Predicting Income Class Using Machine Learning

1.0 Abstract

Nations all over the world are concerned with poverty and inequality in the distribution of wealth. The question ‘what makes some people poor and other rich?’ is one that has been posed for decades. This paper shows how machine learning can be used to answer that question

In this paper, investigation is made into a US Adult Census Income dataset. Three machine learning algorithms are deployed to predict if a person’s income will be less than/equal to or greater than 50,000 dollars per annum based on various information available about the person. The algorithms used are Logistic Regression, Random Forest and Support Vector Machines. The result of the models on the training set are compared. Finally, the model with the best training set score is used on the test set. Support Vector Machines proved to be the best model with a result of 85.85% on the training set and 86% on the test set

2.0 Introduction

In recent times, it has become clear that the applications of machine learning go beyond science and technology. As governments grapple with the issues of wealth inequality and poverty, machine learning can be used as a tool in seeking solutions to these problems. A reduction in income disparity will lead to socio-economic development and better quality of life for the citizens of a nation.

With machine learning techniques the key factors that affect an individual’s income can be analysed and predictions can be made with a high degree of accuracy as to what income bracket a person will fall into given the right parameters.

This is want this paper seeks to show. In this paper, machine learning algorithms are used to answer the question of predicting a person’s income class. The dataset was gotten from Kaggle. Different stages of data analysis such as data preparation, cleaning, building and fine-tuning the models, testing and evaluation have been carried out on the data.

The paper has been divided into introduction, literature review, professional legal and ethical issues, proposed methodology, experiments, results and discussion and finally conclusion and future work.

3.0 Literature Review

A number of research has been done in using machine learning to predict income levels.

One of these is by Chakrabarty and Biswas (2018) in which the Gradient Boost Classifier was used to predict income category

Another is that by Chockalingam et. al (2017), here seven machine learning models were used including Logistic Regression, KNN, Naïve Bayes and the result of the various models were compared with the highest accuracy being 87%

Bekena (2017) used the random forest algorithm to predict income levels with an accuracy of 85%

Lazar (2004) used Support Vector Machines with Principal Component Analysis and obtained an accuracy of 85%

These studies show that machine learning algorithms can easily be used to perform predict income levels with an impressive accuracy rate.

4.0 Professional, Legal and Ethical Issues

Since this study uses data about individuals, professional, legal and ethical issues are considered. However, the data is anonymised to keep the identity of the individuals hidden. Also, the data was gotten from Kaggle which is an open source repository. Hence, there are no legal or ethical breaches in conducting this study

5.0 Proposed Methodology

The dataset used for this project is the Adult Census Income data. The purpose of building the models is to predict whether or not a person's income exceeds 50,000 dollars/annum. In order to accomplish the project problem, the proposed solution can be described in following steps:

1. Collecting the dataset and identifying the problem: The dataset was gotten from Kaggle (https://www.kaggle.com/uciml/adult-census-income). The problem was seen to be predicting whether a person's income exceeds 50,000 dollars/annum based on certain information/attributes about them gotten from the census.
2. Exploratory Data Analysis: The data set will be looked at in more details to gain insights and also to see what pre-processing steps such as data cleaning, checking for missing values, converting data type, correlation and visualisation will need to be undertaken. Here the data will also be split into train and test sets.
3. Data Pre-processing: All the necessary pre-processing and feature engineering steps noted during the exploration phase will be carried out here
4. Building, Train and Fine-Tuning the Models(Experiments): Here we build the models and train them based on the training sets. The models used are Logistic Regression, Random Forests and Support Vector Machines. The models will also be fine-tuned based on certain hyper-parameters
5. Test and Evaluate the Models: The performance of the models and their accuracy are checked here

6.0 Collecting the Data Set and Identifying the Problem.

The dataset which was gotten from Kaggle is the Adult Census Income data. We analyse the data and train models to learn from this data so as to predict whether a person earns above 50,000 dollars per year given the required metrics about them. The dataset has 32,561 rows and 14 attributes (6 numeric and 8 categorical) which show various anonymised information

about a person and one target column which shows whether or not that person's income exceeds 50,000 dollars (<=50K or >50K)

