# Analysis of the Sources of Financial Burden on Students Acquiring Upper Education

## Machine Learning Engineer Nanodegree

Corey Bofill

October 27th, 2016

# I. Definition

### Project Overview

Job-seekers today are facing increased demand for higher education for jobs that even recently were lower level. At the same time students are graduating with more debt than their peers were just a few decades ago. The cost of attending college is increasing at a much faster rate than general inflation.[1] Many graduates are left not only with substantial debt, averaging $37,172 for graduates of 2016,[2] but also are faced with a job market that, while starting to see improvement this year, is competitive and often over-saturated. Roughly half of recent college graduates are underemployed, meaning they are in a job where more than half of workers with that job don't think a degree is required.[3] At the same time the overwhelming majority of new jobs go to college graduates.[4] With such high demand for degrees, lowering the financial burden of attending college is crucial to ensuring the future financial success of young adults.

The College Scorecard (CSC) is a dataset that provides prospective students useful information on colleges. It consists of federal data from undergraduate degree-granting institutions of higher education in the US, including costs and financial aid awards for students, and incomes and loan repayment rates for graduates, amongst others. Data are currently available from 1996 to 2013.[5] The IPEDS Analytics: Delta Cost Project Database is a longitudinal dataset derived from the Integrated Postsecondary Education Data System (IPEDS) data with a focus on revenues and expenditures. Data are available from 1987 to 2012.[6] The CSC dataset draws from the IPEDS and includes the IPEDS Unit ID for each institution, making the two datasets compatible.

### Problem Statement

This project constructs a model of degree granting institutions. This model shows the relationship between the features of an institution and the cost/benefit to that institution's graduates. I then use this model to analyze and determine which features have the greatest impact.

I use data on student debt and post-graduation earnings from the CSC dataset combined with data on degree awards from the IPEDS dataset to create a single metric representing overall financial burden. I then use data on institution revenue, expenses, financial aid, scholarship spending, faculty, and graduation rates from the IPEDS dataset to construct a supervised random forest model. This model maps the institutional financial data to financial burden. I then use the resulting model to analyze feature importance and demonstrate which institutional features are most relevant to student burden.

### Metrics

To measure feature importance I apply a variable permutation technique which removes the predictive contribution of one feature within the model, and then measure the change in the mean squared error of the model’s predictions. Mean squared error (MSE) is the average of the squared difference between a model’s predictions and the true values of what is being predicted. It provides a straightforward measure of the model’s accuracy, which is useful because I am concerned less with the overall success of the model as I am with the effects of individual features on the model’s predictive power.

## II. Analysis

### Data Exploration

The data is drawn from the IPEDS dataset and the CSC dataset. Both include a ‘UnitID’ which is a unique identifier for an institution defined by IPEDS. Features from the IPEDS dataset were chosen based on categories defined within the data, summarized below, along with a count of the features in each.

| **Category** | **Description** | **Feature Count** |
|---|---|---|
| Characteristics | Categorical variables that detail the type of institution | 9 |
| Revenue | Money the institution takes in | 46 |
| Scholarships and Fellowships | Money the institution distributes as student aid | 12 |
| Expenditures | Money spent by the institution | 109 |
| Assets | The value of assets owned or controlled by the institution | 9 |
| Faculty | Profile of persons employed by the institution | 53 |
| Graduation Rates | How many and how quickly students are graduating | 9 |
| Enrollment | Profile of students attending institution | 47 |
| **Total** | | **294** |

These are the features used as input data for the model. The features of the Characteristics type are categorical, which means the value of the feature refers to a category the institution belongs to. For example, the 'control' feature has three categories: 1 – Public institution, 2 – Private nonprofit institution, 3 – Private for-profit institution.

For the output I create a measurement of the cost to a student for acquiring a degree at a given institution. The features I use to do this are:

1) GRAD_DEBT_MDN - The median debt upon entering repayment for students who graduated. From CSC.

2) gt_25k_p[*y*] - The percentage of students making over $25,000 a year at *y* years after enrollment. From CSC.

