CS439 Final Report: Modeling and Interpreting Factors That Influence Student Academic Performance
Github Repo (LINK) | Video (LINK)

Karthik Mudakalli (km1663), Daniel Tang (dt655), Joe Kim (jk1948)

Project Statement: Our project studies what drives student exam performance and how those motivations relate to the way majors and courses are structured at Rutgers. We combined an individual level student dataset with genuine Rutgers level curriculum data and essentially asked two key questions which were "Which behavioral and background factors most correlate exam scores for individual students?" and "How do those patterns line up with the workload and structure of real Rutgers degrees and courses?". At the individual level we work with the Kaggle StudentPerformanceFactors dataset. It includes exam score along with features such as hours studied, attendance, sleep,and other variables. We processed Rutgers course JSON files for Fall 2025 and Spring 2026 for the institutional levels, and a degree requirements CSV for each major. From these we construct course level and degree level workload summaries: credits, average weekly contact hours, number of meeting days, earliest start time and latest end time, along with a label such as STEM, Humanities, Business and so on. So this lets us move between two levels which are the micro level for student behaviors and outcomes and macro level (curriculum design, major requirements and course scheduling). The project relates to topics from CS439 such as data cleaning and feature engineering (flattening JSON, constructing course and degree features), supervised learning with linear regression and random forests, model evaluation with MAE and R^2, feature importance and permutation importance, unsupervised learning with autoencoders, PCA and KMeans clustering We treat the Kaggle regression task as a supervised problem, and then use Rutgers data to contextualize what the model learns about the factors.
Novelty and importance: Students and instructors often talk about success in vague terms like "work harder" or "this major is brutal."(or just more intuitive thoughts). Our goal is to replace some of that intuition with actual data driven evidence.. The Kaggle dataset gives a controlled environment. It lets us estimate how much exam scores move when study hours or attendance change while holding other variables constant. The Rutgers data then grounds these patterns in the reality of majors and course structures. Instead of just saying "more hours studied help," we can ask what that implies for majors with heavy weekly contact time compared to lighter ones. This is useful for students choosing majors or course loads, advisors thinking what courses to recommend, instructors who want to understand how much structure and contact time they are building into their courses. There is a large literature on educational data mining that uses regression or tree based models to predict academic performance from demographic and behavioral features. Many of those works use limited datasets and usually stop at individual level prediction. Our approach differs in two ways. We compared a simple linear model to a more complex random forest and then to deep autoencoders, using the same cleaned feature space. Also, we connected the Kaggle model to real curriculum structure at Rutgers by simulating actual students for each major, based on metrics such as average weekly contact hours and number of meeting days. There is also previous work on representation learning for course catalogs, mostly using text descriptions. Our autoencoder instead works on structural features like credits and contact hours, which highlights load and schedule rather than content. Common issues in data science for education include: treating models as black boxes without examining feature importance, ignoring the data generating process when making simulations, and reporting high accuracy on data sets that resemble inaccurate and unreal institutions We tried to address these by finding both tree based importance and permutation importance, and interpreting them in plain language, explicitly examining how our simple workload to study hours mapping creates correlations in the simulated degree level results, and being honest about the limitations and biases of the Kaggle dataset and the heuristic mapping from degrees to Kaggle features

Progress and Contribution: All three members contributed to planning, debugging, and refining the pipeline, but each focused on different parts of the project. Daniel handled the Kaggle dataset and the supervised learning side, building the Linear Regression and Random Forest pipelines, running correlation and permutation importance analyses, and developing the simulation framework that mapped degree workloads into Kaggle style

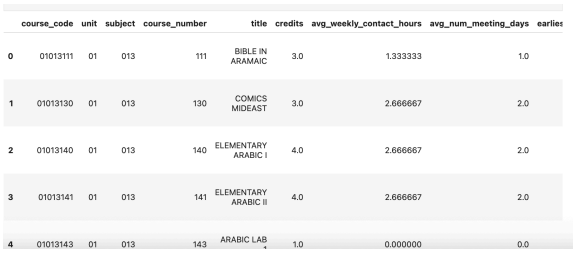
features. Joe focused on the Rutgers data engineering, cleaning the course JSON files, constructing section and course level workload features, parsing degree requirements, extracting course lists, and producing the aggregated degree summary table and discipline labels. Karthik led the unsupervised modeling, implementing and training the shallow and deep autoencoders, generating PCA comparisons and KMeans clustering results, and interpreting latent space structure through visualizations and loss curves. Together, these contributions formed a complete workflow covering data preparation, supervised prediction, unsupervised representation learning, and interpretation across both student level and institutional level datasets.

