Just-In-Time Intervention

Manley Roberts, Matt Yang, Ania Thomas, Hosuk Choi, Anusha Prasad

Introduction

Reminder of Motivation

- We would like to use data about a student (everything we know about them up until now) in order to predict their future success in a course
    - Success: Final grade, dropout prediction
    - What we know
        - **Concrete course progress** up until now (assignment grades, assessment grades)
        - **Participation** with learning management tools (active days, links clicked, video plays)
        - **Forum Engagement** on EdX forums (comments, posts)

Reminder of Motivation

- Overarching Task is to expand upon analysis done last semester
    - Gaining more semesters of data
    - Adjusting time frame of each snippet of data to be cumulative
    - Gaining more attributes
        - Selecting the most effective attributes
    - Improving predictions
- We would like to prove that it is possible to effectively predict student success in a **current semester**, given only data from past semesters

Status Update

- Developed familiarity with tools, their manipulation, and the result of combining such tools
- Acquired data from 3 PostgreSQL queries and a MongoDB aggregation
- Joined the data from these queries into a Pandas dataframe
- Trained several estimation models on the data, and extracted a best hyperparameter configuration for each model.
- Assessed the relative feature importance of each attribute in our dataframe, and re-trained on important features

Data Collection

EdX Forum Team

Wrote query to reference MongoDB forum collection and aggregate the number of **forum posts** and **comments on forum posts**, grouped by user, by week, by course.

Wrote SQL query to extract course start timestamps to subtract from timestamps found in MongoDB results. This allows us to normalize weeks to an offset from the course start **(0 - 15)**.

Recap and Results:

Obtained dataframe with the following attributes:

- **user_id**
- **course_id**
- **week**
- **Comment (count of comments)**
- **CommentThread (count of posts)**

MongoDB Team

↳ Joint PostgreSQL Team

MongoDB unstructured data is aggregated weekly into PostgreSQL by a C21U server event

This team pivoted to writing SQL queries to access this data

Recap and Results:

Still stayed onto the data theme of clickstream and compiling a table using postgres with active days, video plays, subtitles/annotations, etc in order to combine with the grades that the other team would be compiling for the Pandas dataframe.

PostgreSQL Team

↳ Joint PostgreSQL Team

Created a new query, with the intent to merge it with one created by the Mongo DB => PostgreSQL team, and wrap it with a python wrapper but unfortunately we ran into some trouble with some mismatched ids

Edx Data Dictionary

Recap and Results:

Constructed a new query with the attributes:

- edx.courses.course_id
- student_id
- student_item_id
- submission_id
- points_earned
- points_possible
- created_at
- start_ts
- TRUNC(DATE_PART('Day', created_at::timestamp -start_ts::timestamp)/7)

Joint PostgreSQL Team

Ania Thomas, Anusha Prasad, Matthew Yang, Hosuk Choi

In summary:

- Write wrapper class that combined all the collected data/query
- Postgresql, MongoDB queries used in one class to collect and form data set
- SQLAlchemy to nest Postgres queries into the python script
- Coalesced data (edited null values to be 0)
- Pymongo to grab forum data using a certain creds file
- Got all the data into one place!

Data Cleaning & Modeling

Our Final Dataframe

- The results from 3 SQL queries and one MongoDB query were collected in Python and merged using Pandas
- Null values were substituted with zeros, timestamps were corrected, user IDs stripped, and time differentials reduced to purely day count.
- We performed an **aggregation step:** adding 18 new columns to our dataframe, each of which is found by taking the average of an existing column over every week up until this one.
    - **Example**: In week 3 of course 'C' for user '1', the **load_video_avg** column will contain the average of the **load_video** column values for user '1' in course 'C' for weeks 0-3
- Dataframe size: 12396 rows x 38 columns
- Multiple semesters of ISYE6501 data, up through Spring 2019

Our Final Dataframe

Metadata
- week

Forum Event Counts
- Comment
- CommentThread

Assignment Stats
- time_diff
- grade

Final Course Grade
- final_grade

Note: we will predict final_grade on a scale of [0-1] when modeling

Event Counts
- load_video
- play_video
- seq_next
- problem_check
- seq_prev
- seq_goto
- pause_video
- problem_save
- seek_video
- link_clicked
- closed_captions_show
- active_days
- show_transcript
- resume_course

