CSE 5243 - HOMEWORK 4

CLASSIFICATION OF ADULT DATA SET

VEDASRI UPPALA

uppala.1@osu.edu

TABLE OF CONTENTS

1. DATA SET DESCRIPTION

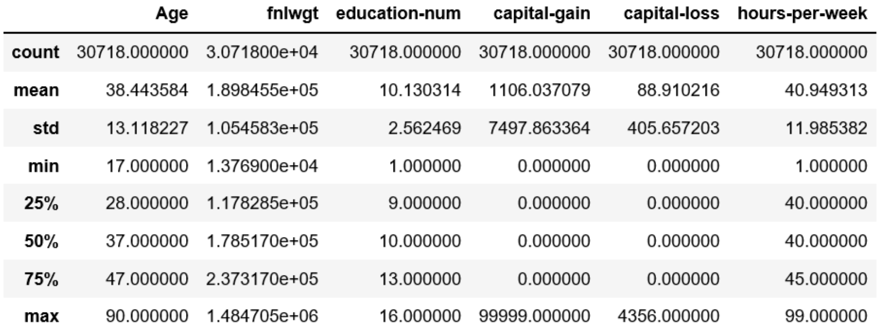
The adult data set in the UCI repository has nearly 49000 instances. The class label of the data set which is either *<=50K* or *>50K* describes the annual income of people under consideration. 76% of the samples in the data set have an annual income which is <=50K. The data set has 15 attributes including the binary class label “Income”. Apart from this there are 14 attributes out of which 7 are categorical, 1 is binary and the remaining 6 are numeric attributes. In the adult data set there are a total of nearly 48000 instances. UCI repository has divided it into training and testing data each having 32561 and 16281 instances respectively.

The following table gives information about the 15 attributes in the data set. The column *‘Attribute Type’* in the table indicates whether the attribute is categorical, binary or numeric. In the next column, we have number of categories for categorical attribute and the number of unique values for numeric attributes. For each attributes I counted the number of instances with missing values and the results can be seen in the last column of the table.

Attribute Name	Attribute Type	Number if categories/unique values	Number of tuples with missing values
Age	Numeric	72	0
Workclass	Categorical	7	1836 (5.64%)
Fnlwgt	Numeric	20495	0
Education	Categorical	16	0
Education Num	Categorical (represented as number)	16	0
Marital-status	Categorical	7	0
Occupation	Categorical	14	1843 (5.66%)
Relationship	Categorical	6	0
Race	Categorical	5	0
Sex	Binary	2	0
Capital-gain	Numeric	118	0
Capital-loss	Numeric	90	0
Hours-per-week	Numeric	94	0
Native-country	Categorical	42	583 (1.79%)
Income	Binary	2	0

2. DATA EXPLORATION AND ANALYSIS

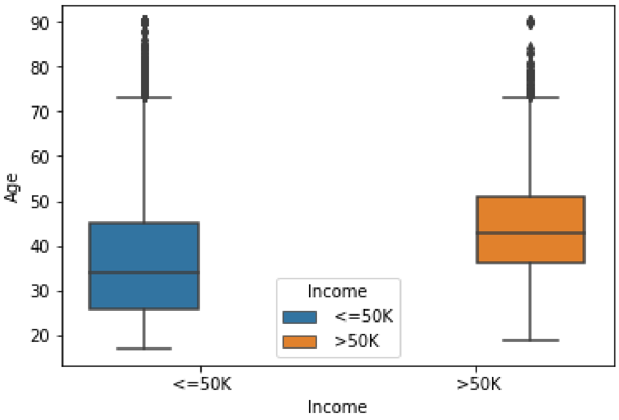
The data set has six numeric attributes and the summary statistics for the same have been shown below.

