Are STEM Classes Substantively Different from non-STEM Classes?

Evelyn Kimbirk, Cameron Faulkner, and Shenova Davis

Introduction and Motivation

Keeping with the ever-growing demand for college graduates in STEM fields, the rate of students graduating with a degree in STEM subjects has increased by 30% in the period from 2006 to 2015, while those graduating with humanities degrees decreased by 20% (American Academy of Arts and Sciences, 2017). This change in the chosen degree paths of students may be dramatically altering the nature of the average student's college experience as STEM classes have been commonly thought of as more difficult, time-consuming, and yielding lower grades overall than non-STEM classes.

As the share of students receiving STEM degrees increases, is the nature of the average student's college experience changing? To put to rest anecdotal generalizations between STEM and non-STEM classes, we must answer the question, "Are STEM classes substantively different than non-STEM classes?".

In order to answer this question, we took survey results from UC San Diego's Course and Professor Evaluations (CAPEs) and applied a support vector clustering algorithm to determine if a given class could be accurately labeled as either a STEM or non-STEM class. Consistent success in doing so would indicate that STEM classes are substantially different than non-STEM classes while a failure suggests that they are fairly similar.

Related Work:

Existing research on the differences between STEM and non-STEM classes has centered around answering the question, “Are STEM courses harder?”. This obviously has great relevance to the question we, ourselves, are looking to answer, but still may not do so comprehensively.

Tomkin and West’s paper “STEM Courses are Harder…”, provides the most relevant answer to our question. In the paper, the authors looked to create a predictive model of student academic performance to determine if students of equivalent academic ability were receiving lower grades in STEM classes. Noting the limitations of using observed GPA to measure academic performance, their predictive model is based on a weighted logistic model of GPA which calculates a given student’s expected grades in all courses offered. In order to determine if STEM classes were more difficult, they compared the predicted GPA for students and the observed grades received and concluded that STEM classes were indeed more difficult.

Difficulty measured by grades received, however, is not the be-all and end-all of a student’s experience. Factors like the hours spent on a given class, the degree to which students recommend a class, and the extent to which they like their professors offer a more holistic assessment of how students’ lives play out while getting their degree. As such, we believe an alternative methodology should be used to answer our research question.

Methods:

Data Collection and Labeling

In order to gauge undergraduate student experience in various classes, we analyzed student evaluation results of the classes offered at UC San Diego. As a large university with a diverse student body, we believe the course evaluations at UCSD should be fairly well representative of the average American university's undergraduate student body.

To gather our data, we turned to a fellow UCSD undergraduate project, SeasCAPE, which had scrapped five years' worth of student evaluations dating from Fall 2017 to Summer Session 2020, totaling 9981 observations.

For each class instance, the dataset includes a variety of metrics, but the ones we found most relevant to the student experience and, as such, used in our model, were as follows: percentage of students who recommend a class, instructor recommendation rate, hours spent on the class, and the GPA of the class.

We chose to label classes as STEM, or not, based on the department that offered them. The dataset includes, of course, the name of the class evaluated, but not explicitly the department it was offered in nor an evaluation of whether it was a STEM class or not. To make our determination, we compiled a list of departments we believe to be STEM, and their course codes, so we could individually label courses as STEM or not to train and evaluate our model.

"STEM" Department	**Department Code(s)**
Biology	BILD, BIBC, BIEB, BIMM, BIPN
Bioengineering	BENG
Chemical Engineering	CENG
Chemistry	CHEM
Cognitive Science	COGS
Computer Science	CSE
Computational Social Science	CSS
Data Science	DSC
Electrical Engineering	ECE