Running Averages
- seq_goto_avg
- play_video_avg
- resume_course_avg
- pause_video_avg
- CommentThread_avg
- seq_next_avg
- seek_video_avg
- show_transcript_avg
- link_clicked_avg
- seq_prev_avg
- time_diff_avg
- active_days_avg
- problem_check_avg
- closed_captions_show_avg
- grade_avg
- Comment_avg
- load_video_avg
- problem_save_avg

The Modeling Task

- **Use Case:** all of a student's available data (including data from the week of the prediction as well as data from earlier in the semester) is used to guess their final grade
- **Data Available:** profiles of student data from all previous semesters of this class, including final grades
- **Translation to Machine Learning Task**: Train a regression model (able to handle continuous output) on data from every old semester available, and then use it to predict the grades of students in a new semester (for the purposes of today, this new semester is Spring 2019)

Training of Models

- Data Subsets: All, Average, No Average
- Hyperparameter Variations:

Estimator	Parameter Variations
Gradient Boosting Regressor	Estimators: [5, 10, 100], Learning Rate: [1e-2, 1e-1, 1, 1e1], Loss: [deviance, exponential]
Multi-level Perceptron Regressor	Hidden Layer Sizes: [(100),(150),(50,50)], Activation: [relu, logistic], Solver: [ADAM, SGD], Alpha: [0.01, 0.001, 0.0001, Max Iter: [200, 300, 500]
Linear Regression	Normalize: [True, False]
Ridge Regression	Solver: [Least Squares, SVD]

Training of Models

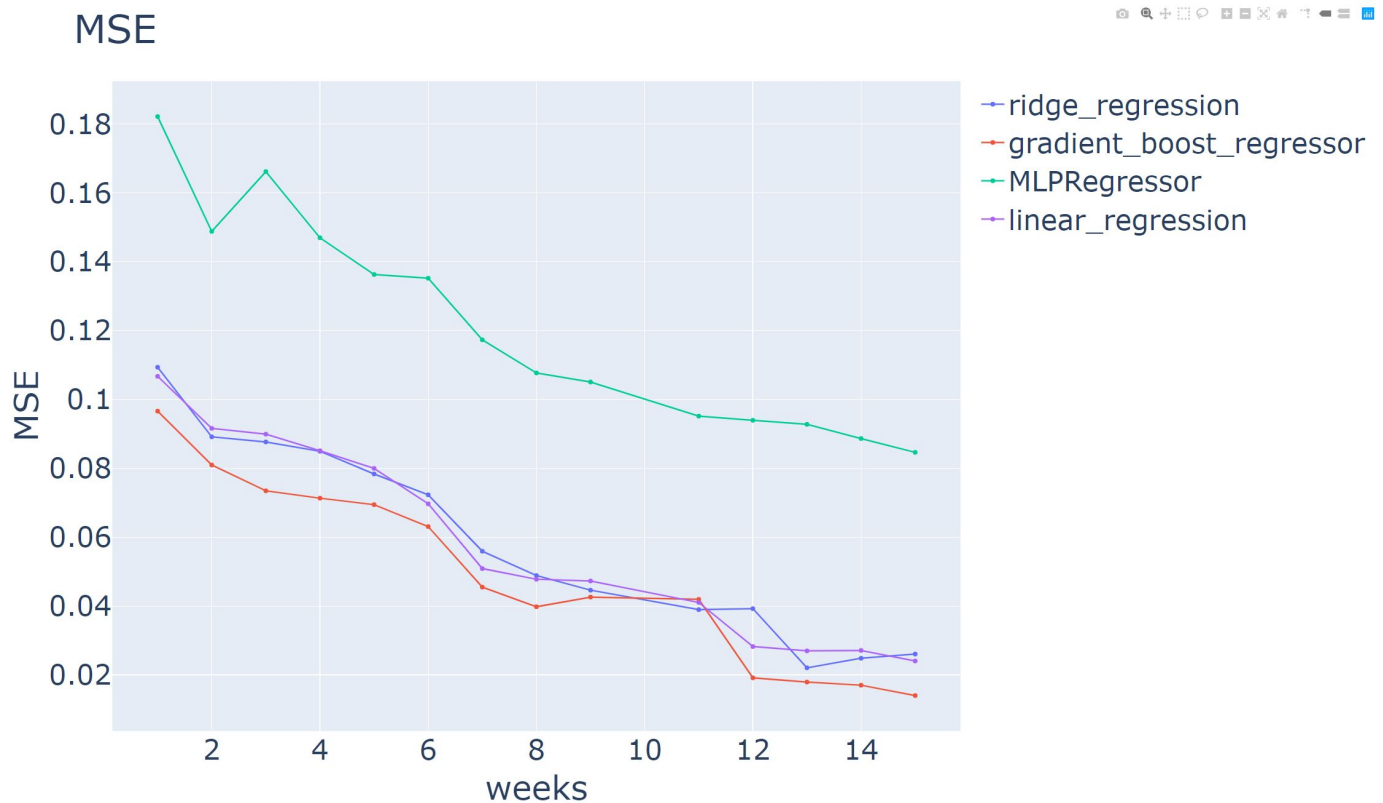
- Varying random seed, data subset, hyperparameters → 8160 models
- We found best hyperparameters for minimizing MSE on **Spring 19**.