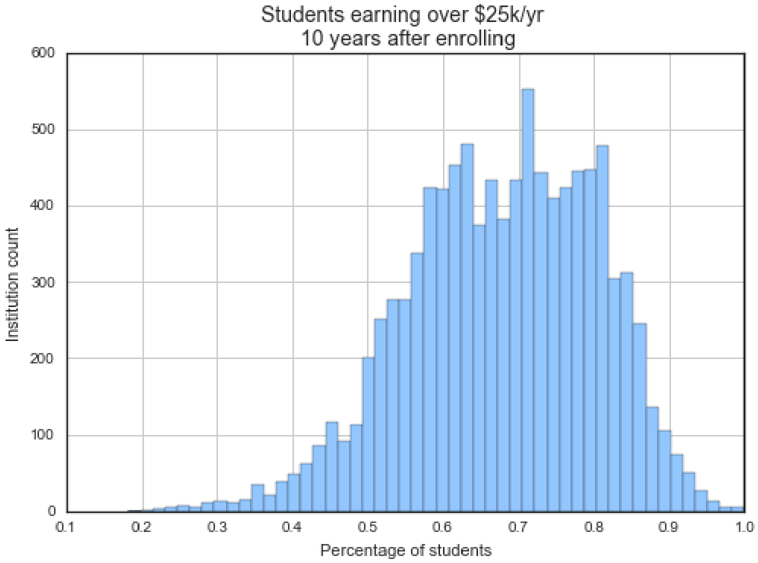
3) [*t*]_deg_share_of_tot_deg – The percentage of degrees awarded of type t. Types include associates, bachelors, masters, doctorates, and first professional degrees. From IPEDS.

The first feature measures the continued financial burden a student carries after leaving an institution. The second is a measure of the value of acquiring a degree – $25k is the median salary of a person 25-34 years old with only a high school degree, so the feature describes the proportion of graduates making more money having acquired a degree compared to had they not. Unfortunately, the reports of this feature are staggered by year so there is little information on how it changes over time, but it makes for a good baseline of the value imparted by an institution. The third measures the proportions of degrees awarded by an institution and can be used to estimate the average time to graduate from that institution, as each type of degree requires a specific minimum number of years of education.

These two sets of features are drawn from their respective datasets and combined into one file, using the UnitID and the academic year to index entries. The resultant dataset contains 52840 entries and is provided in the GitHub repo for this project, along with the code that constructs it. Statistical analysis information of the first five features is shown below:

| feature | acadinststud01 | acadinstsupp01 | acadsupp01 | acadsupp01_fasb | acadsupp02 |
|---|---|---|---|---|---|
| **count** | 6.339400e+04 | 6.042200e+04 | 4.125300e+04 | 2.465900e+04 | 3.661900e+04 |
| **mean** | 1.904906e+07 | 1.360917e+07 | 8.717911e+06 | 9.382099e+06 | 4.761678e+06 |
| **std** | 5.228226e+07 | 4.155264e+07 | 2.722640e+07 | 3.311490e+07 | 1.531347e+07 |
| **min** | 0.000000e+00 | -7.218947e+06 | 2.182707e+01 | 1.000000e+02 | 3.000000e+00 |
| **25%** | 1.852889e+06 | 1.202415e+06 | 8.293812e+05 | 7.907355e+05 | 4.731095e+05 |
| **50%** | 6.177273e+06 | 4.072854e+06 | 2.248671e+06 | 2.198715e+06 | 1.217448e+06 |
| **75%** | 1.634433e+07 | 1.076039e+07 | 6.179581e+06 | 6.070607e+06 | 3.285189e+06 |
| **max** | 2.055708e+09 | 1.822956e+09 | 7.297660e+08 | 7.765980e+08 | 5.134040e+08 |

