Benchmarking Machine Learning Platforms and Implementing a Dynamic Recommendation System

Jinyang Yu (jy2803@columbia.edu)
Tianrui Peng (tp2522@columbia.edu)

I. Introduction

Machine learning is currently under a dramatic growth, in academia, industry, and individual developers. Besides the exciting research projects and commercialized products, Machine Learning as a Service (MLaaS) has received a lot of attention by both customers and technology companies[1]. More and more companies or individual developers want to incorporate mature machine learning techniques, such as classification or clustering, into their own products; on the other hand, they would also like to avoid the hassle of getting expertise in machine learning and diving into complicated machine learning coding. With this motivation, many companies start to develop their own Machine Learning as a Service Platform to provide scalable machine learning services online[2]. Even though the large number of machine learning platforms available nowadays tend to serve purposes different from one to the other, ranging from customer behavior prediction[3] to demographic analysis[4], there are still some popular machine learning platforms that have the same targeted audience and share the similar functionalities. In the case of this project, we focused on machine learning platforms that are considered as “general platform”, which provides general classification and clustering algorithms to users who have less machine learning knowledges. The reason these platforms do not provide algorithms other than classification and clustering is that, these two algorithms are the most widely used solutions to most of the machine learning problems people are trying to solve. We also find that there are a very large number of machine learning platforms that serve some very specific purposes that indirectly provide algorithms other than classification and clustering. These algorithms are not as widely used as the two algorithms, and are usually designed for a very specific task. For example, Google Prediction API[21] is specifically designed for NLP task which utilizes Recurrent Neural Network. Also, Baidu Face[22] targets at people who want to use face recognition service and it is based on using Convolutional Neural Network. In our project, we want to focus on some general platforms which can handle most of the datasets. This is the reason we choose to only analyze platforms with classification and clustering algorithms.

The question then arises: which one should I use for my experiment? People have tried to answer this question by posting third-party reviews or blogs online. However, according to our literature search, most of such articles inevitably analyze these platforms from a machine learning aspect, which we find are particularly hard to read if the reader has less or none machine learning background. In [5][6], both authors mentioned a large amount of techniques terms in machine learning, when they are comparing platform-to-platform. For example, [6] has a table just compares the machine learning algorithms between the platforms, such as gradient descend, batch-size, etc. These comparisons are certainly very helpful for machine learning researchers, but they provide insufficient information to certain groups of people, whom we have described in the previous paragraph.

In this project, we explore the many aspects of some well recognized machine learning platforms, specifically, we focus on Amazon Machine Learning[7], Microsoft Azure Machine Learning Studio[8], and BigML[9]. The key assumption is that we shall analyze them from a solely software engineering aspect. That is, our targeted audience of this analysis are software engineers who may or may not have some naive knowledge of machine learning. Our project is composed of two connected component: the first part is to benchmark and analyze the targeted machine learning platform. We will include all of our findings in this report. After reading this report, the readers should be able to understand the high level differences and similarities among the designated platforms. For the second part of the project, we proposed a dynamic recommendation system, *MLPRec*, to recommend people who have none or less machine learning backgrounds with the best platform for their application or experiment. In *MLPRec*, the user needs to fill in a survey which consists of about ten questions, and the platform will automatically recommend the most suitable platform for the user as well as the reasons. This recommendation is based on our findings described in the first component. We carefully designed a scoring metric which quantifies our benchmarking result. Besides receiving recommendation, we also added a feedback feature in *MLPRec*. The users can add their own user experiments with respect to one machine learning platform to the system and these ratings will be used in the recommendation scoring metric. This is why we called our recommendation system “dynamic”: instead of a fixed, one-time evaluation metric, it can be dynamically modified by receiving feedbacks from users.

Overall, we believe we have made the following unique contributions in our project(**Deliverables**):

1. We comparatively analyzed three popular machine learning platforms from a software engineering perspective. These can be used as a guideline for people who have less machine learning background to select the most suitable platform.
2. We proposed and implemented a dynamic recommendation system, *MLPRec,* for machine learning platforms. In the prototype, the recommendation scoring metric is based on our analysis in the first component. This system can also take in feedbacks from platform users and the feedback will be used during the recommendation process.

The rest of this report is organized as the follow: Section II will describe all of our finding in comparatively analyzing the three machine learning platforms. Then, We will introduce *MLPRec* in Section III. In Section IV, we will discuss the contribution of each teammate, what we learned from doing this project, . In Section V, we will conclude and summarize our project.

II. Machine Learning Platform Analysis

We evaluated the three designated machine learning platforms from many software engineering aspects. We now present all of our finding in this section.

1. Machine Learning Knowledge

In this section, we will describe the machine learning techniques in these platforms. First we will quantitatively look at the amount of different machine learning algorithms each platform supports. Then we will qualitatively evaluate how much machine learning background do they require from users in order to use their services.

a. Algorithm supportability

In our analysis, we look through the platform user interface, documentations, as well as tutorials in order to find all supporting algorithms for each platform. Microsoft significantly outperforms the other two platforms by supporting 8 different classification algorithms and 2 clustering algorithms. These algorithms are shown in Table 1. This can be considered as both advantage and disadvantage, as we will explain in the next subsection. For BigML, it has one algorithm for classification and clustering; while Amazon only has one algorithm for classification. Clustering algorithm is reported as "to-be-implemented" feature according to their official discussion forum[10].

According to our finding, it's easy to notice that Microsoft > BigML > Amazon, in terms of supporting different machine learning algorithms.

b. Machine learning background

Based on part a, we can see Amazon only provides one machine learning algorithm to perform classification. The parameters for this algorithm is also very limited. So we believe Amazon Machine Learning requires the least amount of machine learning background in order to use the service. A user only needs to upload the dataset onto the platform, and he will simply use the only option provided by Amazon to receive the prediction result.

As for Microsoft, it provides more than 10 algorithms for both classification and clustering, which is much more than what Amazon provides. Within each algorithm, it also provides almost every parameter that a normal machine learning library is able to provide, such as Scikit-learn[12]. This can be an advantage as well as a disadvantage. The advantage is that a user with lots of machine learning background can carefully select the best algorithm for the experiment and finetune the model by adjusting the parameters. However, this is also a huge disadvantage. A user with no machine learning algorithm will have a hard time to get started, because these algorithms, such as "Support Vector Machines", or "Neural Networks" are somewhat meaningless to them comparing to terminology such as "classification" or "clustering".

We believe BigML provides an intermediate option between Amazon and Microsoft. For naive users, BigML provides a simple “one-click classification” algorithm, which is very similar to what Amazon provides. User will only need to upload the dataset and run this algorithm to receive predictions. For experienced machine learning users, BigML also provides some advanced algorithm and lots of tuning parameters. It is not as extensive as Microsoft but still quite flexible comparing to Amazon.

So overall, we think Amazon requires the least machine learning background, followed by BigML and Microsoft.

