COMP 5318 Assignment 2

Tutors: Jiayan Qiu

Group Members: Yahia Abbas, 470506323, yabb6855
Chenxiao Cai, 470158649, ccai2201
Yichao Fu, 460283988, yifu6303

Abstract

The objective of this report is to document the application of Random Forest, XGBoost, and Support Vector Machine to perform binary classification on the Adult dataset. The Adult dataset is one of the most classic binary classifications with categorical problems and the insight obtained from this problem can be applied to a wide range of practical classification problems such as diabetes diagnosis, airline delay estimation and weather forecast. The results show the accuracy of all three classifiers which sit at around 85% and the performance on classifying people with higher income (>$50k per year) is relatively undesirable. The paper ends with the conclusion that the imbalanced data distribution is the main culprit that impacts the overall accuracy rate. Because of the nature of ensemble learning algorithm, random forest and XGBoost operate much faster than SVM while the adoption of z-score standardization boosts the accuracy rate of SVM by 10%. Moreover, age and number of education completed are the most determining factors for a person's income level.

1. Introduction

1.1 Problem statement

Classification with categorical variables is one of the mainstream classification problems that sparks numerous machine learning professionals' interests. This is because classification with categorical variables is of much value in many fields. For example, determining whether a flight is expected to experience delays or not by analyzing the relevant information such as time and city of departure, destination, weather condition and airline company, or find the possibility of a person being addicted to computer games given its age, gender, race, nationality and educational level. Apparently, there are considerable unseen patterns behind the categorical variables. Therefore, it is vitally important to apply a viable and efficient method to derive meaningful insights from the categorical variables for decision making and prediction purpose. The problem we want to solve in this paper is designing an efficient and effective classifier to find the most crucial factors that influence the level of income. The findings could be a valued asset for further social sciences studies in the near future.

1.2 Dataset

The dataset selected for this paper is the Adult data set, also known as "Census Income dataset". In total there are 48842 instances in the dataset, of which 24.78% earn more than $50k per year, in a sharp contrast to that of lower income (75.22%). The dataset collects census data across different nations to predict whether a person's income exceed the $50,000 per year based on 14 categorical variables including age, work class, fnlwgt, education, education-num, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, and native country. Each piece of data is listed on a single line of the data set and each variable is separated by ','.

1.3 High-level overview on classifier

The three classifiers used in this paper to study the Census Income dataset are random forest, xgboost and support vector machine (SVM). Random forest is a widely used ensemble machine learning method for classification. The method utilizes a predetermined number of decision trees to find the most likely results based on the prediction from each tree. Similarly, xgboost is another ensemble machine learning algorithm that enables scalable end to end tree boost system. In addition, xgboost achieves state-of-art performances on large scale datasets (Chen and Guestrin,2014). SVM is a supervised machine learning algorithm for tackling classification problems, especially for binary classification. SVM seeks to find an optimal decision boundary to separate two sets of data for binary classification.

This paper, therefore, is going to conduct literature review on the previous work that was successfully implemented in similar datasets. Then the paper will explain the pre-processing of the original Census Income data set and explore the various avenues of the theories behind random forest, xgboost and SVM respectively. The experiment and analysis session will focus on providing performance metrics on each classifier implemented such as accuracy, precision, recall, F1 score and confusion matrix, comparing among three different classifiers in terms of performance and efficiency. Lastly, the discussion on results and justification on design choice together with conclusion will also be considered.

2. Previous work

Antonov (2014) successfully applied decision tree algorithm and naïve Bayes classifier on the “Census Income dataset”. The results revealed that both decision tree and naïve Bayes achieved 83% accuracy. To dig deeper, naïve Bayes outperformed decision tree when it came to predicting high-income people (annual salary >$50k) while decision tree provided more accurate predictions on lower income divisions. Moreover, from the association study, he concluded that categorical variables including marital status, relationships, age, occupation and education were the most determining factors for predicting income level.

Pang (2017) adopted logistic regression to solve the binary classification problem from the “Census Income dataset”. In his experiment, 75% of the data was divided as training data and the remaining 25% became the test data for cross validation. The research used OneHotEncoder to convert all the categorical variables into numbers before fitting all the data to the logistic regression classifier. The accuracy rate of the experiment was 79%, with 85% recall rate on <$50k division and 77% recall on >= $50k class. Moreover, Pang failed to optimize the logistic regression algorithm on this data set after implementing polynomial features and PCA given the lower precision and F1 score yet higher recall.

In the same data set, Chockalingam, Shah and Shaw from the University of California, Diego, implemented a vast variety of machine learning algorithms including logistic regression, extra tree classifier, decision tree classifier, KNN classifier, SVM, gradient boost classifier, stepwise logistic regression, naïve Bayes, as well as ANN1 to ANN6. In the pre-processing phase, they removed education-num and native country given their insignificance on their discretions. The results of all the classifiers are provided in table 1.

