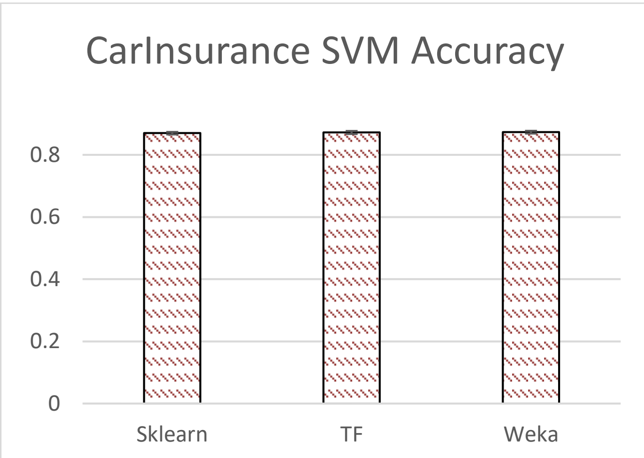


Figure 10: Accuracy for SVM of CarInsurance dataset

Finally, for kNN, Accuracy was chosen to represent performance and from Fig. 11 and Table 5, it shows that, for GooglePlay dataset the accuracy was similar for all the frameworks at ~0.70.

But, for the CarInsurance dataset, as shown in Fig 12 and Table 6, the accuracy is higher with 0.61 for Tensor Flow, Weka gave an accuracy of 0.60 whereas Sklearn performed least with the accuracy of 0.57.

Table 5: Mean and Standard Deviations of kNN Accuracy for GooglePlay dataset

	Sklearn	TF	Weka
Mean	0.702171	0.705827	0.70734
Standard Deviation	0.008014	0.006158	0.003104

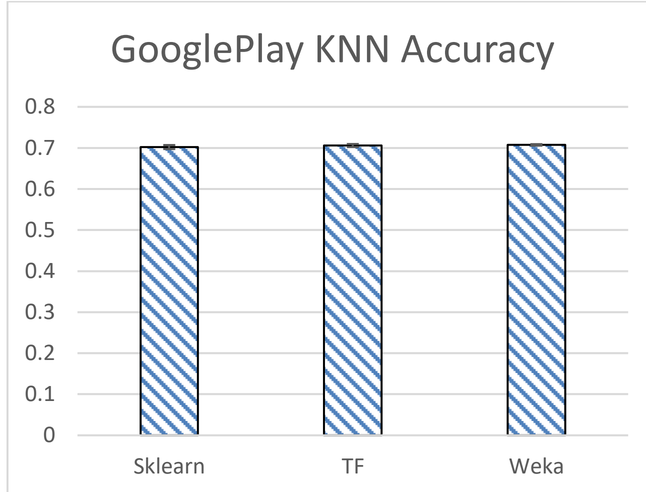


Figure 11: Accuracy for kNN of GooglePlay dataset

Table 6: Mean and Standard Deviations of kNN Accuracy for Car Insurance dataset

	Sklearn	TF	Weka
Mean	0.5704	0.611447	0.605781
Standard Deviation	0.007672	0.010199	0.002663

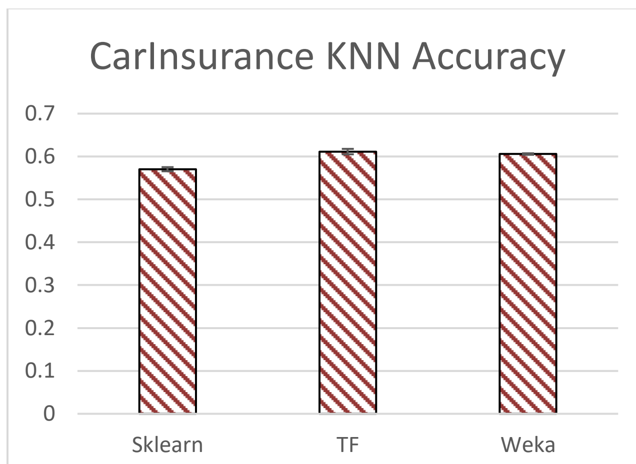


Figure 12: Accuracy for kNN of CarInsurance dataset

The overall low difference in results seems to suggest that the three implementations are more or less consistent with minor differences which can be attributed to platform level differences such as datatypes and precision. 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 results were also verified 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

The conducted experiments offer an overview of comparison between the three libraries across 3 supervised learning algorithms for two datasets. It's by no means a comprehensive analysis. Metrics such as training speed haven't been considered, and more sophisticated algorithms such as unsupervised learning and neural networks 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 should 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.

APPENDIX A

Tabulated results of all individual experiments.

Table 7: 10 instances of RMSE score of Linear Regression on GooglePlay Dataset

Sklearn	TF	Weka
0.525693	0.502069	0.5141
0.513975	0.496161	0.5142
0.510839	0.515049	0.5141
0.522442	0.521769	0.5141
0.500694	0.510616	0.5141
0.521043	0.535021	0.5142
0.530169	0.54003	0.5141
0.514249	0.522553	0.5141
0.515304	0.499053	0.5142
0.486367	0.508731	0.5141

Table 8: 10 instances of RMSE score of Linear Regression on Car Insurance Dataset

Sklearn	TF	Weka
8.459968442	8.337824	8.328
8.392925859	8.305751	8.3304
8.206481983	8.337289	8.3268
8.282365585	8.341064	8.3286
8.276309056	8.272788	8.3274
8.270198995	8.262894	8.3273
8.367282618	8.352019	8.3273
8.27854621	8.328611	8.3311
8.28308333	8.477151	8.3271
8.26186064	8.449074	8.3285

Table 9: 10 instances of Accuracy score of SVM on GooglePlay Dataset

Sklearn	TF	Weka
0.573	0.582482	0.569507
0.576	0.571627	0.568652
0.568	0.583911	0.578542
0.575	0.569016	0.565438
0.581	0.582463	0.563453
0.568	0.577475	0.576543
0.573	0.556839	0.576534
0.573	0.590442	0.589437
0.578	0.578692	0.573407
0.571	0.585298	0.569507

Table 10: 10 instances of Accuracy score of SVM on Car Insurance Dataset

Sklearn	TF	Weka
0.866	0.875768	0.869374
0.863	0.871261	0.869454
0.872	0.863785	0.870451
0.861	0.887818	0.889256
0.865	0.872975	0.869454
0.867	0.857927	0.879304
0.88	0.865145	0.864534
0.879	0.875585	0.879084
0.877	0.874154	0.872344
0.866	0.874752	0.865434

Table 11: 10 instances of Accuracy score of kNN on GooglePlay Dataset

Sklearn	TF	Weka
0.714235	0.704748	0.7107
0.701068	0.703756	0.7045
0.704982	0.709692	0.7078
0.695018	0.708415	0.7054
0.698932	0.712771	0.7045
0.714235	0.715049	0.7077
0.6879	0.698966	0.7045
0.708541	0.692759	0.7085
0.700356	0.705369	0.7053
0.696441	0.706743	0.7145

Table 12: 10 instances of Accuracy score of kNN on Car Insurance Dataset

Sklearn	TF	Weka
0.579333	0.61017	0.606321
0.578	0.594754	0.605121
0.582667	0.597015	0.606921
0.567333	0.599172	0.60072
0.562	0.61625	0.602721
0.564667	0.618298	0.606521
0.561333	0.621789	0.607321
0.572667	0.615486	0.605321
0.574667	0.616648	0.605521
0.561333	0.624889	0.611322