2. Performance

In this section, we conduct experiments using the same dataset on the three platforms. The dataset we used is the official MNIST dataset[11]. We pre-processed the raw dataset by converting the ubyte format to csv format. We also add one extra column for labels after the feature columns. The pre-processed datasets are slightly different from each other as each platform has its own input standard. All of the raw experiment data can be found in **Appendix A**.

a. Experiment Processing Time

We first exam the processing time of each platform. **Table 1** shows the experiment we conduct. We repeatedly run the experiments five times on two different machines and take the average of them, except for the Amazon experiment[1]. We can see that Microsoft’s Decision Forest algorithm has the shortest processing time among all of the algorithms. Note that since we have both the runtime and dataset size, we can simply calculate the speed for each algorithm in (MB/s). We will further discuss this concept later in this report.

Platform	algorithm	Runtime (seconds)	accuracy
Microsoft	Naive Bayes	3480	0.908343
Microsoft	Decision Forest	71	0.945229
Microsoft	Decision Jungle	**55**	0.8696
Microsoft	Logistic Regression	62	0.919943
Microsoft	Neutral Networks	190	**0.971371**
Microsoft	Support Vector Machine	101	0.8976
Microsoft	Average Perception	140	0.913

[1] The reason is each experiment on Amazon approximately takes $1 to compute (no free credit).

Microsoft	Boosted Decision Tree	159	0.964114
Amazon	Multiclass Classification model	1440	0.926
BigML	decision tree	69	0.8382

Table 1: MNIST dataset on different machine learning algorithms.

b. Experiment Accuracy

Table 1 also shows the accuracy of each algorithm. Microsoft's neural network has the best accuracy among all of the algorithms.

c. Scalability

In this subsection, we tested the scalability for each platform. We divided our dataset(70,000 data points) in the following manner: 1k, 2k, 4k, 8k, 16k, 32k, and 70k. We expect to see a close to linear growth for the processing time vs dataset size. We plotted our experiments in **Figure 1, 2, and 3**. Note for Microsoft, we choose to use the neural network algorithm since it has the best accuracy.

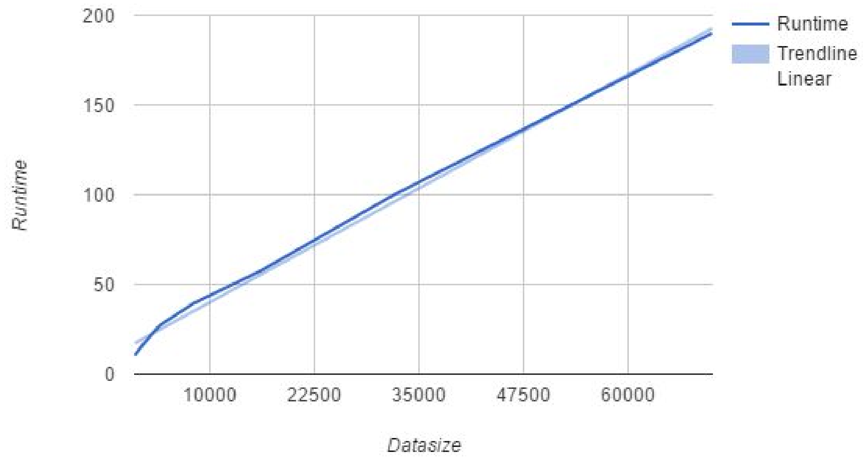


Figure 1. Microsoft Runtime(s) vs Datasize(points)

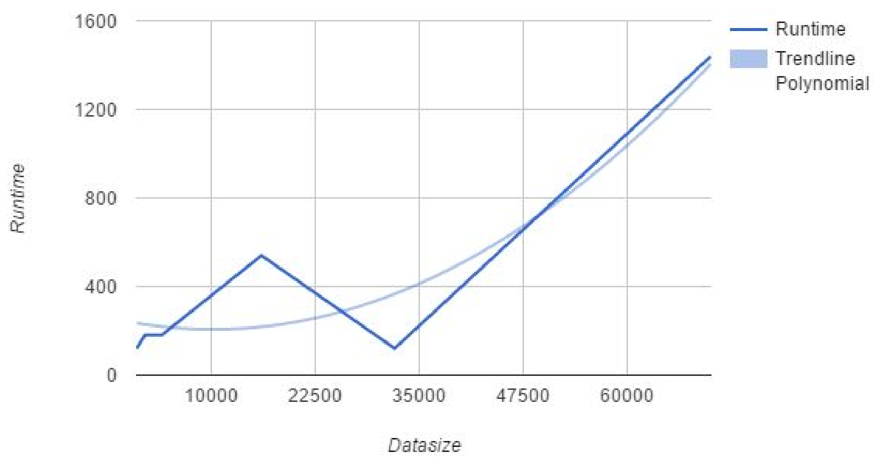


Figure 2. Amazon Runtime(s) vs Datasize(points)

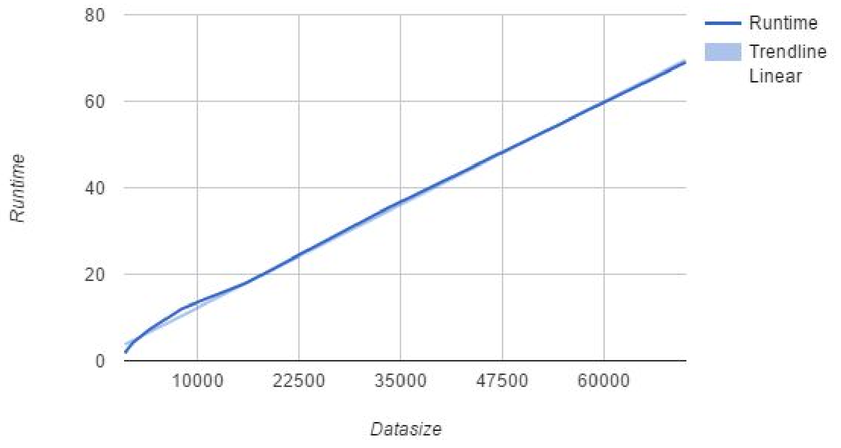


Figure 3. BigML Runtime(s) vs Datasize(points)

As we can see from the figures, all of the platforms has a linear growth in processing time vs datasize, except for a few samples in Amazon's experiment. The outlier from Amazon experiment is caused by the waiting time from the queue on the server. This is because every time a job is created on Amazon Machine Learning, it will be put on a queue and wait for it to start. We were not able to find out why Amazon has a waiting queue through their official document, so our best guess is their server is very limited comparing to the large number of requests received every moment. Note we also repeatedly run the experiments except for Amazon for the same reason.

We then plotted the processing time for each dataset versus its data-size in MB, and we spotted the plot is approximately the same. This is a useful analysis because we can use this raw data to calculate the speed of each platform. We can then estimate the processing time for any dataset in csv format. As the number of experiments grows, this estimation will get more and more accurate.

