The Effect of COVID-19 On Student Plans to Pursue Post-Secondary Education

Emily Jin
CS109: Challenge Project
Winter 2021
https://youtu.be/N7-u1awW_Ek

1. Introduction

When the COVID-19 pandemic hit in March of 2020, schools and students around the world transitioned to a virtual learning environment – practically overnight. As of February 2021, most schools have still not re-opened, leaving over 168 million students to continue attending school online.

This unforeseen transition to distance learning caught government organizations and academic institutions off-guard. Although efforts have been made to make education accessible to all students, school closures have inevitably exacerbated pre-existing education inequities in the United States. High-income students are able to afford better internet access, educational resources, and support. For example, in some communities, parents have created "micro-schools" and "learning pods" for their children. These pods are alternative learning environments, typically formed to allow a small group of children to learn in a shared space led by an adult. These provide children with the opportunity to socialize and study in a safe, in-person environment, which is incredibly beneficial during this time of social isolation.

While students from higher-income families are doing well under the circumstances, the same can not be said for low-income communities. Differences in healthcare already make low-income communities more vulnerable to COVID-19, and this likely affects a student's ability to focus on school. In addition to stress due to health and safety concerns, some students from low-income and rural communities also lack access to computers and the Internet, which are fundamental to distance learning. And, others may not have suitable support or living conditions for education.

As a result, many students have suffered significant learning loss this past year – especially younger students, students with disabilities, low-income students, and English language learners. In the past, these students have been less likely to pursue post-secondary education due to a variety of reasons, including a lack of educational support and opportunity. Consequently, their struggles with virtual learning may further discourage them from pursuing higher education. This would present a serious concern, as these students risk falling behind their peers; this will also have severe long-term effects, as many studies indicate a positive correlation between educational attainment and economic prosperity.

Project Objective: With the realization that virtual learning may discourage lower-income communities from pursuing post-secondary education, my project seeks to understand the risks associated with extended online learning by exploring the following question:

> If a student who currently plans to pursue post-secondary education spends T more weeks distance learning, how likely are they to maintain their plans for post-secondary education? If they change them, what are potential factors that may cause this change?

2. Methods

To do this, we will train separate logistic regression models to predict the probability that a student does each of the following – given information about their financial situation and access to technology during the pandemic, and time spent online learning:

1. Maintains original plans to pursue post-secondary education
2. Changes their plans due to having to take care of someone affected by the Coronavirus
3. Changes their plans due to changes to financial aid
4. Changes their plans due to not being able to pay for classes/educational expenses because pandemic-related income changes

2.1 Our Approach

Suppose that we ask n students m questions about their financial and educational situation. Then, we can train 4 Binary Logistic Regression Classifier to map these students' responses to their answers to the above questions. Once we have our classifiers, we can use it to predict the probability that a new student will respond *'Yes'* to any of the 4 question, as long as they answer the same m questions asked previously.

2.2 Mathematical Formalization

2.2.1 The Sample Space: Features and Labels

To formalize this, let $X = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(n-1)}\}$ be our training set of n students. Each student's responses are represented as a m-dimension feature vector: $\mathbf{x}^{(i)} = [x_0^{(i)}, \ldots, x_m^{(i)}]$. So, X can be thought of as an $n \times m$ array, where rows correspond to students (samples) and columns correspond to answers (features).

Now, let $Y = \{\mathbf{y}^{(0)}, \ldots, \mathbf{y}^{(n-1)}\}$ be the set of true label vectors for the training set. Each $\mathbf{y}^{(i)}$ is a 4-dimensional vector equal to $[y_0^{(i)}, y_1^{(i)}, y_2^{(i)}, y_3^{(3)}]$, where $y_j^{(i)} = 1$ means student i answers *Yes* to the $j+1$ question.

