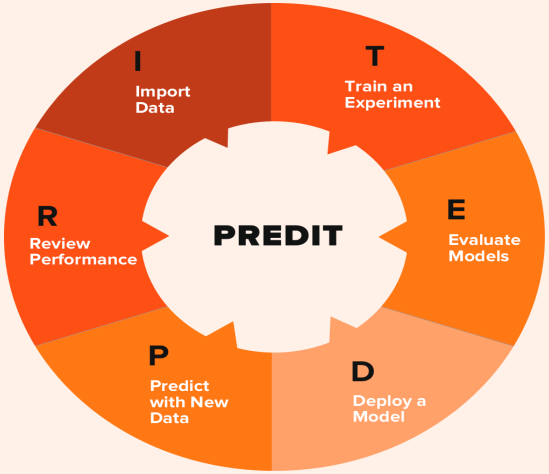
Auger.AI

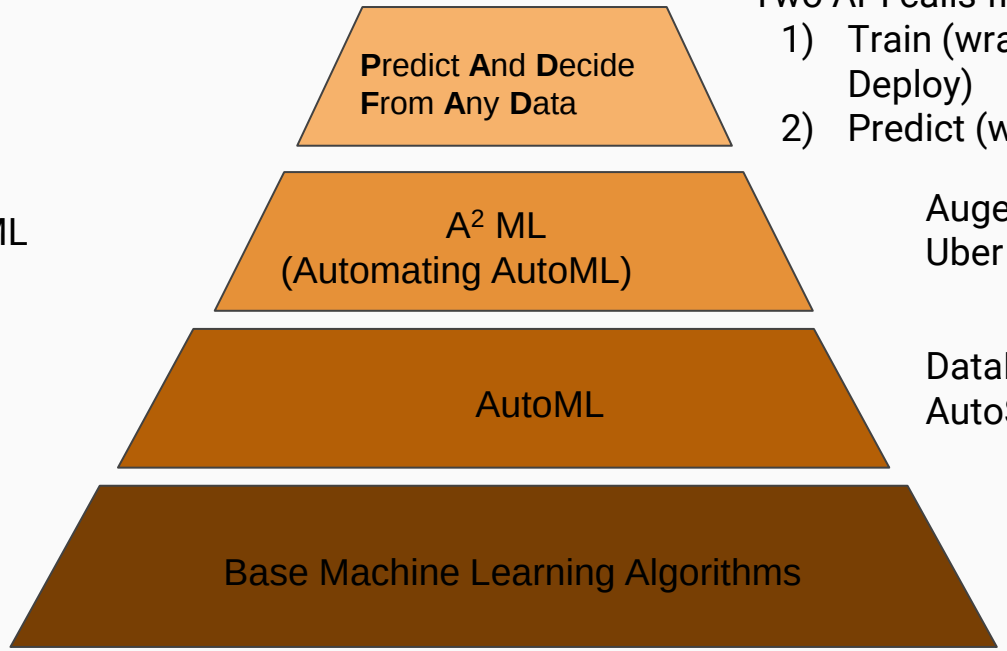
AutoAutoML

Towards a Standardized Automated Machine Learning Pipeline API

Adam Blum, CEO, Auger.AI
adam@auger.ai
@aiauger



Generations of Automating Machine Learning



Two API calls from identified datasets:

1) Train (wraps Import-Train-Evaluate-Deploy)
2) Predict (wraps Predict and Review)

