Project Final Report

Dante Bencivenga
University of Calgary
Calgary, Canada
drbenciv@ucalgary.ca
UCID: 10096414

Jordan Kwan
University of Calgary
Calgary, Canada
jordan.kwan2@ucalgary.ca
UCID: 30063966

Naveed Aria
University of Calgary
Calgary, Canada
naveed.aria@ucalgary.ca
UCID: 30009374

Timothy Macphail
University of Calgary
Calgary, Canada
timothy.macphail@ucalgary.ca
UCID: 30115963

I. Preamble

Timothy Macphail (25% contribution) and Dante Bencivenga (25% contribution) worked on preliminary data analysis and transforming the data to extract features. Jordan Kwan (25% contribution) and Naveed Aria (25% contribution) worked on creating the models. All of the team members worked together to create the final report. A declaration signed by all members that the statement of contributions and estimate of total contribution is true.

This contribution estimate is declared to be true by:

- Timothy Macphail
- Jordan Kwan
- Naveed Aria
- Dante Bencivenga

The data and notebook for our project can be found here.

II. Abstract

The academic performance of a student as measured by their final grade can be predicted by a number of features. The value of being able to predict this is great as it will allow students to know what to expect for their final grade and for teachers to guide sessions in a way that is most beneficial to their students. In this short paper, we will discuss our process of data analysis on the **Educational Process Mining (EPM): A Learning Analytics Data Set** [1] with respect to all the facets of data science including collecting the data, inspecting the quality of the data, extracting subsets of data, transforming data, exploratory analysis, and building models from the data. From this analysis we discovered several interesting trends, including that some attendance and activity in some sessions were more important than others, to the extent that certain models even gave negative weight to their attendance. We also found that when a student performed more keystrokes, they tended to perform better. This is likely because keystrokes are in indication of how engaged the student was.

After our extensive analysis, experimentation with different feature extractions and models, we found that the time spent in sessions was strongly and positively correlated with final grade in the training and cross-validation sets. Time spent on unrelated programs had a moderate and negative affect on final grade, which was expected since time spent unfocused on the objective of the session would most likely result in a lower understanding of the session material. However, the models did not perform well on the test set, so more work would be needed to make them generalize and offer better predictive value.

III. Introduction

The problem we selected for our project is predicting the final grade of a student in a course, based on the logged activity of their computers during the lab sessions. We want to see how lab participation affects a student's final grade. The factors that we chose to represent lab participation are time spent in a session, time spent on "other" programs, and total keystrokes in session. We will go into greater detail on this in the Methodology section.

By creating a well performing model, we can gain insights by exploring its learned weights. Learning the weights of the model can help educators and students to understand the factors that contribute to their learning and academic performance. However, we can only conclude that these factors are correlated with the academic performance of the student and are not the direct cause. Hopefully with this information the quality of education and success of the students can be improved, and hindrances can be identified and eliminated.

There has been significant previous work done in this space of using data to predict the grades of students. For example we were able to find two example projects that have been conducted:

1) https://thecleverprogrammer.com/2021/04/16/student-grades-prediction-with-machine-learning/
2) https://medium.com/analytics-vidhya/data-science-techniques-to-predict-students-grade-step-by-step-using-machine-learning-algorithms-3c839c9c8ff7

The first study uses the grades from the first reporting period and the grades from the second reporting period to predict the final grade of the student. This is different from our methodology which uses the time spent in sessions, time spent not focused on the task, and keystrokes as indicating factors of the final grade. The second example uses the amount of times that a student seeks help from a professor as well as the amount of absences the student has from the class. This is an interesting model because it is testing how important a

factor the professor's office hours has on the final grade. Our project attempts to predict final grades without access to this extra data.

We're leveraging the data produced by logs on the computers of students. This could be particularly novel and useful as we move into the virtual era of education. When students are not physically in classrooms, it is hard for educators to have an idea of what's going on and why they may be succeeding or struggling. With this type of data that we're working with, it can be generated automatically and give educators insight into the happenings of a student's education. This is also a fairly non-invasive and private data collection, since the tracking software only reveals non-personal aspects of the student's activity. Any activity not in the small subset of educational programs is grouped into "Other" to preserve the privacy of the students.