2.2.2 The Classifier: Binary Logistic Regression

Logistic regression is used to model the probability that an event Y occurs given X, under the assumption that the probability is modelled by the sigmoid function, $\sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, where θ is a vector of parameters that defines the classifier. In other words, if $h_\theta(\mathbf{x})$ is the classifier, then:

$$h_\theta(\mathbf{x}) = P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x}),$$

For this problem, we want to train a corresponding log-reg model $h_{\theta^k}(\mathbf{x})$ for every component k in the label vector, meaning that that $h_{\theta^k}(\mathbf{x})$ maps any $\mathbf{x}$ to its true label, or y_k:

$$h_{\theta^k} : \mathbf{x} \to y_k$$

To learn the parameters θ^k, we train each classifier to learn the value of θ^k that maximizes the log-likelihood of the data through gradient ascent (we will drop the k for now and just refer to the parameter as θ). The log-likelihood is defined as:

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\theta^T \mathbf{x}^{(i)}))$$

And gradient ascent finds $\arg\max_{\theta_k} LL(\theta_k)$ by updating the parameters according to:

$$\theta'_j := \theta_j + \eta \frac{\partial LL(\theta}{\partial \theta_j} = \theta_j + \eta \sum_{i=0}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

where η is the learning rate. These updates are performed a certain amount of times, at which point the parameters should converge to the argmax.

2.2.3 Normalizing the Features

Many of the features in the dataset that we use are non-binary (which is discussed later), so we should normalize the data before training. Normalization rescales the input data but does not change the distribution, so it does not affect the output or accuracy. However, it helps the model converge faster by improving numerical stability in gradient ascent and speeding up the training process when the features have different ranges.

We will use Min-Max normalization feature-wise, which rescales the inputs of each feature so that they end up ranging between 0 and 1. For a feature j, let $x_{j,min} = \min_i x_j^{(i)}$ be the minimum value of feature j and $x_{j,max} = \max_i x_j^{(i)}$ be the maximum value of feature j over the training set. Then, Min-Max normalization rescales each sample's j-th feature as:

$$x_j^{(i)} \longrightarrow \frac{x_j^{(i)} - x_{j,min}}{x_{j,max} - x_{j,min}}$$

2.3 Dataset

2.3.1 Dataset

For this project, we obtain data from the United States Census Bureau's Household Pulse Survey. This survey is given weekly to households in the United States, and it aims to produce data on the social and economic effects of the coronavirus on American households.

The dataset includes survey results across 11 consecutive weeks, and in total, there were 109,051 participants who responded to 188 questions. Among these, I chose 22 questions related to education plans, financial information, living situation, and access to technology to use as features. I chose these questions since I felt that they would be most directly correlated to the output labels (for the specific questions whose data were used, refer to Appendix A).The labels were chosen to be the answers to the 4 questions listed above.

Since the goal is to study how virtual learning effects students who currently plan to pursue post-secondary education, I defined the sample space to be all participants who responded that they plan to attend an institution for higher education, and who responded to all 22 questions.

After extracting data for the participants in the defined sample space, there were 37,403 samples left and each had 22 features and 4 labels. for the dataset, see `dataset.txt`). This dataset was then split into a training and testing dataset, using a 70-30 split (for the code, see `create_training_test_data()` in `utils.py`, Appendix C). for the datasets, see `training.txt, testing.txt`).

2.3.2 Training, Testing, and Improving

In the training stage, I trained a classifier for each label using the `LogisticRegression` class, which had the instances:

```
def __init__(self, learning_rate, max_steps, weights=None,
             norm_min=None, norm_max=None):
    self.learning_rate = learning_rate # eta in gradient ascent
    self.max_steps = max_steps # number of iterations of gradient ascent
    self.norm_min = norm_min # array of min's used in normalization
    self.norm_max = norm_max # array of max's used in normalization
    self.weights = weights # parameters theta
```

and the functions: `fit()`, `predict()`, `sigmoid()`, `normalize()` (for the full code, see `logistic_regression.py` or Appendix D).

For each classifier, I first used `normalize()` to perform Min-Max normalization on the data, and then I ran `fit()` on the training data (using all 21 features) to get the parameters, `self.weights`, which I saved to a file called `clf.txt`. After that, I ran `predict()` on the testing set to see how well the classifiers performed, and then analyzed the parameters of the classifiers to find a subset of the most influential features. I determined these to be the features of each classifier whose parameter had the largest absolute values, and then trained new classifiers using only those features.

