UCI Census Income Prediction

Deepshi Mediratta (300086013) Yashika Vittal (300069900)

Faculty of Engineering, University Of Ottawa, Ottawa, Canada
dmedi018@uottawa.ca, yvitt005@uottawa.ca

Abstract. This project aims at training machine learning models to perform binary classification to predict whether a person's income exceeds $50,000 per year. We use the UCI Census Income Prediction dataset for our analysis. The dataset is first analysed to understand the trends using Python. The relationship between the features are analyzed for feature selection. Classifiers are trained to perform predictions and their comparison is discussed. Models discussed include: Logistic Regression, Decision Tree, Random Forest Classifier, Bagging Classifier and AdaBoost Classifier.

Keywords: income prediction, binary classification

1 Introduction

We live in an era where everybody is striving to achieve as low an unemployment rate as possible. More and more people are obtaining degrees and education to get better and more paying jobs. As companies go on to hire new staff to cater to different work tasks, it becomes highly important to know how much a company can invest in salaries.

One of the most crucial aspects for a business to run is so to decide and take wise decisions about how much can be invested on its staff. In such situations, businesses will want to predict what salary an individual is worth based on a number of attributes. Such a prediction system allows the company and the candidate to gauge the income properly.

This paper focuses on predicting the income with the UCI Census Income prediction dataset using a number of features like age, education, occupation, marital status, relationship and more.

2 Case Study- UCI Census Income Prediction

2.1 Data Description

The dataset extraction was done by Barry Becker from the 1994 Census database. The data contains demographic and employment related variables. The dataset is already partitioned in training and test sets. The training data contains 32561 samples and test data contains 16281 samples. The dataset consists of 14 multivariate attributes, some were categorical and some were numeric. The target label is 'Class' attribute which has values- '<=50K' indicating samples with income less than $50K and '>50K' indicating samples with income greater than $50K.

Table 1. UCI Income Prediction Feature description

Feature	Feature Description
Age	The age of the person.
Workclass	Employment category (Private, Local-gov, Without-pay etc)
ID	Some ID representing the sample.
Education	The highest level of education achieved by the person.
Education_num	A Number representing the education level.
Marital Status	Marital status of a person.
Occupation	The profession of the person (Tech-support, Craft-repair etc)
Relationship	Relationship status (Wife, own-child, husband, not in family, others)
Race	The person's identification with one or more social groups.
Sex	Male or Female
Capital Gain	Capital gain for a person.
Capital Loss	Capital loss for a person.
Hours per week	The number of working hours per week.
Native Country	Country of origin.
Class	The target label that represents if the person earns more than $50K/year.

During the exploratory data analysis, the class distribution in the training and test dataset was analysed. Figure 1. shows that the samples in each class are divided in

a similar ratio (75% for <=50K and 25% for >50K) across training and test data. However it was also observed that the class is unbalanced.

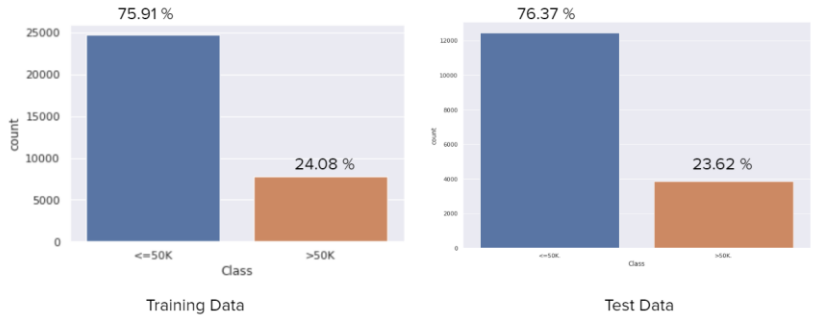

Fig. 1. Distribution of Class label in training data(left) and test data(right)