Models and Algorithms: The Kaggle dataset StudentPerformanceFactors has 6,607 rows and 20 columns. It includes numeric variables such as Hours_Studied, Attendance, Sleep_Hours, Previous_Scores, Tutoring_Sessions and Physical_Activity along with categorical variables such as Motivation_Level, Access_to_Resources, Parental_Involvement, Family_Income, and others. Exam_Score is the target.We keep all rows and treat Exam_Score as a continuous target. For modeling, we split off Exam_Score as y, treat the six obvious quantitative variables as numeric, treat all remaining columns as categorical, and build a ColumnTransformer that passes numeric features through unchanged and one hot encodes categorical features using scikit learn's OneHotEncoder with ignore for unseen categories. We then create an 80/20 train test split using a random seed 42 so that results are reproducible. We used three institutional data sources (combined_courses_fall2025.json, combined_courses_spring2026.json, departments.json) that maps subject codes to department names. Each course offering in the JSON has a unit code, subject, number, core codes, title, credits and a list of sections, where each section has meetingTimes describing day of week, startMinute, endMinute and meetingModeDesc.We first clean these files into a section level table where each row is a single section. We find num_meeting_days as the number of distinct days for that section. We compute weekly_contact_minutes as the sum of (endMinute-startMinute) over all meetings. We record earliest_start_min and latest_end_min. We join all meeting modes into a semicolon separated string. We get 7,695 fall sections and 9,758 spring sections. We then aggregate to a course level table where standardize unit, subject and course_number as zero padded strings and define course_code = unit + subject + course_number, group by course_code and find first unit, subject, course_number, title and credits, average weekly_contact_minutes converted to hours, mean num_meeting_days, minimum earliest_start_min and maximum latest_end_min, union of all meeting_modes

Course features: (5106, 11)

	course_code	unit	subject	course_number	title	credits	avg_weekly_contact_hours	avg_num_meeting_days	earlies
0	01013111	01	013	111	BIBLE IN ARAMAIC	3.0	1.333333	1.0	
1	01013130	01	013	130	COMICS MIDEAST	3.0	2.666667	2.0	
2	01013140	01	013	140	ELEMENTARY ARABIC I	4.0	2.666667	2.0	
3	01013141	01	013	141	ELEMENTARY ARABIC II	4.0	2.666667	2.0	
4	01013143	01	013	143	ARABIC LAB 1	1.0	0.000000	0.0	

This forms 5,106 unique courses with summary workload features. Using departments.json we attach department_name and then a label derived from simple keyword rules such as mapping anything containing "ENGINEERING" or "MATHEMATICS" to STEM, "HISTORY" or Other.



	course_code	unit	subject	course_number	title	credits	avg_weekly_contact_hours	avg_num_meeting_days	earlies
0	01013111	01	013	111	BIBLE IN ARAMAIC	3.0	1.333333	1.0	
1	01013130	01	013	130	COMICS MIDEAST	3.0	2.666667	2.0	
2	01013140	01	013	140	ELEMENTARY ARABIC I	4.0	2.666667	2.0	
3	01013141	01	013	141	ELEMENTARY ARABIC II	4.0	2.666667	2.0	
4	01013143	01	013	143	ARABIC LAB	1.0	0.000000	0.0	

The degrees.csv file lists 508 degrees with an ID, Name, DegreeCode, StartTerm and a requirements JSON string.

```
Degrees shape: (508, 5)
```