Attribute Information:

Input Variables

1. age: The person's age (Numeric Attribute)
2. workclass: The person's job type (Categorical Attribute)
3. fnlwgt: The number of people he census believe share this entry/demography (Numeric Attribute)
4. education: The person's level of education (Categorical Attribute)
5. education.num: The person's level of education represented in numerical form (Numeric Attribute)
6. marital.status: The person's marital status (Categorical Attribute)
7. occupation: The person's occupation type (Categorical Attribute)
8. relationship: Who this person is relative to others in a family setting (Categoric Attribute)
9. race: The person's race (Categorical Attribute)
10. sex: The person's sex (Categorical Attribute)
11. capital.gain: Capital gains for the person (Numeric Attribute)
12. capital.loss: Capital loss for the person (Numeric Attribute)
13. hours.per.week: The number of hours the person works per week (Numeric Attribute)
14. native.country: The person's country of origin (Categoric Attribute)

Output Variable

15. income: Whether or not the person makes more than 50K dollars per annum (Categorical Variable)

7.0 Exploratory Data Analysis

7.1 Loading the Data

The data was stored as a CSV file. The code was run using Python in a Jupyter notebook. The necessary libraries were imported (Numpy, Pandas, Matplotlib and Seaborn). The data was now read as a pandas dataframe using the pd.read_csv feature of pandas.

7.2 Overview of Data

On inspection of the data, it was noticed that there were some missing values which were represented with '?'. The data was also seen to have 24 duplicate rows which were removed. A statistical break down of the data is shown below:

	age	Fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
Max	90	1484705	16	99999	4356	99
Min	17	12285	1	0	0	1
Mean	38.5	189778	10.08	1077.65	87.3	40.4
std	13.6	105550	2.57	7385.29	402.96	

Grouping the mean of the numerical features based on the target label gives us the following:

Income	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
<= 50K	36.78	190345.92	9.59	148.88	53.19	38.84
>50K	44.25	188000.48	11.61	4007.16	195.05	45.47

From the above table, we see that people who are older, more educated and work more hours tend to earn above 50k per annum

7.3 Handling Outliers

Using the zscore method, outliers were removed from the numerical columns. A zscore of 3 was taken as the threshold and only data with zscore less than 3 was retained.

7.4 Feature Selection and Correlation

First we define a function to change the income column to 0's and 1's. values <=50K were replaced with 0 while those >50K were replaced with 1. Also the missing values were replaced with NaN.

Since the education.num column is a numerical representation of the education column, only one of them is needed. Hence, the education column was dropped.

Before proceeding further, the data was separated into train (80%) and test (20%) set to avoid further tampering with the test set.

We check for correlation between the output variable and other numerical variables and we see that capital.gain and education.num have the highest correlation while capital.loss has the least correlation. The heat map showing the correlation is displayed below:

Fig 1: Heat Map Showing Correlation Between the Income Column and other Numerical Attributes

7.5 Data Pre-Processing / Feature Engineering

There are three main data pre-processing steps that we have noted needs to be carried out before we fit our train set to the machine learning models: imputing missing values, categorical attributes encoding and feature scaling. First the train set is split into the dependent variables (X) and target (Y). A pipeline is created for the data pre-processing steps. Since our missing values are all in the categorical column, first we create a pipeline to impute the missing values and then one hot encode the columns. For the imputer strategy, since categorical values don't have a mean or median, we are left with the mode (most frequent).

A column transformer is then constructed which transforms the categorical column using the earlier constructed pipeline and scales the numerical columns. The train set is passed through this pipeline and we are ready to train our model

8.0 Model Selection

The machine learning task is a binary classification type task and so models suitable for classification have been chosen. The models that have been used are Logistic Regression, Random Forest and Support Vector Machines

8.1 Logistic Regression

This is the most widely used model for binary classification. It models the relationship between the input variable X and output variable Y (Levy and O'Malley, 2020). Logistic regression assumes that the data set is reasonably large and that there is no co-linearity between the independent variables (Tutorialspoint, 2021). Hence it is regarded as a good model for this dataset

8.2 Random Forest

Random forests are estimators that work by fitting multiple decision trees on sub-samples of the dataset and merges them together to get a more accurate prediction (BuiltIn, 2020). Random forest has the advantage of not only given better predictions but also being able to give the importance of each features. These advantages made it a choice for this model.

