

# Empirical Evaluation of Supervised Learning Techniques Using Adult Data Set

Beulah Shantini  
baug8178@uni.sydney.edu.au  
470162015

Emil Laurence Pastor  
epas6415@uni.sydney.edu.au  
460466466

Maureen Rocas  
mroc9684@uni.sydney.edu.au  
470313536

The University of Sydney

## Abstract

In this study, we present an empirical evaluation of supervised learning techniques in order to predict whether the income of a person exceeds \$50K per year. Also, this can be applied in the area of marketing analytics to target the customer with the right marketing collateral. Moreover, the paper used different pre-processing techniques like one hot encoding, data binning, feature elimination, handling imbalanced and PCA. Subsequently, we used six supervised learning techniques: Ada Boost Decision Tree, Bagging Classifier, Gradient Boost, Logistic Regression, SVM and XG Boost. The results are evaluated using different performance criteria that lead to AdaBoost having the optimal result between performance and speed closely followed by XGBoost and Gradient Boost whilst Logistic Regression gained significant improvement after model tuning.

## I. Introduction

In machine learning, supervised learning is one of the most extensively used machine learning task that utilises historical data, learn and provide prediction based on a set of inputs. This domain of machine learning is useful in various domains of our society like finance, social services, medicine and computer vision. Having said that, the extent of its application is practically limitless that new algorithms and techniques has been developed to further enhance the overall classification performance.

In this study, the aim is to predict whether the income of a person exceeds \$50K per year based on the adult data set with 48,842 records and 14 features is obtained from University of California Irvine (UCI) Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/adult>). The data consists of 14 features and a label that has two values:  $\leq 50K$  and  $> 50K$ . In addition, the ratio of test to

training data provided 1:2 while the distribution of the labels has ratio of 1:3 as shown in *Figure 1 Data Set Label Distribution* which may require further processing since it is an imbalanced data. The resulting model is useful in the financial services, telecommunication and retail domain since they will have the opportunity to offer targeted marketing campaign based on the predicted income of an existing or potential customer. Also, non-profit organisations will also benefit since the model can aid them to determine the right person to ask for charity donations.

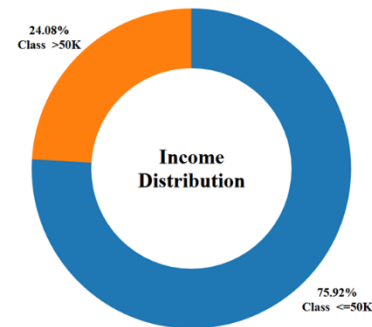


Figure 1 Data Set Label Distribution

This study used pre-processing techniques one hot encoding, data binning, feature elimination, handling imbalanced data and PCA to prepare the data prior to model fitting. Afterwards, six supervised learning techniques is used namely, Ada Boost Decision Tree, Bagging Classifier, Gradient Boost, Logistic Regression, SVM and XG Boost. The results are evaluated using different performance criteria like accuracy, precision, recall, f1-score and. Matthews Correlation Coefficient, execution time supported by Receiver Operating Characteristic (ROC) Curve and Precision-Recall Curve.

## II. Related Work

The data set is used by several paper one of which is from Ron Kohavi's "Scaling up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid" which is a comparison between Naïve Bayes Tree (NBTree) with

C4.5 Decision Tree induction algorithm and Naive Bayes algorithm. NBTree is a hybrid Decision Tree, which utilizes the advantages of both decision-trees (i.e. segmentation) and Naive-Bayes (evidence accumulation from multiple attributes) [1]. Kohavi's experiments show that C4.5 performs better on the Adult dataset compared to Naive-Bayes. His comparison shows that as more training instances were added, the accuracy increases. The highest accuracy score for adult data is by C4.5 at around 86.5% using more than 45,000 training instances while the lowest is by Naive Bayes at around 82.5%.

In addition, Faith Kaya in the paper “Discretizing Continuous Features for Naive Bayes and C4.5 Classifiers [2]” where discretization method results in greater improvements in the classification performance of NaiveBayes as compared to the C4.5 classifier. The paper put emphasis that discretization of continuous attributes is both a requirement and a way of performance improvement for many machine learning algorithms.

Similarly, Mark Hoffman used the adult data set in his paper “Predicting earning potential on Adult Dataset [3]”. This paper follows the CRISP-DM methodology used several pre-processing techniques like feature elimination, feature reduction like SVD and PCA and data binning to simplify the data set. Additionally, it utilise NBTree, Naïve Bayes and k-Nearest Neighbour (kNN) as the modeling algorithm where NBTree had the highest accuracy with 85.93% followed by Naïve Bayes with 85.33% and kNN with 84.92%. Overall, the paper states that the modelling work on the training dataset appears to indicate that there is a classifier accuracy limit of just below 87% similar to the work of Kaya and Kohavi.