	ID	Name	Degree Code	Start Term	requirements
0	369	Degree Audit for Engineering Five - Year	NB	Fall 2003	NaN
1	6061	Major in Social Work	NB	NaN	[{"id": "R1", "logic": "and", "title": "Prereq...
2	2394	Degree Audit for Bioenvironmental Engineering ...	NB	Fall 2019	[{"id": "R1", "logic": "and", "title": "First ...
3	4542	Applied Sciences in Engineering - Packaging Op...	NB	Fall 2020	[{"id": "R1", "logic": "and", "title": "First ...
4	372	Undeclared School of Environmental and Biologi...	NB	Fall 2007	NaN

We parse each requirements JSON, walk through its blocks and requirement entries, and extract course_pool codes. The resulting sorted unique list of codes is attached as course_codes for each degree.

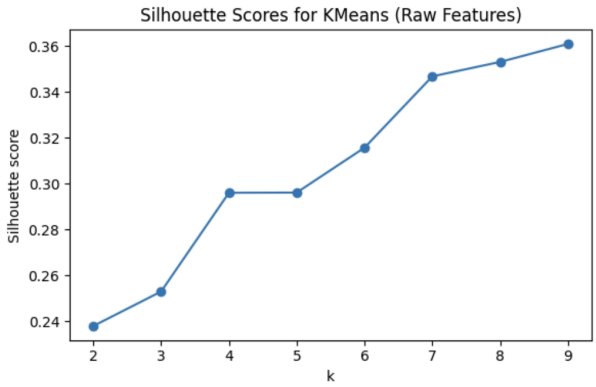
	Name	course_codes
0	Degree Audit for Engineering Five - Year	[]
1	Major in Social Work	[01119150, 01119154, 01119160, 01119182, 01830...
2	Degree Audit for Bioenvironmental Engineering ...	[01119103, 01160159, 01160160, 01160161, 01160...
3	Applied Sciences in Engineering - Packaging Op...	[01160159, 01160160, 01160171, 01160209, 01160...
4	Undeclared School of Environmental and Biologi...	[]
5	Degree Audit for Major in African, Middle East...	[01013201, 01013211, 01013221, 01013231, 01013...
6	Degree Audit for Pre-med	[]
7	Degree Audit for Undeclared	[]
8	Biomedical Engineering (5 year)	[01119115, 01119117, 01160159, 01160160, 01160...
9	Aerospace Engineering	[01160159, 01160160, 01160171, 01355101, 01640...

We then summarize each degree by looking up its required courses in the course_features table and averaging number of mapped courses, mean credits, mean avg_weekly_contact_hours, mean avg_num_meeting_days.

	Name	num_mapped_courses	avg_credits	avg_weekly_contact_hours	avg_num_meeting_days
0	Degree Audit for Engineering Five - Year	0.0	NaN	NaN	NaN
1	Major in Social Work	25.0	3.400000	1.739013	0.919346
2	Degree Audit for Bioenvironmental Engineering ...	50.0	2.780000	2.915657	1.995677
3	Applied Sciences in Engineering - Packaging Op...	41.0	2.951220	2.943991	1.826664
4	Undeclared School of Environmental and Biologi...	0.0	NaN	NaN	NaN
5	Degree Audit for Major in African, Middle East...	3.0	3.000000	2.666667	2.000000
6	Degree Audit for Pre-med	0.0	NaN	NaN	NaN
7	Degree Audit for Undeclared	0.0	NaN	NaN	NaN
8	Biomedical Engineering (5 year)	32.0	2.593750	2.923252	1.968253
9	Aerospace Engineering	41.0	2.853659	2.886441	2.022962