Second gen AutoML

Auger.AI, Google, Azure,
Uber Michelangelo, Techton

First gen AutoML

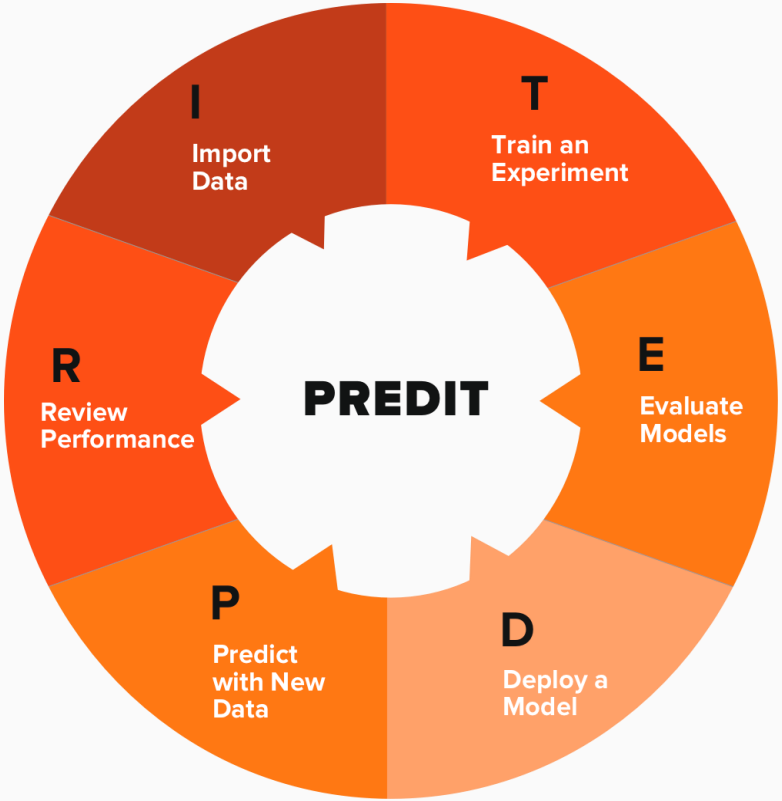
DataRobot, TPOT,
AutoSKLearn, H20 pivot, Auger.AI

Tensorflow,
Theanos,
Pytorch, H20

Auger AI

The Fundamental Phases of (Automated) Machine Learning

- Import
  - Get your data into an ML-optimized store
- Train
  - Try many algorithms and hyperparameters
- Evaluate
  - View the models and choose one
- Deploy
  - Deploy chosen model to the cloud or embedded
- Predict
  - Start inferencing with new data
- Review
  - Monitor the performance of your data



The A2ML CLI

- **Install it**

```
git clone https://github.com/augerai/a2ml.git
pip install -e ".[all]"
```

- **Use it**

```
a2ml [OPTIONS] COMMAND [ARGS]...
  new         Create new A2ML project
  import      Import data for training
  train       Train the model
  evaluate    Evaluate models after training
  deploy      Deploy trained model
  predict     Predict with deployed model
  review      Review specified model info
```

The Generic Config File: CONFIG.YAML

```yaml
name: moneyball
providers: google, azure, auger
source: ../data/baseball.csv
exclude: Team,League,Year
target: RS
model_type: regression
experiment:
 cross_validation_folds: 5
 max_total_time: 60
 max_eval_time: 1
 max_n_trials: 10
 use_ensemble: true
```

Demo A2ML with the PREDIT Pipeline

- `a2ml new`
- `a2ml import`
- `a2ml train`
- `a2ml evaluate`
- `a2ml deploy`
- `a2ml predict`

What Does Replacing Logic with Prediction Mean?

- Sort your business objects (patients, vehicles, parts, field service issues) based on predictive models versus hard-coded criteria
  - App list objects should work like any search engine
- Handle “micro-decisions” that you might otherwise have to write code to do
  - Get rid of those long switch-case and if-elif-else code blocks
- Handle models with barely enough training data
  - Retrain every night with newly supplied data: moving from mediocre to good models
- Don’t present users with hierarchical menus
  - Use AutoML to suggest to users what they do next

AutoML will eat software applications from the inside out

Case Study: Healthtree

Provided dataset contains the following information:

- **Patient treatments:** treatment method, start/end time and outcome
- **Clinical trials:** trial date and available measurements per patient
- **Genetic information:** detected mutations in genes
- **General patient info:** age, location, gender, health conditions
- **Original Target:** five class outcome (much better, slightly better, same, slightly worse, much worse)

Data is represented by SQL tables in normalized form

~3500 samples after all joins.