Estimator	Best Parameters	Best Data Subset	Best Mean Squared Error (MSE)
Gradient Boosting Regressor	Estimators: 100, Loss: lad, Learning Rate: 0.01	All	0.0446
Multi-level Perceptron Regressor	Hidden Layer Sizes: (150), Activation: relu, Solver: ADAM, Alpha: 0.001, Max Iter: 200	Averages Only	0.04851
Linear Regression	Normalize: False	Averages Only	0.05828
Ridge Regression	Solver: Least Squares	All	0.05828

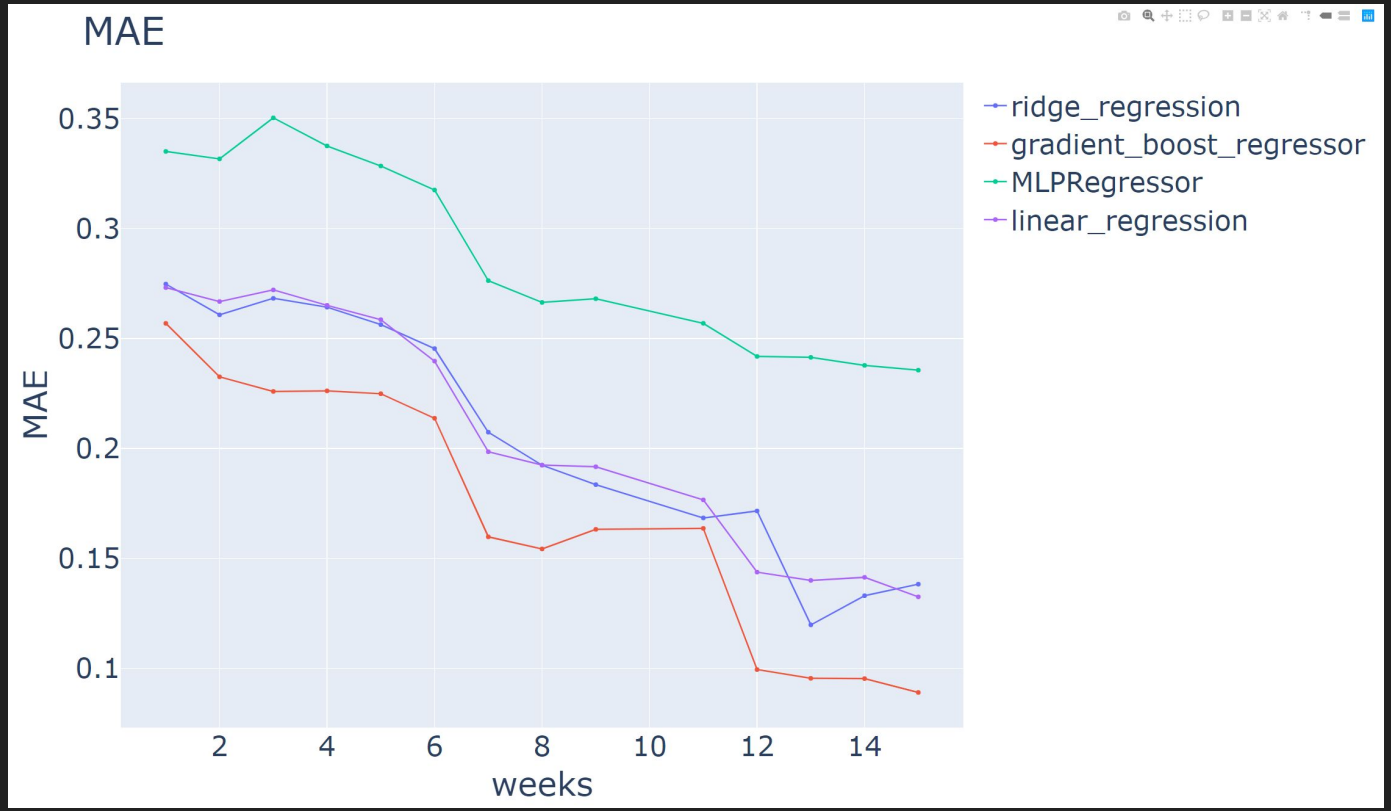
Graphing Results

- After running the models, we selected the 4 best hyperparameter configurations from the models that we ran (Gradient Boosting Regressor, MLP regressor, Linear Regressor, Ridge Regressor)
- Using python IO and plotly library in python (thank goodness for libraries that do it all!), obtained graphs of Mean Absolute Error/Mean Squared Error