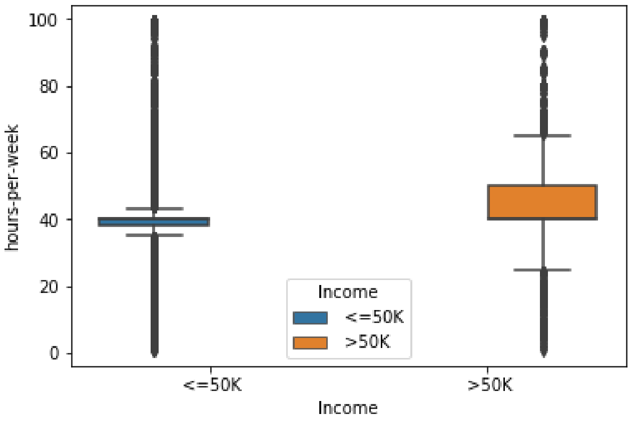


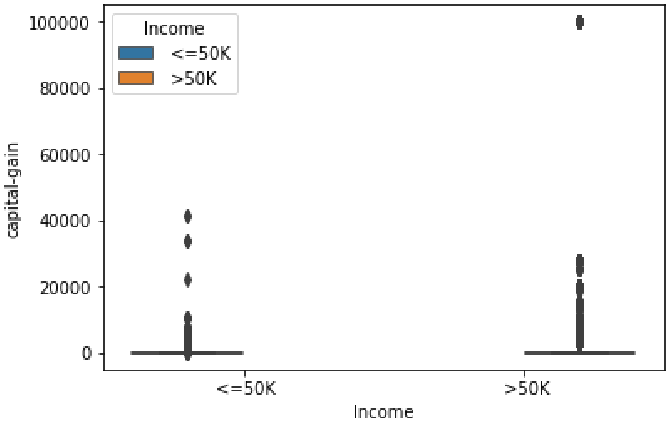
	Age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	30718.000000	3.071800e+04	30718.000000	30718.000000	30718.000000	30718.000000
mean	38.443584	1.898455e+05	10.130314	1106.037079	88.910216	40.949313
std	13.118227	1.054583e+05	2.562469	7497.863364	405.657203	11.985382
min	17.000000	1.376900e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178285e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.785170e+05	10.000000	0.000000	0.000000	40.000000
75%	47.000000	2.373170e+05	13.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

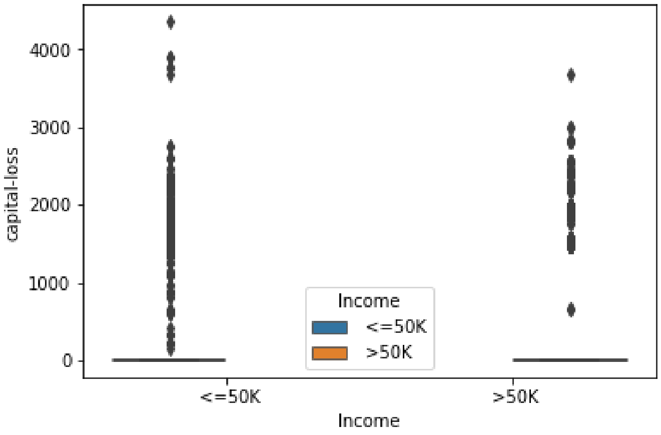
It can be seen that the attribute *fnlwgt* has very large variance. Also, the attribute *fnlwgt* has nearly 20500 unique values among nearly 31000 tuples. This suggests that this attribute is a candidate for removal and it does not have any predictive significance since it is not capable of separating the tuples based on class label.

