Tasks:

- ### SGEMM GPU Kernel Performance Prediction

  The emergence of programmable graphics hardware has led to increasing interest in offloading numerically intensive computations to GPUs. This has made GPUs very prevalent in this era of high performance computing, but GPU programming remains challenging and it would therefore be handy to have a model which captures the main performance factors of GPU kernels that helps to estimate the run time, which could also potentially be used to identify its bottlenecks.

- ### Income Prediction

  Today's millennials are looking for simple and flexible payment solution that suits their financial situation without forcing them into one-sided credit agreements with unfair interest rates. Fintech companies' answer to that is the Pay-Later feature.
  However, there is a major risk involved here in guessing who will default on borrowings. And since not all the growing startups have access to the FICO score of an individual, it can be difficult to predict their repaying capability. So, small fintech companies try to come up with a novel way to evaluate the potential risk posed by lending money to consumers and then set the credit limits accordingly. One of the main parameters that could help us understand the creditworthiness of an individual is their salary. Therefore, in this dataset, we try to predict the income of the individual from basic demographic data and other parameters which can be easily obtained online or from any of the online merchants.

In this project, the objectives are to implement Support Vector Machines, Decision Trees and Boosting to predict GPU performance and salary of an individual.

TASK 1 – GPU Dataset

Dataset Description:

1. SGEMM GPU kernel performance dataset

   This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations. It has **4 dependent** variables and **14 independent** variables - the first 10 are ordinal and can only take up to 4 different powers of two values, and the 4 last variables are binary.

Exploratory Data Analysis:

Refer the code for EDA of GPU dataset. Please note that EDA part of GPU dataset is already explained in Assignment 1.

Data Preparation:

It is important to **scale** the features to a range which is centered around zero i.e. to **have 0 mean and 1 SD**. This is done so that the standard deviation of the features is in the same range. If a feature's variance is orders of magnitude more than the variance of other features, that particular feature might dominate other features in the dataset. To address this, **the data has been normalized by subtracting each value with its mean and divide by its standard deviation.**

The dataset is split into train and test set in the ratio of 70:30.

Experimentation:

Here, the target value must be considered as a binary value since all the below algorithms is run for variable classification. This is done by considering all the values of average runtime **above** the **median as 1** (this means computation takes more time) and **below the median as 0** (this means computation takes lesser time).

This dataset consists of all the possible combination of GPU kernel parameters. Since it represents the complete set of data for the given GPU configuration, it is safe to assume that anything above the median value as high time consuming.

Support Vector Machines:

A SVM model is built using 3 kernels – linear, gaussian and polynomial with default parameters.

In order to tune the hyperparameters, I have used the grid search algorithm. This helps in selecting the optimal values of these parameters.

To get a more accurate estimate of models' performance we perform multiple rounds of cross validation with different subsets from the same data using k-fold cross validation.

The confusion matrix, classification chart, cross validation results and accuracy of the models are as follows:

Linear Kernel:

```
Accuracy: 0.8234866968746369
Confusion Matrix:
 [[29293  5083]
 [ 7071 27409]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.85      0.83     34376
           1       0.84      0.79      0.82     34480

    accuracy                           0.82     68856
   macro avg       0.82      0.82      0.82     68856
weighted avg       0.82      0.82      0.82     68856
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.8234866968746369
Confusion Matrix:
 [[29293  5083]
 [ 7071 27409]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.85      0.83     34376
           1       0.84      0.79      0.82     34480

    accuracy                           0.82     68856
   macro avg       0.82      0.82      0.82     68856
weighted avg       0.82      0.82      0.82     68856
```

After gridsearch - with best parameters
(C:1, gamma: 0.1)

Wall time: ~25 minutes

Cross-validation results:

```
Scores:  [0.81507297 0.81964714 0.81960784 0.8211329  0.808061  ]
Accuracy: 0.82 (+/- 0.01)
```

