Classifying Individuals Income Based on Census Data

Bimochan Khadka
Bipashna Kshetree

September 2021

Abstract

In this report, we have presented the details about the classification task that we performed using different machine learning algorithms like C5.0, Random Forests, XG Boost and Classifium with adult data set taken from the UCI repository. The goal of our project was to visualize different approaches used for classification in machine learning. We performed cross validation, boosting and hyper parameter optimization for achieving highest performance of algorithm in terms of accuracy. Also, for same reason, we did Exploratory Data Analysis for data preprocessing.

1 Introduction

Machine Learning can be defined as the ability of a machine by which it can learn itself by given data and improves its performance while performing the same kind of task in future based on its learning experience. The overall machine learning methodology is classified into three categories: Supervised learning, Unsupervised Learning, Semi-supervised learning and Reinforcement Learning[11].In this project we will use supervised learning, specifically classification algorithms.Classification is the method of determining the group of data or a set of data from a set of categories [12].
In this report, we are classifying whether an individual's yearly income will exceed a given threshold or not based on his provided attributes like age, work class, education, marital status, race, native-country and et cetera. We will use different classification algorithms like C5.0, Cubist, Random Forests, xgboost and Classifium. We will use the 'adult' data set which is taken from the UCI repository [4].

2 Related Work

FP Calmon, et al. used the adult data set to develop an algorithm of data pre-processing for reducing discrimination [2].In their conclusion, they stated that their proposed approach of transforming data led to an unclear classification as compared to the original data sets.
S Rossett did a research to use Area Under the Curve(AUC) as a tool for model selection and discrimination.He took the first ten variables from the adult data set and compared the performance of Naive Bayes models using different subsets of those ten predictors. He used a total of seventeen thousand cases as a training set and tested the models on 100 test data. The result of his research illustrated the usefulness of AUC as a stable classification evaluation measure [9].
R Kohavi proposed a new algorithm called NBTree in his paper. He used adult data set to compare the accuracy of this new algorithm with Naive Bayes and C4.5.In his final remarks, he mentioned that the NBTree works well on real data sets and induces highly accurate classifiers in practice.

2.1 Dataset

The 'Adult' data set is a multivariate data set containing 48842 instances. The data was collected by Barry Becker from the 1994 census data. The data set consists of 14 attributes and a target value. On the basis of these 14 attributes we have to classify the data into two groups: firstly,income is more than $50,000 per year and secondly, income is less than or equal to $50,000 per year. The attributes are both categorical and numerical and consists of missing values. There are 14 attributes in the 'Adult' dataset and a target variable 'Income'. The 'Age', 'Fnlwgt', 'Education num', ' Capital gain', 'capital loss', 'hours per week' are continuous in nature while 'workclass', 'education', 'maritalstatus', occupation', 'relationship', 'race', 'sex', 'native-country' are ordinal in nature. In addition to this, by observation we found out that the attributes 'workclass', 'native-country' and 'occupation' consist of missing values.

2.2 Data Preprocessing

The 'ADULT' dataset consists of 52 duplicate instances and 6465 number of missing values. The 52 duplicates data are dropped and the missing values are replaced by the mode of respective columns.

	AGE	FNLWGT	EDUCATION NUM	CAPITAL GAIN	CAPITAL LOSS	HOURS PER WEEK
count	48842	48842	48842	48842	48842	48842
mean	38.643585	1896641	10.078089	1079.067626	87.502314	40.422382
std	13.710510	105604	2.570973	7452.019058	403.004552	12.391444
min	17	12285	1	0	0	1
25%	28	117550.5	9	0	0	40
50%	37	178144.5	10	0	0	40
75%	48	237642.0	12	0	0	45
max	90	1490400	16	99999	4356	99

Table 1: Table illustrating distribution of continuous data

From table-1, we can see that the value of age varies from 17(lowest) to 90(highest). The mean age value is 38, there is no significant difference between mean and median. The third and the fourth quartile show that the distribution is right skewed. Similarly, final weight is rightly skewed showing huge distance gap between median and large value as compared to min and median value. Capital-gain shows that either a person has no gain or has gain of very large amount(10k or 99k). Most of the "capital-loss" values are centered on 0 and only few are non zero(2282). This attribute is similar to the capital-gain i.e. most of the values are centered on 0(nearly 43000 of them). His attributes indicates the number of hours an individual spends working. The lowest working hour is one hour per week and highest working hour is 99 hours per week. By observing the quartiles, we can see that the data is approximately symmetric. The hours per week attribute varies within the range of 1 to 99. By observation,30-40 hrs people work per week,around 27000 people. There are also few people who works 80-100 hours per week and some less than 20 which is unusual.
Similarly, in 'workclass' attribute most of the data belonged to private workclass, in 'education' attribute higher studies graduate has most of the data, around 33% while pre-school has the minimum. In 'marital-status', Married-civ-spouse has the maximum data while Married-AF-spouse is the least. Similarly, most of the population almost 86% lives in the United States and the frequency of male population is higher than that of female and in 'Race' category most of the population is white. From the correlation analysis, we found out that the attribute 'fnlgwt' is highly uncorrelated, so we can drop this table as it will have no impact in prediction score. In addition to this, we can see that the attributes 'age', 'hours per week and 'Education num' is somewhat correlated with 'Income'.Similarly, 'Education num' and 'Education' features are similar to one another so we will use only the