### III. Methods

#### 1. High Level Methodology

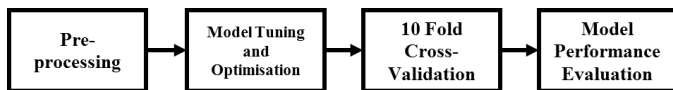


Figure 2 High Level Methodology

The methods of development and evaluation is shown in *Figure 2 High Level Methodology* where data is pre-processed using several techniques, model is tuned and optimised before performing overall performance evaluation. Furthermore, each stage of the development evaluates whether or not it is worthwhile to use the

technique or resort to other options. The succeeding section discusses the details of the different techniques used to develop the classification models.

## 2. Pre-processing Techniques

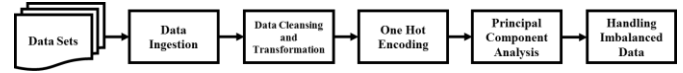


Figure 3 Pre-processing Techniques

### 2.1. Data Ingestion

Since the data set is relatively small, the data set is ingested in AWS PostgreSQL instance so that data is available everywhere and relatively easier to perform data manipulation using SQL. In addition, data loaded in the staging area has VARCHAR data type.

### 2.2. Data Cleansing and Transformation

The raw data loaded in AWS PostgreSQL DB needs to be cleansed to ensure that unwanted data is eliminated or transform in more meaningful way. Around 6% of the data has features like work class, occupation and native country that contain NULL values in both training and test data. Having said that, the effect of NULL values are evaluated whether those records should be filtered.

Data type conversion is performed to convert features into its appropriate data type like integer or float since data are initially loaded as VARCHAR in the database. In addition to data type conversion, the label encoding is performed on the class labels where  $\leq 50K$  is encoded as 0 while  $> 50K$  is encoded as 1. This step is necessary for categorical class label because most supervised learning techniques only handles real numbers.

Table 1 Education and Education\_Num

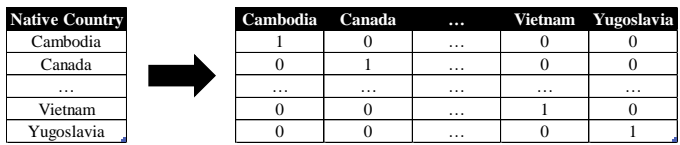
Education	Education_Num
Preschool	1
1st-4th	2
5th-6th	3
7th-8th	4
9th	5
10th	6
11th	7
12th	8
HS-grad	9
Some-college	10
Assoc-voc	11
Assoc-acdm	12
Bachelors	13
Masters	14
Prof-school	15
Doctorate	16

Furthermore, there are features like education can be eliminated because education\_num is sufficient enough to represent the education demographic of the customer as shown in *Table 1 Education and Education\_Num*.

Finally, marital status and relationship features is converted into one feature called as Married. For a customer to be tagged as married, he or she must have a marital status of Married-AF-spouse and Married-civ-spouse or relationship as Husband or Wife.

### 2.3. One Hot Encoding

One hot encoding technique is used for categorical features where each unique category is represented as a column (*Figure 4 One Hot Encoding Process*). This feature representation works better with classification algorithms.



Native Country	Cambodia	Canada	...	Vietnam	Yugoslavia
Cambodia	1	0	...	0	0
Canada	0	1	...	0	0
...	...	...	...	...	...
Vietnam	0	0	...	1	0
Yugoslavia	0	0	...	0	1

Figure 4 One Hot Encoding Process

### 2.4. Principal Component Analysis (PCA)

The main goal of Principal Component Analysis (PCA) is to reduce the dimensionality of the data set while retaining the variation present in the dataset up to the maximum extent. This study evaluate the effect in terms of accuracy and execution when only 90% (70) or 80% (60) of the components are used rather than all of the features.

### 2.5. Handling Imbalanced Data

Since the class label distribution in the data set are not represented equally, data may require additional techniques in handling this scenario because metric such as accuracy is highly influenced by the majority class which results to a lot of false majority class. Consequently, the following techniques are used and verified the impact in the overall performance:

- Random Oversampling (ROS) – this technique involves oversampling of the minority class. This technique does not lead to information loss but more susceptible to model overfit.
- Random Undersampling(RUS) – this technique involves undersampling of the majority class. This technique leads to information loss in the majority

class but lesser accuracy because significant training data records are removed.

- Synthetic Minority Oversampling Technique (SMT) – this technique creates artificial data based on feature space similarities from minority samples to shift the classifier learning bias towards minority class.

## 3. Learning Algorithms

This study used six supervised learning algorithm that is composed of classical, ensemble and trees machine learning algorithms. The following section discusses the parameters and design choices for each algorithm used.

