INTRODUCTION

Two data sets were given for this data analytics project consisting of a training set and a test set.The training set had 35352 rows consisting of the same attributes as found in Assignment 2. The test set had 14400 rows with the same attributes as the training set except the “Salary” column was missing. A requirement for this project was to use the training set to train a data analytics model then apply the model to the test set so it can predict the salary since it is unknown. The two possible values for the Salary, which is a nominal attribute is either ‘<=50K’ or ‘>50K’.

DATA PREPROCESSING AND TRANSFORMATIONS

Data preprocessing was carried out as required and ones chosen differed depending on the model chosen which has been described further below:

I. DE-DUPING

De-duping the dataset will result in removing duplicate values and can be applied using the ‘GroupBy’ node in KNIME. By de-duping the dataset we can ensure that only unique rows will be used for the analysis. However, de-duping decreased the accuracy so the final input was not de-duped.

II. MISSING VALUES

NEURAL NETWORK

The dataset had several missing values in varying columns in both the training set and the test set. The ‘Missing Value’ node was applied on KNIME to handle the missing values where the numerical attributes with missing values were filled with the mean and the string attributes were filled with ‘N/A’. This was done to ensure the rows with missing values were used appropriately as part of training the model.

DECISION TREE

For Decision Tree, the missing values were handled individually for each column. For most string columns the values were set to 'most frequent', however for integer values the missing values were set to mean except for age, which was most frequent.

LOGISTIC AND RANDOM FOREST

1. Removing missing values reduces the dataset by a significant factor thus the missing values in the Occupation, Native country and Employment class have been substituted with corresponding most frequent class values. Missing value handling has been implemented using-

```
newcensusdata<-
data.frame(censusdata$Employment.class,censusdata$Occupation,censusdata$Native.co
untry,stringsAsFactors=FALSE)

replacementVals<-c(Employment.class="Private",Occupation="Not-
Listed",Native.country="United-States")

names(newcensusdata)<-c("Employment.class","Occupation","Native.country")

indx1 <-
replacementVals[col(newcensusdata)][is.na(newcensusdata[,names(replacementVals)])
]

newcensusdata[is.na(newcensusdata[,names(replacementVals)])] <- indx1

censusdata$Native.country=newcensusdata$Native.country

censusdata$Occupation=newcensusdata$Occupation

censusdata$Employment.class=newcensusdata$Employment.class

write.csv(censusdata,"newtraining.csv")

newcensusdata<-read.csv("newtraining.csv",na.strings=c(""),header=TRUE)

censusdata<-newcensusdata

censusdata<-censusdata[-1]
```

III. Removing Outliers

Decision Tree

For Decision Tree, the outliers were not removed as removing the outliers affected the training of the Decision Tree Learner. As the decision tree learner is not affected by the outliers and if the outliers are removed the prediction accuracy actually reduces due to the test dataset having various random scenarios and conditions which decision tree needs to be trained for to handle those random scenarios.

IV. Normalisation

Neural Network

Min-max normalisation was carried out to ensure all the values for all numerical attributes were between 0 and 1. This causes the values to become closer to a normal distribution. This processing of cleaning the data helped reduce the noise so the model could be trained more consistently as it will find patterns or relationships more effectively.

Decision Tree

For decision tree, min-max normalization greatly increased the accuracy of the results by reducing the overhead on the predictions

VI. Binarisation

Decision Tree

For Decision Tree implementation, binarization was used on fields having only two types of values for example Salary, where the values were either <=50K or >50K .The binarization actually rendered no different results and had no effect on the overall accuracy of the results so the binarization was removed from the Decision Tree model.

VII. Partitioning

Neural Network

Since SPSS was used for the Neural Network (MLP) model, partitioning the dataset was different compared to KNIME. It involved creating a new column called ‘Partition’ and randomly assigning values to each row where 1 is for Training, 0 is for test, and -1 is for the holdout sample. It was randomly assigned by using the excel formula ‘rand()’ to assign a random integer to each row then sorting the dataset by this column. The first 70% of rows was assigned to training set and the remaining 30% was assigned to the test set. The given test set data was assigned a partition value of -1 for every row assigning it to the holdout set then combined with the training dataset.

DECISION TREE

The partitioning of the training dataset was done 90% for decision tree learner and 10% for checking the accuracy of the prediction result. The training data was further increased in the partition i.e. more than 95% and the accuracy of the predictions improved.

LOGISTIC AND RANDOM FOREST

Training set has been partitioned into training and testing set for testing the model using the 'createDataPartition' package to create a 60\40% (train-test) split of the testing data set to maintain class balance of the 'Salary' variable with random samples created within each class. R implementation of the splitting is-

```
set.seed(123)

library(caret)

split<-createDataPartition(y = censusdata$Salary, p = 0.6, list = FALSE)

training<-censusdata[split,]

testing<-censusdata[-split,]
```

MODELLING

Each person in the group chose one classifier model each, which were a neural network, decision tree, random forest, and Bayesian model.

I. NEURAL NETWORK - MULTILAYER PERCEPTRON (MLP)

Neural networks are based on the biological structure of the human brain and how it processes information. A multilayer perceptron is the most popular type of neural network and is made up of a set of nodes that comprise an input layer, a number of hidden layers, and an output layer (Stergiou & Siganos n.d.). Neural networks are useful since they are able to gain meaning from complex datasets and find patterns or trends which may be too complicated for humans or other techniques to detect. They are able to learn from experience or training as as well automatically configure its own organisation of nodes during training (Stergiou & Siganos n.d.).

Neural networks are good for complex datasets and have been used in the past for financial institutions, marketing companies, health industries etc (Tu 1996). For example, it can be used by financial institutions to predict a person's probability of defaulting on a

loan. This is done by first training the model with a known dataset with customer information such as their age, gender, annual income, whether they previously default etc. The model can learn which attributes contributed to the resulting answer then apply what has learnt to a new dataset where it is unknown whether the customer will default. Neural networks can also be used to predict a patient's length of hospital stay based on their profile as well as what type of procedure they had and whether there were complications (Tu 1996).

One of the main advantages of using a neural network is the fact that an individual could develop a working neural network within days to weeks dependent upon the availability of a data set and neural network software (Tu 1996). Generally, any data set that could be analysed by logistic regression can also be analysed by a neural network. However, users of logistic regression models require formal statistical training which means a greater time investment is required to learn the technique before application can occur. This is a significant advantage since it means any researcher or user could apply a neural network to a data and gain new knowledge from it. This can be especially important since data mining may not be easily adopted and this barrier may be overcome.

Another advantage of a neural network is that they are able to detect indirectly any complex nonlinear relationships between independent and dependent variables. Neural networks can automatically adjust the connection weights if there is a significant nonlinearity in the given variables. It is known that the relationships between different variables in a complex data set can often be complex and nonlinear (Tu 1996). This will allow any and all relationships involved to be taken into consideration without further manual input.

A significant disadvantage of the use of the neural network is that they are a "black box" in nature which although means it is easier to use; there is a limited capability to explicitly identify any causal relationships. This is because a user would set up the training data and effectively allow the neural network to train itself and determine which input variables are most important. This means the model would not be very good if the use requires a good understanding of the inner workings of the model or be able to explain causal relationships found.. However, researchers have been attempting to develop techniques to improve the understanding of the internal logic in a neural network such as removing input variable nodes on at a time to observe changes in performance (Tu 1996).

Input Layer	Factors	1	Age
		2	Employmentclass
		3	Educationlevel
		4	Educationyears
		5	Maritalstatus
		6	Occupation
		7	Relationshipstatus
		8	Race
		9	Sex
		10	Capitalgain
		11	Capitalloss
		12	Workhoursperweek
		13	Nativecountry
	Number of Units[a]		477
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		15
	Activation Function		Hyperbolic tangent
Output Layer	Dependent Variables	1	Salary
	Number of Units		2
	Activation Function		Softmax
	Error Function		Cross-entropy

a. Excluding the bias unit

Figure 1. Network Information (Output from SPSS)

Cross-entropy was selected as the error function since this model is required to predict a classification of salary. The model automatically selected 477 units in the input layer, 15 units in a single hidden layer, and 2 units in the output later.

Training	Cross Entropy Error	6294.271
	Percent Incorrect Predictions	12.3%
	Stopping Rule Used	Maximum number of epochs (100) exceeded
	Training Time	0:01:10.01
Holdout	Percent Incorrect Predictions	13.2%

Dependent Variable: Salary

Figure 2. Model Summary (Output from SPSS)

The model summary in Figure 2 shows the the model or estimation algorithm stopped training about 1 min and 10 seconds after 100 epochs was exceeded. The percent of incorrect predictions is also roughly the same between the training and holdout set, which is a positive sign.

Sample	Observed	Predicted <=50K	Predicted >50K	Percent Correct
Training	<=50K	16917	1024	94.3%
	>50K	1876	3735	66.6%
	Overall Percent	79.8%	20.2%	87.7%
Holdout	<=50K	7121	500	93.4%
	>50K	829	1594	65.8%
	Overall Percent	79.2%	20.8%	86.8%

Dependent Variable: Salary

Figure 3. Classification (Output from SPSS)

The classification table in Figure 3 shows the model was significantly better at predicting <=50k for Salary compared to >50k for both the training set and holdout set. The overall percentages of correct predictions for the training and holdout set are 87.7% and 86.8% respectively which means it is a relatively accurate model.

