Introduction:

In this project, we used IBM Attrition and Occupancy detection datasets to implement ANN and KNN algorithm. We used caret package to implement both the algorithms.

Dataset Information:

IBM Attrition:

This is a fictional dataset created by IBM data scientists. The dataset contains variables such as monthly income, overtime hours, tenure etc. We must develop a model to understand whether the employee will the organization or not.

This problem is interesting as it helps us understand what are the important variables that make an employee leave an organization.

Occupancy Detection:

Problem statement: Experimental data is used to predict room occupancy (binary classification) from various parameters such as Temperature, Light, and CO2

The dataset helps us in predicting if a room is occupied or not, depending on the room conditions

Data Preparation:

IBM Attrition:

Performed the following operations on the dataset before designing the model with ANN and KNN algorithms.

- The dataset is cleaned so omitting missing values didn't change the row count
- Dropped the categorical variables with only 1 just level
- Converted categorical variables into dummy variables
- Scale the entire dataset using normalize function
- Removed class imbalance using smote methods.
- Partitioned data into 50:50 ratio of Train and Test Dataset

Occupancy Detection:

Performed the following operations on the dataset before designing the model with ANN and KNN algorithms.

- Converted the continuous variable Occupancy into a factor variable with 2 levels
- Normalized all the predictor columns (Humidity, Temperature, CO2, Light, Humidity Ratio)
- Partitioned data into 50:50 ratio of Train and Test Dataset

Algorithm Implementation:

Artificial Neural Network (ANN) on IBM Attrition Dataset:
Trained a model with one hidden layer and 10 nodes, and weight 0.1 using caret package and then tested against Test Dataset. Confusion Matrix for both models are as follows:

```
For Train Dataset
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  747   1
       Yes   0 663

               Accuracy : 0.9993
                 95% CI : (0.9961, 1)
    No Information Rate : 0.5294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9986
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9985
         Pos Pred Value : 0.9987
         Neg Pred Value : 1.0000
             Prevalence : 0.5294
         Detection Rate : 0.5294
   Detection Prevalence : 0.5301
      Balanced Accuracy : 0.9992

       'Positive' Class : No
```

```
For Test Dataset
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  292  19
       Yes  27 265

               Accuracy : 0.9237
                 95% CI : (0.8996, 0.9436)
    No Information Rate : 0.529
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8471
 Mcnemar's Test P-Value : 0.302

            Sensitivity : 0.9154
            Specificity : 0.9331
         Pos Pred Value : 0.9389
         Neg Pred Value : 0.9075
             Prevalence : 0.5290
         Detection Rate : 0.4842
   Detection Prevalence : 0.5158
      Balanced Accuracy : 0.9242

       'Positive' Class : No
```

For Train Dataset: The model accuracy is pretty good (99%) for training dataset. From the confusion Matrix, No Type 1 error and one record were classified incorrectly. Model sensitivity is 1. AUC is 1

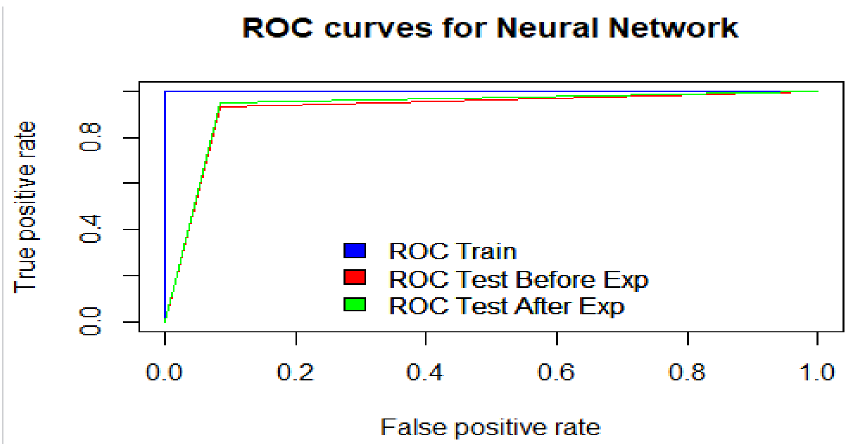
For Test Dataset: The model accuracy for Test Dataset is 92% with sensitivity and specificity 91% and 93% respectively. The Kappa is 0.84. AUC is 0.89

To understand the impact of nodes, hidden layers, and an activation function, we did some experimentation with these parameters and regenerated the models.

Model accuracy with Sigmoid activation function was 1 whereas for tanh activation function was 0.99.

The Model was regenerated with 2 hidden layers with nodes 15 and 10 each and initial weight decay reduced to 0.1 from 0.5. The Model AUC is 1. The newly generated model was tested against Test dataset. Model accuracy for Test dataset increased from 92% to 93%.

The ROC curves for models are as shown below which indicate the area under the curve for train model, test model before experimentation and after experimentation. The ROC curve is as shown below.