Gaussian Kernel:

```
Accuracy: 0.9575345648890438
Confusion Matrix:
 [[33501   875]
 [ 2049 32431]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.97      0.96     34376
           1       0.97      0.94      0.96     34480

    accuracy                           0.96     68856
   macro avg       0.96      0.96      0.96     68856
weighted avg       0.96      0.96      0.96     68856
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.979261066573719
Confusion Matrix:
 [[33798   578]
 [  850 33630]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     34376
           1       0.98      0.98      0.98     34480

    accuracy                           0.98     68856
   macro avg       0.98      0.98      0.98     68856
weighted avg       0.98      0.98      0.98     68856
```

After gridsearch - with best parameters
(C:10, gamma: 0.1)

Wall time: ~8 minutes

Cross-validation results:

```
Scores:  [0.89087345 0.88194293 0.8745098  0.8869281  0.87211329]
Accuracy: 0.88 (+/- 0.01)
```

Polynomial Kernel:

```
Accuracy: 0.9045108632508423
Confusion Matrix:
 [[32398  1978]
 [ 4597 29883]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     34376
           1       0.94      0.87      0.90     34480

    accuracy                           0.90     68856
   macro avg       0.91      0.90      0.90     68856
weighted avg       0.91      0.90      0.90     68856
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.9182642035552457
Confusion Matrix:
 [[32416  1960]
 [ 3668 30812]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     34376
           1       0.94      0.89      0.92     34480

    accuracy                           0.92     68856
   macro avg       0.92      0.92      0.92     68856
weighted avg       0.92      0.92      0.92     68856
```

After gridsearch - with best parameters
(C:10, gamma: 0.1)

Wall time: ~28 minutes

```
Scores:  [0.82937685 0.8296846  0.82077922 0.81966605 0.81966605]
Accuracy: 0.82 (+/- 0.01)
```

In both Gaussian and Polynomial kernel, there was approx. 2% increase in the accuracy after tuning the hyperparameters.

From the above values calculated, Gaussian kernel is found to give the maximum accuracy of 97.9% at the lowest wall time of around 8 minutes.

Decision Trees:

Decision Trees is first built with default parameters. It is later pruned to reduce the complexity of the final classifier, thereby reducing overfitting and improving test accuracy. The confusion matrix, classification chart, cross validation results and accuracy of the models are as follows:

```
Accuracy: 0.9897031485999768
Confusion Matrix:
 [[34013   363]
 [  346 34134]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     34376
           1       0.99      0.99      0.99     34480

    accuracy                           0.99     68856
   macro avg       0.99      0.99      0.99     68856
weighted avg       0.99      0.99      0.99     68856
```

With default parameters

('criterion': 'gini', 'max_depth': None, 'max_leaf_nodes': None)

```
Accuracy: 0.9904002556059022
Confusion Matrix:
 [[34049   327]
 [  334 34146]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     34376
           1       0.99      0.99      0.99     34480

    accuracy                           0.99     68856
   macro avg       0.99      0.99      0.99     68856
weighted avg       0.99      0.99      0.99     68856
```

After Pruning - best parameters

('criterion': entropy, 'max_depth': 20, 'max_leaf_nodes': None)

Wall time: less than a second

Cross-validation results:

```
Scores:  [0.98761398 0.98926337 0.98864096 0.98758286 0.98786257]
Accuracy: 0.99 (+/- 0.00)
```

Boosting:

Boosting is a method of converting weak learners to strong learners.

```
Accuracy: 0.7932351574299988
Confusion Matrix:
 [[26527  7849]
 [ 6388 28092]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.77      0.79     34376
           1       0.78      0.81      0.80     34480

    accuracy                           0.79     68856
   macro avg       0.79      0.79      0.79     68856
weighted avg       0.79      0.79      0.79     68856
```

