Video-watching behavior vs student performance

Aakash Saravanan

Objectives

- Understand student’s performance on in-video quizzes from student video-watching behavior on online course

- Predict whether user will be CFA (correct on first attempt) in answering question from student performance in MOOC (massive open online courses)

Overview

1. Dataset
2. Problem 1: How well can students be naturally grouped or clustered by their video-watching behavior?
    a. Models discussed: t-SNE, K-means
3. Problem 2: Can student's video-watching behavior be used to predict student's performance?
    a. Models discussed: Polynomial ridge regression, simple neural network
4. Problem 3: How well can you predict a student's performance on a particular in-video quiz question?
    a. Models discussed: Random Forest

Dataset

Features examined for a student – video pair: userID, videoID, fracSpent, fracComp, fracPaused, numPauses, avgPBR, numRWs, numFFs, s

- fracSpent: total time spent watching video divided by length of video
- fracComp: fraction of video watched (0 to 1)
- fracPaused: time student paused on video divided by length of video
- numPauses: number of times student paused video
- avgPBR: average playback rate (0.5x to 2.0x)
- numRWs: number of times student rewinds
- numFFs: number of times student fast forwards
- s: student answered question correct or incorrect (1 or 0) immediately after video

3976 students, 0-92 videos, 29,304 student-video pairs in dataset

Problem 1: How well can students be naturally grouped or clustered by video-watching behavior?

Only students that completed at least 5 videos were used.
t-SNE model used: t-distributed stochastic neighbor embedding (sklearn Python package)

- t-SNE is technique that helps in visualizing complex, high-dimensional data by reducing it to two or three dimensions while preserving the structure and relationships in the data.
- Step 1) Pairwise-similarities
- Step 2) Lower-dimensional representation
- Step 3) Optimization Algorithm
- Benefits of model:
  - Preserves local structures: similar points in high-dimensional space remain close together in lower-dimensional space
  - Helps uncover patterns / clusters / outliers in high-dimensional space
- Limitations:
  - Can be slow, especially for larger datasets
  - Not deterministic: stochastic nature so different results on different runs

