Income Group Classification Models

STAT 4249
March 12, 2014

Team 26
Benjamin Seiyon LEE
Sou Min Sonia LEE
Mengying LI
Wuwei TANG
Xiaolei XU

Motivation:

Take a product: a car, a smartphone, or even cereal. Now find a wealthy venture capitalist to inject millions, even billions, into R&D to make it the "next big thing." What good is all this if we can't find the right people to buy it? For premium goods, marketing campaigns must be geared towards those with enough disposable income. Individual-level income data is not readily available. So, we must find a way to classify people into income groups based on other demographic variables such as sex, age, race, education, and marital status. This report present our results from three classification models (Naïve Bayesian Classifier, K-Nearest Neighbor, and Logistic Regression) using a sample from the 1994 Current Population Survey (CPS) administered by the US Census Bureau.

Executive Summary:

Using data from the 1994 CPS, our team utilized three methods to classify our sample into two income groups (<\$50K, \$50K+). The *Naive Bayesian classifier* is an efficient probabilistic learning method based on the Bayes' Theorem. The misclassification rate for income was 19.6%. The *k-Nearest Neighbor* method is a more flexible model, which classifies cases based on the average value of their *k*-nearest neighbors. Our optimal model with a k=60 resulted in a misclassification rate of 15.9%. Next, the *logistic regression model with elastic-net regularization* is a simple logistic regression model that is able to select the optimal predictors and shrink their coefficients through the use of an L1 and L2 penalty term. In other words, it is a more efficient and accurate version of the simple logistic regression model. The misclassification rate is an impressive 14.8%.

Other than each model's misclassification rates, there are other properties and limitations that should be scrutinized. The *Naive Bayesian Classifier* does not require our data to be normally distributed (bell-curve); however, it does require its predictor variables (features) to be conditionally independent. To illustrate, correlated variables like education and income cannot be used in a model. *k-NN* methods are extremely flexible and have low levels of bias, but they lack interpretability. Since parameters are not used like in a traditional regression model, k-NN methods are seen as a "black box." The *Logistic Regression model with elastic net regularization* allows us to "fine tune" a generic logistic regression model using a penalty parameter (λ); however, it may introduce unnecessary extra bias.

Our team recommends using the *logistic regression model with elastic net regularization* because it is strong on two fronts – prediction and interpretability. Its low misclassification rate lends credence to its predictive power. Moreover, its logistic regression roots and its ability to shrink coefficients make this model very useful for interpretability. None of the other methods are strong on both these areas. Moving forward, it may be useful to explore other possible areas of interest such as decision tree modeling.

Notes on Data Pre-processing:

After loading in our data, we converted capital gain and capital loss into categorical variables. More than 90% of the observations had a value of "0" for capital gains or capital losses. We did not use education level in our models since it is highly correlated with education num. By doing this, we can assume that all predictor variables are approximately conditionally independent, which satisfies the assumption of Naïve Bayes and our other models.

I. Naïve Bayesian Classifier:

Background:

Naïve Bayesian analysis is generally used in predicting categorical responses from mostly categorical predictor variables. The basic idea of naïve Bayesian method is to search over the training dataset to find cases that match exactly the values of predictor variables of input data, and then use the most frequent response of the matched cases for deciding whether the prediction for y should be 0 or 1(in our case >50K or <=50K).

The underlying theory of Naïve Bayes is based on the Bayes formula, which we can write as:

$$P(y=1 \mid X_1=x_1, X_2=x_2, \cdots, X_k=x_k) = \frac{P(X_1=x_1, X_2=x_2, \cdots, X_k=x_k \mid y=1)\cdot P(y=1)}{P(X_1=x_1, X_2=x_2, \cdots, X_k=x_k \mid y=1)\cdot P(y=1) + P(X_1=x_1, X_2=x_2, \cdots, X_k=x_k \mid y=0)\cdot P(y=0)}$$

This result is exact, and follows basic conditional probability rules. But also this solution is difficult to implement, because with a fine categorization of predictor variables it will be difficult to estimate the conditional joint probabilities $P(X_1=x_1, X_2=x_2, \cdots, X_k=x_k \mid y=1)$ and $P(X_1=x_1, X_2=x_2, \cdots, X_k=x_k \mid y=0)$. The prior probabilities, $P(y=1)$ and $P(y=0)$, on the other hand, are easy to estimate. We can use the frequencies from the training set.