3. Results and Discussion

3.1 Classifiers Using All Features

Overall, the classifiers from the first training did well and achieved the following accuracy rates on the testing set:

0	1	2	3
0.51	0.98	0.84	0.69

Table 1: Accuracy of each logistic regression classifier when trained on 21 features

However, I felt that there was still room for improvement – either through improving the accuracy rates or decreasing the number of features needed. Since the goal is to use these classifiers to predict education plans, it is better to have less features because it makes it easier to identify a cause-and-effect relationship between the features and the outputs. Often, many features are correlated, and therefore, there may be a lot of repeated information in the data.

By looking at the largest absolute values of the weights for each classifiers, I was able to identify 5 features that seemed to be the most informative across all classifiers. These features were:

WEEK, WHYCHNGD5, INTRNT2, INCOME, COMPAVAIL

The feature, WEEK, corresponded to the length of online learning, and the other features corresponded to the following survey questions:

- In the last 7 days, did your household changed spending due to loss of income?
- Are internet services paid for by someone in the household or family?
- In 2019, what was your household income before taxes?
- How often is a computer or other digital device available for educational purposes?

3.2 Classifiers Using Only 5 Features

In the next stage, I trained and tested new classifiers for each label using the same process and datasets as before, but using just the 5 features: WEEK, WHYCHNGD5, INTRNT2, INCOME, and COMPAVAIL. The accuracy rates for the newly trained classifiers on the test set were:

0	1	2	3
0.62	0.98	0.88	0.73

Table 2: Accuracy of each classifier when trained on the 5 features above

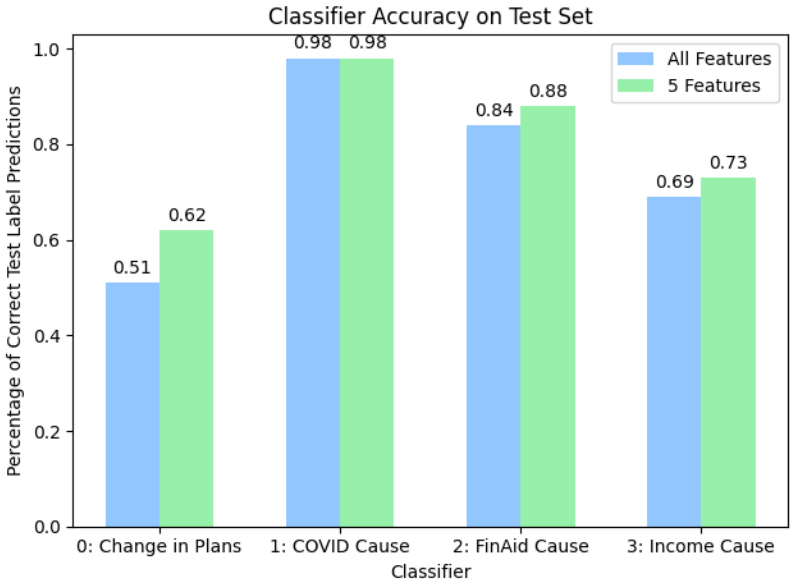

Figure 1: Bar graph comparing the accuracy rates of the 4 classifiers trained on all 21 features versus 5 of the most-weighted features

Furthermore, the above figure provides a side-by-side comparison of the accuracy rates. From the graph, we see that the classifiers trained on the 5 features offered some improvement in accuracy. However, the more useful benefit of these new classifiers was that it was able to achieve better accuracy rates, even with much less features. Using less features sped up the training time, and it also allows us to better understand the relationship between financial and resource-related factors and plans for post-secondary education.