Table 1: Accuracy for training set and test set, by classifier (by Chockalingam, Shah and Shaw)

Model	*Accuracy on Training Set*	*Accuracy on Test Set*
Logistic Regression	0.7908386	0.792619203
Extra Tree Classifier	0.7908386	0.821227956
Decision Tree Classifier	0.99996742	0.803396473
K - Nearest Neighbor Classifier	0.99996742	0.795754409
SVM	0.963966899	0.748856956
Gradient Boosting Classifier	0.864012511	0.862900065
Stepwise Logistic Regression	0.848	0.847
Naive Bayes	0.811	0.809
ANN 1	0.847	0.847
ANN 2	0.852	0.850
ANN 3	0.855	0.852
ANN 4	0.856	0.853
ANN 5	0.862	0.853
ANN 6	0.862	0.855

Like Chockalingam, Shah and Shaw's work, Topiwalla (2017) also tried a wide range of classifiers to attempt the binary classification problem on the "Census Income dataset". Topiwalla asserted that education-num and education were intrinsically correlated, so in the preprocessing stage, education-num was removed to speed up the training and classification process. The results of his experiments on different classifiers are listed below in table 2:

Table 2: Accuracy, precision and recall, by classifier (Topiwalla, 2017)

Model	Accuracy	Precision	Recall
Decision Tree	86.07%	90.25%	85.2%
Naïve Bayes	82%	N/A	N/A
Logistic Regression	85.16%	92.95%	88.22%
KNN	79.18%	95.81%	80.57%
SVM	85.52%	93.78%	87.97%
Random Forest	86.92%	94.26%	90.2%
Xgboost	87.53 %	94.62 %	89.62 %

3. Method

3.1 Random Forest

Breiman (2001) defines random forest algorithm as "an ensemble learning algorithm." In ensemble learning, there are two major algorithms, namely bagging algorithm and boosting algorithm. Random forest algorithm takes full advantages of bagging algorithm. The steps for random forest are provided below:

- For each decision tree (can be C4.5 or CART), apply bootstrap method to collect sample data from the dataset and pick a range of features for training purpose.
- The test data will then be processed by each trained decision tree. For each trained decision tree, the probability of each classification outcome will be calculated. Then the classification outcome with the highest probability becomes the prediction for a given trained decision tree.
- The final decision will be made by selecting the most popular decision among the decision trees in the forest. In other words, the prediction with the highest counts prevails and becomes the final prediction.

The chart below visualizes the basic process of random forest algorithm:

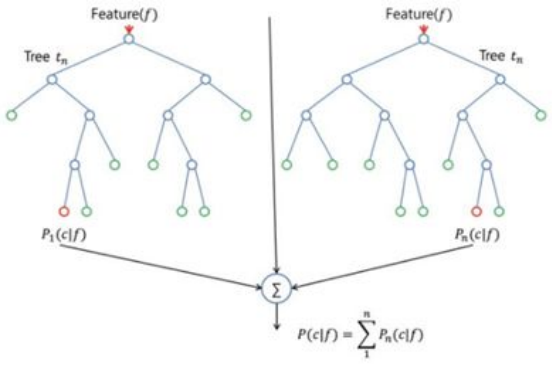

Figure 1: Decision making mechanism for random forest(elecfans,2018)

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X).$$

Figure 2: Nasic function for random forest prediction (Doan,2018)

The basic function is shown in Figure 2, where f(x) is defined as a linear prediction function of X. β_m is the weight assigned to the m^{th} transformation while h_m is the mth transformation of X
Furthermore, when to split the data in the decision tree is another crucial decision for random forest algorithm. This is decided by calculating impurity. The formula is given in Figure 3

$$I(A) = \phi[p(y = 1|A)]$$
$$\text{with } \phi \geq 0,\ \phi(p) = \phi(1-p),\ \text{and } \phi(0) = \phi(1) < \phi(p)$$

Figure 3: Impurity function(Doan,2018)

Empirically speaking, the higher the level of impurity, the less distinguishable for different subjects. Breiman (2001) mentioned the advantages of random forest algorithm are fast learning, high efficiency when dealing with big data and the ability to comfortably process thousands of variables in a large-scale dataset.

3.2 XGBoost

With reference to Chen and Guestrin (2016), XGBoost is a highly scalable end to end system that has been widely used by data science and machine learning professionals to achieve best of breed results for many big data problems. The XGBoost integrates with novel sparsity-aware algorithm, weighted quantile sketch and Parallel and distributed computation to make it possible for massive data processing in an efficient manner.

In addition, Chen and Guestrin (2016) figure out that unlike that of random forest algorithm, XGBoost utilizes a cluster of interdependent CART trees for prediction. The objective of XGBoost is to build up k CART trees to make accurate predictions that matches the correct answers as much as possible while achieve a high level of generalization. The mathematical function is provided in Figure 4.

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

Figure 4: Mathematical function for XGBoost(Chen and Guestrin ,2016)

The k indicates the number of CART trees, F stands for all possible CART trees, f refers to a particular CART tree.

Moreover, the objective function for XGBoost is provided in Figure 5.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$
$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Figure 5: Object function for XGBoost(Chen and Guestrin ,2016)

The object function comprises of two parts, the first half is the loss function and the second half is the regularization term that aims at mitigating the risk of overfitting.

Similar to loss function in logistic regression, the main objective for training the XGBoost is to find an optimal set of parameters in CART trees that minimize the value of loss function. In XGBoost, greedy algorithm is introduced to find optimal parameters. The basic logic for the greedy algorithm implemented in XGBoost is provided in Figure 6.

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Figure 6: Greedy algorithm used in XGBoost (Chen and Guestrin ,2016)

Finding parameters in CART trees is the same as seeking for a suitable point (feature) for splitting. This is determined by calculating the minimal value for a quadratic function in the objective function when the derivative equals 0. As the logic develops, when it comes to choose a node to split, the greedy algorithm only cares about the local optimal solution on the current node and it will greedy grow up to a tree and another tree until reaching a predetermined threat hold (maximum depth of CART tree). The final prediction is derived by considering all the predictions from each tree and selecting the class with highest score and minimum value in cost function.

