Predictive Model for Census Data

Artiticial Intelligence Final Project

Raju Khanal (110849511)
Devesh Sisodia (110951296)
Akasha Roy (111121421)

Abstract

Given various features, the aim is to build a predictive model to determine the income level for people in US. The income levels are binned at below 50K and above 50K.

Introduction

Problem Definition

We have to build a predictive model to determine the income level for people in US. From the problem statement, its evident that this is a binary classification problem.

There exists a significant impact of the variables (below) on the dependent variable.

- Age
- Marital Status
- Income
- Family Members
- No. of Dependents
- Tax Paid
- Investment (Mutual Fund, Stock)
- Return from Investments
- Education
- Spouse Education
- Nationality
- Occupation
- Region in US
- Race
- Occupation category

Remind you, this is not an exhaustive list.

Motivation

The goal of a classification algorithm is to attempt to learn a separator (classifier) that can distinguish the two. There are many ways of doing this, based on various mathematical, statistical, or geometric assumptions. But when we start looking at real, uncleaned data one of the first things you notice is that its a lot noisier and imbalanced. The data used in this project is imbalanced. In real life, some extremely critical situations result in imbalanced data sets.

For example fraud detection, cancer detection, manufacturing defects, online ads conversion etc. Thus, having prior experience of working on such data has many practical real-world applications.

Research on imbalanced classes often considers imbalanced to mean a minority class of 10% to 20%. In reality, datasets can get far more imbalanced than this. Here are some examples:

- About 2% of credit card accounts are defrauded per year1. (Most fraud detection domains are heavily imbalanced.)
- Medical screening for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is 0.4
- Disk drive failures are approximately 1% per year.
- The conversion rates of online ads has been estimated to lie between 10-3 to 10-6.
- Factory production defect rates typically run about 0.1%.

Contributions

Application Prediction task is to determine the income level for the person represented by the record. Incomes have been binned at the 50K level to present a binary classification problem, much like the original UCI/ADULT database. The goal field of this data, however, was drawn from the ”total person income” field rather than the ”adjusted gross income” and may, therefore, behave differently than the orginal ADULT goal field.

The data used for this project has been taken from the following source:

http://archive.ics.uci.edu/ml/machine-learning-databases/census-income-mld/census-income.names

The final data set has been taken from the following, which is a modified version of the previous link.

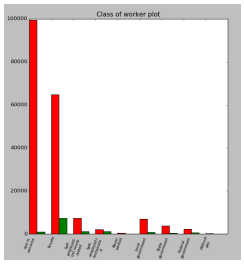
https://www.analyticsvidhya.com/blog/2016/09/this-machine-learning-project-on-imbalanced-data-can-add-value-to-your-resume/comment-119632

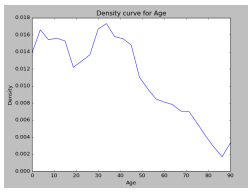
Description

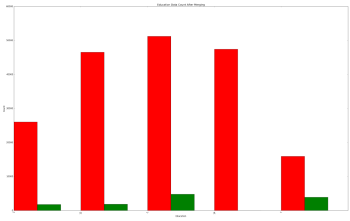
In this project, we will use the US census data which contains information about individuals as our data set. We will attempt to build a model to predict if the income of any individual in the US is greater than or less than USD 50000 based on this information available in the census data. Popularly known as the Adult data set, this dataset, used for the purposes of this project is an extraction from the 1994 census data by Barry Becker and donated to the public site http://archive.ics.uci.edu/ml/datasets/Census+Income. We have followed these steps in order to perform our case study:

1. Acquiring the data- We have downloaded the data directly from the internet, from its original source.

2. Data Exploration- Specifically the predictor variables (also called independent variables features) from the Census data and the dependent variable which is the level of income.
Plotting these graphs helps us better idea about the data that we are using.







3. Reading the data- We then read the downloaded data

4. Check if cleaning required and if necessary, clean it- Usually data obtained from the real world is messy and requires cleaning. We will check if our data needs to be cleaned so as to avoid introducing errors in our learning curve. If required, we restructure the data as necessary so as to aid exploration and modelling and to correctly predict the income level.The training data set is cleaned for missing or invalid data.