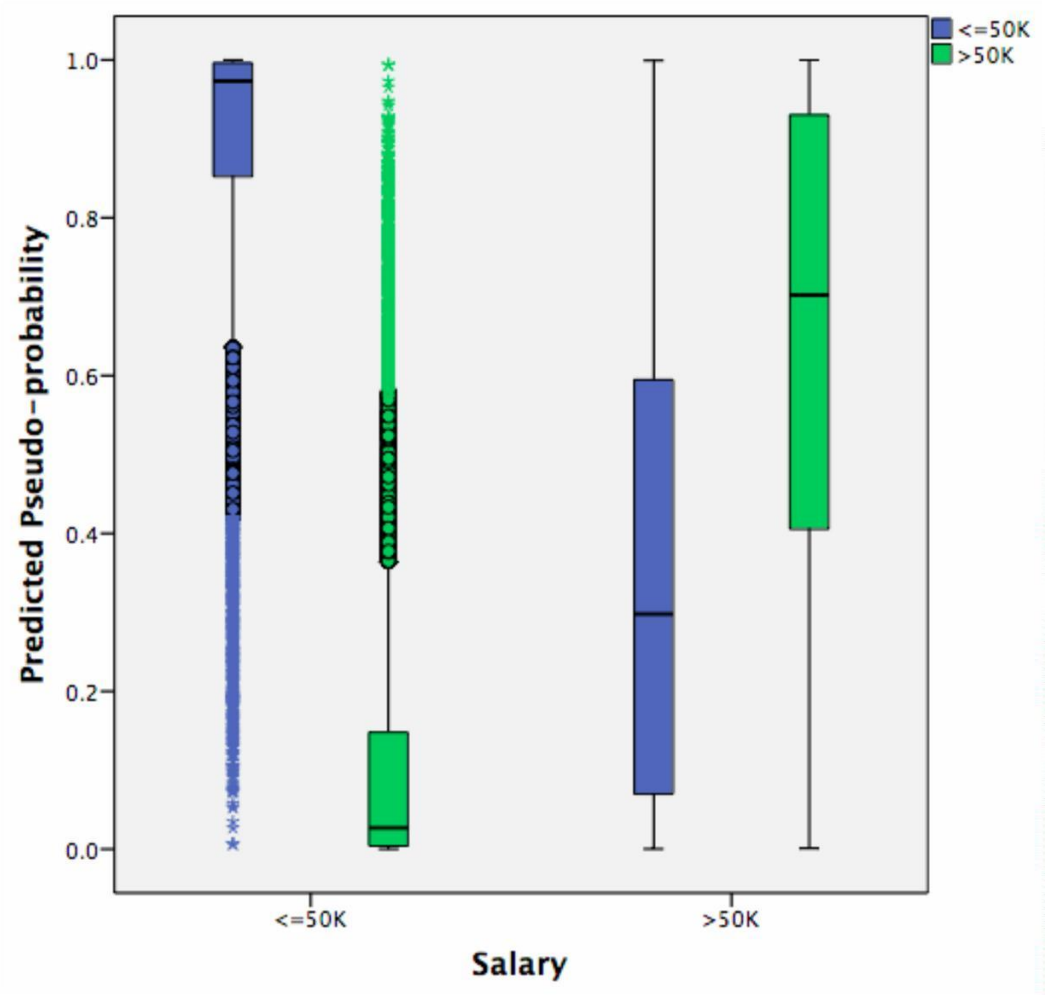

Figure 4. Predicted-by-Observed Chart (Output from SPSS)

Since we are predicting for categorical dependent variables, the predicted-by-observed chart in Figure 4 indicates a clustered boxplot of predicted pseudo-probabilities. The first box plot shows for cases that where the salary was <=50K had a pseudo-probability of <=50K.

This boxplot indicates the model was more accurate at predicting for <=50K for salary than >50K. Since there were only two categories for salary, the box plots are symmetrical for each salary type.

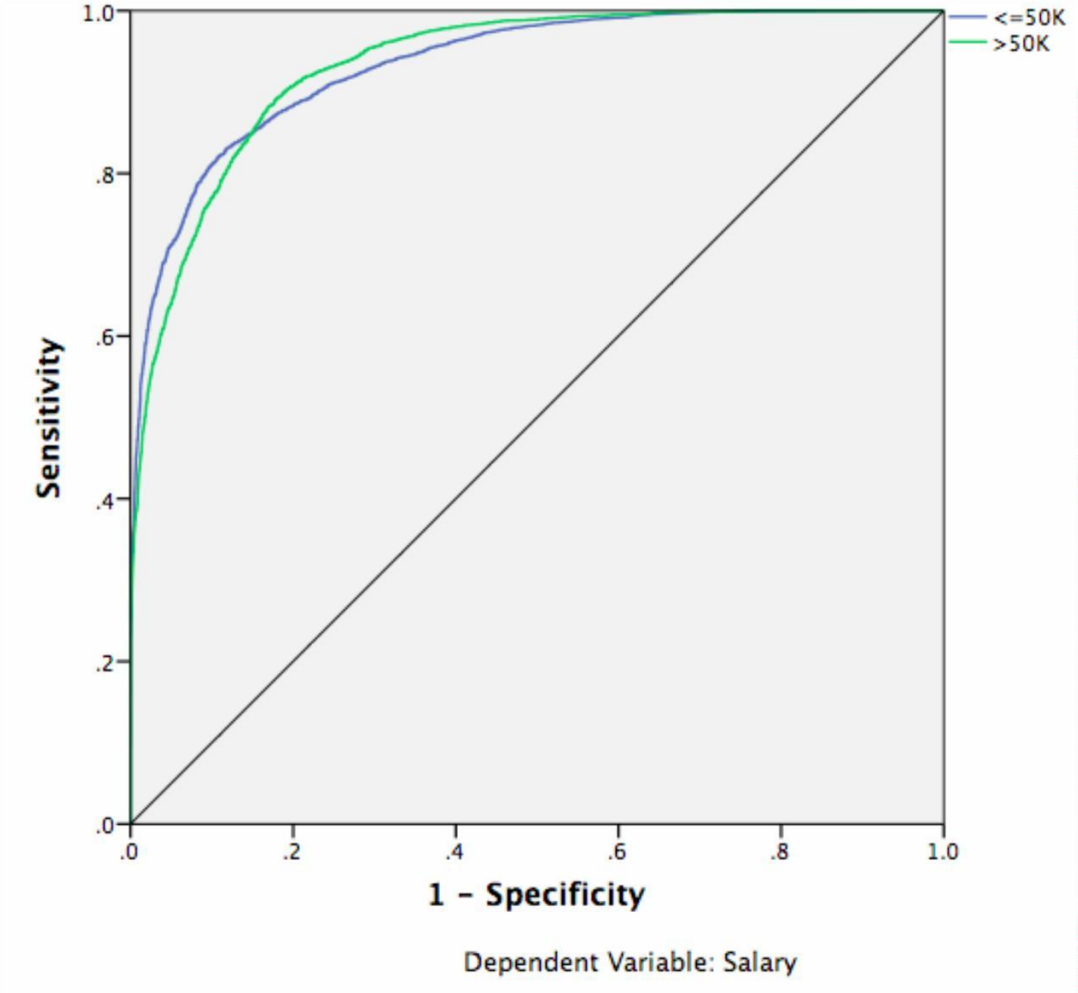

Figure 5, ROC Curve (Output from SPSS)

The ROC curve in Figure 5 shows a visual graph of the sensitivity and specificity for all possible cutoff values in a single plot. The two curves for '<=50K' and '>50K' are symmetrical and the area under the curves will give a summary of the chart. Sensitivity is where the model correct indicates true positives and the Specificity shows true negatives (GraphPad Software 2015).

		Area
Salary	<=50K	.934
	>50K	.934

Figure 6. Area Under the Curve for ROC (Output from SPSS)

Figure 6 shows the area under the curve each respective curve from the ROC Curve and can be used as a summary of the accuracy of the model. A test that is 100% correct would have the area under the curve as 1 so since this model has produced an area of 0.934 for both classifier values, it means it is a very good model. (Tape n.d.).

III. LOGISTIC AND RANDOM FOREST

The preliminary modelling on the census dataset has been carried out using general linear regression. To take into account the dichotomous nature of the dependent/response variable 'Salary' the link in the linear model is logistic. A simple linear regression model is given by-

$$\Pi_i = x_i'\beta$$

where β is the regression coefficients vector and the Π is given in terms of probabilities between 0 and 1. To cater to the restriction of using a range of numerical and categorical variables in the model as determinants to the response variable 'Salary', the linear model needs to be modified or transformed to a form where now the probabilities will only lie between 0 and 1 for any scale of values used as determinants in the model. The transformation carried out is a log transformation on the odds ratio of proportion Π. Thus, a regression model with logistic link is of the form-

$$logit(\Pi_i) = x_i'\beta$$

For dichotomous variables the logistic regression model's 'logit' transformation is given by-

$$log\left(\frac{P(Y=1,X=x)}{P(Y=0,X=x)}\right)=\beta_0+\beta^T X$$

where the two levels of the response variable are given by 0 and 1 with level 0 being the reference/base level. Thus, the maximum likelihood estimator of the logic model is given by the following-

$$logL(\beta)=\sum_i^n ylog(\pi) + (n\text{-}y)log(1\text{-}\pi)$$

To test for the statistical significance of the predictors, the logistic model carries out hypothesis testing on the coefficients of the predictors given by the β vector where the hypothesis testing is of the form-

$$H_0: \beta_i = 0$$

$$H_{alternative}: \beta_i \neq 0$$

where the standard error in the coefficients is $\sqrt{var(\beta)}$

and the corresponding test statistic is $\frac{\beta}{\sqrt{var(\beta)}}$

LOGISTIC IMPLEMENTATION

Logistic regression was carried out in R by following the steps in iteration till all predictors are significant at a α=1% significance level-

- Fitted model to training set using all variables and interaction terms
- Used stepwise regression to remove variables and interaction terms with non-significant influence on model
- Remove collinear variables
- Carry out transformations on variables
- Follow Step 1 again

The selection criteria for selecting between the logistic model is a combination of the following-

- AIC - Akaike Information criteria - Lower values of AIC used to select models representing an added amount of information to lift the model with each predictor added.
- AUC or Accuracy of the model given by the area under the ROC curve representing the deviance of the response variable explained by the model.
- RMSE - Root mean squared error to find the deviance between the predicted values and the model values. Low RMSE is the preferred metric.

- Sensitivity and specificity of the model - Aim is to simultaneously maximise both these measures without there being a high level of imbalance between these.

The following implementation of the GLM model shows some of the basic models that have been fitted but does not cover the entire range of models.

```
#GLM Model Fitting

#Basic model with all variables

logit.fit = glm(glmSalary~., family = binomial(link="logit"),data = training)



#Improving modelling

#Stepwise removal of insignificant variables

fit1sw = step(logit.fit)

#Fit model again without insignificant terms and minimum AIC score

logit.fit = glm(Salary~.-Native.country, family = binomial(link="logit"),data = training)

#Check for multi-collinearities in model

vif(logit.fit)

#Remove related variables from model and fit again

logit.fit = glm(Salary~.-Native.country-Relationship.status, family = binomial(link="logit"),data = training)
```

Logistic stepwise table

Model	Sensitivity	Specificity	AUC
Complete model(all variables,all classes)	0.8437	0.6077	0.7048
Y~Capital.loss + Capital.gain + Native.country+Work.hours.per.week + Age + Education.level + Employment.class + Marital.status + Sex+Race+Occupation	0.8353	0.5966	0.6965
Y~Capital.loss + Capital.gain + Native.country+Work.hours.per.week + Age + Education.level + Employment.class + Marital.status + Sex+Occupation	0.8382	0.6015	0.7003
Y~Capital.loss + Capital.gain + Native.country+Work.hours.per.week + Age + Education.level + Employment.class + Marital.status+Occupation	0.8232	0.6182	0.7103

Y~Capital.loss + Capital.gain +Work.hours.per.week + Age + Education.level + Employment.class + Marital.status+Occupation	0.8334	0.6251	0.7335
Y~Capital.loss + Capital.gain +scaledWork.hours.per.week + scaledAge + Education.level + Education.years+Employment.class + Marital.status + Sex+Occupation	0.9234	0.6698	0.8559

Results and drawbacks For GLM

For the models fitted using GLM none of them were able to provide an accuracy rating of more than 86%. But these models achieved a high sensitivity rate and a low specificity rate thus the next model family fitted in RandomForest to enable an ensemble between the two models for improving the specificity of the final model.