'Education-num' attributes. In 'Income' attributes we replaced the '<=50K' with '0' and '>50K' with '1' and in 'gender' attribute we replaced 'Male' with '0' and 'Female' with '1'. We used one hot encoding technique of the pandas library to change the categorical data into the binary form.
From the correlation analysis, we found out that the attribute 'fnlgwt' is highly uncorrelated, so we can drop this table as it will have no impact in prediction score. In addition to this, we can see that the attributes 'age', 'hours per week and 'Education num' is somewhat correlated with 'Income'.Similarly, 'Education num' and 'Education' features are similar to one another so we will use only the 'Education-num' attributes. In 'Income' attributes we replaced the '<=50K' with '0' and '>50K' with '1' and in 'gender' attribute we replaced 'Male' with '0' and 'Female' with '1'. We used one hot encoding technique of the pandas library to change the categorical data into the binary form.

3 Algorithms

3.1 C5.0

C5.0 algorithm is an updated version of C4.5 algorithm and is suitable for big data set. C5.0 algorithm is better than C4.5 in terms of speed, memory and efficiency [8]. Furthermore, C5.0 algorithm can identify which attributes are relevant or not in the classification and have solved the problem of over fitting and error pruning [6].
In this project, we have used C5.0 algorithm to classify our instances into two classes. In order to achieve this, we have used 70-30 split, that is, seventy percent data is used for training the model and the rest thirty percent is used for testing the model. After running the C5.0 algorithm successfully, we analyzed that our model had yielded 89.2 percent accuracy in the training set generating 11.8 percent error rate and in the test data 86.5 percent data were classified accurately generating 13.5 percent error rate.

3.2 k Fold Cross Validation

Cross validation is a computer intensive procedure utilizing all accessible cases as preparing and test examples[1]. Most hypothesis appears that there exists no universal fair-minded estimator of the fluctuation of K-fold cross-validation[1]. K-fold cross validation strategy is to induced the much accurate model. Here entire information is partitioned into 'K' case where 'K-1' case represents the training data and the remaining data which is K-(K-1)

case represents the testing data(hold out data)[5].The model training accuracy with model 10 round is found to be 86.9 percentage with decrease in the value of k with model 5 round the training accuracy rose by 0.1 percentage reaching to 86.8 percentage.

3.3 Boosting

Boosting is characterised as a strategy for improving the performance of any given machine learning algorithm[10]. In Boosting method,initial classifier is trained with data set, and those data sets which are unclassified during the first iteration of training are assigned extra weight [10]. In the next round of training new classifier is generated with the data sets including those data having extra weight. During this process, classifier can distinguish the previously unclassified data and pay extra consideration while classifying them. This process is continued until the number of rounds we assign to the algorithm. In our experiment we implemented the boosting strategy to improve the efficiency of our model with the same data set. The result is presented in table 1 below.

Strategy:	5-fold CV	10-fold CV	Boosting
Accuracy:	86.8	86.9	89.1

Table 2: Table illustrating the results from k-fold and boosting technique.

3.4 Random Forest

Random forest is one of the type of ensemble learning algorithm which make predictions by calculating the average predictions over several independent base models[3]. It is built by combining the predictions of several trees, each of which is trained in isolation[3].

3.4.1 Hyper parameter tuning in Random Forest

The Random Forest algorithm has several hyper parameters which needs to be set by the user. For example: the number of observations drawn randomly for each tree and whether they are drawn, with or without replacement, total count of variables drawn randomly for each split, the minimum number of samples that a node must contain, and the number of trees[7]. We used a grid search with a limited range for the tuning process as illustrated in the

table below to tune the random forest model.

Hyperparameters	Values
n_estimator	70, 80, 90, 100
max_features	'auto', 'sqrt'
max_depth	100, 50, 150, 300
min_samples_leaf	1, 2, 4
min_samples_split	2, 5, 10
bootstrap	False, True

Table 3: Table illustrating hyper-parameter values for tuning Random Forest model

3.5 XGBoost

XGBoost stands for extreme gradient boosting machine. It was proposed by Tianqi Chen in 2016. In random forest entropy is calculated to perform best speed and to form decision tree whereas in XGboost gradients of training data are calculated. In XGBoost loss function is defined such that it minimises the difference between actual and predicted target values. Tree Boosting is highly effective and broadly utilized machine learning method.