For conditional joint probabilities, the Naive Bayesian approach assumes that the predictors are independent if we condition them on the response. Under this assumption, we can write the previous formula as

$$P(y=1 \mid X_1=x_1, X_2=x_2, \cdots, X_k=x_k) = \frac{P(y=1)\prod_{i=1}^{k} P(x_i \mid y=1)}{P(y=1)\prod_{i=1}^{k} P(x_i \mid y=1) + P(y=0)\prod_{i=1}^{k} P(x_i \mid y=0)}$$

Now we apply this method to the data. Estimates of the prior probabilities (see Table 1) and conditional probabilities $P(X_i=x_i \mid y=1)$ and $P(X_i=x_i \mid y=0)$ for each feature are needed for the implementation of the naive Bayesian approach. We take the probability of income conditioning on Gender as an example (see Table 2). The probabilities are estimated from the training set which consists of half of observations randomly selected from the data provided.

Table 1. Prior Probability

P(y <=50K)	P(y>50K)
0.752	0.248

Table 2. Conditional Probability of Gender

	P(y <=50K)	P(y>50K)
Male	0.613	0.851
Female	0.387	0.149

Then, we apply our model built from the training dataset on our test data. We score a case as 'income>50K' if its probability is 0.5 or larger, and as 'income<=50K' otherwise. From the predicted data, we obtain a 2 × 2 table of classifications (see Table 3), and from that table we could calculate the proportions of incorrect classifications (misclassifications). **The overall misclassification rate of naive Bayesian approach is (875+2330)/16280=19.6%.** More specifically, we fail to identify 875/3926=22.29% of observations with income > 50K correctly as income>50K. For those observations with income<50K, 2330/(10024+2330)= 18.86% of them are incorrectly predicted as income>50K.

Table 3. Misclassification Rate

Actual / Predict	<=50K	>50K
<=50K	10024	875
>50K	2330	3051

Model Summary:

Our naive Bayesian classifier was quite successful on our test data set. The misclassification rate for income was 19.6% (illustrated in Table 3). In contrast to LDA or QDA, which assumes the underlying data are drawn from a Gaussian distribution, no stringent assumptions are needed for the distribution and structure of the data in the naive Bayesian classifier.

The Bayes Classifier requires features that are conditionally independent. For example, among those with income>50K, their education number may have a positive correlation with their working hours per week; therefore education and working hours are not conditionally independent. Our results show that the misclassification rate is still about 20% which is higher than our kNN model (next section). This may stem from the fact that these features are not strictly conditionally independent.

II. k-NN Methods

Background:

Regular linear regression makes assumptions about the structure of the data (high bias), but its predictions are stable (low variance). Models with high bias will inevitably yield stable yet inaccurate predictions, so we need a more flexible model that makes fewer assumptions. In contrast to linear regression methods, the k-nearest neighbor methods implement non-linear, and sometimes disjoint, boundaries to our training and test data.

The k-Nearest Neighbors Method uses the average outcome value of its closest (k) neighbors based on a closeness metric. In this case, the closeness metric is Euclidian distance.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

We are faced with a classification problem where the outcome variable is binary indicator variable (1=Earns more than \$50K, 0=Earns less than \$50K). Our k-nearest neighbor approach will assign a point (in an input space) to either group based on the majority vote of the points in its neighborhood (based on k).

As previously mentioned, the k-NN methods, while unbiased, have high variance when it comes to predictions. To counteract this, we need to split our data set into two independent samples – training and test data. The sum-of-squared errors on the *training* data set will always be the lowest for k=1, so we need to cross-validate on the test data set. Our best model will be the one with parameter k that minimizes the misclassification rate for the test data set.

Methods
The IBk function in R provides k-NN classifiers. The function turns the different levels of categorical variables (e.g. NativeCountry) into dummy variables and also uses normalized distances so that variables with different scales have the same effect on the distance function.
We ran the function to create k-NN classifers for Income based on 12 variables: WorkClass, EducationNum, MaritalStatus, Occupation, Relationship, Race, Sex, HoursperWeek, NativeCountry, CG, and CL (e.g. CapitalGain and CapitalLoss coded as categorical variables). We excluded Fnlwgt because it is uncorrelated with income and we excluded education because we believe it does not provide additional information after including EducationNum.

```
for(i in 1:nrow(knntab1)){
model.knn = IBk(Income ~ Age + WorkClass + EducationNum + MaritalStatus
        + Occupation + Relationship + Race + Sex + HoursperWeek + NativeCountry
        + CG + CL, data=training, control = Weka_control(K=knntab1[i,1]))
prediction.test <-predict(model.knn, newdata = test, type='class')
error.rate.test <- sum(test$Income != prediction.test) / nrow(test)
knntab1[i,2] = error.rate.test
}
```

Results:
A *k* of approximately 60 yields the lowest misclassification rate for our test data. For training data, the misclassification rate grows larger as k increases.