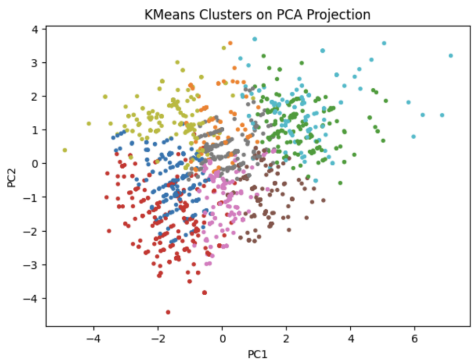
Experimental Design: This produces a deg_summary table measuring how "heavy" each major is in terms of contact hours and meeting frequency. For supervised modeling, we use Linear Regression and a 200 tree Random Forest within the same preprocessing pipeline and evaluate both with MAE and R^2 on the held out test set. We assess feature influence through Pearson correlations, Random Forest Gini importances, and permutation importance. To connect these models to Rutgers majors, we map each degree's workload into synthetic Kaggle style profiles using linear rules for Hours_Studied and Attendance, bounding both to reasonable ranges. Feeding these profiles into the Random Forest yields predicted exam scores for all degrees, and we also vary Motivation_Level and Teacher_Quality for one major to produce a 3x3 scenario heatmap. For unsupervised structure, we train shallow and deep autoencoders on five course workload features, standardizing all inputs and producing 2D latent embeddings for 3,624 courses. The shallow model uses a 32–16–2 architecture, while the deep model uses 128–64–16–2 with dropout, noise, and 500 epochs of training. We visualize both latent spaces and compare them with PCA (two components, reporting explained variance) and

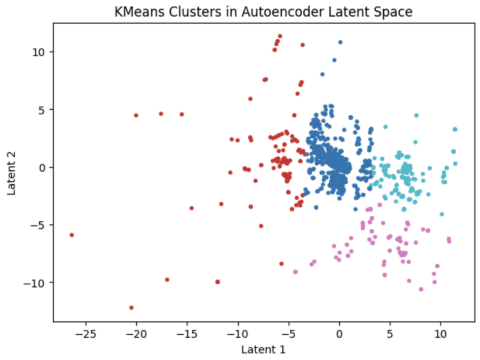
with KMeans clustering on both the raw standardized features and the autoencoder latent space, selecting k via silhouette scores.



We also run KMeans on the deep autoencoder latent space Z for the same k range, finding best k in latent space (which is 4) and storing cluster_latent labels

Below are some of our results:





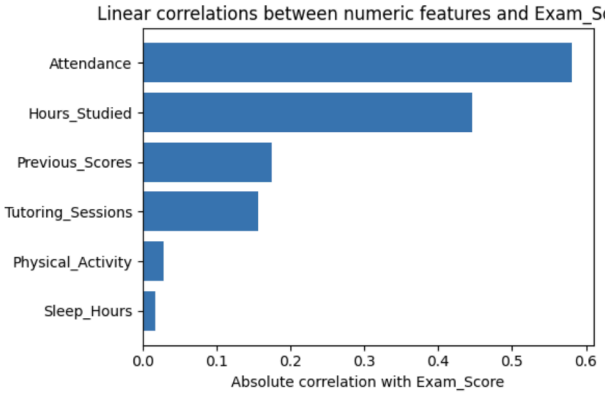

```python
#Deeper, slower, more expressive autoencoder
class BigAutoencoder(nn.Module):
    def __init__(self, input_dim):
        super().__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 128),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(64, 16),
            nn.ReLU(),
            nn.Linear(16, 2)   # latent layer
        )

        self.decoder = nn.Sequential(
            nn.Linear(2, 16),
            nn.ReLU(),
            nn.Linear(16, 64),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(64, 128),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(128, input_dim)
        )

    def forward(self, x):
        z = self.encoder(x)
        out = self.decoder(z)
        return out, z
```

KMeans on Autoencoder Latent Space

```python
Z_km = Z  # already scaled latent vectors

sil_scores_z = []
for k in K_range:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(Z_km)
    sil = silhouette_score(Z_km, labels)
    sil_scores_z.append((k, sil))

best_k_z = max(sil_scores_z, key=lambda x: x[1])[0]
print("Best k (latent space):", best_k_z)

km_latent = KMeans(n_clusters=best_k_z, random_state=42, n_init=10)
cf["cluster_latent"] = km_latent.fit_predict(Z_km)
```

```
Best k (latent space): 4
```

