Inside College Scorecard: Does College Name Mean Everything For Future Income?

Charles Chen, Issac Baron, Hong Fan, Jun Ouyan, Steven Gong, Richard Murphy, Shawn Xiang, Niall Smith, Suba Rajaram, Phoebe Wang, Zenging Zang, Boli Zhang

Department of Computer Science

University of California

1 Shields Ave, Davis, CA 95616

Abstract

We analyze potentially important non-name college features to determine whether the average income of a college graduate can be predicted. The data only includes the features of colleges and their average graduate income. This question is addressed by utilizing several machine learning methods: Linear Regression, Neural Networks, K-Nearest-Neighbors, Random Forest, and Support Vector Machines. The optimal results are achieved with Support Vector Machines with a classification rate average of 84%. Random Forests comes in second with a 70% accuracy rate with cross validation. All other methods fall short of a good prediction. Linear Regression and Neural Networks averaged a $10,000 difference in predicted against actual and K-Nearest-Neighbors achieved a 66% accuracy. The results of Support Vector Machines show that a college's features have the potential to predict the average income of a college graduate. Although it achieves good results, it has difficulty predicting medium incomes. More testing with optimal feature selection across the entire original data may be used to predict with better accuracy.

General Information

This project is an attempt to discover relationships between features of American universities such as admission rate, average SAT scores of admitted students, average family income of admitted students, etc. and earning rates of alumni ten years after enrollment. We used a set of data from the Department of Education that tracked information from thousands of secondary education institutions in the United States from 1996-2013. We only used the most recent values in each category for each college as our data set.

The project answers the following questions: can future earnings be predicted by features other than the college name? Do the admission rate, Average SAT score, percentage of majors, etc affect the average income of a college graduate? By using this data, we hope to determine if there is more to future income than just the college name. We have settled on five methods (Linear Regression, Neural Networks, K-Nearest-Neighbors, Random Forest and Support Vector Machines) to test the accuracy of this claim. Each method outlines the steps taken, results of the accuracy of the predictions, and a conclusion based on the results of each method.

Introduction

This paper analyzes potentially important features for determining the average future income of a college graduate. We used a set of data from the Department of Education that tracked information from thousands of secondary education institutions in the United States from 1996-2013. This paper focuses on four different algorithms to predict future income levels. The methods utilized in this report are Linear Regression, Neural Networks, K-Nearest-Neighbors, Random Forests, and Support Vector Machines. This paper provides insight into the claim that the average future income of a graduate can be predicted by non-name college features.

Methods

Data Cleaning: The initial data are given in form of a SQLite database with a couple .csv datasheets. However, we could not get access to features and results we want directly from those .csv datasheets. For example, the salary information datasheet does not have SAT information or university types in it. Also, the majority of the universities has some of their entries missing due to privacy issues and collecting difficulties. So we decide to use SQLite database to extract the information we want and clean it to suit the purpose of our goal. We take the database and created a view that selected all the features we wanted. Also, we get rid of all the "NULL"s and "PrivacySuppressed" entries in the view. After we export the view from the database, we get a csv table of which has multiple years' data. Then we move our draft datasheet into R, where we finish our cleaning process by isolating the most recent year's data from the rest for each entry and encoding all original non-numeric data. After we have a working dataset, we write a data description and hand our work to the rest of the team.

Linear Models: Our linear regression uses all 46 input features to make the 6 predictions regarding future income. Ordinary least squares and stochastic gradient descent are used to estimate these models.

For the purpose of interpretation, it would be helpful to have a linear model involving only a small subset of these features. Therefore, Lasso regression is then considered here to select variables.

In Lasso regression, the following quantity is minimized:

$$\frac{1}{n}\sum(Y_i - \beta_i X_{i1} - \ldots \beta_P X_{ip})^2 + 2\lambda\sum|\beta_j|$$

In the model, the minimum lambda is achieved at the value which gives lowest mean squared errors (using 10-fold cross-validation). For each of the 6 response variables, we keep features which have non-zero coefficients when fitting Lasso regressions. We then use mean squared errors in training dataset and test dataset to evaluate our models and make comparisons between OLS and Lasso.