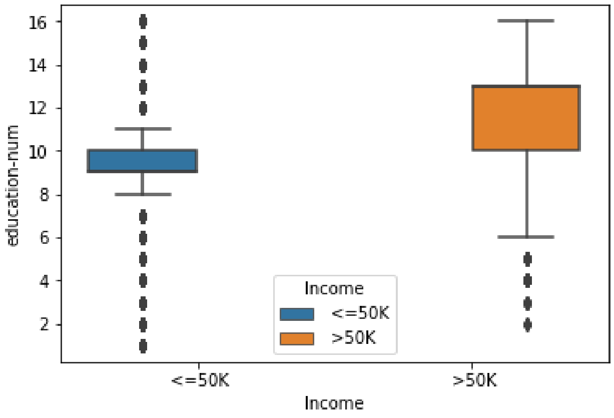
Next, the box plots have been shown for each of the six numeric attributes. Each box plot has been plotted with Income in the x axis and the corresponding attribute on the y axis to visualize which attributes are mostly significant for applying classification algorithms.

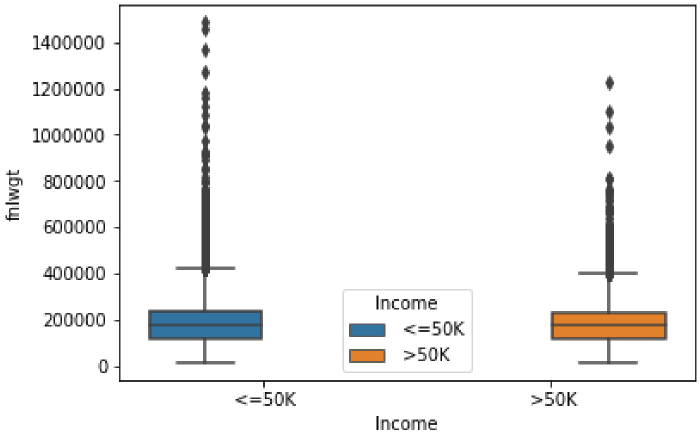








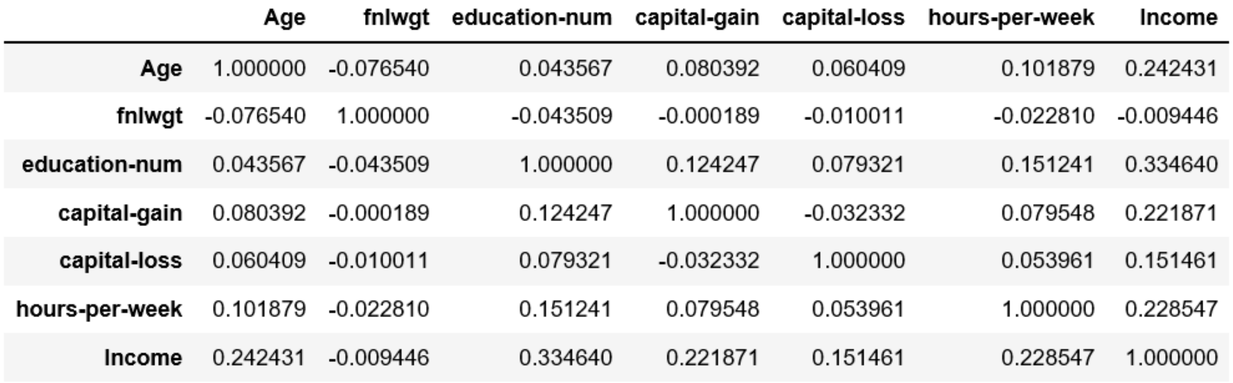




From the box plots the following conclusions have been made.

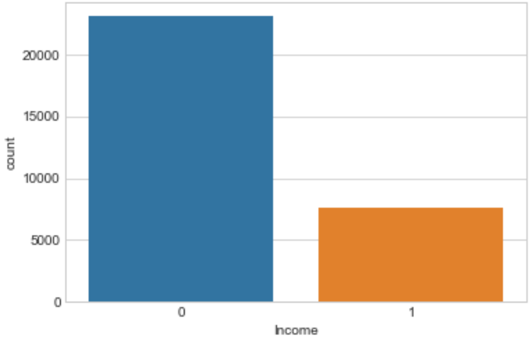
- The range of values for attributes *Age, Education-Num and hours-per-week* is slightly higher for instances with Income > 50K. This suggests that these attributes have predictive significance.
- For the attributes *Capital-gain* and *Capital-loss* majority of the tuples have a value of 0.
- The *fnlwgt* attribute has similar range for both the classes. So this attribute does not possess predictive capability.