The main questions that we hope to answer in this paper are:

1) Does the amount of time spent in sessions have an impact on the final grade of a student for this class? How much of an impact?
2) Does the amount of time spent outside of the main activity in a session have an impact on the final grade of a student for this class? How much of an impact?
3) Does the amount of keystrokes made in a session have an impact on the final grade of a student for this class? How much of an impact?
4) Are certain lab sessions more important than others?

For our project we are proposing that the time spent in session, time spent on other programs during a session, and amount of keystrokes have a significant impact on a student's final grade.

IV. Background and Related Work

The dataset that we chose to explore our question was the **Educational Process Mining (EPM): A Learning Analytics Data Set** [1], provided by the University of Genova. The data contains the activity of 115 first year students across 6 lab sessions in the course on digital electronics. The labs were delivered via the Digital Electronics Education and Design Suite (DEEDS), which is a platform for online learning. The platform includes specialized browsers that track the activity of the students. The data produced by this monitoring software is the source of the data in our analysis. For a more comprehensive explanation of the data refer to README.txt.

For each session at each computer, one log file was produced. Each line in a log file represents a student performing an activity, and contains information such as the name of the activity, the start and end time, the time spent idle on it, the number of keystrokes performed while in it, and more. A full list of features can be found in features.txt.

V. Methodology

A. *Inspecting Data Quality*

Upon the initial exploration and inspection of this data, we noticed some odd entries. In certain activities, the idle time spent in that activity was longer than the total time (end time minus start time) of the activity. We sought out many possible explanations for this phenomena, including mislabeled units, and out-of-order features. We were unsuccessful in reconciling the data with these theories. We wrote a Python script to identify what percentage of the activity logs contained these erroneous entries. This script can be found under the section **"Test Data Integrity"**. After running this script on many different log files, we found that approximately 25% of the idle time data was corrupt (i.e. longer than the activity time). For this reason we decided not to include the activity idle time in the features of our final observations.

B. *Extracting Features*

Given that idle time was not usable as a feature, we decided to define our feature vector as:

$$X = \begin{bmatrix} \text{total time spent in lab (session \#1)} \\ \text{time spent on "Other" or "Blank" (session \#1)} \\ \text{total session-related keystrokes (session \#1)} \\ \vdots \\ \text{total time spent in lab (session \#6)} \\ \text{time spent on "Other" or "Blank" (session \#6)} \\ \text{total session-related keystrokes (session \#6)} \end{bmatrix} \tag{1}$$

We chose to include the total time spent in each lab, and if the log file was absent for this student (the student did not attend), set this value to zero. This is because we believe that attending the sessions would probably be an indicator of the student's commitment to the class and therefore their final grade.

The activity labeled as "Other" represents any browser activity on pages not included in the expected set. According to *activities_info.txt* this is *"When the student is not viewing any pages described above, then we assign 'Other' to the activity. This includes, for the majority of the cases, the student's irrelevant activity to the course (e.g. if the student is on Facebook)."* We chose to include this because it could be an indication that a student is slacking off, which would most likely negatively affect their final grade.

We also included total session-related keystrokes as a feature. Upon inspection of the data, we found that keystrokes mostly occurred in the "hands-on" learning activities. We thought that if the use performs many keystrokes in the session, they are likely engaged in the content. It is worth noting that students don't tend to record many keystrokes while on "Other", since they are probably browsing social media which does not involve much typing.

Having each of these three features repeated once for each of the six lab sessions, rather than accumulating them into just three features allows our model to favor certain lab sessions to learn which are important.

C. *Transforming data*

We found the data transformation phase of the data analysis pipeline to be significantly challenging. In order to transform the $\sim$ 600 semi-structured activity log files into numerical feature vector observations that a linear regression model could understand, we had to perform a significant amount of pre-processing on the data. Utilizing Spark's Resilient Distributed Dataset (RDD) transformation operations, we were able to mold these text files into the appropriate format using a series of `map()`, `reduceByKey()`, `filter()`, and `mapValues()` operations, as well as converting time stamps into an integer number of seconds.

In our `get_observations()` function, we first made a list of all CSV files available for combinations of students and sessions (and we noticed that some files had a "2" appended to the end of the file name so we had to check for those too), then called `sc.textFile` on the full set of paths to make one RDD with all the raw data. We then used a `map` to remove the columns containing data we don't use such as the idle time discussed above. We then converted timestamps to integers (seconds since epoch start) using Python's `datetime` package. Next, we used a `reduceByKey()` to find the earliest lab time, latest lab time, time spent working or in "Other"/"Blank", and number of keystrokes. Finally, we changed the key using another `map` to reduce (by key) all student fields together into 18 features per student. The details of this process can be seen in cell 11 of SENG550.ipynb.