### 3.1. AdaBoost Decision Tree (ADA-DT)

AdaBoost, short for Adaptive Boosting, is a popular boosting algorithm that combines weak learners into a single strong learner [4]. Each iteration learns on a set of training data, after which AdaBoost assigns higher weight to misclassified observations so that they can be included in the training set of the next learner. Furthermore, AdaBoost combines the classification made by the weak learners to form a stronger prediction boundary. Below are the list of parameter used for model tuning:

Table 2 ADA-DT Parameters

Parameters	Values
algorithm	['SAMME', 'SAMME.R']
base estimator max_depth	[2, 4, 6, 8, 10]
learning_rate	[0.1, 0.25, 1, 10]
n_estimators	[50, 100, 250, 500, 700]

### 3.2. Bagging Decision Tree (BAG-DT)

Bagging classifier is an ensemble model with decision tree used as base estimators. It holds all the general advantages of trees efficient on nonlinear, sparse, highly categorical dataset. The computational complexity increases with number of bootstrap trees as each tree needs to be pruned for the training set and the consensus on class needs to be established. Also, the complexity of model is high the tendency to overfit is also a problem including the interpretability of the model. Below are the list of parameter used for model tuning:

Table 3 BAG-DT Parameters

Parameters	Values
base estimator max_depth	[8, 10, 12, 14]
max_features	[60, 65, 70, 75]
max_samples	[0.4, 0.5, 0.6, 0.7, 1.0]
n_estimators	[50, 100, 250, 500, 700]

### 3.3. Gradient Boost Decision Tree (GB)

Similar to Adaboost, GradientBoosting Classifier is another popular boosting algorithm that uses gradients or loss function based on gradient descent to identify misclassified observations. It uses trees as default learners, hence, there is complexity in fine tuning model in both the boosting algorithm and the trees algorithm. Model complexity is influence by the number of trees included for learning, as well as the depth of the tree. Below are the list of parameter used for model tuning:

Table 4 GB Parameters

Parameters	Values
learning_rate	[0.1, 0.25, 1, 10]
loss	['deviance', 'exponential']
max_depth	[2, 4, 6, 8, 10]
n_estimators	[50, 100, 250, 500, 700]

### 3.4. Logistic Regression (LR)

Logistic Regression is a fundamental machine learning algorithm that has low model complexity because of ease of interpretation and limited number of parameters that can used in model [5]. Thus, the availability of regularisation and bias term consequently decreases the risk of overfitting. Below are the list of parameter used for model tuning:

Table 5 LR Parameters

Parameters	Values
C	[0.1, 0.25, 1, 10]
class_weight	[None, 'balanced']
max_iter	[100, 250, 500, 700, 1000]
penalty	['l1', 'l2']

### 3.5. Linear SVM (SVM)

Linear SVM is a highly scalable and one of the fastest modelling algorithm with low complexity. The linear kernel helps address the issue in overfitting as compared to other kernels. Furthermore, it helps maintain the robustness to outliers because of the C parameter. Linear SVM requires pre-processing techniques like PCA that transforms data to attain better results on non-linear data. Below are the list of parameter used for model tuning:

Table 6 SVM Parameters

Parameters	Values
C	[0.1, 0.25, 1, 10]
class_weight	[None, 'balanced']
dual	[True, False]
fit_intercept	[True, False]

### 3.6. XGBoost (XGB)

Extreme Gradient Boosting or XGBoost is another boosting algorithm that can provide lesser complexity because it uses a regularised model strategy to control overfitting [6]. The pruning parameters reduces the parameters that minimises the depth of the tree and sum of instances weight per leaf. In addition, it is computationally efficient, scalable and low memory usage. Below are the list of parameter used for model tuning:

Table 7 XGB Parameters

Parameters	Values
learning_rate	[0.1, 0.25, 1, 10]
max_depth	[2, 4, 6, 8, 10]
n_estimators	[50, 100, 250, 500, 700]
objective	['reg:linear', 'reg:logistic', 'reg:gamma', 'count:poisson']

## 4. Quantitative Evaluation Method

The empirical evaluation used different metrics to compare the performance of the six supervised learning algorithms used to build the binary classification model. Furthermore, the metrics to be used must take into consideration that the data is imbalanced that requires further analysis on the individual classes' performance. Consequently, the following metrics are used for classifier performance evaluation:

### 4.1. Classification Runtime

The classification runtime (EXEC) is measured from the training phase to the prediction phase of the classification in minutes to seconds.

### 4.2. Confusion Matrix

		Predicted	
		Class	
Actual	<= 50k	TN	FP
	> 50k	FN	TP

Figure 5 Confusion Matrix

The confusion matrix is used to describe the performance of a classification model on a set of test data for which the true values are known [7]. Ideally, the table has only values on its diagonal TN and TP. These matrix to have a deeper analysis on the performance of each class.

### 4.3. .Matthews Correlation Coefficient

MCC is used to measure the quality of binary classifiers. It considers true and false positives and negatives and generally regarded as a balanced measure which can be used even if the classes are imbalanced.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

### 4.4. Accuracy

Accuracy (ACC) is the ratio of the correct number of prediction over the number of data as shown

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

### 4.5. Precision, Recall and f1-score

Precision (PREC) is measure of exactness and the formula is derived from the confusion matrix.

$$Precision = \frac{TP}{TP + FP}$$

Recall (REC) is measure of completeness and the formula is derived from the confusion.

$$Recall = \frac{TP}{TP + FN}$$

The f1-score (F1) is the weighted average of the precision and recall, where an F1 score reaches its best value.

$$f1 - score = \frac{2 \times precision \times recall}{precision + recall}$$

### 4.6. 10-fold Cross Validation

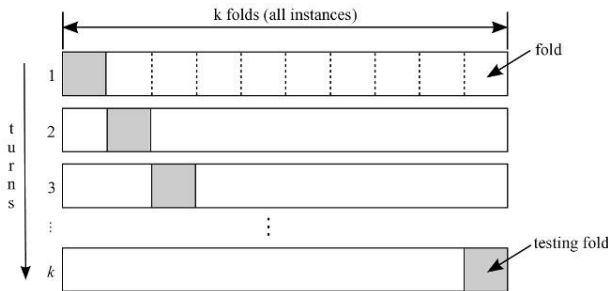


Figure 6 10-Fold Cross Validation

10-fold Cross-validation is a method to evaluate models by partitioning into fold training set to train the model, and the fold test set to evaluate it. The process is repeated 10 times and the performance specifically the accuracy is measured as the mean value of the 10 measurements.