Next the correlation matrix has been calculated for the numeric attributes using Pearson correlation coefficient to see if any of the attributes correlate to each other.



	Age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	Income
Age	1.000000	-0.076540	0.043567	0.080392	0.060409	0.101879	0.242431
fnlwgt	-0.076540	1.000000	-0.043509	-0.000189	-0.010011	-0.022810	-0.009446
education-num	0.043567	-0.043509	1.000000	0.124247	0.079321	0.151241	0.334640
capital-gain	0.080392	-0.000189	0.124247	1.000000	-0.032332	0.079548	0.221871
capital-loss	0.060409	-0.010011	0.079321	-0.032332	1.000000	0.053961	0.151461
hours-per-week	0.101879	-0.022810	0.151241	0.079548	0.053961	1.000000	0.228547
Income	0.242431	-0.009446	0.334640	0.221871	0.151461	0.228547	1.000000

From the above matrix we can see that none of the numeric attributes strongly correlate to each other. Also, all of the numeric attributes except *fnlwgt* correlate to some extent with the class label *Income* suggesting that they have predictive capability.



The given data set is class-imbalanced. Almost 76% of the tuples in the training data set have income <=50K. The results are shown in bar graph on the left.

3. DATA PRE-PROCESSING:

3.1 HANDLING MISSING VALUES:

It has been observed that the data set has missing values in certain instances. I have observed that the values of the attributes *workclass*, *occupation* and *native-country* are missing.

- "*workclass*" is a categorical attribute and from the data set it has been observed that the *"workclass"* attribute has the value "*Private*" in 70% of the instances in the data set. So the missing values for the "*workclass*" attribute have been replaced as "*Private*".

- “*Native-country*" which is a categorical attribute has 42 categories but in 90% of the samples, the value of the attribute is “*United States*”. So, I have replaced the missing values with *“United States”*
- The categorical attribute *‘occupation’* has 15 categories and I have observed that instances have been almost equally distributed among these 15 categories. So, it is not possible to replace the missing values using mode. Since we have reasonably enough amount of data, I have decided to drop the instances which have a value missing for the *‘occupation’* attribute.

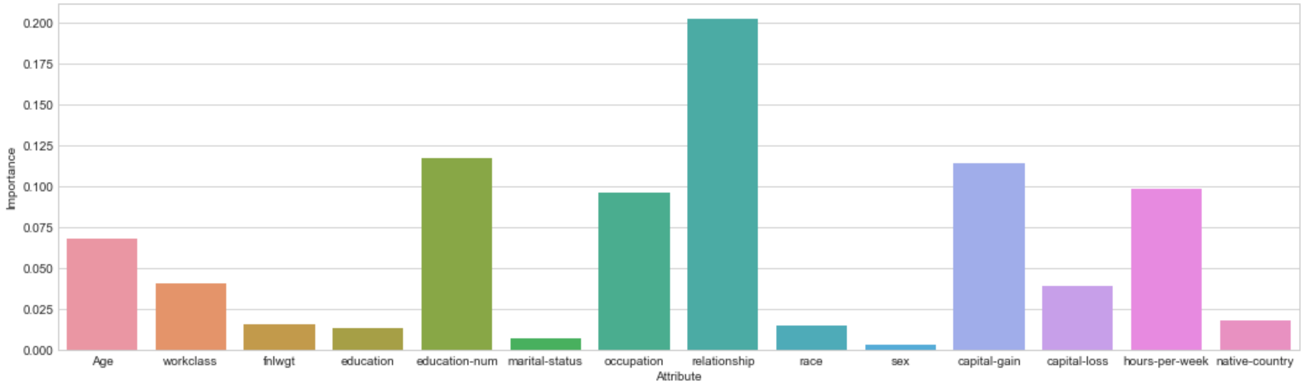
After dropping instances with missing values for ‘*occupation*’ the size of the data set is reduced to 30718.