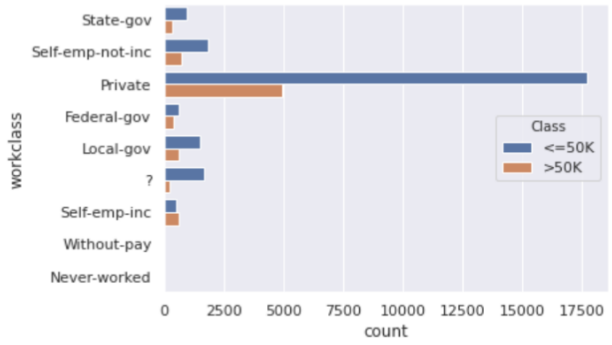

Fig. 2. Workclass Feature distribution with respect to class

The features against the class label were plotted to understand how each feature classified to a particular class label. For Example, in Figure 2 & 3, Occupation and Workclass attributes are plotted against the class label. In the Workclass graph, a high number of samples with ‘private’ Workclass were observed to have income less than $50K/year. It was also observed that there were some samples with missing values marked as ‘?’.

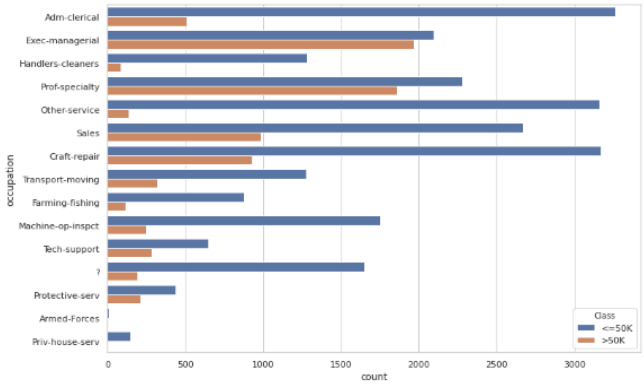

Fig. 3. Occupation Feature distribution with respect to class

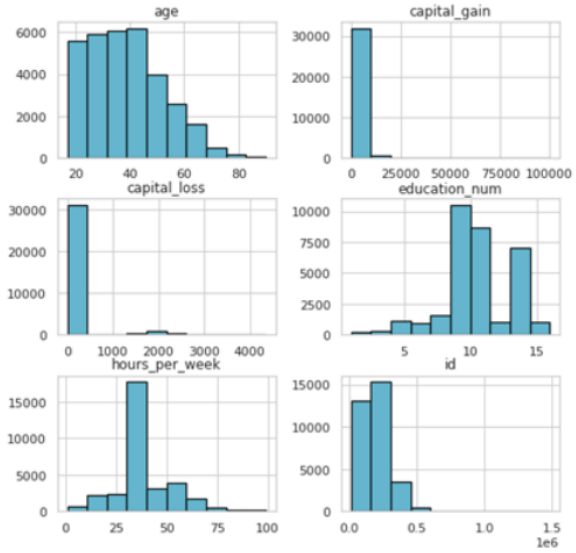

Fig. 4. Feature distribution (Age, Capital Gain, Capital Loss, Education num, Hours per week and ID)

By plotting the numeric features (Figure 4), it was observed that the values for the various numeric attributes are on different scales. Many machine learning models require the values to be on the same scale. This issue was addressed in Data Preprocessing.

3 Methodology

3.1 Data Cleaning and Preprocessing

We have to always do a lot of preparatory work before we feed the data and its features into the Machine Learning algorithms, as data in its raw form is not suitable for training models. Data cleaning is the process of ensuring that the data is correct, usable and consistent. There are a number of benefits of data cleaning , as mentioned below [1]:

1. It removes major errors and inconsistencies which arises when multiple sources of data are pulled into one dataset.
2. Data cleaning tools provide efficient and quick ways to clean the data.
3. It ensures we have as few errors as possible.

There are few best practices for creating the data cleaning like monitoring errors, validating accuracy, identifying and scrub duplicate data, etc.

Data preprocessing is usually carried out to transform the raw dataset into a form that is understandable. It is one of the most fundamental steps in data mining in order to improve the data efficiency [2]. There are many ways of preprocessing the data such as handling missing values, dealing with categorical data, feature selection, splitting dataset into train and test, and many more.

3.2 Handling Missing Data

A dataset can have missing values due to corruption in the way it was collected. It is important to handle these missing values to achieve successful results. We analyzed our dataset and there appears to be missing information in the dataset. Three attributes had missing values in the dataset: Workclass, Occupation and Native Country. A few different approaches were considered to address the problem of missing values-

Case 1 Drop All missing values: Only about 2-5% of the total samples had missing values for the three attributes (Table 2). These samples could be dropped without affecting the dataset.