3.3 Support Vector Machine Classifier

Support Vector Machine (SVM) is a supervised machine learning algorithm that has uses in both classification and regression problems. However, since it deals better with complex, small datasets, it is more commonly used in classification. This algorithm operates in a n-dimensional space, where n is the number of features we have, with each feature having the value of a certain coordinate. Classification is then performed by finding the hyperplane that best separates classes. When determining the correct hyper-plane, we must consider the hyper-plane that best segregates two classes, the one with the higher margin, and the possible addition of another feature. The margin mentioned earlier is the maximum distance between the nearest data point and the hyperplane. What's interesting about SVM is that it is robust to outliers, which means it will ignore outliers when deemed necessary and find the hyperplane with the highest margin. In the case where a linear hyperplane can not be determined, another feature is added in a way that helps segregate the other classes. This process is done through a technique called the kernel trick which performs complex data transformations to make non separable problems separable.

According to Savan Patel (2017), the polynomial kernel can be written as K(x,xi) = 1 + sum(x * xi)^d and exponential as K(x,xi) = exp(-gamma * sum((x — xi²)). Also mentioned by Jason Brownlee (2016), we can also have a more complex radial kernel. For example: K(x,xi) = exp(-gamma * sum((x – xi^2)), where gamma is a parameter that must be specified to the learning algorithm.

3.1.1 Pre-processing

The pre-processing mainly comprises of 4 stages. In the first stage, all the data will be converted to dataframe. The corresponding heading for each variable is also clearly defined and listed in this stage. Each data is presented in the following manner: age, work class, fnlwgt, education, education-num, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, native country and income. For this dataset, we do not intend to subjectively remove any

“irrelevant” variables to reduce training expense because we assume that all variables listed above impact the level of income.

In both training set and test data, some incomplete records have been detected with “?” inserted to fill the blanks. We decide to remove all incomplete records with “?” so that the trained model is free from incomplete data. After the cleaning, there are 45222 data left, with 30162 training data and 15060 test data. The proportion for income level for >50k now becomes 24.78% while the one for <50k is 75.22%.

In the third step is to reconcile the representation inconsistency of income in training data and test data. The income figure in the training set is described either as ‘<50k’ or ‘>50k’ while those in test set is stated as ‘>50k.’ or ‘<50k. ‘。This inconsistency will significantly degrade the performance for all classifiers because instead of having 2 prediction class in place, there are actually 4 possible prediction outcomes if the inconsistency issue remain unsolved. As the argument goes, the classifier trained based on training set will never learn to predict income level of ‘>50k.’ or ‘<50k.’ because they are only trained to predict ‘<50k’ or ‘>50k’. Therefore, in our implementation, all the income data with ‘>50k.’ or ‘<50k.’ is replaced with ‘<50k’ or ‘>50k’. to achieve data consistency.

In the final stage of pre-processing, all the categorical variables are converted to numbers and then standardized as z-score. 8 of 14 variables in the data set are classified as categorical variables including work class, education, marital status, occupation, relationship, race, sex and native country. Standardization is a crucial stage in pro-processing because text cannot be converted to matrix for machine learning. In addition, z-score standardization will transform the encoded categorical variables to a Gaussian distribution with 0 average and 1 variance. One motivation for standardization is to boost the efficiency of each classifier, especially during the convergence process. Some sample outcomes after being processed by standardization are provided in Figure 7.

```
array([[ 0.03067056,  2.15057856, -1.06361075, ..., -0.21665953,
        -0.03542945,  0.29156857],
       [ 0.83710898,  1.46373585, -1.008707  , ..., -0.21665953,
        -2.22215312,  0.29156857],
       [-0.04264203,  0.09005041,  0.2450785 , ..., -0.21665953,
        -0.03542945,  0.29156857],
       ...,
       [ 1.42360965,  0.09005041, -0.35877741, ..., -0.21665953,
        -0.03542945,  0.29156857],
       [-1.21564337,  0.09005041,  0.11095988, ..., -0.21665953,
        -1.65522476,  0.29156857],
       [ 0.98373415,  0.77689313,  0.92989258, ..., -0.21665953,
        -0.03542945,  0.29156857]])
```

Figure 7: Sample pre-processing result

Discussion on choice of decision

3.1.2 SVM

SVM is mainly used for binary classification on relatively small-scale data set. Therefore, we strongly believe SVM is a good fit for “Census Income dataset” because it is a binary classification problem with only 45222 data (predict whether a person’s income level is greater than $50k or not). In addition, SVM uses kernel function to avoid complexity issues in high-dimension space and provide direct solutions to classification problems under high-dimension scenario. Therefore, we decide to implement SVM on this dataset. Implementing SVM is memory efficient because it uses a subset of

training points in the decision function. However, the downside of using SVM is that it does not provide probability estimates nor does it work well when datasets are very noisy.

3.1.3 Random Forest

Random forest is a very robust machine learning algorithm and it is widely implemented in a vast variety of datasets. Moreover, the training process in random forest algorithm uses bootstrap method to select a portion of data for training so that it properly mitigates the risk of overfitting. Moreover, random forest can be efficiently trained within a reasonable time. Another important factor which made us choose random forest in this dataset is that random forest can determine the level of importance of each variable in relation to income. The ability to weigh features not only provides a reasonable assurance on accuracy, but also serves as a valuable asset for us to provide further analysis on the relationship between different features and income. All these reasons explain the use of random forest in this dataset.

3.1.4 XGBoost

We are very impressed by the paper on XGBoost written by Tianqi Chen and so we eventually decide to implement XGBoost to embrace the advanced features embedded in this emerging machine learning technique, for example regularization, novel sparsity-aware algorithm, weighted quantile sketch, parallel and distributed computation, shrinkage and column subsampling, split finding algorithms and weighted quantile sketch. Like random forest, XGBoost can list the feature importance ranking which is the cornerstone for CART trees to find the optimal point to split. We strongly believe XGBoost is favored for this dataset because intuitively the magnitude of impact on level of income significantly differs from one feature to another. Therefore, finding the top determining factors could be the key to achieve a state of art result on this dataset.