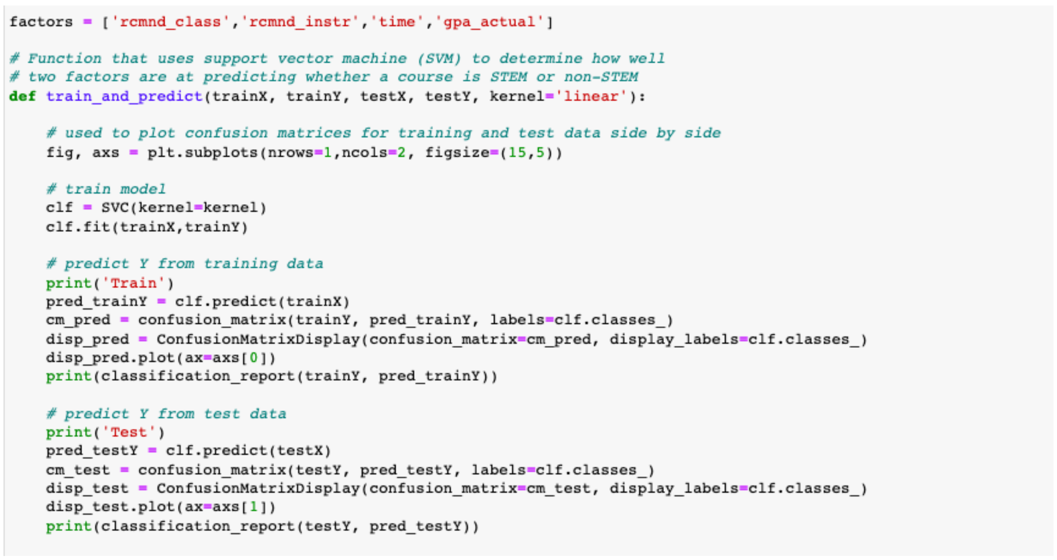
Mechanical and Aerospace Engineering	MAE
Math	MATH
Nanoengineering	NANO
Physics	PHYS
Scripps Institute of Oceanography	SIO
Structural Engineering	SE

Model Selection and Training:

In order to determine if STEM classes were substantively different than non-STEM classes, we trained a machine learning model to predict the type of class based on its training data. Picking a model was difficult, but we eventually settled on using a support vector machine (SVM) classification model. Our research indicated that this algorithm would yield more accurate results than parametric classification models (Winters-Hilt, Merat, 2007).

We created a modeling function that took in as inputs training data and testing data for both X and Y, X being the four factors discussed earlier and Y being a binary value denoting whether the course was a STEM or non-STEM course. Our X and Y values were split into testing and training data using the test_and_split function from the scikit-learn library. This training data for X and Y was then used to train the model.

Finally, we used the model to predict whether the courses in our test data were STEM or non-STEM using only the X variables for both testing and training data. Each of these predictions was compared to the actual labels for STEM vs non-STEM using both a confusion matrix and a classification report.

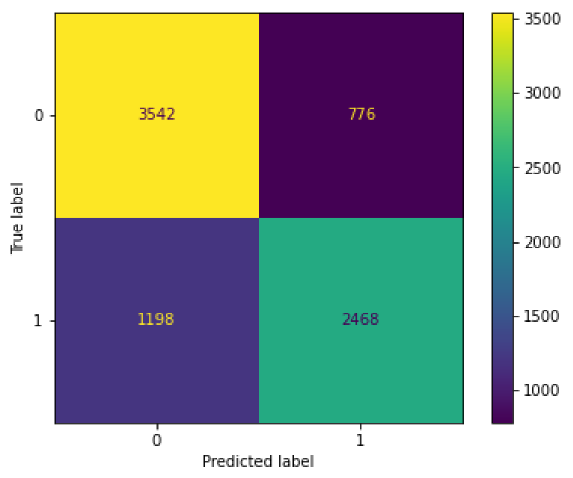

```python
factors = ['rcmnd_class','rcmnd_instr','time','gpa_actual']

# Function that uses support vector machine (SVM) to determine how well
# two factors are at predicting whether a course is STEM or non-STEM
def train_and_predict(trainX, trainY, testX, testY, kernel='linear'):

    # used to plot confusion matrices for training and test data side by side
    fig, axs = plt.subplots(nrows=1,ncols=2, figsize=(15,5))

    # train model
    clf = SVC(kernel=kernel)
    clf.fit(trainX,trainY)

    # predict Y from training data
    print('Train')
    pred_trainY = clf.predict(trainX)
    cm_pred = confusion_matrix(trainY, pred_trainY, labels=clf.classes_)
    disp_pred = ConfusionMatrixDisplay(confusion_matrix=cm_pred, display_labels=clf.classes_)
    disp_pred.plot(ax=axs[0])
    print(classification_report(trainY, pred_trainY))

    # predict Y from test data
    print('Test')
    pred_testY = clf.predict(testX)
    cm_test = confusion_matrix(testY, pred_testY, labels=clf.classes_)
    disp_test = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=clf.classes_)
    disp_test.plot(ax=axs[1])
    print(classification_report(testY, pred_testY))
```