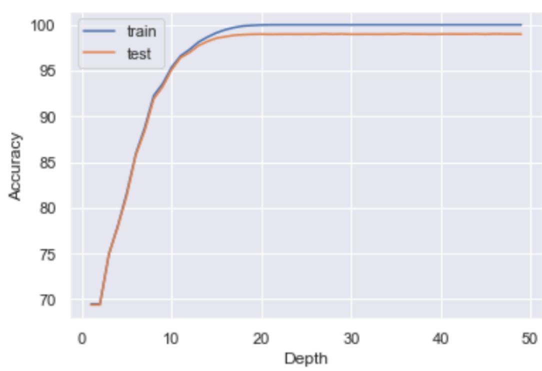
Wall time: 7.28 seconds

Cross-validation results:

```
Scores:  [0.79295428 0.78943765 0.78953101 0.79908505 0.79092493]
Accuracy: 0.79 (+/- 0.01)
```

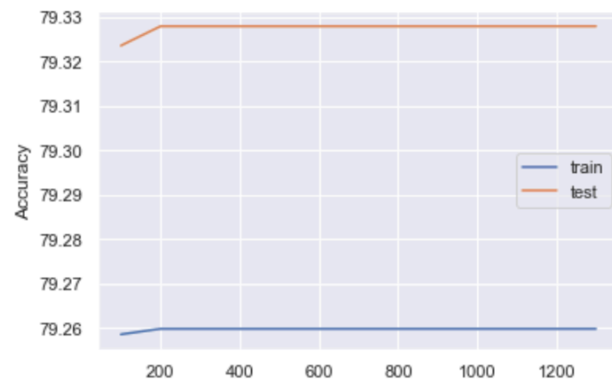
However, since we already had almost 100% accuracy in trees, there is no need to go for boosting in this case.

Learning Curve – Trees



Accuracy vs Depth

Learning Curve - Boosting



Accuracy vs No. of Estimators

From these graphs, it can be seen here that accuracy gets saturated at the depth of 20.

TASK 2

Dataset Description:

2. Income dataset

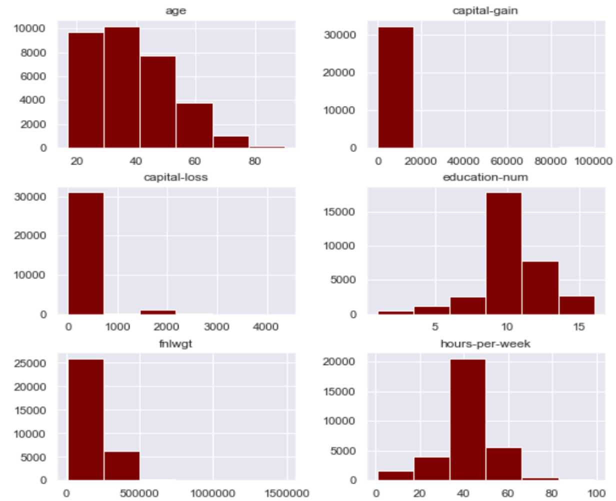
   There are 14 independent variables and 1 dependent variable – salary which is classified into 2 categories: <=50K and >50K

Exploratory Data Analysis:

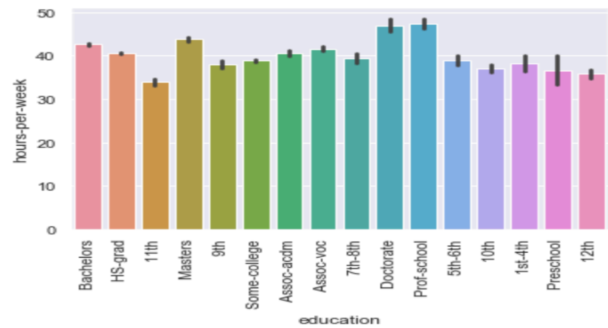
Before jumping in with the implementation of ML algorithm on your data, it is necessary to completely understand the given dataset and check for correlation among variables, skewness, outliers, inconsistent and missing values to build an efficient model. For this purpose, histograms of all the parameters and correlation matrix are plotted.