4. Experiment

Environment, hardware and software specification:

Operating System: macOS High Sierra Version 10.13.6
Processer: 1.3 Ghz Intel Core i5
Memory: 8GB 1867MHz LPDDR3
Software: jupyter 4.4.0 and python 3.6

4.1 Analysis and comaprision on efficiency and performance

The execution time for 10-fold cross validation for each classifier is provided in table 3:

Table 3: Execution time summary

Classifier	Execution time
Random Forest	124.96 second
XGBoost	3.2 second
SVM	873.16 second

From table 3 it turns out that SVM requires far more time than random forest and XGBoost. This is because the empirically SVM is not a good fit for large scale dataset given its high computation

expense. At the beginning, SVM converts all the training data to landmarks and calculate the corresponding feature vectors. When the scale of dataset becomes larger and larger, more landmarks are required. So when finding the parameter θ that minimizes the loss function, the dimension of θ and the number of landmarks can be substantial, leading to a longer execution time.

By contrast, random forest and XGBoost are both ensemble learning algorithms that fully supports parallel processing (based on number of CPU core) with embedded normalization. For XGBoost, when looking for an optimal splitting point, it uses an approximation algorithm to use a percentile method that selects the best candidate splitting point. In addition, the use of novel sparsity-aware algorithm boosts the algorithm efficiency. Furthermore, XGBoost utilizes multi-threading, data compression and slicing to further cut the operation time. Lastly, the most time consuming phase for decision tree-based algorithm is to rank all the features. This, however, is not a big concern for XGBoost, since XGBoost pre-orders the feature ranking and saves in a block structure that will be used in every iteration. The block structure also supports the calculation of gains for each feature on a parallel manner.

The performance metrics after 10-fold cross validation for each classifier are provided in table4:

Table 4: Results on 10-fold cross validation

		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Accuracy	Random Forest	0.84	0.85	0.85	0.84	0.84	0.85	0.85	0.86	0.86	0.85	0.849	1
	XGBoost	0.82	0.83	0.85	0.83	0.85	0.85	0.84	0.85	0.85	0.84	0.841	2
	SVM	0.83	0.84	0.85	0.83	0.84	0.84	0.84	0.85	0.84	0.83	0.839	3
		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Precision	Random Forest	0.84	0.85	0.85	0.84	0.84	0.85	0.85	0.86	0.86	0.85	0.849	1
	XGBoost	0.82	0.83	0.84	0.83	0.85	0.84	0.84	0.85	0.84	0.84	0.838	2
	SVM	0.82	0.84	0.84	0.83	0.84	0.84	0.83	0.85	0.84	0.83	0.836	3
		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Recall	Random Forest	0.84	0.86	0.85	0.85	0.85	0.86	0.86	0.87	0.86	0.85	0.855	1
	XGBoost	0.83	0.84	0.85	0.84	0.85	0.85	0.85	0.86	0.85	0.84	0.846	2
	SVM	0.83	0.85	0.85	0.84	0.84	0.85	0.84	0.85	0.85	0.84	0.844	3
		1	2	3	4	5	6	7	8	9	10	Avg	Rank
F1	Random Forest	0.84	0.85	0.85	0.84	0.85	0.85	0.85	0.86	0.86	0.85	0.850	1
	XGBoost	0.81	0.82	0.84	0.82	0.84	0.84	0.83	0.84	0.84	0.83	0.831	3
	SVM	0.82	0.84	0.84	0.83	0.84	0.84	0.83	0.85	0.84	0.83	0.836	2

The results on 10-fold cross validation shows the accuracy, precision, recall and F1 for the three classifiers are considerably close, ranging from 83% to 85%. So no machine learning algorithm far outperforms others in this dataset. For accuracy, it is random forest that stands out (84.9%), followed by XGBoost(84.1%) and SVM(83.9%) . The rankings for average precision and recall follow the same pattern as the ranking for accuracy.

Observation from average accuracy, precision, recall and f1 seems to provide limited insights of the three models' performance. Class level results on 10-fold cross validation for each classifier is presented in table 5:

Table 5: Class level precison and recall for the chosen classifier

		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Precision(<=$50k)	Random Forest	0.87	0.89	0.89	0.88	0.88	0.89	0.88	0.89	0.89	0.88	0.884	1
	XGBoost	0.84	0.85	0.87	0.85	0.86	0.87	0.86	0.86	0.86	0.86	0.858	3
	SVM	0.85	0.87	0.87	0.86	0.87	0.87	0.86	0.87	0.86	0.86	0.864	2
		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Recall(<=$50k)	Random Forest	0.92	0.93	0.92	0.93	0.92	0.92	0.93	0.94	0.95	0.92	0.928	3
	XGBoost	0.94	0.95	0.95	0.95	0.96	0.95	0.95	0.96	0.95	0.94	0.950	1
	SVM	0.93	0.94	0.94	0.94	0.94	0.94	0.94	0.95	0.94	0.93	0.939	2
		1	2	3	4	5	6	7	8	9	10	5.500	
Precision(>$50k)	Random Forest	0.74	0.74	0.71	0.74	0.73	0.73	0.75	0.78	0.77	0.74	0.743	3
	XGBoost	0.75	0.76	0.76	0.77	0.81	0.77	0.78	0.84	0.79	0.77	0.780	1
	SVM	0.74	0.75	0.74	0.75	0.75	0.75	0.75	0.8	0.77	0.74	0.754	2
		1	2	3	4	5	6	7	8	9	10	Avg	Rank
Recall(>=$50k)	Random Forest	0.62	0.62	0.61	0.62	0.62	0.65	0.62	0.66	0.64	0.65	0.631	1
	XGBoost	0.5	0.47	0.54	0.5	0.54	0.55	0.52	0.54	0.56	0.54	0.526	3
	SVM	0.53	0.55	0.56	0.53	0.57	0.56	0.53	0.57	0.55	0.56	0.551	2

The precision and recall for lower income class (<=$50k) is significantly high, in a sharp contrast to those performance metrics in higher income class. Noticeably, the precision for >$50k class is on average 10% less than the one for <=$50k. When it comes to recall on higher income class, random forest far outperforms XGBoost and SVM by 11% and 8% respectively. Based on the pattern on class level precision and recall table, it can be concluded that all the three classifiers are well trained to predict lower income class (<= $50k) but have difficulties in classifying higher income class (>$50k)

The performance metrics on test data is provided in table 6,7 and 8.