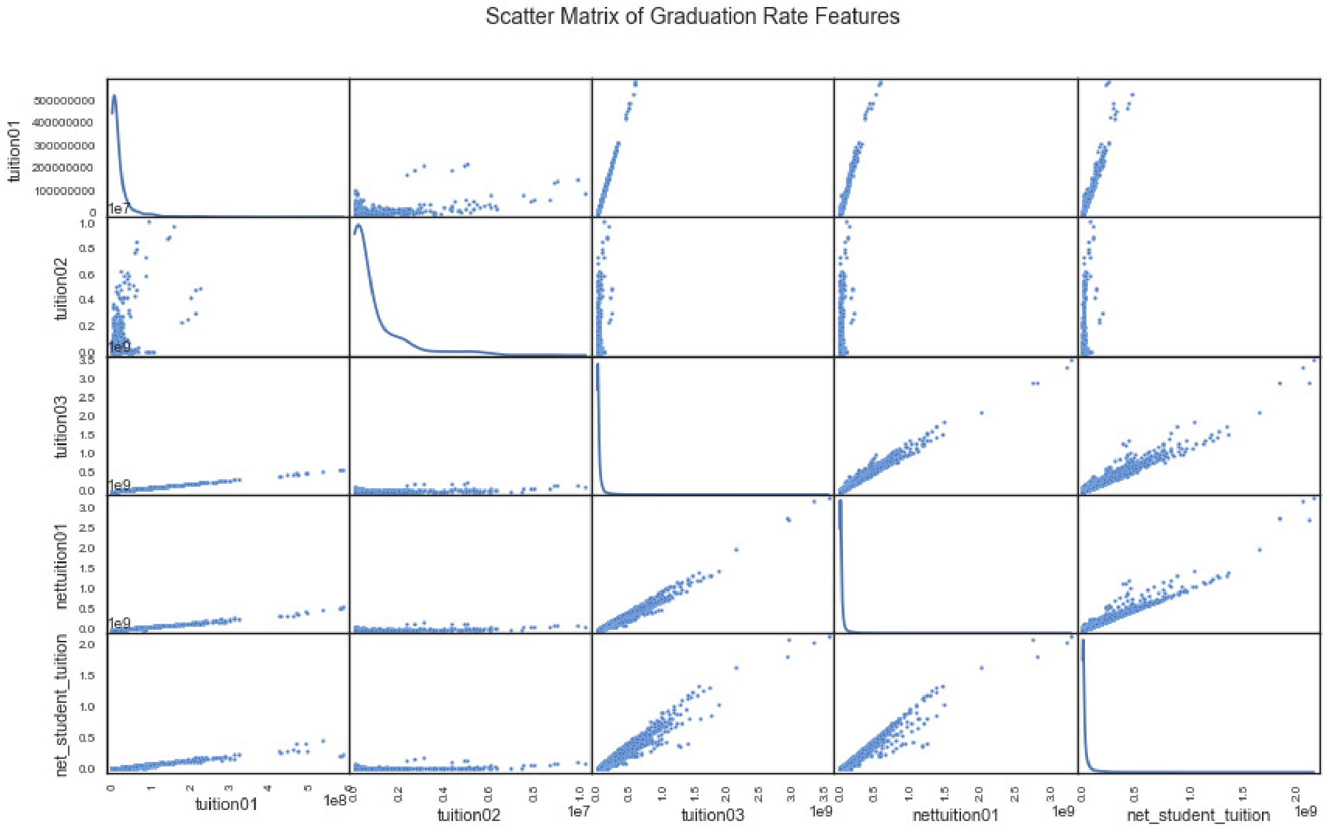
The count feature shows that a significant amount of data is missing. None of the five features above have data for every entry and acadsupp01_fasb has data for roughly a third of the entries. Some of these are data that were not been reported, but others were deliberately withheld for privacy purposes. Additionally acadinstsupp01 has a minimum value of -7.2e6. This feature is created by subtracting studserv01, expenditures on student services, from acadinststud01, expenditures on academic and institutional support and student services. Since acadinststud01 is supposed to include student services a negative result should not be possible. This may be a clerical error, or caused by missing data in one feature but not the other, but regardless of the source the result is that the entry is not reliable.



## Exploratory Visualization

To further understand the data, a histogram is shown of one of the output features: the percentage of graduates making over $25k per year, 10 years after enrollment. We can see that for most institutions about 60-80% of graduates meet the threshold, i.e. have a higher salary than the typical worker without a degree.

Something else to note about the data is that many of the features are related to or computed from one another. As an example I've taken the portion of data on graduation rates and constructed a scatter matrix below.



Scatter matrix showing the relationships of five tuition features. The clear linear shapes that occur are evidence of correlation between the features.

Correlations between features can have many undesired consequences if not accounted for. Many algorithms and techniques are sensitive to structures like these within the input space.

## Algorithms and Techniques

For the model I use a Random Forest Regressor (RFR), which makes predictions using the aggregate estimations of an ensemble of decision trees. Decision trees on their own are prone to overfitting, i.e. predicting based too heavily on the training data and generalizing to new data poorly. Random Forest mitigates this issue by aggregating the predictions of many trees built on randomly determined subsets of the dataset. I used the scikit-learn implementation, which builds individual trees using an optimized version of the CART algorithm.[7] Random forests are resilient to the form the data takes, such as scale or normality, which makes it a good candidate for a dataset with features of varied types such as the one I'm using. Most of the parameters involved in individual tree construction are left at the default values; however, there are a few which are especially pertinent to my goal:

| **Parameter** | **Description** | **Default** |
|---|---|---|
| n_estimators | The number of individual trees in the ensemble to contribute to the aggregated prediction. Generally more is better. | 10 |
| max_features | The number of features to consider when looking for the best split during construction of a decision tree. Using a subset of the total feature space reduces variance in the model, helping counteract the tendency of decision trees to overfit. | 'auto'<br>(n_features) |
| bootstrap | Whether or not to re-use samples when constructing decision trees. Typically this improves model performance by reducing variance. | True |