- Note that no data is present for week 10 as the 10th week of the Spring 2019 semester was Spring Break.

Mean Squared Error



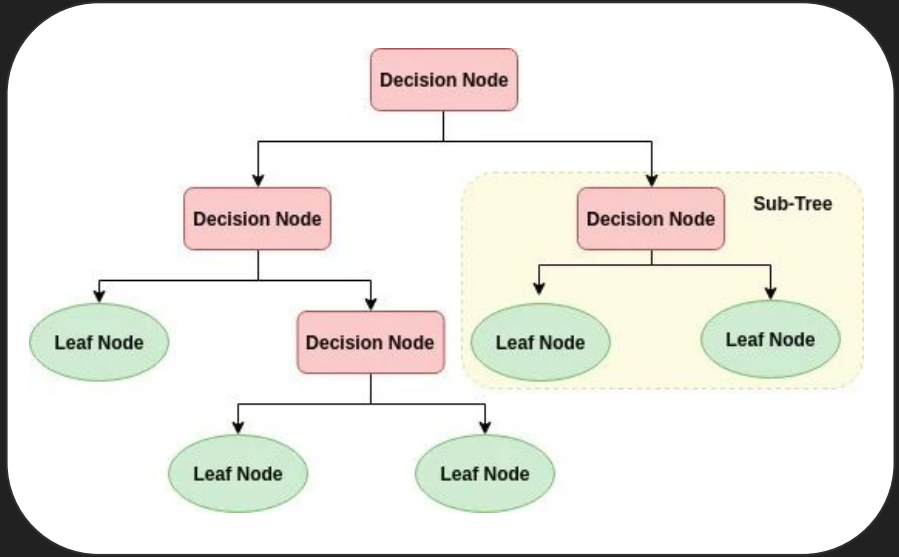
Mean Absolute Error



Building Better Models

Feature Importance

- One objective of our work is to provide insight to students into **exactly which behaviors and features** are affiliated with good class performance.
- Decision Trees are models which define a path of feature comparisons to follow. Features which are used near the top of the tree are **very important.**
- We ran a Random Forest Regressor on the data. This model constructs numerous decision trees, which we can analyze.