About 8% (2399/30162) of the dataset has NAs in them. It is observed that in most of the missing dataset, the workclass variable and occupation variable are missing data together. And the remaining have nativecountry variable missing. We could, handle the missing values by imputing the data. However, since workclass, occupation and nativecountry could potentially be very good predictors of income, imputing may simply skew the model.

5. Study the independent variables- A very crucial step before modelling is the exploration of the independent variables. Exploration provides great insights to an analyst on the predicting power of the variable. An analyst looks at the distribution of the variable, how variable it is to predict the income level, what skews it has, etc. In most analytics project, the analyst goes back to either get more data or better context or clarity from his finding.

6. Build the prediction model with the training data.

7. Validate the prediction model with the testing data- Here the built model is applied on test data that the model has never seen. This is performed to determine the accuracy of the model in the field when it would be deployed. Since this is a case study, only the crucial steps are retained to keep the content concise and readable.

Basic statistics for this data set: — — Number of instances data = 199523
— Duplicate or conflicting instances : 46716
— Number of instances in test = 99762
— Duplicate or conflicting instances : 20936
— Class probabilities for income-projected.test file
— Probability for the label '- 50000' : 93.80
— Probability for the label '50000+' : 6.20
— Majority accuracy: 93.80
— Number of attributes = 40 (continuous : 7 nominal : 33)
— Information about .data file :
— 91 distinct values for attribute 0 (age) continuous
— 9 distinct values for attribute 1 (class of worker) nominal
— 52 distinct values for attribute 2 (detailed industry recode) nominal
— 47 distinct values for attribute 3 (detailed occupation recode) nominal
— 17 distinct values for attribute 4 (education) nominal
— 1240 distinct values for attribute 5 (wage per hour) continuous
— 3 distinct values for attribute 6 (enroll in edu inst last wk) nominal
— 7 distinct values for attribute 7 (marital stat) nominal
— 24 distinct values for attribute 8 (major industry code) nominal
— 15 distinct values for attribute 9 (major occupation code) nominal
— 5 distinct values for attribute 10 (race) nominal
— 10 distinct values for attribute 11 (hispanic origin) nominal
— 2 distinct values for attribute 12 (sex) nominal

— 3 distinct values for attribute 13 (member of a labor union) nominal

— 6 distinct values for attribute 14 (reason for unemployment) nominal

— 8 distinct values for attribute 15 (full or part time employment stat) nominal

— 132 distinct values for attribute 16 (capital gains) continuous

— 113 distinct values for attribute 17 (capital losses) continuous

— 1478 distinct values for attribute 18 (dividends from stocks) continuous

— 6 distinct values for attribute 19 (tax filer stat) nominal

— 6 distinct values for attribute 20 (region of previous residence) nominal

— 51 distinct values for attribute 21 (state of previous residence) nominal

— 38 distinct values for attribute 22 (detailed household and family stat) nominal

— 8 distinct values for attribute 23 (detailed household summary in household) nominal

— 10 distinct values for attribute 24 (migration code-change in msa) nominal

— 9 distinct values for attribute 25 (migration code-change in reg) nominal

— 10 distinct values for attribute 26 (migration code-move within reg) nominal

— 3 distinct values for attribute 27 (live in this house 1 year ago) nominal

— 4 distinct values for attribute 28 (migration prev res in sunbelt) nominal

— 7 distinct values for attribute 29 (num persons worked for employer) continuous

— 5 distinct values for attribute 30 (family members under 18) nominal

— 43 distinct values for attribute 31 (country of birth father) nominal

— 43 distinct values for attribute 32 (country of birth mother) nominal

— 43 distinct values for attribute 33 (country of birth self) nominal

— 5 distinct values for attribute 34 (citizenship) nominal

— 3 distinct values for attribute 35 (own business or self employed) nominal

— 3 distinct values for attribute 36 (fill inc questionnaire for veteran's admin) nominal

— 3 distinct values for attribute 37 (veterans benefits) nominal

— 53 distinct values for attribute 38 (weeks worked in year) continuous

— 2 distinct values for attribute 39 (year) nominal

Evaluation

Our general expected performance is 94%. But we will try to improve this by using several techniques.