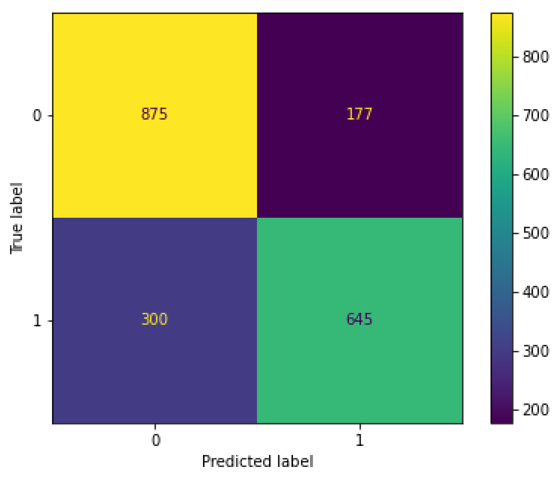
Training and testing function

Results:

Train

```
precision    recall  f1-score   support


           0       0.75      0.82      0.78      4318
           1       0.76      0.67      0.71      3666


    accuracy                           0.75      7984
   macro avg       0.75      0.75      0.75      7984
weighted avg       0.75      0.75      0.75      7984


Test
              precision    recall  f1-score   support


           0       0.74      0.83      0.79      1052
           1       0.78      0.68      0.73       945


    accuracy                           0.76      1997
   macro avg       0.76      0.76      0.76      1997
weighted avg       0.76      0.76      0.76      1997
```





Looking at our classification results on our test data, we were able to correctly label a STEM class as a STEM class 78% of the time, while we correctly labeled non-STEM classes 74% of the time. Overall, our accuracy was 76%. In some difficult to predict applications like sentiment analysis, this might be a fairly good success rate; however, with the binary nature of our classification task, we would be able to correctly predict the right label 50% of the time if we wrote a program that simply guessed answers. With this in mind, our model is not a great predictor of the nature of the classes.

That being said, we are looking to uncover if there is a substantive difference between STEM and non-STEM classes and while our model's prediction rate is not high, it still hints at a weak underlying trend within the data. We can safely dismiss this as the result of overfitting and insufficient data given our nearly identical accuracy scores between our training and test data, and the high number of observations fed into the model.

Given our model's slight degree of success in differentiating STEM and non-STEM classes, but its inability to do so at a high rate despite the inclusion of many data points, we conclude that the student experience differs slightly between STEM and non-STEM classes, but not heavily.

Discussion:

We feel that CAPEs data offers valuable insight into the student experience and that our nearly 10,000 class instances were sufficient to make conclusions. However, a potential threat to the larger validity of our conclusion could lie in our choice of UCSD as a case study. As the university is known as a very STEM-focused institution, there could be potential for selection bias of undergraduates choosing and being admitted into the university enjoying and performing in STEM classes at a higher level than the average college student. Potentially, we may see students spending more time, getting lower grades, and enjoying STEM classes less in other universities. In order to remedy this, our analysis could be expanded to include other universities, especially those with different specialties.

A factor to consider is which departments we define as “STEM” and “non-STEM.” While some classes can clearly be categorized as STEM or non-STEM (for example, physics is clearly STEM while history is clearly non-STEM), some classes and departments are more disciplinary (such as Computational Social Science). It should also be noted that some departments, whether STEM or non-STEM, offer courses that are different from their department category. One example of this is the Cognitive Science Department, which contains courses that can be considered non-STEM (such as COGS 156: Language Development) and STEM (such as COGS 108: Data Science in Practice). A further refinement of our study could consider dividing STEM and non-STEM courses along these factors, as opposed to defining them solely by their parent department.

Works Cited

Cao, David, and Doan, Tung, “Seascape”. Github, March 6, 2021. https://github.com/dcao/seascape.

“Humanities Indicators: STEM Fields Growing among Four-Year College Degree Recipients.” *American Academy of Arts & Sciences*, https://www.amacad.org/news/humanities-indicators-stem-fields-growing-among-four-year-college-degree-recipients.

Tomkin, Jonathan H., and Matthew West. “Stem Courses Are Harder: Evaluating Inter-Course Grading Disparities with a Calibrated GPA Model - International Journal of STEM Education.” *SpringerOpen*, Springer International Publishing, 17 Mar. 2022, https://stemeducationjournal.springeropen.com/articles/10.1186/s40594-022-00343-1.

