Link to GitHub: https://github.com/c1rowe/118B_Final_Project

Income Classification Based on US Census Information

1. Introduction and Motivation

We used supervised and unsupervised learning to determine which factors about someone living in the United States, contribute most to their income (specifically, earning more than $50,000 or less than $50,000 a year). The factors that we considered were sex, education level, age, occupation, native country, hours worked per week, capital gain/loss, race, and familial information. We thought that this would be an important problem to consider because income disparities have been increasing in the U.S. in recent years, and it is crucial to understand how we can prevent income disparities that are due to preventable structural problems. This topic is also important because it can demonstrate how good machine learning can solve real-life social problems.

2. Related Works

2.1 Recent Works

1) A Statistical Approach to Adult Census Income Level Prediction [1]
This paper also used census data to predict income. In this paper, the author used Ensemble Learning and Boosting Algorithm which is known as Gradient Boosting Classifier. This paper gives us some hints on how to start this project. However, we did not replicate what it did, since we prefer to use the models covered in this class and those we are more familiar with. In this paper, the authors also have mentioned many other works using machine learning models for predicting income levels:

2) Income Prediction via Support Vector Machine[2]
In this paper, the author implemented Principal Component Analysis (PCA) and Support Vector Machine(SVM) methods to predict income data from the Current Population Survey provided by the U.S. Census Bureau. This encouraged us to use PCA and SVM in our projects.

3) Using decision tree classifier to predict income levels[3]
In this paper, Random Forest Classifier was used to predict income levels of individuals based on various attributes of 1994 census database and It achieve 85% predictive accuracy on the test data. This encouraged us to also try out random forest in our projects.

2.2 From Class

1) PCA , SVM and K-Means

We learned K-Means, PCA and SVM in this course. We learnt that PCA is an important way to find important feature and reduce dimensions. SVM are supervised machine learning algorithms which can divide data by embedding them in a higher-dimensional space and hence maximize the margin between the decision boundary and the data points. K-mean can cluster data into multiple clusters without supervision. These 3 algorithms seems suitable to our projects.

3. Methods

We used the Income Classification dataset on Kaggle, uploaded by Lorenzo de Tomasi, which was obtained from the US Census. We performed two unsupervised learning methods K-means and PCA. We also use three supervised learning methods random forest, logistic regression and linear regression. We wrote our code in CoCalc, using Python 3, and using the matplotlib, sklearn, pandas, and numpy packages.

3.1 Pre-processing

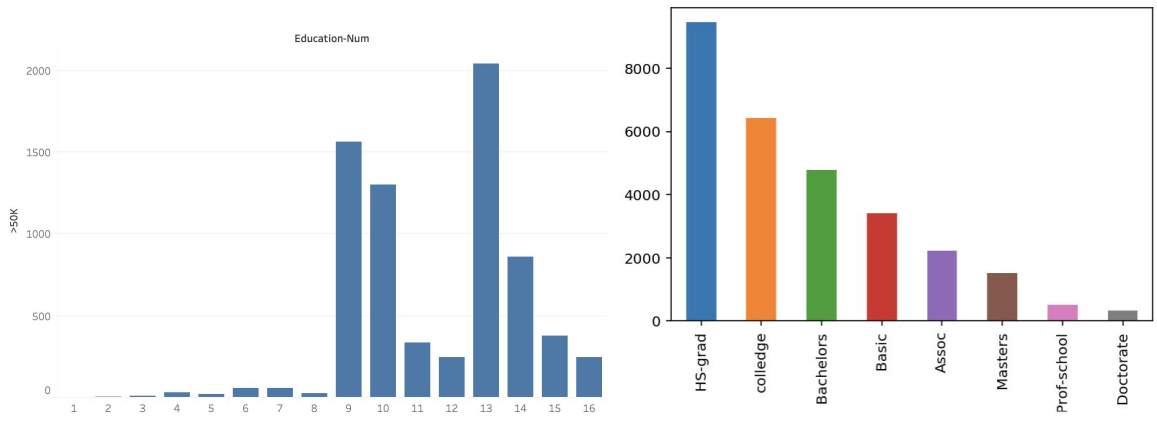
In addition, we pre-processed the data from this dataset using one-hot encoding, since the data were categorical. We created more features (columns), but the entries became numerical, making it easier to perform machine learning algorithms on the data. To pre-process, we also combined many categories together, since they were very similar (e.g. we changed the “separated” marriages to “divorced”, and combined all preschool - high school education into one category). We also disregarded the countries that had less than 100 people originating from them, keeping only the United States, Mexico, the Philippines, Germany, Puerto Rico, and Canada.

