Predicting Annual Earnings Based on Demographic and Employment Data

Maxime Voisin

Introduction

The U.S. Department of Commerce launched Census Bureau to gather more data on the population's annual earnings, employment and demographics. Census Bureau provides a public dataset containing the annual earnings as well as 41 employment/demographics variables for ~300,000 individuals.

I tackle the binary classification problem: based on these 41 employment/demographics variables, how well can I predict if someone earns more than $50,000/year ?

Data Preprocessing

I. From Qualitative to Quantitative Data

I turned qualitative categorial features into nominal categorical variables (label encoding) or separate indicator features (dummy encoding). I also deleted meaningless features (0 variance) and replaced null values with 0s.

II. Selecting Training and Test Examples

I set aside 200,000 examples for training, and the remainder (~100,000) for testing. This split did not lead to any covariate shift: classifiers were unable to recognize whether the data came from the train or test data (66.6% accuracy, no better than the baseline).

III. Normalizing

I normalized all of our training data to have mean 0 and standard deviation 1.

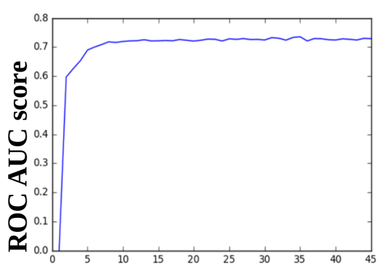
Feature Engineering

I. Computing new features

Based on raw features, I computed new relevant features. This enables models to classify the data more easily. For instance, I computed net capital as the difference between capital gains and capital losses. This engineered feature ranked 3rd most useful feature in the classification task !

II. Feature selection

The previous steps increased the number of features from 41 to 221. I ran sequential backward-based feature elimination. Features were selected based on their mean-squared ROC AUC score using 3-fold cross-validation. The optimal number of features to keep was 171.



	Before feature engineering	After feature engineering
Precision	0.70	0.72
Recall	0.35	0.39

Impact of feature engineering

Fighting Imbalanced Classes

94% of the data belongs to one class of the target variable. The baseline scores very high in terms of accuracy. How do I fight this?

I. Select appropriate metrics

In the context of imbalanced classes, accuracy does not provide valuable feedback on the performance of classifiers. Instead, I rely on confusion matrices (esp. precision and recall) and ROC curves.

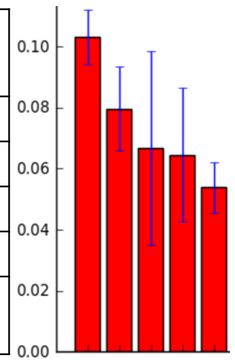
II. Select an appropriate cross-validation strategy

I used stratified 3-Fold cross-validation to make sure that, in each cv fold, the proportion of 0/1 target variable is identical to that of the training set. Otherwise, some cv folds might receive too few training examples from the "rare" class, undermining the classifier's performance.

III. Use class weights

When classifiers allow it, I penalize the most frequent class, by weighting each class inversely proportional to its frequency. It increased ROC AUC by 16%.

Ranking	Feature	Feature importance
1	age	0.10
2	stock options	0.07
3	net capital	0.06
4	industry code	0.06
5	nb of people worked for employer	0.05

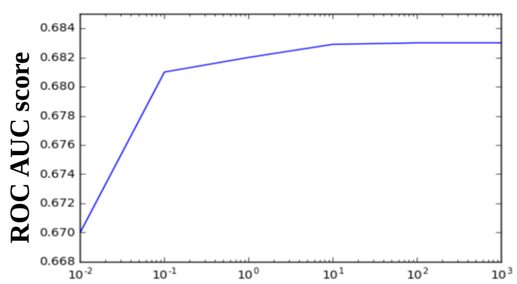


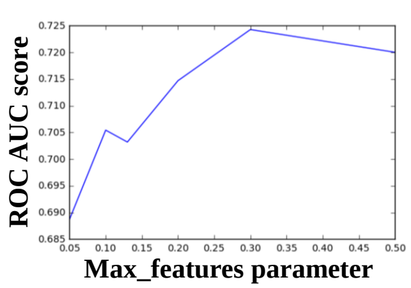
Top 5 features after feature engineering

Fine-tune individual classifiers

I. Logistic Regression

I ran logistic regression with L1 and L2 regularization. I then fine-tuned the parameter that controls regularization – C - in order to maximize the mean-squared ROC AUC score using 3-fold cross-validation: C=10





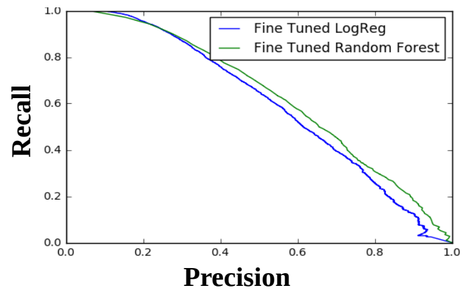
II. Random Forests

I also used random forest models. The ROC AUC score keeps increasing with the number of estimators (trees).I found a tradeoff between computation time and performance, by setting this parameter to 500: after 500, the score increases very slowly with the number of estimators.

I also fine-tuned the max_features parameter, that controls the fraction of features considered by each tree.

Results & Analysis

The following graph shows the precision/recall curve of the models I tested.

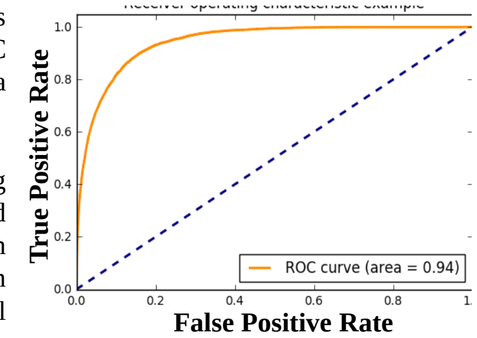


Log Reg and RF have similar confusion matrices (precision / recall)

I look at the ROC curve to decide between the two models.

Log Reg clearly outperforms RF, with the following ROC curve. The classifier scores a ROC AUC of 0.84.

To further improve the Log Reg performance, I changed the classification decision threshold (by default 0.5), in order to have a better recall score.



Ensemble learning

In order to leverage the strengths of each classifier, a grid-searched logistic regression combines the predictions of:
- the fine-tuned logistic regression and random forest described above
- non tuned SVM (linear and polynomial kernel)
- non tuned Naïve Baye classifier

Stacking requires a validation set: I split the training set (200,000 examples) between a working set (150,000) and a validation set (50,000).

The stacking is still running at the time of writing the poster. Ensemble learning results will be included results in the final report.

Discussion

This elegant solution - based on rich preprocessing, feature engineering, optimization of individual classifiers and ensemble learning enabled us to increase the ROC AUC from 0.62 to 0.84 (the ensemble learning results to come might improve this even further).

This project also highlighted the most important features: age, stock options, net capital, industry of work and number of people managed at work.

Finally, a further step could be to incorporate more models in the final stacking. One could also try to tackle this problem from a regression (rather than classification) point of view, by considering the target variable as a continuous variable and trying to predict it.