Table 6: Average accuracy, precision, recall and F1 on test data

Random Forest	Accuracy	0.85
	Precision	0.84
	Recall	0.85
	F1	0.84
XGBoost	Accuracy	0.84
	Precision	0.83
	Recall	0.84
	F1	0.83
SVM	Accuracy	0.85
	Precision	0.84
	Recall	0.85
	F1	0.84

Table 7: Confusion Matrix on test data

			Predicted Class		total acutal
Random Forest			<=$50k	>$50k	
	Actual Class	<=$50k	10495	865	11360
		>$50k	1428	2272	3700
	Total Predicted		11923	3137	15060
			Predicted Class		total acutal
XGBoost			<=$50k	>$50k	
	Actual Class	<=$50k	10787	573	11360
		>$50k	1807	1893	3700
	Total Predicted		12594	2466	15060
			Predicted Class		total acutal
SVM			<=$50k	>$50k	
	Actual Class	<=$50k	10695	665	11360
		>$50k	1667	2033	3700
	Total Predicted		12362	2698	15060

Table 8: Class-level precision, recall and F1 on test data

		Precision	Recall	F1	Total
Random Forest	<=$50k	0.88	0.92	0.9	11360
	>$50k	0.72	0.61	0.66	3700
		Precision	Recall	F1	Total
XGBoost	<=$50k	0.86	0.95	0.9	11360
	>$50k	0.77	0.51	0.61	3700
		Precision	Recall	F1	Total
SVM	<=$50k	0.87	0.94	0.9	11360
	>$50k	0.75	0.55	0.64	3700

The results on test data shows similar pattern when comparing to cross-validation results, indicating the robustness of the trained models. Overall, the accuracy is between 84% and 85% for all three classifiers. While all three classifiers achieves nearly average 90% precision and 95% recall in <=$50k class, the precision and recall in >$50k is undesirable. Especially for XGBoost and SVM, the recall rate in class >$50k is merely 51% and 55%, significantly compromising the accuracy rate.

4.2 Analysis and comparison of algorithms

The uneven distribution of income class in the dataset ought to be one of the main culprits for the underpermance in high income division classification. In the pre-processing session, it is mentioned that after removing all uncompleted records in the original dataset, 75.22% of the data is <=$50k while

the remaining 24.78% belongs to higher income level. Therefore, the low income-tailed data distribution may not serve as a solid training dataset to train the three classifiers because a relatively small number of training data featuring high income pattern is fit to the models during the training phase. As the logic develops, the three classifiers are highly likely to fail to capture the main features for higher income level and therefore provide sub-optimal predictions.

Specifically, the uneven data distribution could significantly hurt the precision and recall rate for higher income division when implementing random forest. This is due to the use of bootstrap sampling method in training phase under random forest. The bootstrap sampling randomly pick a set of training data and import them into a single tree. So if the ratio for higher income division and lower income division is 1:3, it is plausible that for some trees, only data for lower income division is selected for training, resulting in severe underfitting. Thus, those trees not receiving higher income data during the training will never able to tell a person with a high level of income.

One plausible reason why XGBoost's recall rate for higher income level is significantly less than that of random forest may directly attribute to the different mechanisms between these two classifiers. Since the prediction produced by random forest comes from a set of independent decision trees while the final decision for XGBoost relies on the joint efforts by interdependent CART trees. As the argument goes, the wrong decision made by those not-well-trained decision trees may pose no threat on the final decision making given that the majority of the decision trees vote for the opposite prediction. By contrast, the tree building process for XGBoost is starkly different. The essence of XGboost is an ever-devisifying CART tree . For each iteration, a new CART tree will be built based on all training data. This means in each round, the lower income-tailed data will be used to construct an optimal CART tree that minimal value of loss function can be obtained. Therefore, XGboost seems to be far less robust than random forest in this unevenly-distributed dataset because the uneven distribution nature tends to degrade the performance of XGBoost more.

SVM, by contrast, is not very sensitive to imbalance data distribution because only support vector impacts the hyperplane. During the experiment, it turns out that the robustness of SVM model is significantly influenced by standardization. This finding is supported by the observation that the accuracy sees a 10% boost when standardization is applied. Table 9 shows the results of SVM model before/after standardization is adopted.