## 5. Data Infrastructure

The Data Infrastructure as shown in *Figure 7 Data Infrastructure* provides a high-level strategy and tools used in this study. Having said that, Jupyter Notebook is used for prototyping and execution of Python 3.6 codes and Tableau is used for most of the data visualisation activities.

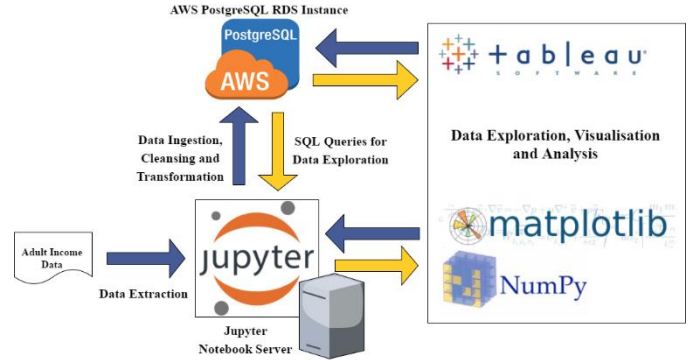


Figure 7 Data Infrastructure

An AWS PostgreSQL instance is created to stage, cleanse and transform data in which enables us to perform parallel prototyping whilst having a single source of truth minimising conflicts in the result. Furthermore, parallel processing is used in Grid Search and 10 fold Cross Validation during model tuning by allowing 3 jobs to run in parallel minimising the execution time and increasing productivity.

## 6. Ease of Prototyping

Python is programming language used in this study because of its ease of use, readability, portability and capability to perform object-oriented programming. In addition, data types are dynamic allowing you to write in lesser line of code compared to Java. Furthermore, extensive set of libraries are available like the numpy for matrix and vector operations, matplotlib for visualisation, scikit and scipy machine learning libraries that is tuneable using parameters and imblearn that provides methods in handling imbalanced data sets. Subsequently, Python provides general purpose programming language that is more flexible to work with unlike R that is designed specifically for statistical computing and graphics. Thus, using Python is best choice particularly for prototyping and experimentation of various machine learning problems that can run in a distributed computing environment.



## IV. Experiments and Discussion

### 1. Hardware and Software Configuration

The table below is the hardware and software specifications used to run the experiment and benchmark the result.

Table 8 Hardware and Software Configuration

Processor	Intel Core i5-6300U @ 2.4GHz 64-bit
RAM	8 GB
Operating System	Windows 10 Enterprise 64-bit
Python	Python 3.6.0 Anaconda 4.3.1 (64-bit)
Python Libraries	Datetime Imblearn Itertools Matplot Numpy Pandas Random Sklearn Xgboost
External Libraries	mingw-w64

### 2. Result

In this section, experiments are performed based on the methods mentioned in *Section III* and the result of each step is used as input in the succeeding process.

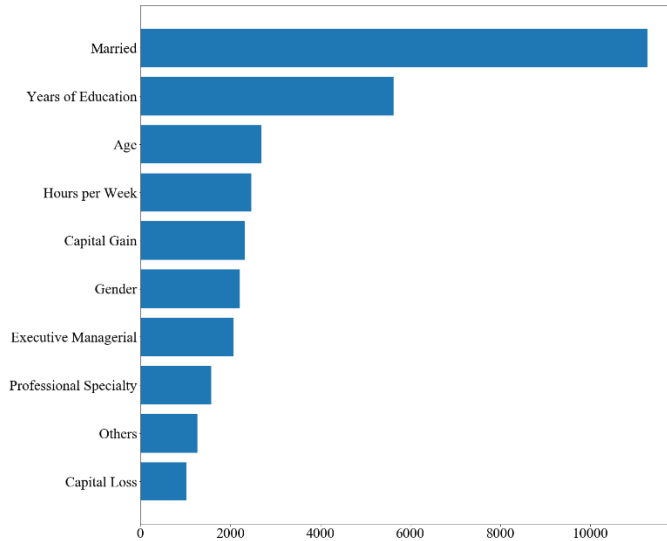


Figure 8 Top 10 SelectKBest Features

Firstly, the features analysis is conducted to determine the highest scoring features using SelectKBest method and `f_classif` scoring function. As shown in *Figure 8 Top 10 SelectKBest Features*, marital status is the highest scoring

feature amongst all attributes in the data set loosely followed by years of education, age and work hours per week.