We also performed preprocessing on the data that we fed into the machine learning algorithms we used; once categorical features were encoded as binary values, all non-binary features were standardized to their feature mean and scaled by feature variance. This was to speed up the search for parameters in building our machine learning models in a way that would have no effect on the final accuracies. All non-binary data was preprocessed in this way before being used as test and training data.

Although our dataset was unbalanced (about 75% of samples were labeled as <=50K) we chose not to resample the data, and instead include precision, recall, and F-score as added accuracy metrics on top of test and validation accuracy.

3.2 Data Exploration

Before running the machine learning algorithms on our data, we also plot some data to have a general idea about what may be important features and how the data spread out:



3.3 Data Modeling

1) Random Forest

We chose to use the random forest classifier because of its high accuracy and, perhaps more importantly, because it enabled us to pull key features out of the model after it was fitted to determine what features played the biggest role in classification, i.e. which features were most related to whether people made more or less than 50K a year. We searched for optimal hyperparameters using a 3-fold cross-validation grid search, with the following test hyperparameters:

- Max Depth: 8, 10, 12, 14, 16
- Number of Trees: 5, 10, 15, 20, 25
- Max Features: Ranged from 1 to 55 (total features in dataset after preprocessing), 5 values evenly spaced

We kept the split quality criterion fixed as entropy-based.

2) Logistic Regression

We use logistic regression because we are predicting whether the income will be higher than or less than 50K, which is a binary outcome. We ran logistic regression on the PCA component and also the original data. To find the optimal parameters we used GridSearchCV. We set the max iteration to max_iter = 100, solver=liblinear, and C=1.Then, we compare the difference between these two.

3) KMeans

We performed KMeans on our preprocessed data, using the sklearn.cluster.KMeans package, where we specified our number of clusters to be 2 (one for income > $50,000 a year

and one for income <= $50,000 a year). We first did PCA on the data, and chose to use 20 of the components since they explained 60% of the variance of the data.

4) SVM

We also used the SVM classifier as another supervised algorithm, varying kernel type and regularization parameter C in our hyperparameter search as follows:

- Kernel Type: Linear or RBF (Radial Basis Function)
- C Regularization Term: 0.1,1,5,10,15,100,1000
    - For this term we chose to focus on values centered on 10 as that seemed to be the most effective term in earlier hyperparameter searches with a wider range of values for C.

The gamma kernel coefficient was kept fixed at $\frac{1}{55 \times Var(X)}$ (n_features = 55).

4. Results

4.1 Random Forest

Plotting validation scores across the parameters we tried showed the following:

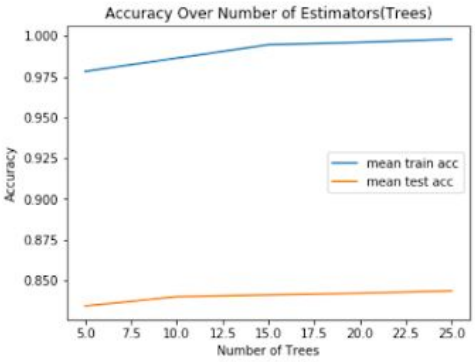


Fig 1: Number of Trees

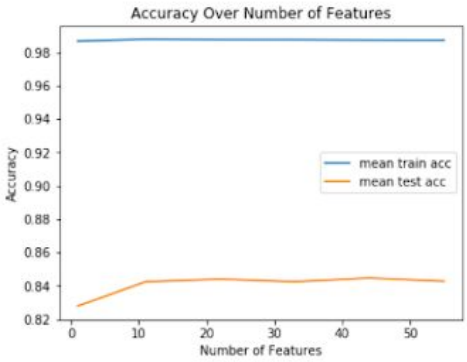


Fig 2: Number of Features

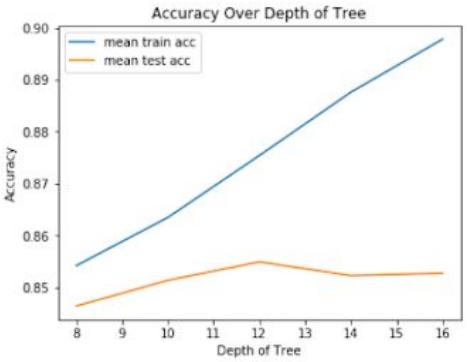


Fig 3: Tree Depth

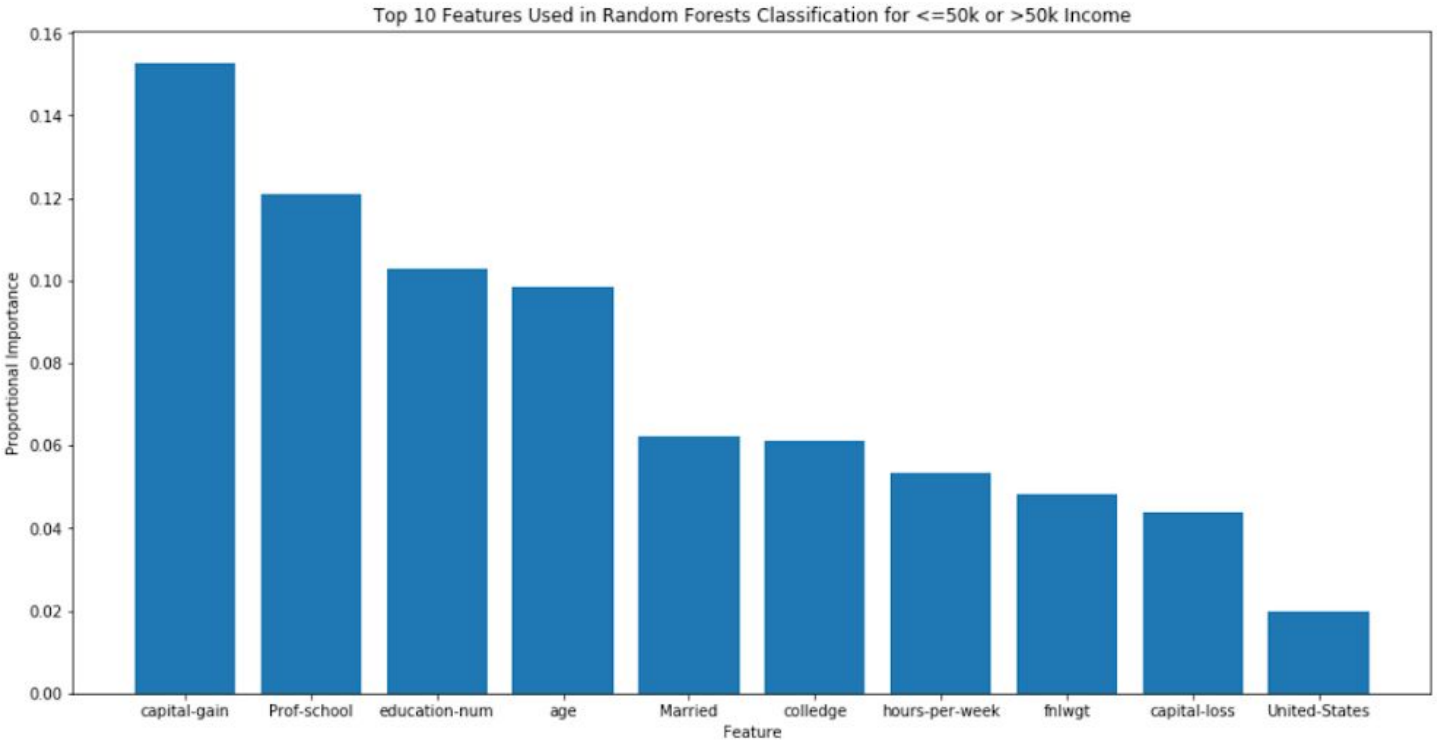
As shown in Fig. 1, both train and test accuracy steadily increased along with the number of trees in the model but more or less flattened out past about 20 trees. Correspondingly, GridSearchCV yielded 20 trees as the optimal number of trees for the final fitted model. For number of features, accuracy plateaued after 11 features (Fig. 2) which became the chosen hyperparameter. Finally, accuracy reached a maximum around the middle of the tree depth range (Fig. 3), and our chosen hyperparameter became 14.