Table 9: Comparision on SVM performance without/with standardization

	Precision	Recall	F1	Total
<=$50k	0.76	1	0.86	11360
>$50k	0.61	0.01	0.01	3700
avg	0.72	0.75	0.65	15060
Accuracy	75.48%			

		Predicted Class		total acutal
		<=$50k	>$50k	
Actual Class	<=$50k	11348	12	11360
	>$50k	3681	19	3700
Total Predicted		15029	31	15060

	Precision	Recall	F1	Total
<=$50k	0.87	0.94	0.9	11360
>$50k	0.75	0.55	0.64	3700
avg	0.84	0.85	0.84	15060
Accuracy	84.50%			

		Predicted Class		total acutal
		<=$50k	>$50k	
Actual Class	<=$50k	10695	665	11360
	>$50k	1667	2033	3700
Total Predicted		12362	2698	15060

It can be inferred that when standardization is absent, the accuracy for SVM is only 75.48%, with merely 1% recall and 1% precision for higher income division. From the confusion matrix, it indicates the trained SVM only succeeds in classifying 19 out of 3700 samples with >$50k so the training fails to train the SVM well. Comparatively, when standardization is in place, the overall accuracy witness a 10% increase, which is mainly attributed to a 54% and 63% boost in higher income division precision

and recall respectively. Implementing z-score standardization is vitally important to achieve a desirable result for similarity/distance-based machine learning algorithm. Because in SVM, if the variance of a feature is significantly than others, the model is inclined to make biased prediction merely based on that high variance feature instead of taking all features into consideration. By implementing z-score standardization, all features will be equally weighted given their predetermined variances of 1.

```
AUC Score : 0.887821
```

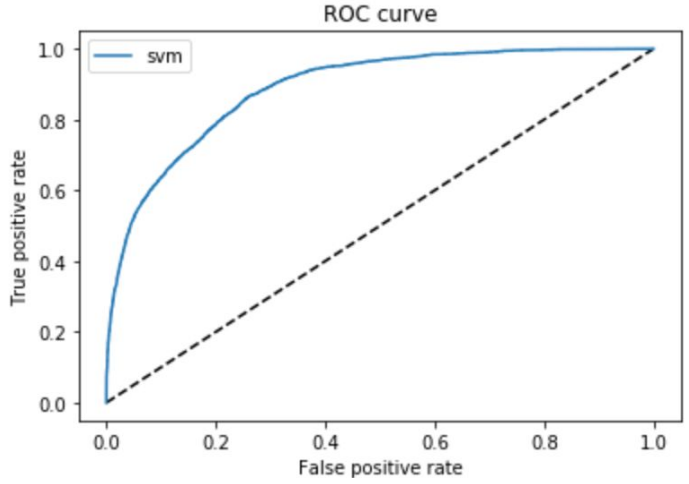

Figure 8: ROC curve for SVM

```
AUC Score : 0.913528
```

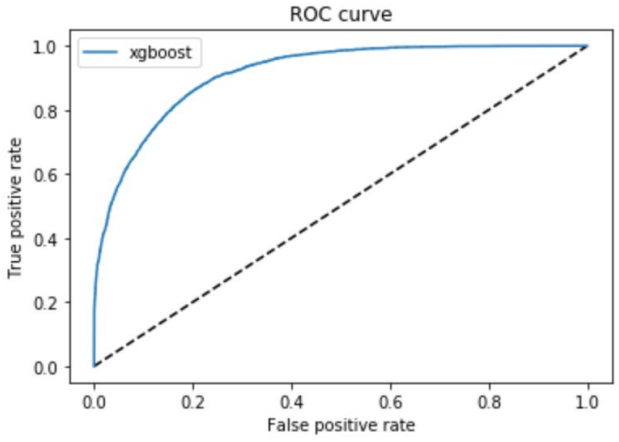

Figure 9: ROC curve for XGBoost

```
AUC Score : 0.901905
```

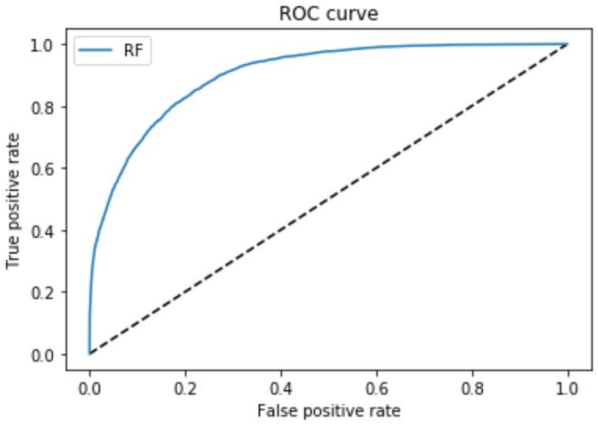

Figure 10: ROC curve for Random Forest

Figure 8,9 and 10 shows ROC curve and AUC score for SVM, XGBoost and Random Forest respectively. The AUC results denote that all three classifiers are well trained to fit the dataset and make highly reliable predictions on the test set given the AUC score ranging from 0.89 to 0.91. This means that on average, for each randomly chosen data in the test set, the possibility of placing a positive sample ranks higher than a negative one is nearly 90%.

From all the three ROC curves, it can be concluded that all three classifiers have a similar curve shape. The highly upper left corner-tailed shape means that all three models provides reliable predictions on classifying people whose annual income is smaller than $50k at the expense of incorrectly put some portions of people with higher income level to lower income division. The false positive rate, however, is relatively low when comparing to the true negative rate. Since the ROC curves only chart the relationship between false positive rate and true positive rate, the main issues identified in these three classifiers are masked. In the previous session, we found that the main sources of errors stems from low recall rate on higher income level (true negative rate), which is out of the scope of the ROC curve.

4.3 Analysis on dataset patterns and insights

Since SVM works differently than Random Forest and XGBoost, it is not able to produce output representing it's sorting of feature importance. With Random Forest and XGBoost, we obtained two tables that show different results for the classification of feature importance.