3.2 FEATURE SUBSET SELECTION

- The attribute ‘*fnlwgt*’ has been removed since it has very high variance and has large number of unique values. Also, it has been observed from the correlation matrix that the Pearson coefficient for fnlwgt and class label Income is very less. Since it does not have any predictive capability this attribute has been removed from the data set.
- The attribute “*education”* and “*education-num*” are highly correlated. The “*education-num*” is just a numerical encoding for the “*education”* attribute. So, the information is redundant. Therefore, I have decided to remove the “*education*” attribute since it does not create any information loss.
- The categorical attribute “*native-country”* which has 42 categories has “*United States*” as the value for 90% of the samples. So, using this attribute we will not be able to classify the tuples into one of the classes. Therefore, I have removed this attribute since it is not providing any useful information necessary for classification.
- From the data set it is observed that the information provided by attribute ‘*marital-status’* could be inferred from the ‘*relationship*’ attribute. If the ‘*relationship’* has value ‘*unmarried*’ then we clearly know the value for ‘*marital-status*’. So ‘*marital-status’* attribute has been eliminated from the data set.

To see the validity of the above decisions to remove the attributes, I have applied Random Forest Classifier to the data set and I have calculated the feature importances based on mean decrease in impurity at the node and this is averaged over all trees in the ensemble. My decision to remove the four attributes was supported by the results obtained by calculating feature importances for the attributes. Moreover, the accuracy on the test set also improved.

A bar graph has been plotted between Attributes and their importances. The four attributes which were removed have relatively low feature importance values. Moreover, it can be seen that attribute ‘*sex*’ and ‘*race*’ also have low importance. But removing these two attributes did not improve the accuracy. So I included them in the data set.



3.3 DISCRETIZATION AND BINNING

Binning using equal-width bins has been applied to the following attributes.

- The attribute '*Age'* has 72 unique values and often we are interested in a range of age while deriving some useful information. So, I have binned the age attribute. ***(Number of bins = 12)***
- The attributes '*Capital-gain'* and '*Capital-loss'* have value 0 for majority of the tuples. In spite of this, they have many unique values. So, Binning has been applied.
  (Number of bins = 1500 for capital-gain and 500 for capital-loss)
- The attribute '*hours-per-week'* has 94 unique values and is a potential candidate for binning since we are more interested in ranges rather than the actual number. ***(Number of bins = 2)***

The performance of Random Forest classifier has been tested before and after applying binning and it has been observed that the accuracy improved slightly (0.21%) after binning. So, I have decided to apply binning. To determine the optimal bin sizes, I have tested the Random Forest classifier with several values for bin sizes for all 4 attributes. At certain point, the accuracy did not improve further by increasing the number of bins and such values have been selected.

From the box plots we can see that the attributes '*Age'* and '*Hours-per-week'* have outliers. But it has been observed that nearly 3% of the instances have such extreme values. Outliers can be eliminated by performing Binning. These outliers will fall into their appropriate ranges when the attributes are binned. So, using Binning, I have eliminated outliers in my data set.

The off-the-shelf implementations of some of the classifiers in *scikit learn* library for Python do not work with categorical attributes in the data set. So, I have encoded all the categorical and binned attributes so that they contain numbers. Similar operations have been applied to test data as well.

4. MODEL DEVELOPMENT

I applied the following five models to the data set. I used the off-the-shelf implementation of these models that is available in the *'scikit learn'* machine learning library for Python. For each model, several parameters have been varied and the ones which yielded maximum accuracy on the test data have been selected.