Table 2. Percentage of missing values in Attributes.

Attributes	Missing %
Workclass	5.63%
Occupation	5.66%
Native Country	1,79%

Case 2 Replace '?' with 'unknown': In this approach, the missing values are replaced with 'unknown', by doing this we are introducing a new category called 'unknown' in the column.

Case 3 Finding Missing value with Logistic Regression: We used the Logistic Regression classification method to find the missing values (see results in Table). For the Workclass column, the classifier predicted that almost all samples belong to the 'Private' Class. So we replaced the missing values with 'Private'. For the Occupation column, majority samples belong to the 'Craft-repair' class. A few of them belonged to other classes such as 'Exec-managerial', 'sales' and 'prof-speciality'. We decided to replace the missing values with the majority class 'Craft-repair'. For the Native country column, the classifier predicted that all samples belonged to the 'united states' Class, which was probably because the majority of the samples were from the US. So we replaced the missing values with 'United states'.

Table 3. Missing value Prediction results from Logistic Regression.

Workclass	**Number of Records**
Private	1835
Self-emp-inc	1

Occupation	**Number of Records**
Craft-repair	1054
Exec-managerial	748
Sales	34
Prof-Speciality	1

Native Country	**Number of Records**
United States	583

Since it was computationally faster and using classification to predict missing values did not improve the performance of the models so we decided to just drop any sample with missing values.

3.3 Outlier Detection and Removal

Most machine learning algorithms are sensitive to the distribution of feature values in the dataset. Outliers in the dataset can skew the training of models and could lead to less accurate models. They could also represent incorrect data or samples that are irrelevant to the problem. Thus outlier detection is an important step in the data preprocessing pipeline.

The records in UCI income dataset were reasonably clean because the records were extracted using the following conditions [1] [2]:

((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

where, AAGE = Age attribute

AGI = Adjusted gross income

AFNLWGT = User ID

HRSWK = Hours per week

We verified this using some boxplots (Figure 5 & 6):

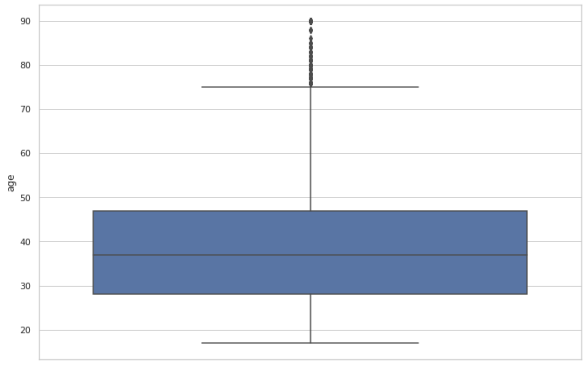

Fig. 5. Boxplot for Age attribute

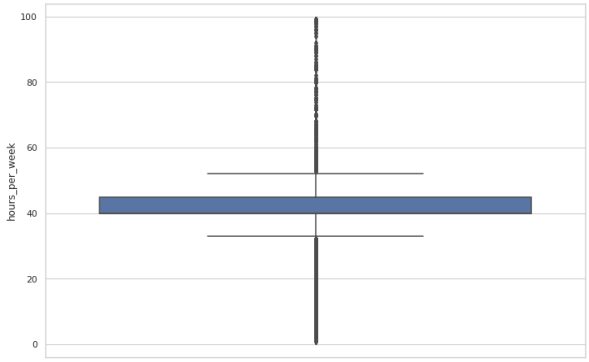

Fig. 6. Boxplot for hours_per_week attribute

3.4 Feature Selection

Correlation Analysis: It is a measure that helps us analyze how two features are linearly related to each other. We want to evaluate which features have the highest impact towards our target prices. To do this, we plot the heat map (Figure 7) which describes the correlation coefficients of our variables. The map shows high dependency of Class with features such as age, education_num, capital_gain, capital_loss, hours_per_week. The variables that are highly correlated with the target are likely significant in regression analysis.