With these optimal hyperparameters, random forests yielded a validation accuracy of approximately 85.6170% and a test accuracy of approximately 85.4101%. For precision/recall metrics, random forests gave the following:

- Precision: 0.80585
- Recall: 0.61007

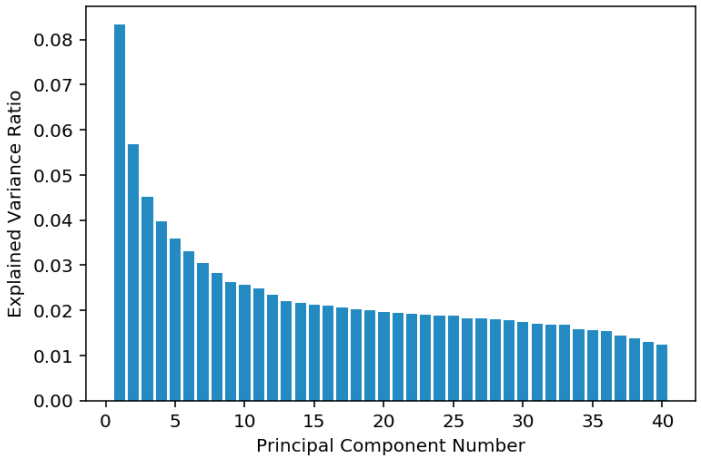
- F-Score: 0.69442

Examining the feature importances of our random forest model revealed that the top four features were capital-gain, Prof-school, education-num, and age. The rest of the top 10 features are represented (by proportional importance) in the figure below.



4.2 Linear Regression

Before performing linear regression, we used PCA to reduce the number of dimensions. After plotting the expected variance ratio of each principal component, we decided to keep 10 of the principal components, as they explained around 40% of the variance in the data. The highest principle component explained around 8% of the variance, so we decided to keep more of the principal components than just the first two.



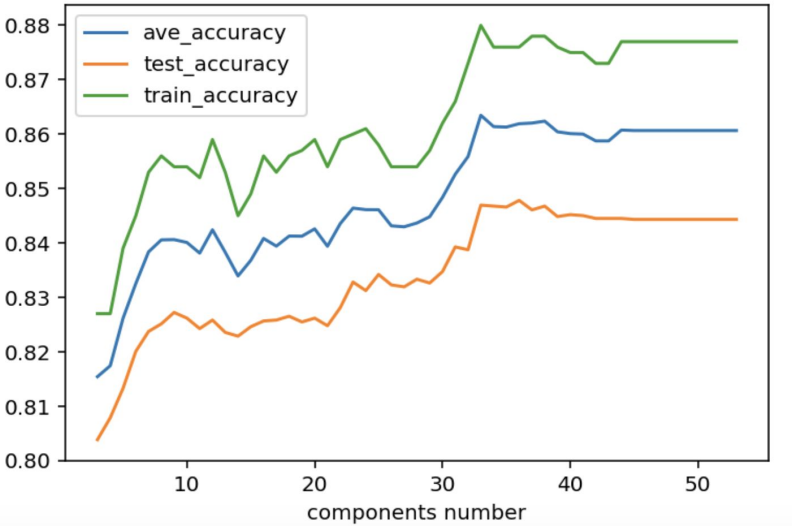
4.3Logistic Regression

On original data (without PCA)

We have the training accuracy of 0.875, and testing accuracy of 0.845. Based on the result, there is no overfitting problem. The time taken to complete the fitting, predicting the training and the testing dataset is 0.0580s. F score for the test result is 0.677

On PCA components

We used the same parameter(max_iter = 100, solver=liblinear, and C=1), and change the number of the components used to measure how this affects accuracy and time.