- Naive Bayes:

  Nave Bayes is good not only when features are independent, but also when dependencies of features from each other are similar between features: Essentially, the dependence distribution; i.e., how the local dependence of a node distributes in each class, evenly or unevenly, and how the local dependencies of all nodes work together, consistently (supporting a certain classication) or inconsistently (canceling each other out), plays a crucial role. Therefore, no matter how strong the dependencies among attributes are, Naive Bayes can still be optimal if the dependencies distribute evenly in classes, or if the dependencies cancel each other out.

- Logistic Regression

  Logistic regression is a technique that is well suited for examining the relationship between a categorical response variable and one or more categorical or continuous predictor variables.

  The estimates from logistic regression characterize the relationship between the predictor and response variable on a log-odds scale.

  A logistic regression model has been built and the coefficients have been examined. However, some critical questions remain. Is the model any good? How well does the model fit the data? Which predictors are most important? Are the predictions accurate? The rest of this document will cover techniques for answering these questions.

- Decision Trees

  Decision Trees are very flexible, easy to understand, and easy to debug. They will work with classification problems and regression problems. But decision trees tend to over fit the training data more so that other techniques which means we generally have to do tree pruning and tune the pruning procedures. Also splitting a lot leads to complex trees and raises probability that we are overfitting.

- Random Forests

  The random forest starts with a standard machine learning technique called a decision tree which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.

  Eventually more trees doesn't do much good. It doesn't improve error. It does take more memory and cpu.

  When you start pruning, both by requiring at least so many samples to make a leaf, and by only allowing the tree to get so many levels deep, it substantially improves memory.

- SVM

  Support Vector Machines (SVMs) use a different loss function from LR. They are also interpreted differently (maximum-margin). However, in practice, an SVM with a linear kernel is not very different from a Logistic Regression. The main reason we would want to use an SVM instead of a Logistic Regression is because our problem might not be linearly separable. In that case, we will have to use an SVM with a non linear kernel (e.g. RBF). The truth is that a Logistic Regression can also be used with a different kernel, but at that point we might be better

off going for SVMs for practical reasons. Another related reason to use SVMs is if we are in a highly dimensional space. For example, SVMs have been reported to work better for text classification.

Unfortunately, the major downside of SVMs is that they can be painfully inefficient to train. So, its hard to implement them for any problem where we have many training examples, eg., "industry scale" applications. Anything beyond a toy/lab problem might be better approached with a different algorithm.

- XGBoost

  XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. A problem with gradient boosted decision trees is that they are quick to learn and overfit training data.

  One effective way to slow down learning in the gradient boosting model is to use a learning rate, also called shrinkage (or eta in XGBoost documentation).

  Advantages:

  – Regularization:
  – Parallel Processing
  – High Flexibility
  – Handling Missing Values
  – Tree Pruning
  – Built-in Cross-Validation
  – Continue on Existing Model

Sampling

If we look at the severity of imbalanced classes in our data, we see that the majority class has a proportion of 94%. In other words, with a decent ML algorithm, our model would get 94% model accuracy.

In absolute figures, it looks incredible. But, our performance would depend on, how good can we predict the minority classes.

We need to apply sampling techniques so that the minority classes are also represented well.

Now, well try to make our data balanced using various techniques such as oversampling, undersampling and SMOTE. In SMOTE, the algorithm looks at n nearest neighbors, measures the distance between them and introduces a new observation at the center of n observations. While proceeding, we must keep in mind that these techniques have their own drawbacks such as:

- undersampling leads to loss of information
- oversampling leads to overestimation of minority class

We see that train.smote gives the highest true positive rate and true negative rate. Hence, we learn that SMOTE technique outperforms the other two sampling methods.

Weight

Assigning class weight tells the algorithm that this (minority) class is more important. The other class may be predicted as well as possible, but the minority class has to be predicted with full certainty since it is given higher weight.

In technical terms, a classifier built on imbalanced data tends to overlook the minority class as noise and ends up predicting the majority class accurately. These weights are nothing but the misclassification cost imposed on classifying the classes incorrectly. Higher weight suggests high cost of misclassification which the algorithm attempts to avoid.

Learnings

Based on our experiment, the following results were obtained. We found that the best result was achieved by XGBoost, which increased the precision of the minority class/ significantly to 75%.