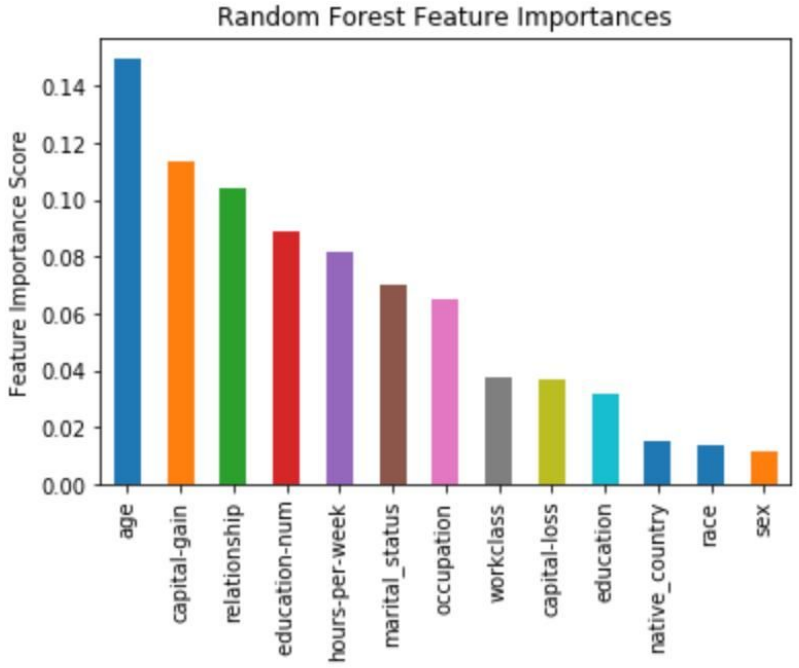

Figure 11: Feature importance ranking by Random Forest

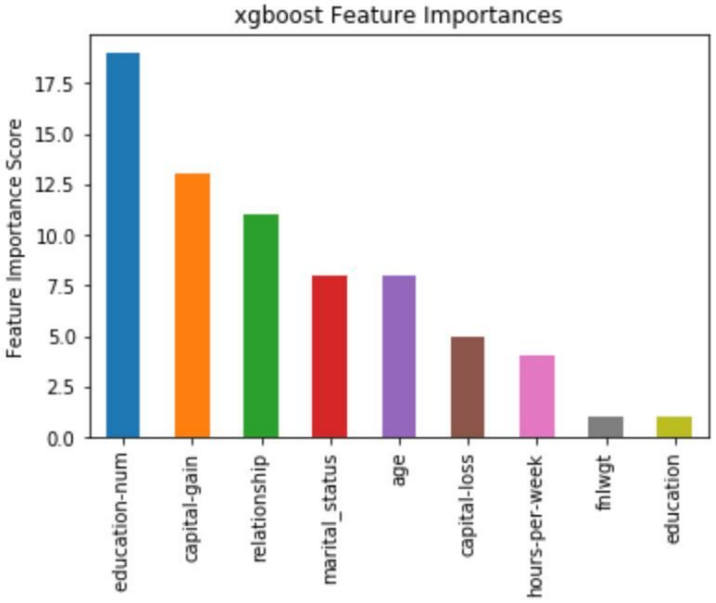

Figure 12: Feature importance ranking by XGBoost

We can already spot disagreements between the two tables from the top three important features in each. For Random Forest, the most important feature is age, followed by capital gain, and then relationship. However, for xgboost, Education number comes first, followed by capital gain, and then relationship. To justify Random Forest classification of age as the most important feature, we have to consider that the majority of the samples were collected from within the United States, so a certain pattern may be apparent. Getting the first job at a certain age, getting the first promotion after a certain time at the job, gathering enough experience to apply for a better job, etc.. As for capital gain, investments and properties undeniably have a huge impact on a person's income, and the better they are managed, the more income they will be able to get. Now relationship comes in the third rank for

both algorithms, which is plausible since the effect of relationship status on income could be somewhat uniform. Having a spouse or a child or someone to take care of means there is a need for a better job, or more jobs; hence more income. Being alone or in someone's care means there is not much need to stress over income. Now to go back to the most important feature on XGBoost's list, education number. Education number means the qualifications of the person, which directly affect his job opportunities and income. Usually, the better qualifications a person has, the better his chance at finding a better job or getting a higher salary. It makes sense that this feature would be the most important for this algorithm.
The bottom three rankings on Random Forest's table are native country, race, and sex.
The nationality distributions of collected samples:

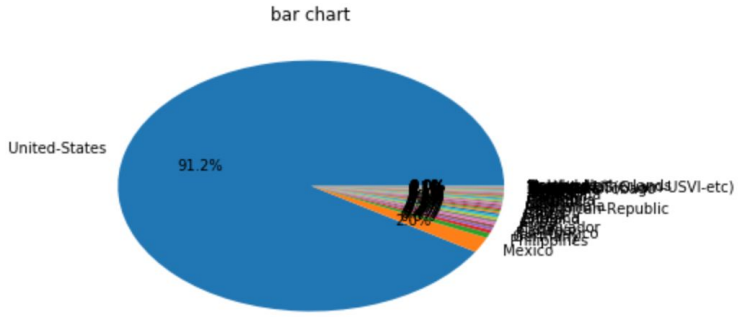

Figure 13: Native country breakdown on dataset

The native country feature can be discarded due to the majority of samples being from within the United States. Also race and sex should be irrelevant due to the laws against discrimination and sexism. As for XGBoost, the least important features are hours per week, fnlwgt, and education. Hours per week is ranked low probably because it is dependant on the hourly wage which widely varies. As for education, experience and knowledge is usually more valued than level of education.

In order to justify the rationale behind the feature ranking provided by random forest and XGBoost, further analysis is performed on age and education num is performed.