Overall, we tested the performance of the three platforms in terms of accuracy, processing time, and scalability. We have acknowledged that the number of experiments we performed in our project is extremely limited and the results are not fully convincing because of it. However, we believe this analysis is strong enough to provide an intuition on an overall performance of the three platforms, rather than an accurate quantitative measurement of the platform's performance.

3. **Learning Curve**
   a. **Tutorials**

All of these three platforms have online tutorials that are designed to help users learn how to start using them. Based on our evaluation, Amazon Machine Learning has the least detailed tutorial comparing to the other two platforms. Amazon only has one official text tutorial with images to help user navigate the platform [16]. However, this documentation only teaches user how to perform the most basic functionalities. If user want to change a small parameter or try other different functionalities, they would have to search the documentation for it. Amazon does not offer any tutorial for more advanced features. Even though they did offer detailed documentation that shows how to use these features, it contains much less images, and is much hard for user to learn. Therefore, Amazon's tutorial would only be suitable for users who just want to use the most basic functionalities, and get a trained machine learning model fast and easy.

In comparison, Microsoft offers more than 80 interactive video tutorials, which user can only access after they log into the platform. One big advantages of these tutorials are that they are interactive. By interactive, it means that while user is watching these videos, Microsoft ML would also run the same experiment showed in the video for the users. BigML also offers more than 19 video tutorials on their official website [17]. However, these tutorials are all third-party YouTube videos, and most of them are video tapes of lectures related to BigML. After watching these lectures, we feel that they are more for introducing advanced features to users. The documentation of BigML is actually more useful for helping the beginners to learn basic functionalities.

   b. **Documentations**

All of these platforms have official and detailed documentation about their platform. We tried to evaluate the freshness of these documentations by their update frequency. In order to find out their update frequency, we tried to find their last update date, and their second to late update date. However, even though we searched online, we could only find the latest update timestamp of these platforms. Based on the latest time stamp, both Microsoft and BigML are updated within three month. The last update of Microsoft's documentation is February 13th 2017, and the last update of BigML's documentation is April 3rd 2017. So, we can state that their current documentations are both up-to-date. However, Amazon's documentation is late updated in August 2nd 2016, and it described the 2015 version of their API. We found on Amazon's website that they update their API daily, so Amazon machine learning's documentation is out-of-date comparing to the other two platforms.

Update: After some much more careful search, we are able to find some histories of each platforms. First of all, Amazon lists out all document history at here[25]. Most of the updates are made once per month but there is no update since August 2016 till now. Similarly, Microsoft provides a service update list here[26]. Microsoft updates its service much less frequently (approximately 3 month per update), but it happens that it recently updates the service at the time of this project(Discussion on this during our demo does happen here!). As for BigML, it updates a lot more frequently(several times a month!)[27]. Each time, it adds new features and modify the documentation that complements the features.

Because of this new finding, we should argue that BigML is the best in terms of regularly update and maintenance for both the service as well as the documentation. Amazon updates very frequently(approximately once per month) until August of 2016 and since then there is no update with respect to the documentation. However, their services does update on a daily basis according to the API development history[25]. As for Microsoft, it only updates the document approximately once in three months, with major features pushed in their platform. We were not able to find the development history of this service itself but our best guess is it is regularly maintained and updated, similar to the other two platforms.

For comparing the contents of these three platforms, we would state that the documentations of Amazon [19]and Microsoft [20] are pretty similar. They both offered detailed description of each functionality and feature offered by the platform. However, because of these detail, they could be hard for beginner to read and understand. On the other hand, BigML adds a lot of images in their documentation [18], and they kept it well organized and easy to read. Even though BigML also offered a lot of details in their documentation, the images make their documentation much easier to read than the other two platforms' documentations.

4. User Experience

a. User Interface

For user interface, Microsoft Azure definitely offers a more unique and intuitive interface than Amazon and BigML. As shown in figure 4, Microsoft offers a drag and drop system, which allows users to easily interact with the system. Moreover, by dragging different components and following the arrows between them, it is really easy for user to visualize the whole process. The whole user interface is intuitive, and really different from a traditional or common machine learning platform because of this innovative drag and drop system.

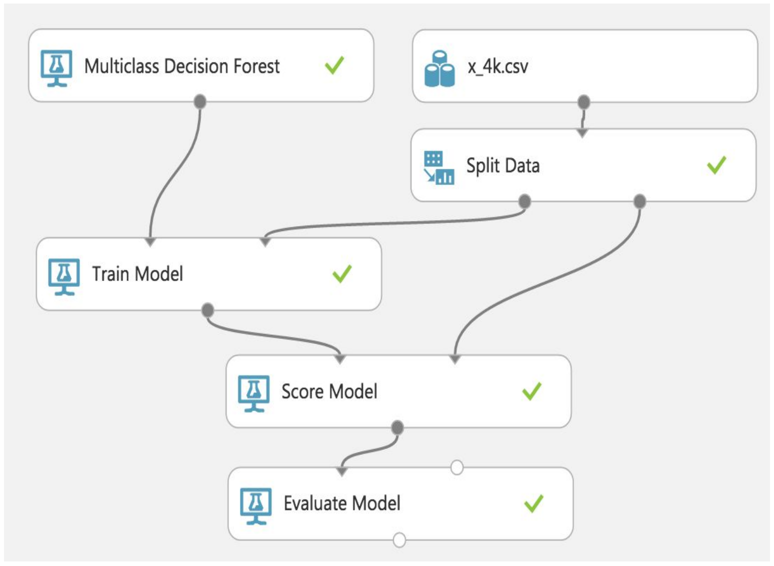
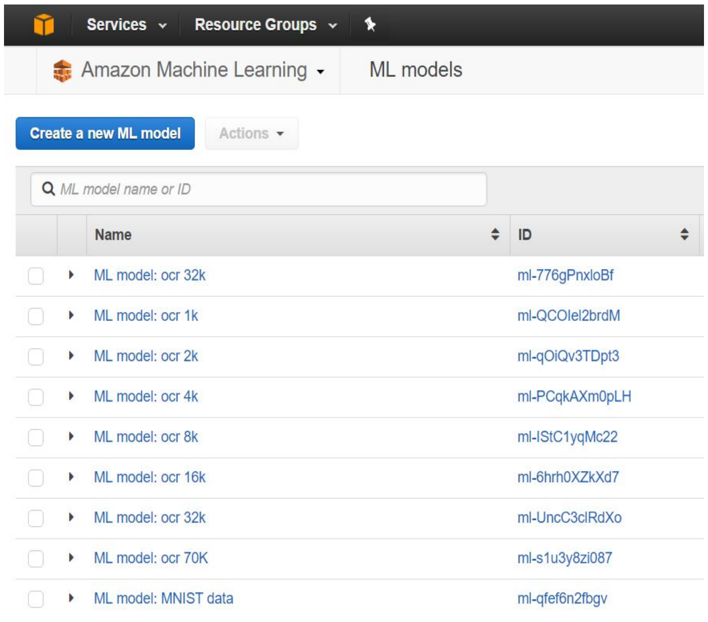
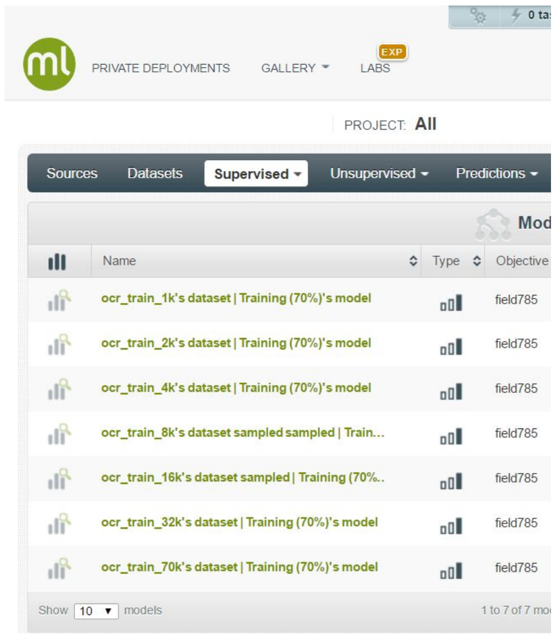