3.5.1 Hyper parameter tuning in XGBoost

Machine Learning algorithm have been used widely. In order to fit machine learning model into different problems, its hyper parameters must be tuned. Experiments are conducted on standard data sets to compare the performance of different improvised methods and provide practical examples of hyper parameter optimisation[13]. We used a grid search with a limited range for the tuning process as illustrated in the table below to tune the xgboost model.

3.6 Classifium

Classifium is automatic machine learning algorithm. It has its own set of algorithms for data pre-processing and for hyper parameter tuning.

Hyperparameter	Values
learning_rate	0.05, 0.1, 0.15
max_depth	3,10,15
min_child_weight	1, 3, 5, 7
gamma	0.0, 0.1, 0.2
colsample_bytree	0.3, 0.4, 0.5

Table 4: Table illustrating hyper-parameter values for tuning XGboost model

4 Result

In this section, the results from the methods explained in section 5 are highlighted.

4.1 C5.0

In c5.0 algorithm, we use 70-30 data split. The c5.0 algorithm achieved the prediction accuracy of 87.5% and an error of 12.5% in the training data-set whereas the prediction accuracy for testing set was of 86.6% with an error of 13.4%
By using k-fold validation, the mean accuracy of the model with k equals to 10 is found to be 86.8 percentage and with decrease in the value of k to 5, yields the mean accuracy of 86.8 percentage.
While implementing the boosting technique in the c5.0 model, the mean error of the model decreased to 11.1%, yielding the mean accuracy of 88.9%.

4.2 Random Forest

Here we implemented grid search for tuning the random forest model. We split 50% of data into training set and used 5 fold cross validation to find the best estimator for the model. The best estimator for the model was found with parameters criterion='entropy', max_depth=50, min_samples_leaf=2, n_estimators=80, oob_score=True, random_state=42. We obtained the cross validation training score as 86.34% and the test accuracy of the model is found to be 85.75%.

4.3 XGboost

Here we implemented grid search for tuning the xGboost model. We split 50% of data into training set and used 5 fold cross validation to find the best

estimator for the model. The best estimator for the model was found with parameters base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.4, gamma=0.2, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.1, max_delta_step=0, max_depth=10, min_child_weight=3, missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact', validate_parameters=1, verbosity=None. We obtained the test accuracy of the model as 86.81%.

4.4 Classifium

In the Classifium algorithm, we did the 50-50 split and the model accuracy in the training set was observed to be 87.44% and in the testing set the accuracy was found to be 87.52%

4.5 Summary

The overall summary of the results are displayed in the table below:

Algorithm	Training Accuracy(%)	Cross Validation Score(%)
C5.0	87.5	86.6
C5.0 with k=5 fold	87.2	86.8
C5.0 with k=10 fold	87.3	86.8
Boosting	88.9	-
Random Forest	97.9	84.22
Hyperparameter tuned RF	86.34	85.75
XGboost	87.64	86.8
Hyperparameter tuned XGboost	87.84	86.81
Classifium	87.44	87.52

Table 5: Table illustrating the model accuracies of different classification algorithms

5 Conclusion and Future Work

Our results show that the accuracy of all the models are somewhat similar. The highest accuracy was obtained from the Classifium algorithm with an accuracy of 87.52% followed by XGBoost with 86.81%. When we tuned the

hyperparameter of XGBoost and the Random Forest model, we found that the sequence of tuned parameters could influence the mean absolute errors of the model. However, due to time limitation we had to limit our tuning parameters. So, XGBoost might get a better result if we can tune the hyperparameters in different sequence. Similarly, in the random forest algorithm without specifying the hyperparameters, the training accuracy was obtained as 97.9% while the testing accuracy was found to be 84.22%. This clearly indicates the model is over-fitted. As the 'Adult' data-set is highly imbalanced that could be the main reason for this under-fitting problem. From our result we can see that the Random Forest algorithms performs the least. However, it is naive to say that the Random Forest performs least in all cases. We may increase the mean absolute accuracy of this model by better tuning the hyper parameters and by implementing different hyper parameter search mechanisms such as grid search, random search, Bayesian optimization and so on.
Using the detail of the adult data set found in UCI Machine Learning Repository, we may learn to predict the monthly income of an individual utilizing the attributes present. In addition to this, we can use the adult data set to determine the gender wage gap by adding some other attributes. Furthermore, the adult data set can be used to give career advices to the individuals as well as can be used to predict whether an individual can sustain financially in the coming years based on their current salary. For the future work, we can implement neural network models using adult data set and compare the results.