The best model fitted using GLM is given below-

```
confusionMatrix(logit.prednumeric,testing$Salary)
Confusion Matrix and Statistics


          Reference
Prediction  low high
      low  5240  626
      high  560 1270


               Accuracy : 0.8559
95% CI : (0.8476, 0.8639)
No Information Rate : 0.7536
P-Value [Acc > NIR] : <2e-16


                Kappa : 0.5801
Mcnemar's Test P-Value : 0.0591
            Sensitivity : 0.9234
            Specificity : 0.6698
         Pos Pred Value : 0.8933
         Neg Pred Value : 0.6940
             Prevalence : 0.7536
         Detection Rate : 0.6809
```

RANDOM FOREST IMPLEMENTATION

A Random forest has been fitted to the training dataset using the following steps in R:

- Tune model using cross validation to obtain optimised parameters- ntree and mtry
- Use parameters to form model
- Select significant determinants of the model using the varImpPlot plotting function.
- Follow step 1 till all variables are significant.

The following implementation tunes the model and the next section is used for model fitting.

```
#Tuning Parameters

bestmtry<-tuneRF(training[-14],training$Salary,
ntreeTry=100,stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE, dobest=FALSE)



#Result mtry=2 as best parameter
```

```
rf.fit <-randomForest(Salary~.-Relationship.status-Native.country,data=training, mtry=2, ntree=1000, keep.forest=TRUE, importance=TRUE)



#Improving Random Forest by selecting important determinants

varImpPlot(rf.fit)



#Refitting model without unimportant factor 'Race'

rf.fit <-randomForest(Salary~.-Relationship.status-Native.country-Race,data=training, mtry=2, ntree=1000, keep.forest=TRUE, importance=TRUE)



#Prediction

rf.preds = predict(rf.fit,type="prob",newdata=testing)[,2]

rf.pred = predict(rf.fit,newdata=testing)

confusionMatrix(rf.pred,testing$Salary)
```

Confusion Matrix and Statistics

Prediction	Reference: low	Reference: high
low	5435	849
high	365	1047

Accuracy : 0.8823

95% CI : (0.8739, 0.8903)

No Information Rate : 0.7036

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7753

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.9371

Specificity : 0.7522

Pos Pred Value : 0.9449

Neg Pred Value : 0.7815

Prevalence : 0.8536

Detection Rate : 0.8506

Detection Prevalence : 0.8365

Balanced Accuracy : 0.8446

'Positive' Class : low

Results of random forest

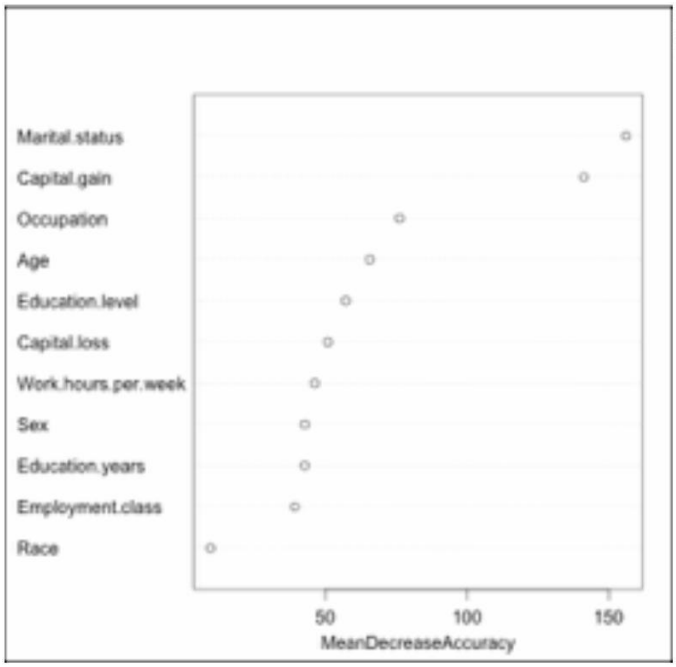

Figure 7. Ensemble of Logit and RandomForest

The ensemble method combines the logistic and Random forest to give the best possible model. The model chosen is a weighted average of predictions of the logistic and the random forest at a 0.8 and 0.2 ratio. Thus, the model chosen has the following equation-

```
predictionsfinal<-0.8*logit.pred+ 0.2*rf.preds
```

```
          Reference
Prediction     0     1
         0 20493  1365
         1  5071  6671
               Accuracy : 0.9085
                 95% CI : (0.9042, 0.9126)
    No Information Rate : 0.8608
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.7455
 Mcnemar's Test P-Value : < 2.2e-16
```

III. DECISION TREE

Decision tree is a fairly simple and easy to understand technique however the prediction and its accuracy rate is very close to any other model, depending upon the the complexity of the dataset. A decision tree is like a flowchart with internal nodes representing a “test” on the input or test data. The output results in branches which tell the outcome of the test on each node, while the decision is calculated considering all the attributes. The classification rules are represented by the paths from root to leaf.

A decision tree normally consists of three types of nodes:

1. Decision Nodes
2. Chance Nodes
3. End Nodes

Decision tree is a graphical representation of possible solutions to decision based on certain conditions and is called a decision tree because it starts with a single box(root) and then branches off into a number of solutions(tree). Decision trees are graphical and fairly simple to understand but also provides a systematic documented thought process for making decisions depending on different conditions. Decision tree helps formalize the brainstorming process so we can identify more potential solutions. A simple decision tree example is shown below:

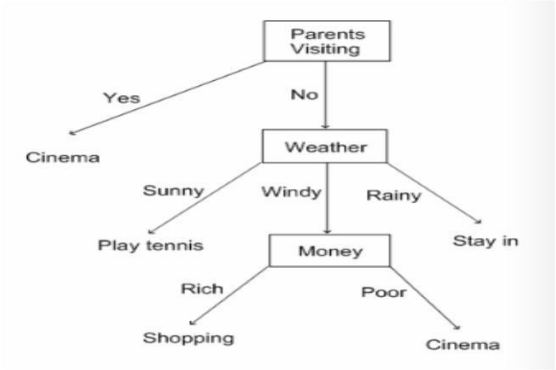

Figure 8. Simple Decision Tree Example

Decision trees are also known as Classification and Regression trees (CART or C&RT) and has several advantages:

1. **Implicitly perform variable screening:** After fitting a decision tree to a training dataset, the top few nodes which split the decision tree are essentially the most important variables within the dataset and feature selection is essentially automatic.

2. **Require little input from user for data preparation:** The decision tree structure remains the same with or without the variable transformation through binarization or normalization although normalization greatly reduces the data processing overhead. Also splitting the data for building trees is not affected by missing values. Decision tree are also not sensitive to outliers as the splitting of the sample depends on proportion of the samples within the sample split range and not on absolute values.

3. **No effect of nonlinear relationships between parameters:** Decision trees do not require any assumption of linearity in the data so we can also use decision trees where the parameters are known to be nonlinearly related.

4. **Easy for interpretation to executives:** Decision trees being intuitive and easy to explain are a great model to help make the user, executive understand the output and results.

However there is one key disadvantage of decision tree model. The tree if not properly pruned or the tree growth limited, would result in a data that has been overfit and makes them somewhat poor predictors.

Model Implementation

For the implementation of the Decision Tree Model, KNIME software with the decision tree learner, and decision tree predictor Nodes was used. The decision tree learner was first trained by the training dataset provided. The training data was filtered for missing values. Following figure represents some of the configuration settings of the missing value node:

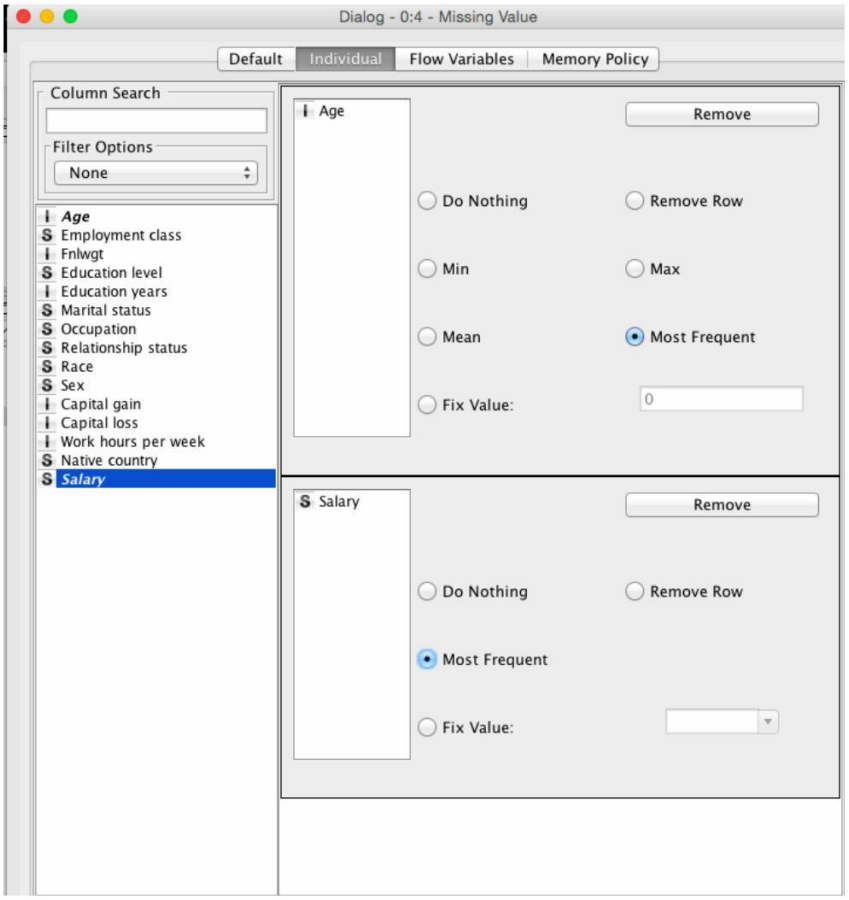

Figure 9. Missing values configurations (Knime Output)