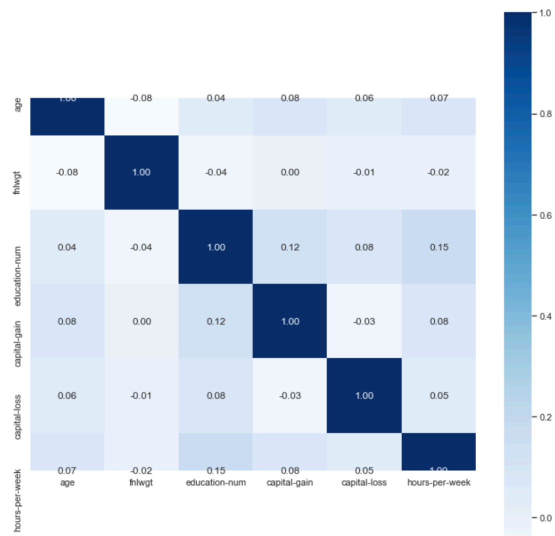
Skewness: From the histogram of the quantitative variables, it can be seen that those are not skewed, and the distribution seems fair.

Missing Values: No missing values in the numerical variables. However, variables like occupation and native country have missing values. Since these variables cannot be perfectly imputed, I have deleted those records from this dataset. We also notice that the countries other than United States is only 10% of the total records. So, I have only retained United States for this analysis.



Feature Distribution Plots





Correlation: For any ML model, we should ensure that there is no high correlation between the variables. This was confirmed from correlation matrix of this dataset that there exists no significant correlation between the variables.

Data Preparation:

Same as the previous dataset

Experimentation:

Here, the target variable consists of 2 values - <=50K which is considered as 0 and >50K as 1 for the analysis purpose.

Support Vector Machines:

A SVM model is built using 3 kernels – linear, gaussian and polynomial with default parameters. In order to tune the hyperparameters, I have used the grid search algorithm. This helps in selecting the optimal values of these parameters.

To get a more accurate estimate of models' performance we perform multiple rounds of cross validation with different subsets from the same data using k-fold cross validation.
The confusion matrix, classification chart, cross validation results and accuracy of the models are as follows:
Linear Kernel:

```
Accuracy: 0.847472299168975
Confusion Matrix:
 [[4030  307]
 [ 574  865]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4337
           1       0.74      0.60      0.66      1439

    accuracy                           0.85      5776
   macro avg       0.81      0.77      0.78      5776
weighted avg       0.84      0.85      0.84      5776
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.847472299168975
Confusion Matrix:
 [[4030  307]
 [ 574  865]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4337
           1       0.74      0.60      0.66      1439

    accuracy                           0.85      5776
   macro avg       0.81      0.77      0.78      5776
weighted avg       0.84      0.85      0.84      5776
```

After gridsearch - with best parameters
(C:1, gamma: 0.1)

Wall time: ~12 seconds
Cross-validation results:

```
Scores:  [0.84866469 0.83636364 0.84267161 0.83487941 0.84786642]
Accuracy: 0.84 (+/- 0.01)
```

Gaussian Kernel:

```
Accuracy: 0.8459141274238227
Confusion Matrix:
 [[4020  317]
 [ 573  866]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4337
           1       0.73      0.60      0.66      1439

    accuracy                           0.85      5776
   macro avg       0.80      0.76      0.78      5776
weighted avg       0.84      0.85      0.84      5776
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.8492036011080333
Confusion Matrix:
 [[4022  315]
 [ 556  883]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4337
           1       0.74      0.61      0.67      1439

    accuracy                           0.85      5776
   macro avg       0.81      0.77      0.79      5776
weighted avg       0.84      0.85      0.84      5776
```

After gridsearch - with best parameters
(C:100, gamma: 0.001)

Wall time: ~8 seconds
Cross-validation results:

```
Scores:  [0.85459941 0.84675325 0.84675325 0.83599258 0.84378479]
Accuracy: 0.85 (+/- 0.01)
```

Polynomial Kernel:

```
Accuracy: 0.8355263157894737
Confusion Matrix:
 [[4050  287]
 [ 663  776]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.93      0.90      4337
           1       0.73      0.54      0.62      1439

    accuracy                           0.84      5776
   macro avg       0.79      0.74      0.76      5776
weighted avg       0.83      0.84      0.83      5776
```