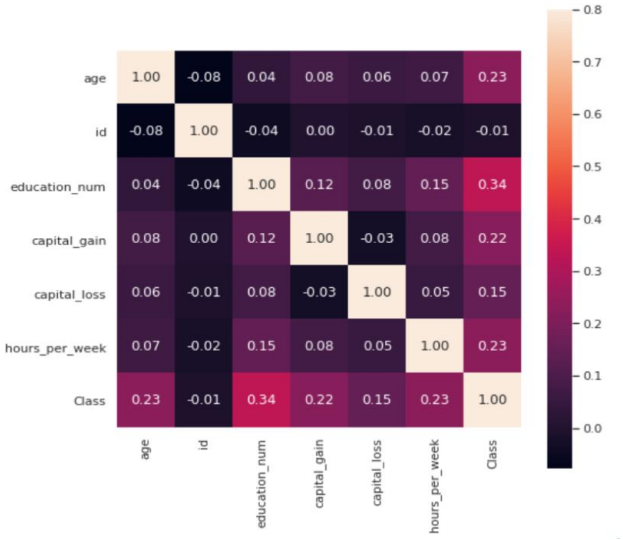

Fig. 7. Correlation Analysis

So, we have narrowed down our features using correlation analysis.

3.5 Final features

The final features to be used in all the regression models are -

Categorical Features: workclass, marital_status, occupation, relationship, race, sex, native_country

Numeric Features: age, education_num, capital_gain, capital_loss, hours_per_week

Target Label: Class

3.6 Normalization

At this point, all of our data has been converted to numeric form. But they are not on the same scale. When we are working with a regression problem, it is important to scale the features to the same range, which will also make their variance in the same range. This is done so that the order of magnitude of the features are the same. To achieve this we use the Min-Max Scaling technique. In this, the data is scaled to a fixed range- usually 0 to 1 to achieve Gaussian with zero mean and unit variance. We have used sklearn.preprocessing.MinMaxScaler to achieve this.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}(new_{max} - new_{min}) + new_{min}$$

3.7 Data Sampling

4 The Models

4.1 Logistic Regression

Logistic Regression [5] is a statistical technique, where it is used to model the probability of a certain class or event existing. This can be extended to model several classes of events such as determining whether it belongs to one of the classes. Each data tuple would be assigned a probability between 0 and 1 and the

sum adding to one. It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

4.2 Decision Tree Classifier

A Decision Tree [6] is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter. It consists of:

1. Nodes: Test for the value of a certain attribute.
2. Edges/Branch: Correspond to the outcome of a test and connect to the next node or leaf.
3. Leaf Nodes: Terminal nodes that predict the outcome (represent class labels or class distribution).

There are two types of decision trees- Classification tree and Regression Tree.

4.3 Random Forest Classifier

Random forests or random decision forests [7] are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

4.4 Bagging Classifier

A Bagging classifier [8] is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

4.5 AdaBoost Classifier

An AdaBoost classifier [9] is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

5 Experiments

5.1 Training and Test datasets

The dataset is already partitioned in training and test sets. The training set contains a known output and is used for learning. The test dataset is used to test the model's prediction. The data set in 66:33 ratio, 66% for the training and 33% for testing.

5.2 Cross Validation

It is a method that estimates test error by holding out a subset of training data from the fitting process. K-Fold Cross Validation is where a given data set is split into K number of sections/folds and then each fold is used as a testing set at some point. In our experiments, we have taken the value of k as 10.

5.3 Evaluation Metrics

Confusion Matrix
A confusion matrix is a table describing the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. Also, It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix. We plotted the confusion matrix to evaluate the performance of the classifiers by comparing the predicted labels against the true labels of instances. We observed that the Decision tree was able to predict most number of positive samples correctly. This was the minority class. However Decision tree did not predict a lot of negative samples correctly. Bagging classifier has the highest true negatives but it failed to predict true positives. By just looking at the confusion matrix we could not decide which model was the best for our problem.