8.3 Support Vector Machine

They work by selection of a line with maximum margins using points called support vectors (Golpour et al, 2020). The support vector model itself chooses the best line to classify datasets making it a good classification model.

9.0 Experiments

The experiments with the machine learning models were carried out in the following order:

1. The models were fitted on the train set to see how they performed.
2. Cross-validation was used on the models to improve their performance
3. The models were fine-tuned using different hyper-parameters to try to further improve performance and the best parameters were gotten

4. A new pipeline was created using the top 5 features as gotten from the random forest classifier
5. The model was fine-tuned again using the new pipeline and new parameters were chosen based on the best parameters gotten earlier.

The evaluation metrics used are the accuracy score and confusion matrix

9.1 Experiment 1: Fitting the models on the train set

Logistic Regression

For this experiment, logistic regression gave an accuracy of 0.8526(85.26%). The confusion matrix gave 20,333 True Positives and Negatives and 3,514 False Positives and Negative

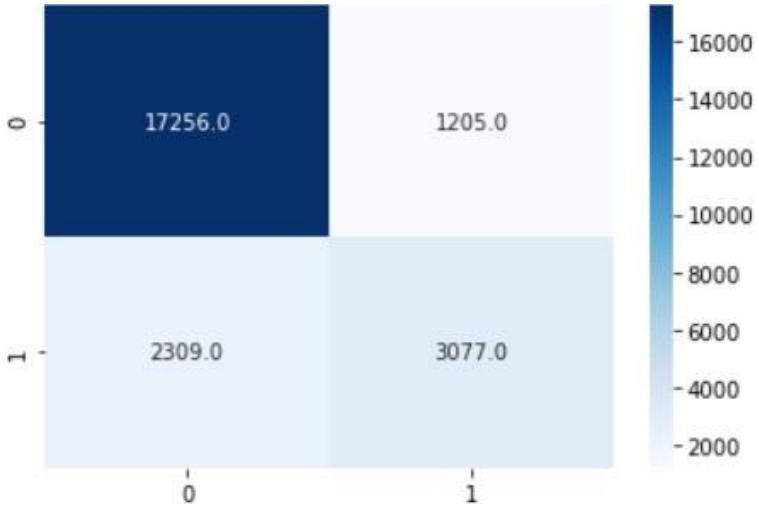

Fig. 2: Confusion Matrix for Logistic Regression for Experiment 1

Random Forest

Random Forest gave an accuracy of 0.9999(99.99%). In this instance the model was badly over-fitting the data

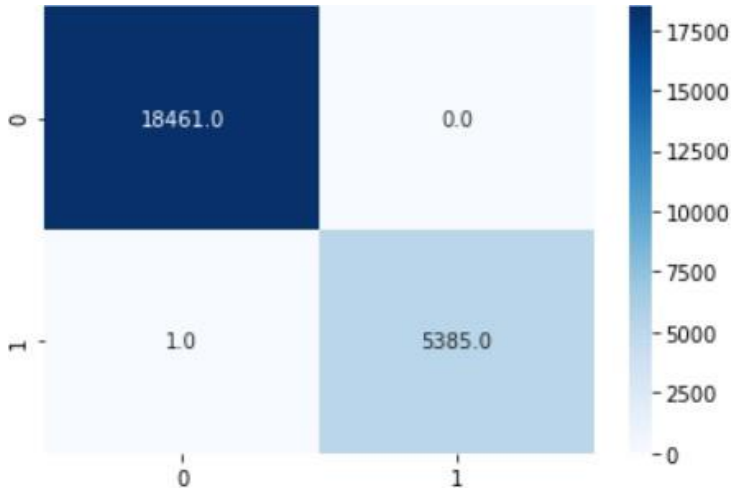

Fig. 3: Confusion Matrix for Random Forest for Experiment 1

Support Vector Machines

This gave an accuracy score of 0.8638(86.38%). The confusion matrix gave 20,601 True Positives and Negatives and 3,246 False Positives and Negatives