I use two standard techniques are to improve the model: data withholding and ideal parameter grid search with cross validation. The former withholds a portion of the data for testing purposes, helping to avoid overfitting to the training data. The latter fits the model using all combinations of potential parameter settings to find the ideal combination. These are implemented using scikit-learn's train_test_split[8] and GridSearchCV[9], respectively.

To measure feature importance I use the permutation method described by Strobl (2009).[10] Rather than scikit-learn's feature_relevance_ attribute, which measures importance by averaging the number of samples affected by a feature across the ensemble, each feature is measured by the following steps:

A) Measure the predictive power of the model

B) Permute (scramble) the feature across all samples, disassociating the feature's information contribution to the prediction

C) Measure the predictive power of the model on the permuted data

The difference between the measurements in (A) and (C) is the contribution of the feature to the predictive power of the model. This method has significantly less bias than the implementation's default method, provided there is no use of bootstrap sampling.[11]

### Benchmark

I use the guidelines for permutation variable relevance described by Strobl et. al. (2009).[10]

> All variables whose importance is negative, zero or has a small positive value that lies in the same range as the negative values, can be excluded from further exploration. The rationale for this rule of thumb is that the importance of irrelevant variables varies randomly around zero. Therefore positive variation of an amplitude comparable to that of negative variation does not indicate an informative predictor variable, while positive values that exceed this range may indicate that a predictor variable is informative.

## III. Methodology

### Data Preprocessing

#### Remove Erroneous Entries

The data contains entries that fall outside the possible range of values for a feature, such as the example of a negative value entered for a non-negative feature described during data exploration. Without more information on the cause of the error these data cannot be interpreted, and I set such entries to ‘nan’ to match other missing data.

#### Create Dummy Variables

The values of categorical features are often arbitrary and can misinform the model if treated as numeric. Rather than include them directly I split them into as many features as there are categories, with each new ‘dummy’ variable a binary representation of each category. For example the feature ‘hbcu’, representing whether an institution is a Historically Black College or University, becomes the features ‘hbcu_1.0’ and ‘hbcu_2.0’, each of which may be 1 or 0. An institution that had a 1 in ‘hbcu’ would then have a 1 in ‘hbcu_1.0’ and a 0 in ‘hbcu_2.0’. This turns the 9 categorical features into 26 binary dummy features, bringing the feature total to 311.

## Implementation

### Construct Cost Metric

I combine the cost variables into a single metric, defined as

<burden in years> = <time to reach wage threshold> + <debt burden>

‘Time to reach wage threshold’ measures the time starting from enrollment for a graduate to be making more money than had they not attended college. This is done in two steps:

1) Estimate the average time a student takes to graduate $y$ using the proportion of each type of degree awarded at an institution with the minimum required years for that degree type:

$$y = \Sigma\ (\text{degree\_share} * \text{minimum\_required\_years})$$

2) Estimate the average time to reach the threshold income $Y$ to be the average graduation time $y$. Iterate through the threshold measurements, updating the estimate with the additional time passed proportional to graduates that remain below the threshold.

$$\text{p\_above\_threshold} * Y + \text{p\_below\_threshold} * \text{years\_since\_enrollment} = \hat{Y}$$

Where $\hat{Y}$ is the new estimate for the mean time to reach the threshold. This attempts to measure the burden of acquiring a degree at an institution while minimizing the effects of student’s individual characteristics. To this end the debt burden is comprised of the median debt upon graduation divided by the threshold income, representing the burden in years of degree-less labor.