References

[1] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105, 2004.

[2] Flavio P Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3995–4004, 2017.

[3] Misha Denil, David Matheson, and Nando De Freitas. Narrowing the gap: Random forests in theory and in practice. In *International conference on machine learning*, pages 665–673. PMLR, 2014.

[4] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[5] Douglas M Hawkins, Subhash C Basak, and Denise Mills. Assessing model fit by cross-validation. *Journal of chemical information and computer sciences*, 43(2):579–586, 2003.

[6] Rutvija Pandya and Jayati Pandya. C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning. *International Journal of Computer Applications*, 117(16):18–21, 2015.

[7] Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019.

[8] R Revathy and R Lawrance. Comparative analysis of c4. 5 and c5. 0 algorithms on crop pest data. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(1):50–58, 2017.

[9] Saharon Rosset. Model selection via the auc. In *Proceedings of the twenty-first international conference on Machine learning*, page 89, 2004.

[10] Robert E Schapire. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999.

[11] Wikipedia contributors. Machine learning — Wikipedia, the free encyclopedia, 2021. [Online; accessed 18-September-2021].

[12] Wikipedia contributors. Statistical classification — Wikipedia, the free encyclopedia, 2021. [Online; accessed 18-September-2021].

[13] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

Evaluation of Individuals Income Based on Adult Dataset Using Neural Network

Bipashna Kshetree
Bimochan Khadka

October 2021

Abstract

In this report, we have used neural network model for the classification task with the adult dataset taken from the UCI repository. The main objective of this project is to find the right encoding technique, describing methodologies, implement different neural network algorithms with varying hyperparameters and building optimized neural network model that can perform classification task efficiently.

1 Introduction

Artificial Neural Network is a network of neurons or hubs or units associated with other neurons to exchange the information or data from each other[11].It can adjust to changing input; so the network creates the finest conceivable result without requiring to update the result criteria. Study of continuous valued functions using neural network can give a better precision, reliable estimation of the generalization error, and active learning[5].Neural Network consists of input layer of neurons, one or two or many more hidden layers of neurons are also shown in figure below:

The architecture shows a typical architecture where lines connecting neurons are shown.Neural network constructed the way above diagram can estimate any computable function to an arbitrary precision. Input neurons numbers are independent variables and the numbers returned from output neuron are dependent variables to the function being estimated by neural network[11].

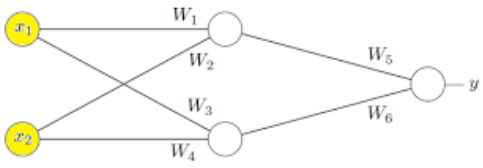

Figure 1: A Neural Network Diagram

1.1 Related Work

HYONTAI SUG used the 'Adult' dataset and the 'Forest Cover' dataset to compare the performance of multilayer perceptrons network and radial basis function networks with respect to training data set size in classification tasks[10]. The author took 200 instances as initial sample size for training and the sample size is doubled each time with 'while' loop until the sample size reached to about half of the dataset size. The rest of dataset left after sampling was used for testing. The author found out that the performance of RBF networks are good when the size of training data set is relatively small, but the performance of MLPs are good when the size of training data set size is relatively large [10].

Bing Yu implemented the LSTM-Casper on the 'Adult' dataset to solve the classification task. LSTM-Casper is a powerful neural network technique that combines Long Short-Term Memory and Casper network [12]. The author implemented 10 fold cross validation technique and Area under curve of Receiver Operating Characteristic (AUC) method to train the model and then compared the accuracy of other classification technique applied on adult dataset like Casper, Naïve Bayes and 3-layer feedforward neural network with the LSTM-Casper. AUC - ROC curve is a performance measurement for classification problem. ROC is a probability curve and AUC represents degree or measure of separability[7].The author concluded that in terms of accuracy, when 10 fold cross validation was implemented,the LSTM-Casper performed better than other models. However,while comparing in terms of AUC value the trained LSTM-Casper model is not performing very well [12].

Laurent Risser et. al proposed a new method to temper the algorithmic bias in Neural-Network based classifiers using the 'Adult' dataset. Their method

is Neural-Network architecture agnostic and scales well to massive training sets of images. Unlike traditional neural architecture search methods, where all of the weight parameters of new architectures need to be trained using a learning algorithm, the key idea behind weight agnostic neural networks is to search for architectures by de-emphasizing weights [3]. The authors found out that in this method, the only overloads the loss function with a Wasserstein-2 based regularization term whose gradient can be computed at a reasonable algorithmic cost which makes it possible to use regularised loss with standard stochastic gradient-descent strategies. The authors concluded that the key advantage is that their strategy can be implemented on any neural network architecture [9].