After performing feature analysis, the data set with marital status and relationship are combined as one feature, records where NULL values are filtered and education feature removed (DCT = YES) does not have significant difference in performance across all metrics compared when utilising the raw data set using the machine algorithms with default setting as shown in *Table 9 Original Data Set vs Transformed Data Set*. Consequently, it is imperative to use the cleansed and transformed data set compared to the raw data since significant performance difference is observed.

Table 9 Original Data Set vs Transformed Data Set

MODEL	DCT	ACC	PREC	REC	F1	MCC
ADA-DT	—	0.857	0.850	0.860	0.850	0.581
BAG-DT	—	0.846	0.830	0.840	0.840	0.454
GB	—	0.855	0.870	0.870	0.860	0.581
LR	—	0.796	0.780	0.790	0.760	0.342
SVM	—	0.771	0.810	0.760	0.690	0.338
XGB	—	0.863	0.870	0.860	0.860	0.611
ADA-DT	YES	0.856	0.850	0.860	0.850	0.589
BAG-DT	YES	0.845	0.840	0.830	0.840	0.554
GB	YES	0.855	0.870	0.870	0.860	0.616
LR	YES	0.795	0.780	0.780	0.760	0.340
SVM	YES	0.772	0.810	0.750	0.690	0.339
XGB	YES	0.863	0.860	0.870	0.860	0.609

Subsequently, the cleansed and transformed data set is evaluated if using PCA (80% and 90% of total features) has impact in the overall execution time. As a result, PCA increased the overall execution of all algorithms without gaining significant performance increase as shown in *Table 10 PCA vs Non-PCA Data*.

Table 10 PCA vs Non-PCA Data

MODEL	PCA	ACC	PREC	REC	F1	MCC	EXEC
ADA-DT	—	0.855	0.850	0.860	0.850	0.588	0.037
BAG-DT	—	0.844	0.840	0.840	0.840	0.555	0.031
GB	—	0.866	0.860	0.870	0.860	0.617	0.037
LR	—	0.793	0.780	0.790	0.760	0.344	0.037
SVM	—	0.772	0.810	0.770	0.690	0.229	7.618
XGB	—	0.864	0.860	0.860	0.860	0.610	0.046

MODEL	PCA	ACC	PREC	REC	F1	MCC	EXEC
ADA-DT	70	0.847	0.840	0.850	0.840	0.562	16.731
BAG-DT	70	0.806	0.790	0.810	0.790	0.419	35.823
GB	70	0.847	0.840	0.850	0.830	0.550	22.073
LR	70	0.836	0.830	0.840	0.830	0.532	0.559
SVM	70	0.589	0.670	0.590	0.620	0.105	15.580
XGB	70	0.847	0.850	0.850	0.830	0.551	5.301
ADA-DT	60	0.843	0.840	0.840	0.840	0.551	14.553
BAG-DT	60	0.813	0.800	0.810	0.800	0.446	27.331
GB	60	0.851	0.850	0.840	0.850	0.563	17.500
LR	60	0.839	0.830	0.840	0.830	0.541	1.820
SVM	60	0.786	0.780	0.780	0.780	0.419	14.061
XGB	60	0.849	0.850	0.850	0.830	0.554	3.859

Subsequently, the data is processed using different sampling methods and tuned based on the parameters mentioned in the previous to improve performance metrics. Having said that, the below shows that Bagging Classifier has the most complex model with maximum of depth of 14 and 700 estimators compared to the rest of the Boosting and Ensemble algorithms whilst AdaBoost Decision is the simplest with maximum depth of 2 and 50 estimators. Furthermore, Logistic Regression works better in the data using the L1 penalty with no class weight and 100 iteration is sufficient for the model to converge.

Table 11 Parameters after GridSearchCV

Model	Parameter
ADA-DT	algorithm='SAMME.R' learning_rate=1 max_depth=2 n_estimators=50
BAG-DT	max_depth=14 max_features=60 max_samples=1.0 n_estimators=700
GB	learning_rate=0.1 loss='exponential' max_depth=6 n_estimators=100
LR	C=1 class_weight=None max_iter=100 penalty='l1'
SVM	C=10 class_weight=None dual=False fit_intercept=False
XGB	learning_rate=0.1 max_depth=6 n_estimators=100 objective='reg:logistic'

The summary of result as shown in *Table 12 Overall Result after Tuning* shows that the tuned data without performing any sampling technique delivers the best result in all metrics compared when performing sampling to handle imbalanced data. In addition, the random sampling methods makes the data more susceptible to overfitting or reduces accuracy because of the undersampling. On the other hand, SMT provide the best performance amongst the sampling technique because the generation of synthetic sample is more defined compared to the random methods but the result is not sufficient enough to be used in further evaluation. In terms of performance measure, XGBoost classifier performs the best closely followed by other Boosting algorithm like Gradient Boost, AdaBoost and Bagging Algorithm. Primarily, the boosting algorithm calls weak learning algorithm repeatedly, each time feeding it a different subset of the training examples. Each time it is called, the base learning algorithm generates a new weak prediction rule, and after many rounds, the boosting algorithm must combine these weak rules into a single prediction rule that, hopefully, will be much more accurate than any one of the weak rules. Meanwhile, Logistic Regression gained significant performance improvement by more than 5% in all metrics compared from its baseline result while SVM does not perform at par with the rest of the algorithms.