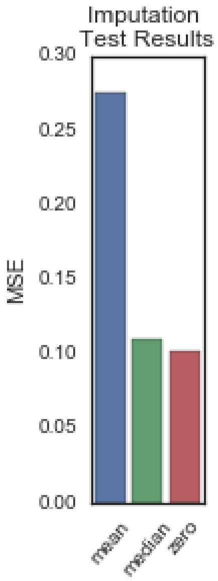
```
# Returns a pandas series representing the estimated mean cost in years to reach
# the threshold income
def calculate_mean_year_cost(df):
    mean_year_cost = pd.Series(index=df.index)
    for unit in df.index.levels[0]:
        # List of tuples of percentage values and the year they correspond to
        threshold_means = [(df.loc[unit]['gt_25k_p6'].mean(),6),
                           (df.loc[unit]['gt_25k_p7'].mean(),7),
                           (df.loc[unit]['gt_25k_p8'].mean(),8),
                           (df.loc[unit]['gt_25k_p9'].mean(),9),
                           (df.loc[unit]['gt_25k_p10'].mean(),10)]
        # Calculate the average amount of time to graduate using the minimum
        # required time to receive each degree type
        mean_years_to_graduate = np.nansum(
                              [df.loc[unit]['assoc_deg_share_of_tot_deg'].mean()*2,
                               df.loc[unit]['bach_deg_share_of_tot_deg'].mean()*4,
                               df.loc[unit]['grad_deg_share_of_tot_deg'].mean()*5,
                               df.loc[unit]['doc_deg_share_of_tot_deg'].mean()*8,
                               df.loc[unit]['prof_deg_share_of_tot_deg'].mean()*6])

        # For each year update the mean cost based on how many students have yet to
        # reach the threshold
        cost_estimate = mean_years_to_graduate
        for t in threshold_means:
            p_above_threshold = t[0]
            p_below_threshold = 1 - t[0]
            years_since_enrollment = t[1]
            cost_estimate = (p_above_threshold*cost_estimate +
                             p_below_threshold*years_since_enrollment)

        mean_year_cost.loc[unit] = cost_estimate

    return mean_year_cost
```

### Sequester Testing Data

Since the model does not bootstrap samples, out-of-bag error estimates for internal cross validation are not available. Instead I sequester a subset of data to test the model on.

### Impute Missing Values

Imputation, aka inferring data, is necessary due to the inherent limitations of the modeling tools in handling incomplete data. To determine the best method for imputation I compare the results of a model built with data imputed by feature mean or median, and by zero. My tests show that imputing all values with zero results in the lowest MSE.



### Construct Model

As described above, I implement the Random Forest regressor using GridSearchCV.

```
# Initialize and fit model
rfr = RandomForestRegressor(bootstrap=False)
params = {'n_estimators':[50],'max_features':['sqrt','log2']}
scorer = make_scorer(mean_squared_error,greater_is_better=False)
```

```
gscv = GridSearchCV(rfr,params,scoring=scorer)
gscv.fit(X_train,y_train)

model = gscv.best_estimator_
```

### Variable Importance Permutation

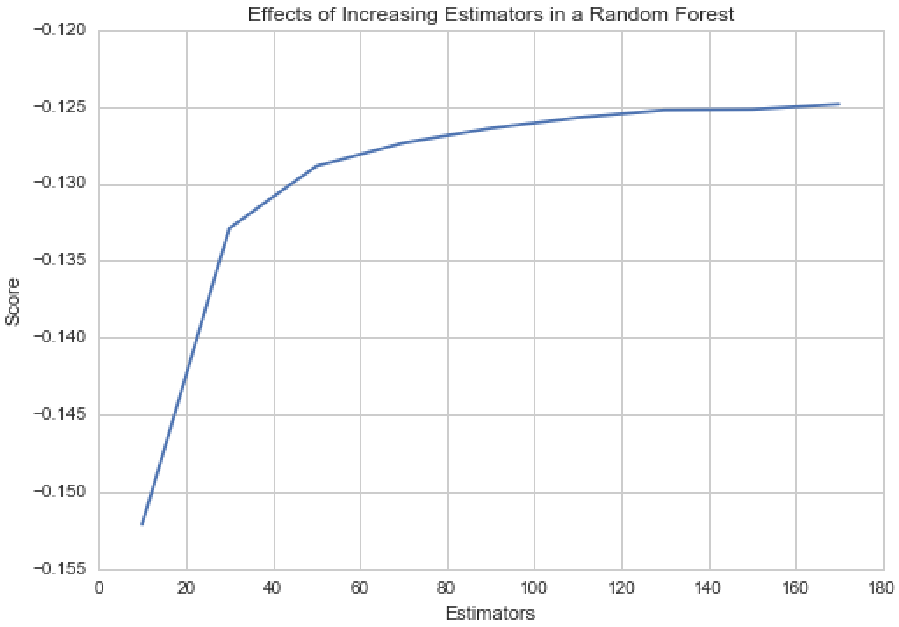
I implement variable importance permutation to create a list of the features ranked by their contribution to the model's MSE.[12]

```
# Take a model and testing data and return a tuple list of feature importance
def variable_permutation_score(model,test_input,test_output):
    scores = defaultdict(list)

    # Record the mean squared error for the unmodified test data
    mse = mean_squared_error(test_output, model.predict(test_input))

    # For each column, permute the data and record the change in test score
    for feature in test_input.columns:
        X_copy = test_input.copy()
        np.random.shuffle(X_copy[feature])
        permuted_mse = mean_squared_error(test_output, model.predict(X_copy))

        # Record scores as difference between permuted and original score
        scores[feature].append((permuted_mse - mse))

    return sorted([(round(np.mean(score), 4), feat) for feat, score in
scores.items()], reverse=True)
```