1.2 Objective

The objective of this project is to implement neural nets to perform classification task. The scope of this project revolves around finding the right encoding techniques, describe methodology, implement different neural network algorithms with varying hidden layers and building optimized neural network model which can classify our target class.

2 Hyperparameters in Neural Network

2.1 Number of Hidden Layers

In neural networks, hidden layers lies between the input layer and the output layer where the neurons take in a set of weighted inputs and produce an output through an activation function. It works like the biological neuron of the human brain; it takes in weighted inputs, works on them using activation function and provides the output.

2.2 Number of Neurons

An artificial neuron is similar to that of a biological neuron. Selection of the number of neurons in different layers of an artificial neural network (ANN) is a key decision-making step involved in its successful training. The number of neurons in the hidden layer is not fixed, whereby the overall efficiency of the Artificial Neural Network (ANN) depends upon the correct modeling of neural network whereas the number of neurons in input layers depends upon the number of input parameters and the number of output parameters respectively [1].

2.3 Activation Function

Activation function is one of the building block on neural network.When our brain is fed with a lot of information, it differentiates the information into beneficial and not so beneficial information.
Sigmoid function is one of the most widely used non linear activation function. It transforms the values between the range 0 and 1. It is S-shaped function and is continuously differentiable. It is used in output layer.
Rectified Linear Unit is another non-linear activation function. It does not activate all the neurons at a time. This means neurons will only be deactivated if output of linear transformation is less than zero.
Scaled Exponential Linear Unit induce self normalizing properties. It is defined as:

- if x $>$ 0: return scale * x
- if x $<$ 0: return scale * alpha * $(exp(x) - 1)$

where, alpha=1.67326324 and scale=1.05070098
The **Exponential Linear Unit** with alpha $>$ 0 is: x if x$>$ 0 and alpha * (exp(x) - 1) if x $<$ 0.The ELU hyperparameter alpha controls the value to which an ELU saturates for negative net inputs.
Softmax converts a vector of values to a probability distribution. The elements of the output vector are in range (0, 1) and sum to 1.
Leaky_Relu allows a small gradient when the unit is not active. It is given as:
f(x) = alpha * x if x $<$ 0
f(x) = x if x $>=$ 0

2.4 Loss Function

The main objective while training the neural network is to minimize the errors. The function used to minimize the errors are termed as the loss function [4]. The loss function must be chosen to calculate the error of the model during the optimization process [8]. The choice of loss function is directly related to the activation function used in the output layer of neural network and the nature of the prediction problem [4]. Since, our classification task is binary classification, we have used binary_crossentropy as the loss function.

2.5 Optimizer

In neural network, optimizers are the algorithms used to update the parameters of the network such as weights to minimize the losses. To train the network such that it performs better on the unseen datasets, we need to take the help of the loss, that is, the main objective is to minimize the loss, as lower loss implies that the neural network model is going to perform better. The main job of the optimizer is to solve the optimization problems by minimizing loss. There are many gradient based optimizers but we have explained only those optimizers that we have used in our model.
In **Stochastic Gradient Descent** we do not have to wait to update the parameter of the model after iterating all the data points in our training set. Instead we just update the parameters of the model after iterating through every single data point in our training set. It will learn the optimal parameter of the model faster which then in terms minimize the training time.
Adam(Adaptive Gradient Descent) is the most widely used optimizer in deep learning. Adam takes the advantage of both RMSprop (to avoid a small learning rate) and Momentum (for fewer oscillations). In adam, the focuses is on computing the running average of the squared gradients as well as computing the running average of the gradients. **Nesterov-accelerated Adaptive Moment Estimation** is an extension of the Adam algorithm that incorporates Nesterov momentum and can result in better performance of the optimization algorithm.A limitation of gradient descent is that the progress of the search can slow down if the gradient becomes flat or large curvature. Momentum can be added to gradient descent that incorporates some inertia to updates. This can be further improved by incorporating the gradient of the projected new position rather than the current position called the Nesterov momentum.
Adamax is basically a variant of Adam; which is an algorithm for first-order gradient-based optimization of stochastic objective functions. Adam is based on adaptive estimates of lower-order moments.
RMSProp is a gradient based optimisation technique used in training neural network.The utilization of RMSProp(Root Mean Square Propagation) makes it possible to understand and study multi-layer neural network.
Adadelta is a novel per-dimension learning rate method for gradient decent. The promising results are shown compared to other methods on the MNIST digits classification task. This method do not demand manual tuning of a learning rate and selection of hyperparameters[13].
Adaptive Gradient methods (Adagrad) in particular, both theoretical results and extensive numerical experiments means that Adagrad-Norm is

robust to the unknown lipschitz constant and level of stochastic noise on the gradient.
FTRL is a generalised linear model is also a proximal online learning algorithm. FTRL has a outstanding exiguity and convergence properties as well. It also uses the per-coordinate learning rate mechanism.