After handling the missing values in the training data, the resultant data was partitioned by the Knime Partition Node to test the accuracy of the model. The partition was set to 90% relative i.e. 90% data for decision tree learning and 10% for decision tree prediction, using random seed for salary column target. Random seed was configured for partitioning so that the model can be trained on random data rather than periodic data and hence help increase the accuracy of the predictions. Settings for partition shown by the figures below:

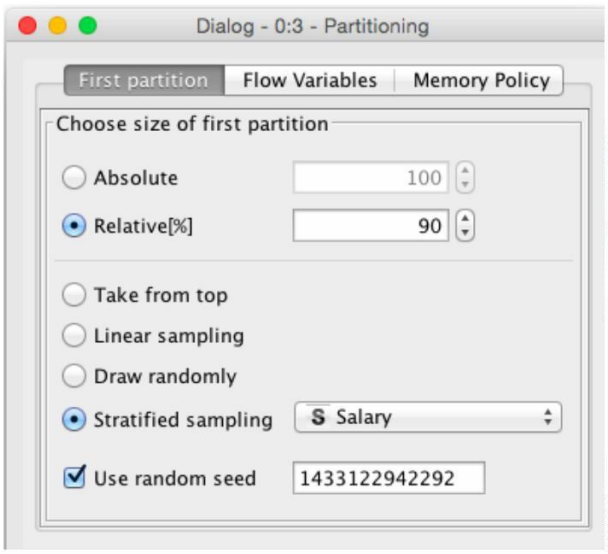

Figure 10. Partition configuration for training data (Knime Output)

After partitioning the data was divided into 2 datasets. First for learning which contained 90% of the rows from the training as shown by the figure below:

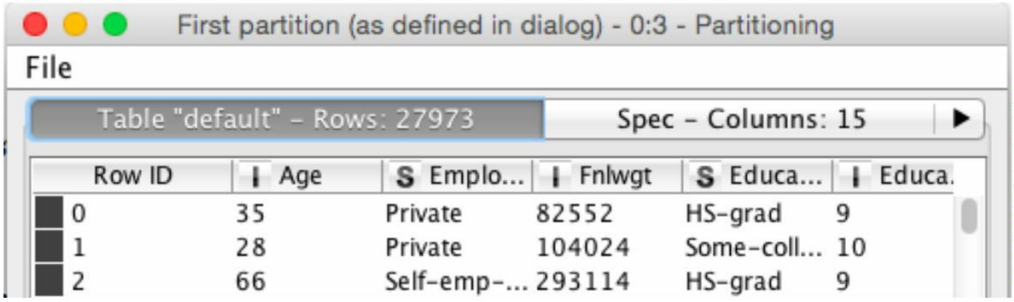

Figure 11. First partition for training data (90%) (Knime Output)

The second partition contained 10% of the training data as shown by the figure below:

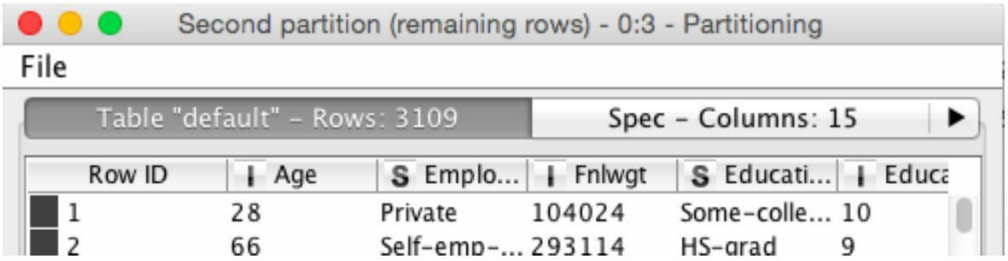

Figure 12. Second partition for testing the accuracy of the training data (10%) (Knime Output)

Following figure represents the decision tree learner output for 90% training data. The training data has 33601 number of rows which was divided into 27973(90%) for first partition and 3109(10%) number of rows for the second partition.

The First partition of the data was used to train the Decision tree learner. The decision tree learner was configured as shown by the figure below:

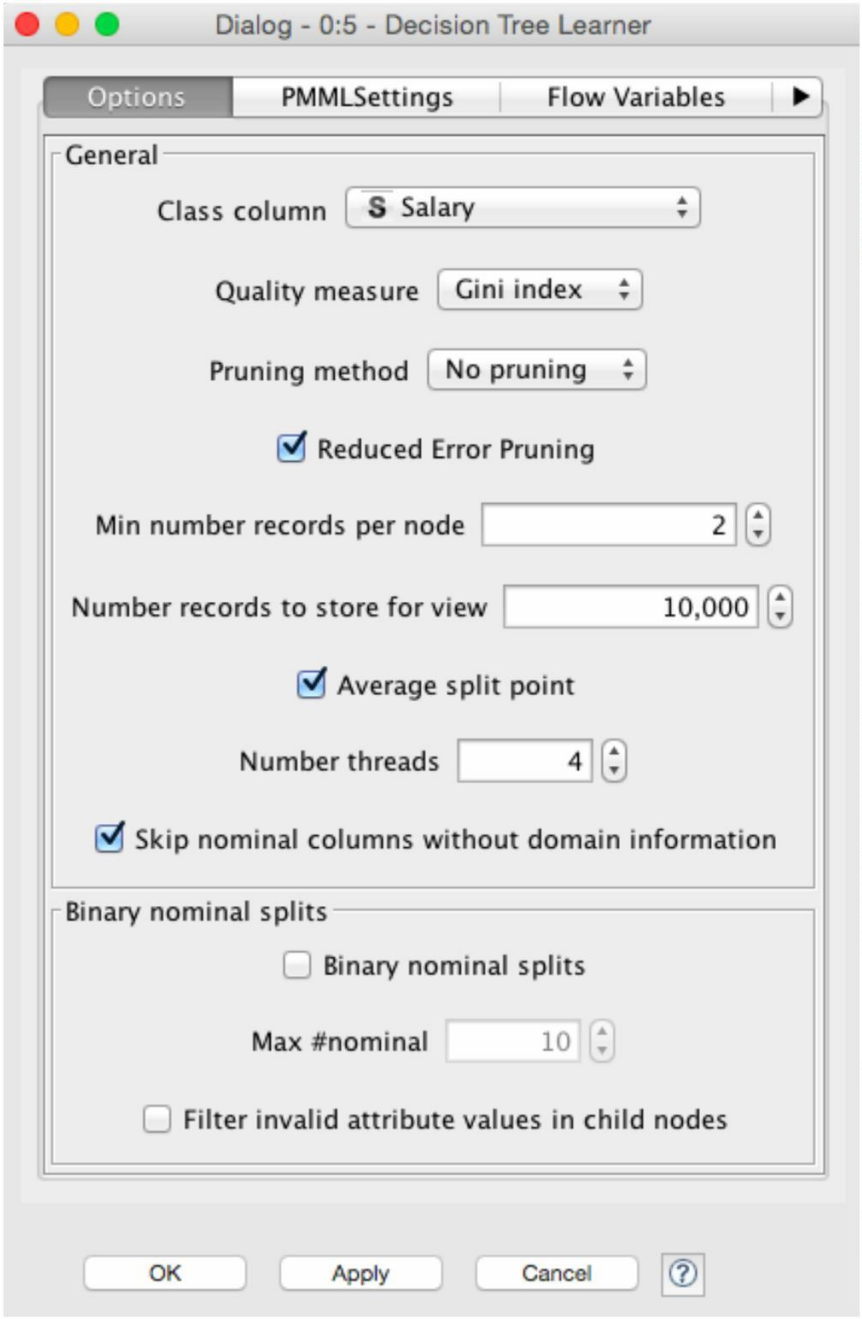

Figure 13. Decision Tree Learner Configurations (Knime Output)

Most of the settings for the decision tree learner was set to default. However the number of records per node field has been very important for enhancing the accuracy. Most of the settings were tested for the optimal results however, best results were achieved for default settings as random tree itself is a powerful tool for handling the various conditions and dataset exceptions (i.e. outliers or missing values).

The decision learner resultant output is shown below:

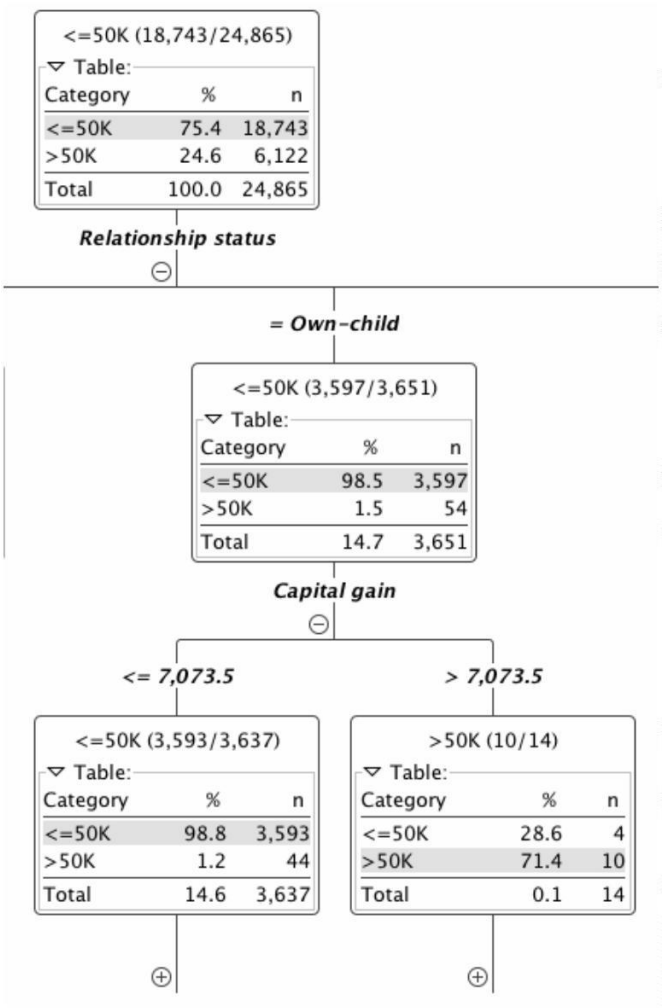

Figure 14. Output for decision tree learner (Knime Output)

The figure shows only a part of a branch of the complete resultant decision tree where ‘n’ shows the number of rows or data that lies in the two salary categories i.e. greater than 50K and less than or equal to 50K. Each leaf summarises the findings of the

decisions taken at a particular branch. In the above case, the relationship status of the user has been considered where only the users who own children have a been divided into two capital gain branches with values greater than and less than or equal to 7073.5.