Figure 4. Microsoft Azure User Inteface



Name	ID
ML model: ocr 32k	ml-776gPnxloBf
ML model: ocr 1k	ml-QCOIel2brdM
ML model: ocr 2k	ml-qOiQv3TDpt3
ML model: ocr 4k	ml-PCqkAXm0pLH
ML model: ocr 8k	ml-lStC1yqMc22
ML model: ocr 16k	ml-6hrh0XZkXd7
ML model: ocr 32k	ml-UncC3clRdXo
ML model: ocr 70K	ml-s1u3y8zi087
ML model: MNIST data	ml-qfef6n2fbgv

Figure 5. Amazon User Inteface



Name	Type	Objective
ocr_train_1k's dataset | Training (70%)'s model		field785
ocr_train_2k's dataset | Training (70%)'s model		field785
ocr_train_4k's dataset | Training (70%)'s model		field785
ocr_train_8k's dataset sampled sampled | Train...		field785
ocr_train_16k's dataset sampled | Training (70%..		field785
ocr_train_32k's dataset | Training (70%)'s model		field785
ocr_train_70k's dataset | Training (70%)'s model		field785

Show 10 models

1 to 7 of 7 mo

Figure 6. BigML User Inteface

Comparing to Microsoft Azure, as shown in the figure 5 and 6, the user interfaces of Amazon machine learning and BigML are pretty similar. They both have a table like interface for users to pick their models. However, one thing that makes BigML's user interface much easier to use is its separation of the whole machine learning process. As shown in figure 6, BigML separates the whole machine learning process to different sections: Source, Datasets, Supervised, Unsupervised, Predictions, and Tasks. And users can navigate to each section using the top bar easily. This separation of different processes actually makes the whole user interface much easier to understand and to use. On the other hand, Amazon machine learning offers a more sequential experience, which can be confusing sometimes. For example, if the user did one step wrong, and want to go back a few steps to change an option, they would have to click on the back button multiple times, make the changes, and then click on next button multiple times to go back to their current step. It is much easier to perform this action through BigML's user interface.

b. Visualization

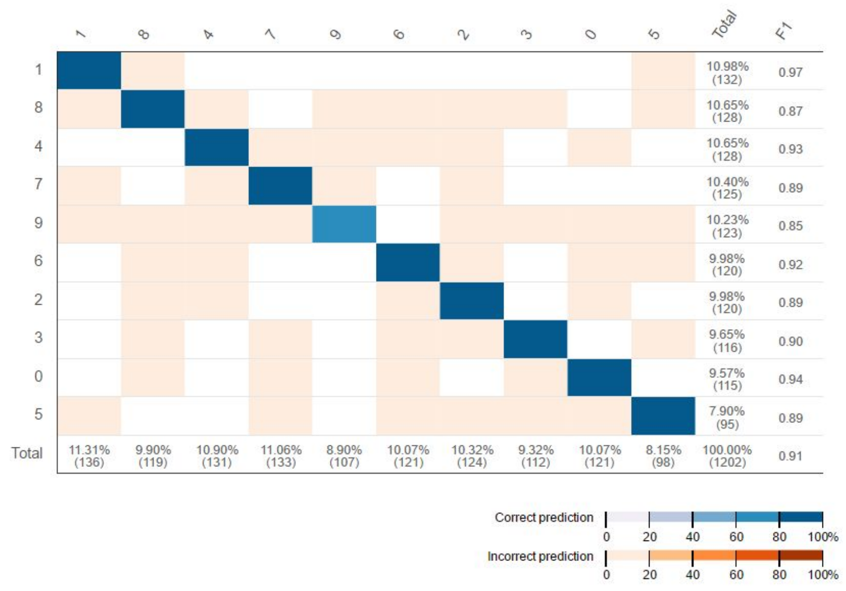
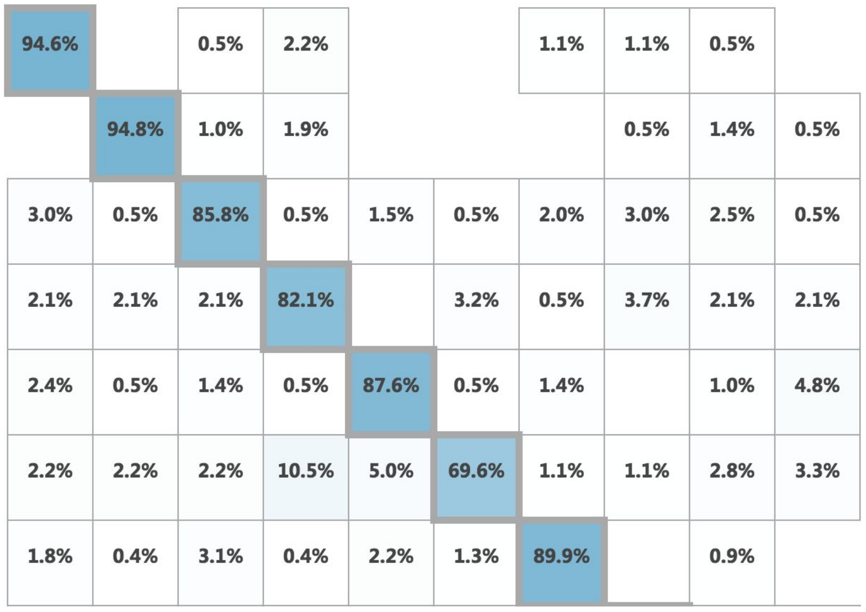


Figure 7.Amazon Visualization

For visualization, Amazon machine learning does not offer a lot of visualization functionalities. The only visualization offered by it is a graph related to the result of the trained machine learning model. As shown in the figure 7, Amazon shows the graph of the confusion matrix to the user. It's okay if the readers with no machine learning background do not understand what is a confusion matrix. This graph is just useful for users who want to know a bit more detail about the result. However, it can be confusing for beginners with on machine learning background to understand.