Neural Networks: We create a neural network that would act as a predictor for this problem. Our ANN uses all 46 input features to make the 6 predictions regarding future income. The nodes in the final layer were linear functions. In order to find an optimal architecture we compare the testing errors of the ANN while increasing the number of nodes from 3 to 12 and the number of hidden layers from 1 to 3. We run this test with different random samplings for training. There isn't a consistent optimal error with these different training sets, but an architecture with 2 layers with 8 to 10 nodes each seemed to perform well. Using 2 layers and 10 nodes, our ANN has an average training error between 0.032 and 0.035 and an average testing error between 0.064 and 0.070.

When samples were fed into the neural network after it is trained the predictions were generally within $10,000 of the actual reported values. Thus this system could be used to give a rough estimate of earnings based on a school's characteristics. A weakness of the ANN for this problem is that it is difficult to analyze which of the input features affect expected income the most.

Random Forest: Due to the size of dataset, Random Forest is a good approach to determine what features, compare to other related features, are most correlated. Random Forest is an ensemble learning method for constructing a decision tree that corrects decision trees' habit of overfitting to their training set. The algorithm will apply the general technique of bootstrap aggregating or known as bagging. Suppose a training set has input $x = x_1, \ldots x_n$ and output(that is the expected earning of graduates) $y = y_1, \ldots y_n$, the bagging will repeat n times will give the following prediction based on all individual regression tree on x' and f is the result of prediction score,

$$f = \frac{1}{n}\sum_{n=1}^{n} f_b(x')$$

Random forest tree also split the dataset to allow "feature bagging". This precludes features over dominating the prediction score.

Random Forest will give a general overview of which feature relate the most to the output. However, it is important note that it does not tell us the relationship or correlation coefficient between the two. The algorithm can be computed using scikit-learn package in python.

Support Vector Machine: SVM is a supervised learning method. The goal of the support vector machine is to find the optimal separating hyperplane which maximizes the margin of the training data. It can be used to examine the relationship between various features, including demographic, major offered and academic achievement to the expected earning of a college graduate. The output, expected earning of an average student, is a continuous data, so the data has to be categorized into a number of classes. The output requires scaling and normalization, then it will be divided into 5 groups for multi-label classes (low, low-medium, medium, medium-high,

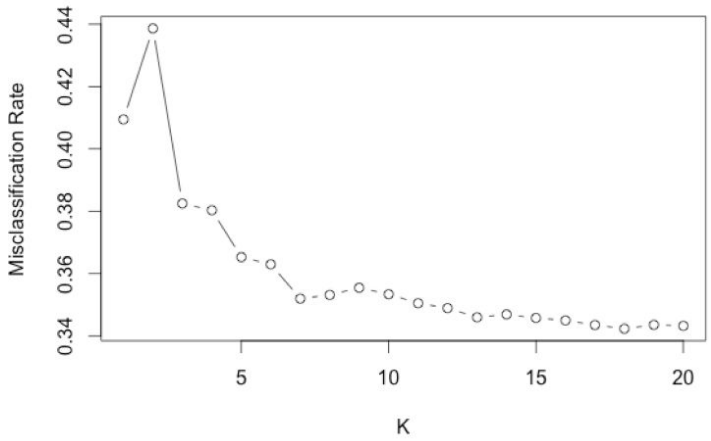
high). There is tradeoff in the number of groups to choose. As the number of groups increases, the prediction are more meaningful with more precise result, however, it will also be make SVM less efficient when there are too many classes. The algorithm is then applied to all features that are likely to be related by examining AUC value.

There are two approaches to multi-label SVM, one is constructing and combining several binary classifiers while the other is directly considering all data into one optimization formulation. One-against-all and directed acyclic graph method produce the most promising result. One-against-all is used in this project due to its supports in most library and relatively fast running time compared to more complex methods. In one-against-all method, the *m*th SVM is trained with all the example of mth class with positive labels and all other examples with negative labels. In addition, to identify the type of kernel to use, the algorithm will use trial and error for linear, polynomial and rbf.