The labeled data points from the raw data were stored in two separate CSV files (final_grades_first.csv, final_grades_second.csv), and they contained the final exam scores of the students remaining the course. From analysis of the final grades we found that several students who were present in the sessions did not write the exam either time. The exam was offered to be written twice, and the questions were only altered in the numbers used, not the problem type or question asked. Many students wrote the exam the first time it was offered and wrote again the second time. Some students only opted to write the exam the second time offered. This is a major flaw of the data labels. First, students who wrote the exam twice would have an advantage not reflected in the features. Second, students who only wrote the second time could have consulted the students who wrote the first time and gained an advantage not reflected in the data.

To account for this flaw we attempted two things. First, for students who wrote the exam twice, we only took their first exam score, and for students who only wrote the exam the second time it was offered we took only those scores. For the few students who dropped the course or did not write the final exam, they were given a score of 0. This RDD contained 115 students. We ran our training and cross validations models on this dataset. Next we decided that the students who wrote the second time had too unfair of an advantage, students who dropped the course should not be included in the LabelPoint dataset, and these factors could be resulting in increased error in our models. Thus we removed the students who wrote the second time and our RDD contained 93 students compared to 115.

To develop the final RDD, gradesRDD was made which contained entries (student_id, exam_score) from the *final_grades_first.csv*, using the `textFile(path)`, and `map()` transformations with a function `getKVPairs()`. The feature observations RDD was made using the get_observations function as described above. `gradesRDD` and `observationsRDD` were then joined and turned into labeled point elements in the form `labeledpoint, features[1-18]`, where labeled point is the final grade, and stored in the `labelled_obs` RDD.

Next, the `LabeledPoint` object from `pyspark.mllin.regression` was used to transform our RDD observations to `LabeledPoint` objects. The `map()` transformation was used along with a `parseData()` function that strips the observations in our labeled and turn each entry to `LabeledPoint(label, [features])` format, and stored in a `parsedSamplePointsRDD`.

The `LabeledPoints` were then normalized using the `getNormalizedRDD()` function and stored in `normalizedRDD`. The `getNormalizedRDD()` was run on the `parsedSamplePointsRDD` which took each entry and scaled the features. The function calculates the mean and standard deviation of the `nonNormalized` features, then applies transformations using the `normalizeFeatures()` function. The result is a `normalizedRDD` containing elements of entries `LabeledPoint(Label, [normalizedFeatures])`.

D. *Normalizing data*

The `normalizedRDD` was partitioned into three RDDs, `parsedTrainData`, `parsedTestData`, and `parsedCVData` with weightings 60%, 20%, and 20% respectively. Since these RDDs would be used frequently they were cached using `cache()`. Because we later noticed that linear regression regularization decreased the intercept of the model (contrary to standard practice), we changed the normalization function to only normalize based on the training data, to avoid information leakage from the test and cross-validation sets. Thus, in our final version we normalize the latter two sets using the mean and standard deviation from the training set only.

We ran a simple model to benchmark the models' performance. The baseline model used the average grade (zero with normalization by definition) then compare it to the actual grades. The error calculation was a Root Mean Squared

Error (RMSE), using the functions `squaredError()`, and `RootMeanSquareError()`.

The first model is a Linear Regression with Stochastic Gradient Descent (SGD). The model was trained on `parsedTrainData` with the following parameters: number of iterations = 500, stepsize = 1.0, miniBatchFrac = 1.0, regularization parameter = 1.3 (manually tuned to minimize cross-validation error), regType = L2, useIntercept = False (because the mean of the normalized training data equals zero, the intercept of a least-squares fit also equals zero).

The second model is a Second Order Linear Regression with SGD. The `LabeledPoint` RDD was once again transformed to the Second Order by the `map()` transformation and function `transformOrderTwo()`, adding the squared version of each of the original features and normalized again based on the training data. The Second Order SGD was trained using the same parameters as the first model, except a large regularization parameter of 20 as that minimized the cross-validation error.

The third and final model used was the Random Forest model. The `parsedTrainData` was used to train the Random Forest model, the number of trees = 8, maxDepth = 5, maxBins = 32, and impurity=variance.

We used these models and their parameters for all of our iterations.