To do so, we need to put the parameters and labels into context. First, recall that classifiers 0, 1, 2, and 3 calculate the probabilities that a given student: maintains education plans, changes plans due to health and safety concerns, changes plans due to changes in financial aid, and changes plans due to income changes – in the following weeks. We also need to understand the meaning of the features and their values, which are provided in the following table:

Feature	Feature Name	Range	Context
1	WEEK	0 - 10	Number of weeks learning online
2	WHYCHNGD5	0, 1	1 = Spending change due to recent income loss
3	INTRNT2	0, 1	1 = Internet is paid for by someone in the household
4	INCOME	0-7	Bucketed incomes. Higher = higher income
5	COMPAVAIL	0-4	Higher = less access to computer

Table 3: Range indicates the min-max of the feature values, and context provides the meaning of the feature values

Feat	Clf 0	Clf 1	Clf 2	Clf 3
1	-0.148	1.716	0.715	0.206
2	0.447	-0.780	0.171	0.967
3	0.474	-0.335	0.011	0.469
4	0.305	-0.436	0.171	0.312
5	-1.235	-5.378	-3.124	-2.365

Table 4: Weights of each of the 4 classifiers, where row i is weights of the i-th feature, column j is the weights of the j-th classifier.

Using tables 3 and 4, we observe that the longer a student spends learning online, the more likely they are to change their plans for a variety of reasons. And, students with more computer access and higher income are more likely to maintain their plans.

For a deeper analysis of these results, we use the line graphs in Figure 2 below, which plot probabilities based on a general categorization of student financial backgrounds and resources. From these, we see that as students of different socio-economic backgrounds spend more time learning virtually, low-income and under-resourced students face a significantly greater risk of being discouraged from post-secondary education.

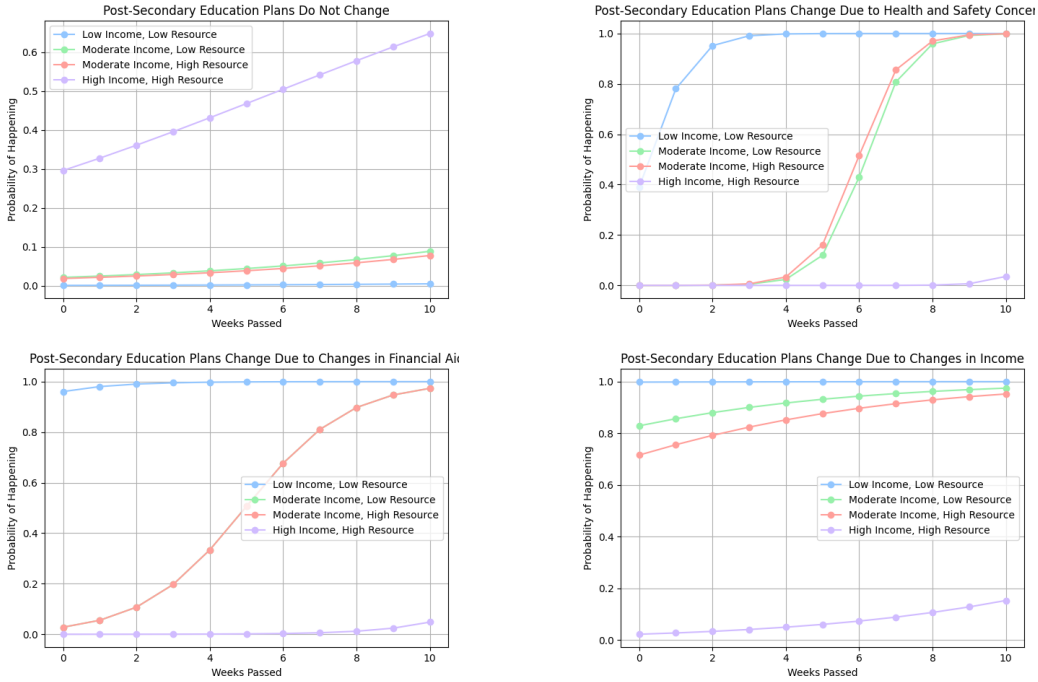

Figure 2: These demonstrate the probability that students with different financial backgrounds and resources are affected by virtual learning as time passes. Specifically, these plot how likely a student: maintains education plans (top-left), changes due to health concerns (top-right), change due to finaid (bottom-left), change due to income changes (bottom-right)

4. Conclusion

From this, we conclude that it is essential that we, as a society, prioritize reopening of schools quickly or work to improve the quality and distribution of teaching resources for distance learning. Otherwise, there exists a tremendous risk that previously disadvantaged students will become discouraged from pursuing higher education. This would perpetuate the pre-existing education gap in our country today and counteract years of efforts made by nonprofit and government organizations to improve education equity. Most significantly, the education gap will be detrimental to those with less education, as past studies have found educational attainment to be a strong indicator of economic prosperity.