4.1 DECISION TREE CLASSIFIER:

The DecisionTreeClassifer package from scikit learn has been used to fit a decision tree model to the data set. The function to create the model has some parameters to control the growth of the tree. If default values are used it leads to fully grown and unpruned trees. So the overfitting may lead to high generalization error. To prevent this I varied values for the parameters '*max_depth'* and '*min_samples_leaf'* which denote the maximum depth of the decision tree and the minimum number of samples that are required to be present at each leaf node respectively. With my experiements, I observed that when '*max_depth'* is set to 4 and *'min_samples_leaf'* is set to 500 I have achieved the highest accuracy.

4.2 ARTIFICIAL NEURAL NETWORK CLASSIFIER:

A build a neural network for the data set I have used the MLPClassifier package from scikit learn. The model I used optimizes the log-loss function using stochastic gradient descent. I have chosen '*logistic-sigmoid'* as my activation function because it has given the highest accuracy. Also, there is a parameter '*hidden_layer_sizes'* to specify the sizes for each of the hidden layers. After some experimentation, I got maximum accuracy when 2 hidden layers of sizes 6 and 7 are used. To prevent the neural network from overfitting the data, I have specified the regularization term for the model. I have chosen the value of 0.00005 as optimal after checking the generalization error.

4.3 SUPPORT VECTOR MACHINE

The SVC method from the SupportVectorMachine package from scikit learn has been used to build a Support Vector Machine for the model. The parameter "*probability*" is set to '*true*' to get the probability estimates for the class labels. The parameter degree specifies the degree of the polynomial kernel function that is used in the fit. The optimal value for this data set is found to be 4. The optimization of the support vector function is controlled using the '*tol*' parameter to specify the tolerance. This value has been set to 0.001 and training terminates when gradient of the optimized function is less than or equal to this value.

4.4 ENSEMBLE LEARNER – RANDOM FOREST:

A Random Forest which is an ensemble of several decision trees is built on the data set. By using a number of decision trees, over fitting is controlled and accuracy is improved. I have used 15 Decision Trees to build my Random Forest model. The sub-samples of the given data set is fed into each decision tree. The size of the input fed into each decision tree is kept constant by setting the parameter *BootStrap = True*. So, the size of each sub sample is the same and sampling is done with replacement. I have specified the maximum depth of each tree as 6. By using these parameters, I was able to improve the generalization error.

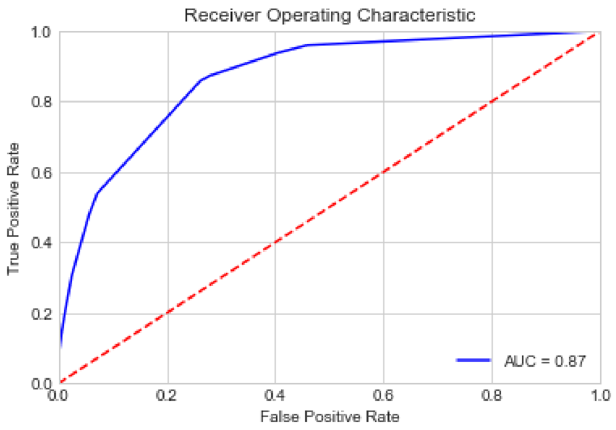
4.5 K NEAREST NEIGHBORS:

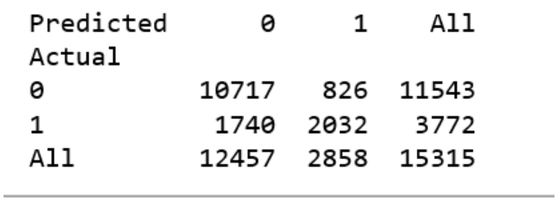
A KNN model is built for the data set using NearestNeighbors package in scikit learn library. Euclidean distance has been used as the distance measure. I varied the k value and found that when *k = 21* the accuracy on the test set was found to be maximal.