In order to train SVM faster, we implement SMO(sequential minimal optimization) using the fact that $a_i\ y_i + a_j\ y_j$ = constant. If we change a_i, we have to do a corresponding change in a_j in order to maintain the relation that $a_i\ y_i + a_j\ y_j$ = constant. We first sequential select i, then we randomly select j . We continue to train until convergence. The definition of convergence here is that after 100 times continuous updates, there is nothing changes in a_i or a_j, or the change of a_i is less than 10^{-5}.

K-Nearest-Neighbors: We create a K-Nearest Neighbors Algorithm with 10 fold Cross Validation in R to classify levels of income. KNN is a non-parametric, supervised learning method which utilizes the distance matrix between observations to predict labels. We use the euclidean distance formula and use all the features to create our distance matrix (we found that lowering the features didn't help our accuracy). For our labels, we arrange income groups into Low, Medium, and High. Low Income is less than $37,500 a year, Medium is between $37,500 and $49,250 a year, and High is greater than $49,250 a year. These bin measures are selected based on a box plot of the income. We run our KNN procedure for each K from 1 to 20, 1000 times and average the misclassification rate for each K to find the K that has the lowest average Misclassification Rate.

Graph 1: Average Misclassification Rate vs K over 1000 iterations



This is the case when K is equal to 18. In general, KNN could not achieve a classification rate above 67% (or misclassification rate below 33%). This is one case with K=18, where we only classify 66% (1010 samples correct and 522 samples false) of the data correctly with KNN.

Table 1: Training and prediction result of KNN

Actual/Predicted	High	Medium	Low
High	221	148	14
Medium	93	599	77
Low	22	169	90

In our confusion matrix, we find that we have the hardest time predicting low values. We only classify 50% of Low values correctly, and of these, we misclassify a high amount of Low values as Medium.

Results

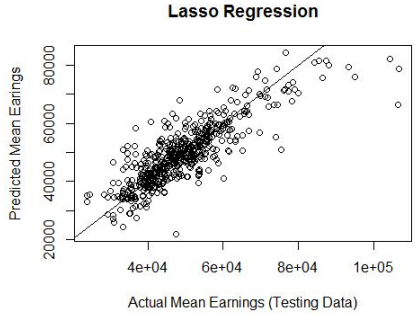
Linear Models: The following table compares OLS and Lasso methods for each response variable. All the variables are standardized before fitted into the model.

Table 2: Comparisons between OLS and Lasso for Each Linear Model

	Mean Earnings	Medium Earnings	10th Percentile of Earnings	25th Percentile of Earnings	75th Percentile of Earnings	Standard Deviation of Earnings
Test MSE (OLS)	0.345	0.830	1.198	0.713	0.279	0.455
Test MSE (Lasso)	0.238	0.292	0.375	0.372	0.252	0.412

The coefficients for the standardized regression using OLS method show that family income, percentage of students majoring in health, percentage of dependent students, average SAT scores, percentage of white students, percentage of students majoring in engineering are the six predictors which have greater impact on earning related variables.