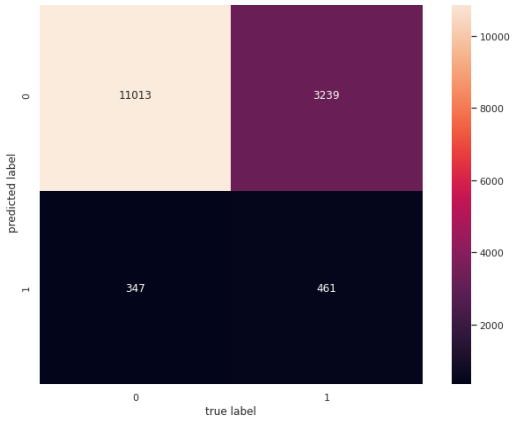

Fig. 8. Confusion Matrix: Logistic Regression

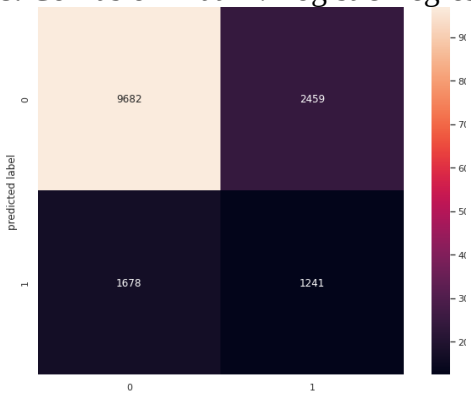

Fig. 9. Confusion Matrix: Decision Tree Classifier

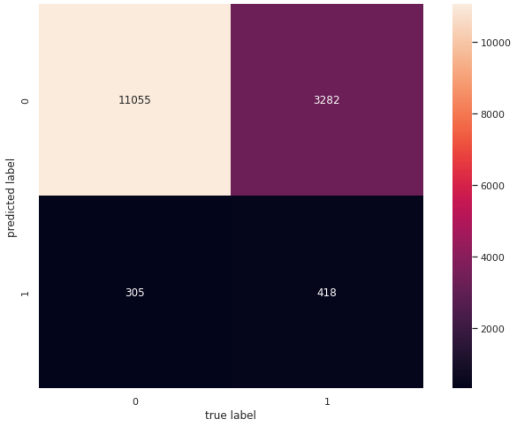

Fig. 10. Confusion Matrix: Random Forest Classifier

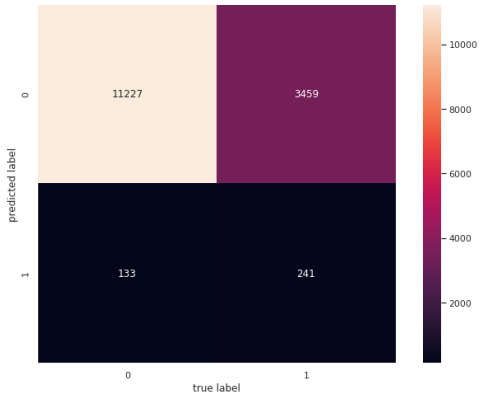

Fig. 11. Confusion Matrix: Bagging Classifier

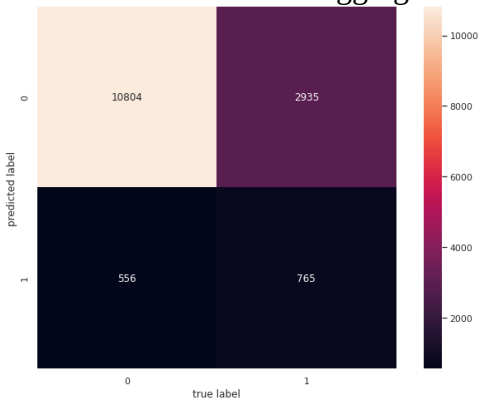

Fig. 12. Confusion Matrix: AdaBoost Classifier

ROC

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as probability of false alarm and can be calculated as (1-specificity).

We plotted the true positive rate and false positive rate for the models and calculated the area under the curve. This evaluation method showed us that even though the decision tree has highest true positives, it does not have a good balance between TPR and FPR which results in a low AUC. Logistic regression has the highest AUC. Random forest, bagging and adaboost had similar AUC.