Similarly another tree branch can be shown in the below figure which is very descriptive of the automatic partitions of the decision tree based on relationship status and latter capital gain:

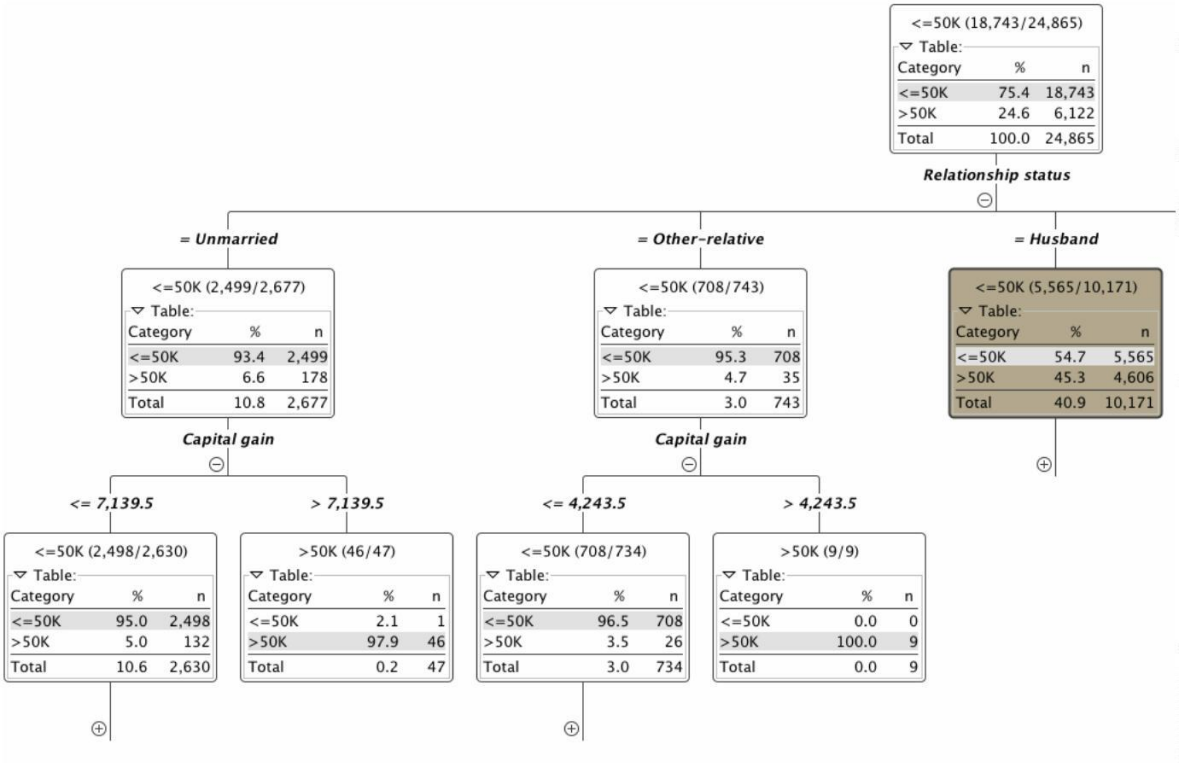

Figure 15. The Decision Tree Training output (Knime Output)

After the training of the decision tree learner, the decision tree predictor node was used which had two inputs. One input for the training data and one for the test data. The remaining 10% of the data from the splitter was used to determine the accuracy of the model and was connected to the decision tree predictor test input. Decision tree predictor was configured as shown by the diagram below:

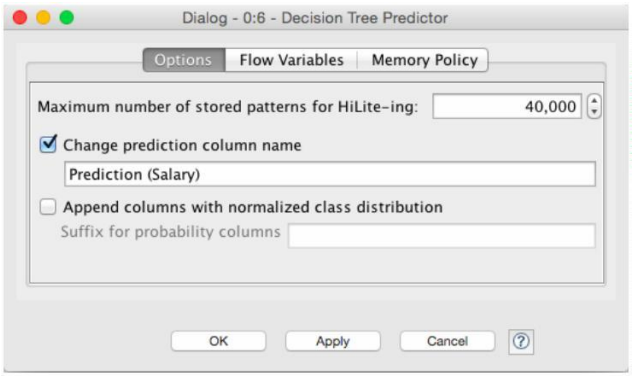

Figure 16. Decision Tree Prediction Node configuration (Knime Output)

The resultant output of the decision tree predictor were as follows:

File

Table "default" - Rows: 14400 | Spec - Columns: 16 | Properties | Flow Variables

Row ID	Capital ...	D Work h...	S Native ...	S Salary	S Predicti...
3		0.48	United-States	?	<=50K
4		0.5	United-States	?	>50K
13		0.398	United-States	?	<=50K
14	46	0.449	United-States	?	>50K
15		0.806	United-States	?	<=50K
17	79	0.398	United-States	?	<=50K
19		0.398	United-States	?	<=50K
30	81	0.398	United-States	?	>50K
36		0.398	United-States	?	<=50K
38		0.398	United-States	?	>50K
40		0.398	United-States	?	<=50K
49		0.398	United-States	?	<=50K
52		0.194	United-States	?	<=50K
55		0.398	United-States	?	<=50K
60		0.398	United-States	?	<=50K
61		0.316	United-States	?	<=50K
64		0.398	United-States	?	<=50K

Figure 17. Extra Salary Prediction column added to the test dataset (Knime Output)

A scorer was used for the model accuracy prediction. The scorer was connected to the decision tree prediction output and the comparison of accuracy between the actual salary column and the predicted salary column were made. The resultant accuracy was 93.3%. Initially the accuracy was lower but after certain adjustments to the decision tree number of records per node and handling the missing values the accuracy was improved. The scorer results are shown in the figure below:

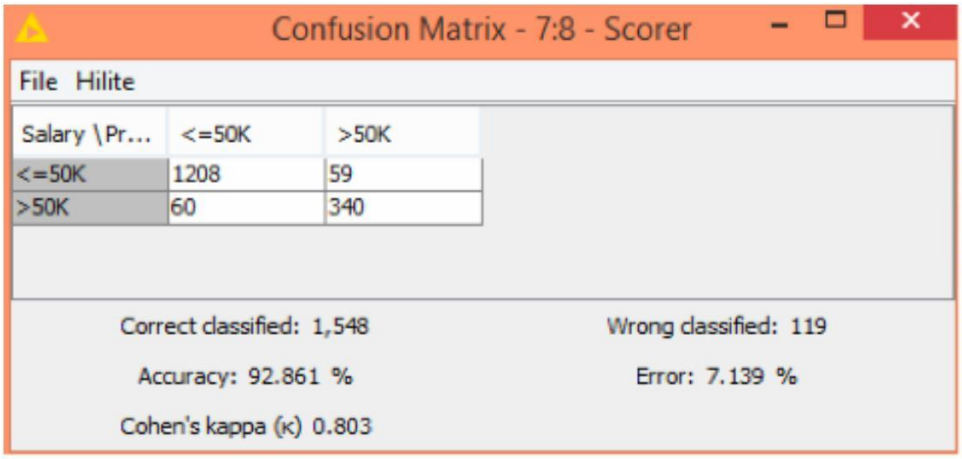

Figure 18: Decision Tree Training result which was close to Kaggle 93.3% (Knime Output)

Another CSV reader was used to read the test file and was also handled for missing values. Some of the columns were normalized to decrease the overhead of the predictions on the nodes. The accuracy of the output of the predictions were assumed to be similar to the model accuracy output. The resultant dataset predictions to the test data were made and a CSV writer was used to write the output.

The final flow of the decision tree model is shown by the below figure:

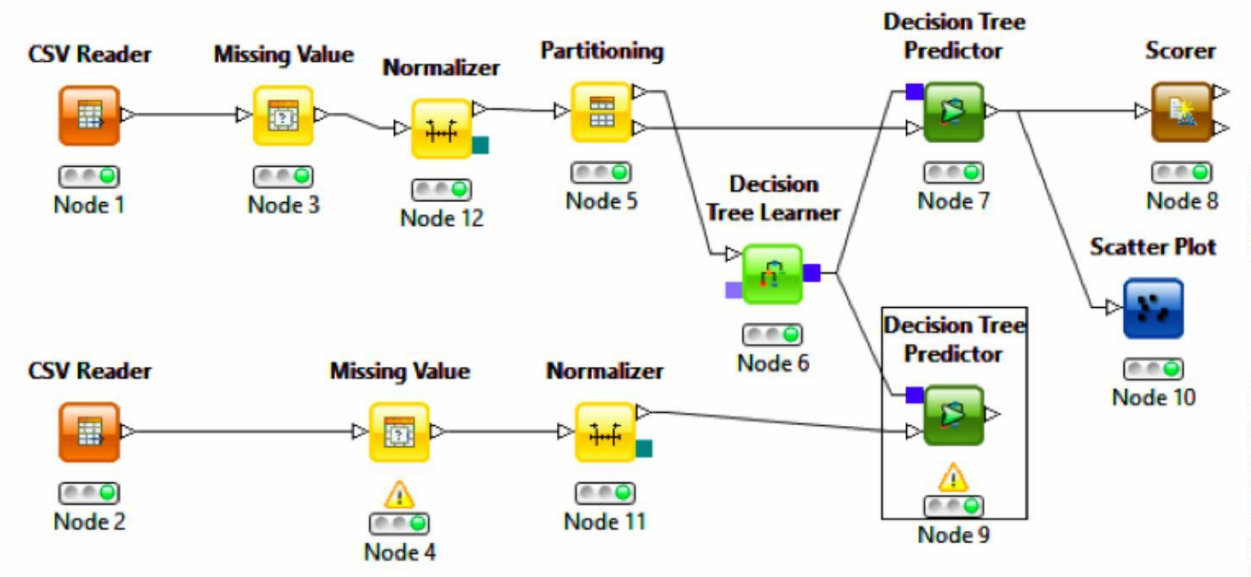

Figure 19. Decision Tree Model (Using Knime)

III. Random Forest

Random forest a well known method used for machine learning purposes. The goal is to reduce the variance by taking the average of multiple decision trees. It works as a massive anthology of the co-related decision trees and use them to make a classification. That is why it is a technique based on the bagging that uses the combination of learning models to increase the classification accuracy(Breiman 2001).