Most observations are missing key features

Dataset size becomes **~400** dense samples

if we insist on all features

Features

- Cleaning data based on heuristics
- Iterative imputation of missing values
- Living area population and income (extracted from zipcode and public API)
- Treatment duration and past treatment info
- Converted target to
  - binary class problem (treatment succeeds or not)
  - regression problem (time between treatments)

Results

- *Naive approach* (using raw data) gives **~40%** accuracy on multi-class classification problem and **~26%** R2 score for regression problem
- Dataset enrichment with *hand-crafted features* improves accuracy up to **~55%** for multi-class and up to **~60%** for binary class problems
- *Cleaning heuristics* and more sophisticated *imputing method* allows to get up to **68%** accuracy for binary classification
- Tuning *prediction threshold* on holdout set (20% of data) achieves at best **71%** accuracy (which is likely maximum accuracy for such dataset)
    - * prediction threshold feature not in Auger yet

Auger.AI

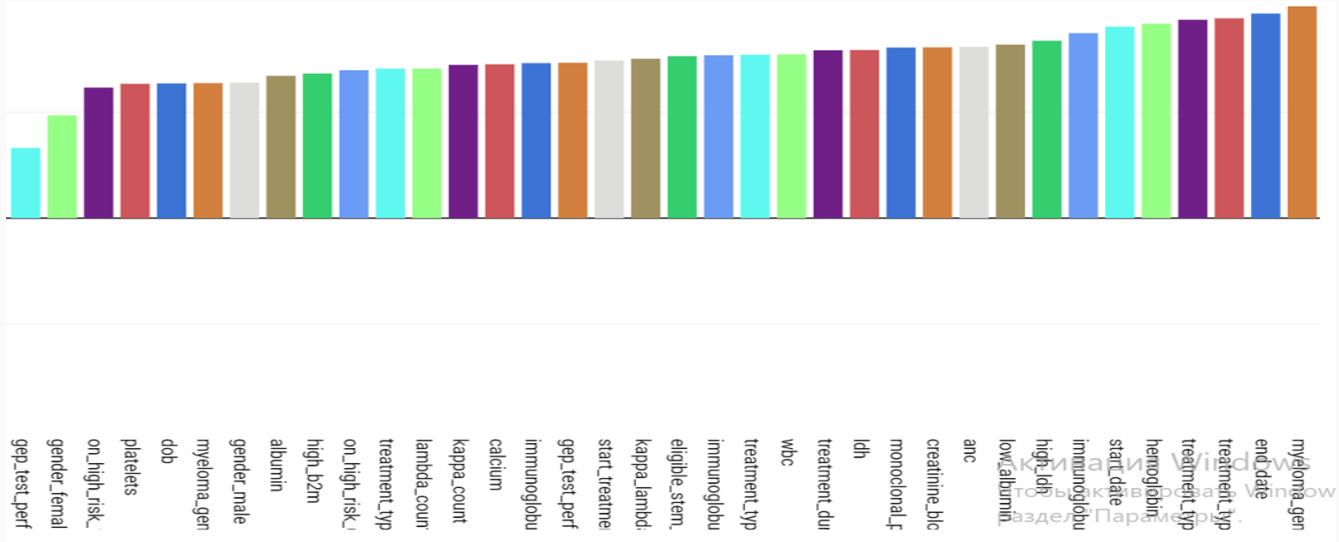
AutoML Allows Experimentation with Different Models