Table 12 Overall Result after Tuning

MODEL	TUNE	SAMP	ACC	PREC	REC	F1	MCC
ADA-DT	YES	—	0.865	0.860	0.860	0.860	0.620
BAG-DT	YES	—	0.863	0.860	0.860	0.860	0.606
GB	YES	—	0.871	<b>0.870</b>	<b>0.870</b>	<b>0.870</b>	0.635
LR	YES	—	0.847	0.840	0.850	0.840	0.564
SVM	YES	—	0.795	0.780	0.790	0.760	0.350
XGB	YES	—	<b>0.872</b>	<b>0.870</b>	<b>0.870</b>	<b>0.870</b>	<b>0.637</b>
ADA-DT	YES	ROS	0.855	0.860	0.850	0.860	0.620
BAG-DT	YES	ROS	0.860	0.860	0.860	0.860	0.614
GB	YES	ROS	0.863	0.860	0.860	0.860	0.634
LR	YES	ROS	0.842	0.840	0.840	0.840	0.578
SVM	YES	ROS	0.788	0.770	0.790	0.760	0.334
XGB	YES	ROS	0.861	0.860	0.860	0.860	0.629
ADA	YES	RUS	0.856	0.860	0.860	0.860	0.619
BAG-DT	YES	RUS	0.854	0.850	0.850	0.850	0.608
GB	YES	RUS	0.860	0.860	0.860	0.860	0.631
LR	YES	RUS	0.841	0.840	0.840	0.840	0.576

MODEL	TUNE	SAMP	ACC	PREC	REC	F1	MCC
SVM	YES	RUS	0.787	0.770	0.790	0.760	0.330
XGB	YES	RUS	0.860	0.860	0.860	0.860	0.627
ADA-DT	YES	SMT	0.864	0.860	0.860	0.860	0.620
BAG-DT	YES	SMT	0.862	0.860	0.860	0.860	0.606
GB	YES	SMT	0.867	0.860	<b>0.870</b>	0.860	0.630
LR	YES	SMT	0.843	0.840	0.840	0.840	0.581
SVM	YES	SMT	0.788	0.770	0.790	0.760	0.333
XGB	YES	SMT	0.868	0.860	<b>0.870</b>	0.870	0.631

The confusion as shown in Table 13 Confusion Matrix demonstrates that the boosting algorithms have similar minority class accuracy, precision, recall and f1-score whilst SVM performs poorly in predicting correctly the minority class based on the tuned data without any sampling method.

Table 13 Confusion Matrix

ADA-DT				BAG-DT			
Actual	Predicted			Actual	Predicted		
	Class	<=50K	>50K		Class	<=50K	>50K
	<=50K	10613	747		<=50K	10793	567
	>50K	1289	2411		>50K	1498	2202

GB				LR			
Actual	Predicted			Actual	Predicted		
	Class	<=50K	>50K		Class	<=50K	>50K
	<=50K	10667	693		<=50K	10541	819
	>50K	1259	2441		>50K	1479	2221

SVM				XGB			
Actual	Predicted			Actual	Predicted		
	Class	<=50K	>50K		Class	<=50K	>50K
	<=50K	10990	370		<=50K	10719	641
	>50K	2720	980		>50K	1291	2409

Moreover, the Table 14 Cross Validated Training Accuracy and Test Accuracy shows that the models are not overfitting since the cross validated training accuracy is at par with the test accuracy.

Table 14 Cross Validated Training Accuracy and Test Accuracy

Model	Training Accuracy (CV)	Test Accuracy
ADA-DT	0.864	0.865
BAG-DT	0.863	0.863
GB	0.869	0.871
LR	0.847	0.847
SVM	0.793	0.795
XGB	0.870	0.872

The Receiver Operating Characteristic (ROC) curve demonstrates that AdaBoost, Bagging, Gradient Boost, Logistic Regression and XGBoost performs at par which other while the Linear SVM is lagging behind the rest significantly.

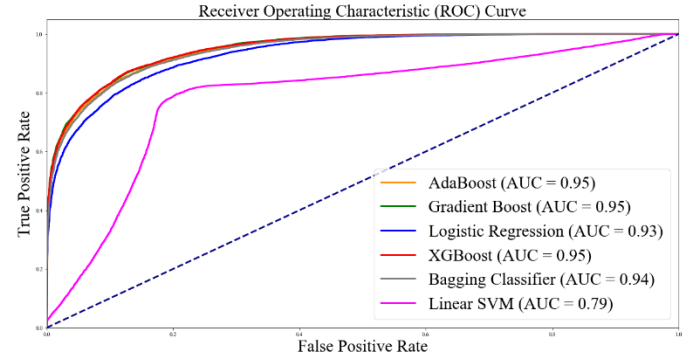


Figure 9 Receiver Operating Characteristic (ROC) Curve

Again, similar scenario is observed in the overall precision-recall curve of the data where AdaBoost, Bagging, Gradient Boost, Logistic Regression and XGBoost performs similarly whilst Linear SVM underperforms significantly.