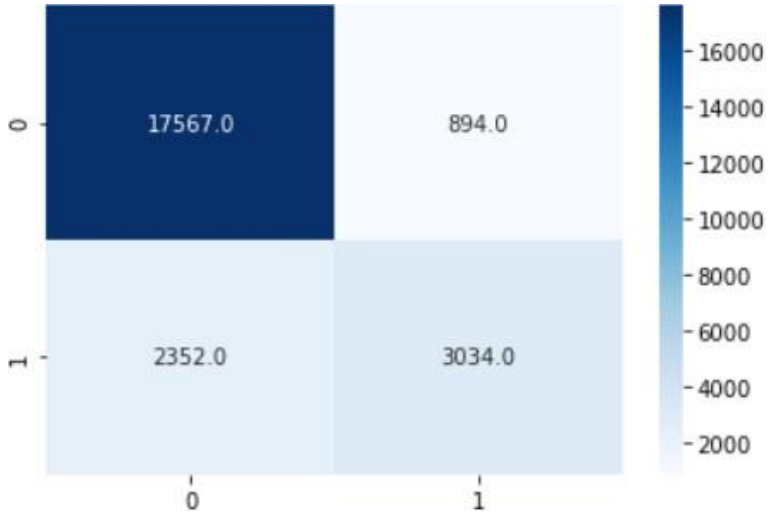

Fig. 3: Confusion Matrix for Support Vector Machines for Experiment 1

Experiment 2: Using cross-validation on the models to improve their performance

Logistic Regression

For logistic regression, the mean score from cross-validation was 0.8521(85.21%)

Random Forest

For this model, the mean cross validation score was 0.8512(85.12%). This shows an improvement from the 0.9999 gotten earlier

Support Vector Machines

The mean cross-validation score for this model was 0.8585(85.85%)

Experiment 3: Fine-Tuning The Models

Logistic Regression

The parameters used were: penalty: [l1, l2, none], C: [np.logspace(-4, 4)], solver: [lbfgs, newon-cg, liblinear], max_iter: [100, 1000, 2500]

The best parameters were gotten to be: C: 0.05963623316594643, max_iter: 100, penalty: l2, solver: liblinear

The best score gotten was 0.8529(85.29%)

Random Forest

The parameters used were: {n_estimators: [10, 30, 50], max_features: ['auto'], max_depth: [100,1000]}, {n_estimators: [10, 100, 1000], max_features: [2,4,6,8]}

The best parameters were gotten to be: max_depth: 100, max_features: auto, n_estimators: 30

The best score gotten was 0.8526(85.26%)

Support Vector Machine

The parameters used were: {kernel: [linear', poly, rbf], C: [1, 5, 10], degree: [3, 8]}, {coef0: [0.01, 0.5, 10], gamma: [auto, scale']}

The best parameters were gotten to be: C: 1, degree' 3, kernel: rbf

The best score gotten was 0.8585(85.85%)

Experiment 4: Fine-Tuning Again based on the Best Features and Hyper-Parameters

From the Random Classifier, the feature importance of all the features was obtained. The top 5 features were seen to be: age, fnlwgt, capital.gain, capital.loss and workclass. A new transformation pipeline was built and based on the hyper-parameters from the initial tuning, a new set of hyper-parameters were gotten. The models were tuned again with these new hyper-parameters and based on the new pipeline

Logistic Regression

The parameters were: penalty: l2, C: np.logspace(-1, 1), solver: lbfgs, max_iter: [50, 100]

The best parameters were gotten to be: C: 0.14563484775012436, max_iter: 100, penalty: l2, solver: lbfgs

The best score gotten was 0.8529(85.29%)

Random Forest

The parameters were: n_estimators: [5, 10, 30], max_features: auto, max_depth: [50, 100]

The best parameters were gotten to be: max_depth: 50, max_features: auto, n_estimators: 30

The best score gotten was: 0.8515 (85.15%)

Support Vector Machines

The parameters were: kernel: rbf, C: [-1, 1], degree: [1, 2, 3]

The best parameters were: C: 1, degree: 1, kernel: rbf

The best score gotten was: 0.8585(85.85%)

9.0 Results and Discussion

The table below shows the results gotten for each of the experiments that was run on the models

Model	Without CV	With CV	First Tuning	Second Tuning
Logistic Reg	0.8526	0.8521	0.8529	0.8529
Random Forest	0.9999	0.8512	0.8519	0.8515
SVM	0.8638	0.8585	0.8585	0.8585

The result show that the models had a lesser accuracy score with cross validation. This is probably because it helped to solve the problem of over-fitting especially with the Random Forest Classifier.