2.6 Metrics

A metric is a function that is used to measure the performance of the model. Metrics are different from loss function as loss function show a measure of model performance and are usually different in the model's parameter whereas metrics are used to monitor and measure the performance of a model and do not need to be differentiable. There are dozens of metrics for classification task, but for this project we have chosen accuracy as the metrics. Accuracy is the fraction of predictions our model got sight. For binary classification, Accuracy=(TP+TN)/(TP+TN+FP+FN) Where TP+TN=No. of predictions, also TP+TN+FP+FN=Total No. of predictions.

2.6.1 Epochs

Epoch refers to one cycle through the full training dataset. It gives the network a chance to see the previous data to readjust the model parameter's so that the model is not biased towards the last few data points during training which in terms provides better generalisation when given a new unseen data.

2.6.2 Batch-size

The number of training examples utilized in one iteration is termed as Batch-size. In other words, a batch size of 64 means that 64 samples from the training data set will be used to estimate the error gradient before the model weights are updated.

3 Methodology

3.1 Encoding

Our model requires all input and output variables to be numeric [4]. In other words, we have to encode our categorical data to numerical before we can fit and evaluate our model. We used the OneHotEncoder method of the

scikit-learn library to transform our categorical data to numeric. The one hot encoding technique maps each label to a binary vector. The following figure explains the basic principle of the one hot encoding technique.

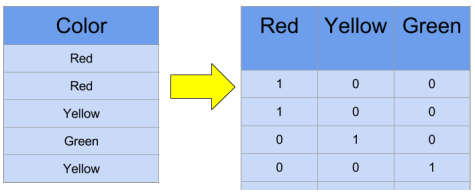

Figure 2: OneHot Encoding

3.2 Splitting Data

Before training the model, we have to split the data into training set, validation set and test set to avoid the overfitting problem. We split 60% of our dataset to training set, 20% to validation set and the rest 20% to test set. We trained our model with 60% of data, and performed cross validation and hyperparameter optimization using validation set of data. We tested our model built with optimized hyperparameters using unseen testing data set at the end.

3.3 Building Initial Model

At first, we simply built a neural network model using keras library and scikit learn library. We checked for overfitting problem of the model using validation data set. The hyperparameters used during the initial phase with their corresponding values are represented in the table below.

Hyperparameters	**Value**
Hidden Layers	1
Neurons	10
Epochs	250
Batch_size	64
Activation	adam
Optimizer	relu

Table 1: Initial Hyperparameters

3.4 Hyperparameter Optimization

We tried to optimize the hyperparameters to solve the overfitting problem and to increase the accuracy of our model. For doing this, we followed the step-by-step manual approach. The followed steps were followed to tune the hyperparameters.

1. Number of hidden layers and neurons.
2. Number of epochs and batch size.
3. Tune the activation function
4. Tune the optimizers

3.4.1 Tune the number of hidden layers and neurons

Initially, our model was randomly set to one input layer, one hidden layer and one output layer. The number of neuron was selected as 10 and the activation function used is rectified linear unit and the optimizer is 'adam'. The number of epochs was set to 250with batch-size 64. We obtained the mean accuracy of training set as 0.85182 and of validation set as 0.89592. We then add another hidden layer with the same condition as before and checked the result and evaluate the results. The training accuracy was increased to 0.85322 and validation accuracy to 0.84926. We then add another hidden layer and evaluate the accuracy which is found to be 0.85104 and 0.84646 repectively. Since there was a right change in accuracy we set for hidden layer=2.Next, we increased the number of neurons to 20, on each layer and noticed accuracy to be 0.855144 and 0.849101 respectively. Next, we decreased the number of neurons to 5 and noticed accuracy to be 0.841132 and 0.83922 respectively.The result are all listed on table below:

Hidden Layer	**Neurons**	**Training Acc**	**Validation Acc**
1	10	0.85182	0.84592
2	10	0.85322	0.84926
3	10	0.85104	0.84548
2	20	0.855144	0.849101
2	8	0.84384	0.83756
2	5	0.841132	0.83922

Table 2: Tuning hidden layers and neurons

The best score we obtained is when hidden layer=2 and number of neurons = 10.