94.6%		0.5%	2.2%			1.1%	1.1%	0.5%	
	94.8%	1.0%	1.9%				0.5%	1.4%	0.5%
3.0%	0.5%	85.8%	0.5%	1.5%	0.5%	2.0%	3.0%	2.5%	0.5%
2.1%	2.1%	2.1%	82.1%		3.2%	0.5%	3.7%	2.1%	2.1%
2.4%	0.5%	1.4%	0.5%	87.6%	0.5%	1.4%		1.0%	4.8%
2.2%	2.2%	2.2%	10.5%	5.0%	69.6%	1.1%	1.1%	2.8%	3.3%
1.8%	0.4%	3.1%	0.4%	2.2%	1.3%	89.9%		0.9%	

Figure 8.Microsoft Visualization

As shown in the figure 8, Microsoft Azure also offers the graph for showing the confusion matrix. However, we think Microsoft Azure has a better visualization because it's user interface natural offers a great visualization of the whole machine learning training and testing process. However, Microsoft Azure does not offer any extra visualization functionalities.

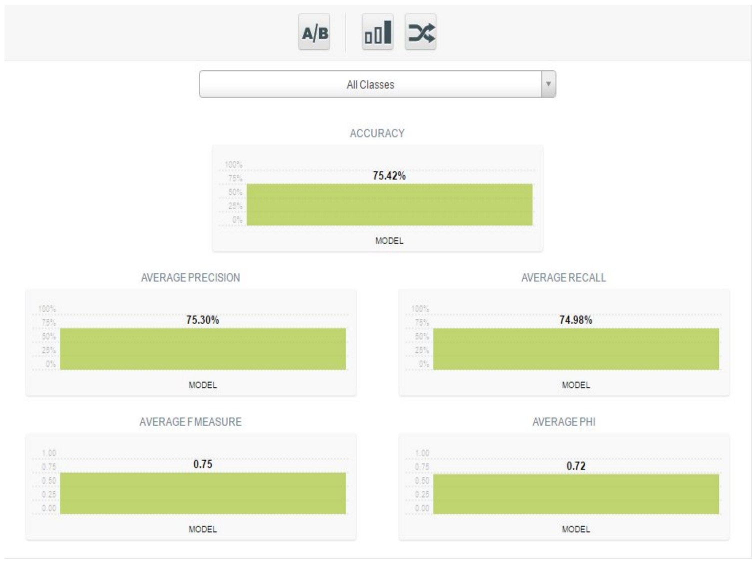


Figure 9. BigML Result Visualization

Comparing to the other two platforms, BigML offers much more visualization functionalities. Firstly, BigML also offers graphs and plots for visualizing the results as shown in the figure 9. User can see accuracy, precision, and confusion matrix of the results. Moreover, BigML offers visualization of the training model. For example, as shown in the figure 10, for decision tree mode, BigML shows an interactive graph of the decision. This graph could really help user understand what is happening during the training process.

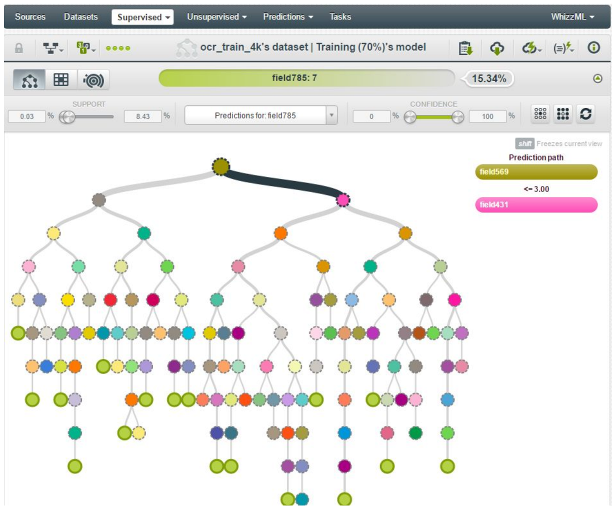


Figure 10. BigML Decision Tree Visualization

c. Pricing

For pricing, most of the machine learning cloud services offer both free tier and paid services. However, one big disadvantage of Amazon machine learning is that it does not have a free tier. User have to pay if they want to use the machine learning functionality. On the other hand, both Microsoft and BigML offers a free tier. Moreover, they both allow users to try the professional version without paying for a limited amount of time. Microsoft allows user to try all the services within 8 hours, and BigML allows for 1 day.

For the paid service, all these three platforms have their own unique ways of charging the prices. For Microsoft, it's really straight-forward. Users can just pay 10 dollars every month for unlimited services. For Amazon, there are different prices for model building and prediction. User need to pay 0.42 per hour for model building, and pay 0.1 dollar per 1000 prediction. So, if users only want to use the model a few times every money, it's probably cheaper to use Amazon. If the users want to process a large amount of data every month, the Microsoft Azure is probably cheaper in the long run.
BigML offers both subscription plans like Microsoft, and "pay as go" option as Amazon. Therefore, it gives the user a lot of freedom to pick the cheapest option based on their

situation. For the subscription plan, there are various plans, such as "standard plan", "processional plan", and "gold plan". BigML also allows users to cancel, update and downgrade their subscription plans at any time. So, it really gives user a lot of flexibility. Both individuals and companies would be able to find a suitable plan for themselves.

III. Dynamic Recommendation System

In this section, we describe our proposed system, MLPRec, in details. MLPRec is a dynamic recommendation system that can provide help users to find the most suitable machine learning platform for their experiment. User can also incorporate their own feedbacks into the system, or add new platforms, by filling out a survey. This project's code has made publically available on GitHub. More information is provided in Appendix B.