Winters-Hilt, Stephen, and Sam Merat. “SVM Clustering.” *BMC Bioinformatics*, U.S. National Library of Medicine, 1 Nov. 2007, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2099486/.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, precision_recall_fscore_suppo
```

In [ ]:
```
df = pd.read_csv('/Users/cameronfaulkner/Downloads/CAPES_data.csv')
df
```

Out[ ]:

	Unnamed: 0	instr	course	term	evals	rcmnd_class	rcmnd_instr	time	class_weight
0	0	Eldridge, Justin Matthew	DSC 10	S320	7	1.000	1.000	7.07	
1	1	Saad, Andrew S	ECE 100	S320	33	0.848	0.727	9.41	
2	2	Lockton, Marie	EDS 115	S320	7	1.000	1.000	5.07	
3	3	Stephens, Ramon L	EDS 117	S320	7	1.000	1.000	6.21	
4	4	Jones, Gabrielle Anastasia	EDS 125	S320	14	0.500	0.786	10.64	
...	...	...	...	...	...	...	...	...	
9976	9976	Adler, Amy	VIS 80	FA17	53	0.962	0.961	8.58	
9977	9977	Cartwright, Lisa	VIS 84	FA17	115	0.858	0.864	4.31	
9978	9978	McCleary, Keith Long	WCWP 100	FA17	18	1.000	0.944	8.39	
9979	9979	Blomstedt, Elizabeth Ann	WCWP 100	FA17	12	0.909	1.000	5.95	
9980	9980	Gagnon, Jeffrey C	WCWP 10A	FA17	509	0.659	0.800	5.59	

9981 rows × 14 columns

Determining STEM vs NON-STEM MAJORS

In [ ]:
```
# Shortening courses to just department
def shorten(course):
    department = ''
    i=0
    while course[i] != ' ':
        department +=course[i]
        i+=1
    return department

df['dep'] = df['course'].apply(shorten)
df.loc[(i[0] == 'C' for i in df['dep'])]['dep'].value_counts()
# ^ used to determine what classes are STEM or not STEM
```

Out[ ]:

```
CSE     526
CHEM    414
COGS    302
COMM    248
CENG     90
CAT      73
CHIN     61
CGS      27
CSS       2
CCS       1
Name: dep, dtype: int64
```

In [ ]:

```
def is_STEM(course):

    # NOTE: remove COMM, CHIN, PHIL
    STEM = ['BI','CS','CH','CO','CE','DS','EC','MA','NA','PH','SI','SE']
    if course[0:4] in ('COMM','CHIN','PHIL'):
        return 0
    elif course[0:2] in STEM:
        return 1
    else:
        return 0

df['STEM'] = df['course'].apply(is_STEM)
df
```

Out[ ]:

	Unnamed: 0	instr	course	term	evals	rcmnd_class	rcmnd_instr	time	class_weight
0	0	Eldridge, Justin Matthew	DSC 10	S320	7	1.000	1.000	7.07	
1	1	Saad, Andrew S	ECE 100	S320	33	0.848	0.727	9.41	
2	2	Lockton, Marie	EDS 115	S320	7	1.000	1.000	5.07	
3	3	Stephens, Ramon L	EDS 117	S320	7	1.000	1.000	6.21	
4	4	Jones, Gabrielle Anastasia	EDS 125	S320	14	0.500	0.786	10.64	
...	...	...	...	...	...	...	...	...	
9976	9976	Adler, Amy	VIS 80	FA17	53	0.962	0.961	8.58	
9977	9977	Cartwright, Lisa	VIS 84	FA17	115	0.858	0.864	4.31	
9978	9978	McCleary, Keith Long	WCWP 100	FA17	18	1.000	0.944	8.39	

	Unnamed: 0	instr	course	term	evals	rcmnd_class	rcmnd_instr	time	class_weight
9979	9979	Blomstedt, Elizabeth Ann	WCWP 100	FA17	12	0.909	1.000	5.95	
9980	9980	Gagnon, Jeffrey C	WCWP 10A	FA17	509	0.659	0.800	5.59	

9981 rows × 16 columns

Train The Model

Testing Variables:

1. percent who recommend class,
2. recommend instructor,
3. time spent,
4. gpa actual

In [ ]:

```
df[['rcmnd_class','rcmnd_instr','time','gpa_actual']]
```

Out[ ]:

	rcmnd_class	rcmnd_instr	time	gpa_actual
0	1.000	1.000	7.07	3.44
1	0.848	0.727	9.41	3.04
2	1.000	1.000	5.07	3.89
3	1.000	1.000	6.21	3.94
4	0.500	0.786	10.64	3.88
...	...	...	...	...
9976	0.962	0.961	8.58	3.28
9977	0.858	0.864	4.31	3.48
9978	1.000	0.944	8.39	3.44
9979	0.909	1.000	5.95	3.44
9980	0.659	0.800	5.59	3.12

9981 rows × 4 columns

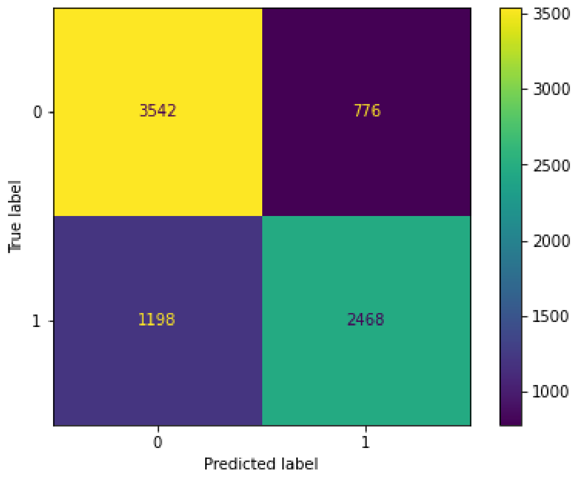
In [ ]:

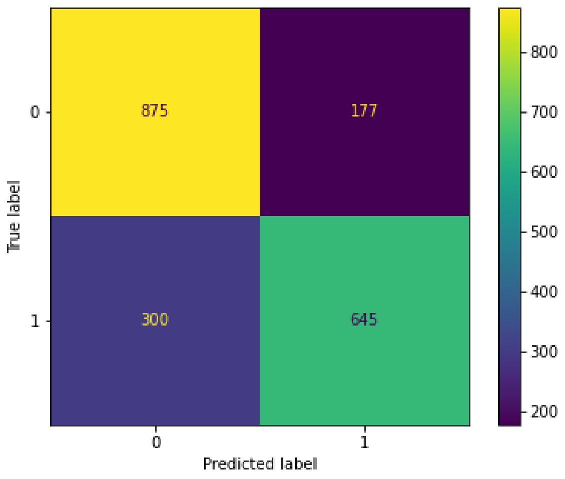
```
factors = ['rcmnd_class','rcmnd_instr','time','gpa_actual']

# Function that uses support vector machine (SVM) to determine how well
# two factors are at predicting whether a course is STEM or non-STEM
def train_and_predict(trainX, trainY, testX, testY, kernel='linear'):

    # used to plot confusion matrices for training and test data side by side
    fig, axs = plt.subplots(nrows=1,ncols=2, figsize=(15,5))

    # train model
    clf = SVC(kernel=kernel)
    clf.fit(trainX,trainY)

    # predict Y from training data
    print('Train')
    pred_trainY = clf.predict(trainX)
    cm_pred = confusion_matrix(trainY, pred_trainY, labels=clf.classes_)
    disp_pred = ConfusionMatrixDisplay(confusion_matrix=cm_pred, display_labels=
    disp_pred.plot(ax=axs[0])
    print(classification_report(trainY, pred_trainY))

    # predict Y from test data
    print('Test')
    pred_testY = clf.predict(testX)
    cm_test = confusion_matrix(testY, pred_testY, labels=clf.classes_)
    disp_test = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=
    disp_test.plot(ax=axs[1])
    print(classification_report(testY, pred_testY))
```

In [ ]:

```
# apply train_and_predict function to the factors
trainX, testX, trainY, testY = train_test_split(df[factors],df["STEM"],test_size
train_and_predict(trainX, trainY, testX, testY,kernel='rbf')
```

```
Train
              precision    recall  f1-score   support

           0       0.75      0.82      0.78      4318
           1       0.76      0.67      0.71      3666

    accuracy                           0.75      7984
   macro avg       0.75      0.75      0.75      7984
weighted avg       0.75      0.75      0.75      7984

Test
              precision    recall  f1-score   support

           0       0.74      0.83      0.79      1052
           1       0.78      0.68      0.73       945

    accuracy                           0.76      1997
   macro avg       0.76      0.76      0.76      1997
weighted avg       0.76      0.76      0.76      1997
```


3542
776
1198
2468
True label
Predicted label
0
1
0
1
3500
3000
2500
2000
1500
1000


875
177
300
645
True label
Predicted label
0
1
0
1
800
700
600
500
400
300
200

```
In [ ]:
```