Analogy: It is like flattening a crumpled piece of paper

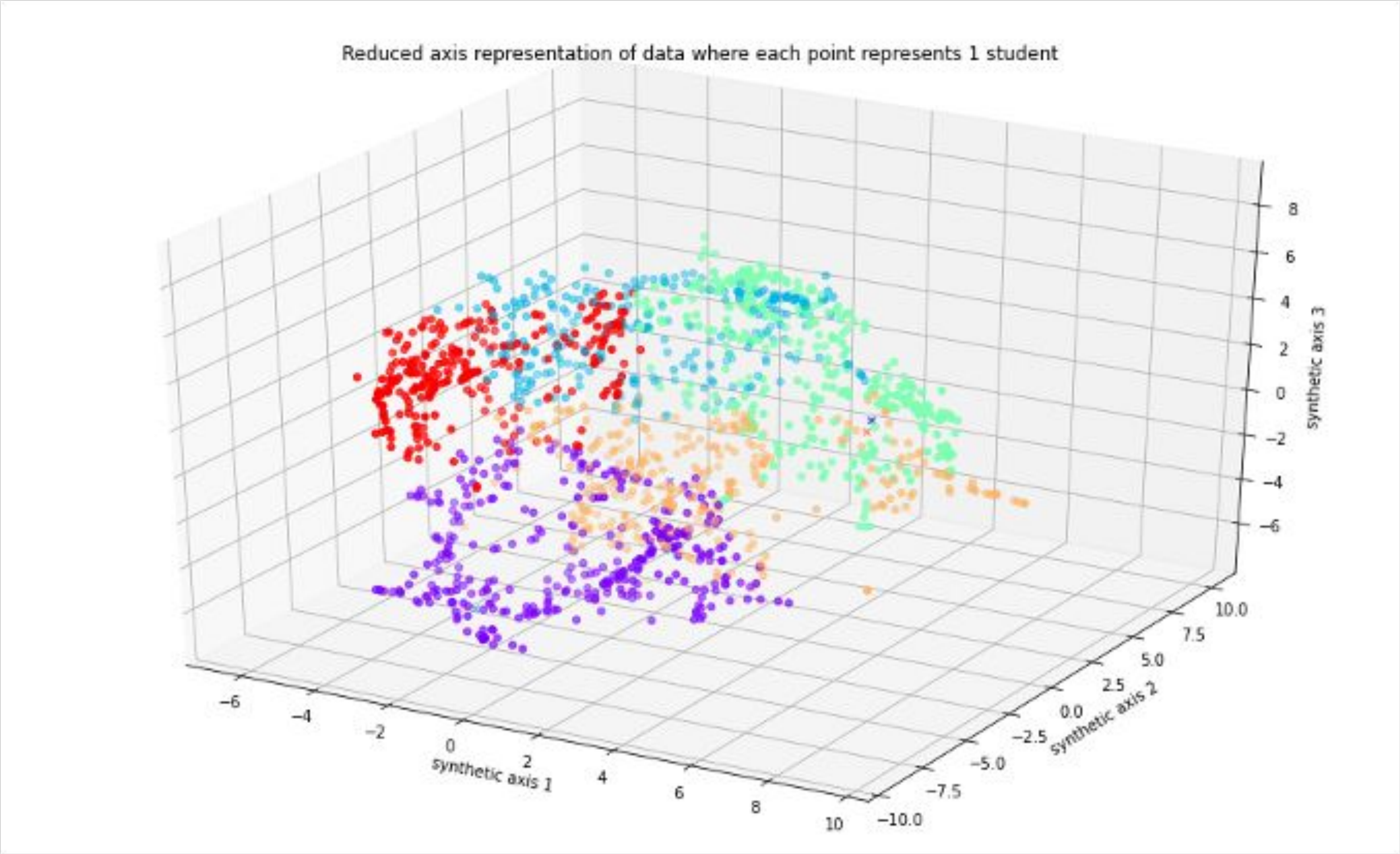


Figure 1: TSNE representing video-watching behavior features reduced to 3 dimensions

K-means

Clustering algorithm used to partition dataset into k distinct clusters based on feature similarity

Goal: group similar data points together

How it works:

- Randomly select k initial centroids
- Assign each data point to nearest centroid based on distance
- Calculate new centroids by taking mean of data points assigned to each cluster
- Repeat the assignment and update steps until convergence