3.4.2 Number of epochs and batch-size

Epochs	**Batch Size**	**Training Accuracy**	**Validation Accuracy**
250	64	0.855422	0.84926
200	64	0.84985	0.843157
300	64	0.851631	0.848927
	128	0.8515	0.846229
	32	0.852161	0.847578

Table 3: Tuning Epochs and Batch Size

Initially, we set the batch size to 64 and epochs to 250. First we try to tune the number of epochs by setting the batch size constant at 64. We then set the number of epochs to 200 and 300 and notice the accuracy. The results are obtained in the table above. Next, we set the epoch to 300and determine the batch size.The batch size was taken in the form of 2^n. The best result was obtained with epoch 300 and batch size 128.

3.4.3 Optimizing Activation Function

In this step, we try too find the best activation function for our model. For this we use a number of activation functions. All the other hyperparameters were set to previous values. The activation functions and their corresponding accuracy are listed in the table below.

Activation	Train Acc.	Val Acc.	Train loss	Val loss
relu	0.85128806	0.84763947	0.32767634	0.33605489
elu	0.84000113	0.83688197	0.37825552	0.33560588
selu	0.85406108	0.85015354	0.31213355	0.3306159
softmax	0.76962384	0.77463984	0.53196522	0.52509746
LeakyRelu	0.84993379	0.84609546	0.32214496	0.33337199

Table 4: Table showing Different Activation Funtion

From the above table, we can observe, that while using activation functions 'relu' and 'selu' generate somewhat similar results. However, the ac-

curacy of the model while using 'selu' is slightly higher. So, we use 'selu' as an activation funtion for our model.

3.4.4 Optimizing optimizers

We use different optimizers and try to find the best fit for our model. To do this, we used optimized values of all the other hyper-parameters obtained from previous methods and tried to find the best optimizers for our model, among eight different optimizers.

Optimizer	Training Acc.	Validation Acc.	Training loss	Validation loss
adam	0.85406108	0.85015354	0.31213355	0.3306159
nadam	0.85313507	0.84979185	0.32954871	0.32855955
SGD	0.75945984	0.76578974	0.56658826	0.54459493
RMSprop	0.85258784	0.84720715	0.31840325	0.34047651
Adadelta	0.74057897	0.74454293	6.72782209	7.29084942
Adamax	0.84875652	0.84466577	0.32586419	0.33322938
Adagrad	0.81474879	0.80884805	0.3754761	0.38320046
Ftrl	0.81319183	0.8133074	0.47882802	0.41708393

Table 5: Tuning Optimizers

From the above table we can see that model can get somewhat similar accuracy while using optimizers,'adam', 'nadam' and 'RMSprop'. However, among them we have selected 'adam' as the tuned optimizer for our model as it has slightly better accuracy than the other two optimizers. The best combination for our model is listed in the table below:

Hyperparameter	**Value**
Hidden Layers	2
Neurons	10
Epochs	300
Batch_size	128
Activation	selu
Optimizer	adam

Table 6: Best Hyperparameters Combination

Using above combination, we used those values to test the performance of our model on testing data set. The accuracy of our model was 84.814%.

The following graphs represent the accuracy and loss of test datasets and training datasets in each epochs.

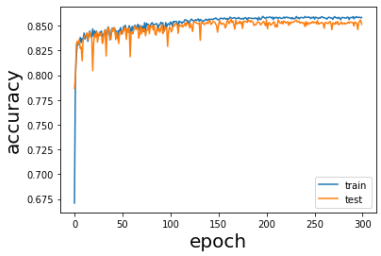

Figure 3: Train-Test Accuracy

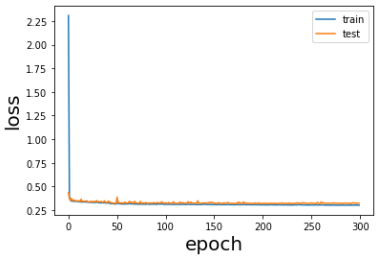

Figure 4: Train-Test Loss.

3.5 Evaluation

In this section, we discuss our findings and try to evaluate the results. There was a problem of overfitting In our first approach, even though, the accuracy of the model was somewhat similar to that of our final model, we could not test the model for our test data set because of the problem of the overfitting. The following picture represent the issues perfectly.

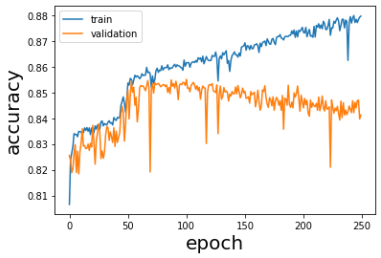

Figure 5: Initial Train-Validation Accuracy

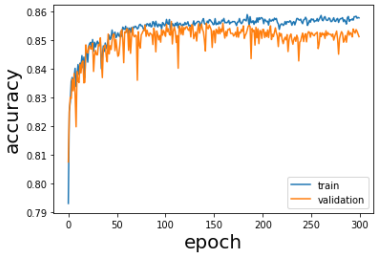

Figure 6: Train-Validation Accuracy after tuning

The figure above shows that our initial model suffers from overfitting and classification cost problem. To tackle this issues, we performed the hyperparameter tuning of our models. We first start by tuning number of hidden layers and number of neurons on each hidden layer. Since, the adult dataset is highly imbalanced dataset, initially there was a problem of differing classification cost. The main reason for differing classification cost and overfitting is due to highly imbalance dataset and that the initial model suffers from low bias and high variance. These problems were solved by using

different hyperparameters combination (using trial and error approach). The solution can be seen in figure-6.