Naive Bayes

	precision	recall	f1-score	support
Less than 50k	0.99	0.79	0.88	93576
More than 50k	0.21	0.82	0.33	6186
avg / total	0.94	0.79	0.84	99762

SVM using class_weight = Balanced

	precision	recall	f1-score	support
Less than 50k	0.98	0.77	0.86	93576
More than 50k	0.18	0.73	0.28	6186
avg / total	0.93	0.77	0.83	99762

Random Forest

	precision	recall	f1-score	support
Less than 50k	0.96	0.99	0.97	93576
More than 50k	0.65	0.38	0.48	6186
avg / total	0.94	0.95	0.94	99762

Decision Tree

	precision	recall	f1-score	support
Less than 50k	0.96	0.96	0.96	93576
More than 50k	0.43	0.46	0.44	6186
avg / total	0.93	0.93	0.93	99762

XGB Classifier

	precision	recall	f1-score	support
Less than 50k	0.96	0.99	0.98	93576
More than 50k	0.75	0.35	0.47	6186
avg / total	0.95	0.95	0.94	99762

XGB Classifier Smote 0.4

	precision	recall	f1-score	support
Less than 50k	0.97	0.97	0.97	93576
More than 50k	0.56	0.54	0.55	6186
avg / total	0.94	0.95	0.94	99762

We can see from the result that Naive-Bayes gets only 21% precision for minority class. This was increased to 28%

by SVM. We found that RF was giving 68% precision for Minority class. Similarly Decision Tree was found to give only 43%, whereas the best was achieved by XGBoost giving 75% precision, and hence we selected this as our predictive model because the accuracy of the majority class is still 96%.

Comparisons

There are a number of dimensions we can look at to give we a sense of what will be a reasonable algorithm to start with, namely:

- Number of training examples
- Dimensionality of the feature space
- Do I expect the problem to be linearly separable?
- Are features independent?
- Are features expected to linearly dependent with the target variable?
- Is overfitting expected to be a problem?
- What are the system's requirement in terms of speed/performance/memory usage...?

Comparison between NB and Logistic Regression

They both train feature weights for the linear decision function. The difference is how we fit the weights from training data.

In NB, we set each feature's weight independently, based on how much it correlates with the label. (Weights come out to be the features' log-likelihood ratios for the different classes.)

In logistic regression, by contrast, we set all the weights together such that the linear decision function tends to be high for positive classes and low for negative classes. (Linear SVM's work the same, except for a technical tweak of what "tends to be high/low" means.)

The difference between NB and Logistic Regression happens when features are correlated. Say we have two features which are useful predictors – they correlate with the labels – but they themselves are repetitive, having extra correlation with each other as well. NB will give both of them strong weights, so their influence is double-counted. But logistic regression will compensate by weighting them lower.

This is a way to view the probabilistic assumptions of the models; namely, Naive Bayes makes a conditional independence assumption, which is violated when we have correlated/repetitive features.

NB can perform better when there's a low amount of training data. But LR should always outperform given enough data.

Comparison between NB and Decision Trees

Naive Bayes requires us to build a classification by hand. Decisions trees will pick the best features for us from tabular data. If there were a way for Naive Bayes to pick features we would be getting close to using the same techniques that make decision trees work like that. NB can perform quite well, and it doesn't over fit nearly as much so there is no need to prune or process the network. That makes them simpler algorithms to implement. However, they are harder to debug and understand because it's all probabilities getting multiplied 1000's of times so we have to be careful to test it's doing what we expect. Naive bayes does quite well when the training data doesn't contain all possibilities so it can be very good with low amounts of data. Decision trees work better with lots of data compared to Naive Bayes.

Conclusions

Due to imbalanced data, our minority class precision value was very less (Using NB it was only 21%). Hence to improve the precision of this minority class, we applied different techniques like sampling, Random Forest Classifier and XGBoost. We found that XGBoost performs the best and gives an accuracy of 75% for minority class. Hence XGBoost is a better classifier for this kind of imbalanced data.

References

*h*ttps://www.analyticsvidhya.com/blog/2016/09/this-machine-learning-project-on-imbalanced-data-can-add-value-to-your-resume/comment-119632

*h*ttps://archive.ics.uci.edu/ml/datasets/Census+Income

*h*ttps://www.knowbigdata.com/blog/predicting-income-level-analytics-casestudy-r

*h*ttp://stats.stackexchange.com/questions/23490/why-do-naive-bayesian-classifiers-perform-so-well

*Q*uora.com

*h*ttps://svds.com/learning-imbalanced-classes/