5. MODEL EVALUATION:

The models built in the previous step have been tested using the test data in the UCI repository. For each model, the ROC Curve, Confusion Matrix, Accuracy, Precision, Recall and F-measure have been calculated and plotted. In the confusion matrix, 0 corresponds to class <=50K and 1 corresponds to class >50K. Since the dataset is imbalanced, accuracy may not always give a true measure of classifier performance. Hence Fscore has been calculated for each model.

1. DECISION TREE



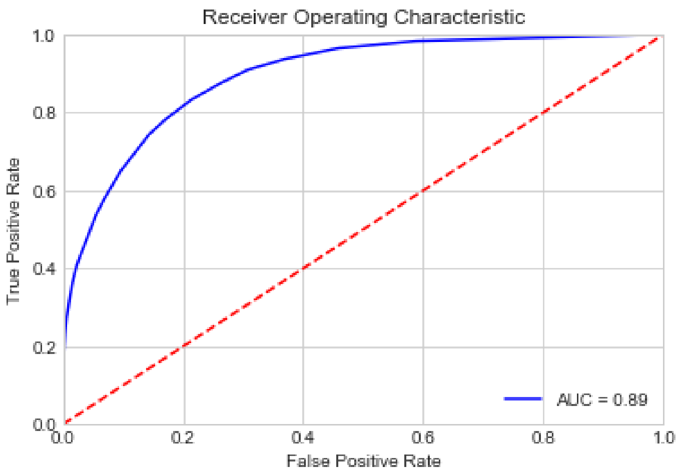


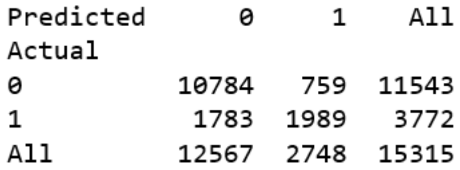
```
Predicted      0     1    All
Actual
0          10717   826  11543
1           1740  2032   3772
All        12457  2858  15315
```

Accuracy : 83.245 *Precision: 82%*

Fscore : 82% *Recall : 83%*

2. ARTIFICIAL NEURAL NETWORK



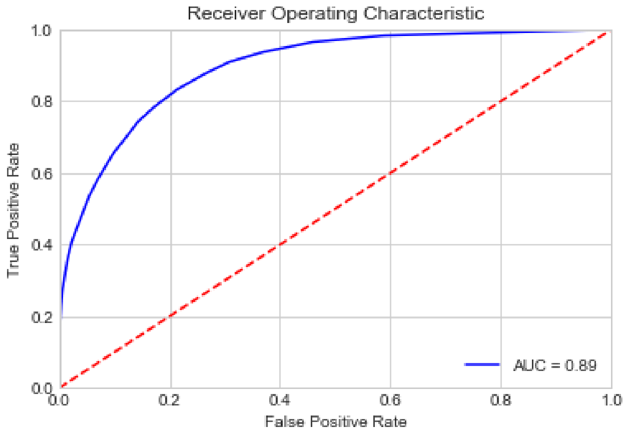


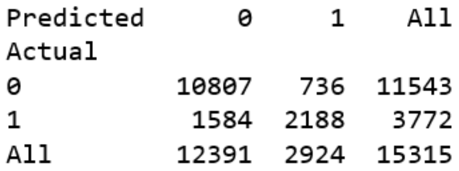
```
Predicted      0     1    All
Actual
0          10784   759  11543
1           1783  1989   3772
All        12567  2748  15315
```