A. System architecture

MLPRec has two main components: suggestion survey and feedback survey. A system diagram is shown below:

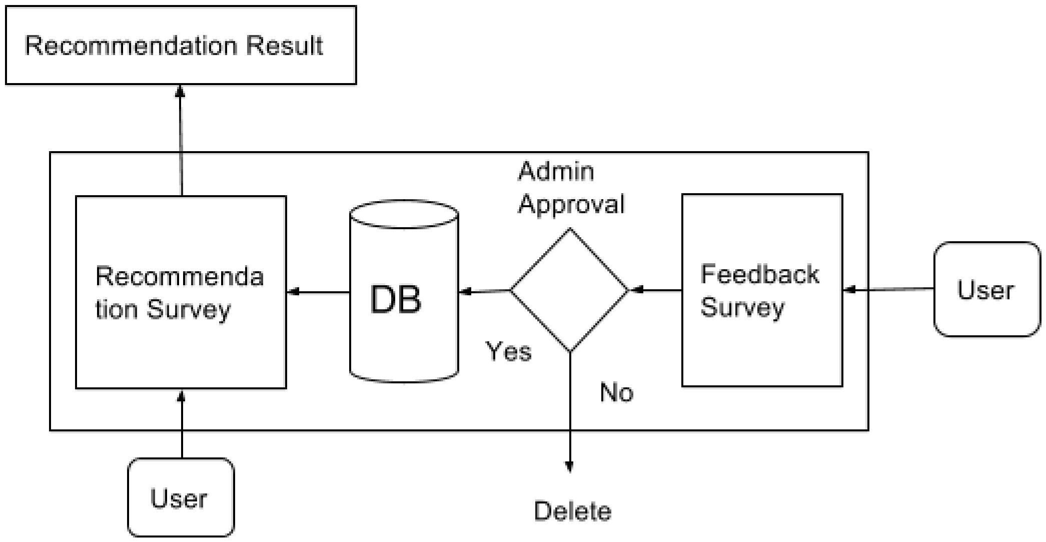


Figure 4: System diagram for MLPRec

B. Evaluation metric

After user filling out the survey, a score will be calculated for each platform and the platform with the highest score will be recommended to user. The following figure demonstrates the evaluation metric of our scoring system.

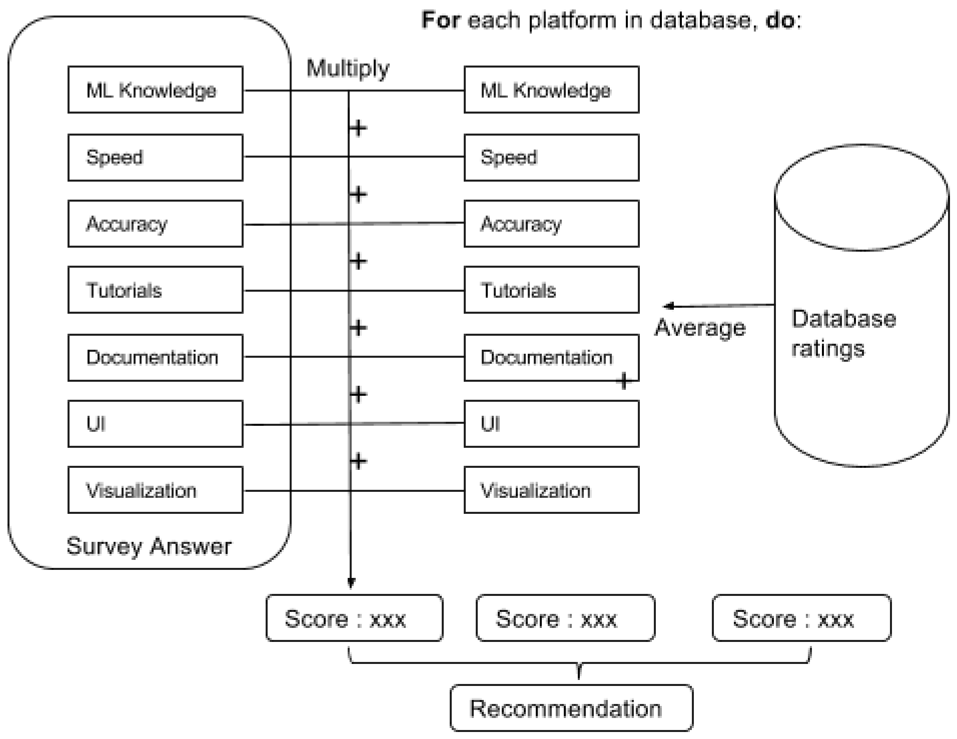


Figure 5: Evaluation Metric

For each attribute in the equation, we first take the average of this attribute in the database, then it will multiply with the user weight factor provided in the survey. The scores for all attributes will be summed up and rank in descending order.

In order to get the user weight factor for different aspect. We force a number of comparisons among them. For example, user is forced to value more on accuracy or runtime, as this is considered as a trade off. The specific list of questions are shown in the following diagram:

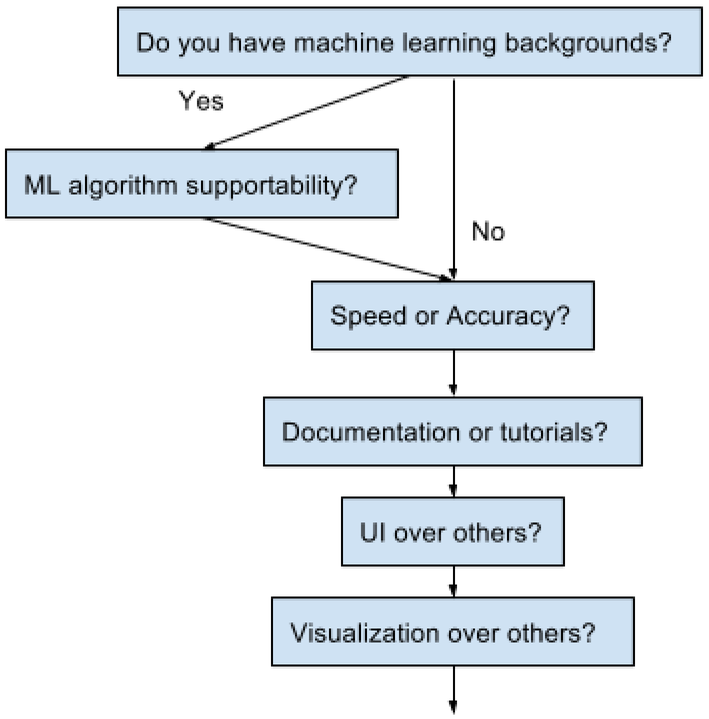

Do you have machine learning backgrounds?
Yes
ML algorithm supportability?
No
Speed or Accuracy?
Documentation or tutorials?
UI over others?
Visualization over others?

Figure 6: Survey flow chart

Each attribute in the averaged rating from database has a maximum value of 3. For speed, we normalized the parameter to the 3 scale by dividing the speed from the highest speed among all platforms and multiply by 3.

Besides the recommendation, the system also provides a bar plot of all of the aspects for each platform. A sample plot is shown in **Figure 7**. This plot shows the average rating used in evaluation, which is generated by numerous user feedbacks. From this plot, the user can have a general idea on what each platform is good at, according to other users feedback.