How does it Work?

$$S = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & C_1 \\ \vdots & & \vdots & \\ f_{AN} & f_{BN} & f_{CN} & C_N \end{bmatrix}$$

Figure 20.Random Forest- Matrix of S (Breiman 2001)

In figure above, suppose a matrix S of training sample will be used to submit to algorithm to create a classification model. In this case fA1, fB1, fC1 are the features such that fA1 is the feature of the 1st sample that continues with all the samples up to Nth sample. Similarly, the last column C1 that runs up to CN means it has lots of features and a training class. The aim is to create a random forest to classify the sample set.

Different subsets will be created from this sample set. For Example in the first subset the line number 12, the line number 15 and also some random elements were used. So with subset from the matrix S different decision trees will be created as shown in figure below.

$$S_1 = \begin{bmatrix} f_{A12} & f_{B12} & f_{C12} & C_{12} \\ f_{A15} & f_{B15} & f_{C15} & C_{15} \\ \vdots & & \vdots & \\ f_{A35} & f_{B35} & f_{C35} & C_{35} \end{bmatrix} \quad S_2 = \begin{bmatrix} f_{A2} & f_{B2} & f_{C2} & C_2 \\ f_{A6} & f_{B6} & f_{C6} & C_6 \\ \vdots & & \vdots & \\ f_{A20} & f_{B20} & f_{C20} & C_{20} \end{bmatrix}$$

$$S_M = \begin{bmatrix} f_{A4} & f_{B4} & f_{C4} & C_4 \\ f_{A9} & f_{B9} & f_{C9} & C_9 \\ \vdots & & \vdots & \\ f_{A12} & f_{B12} & f_{C12} & C_{12} \end{bmatrix}$$

Figure 21. Random Forest- Subsets of Matrix S (Breiman 2001)

Similarly for M number of samples there will be M number of decision trees. That is why it is called Random Forest because it has lots of decision trees in it. With all these decision trees, we have different variation of the main classification which will further be used to create a ranking of the classifier.

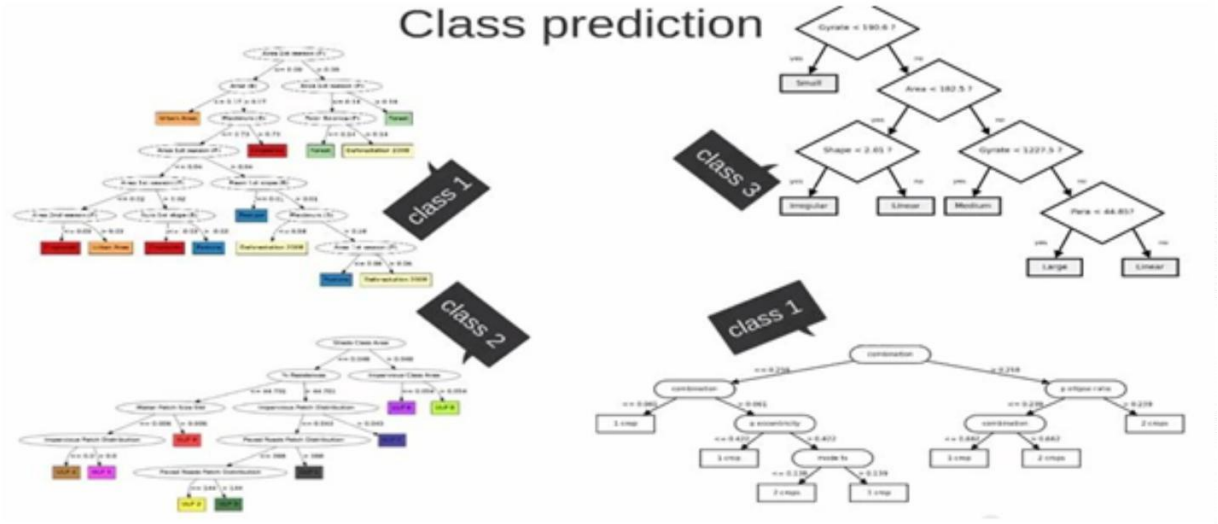

Figure 22. Random Forest- Decision Tree Ranking & Classification (Breiman 2001)

Suppose, our forest has four decision trees and a new element to classify will ask them one by one about their prediction and rank them into different classes accordingly. So we have four independent decision trees that were created using subsamples of the entire sample set. And they can be account for the number of votes for each class. As we can see in figure 3 we have two votes for class 1 and single for the rest of decision trees. So the class 1, a random forest will classify all the elements and class 1 will be the selected class of this classification.

KNIME IMPLEMENTATION AND RESULTS

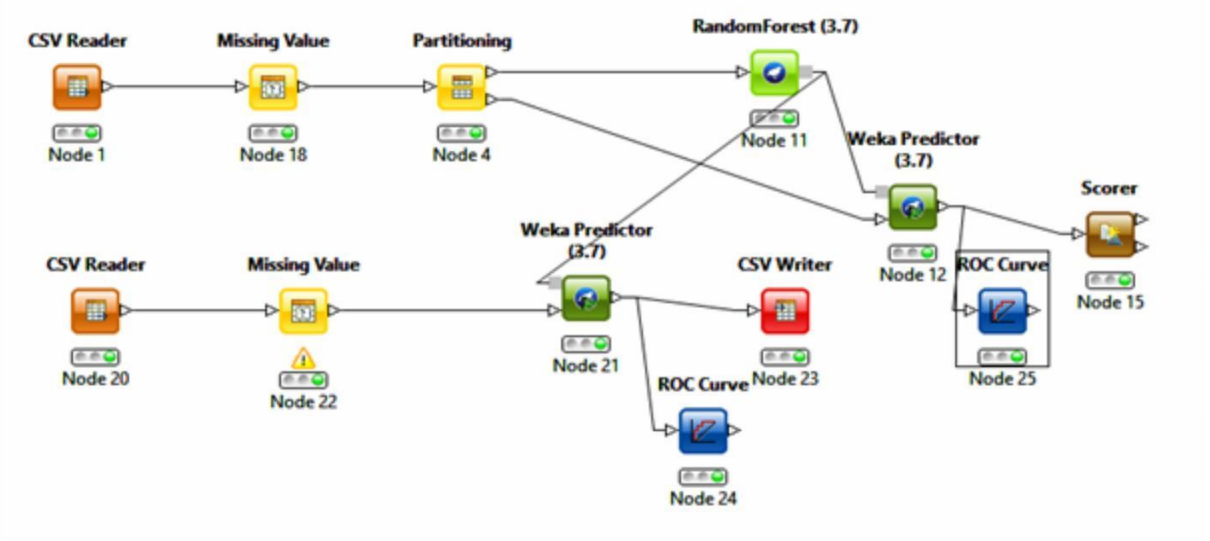

Figure 23. Random Forest- Implemented Model

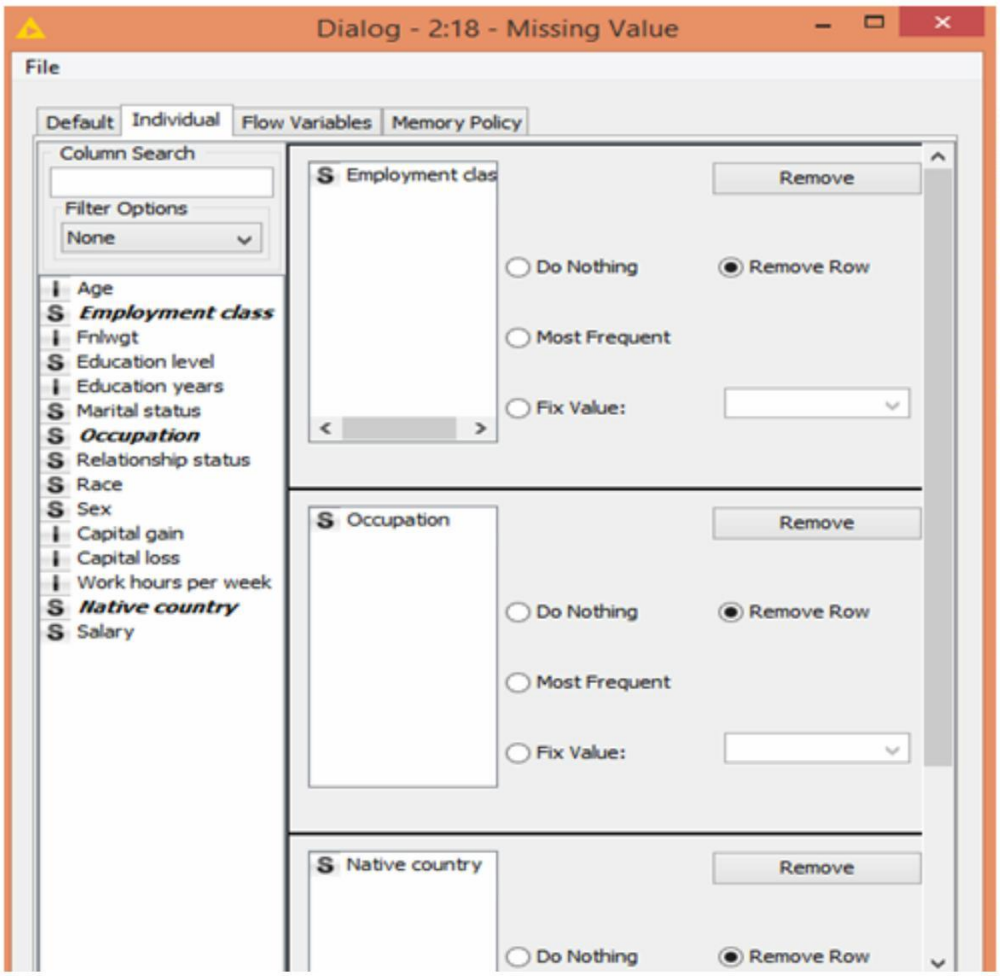

Figure 24. Random Forest- Missing value node

Than we partitioned our data using partitioned node by taking 70 % relative partition size which is sampled through Salary attribute.