Graph 2: Predicted Mean Earnings vs Actual Mean Earnings by Lasso Regression



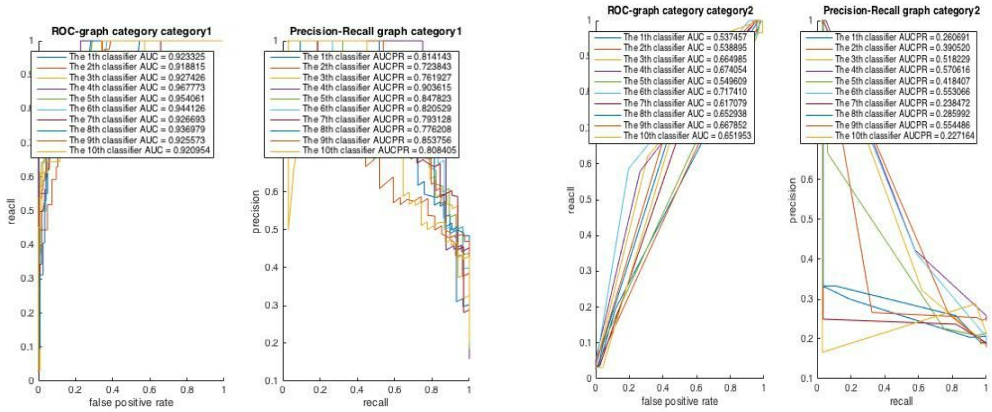
Neural Networks: We use a holdout method of setting aside a test set and a training set. We maximize our performance using 2 layers and 10 nodes, and our ANN achieved average training error between 0.032 and 0.035 and an average testing error between 0.064 and 0.070. The predictions are around $10,000 of the actual reported values. Although this system does not achieve very accurate results, this system could be used to give a rough estimate of earnings based on a school's characteristics. A weakness of the ANN for this problem is that it is difficult to analyze which of the input features affect expected income the most.

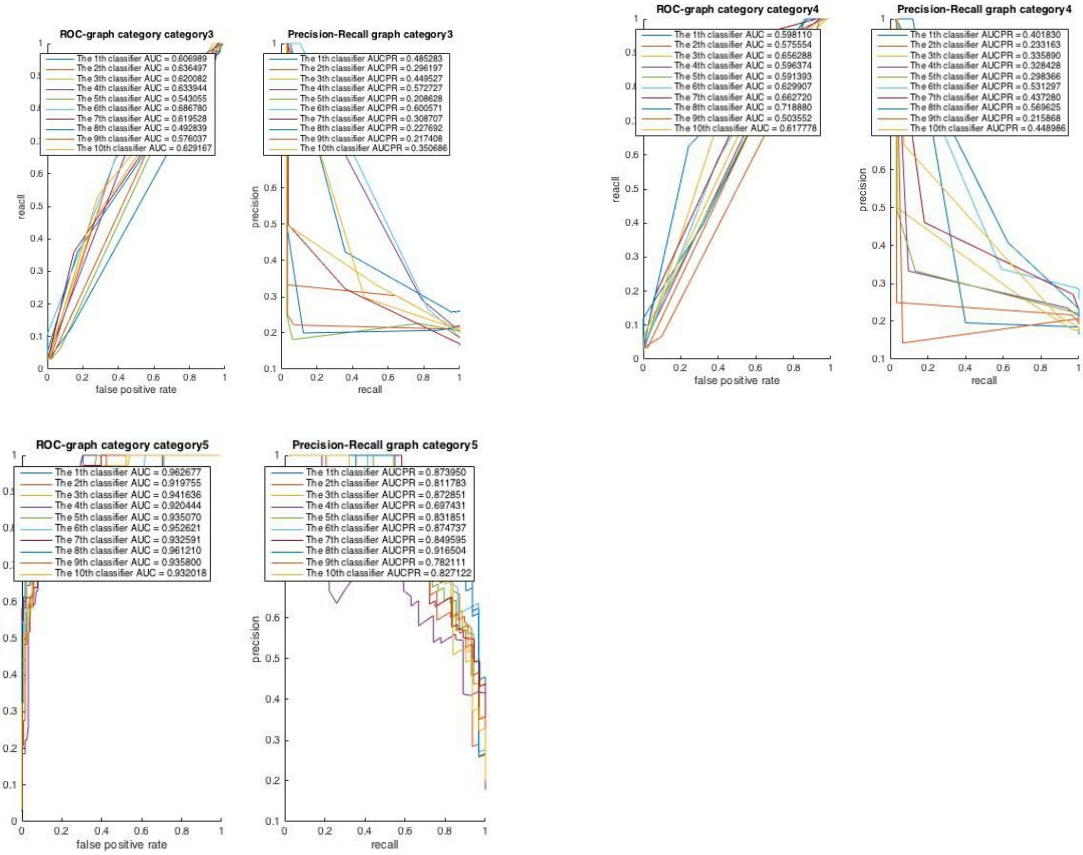
K-Nearest Neighbors: From our methods, we achieve the best lowest average misclassification rate when K was set to 18. The KNN algorithm could not achieve a classification rate above 67%. In general, the algorithm has a hard time predicting High and Low incomes which is likely due to the method in which we bin and label the actual average incomes. If our algorithm could classify the results with an accuracy rate of 75% or higher, it would be considered a success at predicting the average college graduate's income 10 years from now based on the features of the college he/she attended. Since we only achieve a maximum classification rate of 67%, this is not the case (KNN does not provide a good prediction with these features).

