Classifer training to predict annual income

I. Approach

The project consists of a binary classification task to predict the annual income of individuals. There are several machine learning algorithms which complete this. In our approach we tested different algorithms to calculate a baseline score for the method and afterwards we chose the algorithm with the highest accuracy. We used stratified k fold cross-validation [11] procedure to evaluate model performance. Due to the high skewness of the data a stratified approach better represented the data selection to train the model.

The algorithms tested are:

- Support Vector Machine [4]
- Random Forest Classifier [3]
- Bagging Classifier [2]
- Gradient Boosting Classifier [6]

The algorithm with the highest score was gradient boosting with the cross validation score of 0.865 and standard deviation of (0.005)

A. Gradient Boosting Classification

Gradient Boosting Classification often called "Gradient Boosting Machines" is a classification framework based on weak learners, often decision trees which are optimised by a differential loss function.

The algorithm procedure starts with a weak learner (a decision tree) classifying data often at random, which is optimised using the loss function. The algorithm is an additive model where in an iterative approach the decision trees are added togerther to reduce and optimise the loss function.

In essence the algorithm can be split into three parts: Loss function, The weak learner (decision tree) making predictions, and an additive model to add steps to weak learners

The Algorithm Procedure can be described as follows:

The inputs for the algorithm: data-set for training, $\{(x_i, y_i)\}_{i=1}^{n}$ a loss function , $L(y, F(x))$

Step 1) Initialize model with a constant value:

$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$

Step 2) For m = 1 to M:

A) Calculate residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

B) Fit a regression tree for r_{im} values and create terminal regions R_{jm} for j=1...J_m

C) For j = 1...J_m calculate multiplier γ_m by solving:

$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma\right).$

D)Update the model:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma I(x \in R_{jm})$$

The aim of the algorithm is to find $\hat{F}(x)$ that approximates the annual income from the individual parameters. The loss function $L(y, F(x))$ is chosen to be a logarithmic loss function calculated using sklearn.ensemble.GradientBoostingClassifier. A decision tree is used to classify the target variable an in each step the multiplier parameter γ is calculated such that the loss function is minimised. This is done by calculating the residuals from predicted outcome and true outcome. The decision tree is added in each step until the algorithm is fully optimised [5] [9].

II. Methods

A. Data Preprocessing

The training data-set consists of 32,561 entries with parameters representing an individuals background: age,education,hours worked a week etc. The data is from [8] with the target variable being whether the individual makes below or above 50,000$.

The data-set consists of discrete and continuous parameters. The discrete parameters e.g marital status are given numerical labels from 0 to number of different classes using sklearn.preprocessing.LabelEncoder. Before moving unto the machine learning algorithm we analyse the data and note that the target variable is highly skewed. There are significantly lower number of individuals making >50k compared to number of people making <50k.

Class= <=50K	Count=24720	Percentage=76
Class= >50K	Count=7841	Percentage=24

Therefore the usual train-test split method would give inconsistent results depending on entries randomly chosen since the different sets would not have the ratio of target variables. Instead, we use a stratified approach using RepeatedStratifiedKFold scikit-learn. In this procedure with k=5 each fold will contain 6512 entries with equal ratios of target variables, hence approximately 76% of entries with <50k and 24% of entries with >50k. Cross validation score will be a baseline test of the model using sklearn.modelselection.crossvalscore. Using three different repeats we can give a score for the model and calculate the standard deviation of the error.

B. Algorithm Evaluation

The cross validation score was a baseline test for the different algorithms. We chose Gradient Boosting ma-

chine as our classifying method since it had the highest cross validation score. The cross validation scores of the different algorithms can be seen in table below

Algorithm	Mean accuracy	Std
Support Vector Machine	0.816	0.004
Random Forest	0.846	0.004
Bagging Classifier	0.850	0.004
Gradient Boosting	0.865	0.005

To evaluate the model we will train it on our training data-set and apply it to the testing data set. The performance of the classifier will be will be tested further by calculating the Jaccard Index Evaluation, Log loss evaluation and through the analysis of the confusion matrix and the F1 score

III. Results and Discussion

The Gradient Boosting Machine was trained on our training data-set and tested on the testing data-set made up of 16,281 entries.

A. Jaccard Index Evaluation

The Jaccard Index can be used to evaluate the classification model [7]. It is one of the simplest evaluation metrics which shows the intersection between the predicted labels and the observed label from the training date-set divided by the size of the union. It is defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

If all of the predicted labels were correct one would have the perfect score of 1 and a score of 0 if all labels were incorrect.