As a future step, it would be interesting to collect more data across a longer period of time, to see the effects of virtual learning on the probability that educational outcomes change at a larger scale.

Appendix A: Survey Questions Used from HouseHold Pulse Survey

https://www.census.gov/programs-surveys/household-pulse-survey/technical-documentation.html

1. RRACE: What is your race?
2. EEDUC: What is the highest degree or level of school you have completed?
3. THHLD_NUMPER: How many total people – adults and children – currently live in your household, including yourself?
4. WHYCHNGD5: In the last 7 days, did your household spending change due to loss of income?
5. WHYCHNGD7: In the last 7 days, did your household spending change due to concerns about being laid off or having hours reduced?
6. SPNDSRC5: In the last 7 days, did your household use unemployment insurance (UI) benefit payments for spending needs?
7. SPNDSRC6: In the last 7 days, did your household use stimulus (economic impact) payments for spending needs?
8. TEACH1: Has the pandemic caused your school to cancel classes normally taught in person?
9. TEACH2: Has the pandemic caused classes normally taught in person to move to a distance-learning format using online resources?
10. TEACH3: Has the pandemic caused classes to move to a distance-learning format using paper materials sent home?
11. COMPAVAIL: How often is a computer or other digital device available for educational purposes?
12. COMP1: Is the computer or other digital device provided by the your school or school district to use outside of school?
13. COMP2: Is the computer or other digital device your own?
14. COMP3: Is the computer or other digital device provided by another source?
15. INTRNTAVAIL: How often is the internet available for educational purposes?
16. INTRNT1: Are internet services paid for by your school or school district?
17. INTRNT2: Are internet services paid for by someone in the household or family?
18. INTRNT3: Are internet services paid for by another source?
19. SCHLHRS: How much virtual contact did you have with teachers in the last 7 days?
20. TCH_HRS: How many hours did you spend on all teaching activities with teachers in the last 7 days?
21. INCOME: In 2019 what was your total household income before taxes?

Appendix B: main.py

```
import os
from load_data import get_features_labels
import utils as ut

DATASET_DIR = os.path.join(os.path.dirname(__file__), 'datasets')
SAVE_CLFS = True # if True, will save the trained classifier weights to the file 'clfs.txt'
TRAIN_SUBSET_FEATS = True # if True, the classifier will on train using features [0, 4, 11,

if __name__ == '__main__':
    # train and test the classifier
    train_features, train_labels = get_features_labels(DATASET_DIR, 'training.txt')
    test_features, test_labels = get_features_labels(DATASET_DIR, 'testing.txt')

    if TRAIN_SUBSET_FEATS:
        subset_features = [0, 4, 11, 15, 21]
        train_features = train_features[:, subset_features]
        test_features = test_features[:, subset_features]

    # train a logreg classifier for each target label
    clfs = ut.train_clfs(train_features, train_labels, DATASET_DIR, SAVE_CLFS)

    # predict labels of samples in test set and check accuracy of classifiers
    pred_labels = ut.predict(clfs, test_features)[1]
    accuracy = ut.clf_accuracy(pred_labels, test_labels, clfs)
    print(accuracy)
```