The first set of fine-tuning produced a slight increase in the accuracy score while the second with fewer features had almost no impact on the accuracy score. While this might be taken to mean that there was no need for the second set of hyper-parameter tuning, it actually proves that that the top 5 features as determined by the Random Forest Classifier (age, fnlwgt, capital gain, capital loss and workclass) are the most important in predicting a person's income.

9.1 Evaluating The Test Set

Overall, Support Vector Machines had the best accuracy score so it was used on the test set. The testing set accuracy gotten was 0.8604(86.04%). The confusion matrix showed that there were 5130 correct predictions (True Positives and True Negatives) and 832 wrong predictions (False Positives and False Negatives)

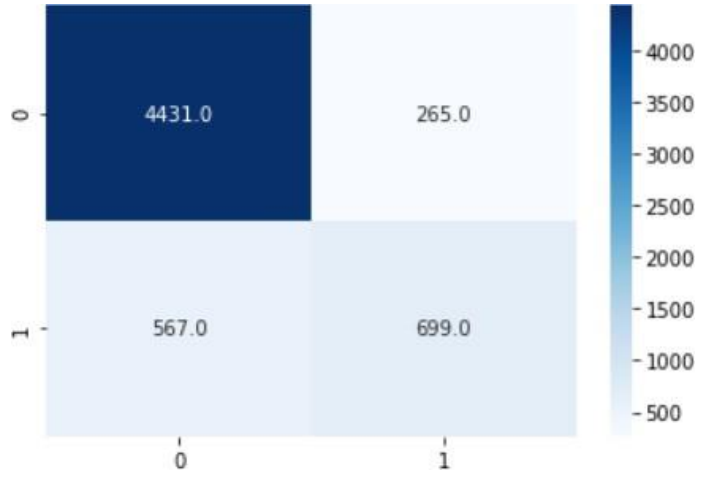

Fig. 4: Confusion Matrix of the Test Result

10.0 Conclusion and Future Work

In this paper, various experiments were conducted using 3 machine learning algorithms on the Adult Census Income dataset to predict income classes. The experiments showed that Support Vector Machines performed best with a test accuracy of 86%. This study shows that machine learning can be used to identify factors that cause disparity in incomes and with this knowledge, the correct steps can be taken to bridge the income divide.

In terms of future work, more features and attributes can be added to those already included in the dataset and we can see the effects of these features on the accuracy of model prediction. Also other advanced machine learning techniques can be used and their accuracy compared to the models that have been used in this paper

11.0 References

- Bekena, S. M., (2017) "Using decision tree classifier to predict income levels", *Munich Personal RePEc Archive*, July 2017.

- BuiltIn (2020). Available at: https://builtin.com/data-science/random-forest-algorithm (Accessed 12th May, 2021)
- Chakraborty N. and Biswas S., (2018) "A Statistical Approach to Adult Census Income Level Prediction," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 207-212
- Chockalingam, V., et al (2017). Income Classification using Adult Census Data (CSE 258 Assignment 2).
- Golpour, P et al. (2020) "Comparison of Support Vector Machine, Naïve Bayes and Logistic Regression for Assessing the Necessity for Coronary Angiography." International journal of environmental research and public health vol. 17,18 6449.
- Lazar, A. (2004) "Income prediction via support vector machine," 2004 International Conference on Machine Learning and Applications, 2004. Proceedings, pp. 143-149
- Levy, J.J., O'Malley, A.J. (2020)Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning. BMC Med Res Methodol **20,** 171
- Tutorialspoint (2021). Available at: https://www.tutorialspoint.com/machine_learning_with_python/classification_algorithms_support_vector_machine.htm (Accessed 12th May, 2021)