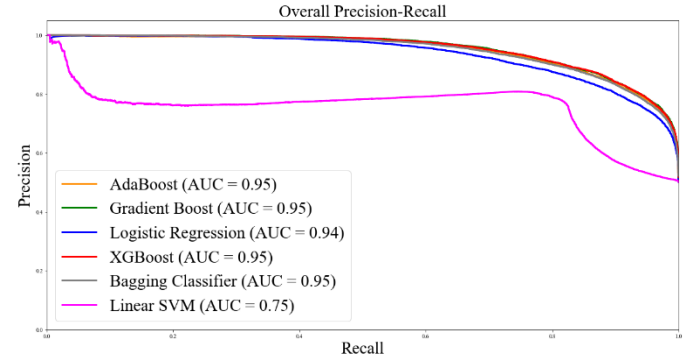


Figure 10 Over Precision-Recall Curve

Moreover, the precision-recall curve shown in Figure 11 Class Based Precision Recall Curve provides the individual performance of each class as a function of precision and recall. It can be observed that the majority class <=50K has an almost 100% area under the curve value (AUC) except for Linear SVM where it performs significantly behind the other models. Meanwhile, the bagging and boosting algorithm still performs quite well in the minority class >50K with an AUC value of more than 0.8 and Logistic Regression performs relatively well. However, Linear SVM has the worst performance at 0.45 AUC which explains why there are more false positives



and false negatives compared to true positive in the confusion matrix.

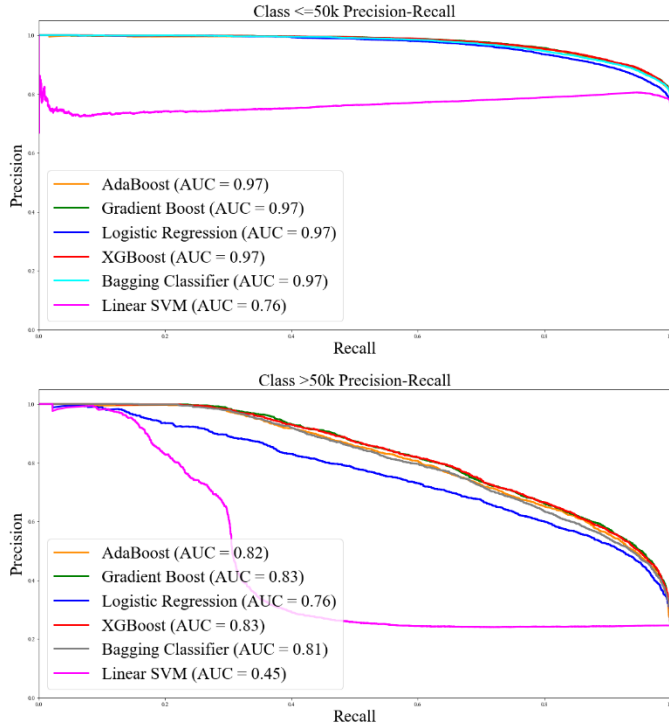


Figure 11 Class Based Precision Recall Curve

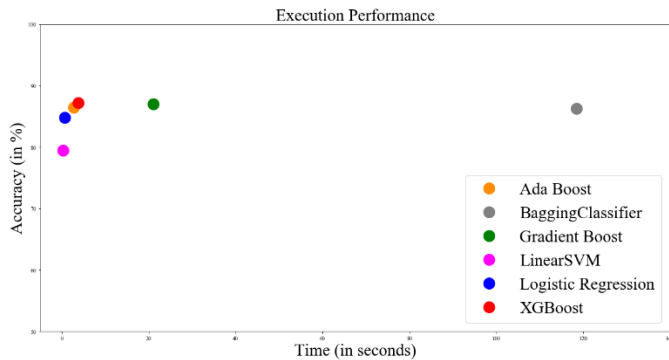


Figure 12 Machine Learning Algorithm Execution Time

In terms of execution performance, the fundamental algorithms LR and SVM performs under 1 second primarily because those two have lesser computational complexity than the rest of the models. Subsequently, AdaBoost and XGBoost under 5 seconds because the model complexity since the depth of the base estimator is only set at 2 for AdaBoost whilst XGBoost uses memory optimization that allows parallel threads to be run. Meanwhile, Gradient Boost lags behind because of computational complexity that slows the convergence of the observed values. Finally, the Bagging is 5.5 times slower than the next slowest (Gradient Boost) because of

the depth of the tree it has to produce at 14 before finally converging the observed samples.

### 3. Reflection

The data set itself is a challenge of imbalanced distribution of the classes thus, a thorough investigation of the overall performance and each class performance is required. Each phase of development from data analysis, model and parameter selection, data cleansing and pre-processing, sampling methods and model tuning requires detailed analysis as the chosen design impacts the succeeding phase. Data cleansing and transformation helps simplify the data set whilst PCA and handling imbalanced data like oversampling and undersampling does not gain performance improvement.