Artificial Neural Network (ANN) on Occupancy Dataset:

Trained a model with one hidden layer and 10 nodes, and weight 0.1 using caret package and then tested against Test Dataset. Confusion Matrix for both models are as follows:

For Train Dataset: The model accuracy is pretty good (98%) for training dataset. From the confusion Matrix, we have 2 observations of Type 1 error and 2 observations of type 2 error. Model sensitivity is 0.9744.

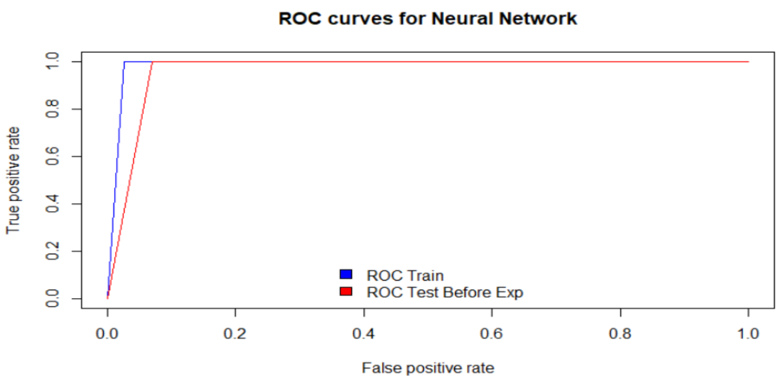
For Test Dataset: The model accuracy for Test Dataset is 94% with sensitivity and specificity 93% and 100% respectively. The Kappa is 0.7953.

To understand the impact of nodes, hidden layers, and an activation function, we did some experimentation with these parameters and regenerated the models.

```
For Train Dataset
> print(conf1)
Confusion Matrix and Statistics

          Reference
Prediction  yes   no
       yes 2855    2
       no    75 1139

               Accuracy : 0.9811
                 95% CI : (0.9764, 0.985)
    No Information Rate : 0.7197
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.954
 Mcnemar's Test P-Value : 2.303e-16

            Sensitivity : 0.9744
            Specificity : 0.9982
         Pos Pred Value : 0.9993
         Neg Pred Value : 0.9382
             Prevalence : 0.7197
         Detection Rate : 0.7013
   Detection Prevalence : 0.7018
      Balanced Accuracy : 0.9863

       'Positive' Class : yes
```

```
For Test Dataset
> print(conf2)
Confusion Matrix and Statistics

          Reference
Prediction  yes   no
       yes 3243    0
       no   241  588

               Accuracy : 0.9408
                 95% CI : (0.9331, 0.9479)
    No Information Rate : 0.8556
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7953
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9308
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.7093
             Prevalence : 0.8556
         Detection Rate : 0.7964
   Detection Prevalence : 0.7964
      Balanced Accuracy : 0.9654

       'Positive' Class : yes
```

The ROC curves for models are as shown below which indicate the area under the curve for train model, test model before experimentation. The ROC curve is as shown below.

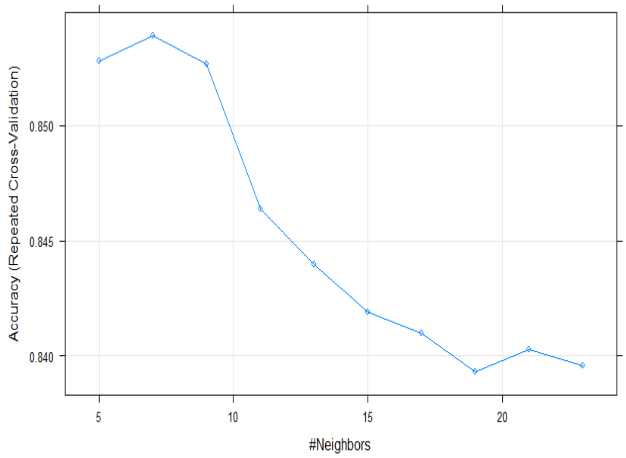


KNN Algorithm

K Nearest Neighbors (KNN) on IBM Attrition Dataset:

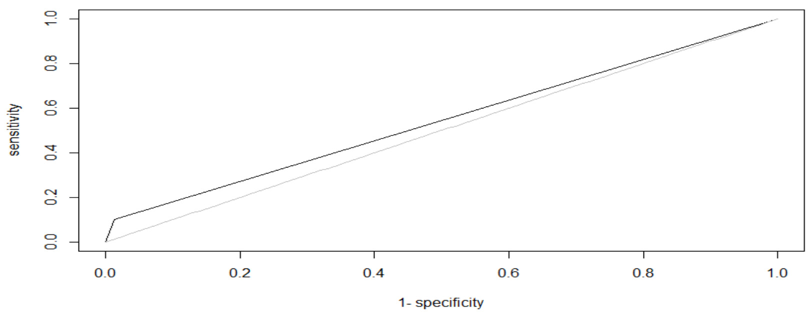
Trained a K nearest model on 50% data and tested the model on the rest 50% of the data.