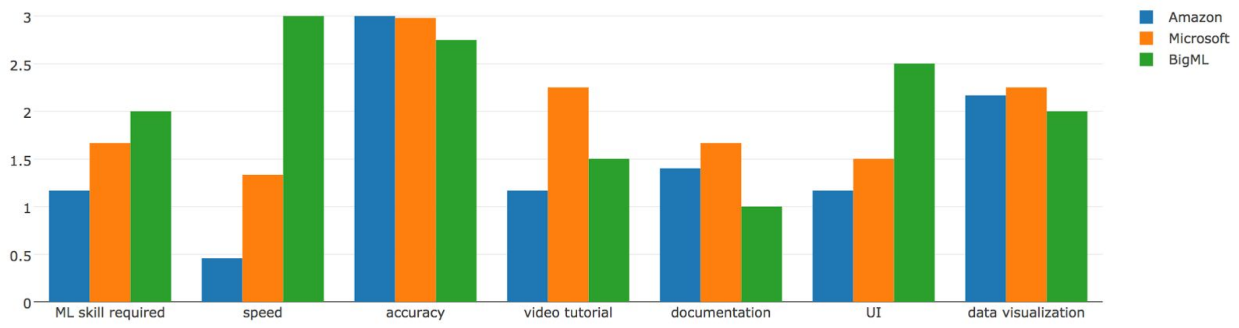


Figure 7: Bar plot for different aspect of each platform (Synthetic data[2])

[2] We used arbitrary fake data in this figure just for the illustration of this plotting feature. This does not stand for the actual rating of these three platforms.

C. Feedback

The user feedback survey is very similar to the recommendation survey. Besides adding new ratings to existing platforms, a user can also submit feedbacks on a new platform. Instead automatically inserting the feedback into database, we created a human-in-the-loop filter. After an user submits a feedback, an e-mail notification will send to admin, and it requires admin's approval in order to store in the database. This layer can effectively prevent user providing unreasonable data. However, this cannot detect fraud data within reasonable rating range. For example, an entirely random generated feedback could be passed into the database because the admin did not spot any abnormal ratings from the feedback.

D. Technology

For this web application, we used NodeJS[13] as our backend web service. In particular, we adopted ExpressJS[14] framework. We chose to use MongoDB[15] as our data storage method.

E. Assumptions, drawbacks and further discussions

There is one major assumption for our proposed system. We assumed that there are many users submitting feedbacks to the platform, thus eventually the average rating provided by the platform will be unbiased and accurate. This idea is similar to many common rating-based platforms, such as Amazon, Yelp, etc.

We have acknowledged that our finding discussed in the previous section is very subjective. The proposed system only used our findings for the demo of recommending platforms. The evaluation metric is not a static, hard-coded constant; rather, it should be a dynamic rating that depends on user feedbacks. The key idea we try to express is that we proposed a platform for users to submit their experiences with respect to a specific machine learning platform(s). We also proposed an evaluation metric to quantify both the user's need(recommendation survey), as well as averaged user feedback, in order to recommend the most suitable platform.

For now, we do not expect this system will provide very accurate recommendation due to the lack of feedbacks we have. This is the major drawback of our system. However, with the hope that many users will submit their own feedbacks, this system will eventually have unbiased and accurate ratings across multiple platforms, and produce meaningful recommendations.

We would also like to envision this project going further in the future. How will MLPRec be useful in practice to users and how will it continue to be available long after this project? In order to make MLPRec useful in practice, it requires a lot of user feedbacks and ratings. A major assumption of our prototype to work is that it has a lot of feedbacks and ratings from different users. A major native disadvantage of our feedback system is that, a user will likely only submit feedback for the one platform he/she chooses to use. This feedback then will very likely to be biased. For example, A user might tend to give out low scores for any kind of survey. Also, it will be an unfair comparison if we use the feedback from one user to compare with a feedback from another user. However, our idea is that the platform is likely going to be heuristic, not in a way that we use any machine learning algorithm, but as the number of feedbacks grows significantly, the feedbacks and ratings will be eventually unbiased. This is similar to the situation of any rating based application, such as Amazon, or Yelp. For example, if a user were to compare two restaurants, he will look for the rating on Yelp. The ratings and reviews of the two restaurants will be likely from two different set of users. This is an unfair comparison, however, still is very meaningful, as the reason being that the large number of ratings will average out the bias factors. This is essentially what we are trying to achieve here with MLPRec. Again, we believe the platform to be useful in practice as long as it receives a large number of user feedbacks for different platforms and improves based on that.

It should be predictable that certain platforms will be recommended much more often comparing to others, especially after user submitting a lot more platforms. During our experiments, we can easily “feel” or “smell” the quality of a platform. We believe this will be much more obvious when it receives a lot more ratings. There will be platforms that are most popular and recommended most frequently and the reason simply is it delivers better services, tutorials, or documentations.

Another improvement we think is to have a more carefully designed evaluation metric. Currently, it will scale all of the ratings to a 0-3 range with a user inputted weight factor. The survey only compares questions pairwise, for example, it only asks users to choose processing time vs speed. It makes sense but still not very ideal. An ideal scenario is to somehow obtain users ranking for all of the aspects. Then the weight factors in evaluation metric will become much more meaningful and provide a fair comparison.

How will it be available long after this semester? We have a couple of thoughts. First of all, it should be hosted on a public server and database so that it can be continuously accessed by anyone. Secondly, we should use a fraud detection mechanism to replace the admin manual check. Since the admin will value any feedback that is reasonably written, a computer program could handle this task even better. This will ensure the platform is still working with no human in the loop. Thirdly, instead of user filling out the “runtime” and “dataset size” in the survey, a better way would be have the website to run the uploaded dataset automatically on all of the platforms and evaluate the performances. This way, we minimize the unfair comparison caused by different factors including single platform evaluation and user's internet speed.

Another direction we can look at this problem is what if some developer takes over our project and improve on it. First of all, we have made every work in this project publically available and we are confident any person with computer science knowledge can easily read and follow up with our progress. An ideal starting direction would be adding more platforms by analyzing follow our schemas. For example, we looked at Google Cloud Platform Machine Learning[23] and IBM Machine Learning[24]. We didn't choose to do them because they tend to focus on solving large scale problems, especially for large companies like Wholesale clubs, Estate Realtors, etc. These parties are definitely great users of our MLPRec! So the hypothetical developer can start with these two platforms, and also add some analysis on how they handle large scale problems, like calculating in real-time, parallel processing, etc.

Another direction should be testing. This is a very important Software Engineering angle that we did not really touch on in this project. Even though these are platforms instead of softwares, they are still very testables. In fact, we have found several bugs, or software design defect through our experiments. For example, BigML would show error message directly to users through their notification bar, and these messages are essentially meaningless to users. Also, Microsoft Azure would get stuck on uploading files, if the user tries to upload a new dataset before the previous one finishes. There are apparently a lot more bugs than our finding within these platforms. A comprehensive analysis should involve writing a test suites for each of the platform, or carefully design a testing mechanism and report all of the findings. This analysis will be very useful and should weigh highly in the evaluation metric.

IV. Discussion

A. How did we divide the work?

The project is divided into three major works: Analyzing three platforms, implementing the recommendation system, and documentation/presentation write up. We carefully divided our labor prior to starting the project based on our specialty and interest. We both feel this is a fair share of labor and are satisfied with the outcome.

1. Jinyang Yu

In this project, I analyzed Microsoft Azure Machine Learning according to the above metric. I wrote the backend and evaluation metric parts for the recommendation system. I also wrote a major portion of the documentation and presentation.