Random Forest: From the Random Forest, SAT average has a surprisingly high correlation percentage of 38.6%, following by median household income(9.1%), median family income (8.2%), admit rate (5.2%), Major in Engineering(3.9%), Major in Health (3.1%), gender (3.1%), birthright citizen (2.9%).
The prediction accuracy using 10-fold cross validation has an average of 70%.

Support Vector Machine
Using an improved version of SVM, it gives a 84.57% correct prediction overall in the case of binary classifier through random sampling and 10-fold cross validation. We use 5 models to to predict the label. We obtain 48.37% correct prediction for all 5 classes.
In particular it works extremely well on income below 20% or higher than 20%. However, it does not work very well on the middle range.

Graph 4-7: ROC and PR curve for 5-Class SVM





Discussion

Linear Models: Table 1 shows that Lasso has better prediction performance than ordinary least squares. Most of the predicted values for those six earning-related variables are within $10000. Our conclusion is Lasso regression only gives rough estimates of those earning-related variables and it is not advisable to use linear models to predict those earning variables. In terms of analyzing relationship between response variables and covariates, Lasso does give informative covariates non-zero coefficients. However, when a group of covariates are highly correlated, lasso tends to pick only one covariate from the group and there is no criterion for which one will be selected[1]. One the other hand, the Lasso also helped us to optimize the running time. The first time we run 46 features it took more than one hour to apply training using stochastic gradient descent. After using lasso to minimize the features, it only took half minute for training. It really increased the performance of computation.

Neural Networks: When samples are fetched to the neural network after it is trained, the predictions are generally within $10,000 of the actual reported values. Our conclusion is that Neural Networks can only give a rough estimate of the income but not a strongly accurate prediction.

K Nearest Neighbors: One of the potential reasons why we have misclassified the ends of our labels (High and Low) is likely due to the method in which we binned the labels to come up with our 3 categories. The Low, Mid, and High values are not distributed equally. Low and High

constitute about 50% of the labels together, while Medium takes the remaining 50%. Nevertheless, from the results, K Nearest Neighbors does not do a good job of predicting our desired labels as its classification rate falls below the 75% mark.

Random Forest: From the result, academic achievement, such as SAT and family income have significant (>8%) impact to one's expected earning. Major and demographic also have considerable impact (>3%). The random forest is a good predictor with 70% correctly with using 10-fold cross-validation.

Support Vector Machine: The overall correct classification for the entire SVM is not perfect for a continuous output. But after splitting the input into several group of features, results gave some interesting insights on what features relate the most with earning after graduation.

Table 3: Prediction accuracy using the result from the linear kernel for each group of features

Feature	Binary classification accuracy	Multi-class (5 classes) accuracy
Demographic	67%	34%
Family Income	74%	32%
Race	71%	30%
Academic achievement	76%	38%
Major	76%	40%

Table 3 summarizes the relationship between each separated groups of features. Major and academic achievement has the highest prediction accuracy and it is closely followed by family income, demographic and race.