Output: k clusters and each data point assigned to a cluster

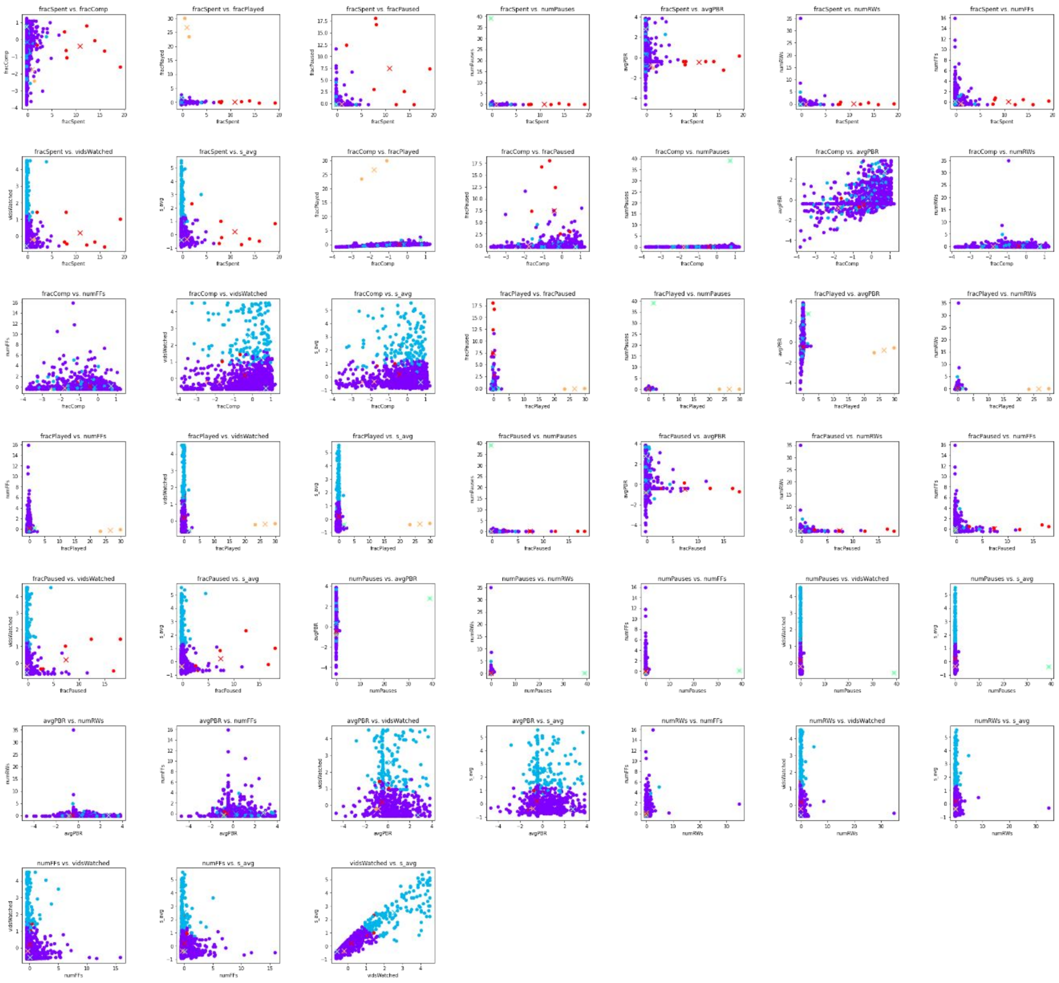


Figure 2: K means output plotting video-watching behavior features against each other

Problem 2: Can student's video-watching behavior be used to predict student's performance?

- All students that complete at least half of quizzes were used (students who watch at least half of total videos)
- Polynomial ridge regression model to predict future student's performance given video-watching behavior parameters
  - Purpose: Polynomial model while applying ridge regularization to control for overfitting
    - Polynomial regression: model relationship between input features and output as an nth degree polynomial
    - Ridge regression: linear regression that includes regularization term (lambda) to prevent overfitting
    - Cost function has an added penalty term
  - Benefits of model:
    - Preventing overfitting
    - Generalizes better to unseen data
    - Fit non-linear relationships
  - Limitations:
    - complex with higher degree polynomials

$$\text{Cost} = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{n}\beta_j^2$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_n x^n + \epsilon$$

Develop / Train model

Step 1) Import libraries

- import matplotlib.pyplot as plt
- import numpy as np
- import seaborn as sns
- from sklearn.pipeline import make_pipeline
- from sklearn.linear_model import Ridge
- from sklearn.preprocessing import PolynomialFeatures

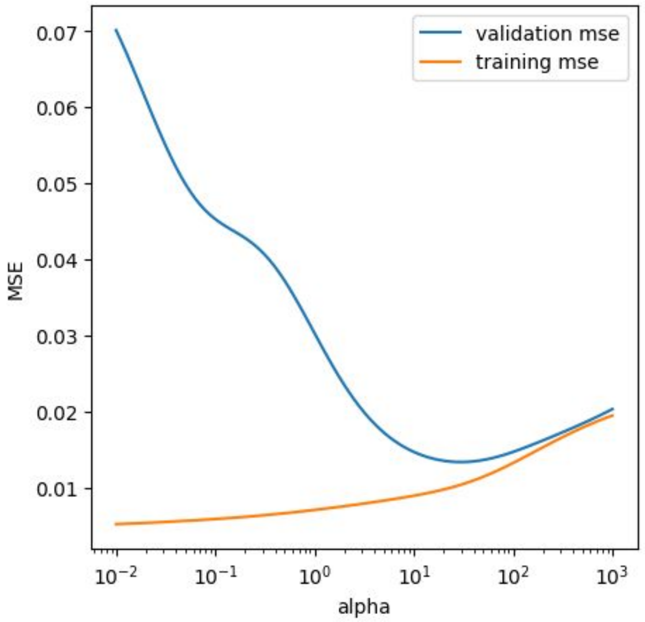
Step 2) Working with dataset

- Get dataset and split into training, testing dataset

Step 3) create polynomial and ridge regression model

- transform original input features into polynomial features
- use ridge regression on transformed polynomial feature set