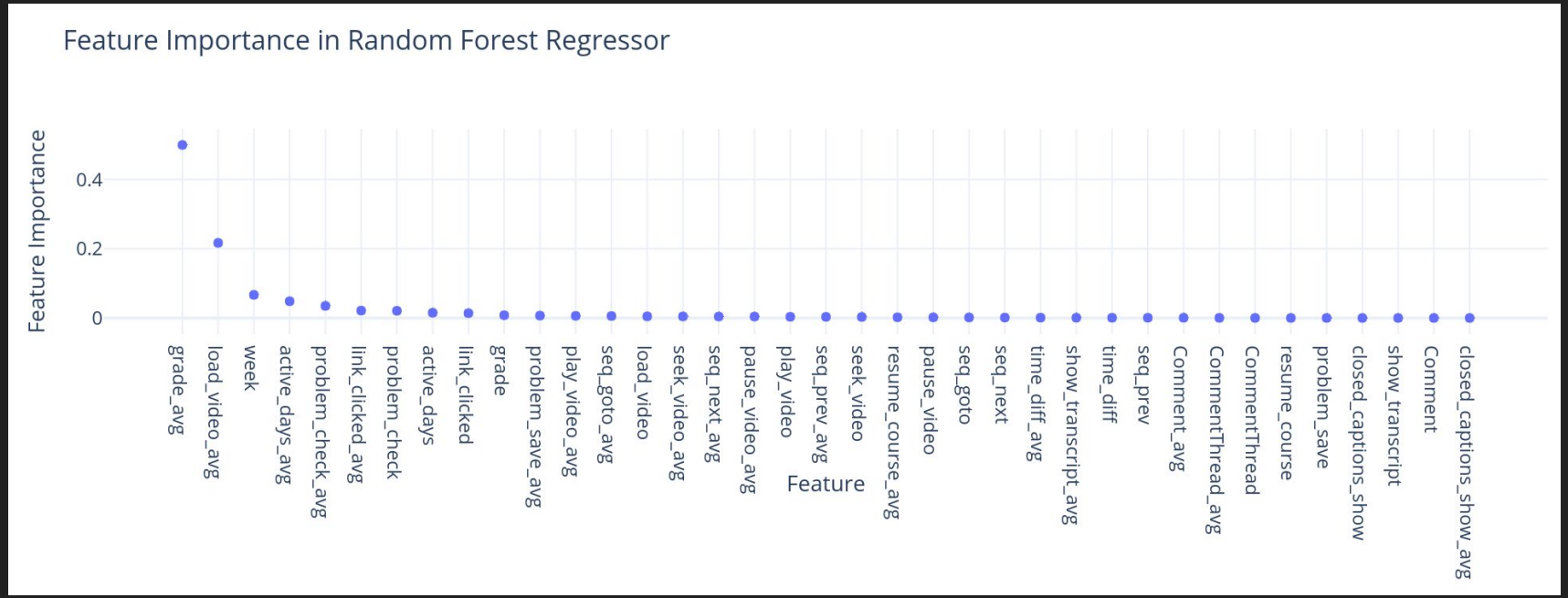


Feature Importance

- The Random Forest Regressor uses a notion of **feature importance** based on 'explained variance.'
- This notion scores each attribute at each level while building decision trees, based on how much that feature could explain variation in the data. Higher scores indicate better features.
- **The top-level scores can be exported in order to provide relative importance of various features.**
- The scores affiliated with the best-performing Random Forest Regressor are at right.

grade_avg	0.49955
load_video_avg	0.21673
week	0.06669
active_days_avg	0.04831
problem_check_avg	0.03483
link_clicked_avg	0.0213
problem_check	0.02074
active_days	0.01508
link_clicked	0.01382
grade	0.00795
problem_save_avg	0.00676
play_video_avg	0.00608
seq_goto_avg	0.00562
load_video	0.00462
seek_video_avg	0.00441
seq_next_avg	0.00411
pause_video_avg	0.00396
play_video	0.00335
seq_prev_avg	0.00317
seek_video	0.00287
resume_course_avg	0.0019
pause_video	0.00177
seq_goto	0.00147
seq_next	0.00112
time_diff_avg	0.00085
show_transcript_avg	0.0007
time_diff	0.0006
seq_prev	0.00058
Comment_avg	0.00055
CommentThread_avg	0.00031
CommentThread	0.00013
resume_course	8.59E-05
problem_save	0
closed_captions_show	0
show_transcript	0
Comment	0
closed_captions_show_avg	0