Conclusion

The goal of this paper is to determine if non-name college features can be used to accurately predict an average college graduate's income. From Linear Regression and Neural Networks, the continuous predictions averaged $10000 from the expected results. When the majority of our incomes range from $25000 to $45000, this is a very large amount. The estimate does not perform well enough to strongly suggest that with the current features, average future college incomes can be predicted. Amongst the categorical predicting methods, Support Vector Machines stands out with the highest average accuracy of 84% in the case of binary classifier. It does extremely well for predicting the tails (bottom 20% and upper 20%) but really bad for predicting in between. Random Forests stands out with a second best accuracy rate of about 70% followed by K-Nearest Neighbor with 66%. This distribution of results for Support Vector Machines suggests that there are some key differences in features that help distinguish high and low incomes. The only result that strongly supports our goals is Support Vector Machines and Random Forests to an extent. As for what factors affects average earning after graduation, academic achievement (such as SAT) has the highest correlation, followed by family income, major and demographic. In conclusion, college names, or more precisely the academic achievement of the enrolling students, may truly be the most important feature in accurately measuring an average college graduate's income however from the results of Support Vector Machines, the features of a college can also tell an accurate story. Further testing with optimal

feature selection across the entire original data may yield even more accurate results toward predicting the average future income of a college graduate.

Reference

Balcázar, José (Jul 1997). "Computational Power of Neural Networks: A Kolmogorov Complexity Characterization". *Information Theory, IEEE Transactions*

Bach, Francis R *(2008). "Bolasso: model consistent lasso estimation through the bootstrap"*

Bailey, T., Jain, A. A note on distance-weighted k-nearest neighbor rules. IEEE Trans. Systems, Man, Cybernetics, Vol. 8, pp. 311-313, 1978.

Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software

Hsu-Wei, Chih, Chih-Jen Lin. "A Comparison of Methods for Multi-class Support Vector Machines." *A Comparison of Methods for Multi-class Support Vector Machines* (n.d.): Web.

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5

Li, Baoli. "An Improved K-Nearest Neighbor Algorithm for Text Categorization." *An Improved K-Nearest Neighbor Algorithm* (n.d.): n. pag. Web.

Marti A. Hearst, "Support Vector Machines", *IEEE Intelligent Systems*, vol.13, no. 4, pp. 18-28, July/August 1998, doi:10.1109/5254.708428

Ng, Andrew. *CS 229, Autumn 2009 The Simplified SMO Algorithm* (n.d.): n. pag. Web.

Platt, John C. (1998). "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines " Microsoft Research

Rokach, L.; Maimon, O. (2005). "Top-down induction of decision trees classifiers-a survey". *IEEE Transactions on Systems, Man, and Cybernetics*

Zou Hui, Hastie Trevor (2004). "Regularization and Variable Selection via the Elastic Net." Journal of the Royal Statistical Society, Series B

Author Contribution
Data Cleaning: Jun Ouyang, Zhening Zhang, Richard Murphy
Linear Models: Niall Smith, Boli Zhang, Hong Fan
Neural Networks: Isaac Baron, Suba Rajaram, Phoebe Wang
Random Forest: Shawn Xiang
Support Vector Machine : Shawn Xiang & Steven Gong
K-Nearest-Neighbors: Richard Murphy & Charles Chen
Interactive predictor: Shawn Xiang

Appendix

1.Source code

a. Data Cleaning (SQLite)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpa1ZUeU5fUXRCaUk/view?usp=sharing

b. Linear models (Matlab)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpN19NMkJCS3lRZFk/view?usp=sharing

c. Neural Network (Matlab)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpR1NxTXQydC1Kd2c/view?usp=sharing

d. K Nearest Neighbor (R)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpbHM4UTZPVldMVm8/view?usp=sharing

e. Random Forest (Python)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpYzNpaTE0TzRTbjA/view?usp=sharing

f. SVM(Matlab)
https://drive.google.com/a/ucdavis.edu/file/d/0Bxiq5u8qnnbpNnZfd2tibnZTNTQ/view?usp=sharing

2. Interactive predictor

From our result, we also built an interactive calculator on earning based on features such as SAT score, admit rate, family income and other options.
http://shawnxiang.com/171/