Step 4) Train / tune model



Evaluation of model

- MSE (mean squared error) is metric used to evaluate regression models
  - Measures the average of the squares of the errors

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

n = number of samples in training dataset

y_i = ground truth

$\hat{y_i}$ = predicted value

  - Lower MSE indicates better model
  - From polynomial regression model, MSE = 0.0097

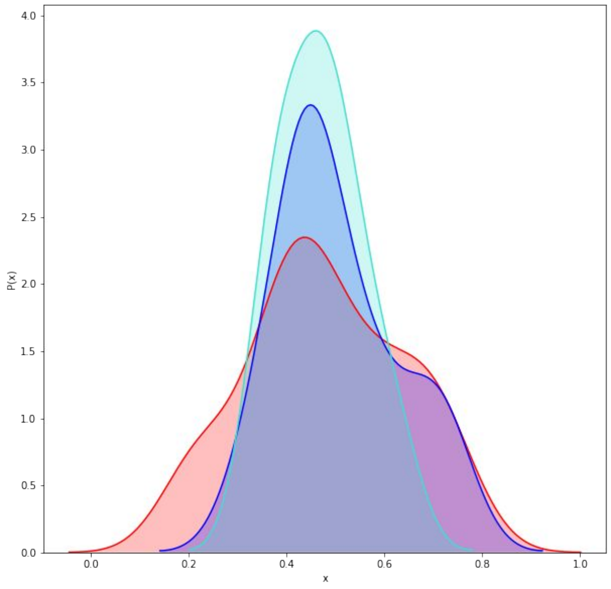


Figure 4: Polynomial Ridge Regression Probability Distribution

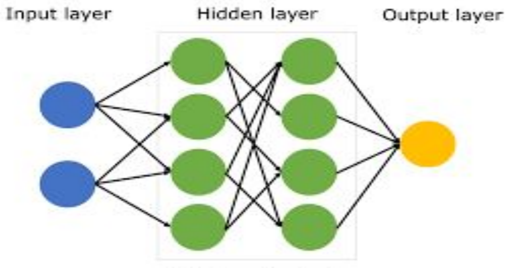
(Red: Training, Blue: Testing, Turquoise: Predicted)

Problem 2: Can student's video-watching behavior be used to predict student's performance?

All students that complete at least half of quizzes were used (students who watch at least half of total videos)

Simple Neural Network

- A neural network is a sequence of layered neurons in which each neuron is a linear function with a non-linear activation function.
- Key feature is neurons can learn from process called back propagation which uses an error/loss function to guide network in its learning process
- Allows model to fit non-linear features of dataset
- Result: MSE = 0.0065



Conclusion: From both approaches, there is a relationship between students' video watching behavior and their overall performance on video quizzes considering low MSE values.

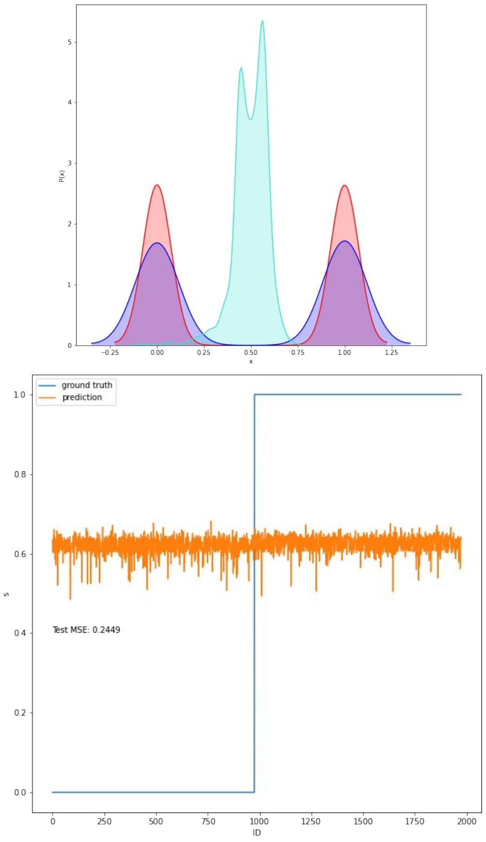
Problem 3: How well can you predict a student's performance on a particular in-video quiz question?