The Jaccard Score in our model was: 0.485 Which suggests that significant number of labels weren't predicted correctly. However, this score doesn't give insight on which class was incorrectly predicted.

B. Log Loss Evaluation

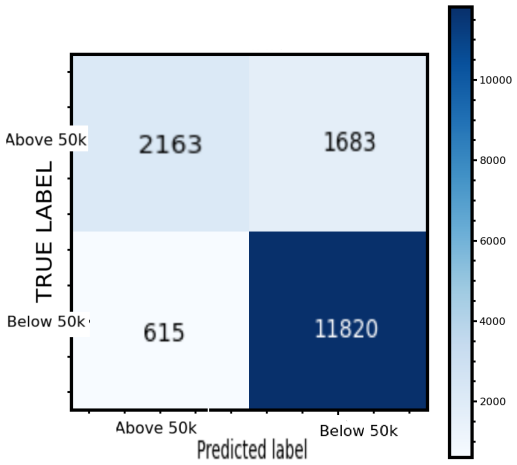
The Gradient Boosting Machine is able to give a probability for an output between 0 and 1 i.e if the model gives an probability for an individual as 0.97 it is very confident that the individual should be classified as having an income of >50k. Therefore the log loss evaluation can accurately tell if the confidence of a classifier (predicted probability) is far from the observed value [1]. Log loss is defined as:

$$LogLoss = -\frac{1}{n}\sum(y * log(\hat{y}) + (1-y) * log(1-\hat{y})$$

$\hat{y}$ = predicted probability y=observed value.

The LogLoss value of the model was 0.31 (0 perfect prediction) low enough, suggesting that the predicted probabilities were close to the output. This is interesting since the LogLoss score is much better than the Jaccard score, hence the model made a significant number of classification errors but the ones it did correctly it was very confident in doing so.

C. Confusion Matrix and F1 Score



The confusion matrix shows that the model can predict with very high accuracy entries which have income <50k (11820 were predicted correctly out of 12435). Meanwhile for entries with income >50k 2163 were predicted correctly out of 3846 (only 56% correct.) This can explain the low Jaccard score since quite significant number of entries were predicted incorrectly above 50k. The log loss score was low because the classifier very confidently predicted the individuals with income <50k.

This is evident in the breakdown of the F1 score [10]:

Class	Precision	Recall	F1	Support
<50k	0.88	0.95	0.91	12435
>50k	0.78	0.56	0.65	3846
Accuracy	-	-	0.86	16281

For <50k we have an F1 score of 0.65 which is quite poor. Overall the model has an average accuracy of 0.86

D. Limitations

Our model was chosen since it had the highest cross validation score. We have found that it has an average accuracy of 0.86 and yet it is only 56% accurate at predicting individuals with income >50k. However it is 95% accurate at predicting classes with income <50k. This could be attributed due to the high skewness of the data, since there aren't many instances of data points >50k. Therefore, our model was over trained in one class. This can further be explained in a more fundamental approach. Gradient Boosting works by minimising the loss function, the easiest path to do so is to confidently predict classes <50k, hence overfitting our model. In the future a much larger data-set should be used for training. Furthermore, different models should be applied and compared with the F1 score rather than cross validation. Depending on the task at hand a model should be chosen which more accurately predicts the needed class i.e. If we need to identify high income individuals Gradient Boosting would be a bad model even if it has the highest average accuracy.

[1] Bishop, C. M. Pattern recognition. *Machine learning 128*, 9 (2006).

[2] Breiman, L. Bagging predictors. *Mach. Learn. 24*, 2 (Aug. 1996), 123–140.

[3] Breiman, L. Machine learning, volume 45, number 1 - springerlink. *Machine Learning 45* (10 2001), 5–32.

[4] Chang, C.-C., and Lin, C.-J. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. 2*, 3 (May 2011).

[5] Friedman, J. Stochastic gradient boosting. *Computational Statistics Data Analysis 38* (02 2002), 367–378.

[6] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics 29*, 5 (2001), 1189 – 1232.

[7] Jaccard, P. The distribution of the flora in the alpine zone.1. *New Phytologist 11*, 2 (1912), 37–50.

[8] Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *KDD* (09 1997).

[9] Lu, Z.-q. The elements of statistical learning: Data mining, inference, and prediction, 2nd edn by t. hastie; r. tibshirani tibshirani; j. friedman. *Journal of the Royal Statistical Society. Series A (Statistics in Society) 173* (01 2010), 693–694.

[10] Powers, D. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Mach. Learn. Technol. 2* (01 2008).

[11] Vanwinckelen, G., and Blockeel, H. On estimating model accuracy with repeated cross-validation. De Baets, Bernard, pp. 39–44.