Accuracy : 83.401% *Precision: 83%*

Fscore : 82% *Recall : 83%*

3. SUPPORT VECTOR MACHINE



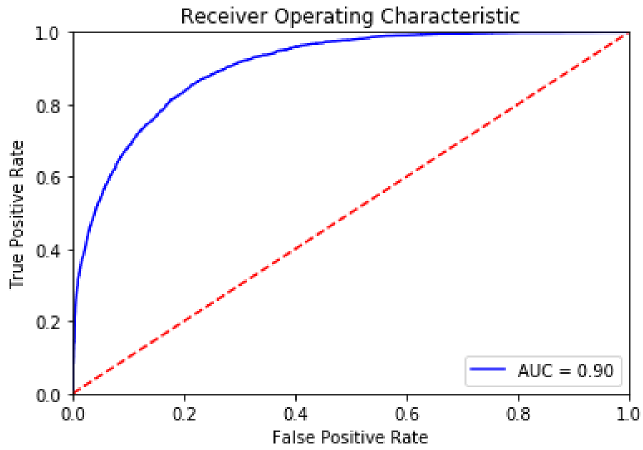


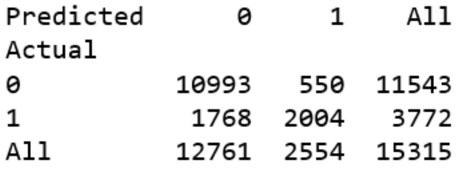
```
Predicted      0     1    All
Actual
0          10807   736  11543
1           1584  2188   3772
All        12391  2924  15315
```

Accuracy : 84.851% *Precision: 84%*

Fscore : 84% *Recall : 85%*

4. RANDOM FOREST



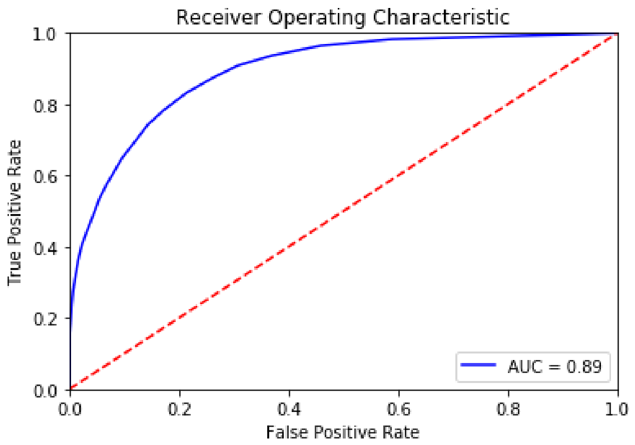


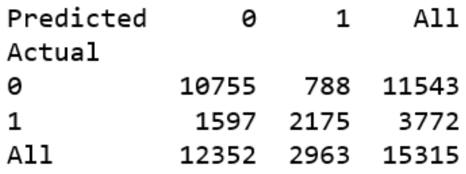
```
Predicted      0     1    All
Actual
0          10993   550  11543
1           1768  2004   3772
All        12761  2554  15315
```

Accuracy : 84.964% *Precision: 84%*

Fscore : 84% *Recall : 85%*

5. K NEAREST NEIGHBORS





```
Predicted      0     1    All
Actual
0          10755   788  11543
1           1597  2175   3772
All        12352  2963  15315
```

Accuracy : 84.427%

Precision: 84%

Fscore : 84%

Recall : 84%

From the above figures and statistics we can see that the models Support Vector Machine, K nearest Neighbors and Random Forest are comparable because they have nearly same values for Accuracy, F-measure, Precision and Recall. Similarly, the models Artificial Neural Network and Decision Tree have similar values for all 4 measures. However, Artificial Neural Network has a slightly better Area Under ROC (AUC) compared to decision tree. But the performance of Decision Tree and ANN on the test data is slightly poor compared to the other three. Among SVM, KNN and Random Forest , it can be seen that Random Forest has a slightly higher area under ROC (AUC)compared to the other two.

Over all, Random Forest is the best model to the data set. This conclusion has been derived by comparing Accuracy, Fmeasure and Area under ROC Curve. However, Random Forest is time consuming since it builds several decision trees internally. Also, several parameters like number of trees, maximum depth of each tree and minimum samples at each node have to be configured beforehand. Therefore good performance of the Random forest Classifier is achieved at the expense of speed.