E. Experiment

The models were run initially using the 115 `LabeledPoints`. The best model for these observations was the Second Order SGD with a RMSE of 22.9, 14 points lower RMSE than the first model SGD. We decided to go with this model. The models were run again once we decided to remove students who wrote only the second time, and students who dropped the course in hopes of reducing the error. We also decided not to change the feature set as the other attributes in the dataset either did not pass the integrity check or had too few elements. The second iteration was run using 93 `LabelPoints`. These included only students who wrote the exam once. The error was still higher than the baseline therefore we opted to correct the overfitting by changing the regularization parameter. For the first model, the Linear Regression SGD, the regularization parameter was 1.3. The second order model regularization parameter was set to 20. From these new regularized parameters we found new RMSE values.

F. Performance Metrics

Since we are not using a classifier, we need to use root mean squared error as the metric to test the accuracy of our models.

VI. Results

We diagnosed our model using root mean square error. Before removing students who did not take the final, resulting in their final grade being a 0, we set a baseline of just using the average grade as a guess and found the error to be 38.9 which equals the standard deviation for this type of model. We found that our first order linear regression model had a slightly lower error of 36.2. We found that increasing the order of the model significantly decreased the initial error giving us a result of 22.9. We also attempted a decision tree model and that gave us an error of 32.2 which is much higher than our second order linear regression model. Even though our second order linear regression model was the lowest out of all the models we attempted, the error is still very high.

Model	Root Squared Mean Error of CV Set
Base Error	38.9
Linear Regression 1	36.2
Linear Regression 2	22.9
Random Forest	32.2

TABLE I: RMSE Error of Models using 115 Observation RDD

Model	Root Squared Mean Error of CV Set
Base Error	25.7
Linear Regression 1	30.3
Linear Regression 2	33.7
Random Forest	25.0

TABLE II: RMSE Error of Models using 93 Observation RDD

Model	Root Squared Mean Error of CV Set
Base Error	0.953
Linear Regression 1	0.897
Linear Regression 2	0.947
Random Forest	0.810

TABLE III: RMSE Error of Models using 93 Observation RDD with Regularized Parameters

Model	Root Squared Mean Error of Test Set
Base Error	1.22
Linear Regression 1	1.28
Linear Regression 2	1.22
Random Forest	1.22

TABLE IV: RMSE Error of Models on the Test Set using 93 Observation RDD with Regularized Parameters

After the data adjustment and only using 93 students, we found a new average baseline of 25.7 which is much better than the previous baseline. Surprisingly we also found that both our linear regression models of orders 1 and 2 performed worse than guessing the average grade with respective errors of 30.3 and 33.7 for linear regression orders 1 and 2. Our random forest model gave us a slightly lower error of 25.0. This error is also still very high but this model is more accurate than guessing the average grade.

Finally, we added a regularization parameter to the linear regression models normalized the labels and found that this helped the overfitting of our models. The end result was a RMSE of 0.953 for the baseline model, and 0.810 for the Random Forest, and the best model the first order at 0.897. From this we concluded the first order was the best model between the first and second order models.

From our results we can confirm that time spent in sessions is the most important factor contributing to getting a higher final grade. We can also see that the time spent on other programs and not focusing on the session activity has a strong correlation with lowering the final grade. Keystrokes also contribute to increasing the final grade but not as much as the time spent in sessions.

Additionally, we can conclude that the amount of time spent in a session, the time spent on other programs, and keystrokes may have some correlation with the final grade of a student. Unfortunately, when evaluated our models against the test set, all of them did worse than a fixed prediction of the average grade for each student. Interestingly, the random forest model had the lowest cross-validation error (but a high test set error), which suggests that the training split itself had a lot of variance, making it difficult to say how well our models would predict grades in a real setting.

For future work, we could try to decrease the error to a more acceptable standard. To do this we could attempt a number of things such as going to higher orders, adding different features, or adding initial weights to decrease variance. The integrity of the data comes into question with many of the attributes tracked having inconsistent tracking and inaccurate tracking. Adding different features, and adjusting for the weight of each session towards the final exam material should result in a better model and would be the next steps we would take.

VII. Citations

[1] M. Vahdat, L. Oneto, D. Anguita, M. Funk, M. Rauterberg.: A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator. In: G. Conole et al. (eds.): EC-TEL 2015, LNCS 9307, pp. 352-366. Springer (2015).