2. Tianrui Peng

In this project, I analyzed Amazon Machine learning and BigML platforms according to the above metric. I implemented the two surveys, frontend, UI parts for the recommendation system. I also wrote the other portion of the documentation and presentation.

B. What we have learned?

1. Jinyang Yu

There are many things I learned from this project. Technical-wise, I learned the differences between the three machine learning platforms. This is something I did not know before. I also learned the popularity of MLaaS nowadays and how competitive the market is.

More importantly, I learned a lot during working on the project. We greatly expanded our goal from our project proposal. In our project proposal, we only discussed comparatively analyzing three machine learning platforms from software engineering perspectives. However, as we were working on this goal, I realized a recommendation system is as important as a technical comparison among some popular platforms. Thus, we expanded our project to building a dynamic recommendation system for machine learning platforms. This nearly doubled our scope for this project, but I find this to be well worthy. The lesson I learned here is, even though this works out in the end, I think we should identify what is truly important in the project prior to our start. That is, I should spend more time research on different aspects of ML platforms and simply define our goal as making a well designed ML platform recommendation system. Even though we have implemented the proposed system, I still feel the system is subject to many flaws and could be significantly improved after more careful designs. If we only targeted ourselves to design the system, maybe it will be much better.

2. Tianrui Peng

I learned a lot from this project. Except from learning how to use these different machine learning platforms, I learned a lot about how to perform evaluation from a software engineering perspective. Before doing this project and taking this course, I didn't think about a lot of the software engineering aspects of the programs that I'm using. Now, when I try to pick a new software to use, I learned to first check for various software engineering aspect of this tool. Moreover, when I design my own program, I started to think about these aspects, and how to improve my program according.

One other thing that I learned is the importance of making program dynamic. I found out that even for a school project, our goal and product is constantly changing as we get feedback from giving presentation and demo. I learned that it is really important to design products that are easy to adapt these kinds of changes.

C. What can be improved?

There are several things that we either planned to do but didn't do, or we spotted the flaw after implementing them. First of all, we planned to run all of the experiments many times, but we only managed to do it 5 times. We also think the recommendation survey could be more carefully designed, but we don't have enough time to do so in the end. The ratings can also be further improved by asking people to fill out the feedback survey to expand our dataset.

D. Unexpected problems

One unexpected problem that we met during the project was that we had to change our recommender system after giving the presentation and demo. We got a lot of feedback from people, and we realized that we need to make our system more dynamic. For example, we added the functionality of allowing users to input their own ratings after giving the presentation. We didn't design or plan to have this functionality at all at the beginning. But we realized that just having three platforms and our own ratings is not enough for a recommender system. Moreover, from the final demo, we realized that our questions need to be changed. We changed our questions to allow user pick one aspect to trade off the other one such as "what is more important for you, accuracy or runtime?" Even though these are problems that we didn't think about at the beginning of the project, but we learned a lot and continuously improved our system.

V. Conclusion

In this report, we discussed what we have accomplished in our final project. We first analyzed three different machine learning platforms from software engineering perspectives. We have reported all of our findings in this report. Also, we proposed a dynamic recommendation system for users to find the most suitable platform by answering a survey. This platform can also take user feedbacks and the data will be used in later evaluation for the recommendation system.

Appendix A

The following three tables are in support of the plots in Figure 1, 2, and 3. For the complete set of raw data, please click on the following url to access the google spreadsheet, or simply click here.
https://docs.google.com/a/columbia.edu/spreadsheets/d/1F2ssf1L8FA3Ap6_cEQPkqWK61PDA10iI51GDodEFFns/edit?usp=sharing

Tables for individual platform:

Amazon

Datasize (number of data points)	Runtime (Seconds)
70000	1440
32000	120
16000	540
8000	300
4000	180
2000	180
1000	120

Microsoft

Datasize (number of data points)	Runtime (Seconds)
70000	190
32000	100
16000	57.429
8000	39.347
4000	27.134
2000	16.556
1000	10.358

BigML

Datasize (number of data points)	Runtime (Seconds)
70000	69
32000	34
16000	18
8000	12
4000	7.11
2000	4.065
1000	1.776

Appendix B

Our project has posted on GitHub repository. Please view it here, or click on the following url. The GitHub readme file includes a complete guidance on how to run our experiment.

References

[1]:Machine Learning as a Service: How Data Science is Hitting the Masses. Laura Dambrosio. Mar 29 2017. *The Huffington Post*.

[2]: 20+ Machine Learning as a Service Platforms. Bulteranalytics. April 10 2017. http://www.butleranalytics.com/20-machine-learning-service-platforms Access time: April 20 2017.

[3]: Automatic Business Modeler. http://e-abm.com Access time: April 20 2017.

[4]: Civis Analytics. https://www.civisanalytics.com Access time: April 20 2017.

[5]: Cloud Machine Learning Wars: Amazon vs IBM Watson vs Microsoft Azure, Zachary Chase Lipton, UCSD, April 2015.

[6]: Machine Learning as a Service, Ines Almeida, May 12 2015. https://blog.onliquid.com/machine-learning-service-benchmark/ Acces time: April 21 2017.

[7]: Amazon Machine Learning, https://aws.amazon.com/machine-learning/

[8]: Microsoft Azure Machine Learning Studio, https://studio.azureml.net/

[9]: BigML, https://www.bigml.com

[10]: https://forums.aws.amazon.com/thread.jspa?messageID=754717&tstart=0

[11]: The MINIST database, http://yann.lecun.com/exdb/mnist/, Y. LeCun.et al.

[12]: Scikit-learn. http://scikit-learn.org/stable

[13]: NodeJS. http://nodejs.org

[14]: ExpressJS. http://expressjs.com

[15]: MongoDB. http://mongodb.com

[16]: Amazon Machine Learning Tutorial.
http://docs.aws.amazon.com/machine-learning/latest/dg/tutorial.html

[17]: BigML Tutorial. https://bigml.com/tutorials/

[18]: BigML Documentation. https://bigml.com/documentation/dashboard/

[19]: Amazon Documentation.
http://docs.aws.amazon.com/machine-learning/latest/dg/machinelearning-dg.pdf

[20]: Microsoft Documentation.
https://docs.microsoft.com/en-us/azure/machine-learning/

[21]: Google Prediction API. https://cloud.google.com/prediction/docs/

[22]: Baidu Face Recognition. https://cloud.baidu.com/product/face.html

[23]: Google Cloud Machine Learning. https://cloud.google.com/products/machine-learning/

[24]: IBM Machine Learning.
https://www.ibm.com/us-en/marketplace/machine-learning-for-zos

[25]: Amazon Document History

http://docs.aws.amazon.com/machine-learning/latest/dg/history.html
[26]: Microsoft Service Updates
https://azure.microsoft.com/en-us/updates/?product=machine-learning
[27]: BigML What’s new. https://bigml.com/whatsnew