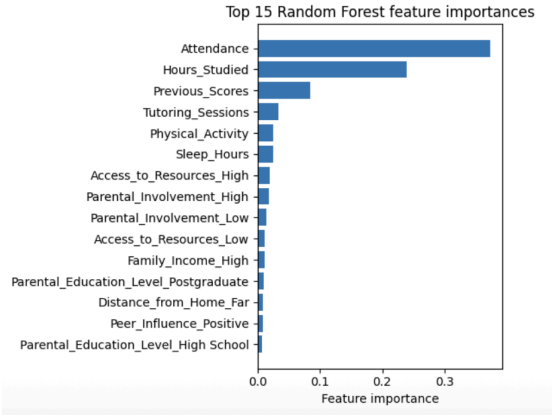
Detailed Analysis of Results and Key Results and Evaluation: This lets us compare linear vs nonlinear representations for clustering.For individual level drivers of exam performance, we did correlation analysis on the Kaggle features shows us that Attendance has a correlation of about 0.58 with Exam_Score, Hours_Studied has a correlation of about 0.45, Previous_Scores and Tutoring_Sessions have much smaller positive correlations, and Physical_Activity and Sleep_Hours are near zero with sleep being slightly negative



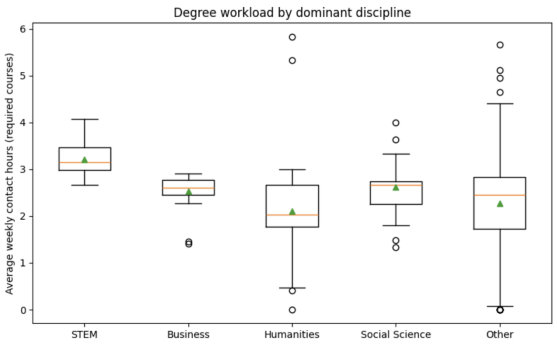

```
Correlation with Exam_Score:
Exam_Score                1.000000
Attendance                0.581072
Hours_Studied             0.445455
Previous_Scores           0.175079
Tutoring_Sessions         0.156525
Physical_Activity         0.027824
Sleep_Hours              -0.017022
Name: Exam_Score, dtype: float64
```

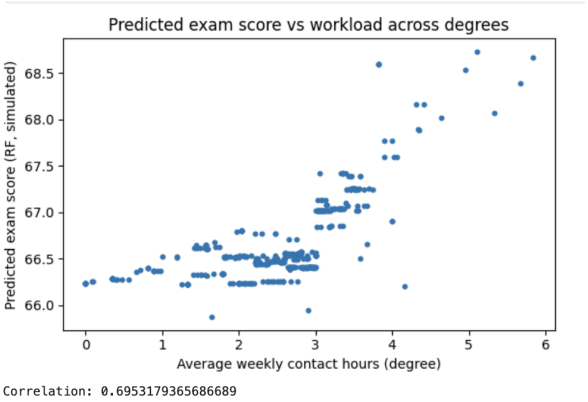
The linear regression model achieves: MAE approx 0.45 and R^2 approx 0.77. The random forest performs worse with MAE approx 1.08 and R^2 approx 0.67. This shows us that the underlying relationships in the dataset are mostly linear and have low noise. Adding tree based complexity just overfits detail rather than improving generalization. Random forest feature importance and permutation importance both reinforce that Attendance and Hours Studied dominate and have a much greater importance than any other variable. Variables like Access_to_Resources_High, Parental_Involvement_High, Family_Income_High and Teacher_Quality_High only contribute a small importance.

	feature	importance
1	Attendance	0.374569
0	Hours_Studied	0.239534
3	Previous_Scores	0.084366
4	Tutoring_Sessions	0.033088
5	Physical_Activity	0.024895
2	Sleep_Hours	0.024207
9	Access_to_Resources_High	0.018669
6	Parental_Involvement_High	0.018005
7	Parental_Involvement_Low	0.013118
10	Access_to_Resources_Low	0.011244
19	Family_Income_High	0.010878
35	Parental_Education_Level_Postgraduate	0.009203
37	Distance_from_Home_Far	0.007924
30	Peer_Influence_Positive	0.007638
34	Parental_Education_Level_High School	0.007171
22	Teacher_Quality_High	0.006806
28	Peer_Influence_Negative	0.006598
20	Family_Income_Low	0.006506
14	Motivation_Level_High	0.006458
15	Motivation_Level_Low	0.006368