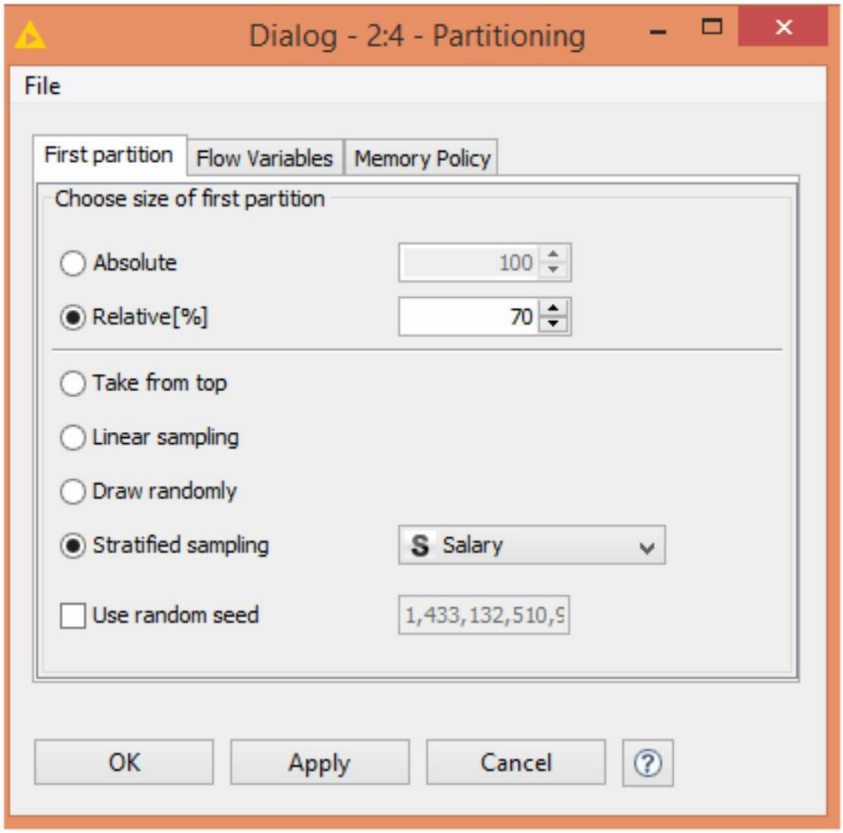

Figure 25. Random Forest- Partitioning Node

We used random forest node for training purposes and set decision trees to forty five and target attribute set to Salary column.

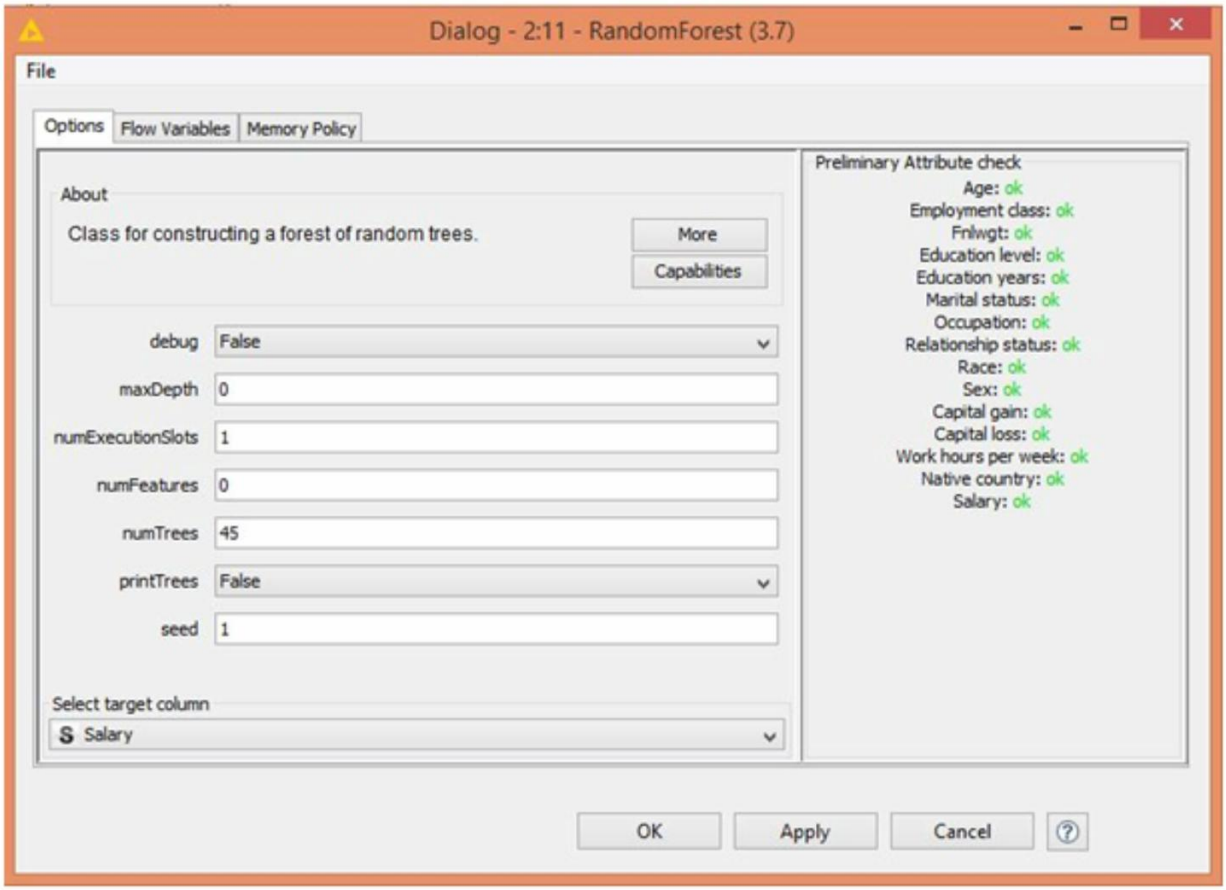

Figure 26. Random Forest learner configuration

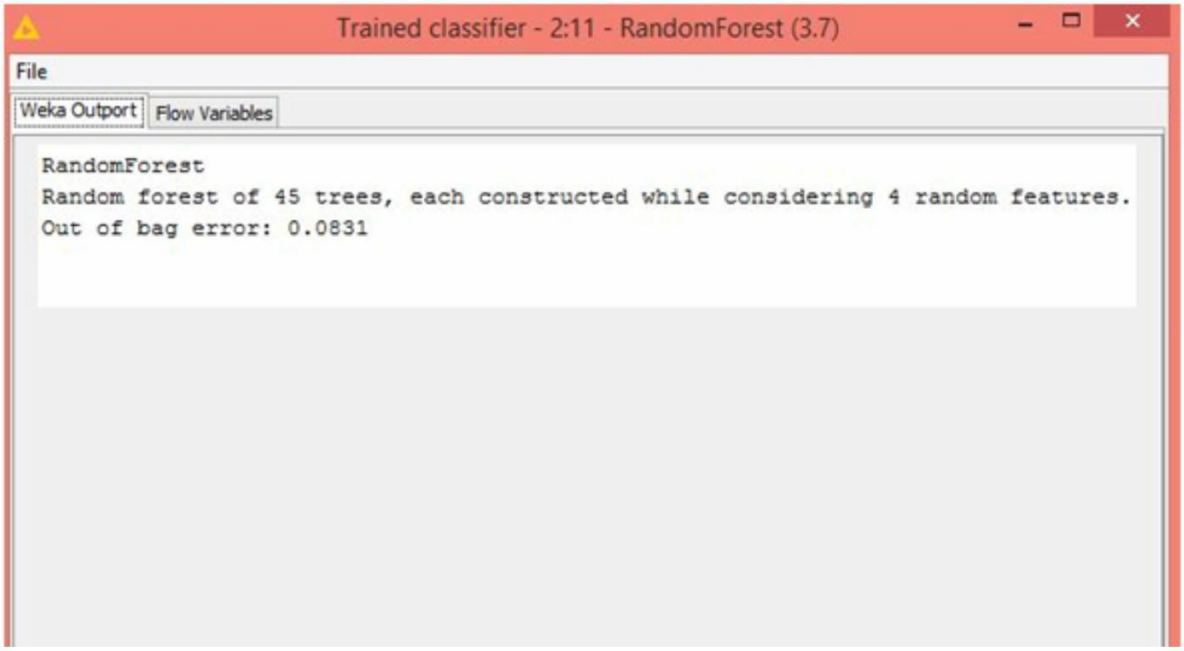

Figure 27. Random Forest- Predictor output

Further weka predictor was used to predict the possible outcome and for accuracy of our model we used scorer node that gave us 93.008% accuracy of our model on training data set

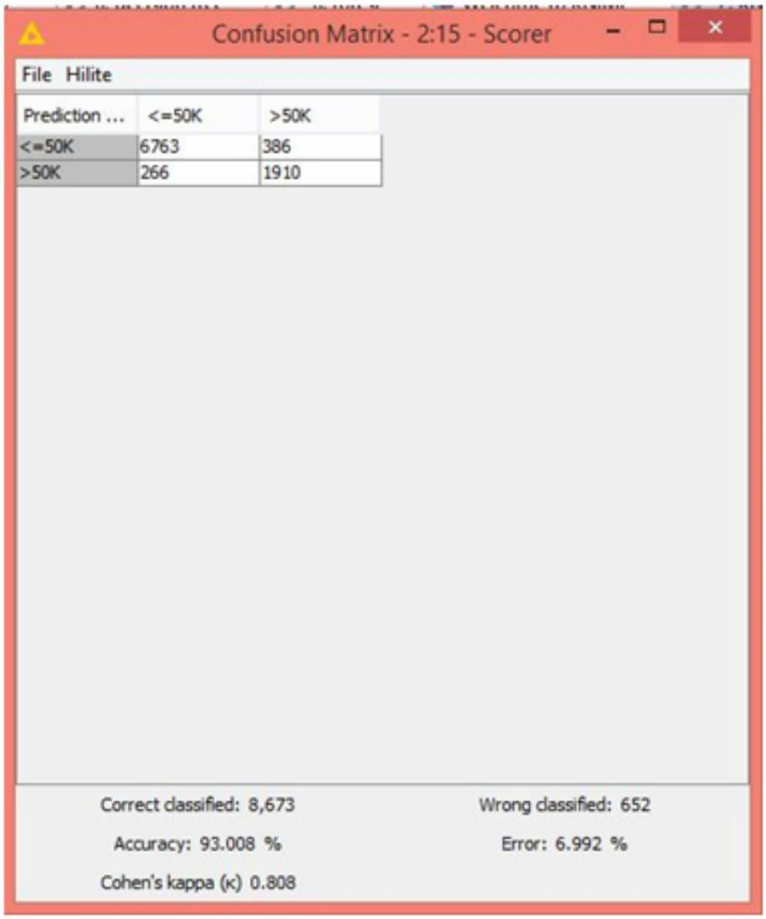

Figure 28. Random Forest- Scorer Node outcome

We used the trained model and test data in weka predictor node to check the accuracy of our model in Kaggle. The resulted ROC curve for salary less than and equal to 50k and greater than 50k are are show below