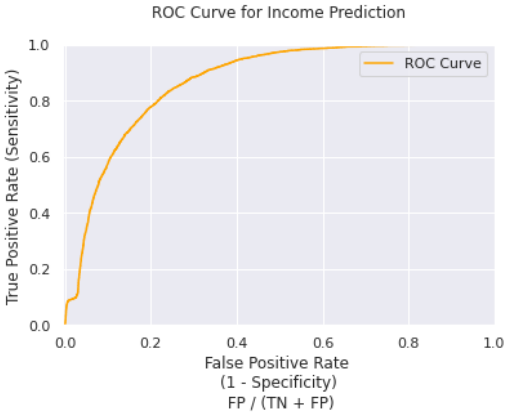

Fig. 13. ROC: Logistic Regression

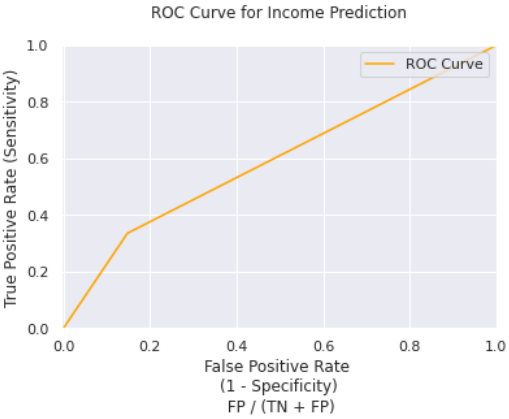

Fig. 14. ROC: Decision Tree Classifier

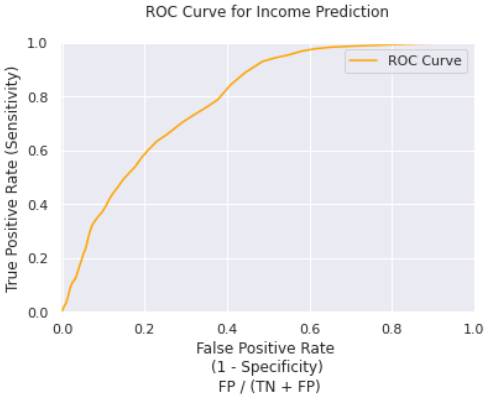

Fig. 15. ROC: Random Forest Classifier

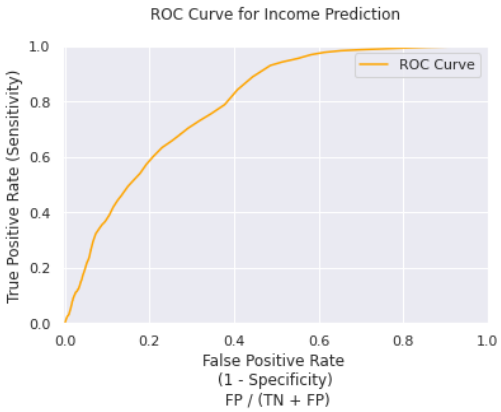

Fig. 14. ROC: Bagging Classifier

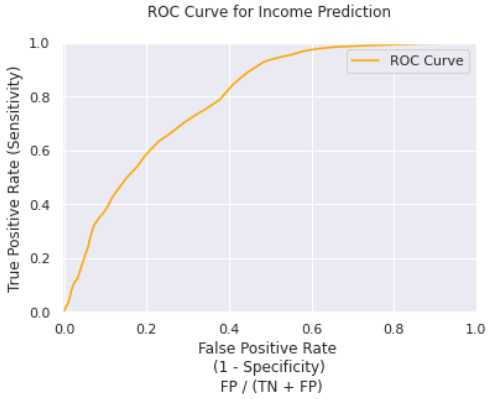

Fig. 15. ROC: AdaBoost Classifier

Box-plot

Then we used the 10-fold cross validation procedure to evaluate each algorithm. We configured the cross validation with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way.

The provided box and whisker plot shows the spread of the accuracy scores across each cross validation fold for each algorithm. We observed that random forest and bagging have similar scores. After these, the decision tree has the best performance.