Overall, the supervised models show that routine behavior like attending class or studying explains most of the variation in exam scores. Background and environment variables mostly tweak predictions by small amounts. The degree level workload plots show clear differences across disciplines. STEM majors generally have higher average weekly contact hours while Business majors have moderate levels. Humanities majors have a wide spread with some low and some high workloads, and Social Science majors land in the middle. “Other” degrees are diverse with many outliers.

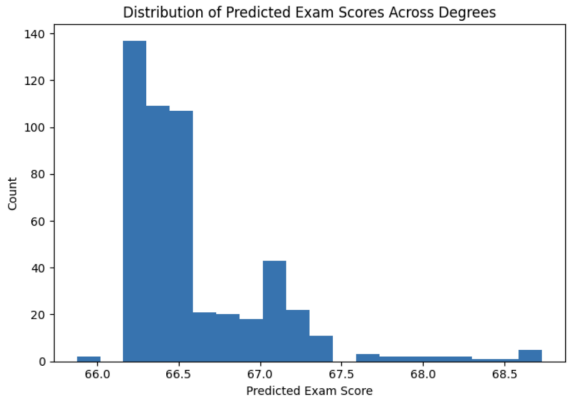


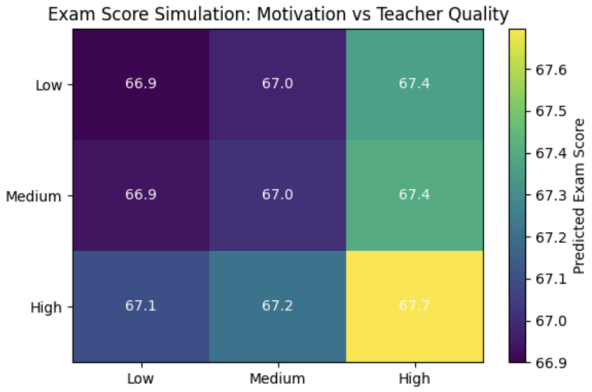


When we run the workload based simulation and feed the synthetic degree profiles into the random forest, we see predicted exam scores for almost all degrees lie in a narrow band around 66-69, degrees with the highest predicted scores tend to have higher contact hours, often 5+ hours per week, and degrees with the lowest predicted scores tend to have small numbers of mapped courses and low contact hours. The correlation between avg_weekly_contact_hours and predicted_exam_score_rf across degrees is about 0.69. On the surface this seems like strong evidence that heavier majors produce higher scores but in reality it mostly reflects our simulation rule. We explicitly turned workload into more study hours and the Kaggle model rewards higher Hours_Studied. The model has no knowledge of majors themselves.

	avg_weekly_contact_hours	predicted_exam_score_rf
avg_weekly_contact_hours	1.000000	0.695318
predicted_exam_score_rf	0.695318	1.000000

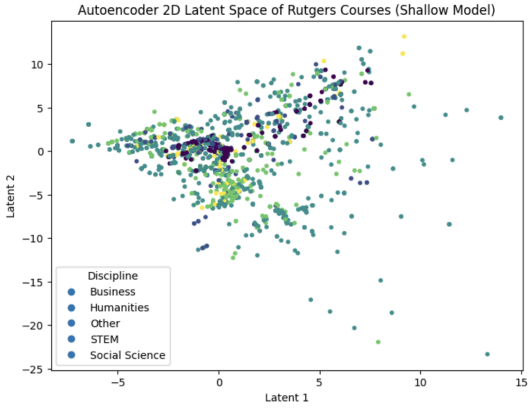
The histogram of predicted scores across all degrees shows a tight unimodal distribution. There is no clear separation between disciplines. This again supports the idea that the model is really responding to our heuristic mapping rather than discovering deep differences across fields.