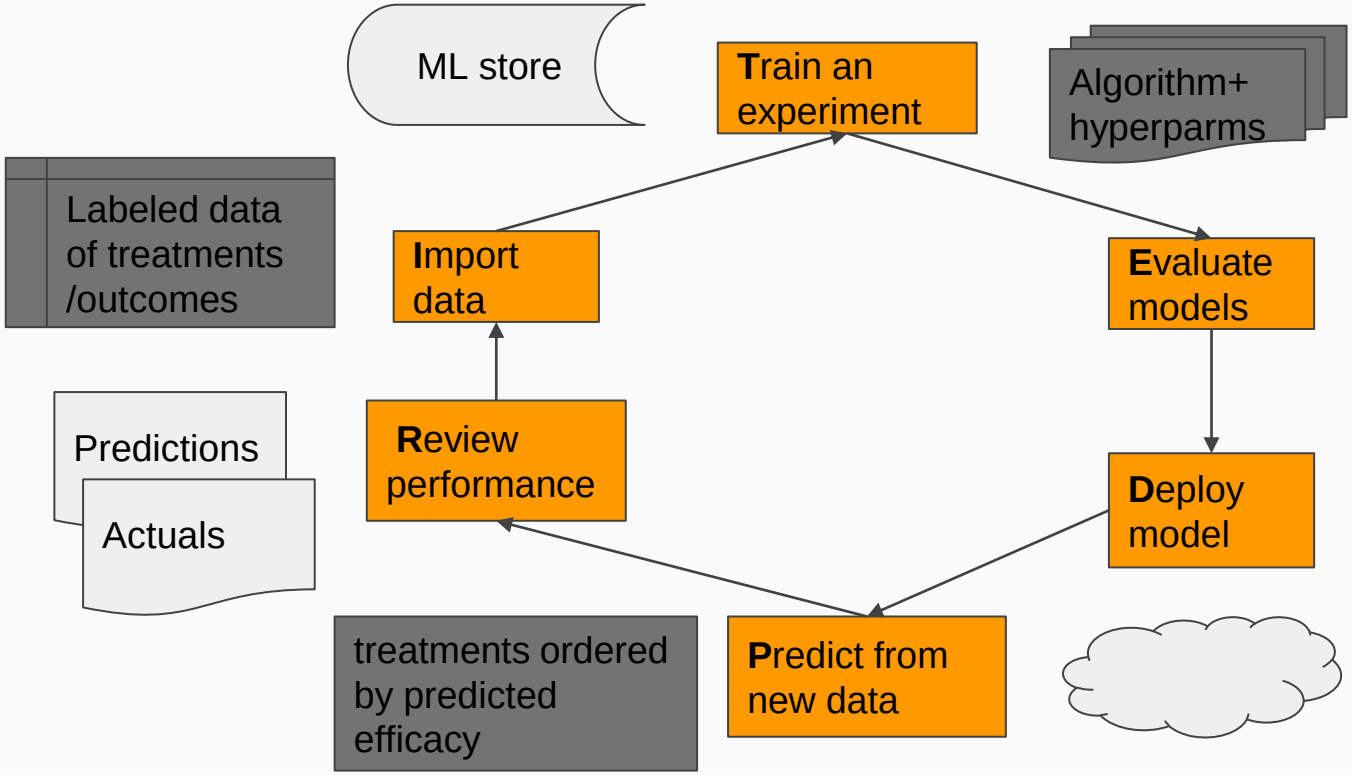
- Many features, less observations:
  - https://app.auger.ai/auger-mt/vlad_ht_new/ht-trunc/6110445eda1f10be/6191618adc21c592/leaderboard
  - Accuracy: 0.66
- Less features, more observations:
  - https://app.auger.ai/auger-mt/vlad_ht_new/ht-fix/f3c027608d81a7c8/2dc204607f1beebf
  - Accuracy 0.678

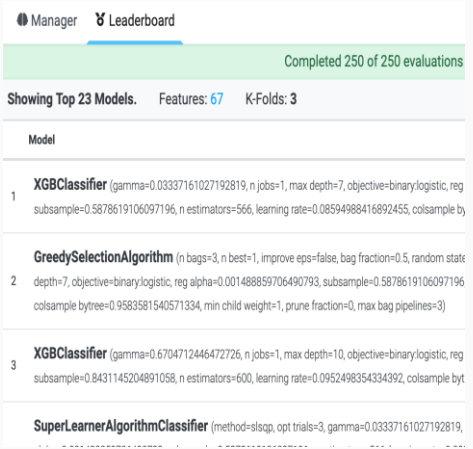
Auger.AI

Feature Importance

Feature importance shows that most of features almost equally contribute to the model. Given such low accuracy suggests target depends on some other features we do not observe.




ML store
Train an experiment
Algorithm+ hyperparms
Labeled data of treatments /outcomes
Import data
Evaluate models
Predictions
Actuals
Review performance
Deploy model
treatments ordered by predicted efficacy
Predict from new data


Manager
Leaderboard
Completed 250 of 250 evaluations
Showing Top 23 Models.
Features: 67
K-Folds: 3
Model
1 XGBClassifier (gamma=0.03337161027192819, n jobs=1, max depth=7, objective=binary:logistic, reg subsample=0.5878619106097196, n estimators=566, learning rate=0.08594988416892455, colsample by
2 GreedySelectionAlgorithm (n bags=3, n best=1, improve eps=false, bag fraction=0.5, random state depth=7, objective=binary:logistic, reg alpha=0.001488859706490793, subsample=0.5878619106097196 colsample bytree=0.9583581540571334, min child weight=1, prune fraction=0, max bag pipelines=3)
3 XGBClassifier (gamma=0.6704712446472726, n jobs=1, max depth=10, objective=binary:logistic, reg subsample=0.8431145204891058, n estimators=600, learning rate=0.09524983543343 92, colsample byt
SuperLearnerAlgorithmClassifier (method=slsqp, opt trials=3, gamma=0.03337161027192819,