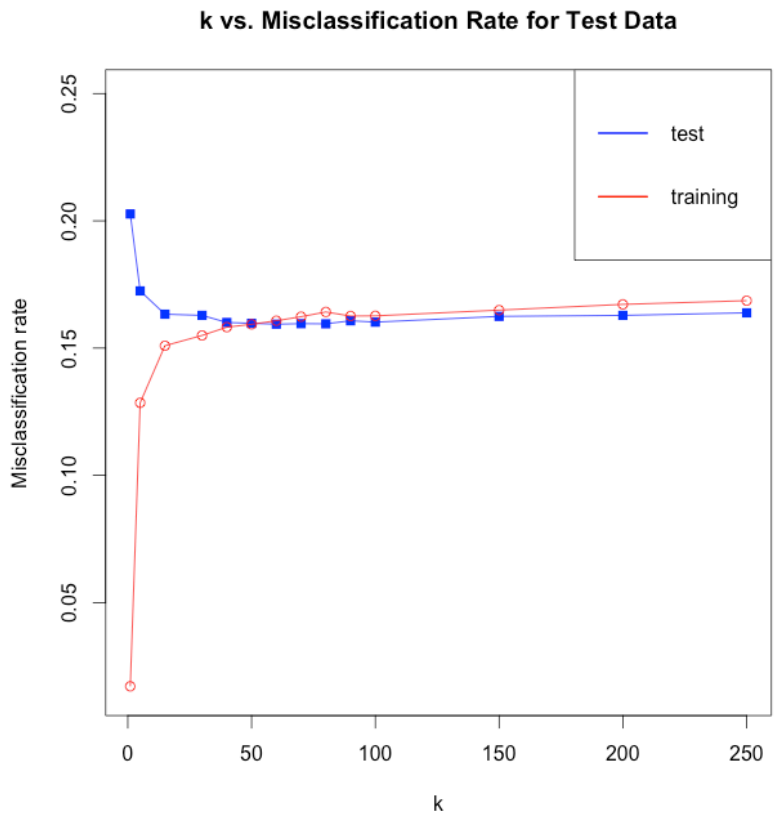


Table 4. k-NN Test Misclassification Rate

	Misclassification Rate	
k	**Test**	**Training**
1	0.2027	0.0171
5	0.1725	0.1285
15	0.1633	0.1509
30	0.1629	0.1550
40	0.1601	0.1582
50	0.1597	0.1594
60	0.1594	0.1607
70	0.1596	0.1623
80	0.1595	0.1642
90	0.1607	0.1626
100	0.1602	0.1627
150	0.1625	0.1649
200	0.1629	0.1672
250	0.1638	0.1686

Model Summary:

Our tests show that the optimal parameter *k* is 60 with a test data misclassification rate of 15.94%. The KNN method permits us to use a more flexible decision boundary, in contrast to the rigid regression boundaries. However, KNN method proves to be problematic with high variance and also low interpretability of our models.

1. **High Variance**: The bias-variance tradeoff is present in KNN methods for classification. While there is low bias, models tend to vary more frequently from one to another. An effective k-parameter in one model will not be effective in others. Therefore, cross-validation (test data set) is extremely important.
2. **Prediction vs. Interpretability**: In cases where prediction is our main purpose, the k-NN methods will prove to be an effective classifier. However, it does not allow us to interpret the effect of different predictor variables on our dependent variable of interest. Regression models have p-parameters so we can interpret our model results easily. For instance, "holding all variables constant, a one unit increase in predictor x1 will lead to a 0.556 unit increase in the logit (log-odds) of being in class A as opposed to class B." Since kNN methods only have one parameter (k), we cannot make any inferences about our data.

III. Logistic Regression:

Background:
In statistics, logistic regression is a type of probabilistic statistical classification model. It is also used to predict a binary response from a binary predictor, used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features). The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory variables, using a logistic function.

The particular model used by logistic regression, which distinguishes it from standard linear regression and from other types of regression analysis used for binary-valued outcomes, is the way the probability of a particular outcome is linked to the linear predictor function:

$$\log it(E[Y_i \mid x_{1,i},\cdots x_{m,i}]) = \log it(p_i) = \ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_0 x_{1,i} + \cdots + \beta_m x_{m,i}$$

Results:
Our logistic regression model yields a misclassification rate of 34.58%, which is much higher than our previous two models.

Table 5. Misclassification Rate

Actual / Predict	<=50K	>50K
<=50K	9035	2982
>50K	2273	791

Model Summary:
Without parameter selection or other process, we can tell from the result above that logistic regression itself, with a misclassification rate as high as 34.84% in the test set, is insufficient.

IV. Logistic Regression with Elastic Net Regularization:

Background:
In statistics and, in particular, in the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods. The elastic penalty provides a compromise between ridge and lasso.