The Boosting algorithms performs excellent job in learning the data particularly AdaBoost that has lesser complexity compared to the rest of the boosting algorithms. Remarkably, Logistic Regression performs almost at par with the more complex algorithms and gains a lot of performance improvement after tuning and optimisation of parameter. The Support Vector Machine has quite good execution speed performance. In addition, it is observed that the model complexity is correlated with the execution speed of model. Meanwhile, the overall accuracy is highly influenced by the majority class in an imbalanced data that is why it is imperative to have further investigation on class based precision and recall.

## V. Conclusion

Table 15 Summary of Result

Model	Model Complexity	Performance Measure	Execution Speed ( in sec)
ADA-DT	Medium	High	2.70
BAG-DT	High	High	118.511
GB	High	High	21.044
LR	Low	High	0.554
SVM	Low	Low	<b>0.193</b>
XGB	High	<b>High</b>	3.784

Based on the experiment, it has been found that the AdaBoost Decision Tree classifier has the right balance of model complexity, performance measurement and execution speed compared closely followed by XGBoost and the simpler model, Logistic Regression. Gradient Boost and Bagging Decision Tree model has slower model convergence amongst other because of the model and computational complexity it requires while Support

Vector Machine has lacklustre performance except for execution speed. In terms of accuracy, the boosting algorithms Gradient and XGBoost performs above the previous work that was done in the data set at 87% and 86%, respectively. Therefore, the pre-processing and supervised learning techniques performed in this study are close to optimal.

The experiment has still room to improve the over-all performance of the classifier or conduct a study in a different perspective. In the data set perspective, collection of more data to test the robustness of each model since the data set is relatively small and has low dimensionality. Moreover, additional features about the person like spending habit and bank details can contribute to the overall performance of any future study. Furthermore, study about detecting outliers can help further improve the overall performance model to remove unwanted data. Future work can study using neural networks and deep learning models.

## VI. References

- [1] R. Kohavi (1996). Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid, 202–207
- [2] F. Kaya (2008). Discretizing Continuous Features for Naive Bayes and C4.5 Classifiers, 1–13
- [3] M. Hoffman (2012). Predicting earning potential on Adult Dataset, 1–30
- [4] R. Schapire. Explaining AdaBoost, 1–16
- [5] S. Dreiseitl, L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review
- [6] F. de Pison, E. Garcia, J. Cabello and A. Pernia (2016). Searching Parsimonious Solutions with GA-PARSIMONY and XGBoost in High-Dimensional Databases, 352–359
- [7] K. Markham (2014). Simple guide to confusion matrix terminology. Retrieved from <http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>

- [8] R. Caruana, A. Mizil, G. Crew, A. Ksikes (2004). Ensemble Selection from Libraries of Models, 1–9

# Appendix A: How to Run the Code

## Pre-requisites:

- Python 3.6.0 Anaconda 4.3.1 (64-bit) Distribution
- xgboost
- mingw-w64

## Installation on xgboost and mingw-w64 with Anaconda:

[https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing\\_XGBoost\\_For\\_Anaconda\\_on\\_Windows?lang=en](https://www.ibm.com/developerworks/community/blogs/jfp/entry/Installing_XGBoost_For_Anaconda_on_Windows?lang=en)

## Running the code:

There are two arguments passed:

**-c** the classifier to run. Can be any of the following values, default is **ALL**:

- AdaBoost
- GradientBoost
- LogisticRegression
- XGBClassifier
- BaggingClassifier
- LinearSVM

**-m** the mingw64 installation bin directory. This is needed to run the C++ library, libxgboost.dll, that comes with xgboost installation. Importing xgboost directly results in an error even if the path was added to system PATH variable. Thus, this is added to PATH environment at run time.


Note:

1. the default path is "C:\Program Files\mingw-w64\x86\_64-7.1.0-posix-seh-rt\_v5-rev0\mingw64\bin"
2. This is only needed for "XGBClassifier" and "ALL"

## To run a classifier, i.e. AdaBoost:

\*If xgboost is installed in the Anaconda, run in Anaconda prompt instead

```
python.exe comp5318_assign2.py -c AdaBoost
```

 Anaconda Prompt

```
(C:\Users\pastoem\AppData\Local\Continuum\Anaconda3) C:\Users\pastoem>python comp5318_assign2.py
```

## To run ALL classifiers:

```
python.exe comp5318_assign2.py -c ALL -m <mingw64 bin>
```

## For the data exploration:

Open the **comp5318\_experiments.ipynb**

## Appendix B: Group Contribution

We as a group have an effective communication to discuss and resolve challenges faced. Knowledge sharing has been a continuous process. A healthy competition and discussion prevailed to achieve better results. Overall, we are satisfied with quality and on time contributions in terms of both knowledge and techniques. The individual contribution are as below.

Name	Tasks
Emil Laurence Pastor	Contribution to main Framework Xgboost Logistic regression Main Report generation
Beulah Shanthini	Data Analysis and pre-processing Bagging Tree classifier Linear SVM Contributions to Report
Maureen Rocas	Code Integration Ada Boost Gradient Boost Contributions to Report