Used repeated cross-validation to avoid overfitting on training data and reduce variance.




```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  609 106
       Yes   8  12

               Accuracy : 0.8449
                 95% CI : (0.8167, 0.8703)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 0.3663

                  Kappa : 0.1336
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9870
            Specificity : 0.1017
         Pos Pred Value : 0.8517
         Neg Pred Value : 0.6000
             Prevalence : 0.8395
         Detection Rate : 0.8286
   Detection Prevalence : 0.9728
      Balanced Accuracy : 0.5444

       'Positive' Class : No
```

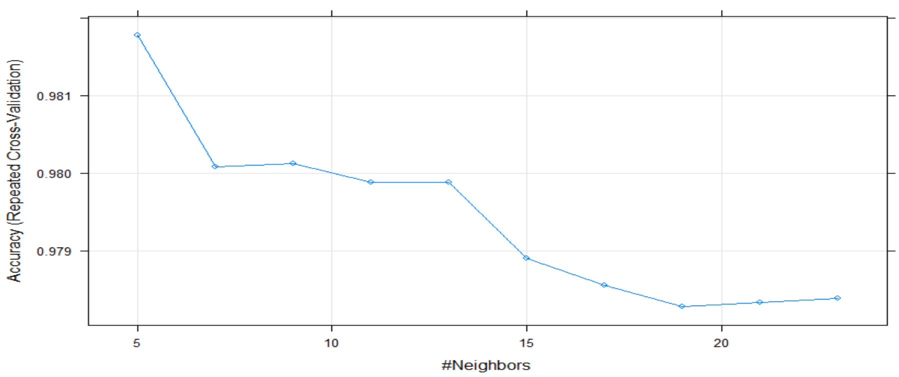
Using repeated cross-validation, the best accuracy was achieved with K = 7.
The confusion matrix shows an accuracy of 84.49% but the Specificity of the model is low at .1017



The AUC score of the model is also very low at 0.5443645

K Nearest Neighbors (KNN) on Occupancy Detection Dataset:

Trained a K nearest model on 50% data and tested the model on the rest 50% of the data.
Used repeated cross-validation to avoid overfitting on training data and reduce variance.



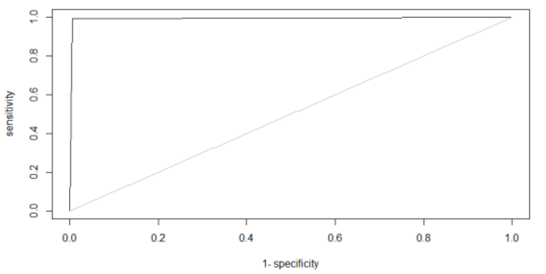
```
k-Nearest Neighbors

4071 samples
   5 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 3664, 3664, 3664, 3664, 3664, 3664, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9817732  0.9555820
   7  0.9800781  0.9514888
   9  0.9801272  0.9516735
  11  0.9798817  0.9511274
  13  0.9798816  0.9511916
  15  0.9788990  0.9488368
  17  0.9785550  0.9480397
  19  0.9782848  0.9474026
  21  0.9783339  0.9475247
  23  0.9783830  0.9476461

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 5.
```

```
k-Nearest Neighbors

735 samples
 85 predictor
  2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 661, 662, 661, 661, 661, 662, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.8527914  0.19087514
   7  0.8538742  0.16243935
   9  0.8526580  0.14367416
  11  0.8463973  0.08359400
  13  0.8439556  0.06027367
  15  0.8419194  0.04343927
  17  0.8409622  0.03619915
  19  0.8393314  0.01746633
  21  0.8402755  0.02126545
  23  0.8395998  0.01465638

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 7.
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3464    4
         1   20  584

               Accuracy : 0.9941
                 95% CI : (0.9912, 0.9962)
    No Information Rate : 0.8556
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9764
 Mcnemar's Test P-Value : 0.0022

            Sensitivity : 0.9943
            Specificity : 0.9932
         Pos Pred Value : 0.9988
         Neg Pred Value : 0.9669
             Prevalence : 0.8556
         Detection Rate : 0.8507
   Detection Prevalence : 0.8517
      Balanced Accuracy : 0.9937

       'Positive' Class : 0
```

Using repeated cross-validation, the best accuracy was achieved with K = 5.

The confusion matrix shows an accuracy of 99.41%, both the Sensitivity and Specificity are high at 0.9943 and 0.9932 respectively.



The AUC score of the model is very high at 0.9937284.

Comparison: Based on ROC and AUC for both the algorithms, we can clearly observe ANN is doing well as compare to KNN. KNN performance is better for occupancy dataset but not for IBM attrition.

Overall comparison – Highlighted values shows the best model with algorithms

Model	Accuracy IBM	AUC IBM	Accuracy Occupancy	AUC Occupancy
SVM	89.31%	0.7486354	97%	0.9855
Decision Tree	84.54%	0.5893736	97.86%	0.9825
Boosting	87.04%	0.6497767	96.55%	0.9874
KNN	84.49%	0.5443645	99.41%	0.9937284
ANN	89.37%	0.8964721	91.23%	0.94877

References:

https://gist.github.com/primaryobjects/d02b93f1e539a9dd2c85,
http://topepo.github.io/caret/train-models-by-tag.html#Neural_Network
https://beckmw.wordpress.com/tag/nnet/