Random Forest classifier

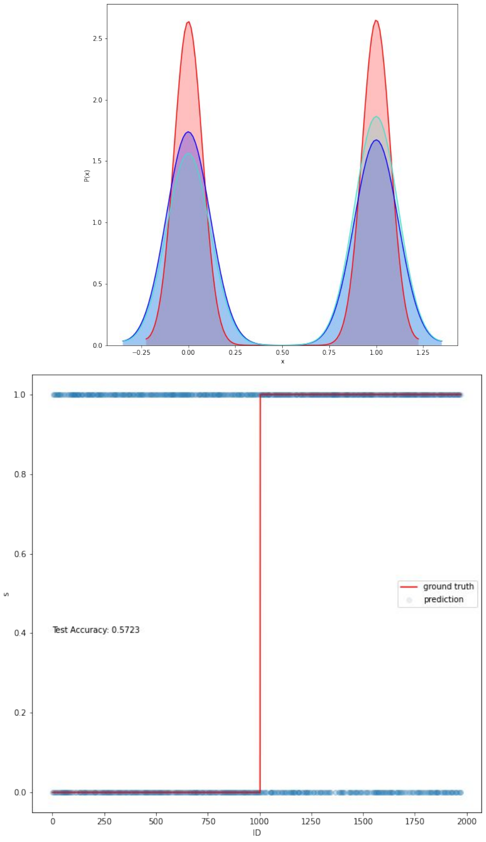
- Purpose: classification model from building a set of decision trees and outputs a class label (0 or 1 since we are interested in binary classification)
- Observations from model: combines multiple individual trees, each tree is trained on a random sample of training data
- Benefits of model:
  - high accuracy
  - less overfitting
- Limitations:
  - complex and computationally intensive

Other models tried:

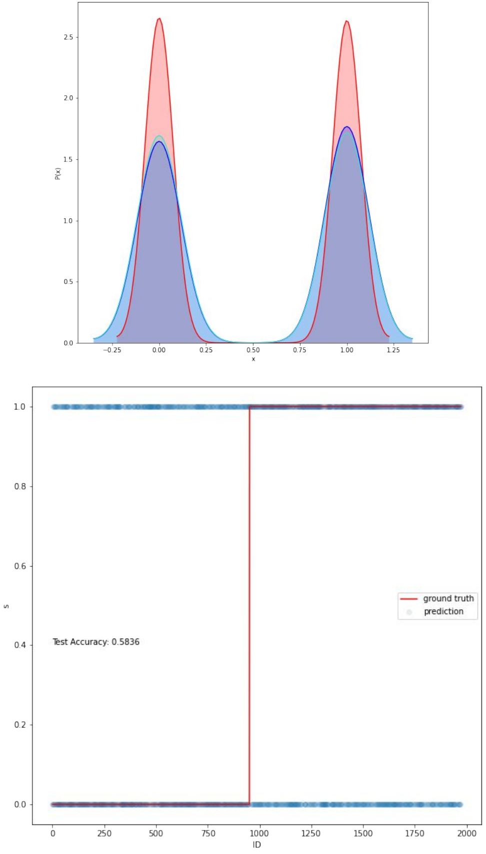
- Linear Ridge Regression model
- Logistic regression model
- Gaussian Bayes model
- kNN model
- Multinomial Naïve Bayes model
- Neural network



Linear Regression



Logistic Ridge Regression



Random Forest

Evaluation