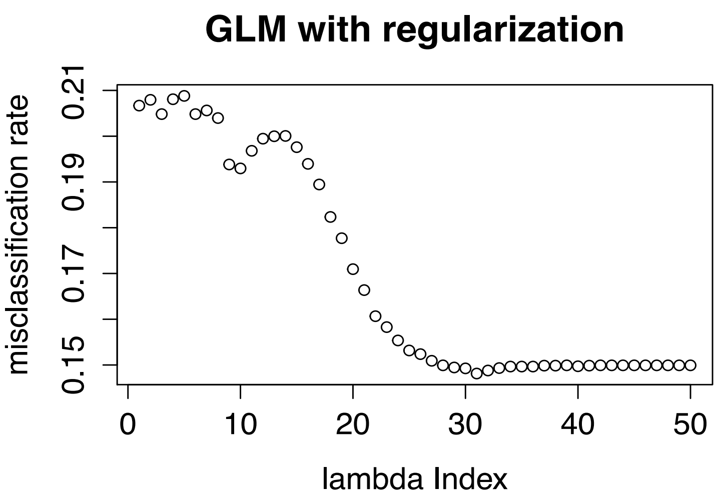
$$\lambda \sum_{j=1}^{p} (\alpha \beta_j^2 + (1-\alpha)|\beta|)$$

The elastic-net selects variables like the lasso, and shrinks together the coefficients of correlated predictors like ridge. It has computational advantages.

2. Code & Results:

```
# Tune parameter α ∈ (0,1) with the lowest misclassification rate
> grid1 = 10^seq(3,-8,length = 50)
> mat = sparse.model.matrix(~.,data = training[,-15])
> grid2 = seq(0,1,by=0.1)
> correctness = rep(0,11)
> lambda = rep(0,11)
> for(j in grid2){
+   elasticnet.mod = glmnet(mat,training[,15],family = "binomial",alpha=j,lambda=grid1,
standardize = FALSE)
+   mat.test = sparse.model.matrix(~.,test[ ,-15])
+   LRpredict = predict(elasticnet.mod,newx=mat.test,type="response")
+   income.test = as.numeric(unclass(income.test))
+   LRpredict = ifelse(LRpredict<0.5,1,2)
+   score = rep(0,50)
+   for(i in 1:50) {
+     score[i] = 1-sum(LRpredict[,i]!=income.test)/length(income.test)
+   }
+   lambda[10*j+1] = grid1[which.max(score)]
+   correctness[10*j+1] = max(score)
+ }
> alphaa = grid2[which.max(correctness)]
```

Misclassification Rate Vs. Lambda index with the selected tuning parameter α=0.1:



3. Model Summary:

After adding penalty term for regression, the misclassification rate drops dramatically to 14.81%. Specifically, we use the combination of L1 and L2 penalty, which is called elastic net, with tuning parameters $\alpha = 0.1$ and $\lambda = 0.00018$.

The elastic net is a novel shrinkage and selection method, producing a sparse model with good prediction accuracy, encouraging a grouping effect.

1. Variable selection/regularization & Grouping effect:
Similar to the Lasso, the elastic net simultaneously does automatic variable selection and continuous shrinkage, and is able to select groups of correlated variables. However, it can deal with the situation when p>n, while Lasso can't. Elastic Net penalty is strictly convex (because of the quadratic term); Lasso is convex, but not strictly. Also, in Elastic Net, highly correlated predictors will have similar regression coefficients, which is called the grouping effect. In contrast, the Lasso process just randomly selects one variable among highly correlated ones.

2. Computation & Double Shrinkage:
An accurate penalization method achieves good prediction performance through the bias-variance trade-off. The elastic net estimator is a two-stage procedure: for each fixed α, we find the penalty parameter λ. It appears to incur a double amount of shrinkage, which does not help to reduce the variances much and introduces unnecessary extra bias, compared with pure Lasso or Ridge shrinkage.

Conclusion:
This report presents our results from three classification models: K-Nearest Neighbor, Naïve Bayesian Classifier, and Logistic Regression. The following table presents a summary of the misclassification rates for each model.

Summary of Misclassification Rates for Models

Model	Misclassification Rate
Naïve Bayesian Classifier	19.6%
k-NN	15.9%
Logistic Regression with Elastic Net Regularization	14.8%

Recommendations
Our team recommends using the *logistic regression model with elastic net regularization* because it is strong on two fronts – prediction and interpretability.
The coefficients of this model suggest that individuals with the following characteristics are more likely to have incomes over 50K:

- Married and with children
- Male (versus female)
- Higher levels of education
- Asian and white individuals
- Occupations such as farming/fishing, exec-managerial, specialty, tech-support

Moving forward, we also recommend exploring other models such as decisions trees, which may yield better classification rates than the models discuss above. However, we believe that the interpretability of the logistic regression model is best suited for purposes of targeting high income (>50K) individuals.