With default parameters
(C:1, gamma: 'Scale')

```
Accuracy: 0.8325831024930748
Confusion Matrix:
 [[3998  339]
 [ 628  811]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.92      0.89      4337
           1       0.71      0.56      0.63      1439

    accuracy                           0.83      5776
   macro avg       0.78      0.74      0.76      5776
weighted avg       0.82      0.83      0.83      5776
```

After gridsearch - with best parameters
(C:10, gamma: 0.01)

Wall time: ~6 seconds

In all the kernels, there were minor difference in the accuracy after tuning the hyperparameters.

From the above values calculated, both Linear and Gaussian kernels are found to give the maximum accuracy of around 85% at the lowest wall time of around 10 seconds.

Decision Trees:

Decision Trees is first built with default parameters. It is later pruned to reduce the complexity of the final classifier, thereby reducing overfitting and improving test accuracy. Pruning is done by altering the depth and leave nodes and the best results were identified with the help of gridsearchcv function. The confusion matrix, classification chart, cross validation results and accuracy of the models are as follows:

```
Accuracy: 0.8043628808864266
Confusion Matrix:
 [[3760  577]
 [ 553  886]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      4337
           1       0.61      0.62      0.61      1439

    accuracy                           0.80      5776
   macro avg       0.74      0.74      0.74      5776
weighted avg       0.81      0.80      0.80      5776
```

With default parameters

('criterion': 'gini', 'max_depth': None, 'max_leaf_nodes': None)

```
Accuracy: 0.846606648199446
Confusion Matrix:
 [[4012  325]
 [ 561  878]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4337
           1       0.73      0.61      0.66      1439

    accuracy                           0.85      5776
   macro avg       0.80      0.77      0.78      5776
weighted avg       0.84      0.85      0.84      5776
```

After Pruning - best parameters

('criterion': 'gini', 'max_depth': 10, 'max_leaf_nodes': 20)

Wall time: less than a second

Cross-validation results:

```
Scores:  [0.85200297 0.84044527 0.83302412 0.84489796 0.84526902]
Accuracy: 0.84 (+/- 0.01)
```

Boosting:

Boosting is a method of converting weak learners to strong learners. We use AdaBoost algorithm for this purpose.

```
Accuracy: 0.8623614958448753
Confusion Matrix:
 [[4026  311]
 [ 484  955]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      4337
           1       0.75      0.66      0.71      1439

    accuracy                           0.86      5776
   macro avg       0.82      0.80      0.81      5776
weighted avg       0.86      0.86      0.86      5776
```

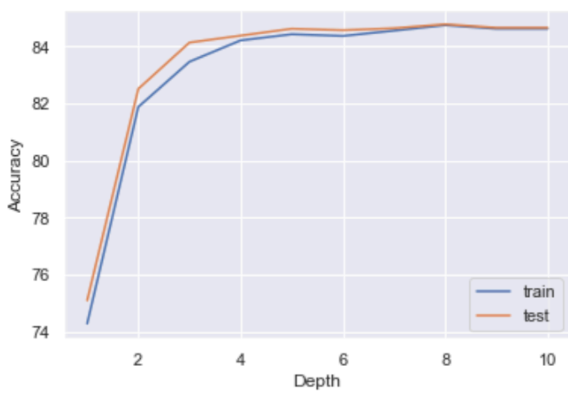
Wall time: 12.3 seconds

Cross-validation results:

```
Scores:  [0.87203264 0.86419295 0.8567718  0.86085343 0.86382189]
Accuracy: 0.86 (+/- 0.01)
```