Table 10: statistics on age and education num on training set

Age Statistics	>=$50k	<$50k
Max	90	90
Min	21	17
Avg	43.96	36
Median	43	23
SD	5.12	14.67
Edu-num Statistics	>=$50k	<$50k
Avg	11.64	9.6
Median	12	9
SD	1.18	2.1

From the training set, it can be referred from table 10 that both ages and number of education indeed plays an important role on people's income level. This is justified by the statistics in table 9 that people from high income division tends to cluster around age of 43 with standard deviation of 5.12. For lower income division, the average age is 36 with standard deviation of 14.67. The 8 and 20 years gap in

average age and median age indicates that age can potentially one of the most determining factors to impact people's income level. It is plausible for random forest classifier to rank age as the number one feature when constructing the decision tree in the forest. Possibility, the root node is based on age and the splitting point is set at around the average or median age of people from high income division.

Similarly, education num potentially can be an effective benchmark for XGBoost. Given that for high income people, the number of education achieved is on average 2 greater than those earn less than $50k per year. From our perspective, education num is a far more robust indicator for higher income because the standard deviation for high income division is only 1.18, indicating for high income people, the number of education completed is closely clustered between 11 and 12, which is on average 2 more than people from low income division. Therefore, we predict that for XGBoost, the root node is based on education num. Moreover, one potential optimal splitting point at the early decision phase can be assessing whether a person's number of education reaches a threshold of 12 or not.

5. Conclusion and future work

5.1 Conclusion

After implementing random forest, XGBoost and SVM, the accuracy is around 85% for all classifiers. It also draws the fact that all three classifiers perform an impressive job in classifying people whose annual income is smaller than $50k while face difficulty in recognizing people from high income division. The main reason is unevenly distributed date given that 75% of the data is collected from lower income division. In terms of efficiency, as ensemble learning algorithm, both random forest and XGBoost runs significantly faster than SVM. This is due to the inherent difference between ensemble learning algorithm and SVM. When it comes to feature importance, by summarising and studying the statistics on the dataset, age and number of education completed are justified to be the most crucial factors to determine a person's income level.

5.2 Future Work

Looking at the results given from the experiments with the difference techniques, it is apparent that there is room for much improvement. Most of the work will be focused on fine-tuning and optimizing the techniques used so they provide better and faster results. When looking at SVM, we plan to optimize the technique in a way that puts more weight to features. In addition, more experiments will be performed to find the optimal set of parameters such as C, gamma and kernel function. Moreover, SVM ought to be redesigned in a way that supports multi-threading in order to cut the training time. For Random Forest, the calculation of probabilities will undergo further experimentation and trial to give better results. As for XGBoost, the parameters it needs will be looked over more thoroughly and ensured to be pristine. In terms of pre-processing, some of the irrelevant features can be removed such as sex, nationality and fnlwgt given their little weight.

References

[1] Chen.,T., Guestrin, C.(2014). XGBoost: A Scalable Tree Boosting System. Retrieved from http://delivery.acm.org/10.1145/2940000/2939785/p785-chen.pdf?ip=129.78.56.194&id=2939785&acc=CHORUS&key=65D80644F295BC0D%2E87D9DBB8EF508751%2E4D4702B0C3E38B35%2E6D218144511F3437&__acm__=1540774695_5fab4295451cff60d9ebd53ca2830aaa

[2]Antonov, A. A. (2014, March 30). Classification and association rules for census income data. Retrieved from https://mathematicaforprediction.wordpress.com/2014/03/30/classification-and-association-rules-for-census-income-data/

[3] Y. P. (2017). Analysis of the Adult data set from UCI Machine Learning Repository. Retrieved from https://yanhan.github.io/posts/2017-02-15-analysis-of-the-adult-data-set-from-uci-machine-learning-repository.ipynb.html

[4] Chockalingam V., Shah S., Shaw R. (n.d.). Income classification using adult census data. Retrieved from:https://pdfs.semanticscholar.org/3dd5/e9f335511efbb81d65f1d6d4995019f8b5fd.pdf

[5] Topiwalla M. (2017). Machine learning on UCI adult data set using various classifier algorithms and scaling up the accuracy using extreme Gradient boosting. Retrieved from: https://datascience52.files.wordpress.com/2017/02/machine-learning-on-uci-adult-data-set-using-various-classifier-algorithms-and-scaling-up-the-accuracy-using-extreme-gradient-boosting.pdf

[6] Breiman L. (2001). Random Forests (R. E. Schapire, Ed.). Retrieved from https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf

[7] Elecfans. (2018). Theory behind random forest algorithm. Retrieved from http://www.elecfans.com/d/647463.html

[8] Quoc, A. A., Doan. (n.d.). INTRO TO RANDOM FOREST. Retrieved from https://web.csulb.edu/~tebert/teaching/lectures/551/random_forest.pdf

[9] Patel, S. (2017, May 03). Chapter 2 : SVM (Support Vector Machine) - Theory – Machine Learning 101 – Medium. Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[10] Ray, S. (2015, October 6). Understanding Support Vector Machine algorithm from examples (along with code). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Appendix

Hyperlink to the dataset: https://archive.ics.uci.edu/ml/datasets/adult
External Library used:

1. xgboost 0.7.2
2. numpy 1.9.3
3. pandas 0.9.1
4. matplotlib3.0.0

Code execution instruction

- first please make sure that all the libraries mentioned above are installed and updated
- Make sure that the train data and test data are placed in the same folder with the code
- The code named SVM, Forest and newxgoost are the three classifiers we develop for this assignment, in order to run it, go to jupyter and find the corresponding folders storing these three files. Execute it in jupyter and it should work cell by cell.
- Each file is expected to provide results on 10-fold cross validation (accuracy, precision, recall and f1 score, classification report, confusion matrix), results on test data (same as for 10-fold cross validation), ROC curve together with AUc value. For XGBoost and forest, bar chart on variable significance ranking is plotted.

Contribution : we three equally contribute to the delivery of the assignment.