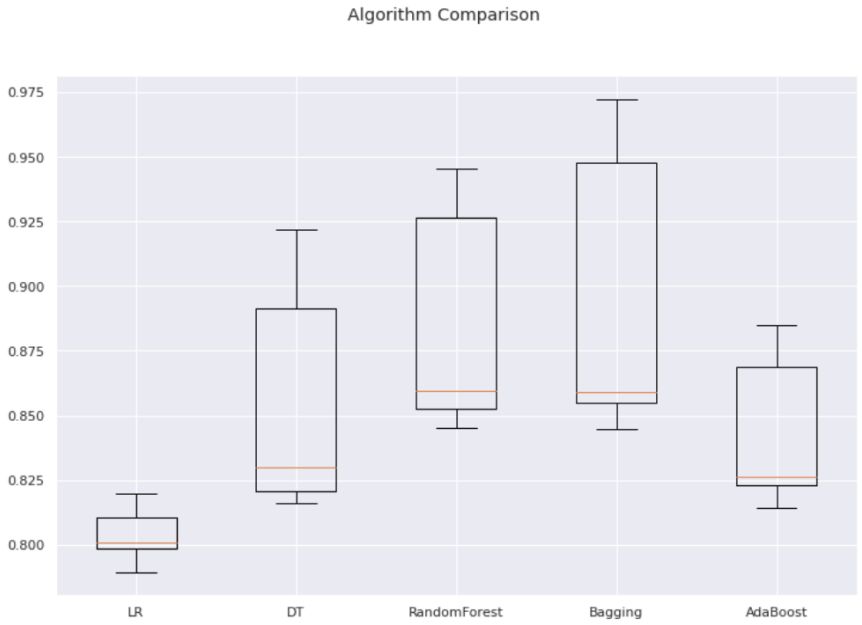

Fig. 16. Box-plot

6 Results/Model Comparison

The results from the various classification models, mentioned in the previous sections, on the UCI Income Prediction dataset is shown in table 2 and table 3. Based on the evaluation methods discussed so far, the top three models are DT, random forest and bagging classifier. If we look at the sensitivity that is the TPR, then DT has performed better on our dataset but if we focus on the specificity that is the true negative rate then the bagging classifier performed better. Since our problem focuses on predicting whether a person's income exceeds $50k / yr or not. Which means we want a model that would give us a high TPR so we choose DT as our final model.

Table 4. Model Comparison

Model	Accuracy	AUC	Sensitivity/ Recall	Specificity
Logistic Regressio	76.18	86.9	12.46	96.95
Decision Tree	72.52	59.38	33.54	85.23
Random Forest	76.18	79.31	11.30	97.32

Classifier				
Bagging Classifier	76.14	79.31	6.51	98.83
AdaBoost Classifie	76.81	79.31	20.68	95.11

7 Conclusion

We worked on the UCI Census Income dataset to predict the income of an individual based on a number of features. We set up a process with many steps in order to achieve good accuracy and efficient results.
We explored and understood the dataset by exploratory data analysis with the help of histograms, graphs of features against class and more. This helped us understand how the dataset was distributed and if there was any bias in the dataset that we would have to handle before training models. We also plotted the label data and realised the class label was highly unbalanced, and fixed it with Smote Analysis.
Further, With the help of various preprocessing techniques, such as Handling missing, Outlier detection, feature selection, categorial to numeric transformation, normalization and data sampling, we attempted our best to clean and transform data to improve data efficiency. After cleaning and preprocessing the data, we trained various models on the dataset.
We trained 5 models i.e,, Logistic Regression, Decision Tree, Random Forest Classifier, Bagging Classifier, AdaBoost Classifier. Most of the models gave an accuracy of around 70%. Although AdaBoost Classifier gave us the highest accuracy, Decision Tree predicted the best based on its sensitivity. Decision Tree also shows a great balance between the specificity and sensitivity.

References

[1] L. Gimenez, «6 Steps for Data Cleaning and Why it Matters | Geotab,» Geotab, 24 May 2018. [Online]. Available: https://www.geotab.com/blog/data-cleaning/.
[2] U. Verma, «Data Cleaning and Preprocessing,» Medium, 19 November 2019. [Online]. Available: https://medium.com/analytics-vidhya/data-cleaning-and-preprocessing-a4b751f4066f.
[3] «UCI Machine Learning Repository: Census Income Data Set,» Archive.ics.uci.edu, [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Census+Income.

[4] «Census-Income Database,» Kdd.ics.uci.edu, [Online]. Available: https://kdd.ics.uci.edu/databases/census-income/census-income.data.html.
[5] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
[6] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.
[7] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.
[8] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.
[9] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.