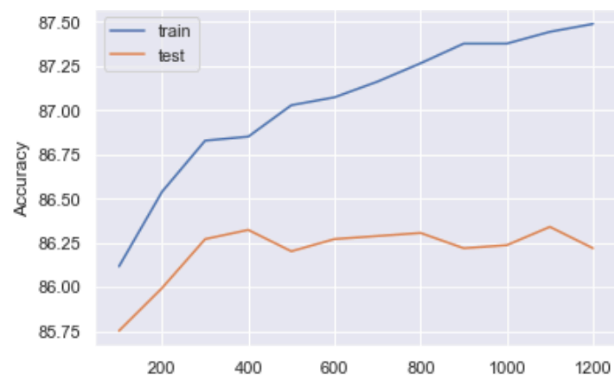
There is a 2% increase in accuracy when we use boosting

Learning Curve – Trees



Accuracy vs Depth

Learning Curve - Boosting



Accuracy vs No. of Estimators

From these graphs, it can be seen here that accuracy gets saturated at the depth of 8.

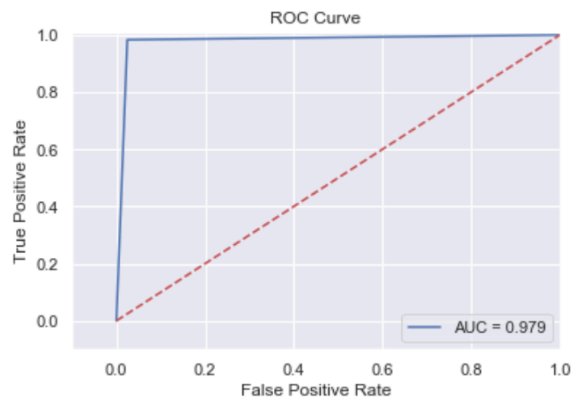
CONCLUSION:

GPU Dataset:

Following are the accuracy and test errors of the different models:

Models	Test Accuracy	Test Error
SVM - Linear	82.34%	0.18
SVM - Gaussian	97.92%	0.02
SVM - Polynomial	90.54%	0.09
Decision Trees	98.90%	0.01
Pruned Decision Trees	99.04%	0.01
Boosting - AdaBoost	79.32%	0.21

Of the 3 different algorithms used, Pruned Decision Trees gave the best results for this dataset.
Here, the boosting doesn't improve the result since each weak classifier is dedicated to fix its predecessors' shortcomings, the model may pay too much attention to outliers. Also, decision trees already gave perfect classification for the train and test sets.

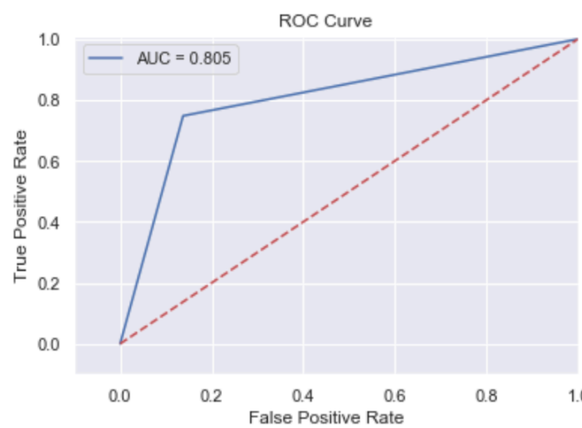


ROC Curve – SVM (Gaussian)

Income Dataset:

Following are the accuracy and test errors of the different models:

Models	Test Accuracy	Test Error
SVM - Linear	84.74%	0.152
SVM - Gaussian	84.92%	0.150
SVM - Polynomial	83.25%	0.167
Decision Trees	80.43%	0.195
Pruned Decision Trees	84.66%	0.153
Boosting - AdaBoost	86.23%	0.137



ROC Curve – AdaBoost

Of the 3 different algorithms used, Boosting gave the best results for this dataset since AdaBoost identifies misclassified data points, increasing their weights so that the next classifier will pay extra attention to get them right.
Though ~85% accuracy is decent enough, we don't know if this dataset covers all the states of USA. So, we may not be sure if this model would accurately predict the income of any individual from the United States. It is possible to get better results for the real-time data if the sample represent the whole population. Apart from that, the above accuracy can itself be improved by implementing advanced ML techniques like neural networks.