## Refinement

To refine the model I expand the parameter space of GridSearchCV. This is done in two steps to minimize computation time: first I test each parameter on a small range of values including the default to find if any require closer attention. This confirms that the model only benefits from tuning the parameters discussed above: n_estimators and max_features. Second I test the parameters of interest across a wider range of values to determine the optimal parameter set. Under most circumstances the performance of a Random Forest model will improve with an increase in the number of estimators; however, the diminishing returns of the improvement and the increase in computation time make an arbitrarily large ensemble impractical.

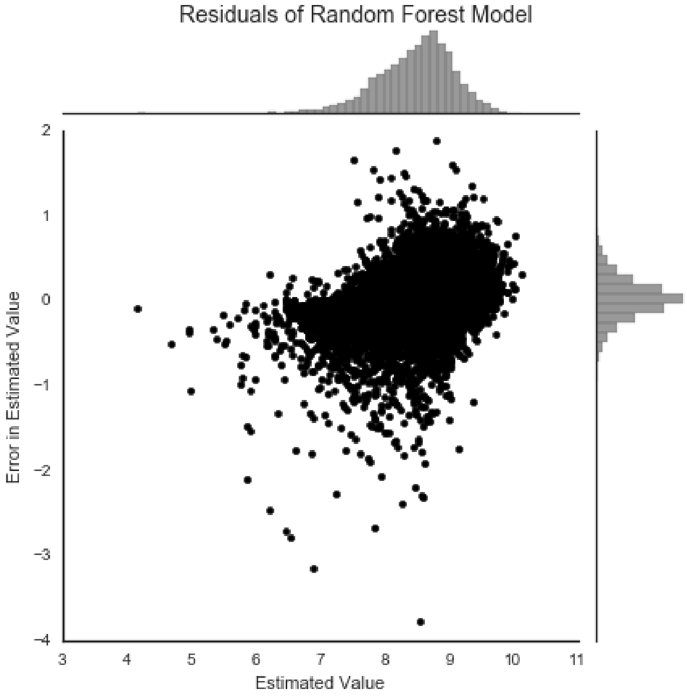
I measure model performance with a range of estimators and determine an optimal number of estimators to be 130. Similarly, setting max_features to the square root of the total number of features consistently shows the most gain in performance.

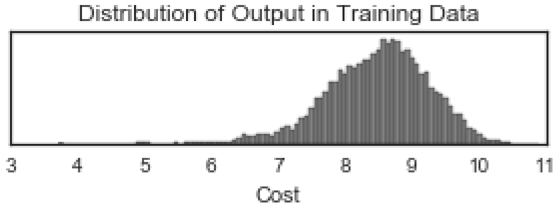


# IV. Results

## Model Evaluation and Validation

To evaluate the final model I impute the sequestered testing data using the method determined by the training data, then compute the mean squared error. The model achieves a MSE of 0.1021. While MSE is a useful metric for model performance due to it measuring both bias and variance, it can be difficult to interpret. The square root of the MSE, the root mean square error (RMSE), is in the same units as the output and estimates the standard deviation of the output. The RMSE of the model is 0.3195, which compares favorably to the mean of the output: 8.476.





To further evaluate the model I create a visualization of the model's residuals, the difference between the model's estimate and the true values. Ideally the residuals will have a uniform distribution, caused by the model explaining all internal structures of the output that are not random. The residuals of my model show a few notable characteristics. The first, the tail on the left, is explained by the distribution of the output data, shown adjacent. Second, the plot shows signs of heteroscedasticity, where the variability of the errors changes relative to the estimate. While the distribution around zero looks normal, as shown on the left of the plot, the model tends to overestimate more from 6-8.   This is typically caused by a problem with the feature space; the model is not accounting for some characteristic of the input. Since Random Forest models are not very sensitive to feature transformation, this is most likely due to some determining feature being absent from the dataset.