Tips on Applying AutoML to Your Business

- Always use AutoML for predictive models
  - Not just when “data scientists” are busy
  - It will usually outperform data scientists
  - And it’s always useful for baselining, even with “grid search AutoML”
- Even if you don’t automate the phases (the PREDIT pipeline):
  - Think about all of the phases
  - Especially the Review phase
- Insure that you plan how all phases are embedded and controlled by the app
  - How do you get data, train optimally, deploy predictive models, start predicting and then have a plan for what the app does when model degrades
- Use “Automating AutoML” paradigm to pick an AutoML tool
  - Train your dataset with multiple AutoML tools for your early efforts
  - Until you find AutoML that reliably outperforms for you and your problems

Automating AutoML (A2ML) Advisory Board

Requirements

- longtime developer experience, preferably in Python
- experience embedding machine learning or other predictive models into applications
- currently working in an enterprise (not a software company)

Time commitment

- two hours per month (one hour phone call per month, one hour of reading docs and using the framework)

Why they would do it?

- broad applicability to anyone building predictive models into applications
- removes lockin to any AutoML
- stock options (negotiable)
- free usage of Auger (up to 100 hours per month)

Auger.AI

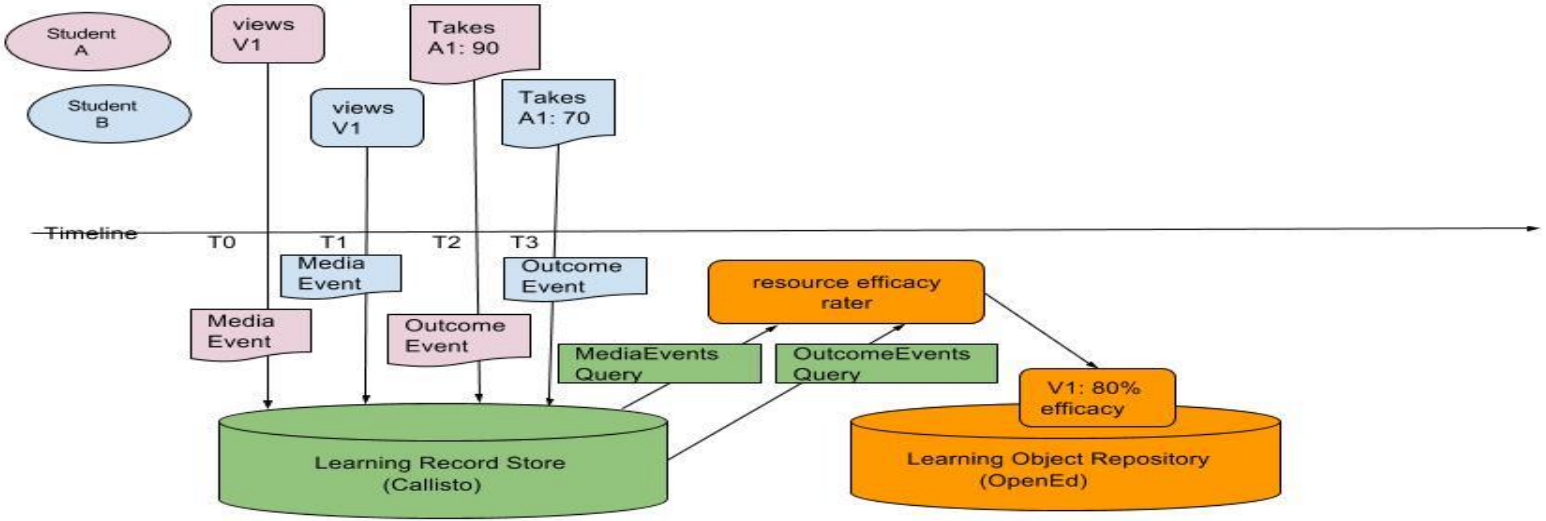
Backup

Auger.AI

Predicting Outcomes with Largest Educational Resource Library

- OpenEd.com - largest educational resource library
  - 1M videos
  - 12% of US classrooms
  - 4M video consumption events
- Efficacy is long accepted method of determining what resources work
- By automating AutoML we turned efficacy into a prediction problem

Auger.AI



Resource Efficacy Computation

From Basic to Contextual Efficacy

Let's look at this as a classic machine learning prediction problem...