The testing accuracy increases fast when we increase the component number form 3 to 10 and then increase gradually to 33 components. Since the accuracy increases little after using 33 components, we kept 33 components for the logistic regression.

The training accuracy of this model is 0.876, and testing accuracy of 0.848. There is no overfitting problem. The time taken to complete the fitting, predicting the training and the testing dataset is 0.0440s.

PCA components VS Original Data

The logistic regression with 33 PCA components is 0.014s, which is 24% faster than with the original data, while the accuracy of it is even slightly better.
In general, logistic regression with 33 PCA is a better model for predicting whether one has income greater than 50K

4.4 KMeans

On PCA Components

We chose to keep 20 of the components for KMeans, since they explained around 60% of the variance in the data. We had a training accuracy of around 0.713, and a testing accuracy of around 0.707 (we disregarded the very low accuracies because these were probably due to KMeans clustering incorrectly, as it is known to do). We ran KMeans 10 times, to see if the accuracies changed with different iterations. Some of the iterations seemed to cluster in a way that was not useful to us, as about 4 out of 10 of the trials had a training accuracy of 0.296 and a testing accuracy of 0.288.

Without PCA

We also performed KMeans without PCA, to see if the accuracy changed. We performed KMeans 10 times to make sure that we had a good sense of the training and testing accuracies (as they change with each iteration of KMeans). The higher training accuracy was 0.691 and the higher testing accuracy was 0.689. The lower ones (3 out of 10 trials) had a training accuracy of around 0.312 and a testing accuracy of 0.310.

PCA components VS Original Data

Performing KMeans on the data after doing PCA seems to increase the testing accuracy by about 1.8%, which is just slightly better. Thus, we can see that reducing the dimension makes KMeans predict income greater than $50,000

4.5 SVM

SVM achieved a test accuracy of about 84.8168% with the following precision/recall measures:

- Precision: 0.77240
- Recall: 0.58993
- F-Score: 0.66895

Discussion

With regards to determining the major factors that govern a person's income, our results from random forests suggest that the amount of capital gained, whether a person attended professional school, years of education, and age play the largest part in predicting a person's income. Aside from these top features, the rest of the top 10 features were based primarily on education or work hours except for marriage status and country of origin. This may suggest that people who are married or from places outside of the US may face differences in income, and presents an opportunity for further research to determine the actual underlying relationships between these features and income.

Between our two supervised learning algorithms, random forests did better overall than SVM in solving our binary classification task as we expected. Though random forests had only a marginally better testing accuracy, the precision and recall metrics were significantly better for random forests than SVM, leading to a superior F-score. This suggests that random forests was more robust to the unbalanced nature of this dataset than the SVM classifier.

In this project, we learnt about how to deal with a lot of categorical data. One hot encoding makes the dataset has a lot higher dimensions. Then we practiced PCA we learnt in class to reduce such huge dimension. Our models works fairly well with PCA components.

There are many parts need improvements. In the pre-processing process, since our dataset is not balanced, there are more people have income lower than 50K, we may need to resampling our data so that the recall may be lower and hence we may have better F1 scores. In the data processing part, we may also try deep learning, like ANN because in other papers, it seems works extremely well. We should also understand the parameter of different model better, so we may be able to find more appropriate ones.

Reference

[1] *A Statistical Approach to Adult Census Income Level Prediction, 2018*
https://arxiv.org/ftp/arxiv/papers/1810/1810.10076.pdf

[2] *Income Prediction via Support Vector Machine*, Alina Lazar, International Conference on Machine Learning and Applications - ICMLA 2004, 16-18 December 2004, Louisville, KY, USA.

[3] *Using decision tree classifier to predict income levels*, Sisay Menji Bekena, Munich Personal RePEc Archive 30th July, 2017