3.5.1 Confidence Interval

A confidence interval is a bounds on the estimate of a population variable. It is an interval statistic used to quantify the uncertainty on an estimate [6]. In the case of classification accuracy, the radius of the interval can be calculated as:
interval = z * sqrt((accuracy * (1 - accuracy)) / n) where,
interval is radius of the confidence interval, accuracy is classification accuracy, n is the size of the sample, sqrt is square root function, and z is the number of standard deviations from the Gaussian distribution.
We calculate the interval at 95% level of confidence and got the interval of 0.007. This signifies that we can make claim that the classification accuracy of the model is 85.814% +/- 0.7%.

4 Comparison and criticism of Neural Network

Neural Network are trained in order to solve fix category of problems. Their learning ability makes neural network well able to solve and tackle problems[2].Based on our observation and comparison of neural network algorithm with classification tree the architecture of neural networks is more complex than decision tree model. While the decision tree have simpler architecture and contains nodes, branches and leaves the neural network have interconnected layers. We can not derive rules from neural network algorithm whereas we may simply derive rules from the architecture of decision tree.

- Time : Neural network model generally take the long time for training in comparision to decision tree. In tree based algorithm data set and hyperparameters are the only factors which can increase the training time of model.
- Accuracy : On our data set decision tree yielded more level of accuracy.Neural network algorithm demands excellent hardware and software and yet yields high efficiency. However neural network algorithm are more accurate than the decision tree based algorithm when we classify image,text, and audio data.
- Configuration :Neural Network model are more complex and demands standard set of hardware and software whereas decision

tree model only need to declare initial model architecture. From decision based classifier we can derive the classification rule.

5 Conclusion and Future Work

In this project, we implemented neural network in the adult dataset. We found that without hyperparameter optimization, there will be an issue of overfitting and differing classification cost. With proper hyperparameter optimization these problems were properly addressed. The best accuracy was obtained when the number of hidden layers is 2, number of neurons is 10, epochs is 300, batch_size is 128, activation function is 'selu' and the optimizer is 'adam'. The overall test accuracy obtained is found to be 84.814%. This accuracy is the least compare to other algorithms implemented in our first project (C5.0, Random Forest, XGBoost, and Classifium). Neural Network model has slightly more complex architecture than the other algorithms. However, it is useful in classifying text, images and speech.

For the future work, we could have implement regularization technique such as dropout in our model. Similarly, we could have implemented other approach for hyperparameter optimization such as implementing RandomizedSearch or Grid Search for model selection. We could also implement kerastuner for hyperparameter optimization and compare the results.

References

[1] Mohammad Adil, Rahat Ullah, Salma Noor, and Neelam Gohar. Effect of number of neurons and layers in an artificial neural network for generalized concrete mix design. *Neural Computing and Applications*, pages 1–9, 2020.

[2] Tony Florio, Stewart Einfeld, and Florence Levy. Neural networks and psychiatry: candidate applications in clinical decision making. *Australian and New Zealand journal of psychiatry*, 28(4):651–666, 1994.

[3] Adam Gaier and David Ha. Weight agnostic neural networks. *arXiv preprint arXiv:1906.04358*, 2019.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[5] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Neural Information Processing Systems 7*, pages 231–238, 1995.

[6] William Q Meeker, Gerald J Hahn, and Luis A Escobar. *Statistical intervals: a guide for practitioners and researchers*, volume 541. John Wiley & Sons, 2017.

[7] Sarang Narkhede. Understanding auc-roc curve. *Towards Data Science*, 26:220–227, 2018.

[8] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[9] Laurent Risser, Quentin Vincenot, and Jean-Michel Loubes. Tackling algorithmic bias in neural-network classifiers using wasserstein-2 regularization. *arXiv preprint arXiv:1908.05783*, 2019.

[10] Hyontai Sug. Performance comparison of rbf networks and mlps for classification. In *Proceedings of the 9th WSEAS International Conference on applied Informatics and Communications (AIC'09)*, pages 450–454, 2009.

[11] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.

[12] Bing Yu. The performance of lstm-casper neural network with classification task on adult dataset.

[13] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.