**Feature Importance**

I evaluate feature importance with the permutation technique describe above. The scores determined relevant by the benchmark set range from 0.0001 to 0.0283, reflecting an increase in the model's MSE from 0.1% to over 28%. This wide range is a good indicator of the success of the solution to distinguish features.

**Sensitivity Analysis**

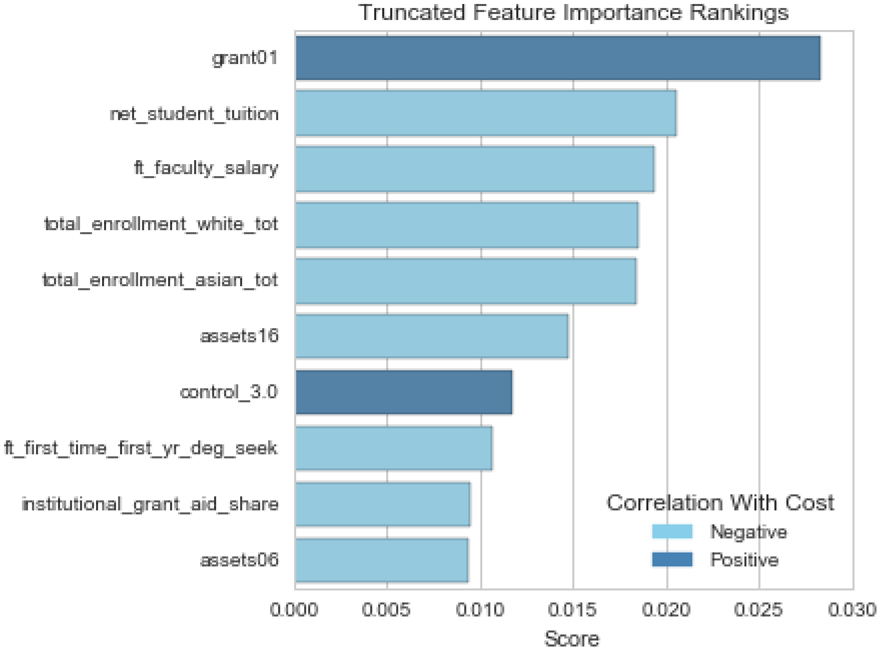
To validate the robustness of the model I alter the random state variable of the algorithms used to sequester the data and construct the model. To assure the test results are reliable it is critical to prevent information from the testing data being used in the construction of the model. By keeping the random state constant the training and testing subsets are always the same, and no information leak occurs. Once the model is finalized, the sensitivity of the model to the input space can be tested by altering the random state and thus the subsets of data with which the model is built and tested. This results in a model MSE of 0.1010, showing the model is resilient to perturbations in the data. The feature importance rankings were not perfectly stable – most fell by roughly 0.002-0.004 – however the change was fairly uniform with a few exceptions of features' scores increasing. This results in a change in the rankings, however the relative scores of each feature are somewhat stable.

**Justification**

The benchmark I established states that features with permutation scores that are negative, zero, or as close to zero as any negative scores do not influence the prediction. Of the 311 input features, 32 fell within this range. Thus 279 features contribute to the accuracy of the model. The scores of these features ranged across four orders of magnitude, and while the rankings of the feature's relevance are not stable, it does suit the goal of the project which is to highlight areas worth further investigation.

# V. Conclusion

## Free-Form Visualization



The variable importance rankings are a measure of how much predictions suffer if the information from that feature is lost while all others are kept the same. To help interpret the feature permutation scores I calculate the correlation between these features and the output. The scale of the correlation alone isn't helpful, as any information on the feature's place within the context of the input space is lost; however, the polarity of the correlation provides insight into why the feature is influential. I plot the top ten features colored by the polarity of their correlation with the output.

| Feature | Description from data dictionary |
|---|---|
| grant01 | Pell Grants disbursed or otherwise made available to recipients by the institution. |
| net_student_tuition | Net tuition revenue coming directly from students (not including Pell, Federal, State, and Local grants). |
| ft_faculty_salary | Average salary for full-time faculty |
| total_enrollment_white_tot | Total enrollment (White) |
| total_enrollment_asian_tot | Total enrollment (Asian) |
| assets16 | Long-term investments |
| control_3.0 | Private for-profit institution |
| ft_first_time_first_yr_deg_seek | Total number of full-time first-time degree/certificate-seeking undergraduates |
| institutional_grant_aid_share | Share of total financial aid from institutional grants |
| assets06 | Total assets |

While further research would be required to make definitive claims, a pattern is apparent from the highest ranking features. Pell Grants, grants given to students based on financial need, is the highest ranked feature yet it has a positive correlation with cost. This means that institutions that provide more Pell Grants end up imposing more of a financial burden on graduates. This is in contrast to the second ranked feature, net tuition revenue coming directly from students rather than grants, which is inversely correlated with cost. The more students are paying directly for their tuition, the less of a burden that occurs.