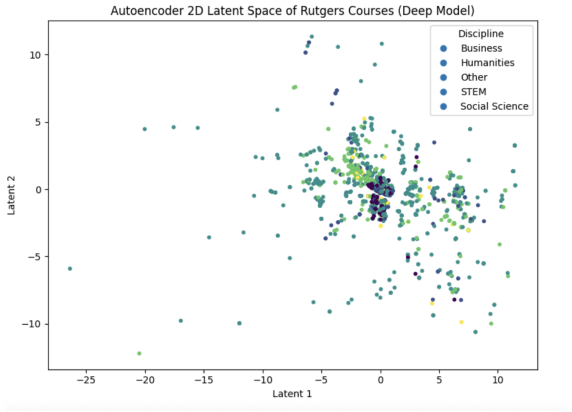




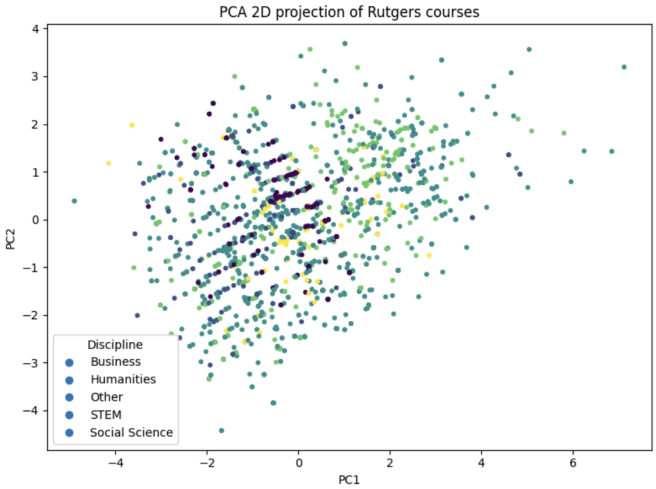
For a selected major whose name contains “Computer Science,” varying Motivation_Level and Teacher_Quality in {Low, Medium, High} gives predicted exam scores between roughly 66.9 and 67.7. Each step up in motivation or teacher quality yields a small improvement, and the best combination is high motivation with high teacher quality. The 3 by 3 heatmap below is smooth and monotone. We noticed no sharp nonlinearities or interactions. This tells us that in the Kaggle model these two categorical features behave like gentle add ons on top of the main drivers of attendance and study hours. The shallow autoencoder shows a quick drop in reconstruction loss from about 101 to about 14 over 50 epochs. The loss flattens early. The corresponding latent map of 3,624 courses in 2D has a central cluster with several diffuse branches. Disciplines

overlap heavily, and outliers show up along both axes. The model captures coarse workload patterns but does not separate nuanced scheduling types.





The deep autoencoder trains much longer. Total loss starts around 98 and gradually decreases to about 8.7 by epoch 450. The slower, more gradual improvement suggests that the deeper architecture with dropout and noise is learning a more expressive compression. The latent scatter plot still has one dense core corresponding to standard Rutgers courses with moderate credits and normal schedules, but outlier courses with unusual loads or times are pulled further away. The branches and isolated groups are easier to see than in the shallow model, and KMeans on this latent space prefers only four clusters, compared to nine clusters on the raw standardized features. This suggests that autoencoder smooths space and makes cluster structure clearer. PCA on the same five features produces a more symmetric cloud, where PC1 and PC2 account for about 34% and 31% of the variance. The PCA plot still shows a central mass with outliers but lacks some of the curved branches visible in the autoencoder maps. At the same time PCA is highly interpretable and cheap to compute.




```python
from sklearn.decomposition import PCA

X_pca = cf[auto_features].values.astype(np.float32)
X_pca_scaled = scaler.fit_transform(X_pca)

pca = PCA(n_components=2)
Z_pca = pca.fit_transform(X_pca_scaled)

plt.figure(figsize=(8,6))
plt.scatter(Z_pca[:,0], Z_pca[:,1], c=colors, s=10)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("PCA 2D projection of Rutgers courses")

handles = [plt.Line2D([], [], marker="o", linestyle="", label=d, markersize=6)
           for d in unique_disc]
plt.legend(handles=handles, title="Discipline")

plt.tight_layout()
plt.show()

print("Explained variance ratio:", pca.explained_variance_ratio_)
```