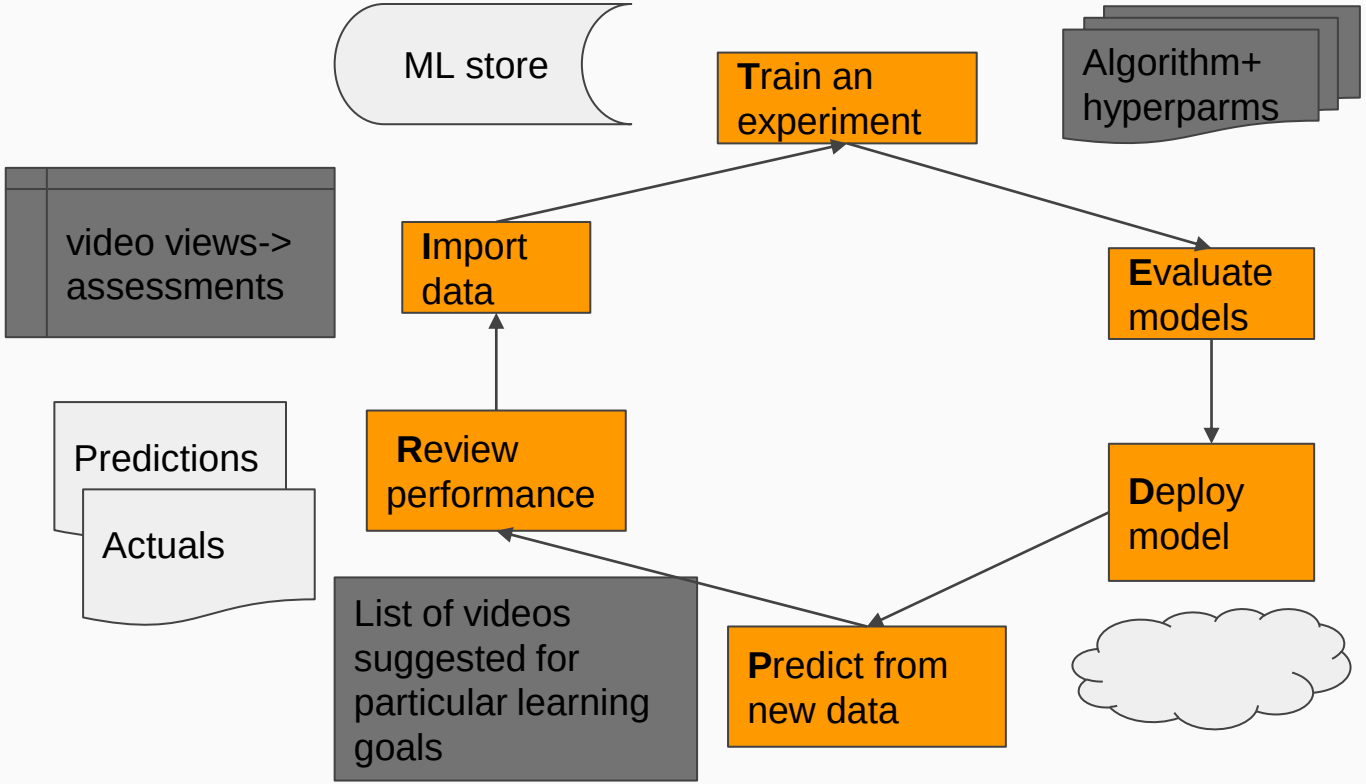
- Features are attributes of the resource, **attributes of the learner, attributes of the learning event**
- Target is the result on a subsequent related assessment
- Each "resource consumption" event or fact is represented with is uniquely represented with:
  - Composite key of "resource ID"-"learner ID"-"skill ID"-"date/time of consumption"
  - Features derived from attributes of the resource and student
  - Target is score on next assessment
- This lets us include *context* about the student in a "predicted efficacy" of the resource