To interpret why the total enrollment of whites and asians ranks so high it is important to keep in mind that these rankings are of the independent effects of a feature – how much worse the prediction is for institutions where everything else is kept equal. These ranking paint a picture of privilege: the more students come from historically wealthy demographics and the less they rely on financial aid, then the less of a burden attending college will be. This may not be a surprise, but it drives home the point that the demand for college degrees in the workforce is disproportionate to the financial capacity of laborers to acquire them.

Another interesting note is that faculty salary decreases cost, which suggests that higher paid faculty are beneficial to students.

## Reflection

This project is made of three major portions: constructing the dataset, implementing a model, and implementing a feature importance measure. I chose the problem upon finding the CSC has data on graduate income, but included the IPEDS dataset because I needed information about institutions that the CSC dataset didn’t have on its own. The process of creating a single dataset was new to me, and ended up being very involved. This was especially true dealing with the substantial amount of missing data. Designing the cost metric was also a challenge, as the project revolves around its success in measuring financial burden.

Implementing the model was the most straightforward part of this project. I initially planned to use linear regression, but after exploring the data and running into issues scaling and transforming such varied features I decided to use a random forest to simplify the process. Having worked with the model before I had no difficulties implementing and optimizing.

Finally, implementing the feature importance metric took a great deal of work. While researching the problem I discovered that the default feature importance metric for the sklearn Random Forest is heavily biased and unreliable. Though I utilized a couple methods of minimizing the bias, the amount of work to implement all of the improvements that have been researched quickly became obviously beyond the scope of the project.

These three portions of crafting my solution to this problem happened simultaneously, with considerations for feature importance bias being taken early on and inclusions to the dataset being made very late in the project. I feel that when starting this project I was unaware of a number of problems I'd have to solve, and despite taking much longer than anticipated I found it a very educational experience. Besides the technical issues I discuss in the improvements sections, I think my solution works well for this type of problem.

## Improvement

There are two main improvements that this project would benefit from. First, the construction of the dataset suffered from some arbitrary feature selection. Including the entire IPEDS dataset and performing a thorough feature selection analysis would likely reduce the heteroscedasticity of the residuals. The analysis of feature importance shows that escaping the effects of graduate's financial demographic is more complicated than I thought, and a more informative analysis would likely come from updating either the input data or the cost metric with demographic information.

Second, there are numerous improvements that can be made to the bias of the feature selection method. This bias, as explained by Strobl et al. (2007)[11], is inherent in the CART algorithm used by sklearn's Random Forest implementation. Methods of unbiased recursive partitioning have been described by Hothorn et al (2006)[13], but are not available in python. A large part of this bias is caused by correlations between features, which can be accounted for using a conditional variable importance measure as described by Strobl et al. (2008)[14] but is also not available in python. I attempted to reduce this bias by implementing principal component analysis using a correlation matrix, however the resulting model was significantly less accurate.

## References

[1]:  https://en.wikipedia.org/wiki/College_tuition_in_the_United_States#Growth_of_college_costs

[2] https://studentloanhero.com/student-loan-debt-statistics/

[3] http://www.usnews.com/news/articles/2016-04-08/job-prospects-revised-down-for-class-of-2016

[4] http://money.cnn.com/2016/06/30/news/economy/college-grads-jobs/

[5] https://catalog.data.gov/dataset/college-scorecard

[6] http://nces.ed.gov/ipeds/deltacostproject/

[7] http://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart

[8] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[9] http://scikit-learn.org/0.17/modules/generated/sklearn.grid_search.GridSearchCV.html

[10] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2927982/  (Strobl et al, improve this)

[11] https://link.springer.com/article/10.1186%2F1471-2105-8-25 (other Strobl, also improve)

[12] http://blog.datadive.net/selecting-good-features-part-iii-random-forests/

[13] https://eeecon.uibk.ac.at/~zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf

[14] http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-307