Feature Importance



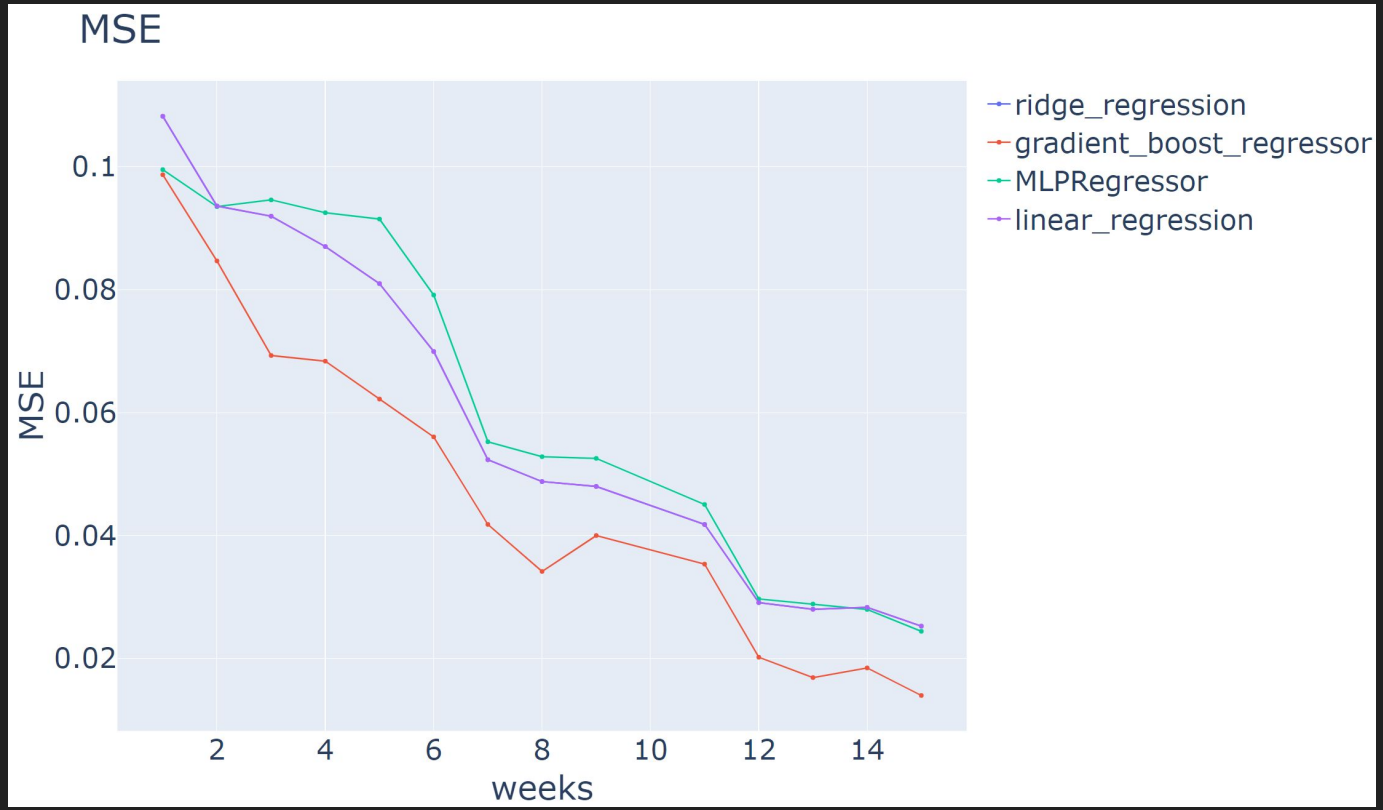
Top 6 Attributes: grade_avg, load_video_avg, week, active_days_avg, problem_check_avg, link_clicked_avg