PCA and the autoencoders show that Rutgers course structure is dominated by one large cluster with a set of scattered unusual courses while deep autoencoder provides cleaner separation of those outliers through nonlinear modeling. Key strengths of the project include the full pipeline from raw JSON to structured course and degree features, direct comparison of linear and nonlinear supervised models, the use of multiple interpretability tools, and linking individual level predictions to institutional structures through a transparent workload based mapping. Limitations include the synthetic nature of the Kaggle dataset, the hand tuned mapping from workload to study effort, the lack of real instructor quality data (SIRS), and the fact that our autoencoders model only five structural attributes and therefore miss content and enrollment based complexity. The weaker performance of the random forest compared to linear regression also limits the benefit of more complex interactions for the supervised task.

Changes after proposal: We originally planned to use Kaggle data along with Rutgers SIRS ratings and to store everything in PostgreSQL. In the end we used only the Kaggle dataset and the Rutgers course and degree files, since no reliable instructor level data was available and database integration added little value. The supervised part stayed with linear regression and random forest as proposed, but the unsupervised portion expanded to include shallow and deep autoencoders, PCA and KMeans. We also shifted away from validating Kaggle predictions against Rutgers outcomes and instead used Rutgers data to simulate and contextualize model behavior, since no real exam or GPA data exists for the catalog.

Conclusion: Our results show that in the Kaggle dataset exam performance is mostly driven by attendance and hours studied, and linear regression captures these patterns better than a more complex random forest. By engineering workload features for Rutgers courses and degrees, we link these insights to major structures. The simulation framework illustrates how heavier degrees would map to higher predicted scores under a model that rewards study time, though this effect is driven by our hand written mapping rather than true causal signals. Deep autoencoders and PCA give a compact view of the course catalog, revealing a large core of standard courses plus distinct unusual offerings such as labs or once weekly evening classes. Clustering in the autoencoder space suggests that structural families of courses exist even when academic disciplines overlap.

TA Xi Zhu question from Oral Exam: You tried a lot of linear and deep learning models. Which do you think is the best? And is there any talk about the trade off between the complexity and resources needed?

We tested linear models, tree models, PCA, and shallow and deep autoencoders, and the best model depends on the task. For predicting exam scores, Linear Regression was clearly strongest, reaching R^2 approx 0.77 and MAE approx 0.45, while Random Forest performed worse with R^2 approx 0.67 and MAE approx 1.08, despite being more complex and more computationally expensive. For representing Rutgers course structure, nonlinear models were better. PCA captured only linear variation, while the deep autoencoder trained over 500 epochs reduced reconstruction loss from approx 98 to approx 8, producing a smoother and more expressive 2D latent space that separated unusual course types more clearly than the shallow model. The trade off is that deep models require far more training time and tuning and heavier resources (GPU). In short, Linear Regression is best for prediction and efficiency, while the deep autoencoder is best for uncovering nonlinear structure, showing how increased model complexity raises resource costs but can reveal patterns simpler methods miss.

Future work could replace the synthetic Kaggle dataset with real Rutgers student records if available under proper privacy constraints, learn the workload to effort mapping from data instead of fixing it by hand, incorporate additional course level features such as enrollment, failure rates or text embeddings of descriptions, and connect autoencoder clusters to student outcome statistics to see whether structural course families align with difficulty or grade distributions. Overall, the project meets goals by demonstrating data cleaning, supervised and unsupervised modeling, careful evaluation and critical discussion of limitations, while tying all of these tools back to a question that matters to students and instructors.

Acknowledgements: Kaggle data by Lai Ng - SOURCE and a sincere thanks to Akash Dubey (ad2046), Ayan Zia (az610), Ani Tiwary (at1610) for providing the Rutgers data.