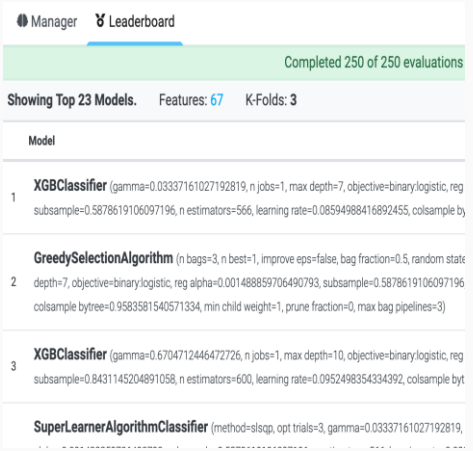
Attribute	ML Type	Data Type	Description
resourceID	Informational	ID	The unique ID of the resource
skill_or_standard	Informational	ID	A skill associated with the resource
duration	Feature	Time Duration	Length of video
subject	Feature	Categorical	Subject within the OpenEd two level subject taxonomy, e.g. "Math/Geometry"
textComplexity	Feature	Floating Point (0-1)	As measured with Flesch-Kincaid algorithm. 90 if not supplied
gradeResource	Feature	Integer (0-12)	Minimum grade of resource
basicEfficacy	Feature	Floating Point (0-1)	The current measured efficacy of the resource
localHour	Feature	Category (0-23)	Based on time of consumption of the resource, we can impute category given that we know the student zip code (or state)
numberOfStudentViews	Feature	Integer	Number of views of video by this student
percentComplete	Feature	Floating Point (0-1)	Percentage completed watching video
studentId	Informational	ID	The unique ID of the learner
gradeStudent	Feature	Integer (0-12)	Grade the student is in
historicalQuizScoreArea	Feature	Floating Point (0-1)	Average score for this student on previous assessments in the past within the same area (top level subject such as Math or ELA)
zipCode	Informational	String	
medianIncome	Feature	Floating Point	Average income in the area that the student's school is in. If we can't get zip code of school or district use median income of state (if known). Otherwise use overall median income ($60K if not otherwise present)
assessmentPerformance	**TARGET**	**Floating Point (0-1)**	**Score on next assessment taken on same skill/standard, within one month**

Model Summary

- # of events: 140K
- # of users: 20K
- # of features: 12
- Best Model (via AutoSKLearn)
- SuperLearnerAlgorithmRegressor (method=nnls, opt trials=3, n jobs=1, bootstrap=false, criterion=mse, max features=0.723092347969325, n estimators=100, min samples leaf=2, min samples split=8, n jobs(LGBMRegressor)=1, silent=-1, verbose=-1, max depth=7, reg alpha=0.01203740481919808, subsample=0.9535557593241302, num leaves=54, reg lambda=0.009631522259307235, n estimators(LGBMRegressor)=1484, learning rate=0.2487382194790653, colsample bytree=0.8562743228997485, min child samples=1)

- R2: 0.8856


ML store
Train an experiment
Algorithm+ hyperparms
video views-> assessments
Import data
Evaluate models
Predictions
Actuals
Review performance
Deploy model
List of videos suggested for particular learning goals
Predict from new data


Manager
Leaderboard
Completed 250 of 250 evaluations
Showing Top 23 Models.
Features: 67
K-Folds: 3
Model
1
XGBClassifier (gamma=0.03337161027192819, n jobs=1, max depth=7, objective=binary:logistic, reg
subsample=0.5878619106097196, n estimators=566, learning rate=0.08594988416892455, colsample by
2
GreedySelectionAlgorithm (n bags=3, n best=1, improve eps=false, bag fraction=0.5, random state
depth=7, objective=binary:logistic, reg alpha=0.001488859706490793, subsample=0.5878619106097196
colsample bytree=0.9583581540571334, min child weight=1, prune fraction=0, max bag pipelines=3)
3
XGBClassifier (gamma=0.6704712446472726, n jobs=1, max depth=10, objective=binary:logistic, reg
subsample=0.8431145204891058, n estimators=600, learning rate=0.0952498354334392, colsample byt
SuperLearnerAlgorithmClassifier (method=slsqp, opt trials=3, gamma=0.03337161027192819,