Appendix C: logistic_regression

```
import numpy as np

class LogisticRegression:

    def __init__(self, learning_rate, max_steps, weights=None,
                 norm_min=None, norm_max=None):
        self.learning_rate = learning_rate
        self.max_steps = max_steps
        self.norm_min = norm_min
        self.norm_max = norm_max
        self.weights = weights

    def fit(self, train_features, train_labels):
        # Prep train_features array
        train_features = np.insert(train_features, 0, 1, axis=1)
```

```
        self.norm_min, self.norm_max = normalization_constants(train_features)
        train_features = normalize(train_features, self.norm_min, self.norm_max)
        train_features.setflags(write=False)

        # Initialize theta
        theta = np.zeros(train_features.shape[1])

        # Gradient ascent
        for i in range(self.max_steps):
            gradient = train_features.transpose() @
                       (train_labels.reshape(train_labels.size, 1)
                       - sigmoid(train_features @ theta.reshape(theta.size, 1)))[:, 0]
            theta += self.learning_rate * gradient
            if i % 2000 == 0:
                print('Progress: ', i * 100 / self.max_steps, '%')
        self.weights = theta
        print('------------Done------------')

    def predict(self, test_features):
        test_features = np.insert(test_features, 0, 1, axis=1) # add bias term
        test_features = normalize(test_features, self.norm_min, self.norm_max)
        test_features.setflags(write=False) # make immutable

        preds = sigmoid(test_features @ self.weights)
        return preds

# From the provided PSET 6 Code
def sigmoid(vec):
    positive_mask = vec >= 0
    negative_mask = vec < 0
    exp = np.zeros_like(vec, dtype=np.float64)
    exp[positive_mask] = np.exp(-vec[positive_mask])
    exp[negative_mask] = np.exp(vec[negative_mask])
    top = np.ones_like(vec, dtype=np.float64)
    top[negative_mask] = exp[negative_mask]
    return top / (1 + exp)

# min-max normalization
def normalize(data, min, max):
    min_mat = np.repeat(min, data.shape[0], axis=1).transpose()
    max_mat = np.repeat(max, data.shape[0], axis=1).transpose()
    denom_mat = max_mat - min_mat
    denom_mat = np.where(denom_mat == 0, 1, denom_mat)
    normed = np.divide((data - min_mat), denom_mat)
    return normed
```

```
def normalization_constants(data):
    min = np.reshape(data.min(axis=0), (data.shape[1], 1))
    max = np.reshape(data.max(axis=0), (data.shape[1], 1))
    return min, max
```

Appendix D: utils.py

```
from logistic_regression import LogisticRegression
import numpy as np
import os

def train_clfs(train_features, train_labels, dataset_dir='', save_clfs=False):
    """Train a classifier for each class"""
    print('Starting training classifiers')
    clfs = {i: (fitting(train_features, train_labels[:, i]))
           for i in range(train_labels.shape[1])}
    print('Done training classifiers')
    if save_clfs and dataset_dir != '':
        print('Saving all classifiers to clfs.txt')
        clf_weights = np.vstack((clfs.get(0).weights, clfs.get(1).weights,
                      clfs.get(2).weights, clfs.get(3).weights))
        norm_min = clfs.get(0).norm_min
        norm_max = clfs.get(0).norm_max
        np.savetxt(dataset_dir+'/clfs.txt', clf_weights, fmt='%f', delimiter=',')
        np.savetxt(dataset_dir+'/norm_min.txt', norm_min, fmt='%f', delimiter=',')
        np.savetxt(dataset_dir+'/norm_max.txt', norm_max, fmt='%f', delimiter=',')
    return clfs

def fitting(train_features, train_labels):
    clf = LogisticRegression(learning_rate=0.0001, max_steps=10000)
    clf.fit(train_features, train_labels)
    return clf

def predict(clfs, test_features):
    pred_probs = np.empty((test_features.shape[0], len(clfs)))
    # for all samples, make predictions for each label (col)
    for i in range(len(clfs)):
        clf = clfs.get(i)
        pred_probs[:, i] = clf.predict(test_features)
    pred_labels = np.around(pred_probs)
    return pred_probs, pred_labels

def clf_accuracy(pred_labels, test_labels, clfs):
    """return prediction accuracy for each classifier"""
```

```
num_samples, num_classes = test_labels.shape
accuracy = [np.sum(np.equal(pred_labels[:, i], test_labels[:, i])) / num_samples
            for i in range(num_classes)]
return accuracy
```