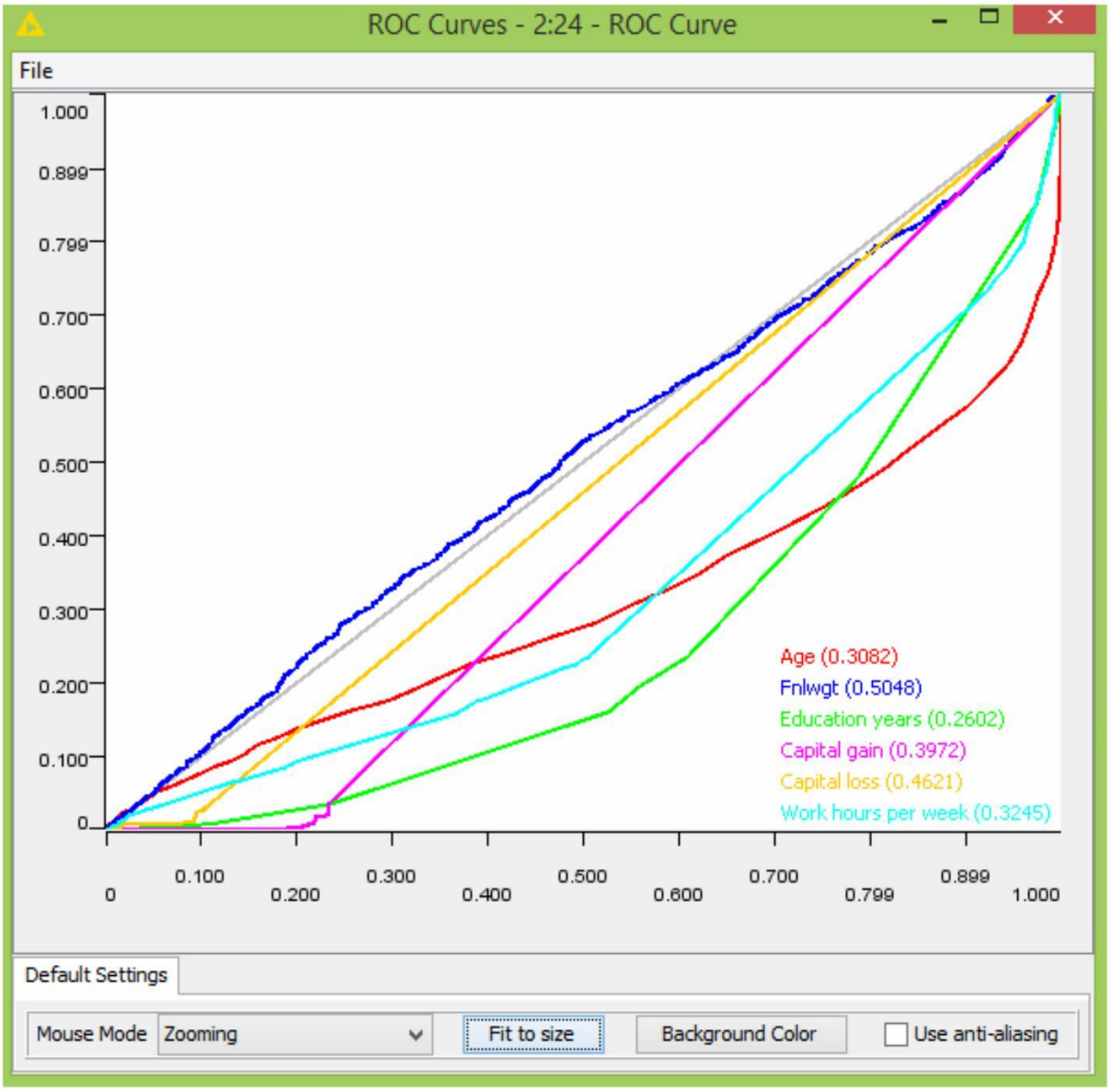

Figure 29. Random Forest- ROC curve for <=50k

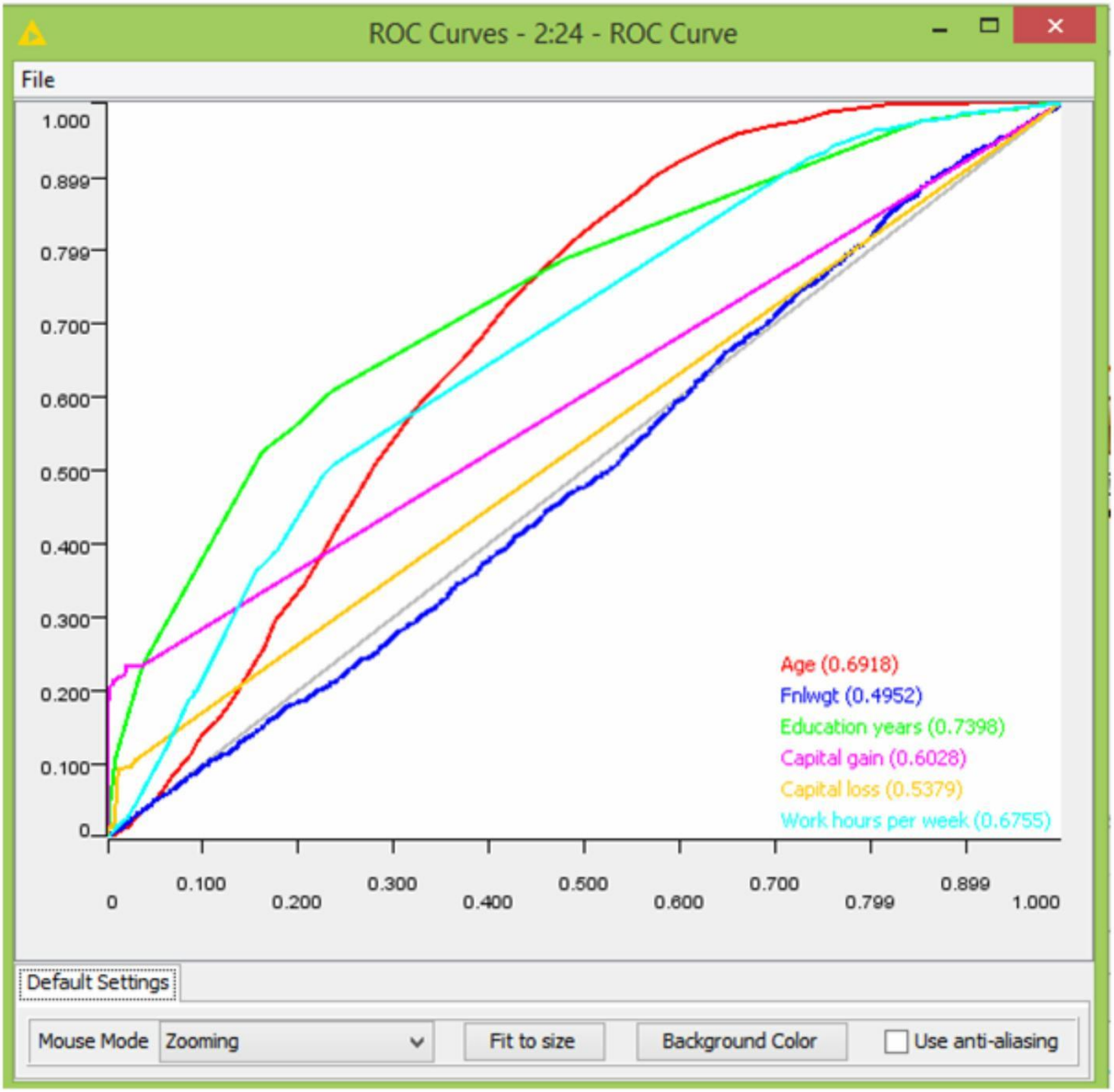

Figure 30. Random Forest-ROC Curve >50k

CHOSEN CLASSIFIER MODEL

The type, performance, accuracy and errors on the training set and on the cross validation set, reasons for selection in comparison to the other techniques tried.

We chose Decision Tree Prediction Model for predicting the salaries for the test dataset. The best accuracy of the model obtained was 93.3% which was calculated on the test dataset. The same 93.3% percent accuracy was obtained from cross validation test set on kaggle. However while optimizing the model, the accuracy obtained also peaked at 94.5% but the same results were not obtained through the cross validation set.

The reason we chose decision tree model as our selected classifier is based on the fact that optimal accuracy results were obtained through Decision Tree Prediction model. It was simple to implement and fairly easy to understand.

Decision Trees perform automatic variable feature selection. The first split in the decision tree shows the most important variables and it all works automatically.

Decision Trees require very little data pre processing as outliers do not affect the tree structure and the tree structure remains the same with or without the transformation. Splitting of the data depends on proportion of the samples within the split range and not on absolute values.

Decision trees are not affected by non-linear values and do not require any assumptions for linearity and hence are used when it is known that the parameters are nonlinear.

SUMMARY

Our team of four people chose the following models which were a neural network (multilayer perceptron), logistic regression, decision tree, and random forest. The given training set was used to train each model which was then applied to the test set. Each model gave an accuracy based on a partition the training set which allowed to select the best one.

Based on the accuracies returned, we propose the use of a decision tree model for the analysis of this type of data. On our attempts, we managed to attain an accuracy of around 93% for the decision tree model which was superior to the others.

It is important to note some limitations to the decision tree model such as the fact that is will require the target variable to have discrete values only. It also performs much better with if a few highly relevant attributes are present but not as well if there are many complex patterns or relationships.

Other factors may affect a person's yearly salary so if possible and available, more attributes could be used in the analysis. For example, information regarding how many years of experience a person has or what type of role they work (ie. management or entry level). Since de-duping reduced the accuracy for every model used, it may require further investigation as to why this is the case. Obtaining new data for input to further train the model would be helpful too as well as combining it with information regarding their previous years salary where available.

REFERENCES

Breiman, L. 2001, 'Random Forests', *Machine Learning*, vol. 45, no. 1, pp 5-32.

GraphPad Software 2015, *Interpreting Results: ROC Curves*, viewed 19 May 2015, <<http://www.graphpad.com/guides/prism/6/statistics/index.htm?sensitivity_and_specificity.htm>.

Rokach, L. & Maimon, O. 2010, *Data Mining and Knowledge Discovery Handbook*, 2nd edn, Springer, New York.

Stergiou, C. & Siganos, D. n.d., *Neural Networks,* Imperial College of London, viewed 20 May 2015, <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>.

Tape, T. G. n.d., *The Area Under an ROC Curve*, University of Nebraska Medical Center, viewed 19 May 2015, <http://gim.unmc.edu/dxtests/roc3.htm>.

Tu, J. V. 1996, 'Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes', *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225-1231.