Feature Selection

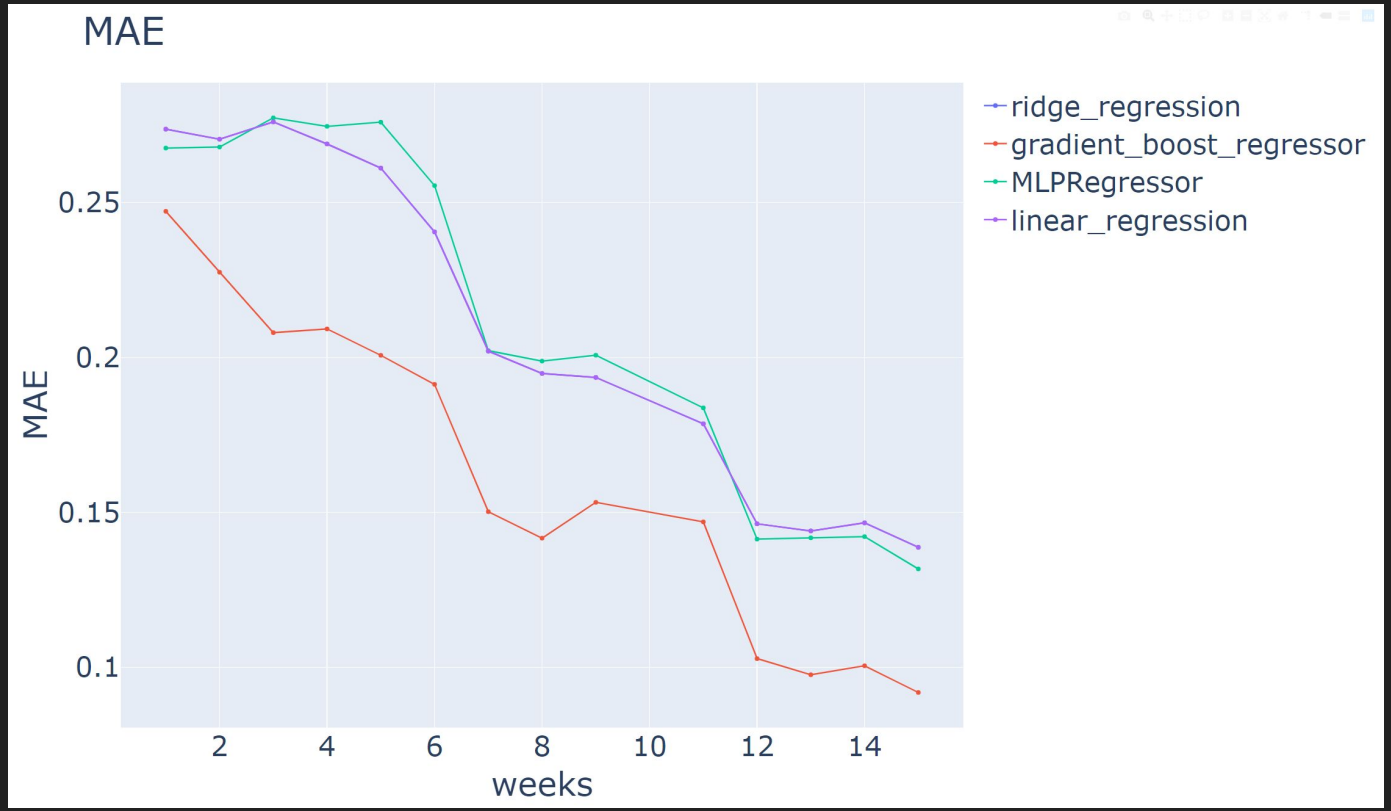
- We performed the same grid search of all hyperparameters for our four main models, only training on the 6 most important features.
- We found best hyperparameters for minimizing MSE on Spring 19.

Estimator	Best Parameters	Best Mean Squared Error (MSE)
Gradient Boosting Regressor	Estimators: 5, Loss : lad, Learning Rate: 0.1	0.0494
Multi-level Perceptron Regressor	Hidden Layer Sizes: (50, 50), Activation: Logistic, Solver: ADAM, Alpha: 0.01, Max Iterations: 200	0.05575
Linear Regression	Normalize: False	0.06151
Ridge Regression	Solver: SVD	0.06186

Mean Squared Error: Top 6 Features



Mean Absolute Error: Top 6 Features



Using Fewer Features

- We see that we achieve marginally larger error when running our models on these best 6 attributes vs. all 37 potential features.

Estimator	Best MSE (Expanded Data)	Best MSE (Top 6)
Gradient Boosting Regressor	0.0446	0.0494
Multi-level Perceptron Regressor	0.04851	0.05575
Linear Regression	0.05828	0.06151
Ridge Regression	0.05828	0.06186

Choosing Fewer Features

- Simpler models based on fewer features run notably faster and improve generalization
- Simpler models can help us explain exactly what factors go into black-box predictions of a student's final grade.
  - Average grade on homeworks (grade_avg)
  - Average times student loaded a video (load_video_avg)
  - What week it is right now (week)
  - Average days per week student active on EdX (active_days_avg)
  - Average times student submitted a problem to be checked for accuracy (problem_check_avg)
  - Average times student clicked a link (link_clicked_avg)

Conclusion

Challenges & Reflection

- Much time this semester spent developing tool and mindset familiarity
- Data format is often inconsistent with EdX documentation, requires some extra data exploration
- Data is highly distributed among numerous distinct tables and collections, challenging to discover what potential data sources might exist
- EdX forum data is largely worthless (poorly used)
- Due to time constraints, unable to properly formulate the problem for dropout prediction

Next Semester's Plan

- Formulate dropout prediction task and apply **existing dataframe** to this task
- Apply prediction to multiple GT online courses, determine differences in important feature lists between courses
- Incorporate NLP team's Piazza analytics into the set of features we can train on
- Package our script into a containerized web server which trains models periodically, stores them, and applies them on-demand to current student data
    - Build web user application to access web server

Questions?

- ...about the technologies used?
- ...about the motivation?
- ...about the next steps?