Predictive Modelling on Social Media Usage

Problem statement: The aim of this study is to develop a predictive model to identify whether a person is likely to be addicted to social media based on their social media usage.

Model Approach:

The data is imported and cleaned in the following manner.

- It then removes a column from the data that represents the number of times an app was opened. The next step removes any missing values from the numeric columns of the data by replacing them with the mean value of the column. After this, the code splits the data into two parts for training and testing. The target variable is extracted from columns 3 to 15 of the data, and the 12th column is removed from the target variable. This will allow the remaining columns to be used as predictors for the model.

Creating a stratified random sample of row indices for the testing set

- The sample.split() function is used to randomly split the data into training and testing sets based on the target variable Social.Media.Addiction, with a split ratio of 70% for the training set and 30% for the testing set.

- The training and testing sets are then created using subset() function on the original dataset, based on the row indices generated by sample.split() function.

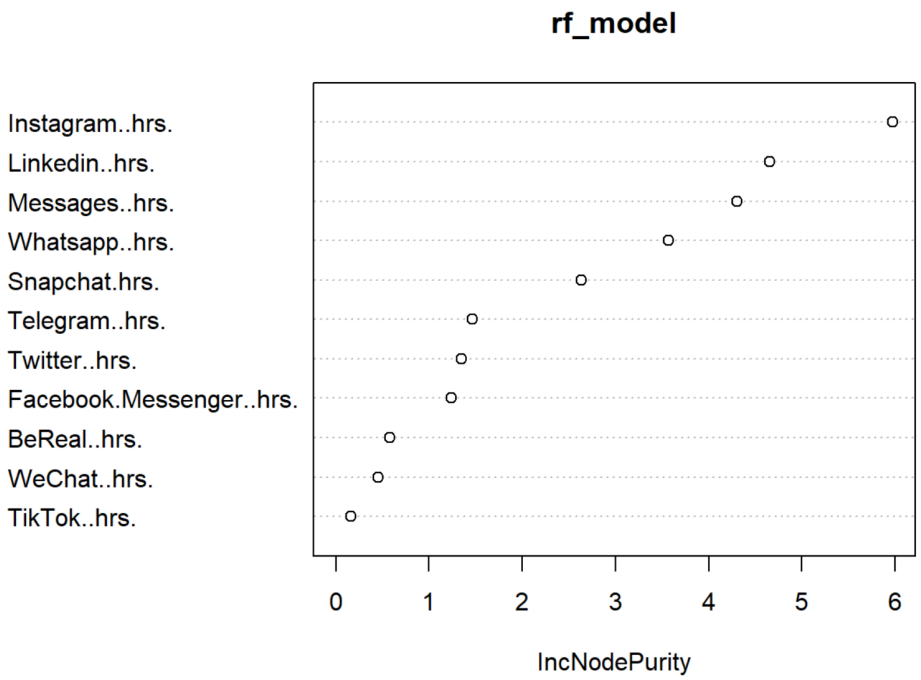
- The predictive variables and the target variable are then separated from both the training and testing sets using indexing. The target variable is transformed into dummy variables using dummyVars() function from the caret package, which is a preprocessing step required for some models.

- Finally, the summary() function is used to display the summary statistics of the target variable and the predictive variables in the dataset, after removing the target variable Social.Media.Addiction

Feature Selection using Random Forest Algorithm:

- This part of the code is related to feature selection and dimensionality reduction using Random Forest algorithm. The first step is to convert the one-hot encoded target variable (dummy_Ytrain_data) to a factor variable (Ytrain_data_factored) using the apply function with the which.max function as an argument.

- After that, a random forest model (rf_model) is trained using the training data (Xtrain_data and Ytrain_data_factored). The importance of each feature is then calculated using the varImpPlot and importance functions.



- The variable importance scores are stored in the imp object, and the importance values for each feature are extracted and stored in imp_values. The features with an importance score greater than 2 are selected and stored in important_features.

- Finally, the selected important features are extracted from the training data and stored in Xtrain_data_selected. The head function is used to display the first few rows of the new dataset.

```
##           Whatsapp..hrs.          Instagram..hrs.           Snapchat.hrs.
##                3.5718282                5.9765177                2.6312841
##          Telegram..hrs. Facebook.Messenger..hrs.           BeReal..hrs.
##                1.4599586                1.2344801                0.5770531
##            TikTok..hrs.             WeChat..hrs.          Twitter..hrs.
##                0.1648888                0.4570194                1.3476037
##          Linkedin..hrs.           Messages..hrs.
##                4.6544925                4.3030171
```

Fitting the model

- Logistic regression is being built to predict the target variable based on the selected important features. Before building the model, the data is scaled using the scale() function and converted

to a data frame using as.data.frame(). The outliers are then removed from the scaled data using the winsorize() function and converted back to a data frame.

- The target variable is then converted from a factor into binary form with values of 0 and 1 using the ifelse() function, which sets the value of 1 if the original value was 1 and 0 otherwise.

- Finally, the model is fit using the glm() function, with the target variable and important features as the formula, and the data as the input. The family is set to binomial, indicating that it is a logistic regression model. The summary() function is used to print out a summary of the model, which includes information such as coefficients, standard errors, z-values, and p-values for each of the variables in the model.

```
##
## Call:
## glm(formula = Ytrain_data_binary ~ ., family = binomial, data = Xtrain_data_winsorized)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3848  -0.9514   0.4281   0.9528   1.9587
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.49148    1.15146   1.295 0.195220
## Whatsapp..hrs.   0.08642    0.28662   0.302 0.763025
## Instagram..hrs.  1.13858    0.31473   3.618 0.000297 ***
## Snapchat.hrs.   -0.72794    0.60678  -1.200 0.230268
## Linkedin..hrs.   2.28894    0.72315   3.165 0.001550 **
## Messages..hrs.   1.90611    3.73330   0.511 0.609652
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 166.46  on 121  degrees of freedom
## Residual deviance: 134.94  on 116  degrees of freedom
## AIC: 146.94
##
## Number of Fisher Scoring iterations: 4
```

- Based on the output, it seems that only the Instagram and Linkedin hours are significant at the 0.05 level of significance, as indicated by the corresponding p-values. The other features (Whatsapp hours, Snapchat hours, and Messages hours) do not appear to have a significant impact on the target variable.

- Selecting only the significant features is a common approach in logistic regression, as it can improve the interpretability and performance of the model.

Building the model again with significant features i.e Instagram and Linkdin

```
##
## Call:
## glm(formula = Ytrain_data_binary ~ ., family = binomial, data = Xtrain_data_winsorized_new)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0017  -0.9650   0.5536   0.8955   1.8518
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.6642     0.2412   2.754  0.00589 **
## Instagram..hrs.   0.9606     0.2453   3.917 8.98e-05 ***
## Linkedin..hrs.    1.1365     0.5291   2.148  0.03170 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 166.46  on 121  degrees of freedom
## Residual deviance: 141.69  on 119  degrees of freedom
## AIC: 147.69
##
## Number of Fisher Scoring iterations: 4
```

- Based on this output, we can conclude that Instagram..hrs. and Linkedin..hrs. are significant predictors of the dependent variable Ytrain_data_binary. However, it's worth noting that there may be other important predictors that were not included in this model.

Evaluating the model with test data using Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26  4
##          1 11 12
##
##                Accuracy : 0.717
##                  95% CI : (0.5765, 0.8321)
##     No Information Rate : 0.6981
##     P-Value [Acc > NIR] : 0.4484
##
##                   Kappa : 0.4027
##
##  Mcnemar's Test P-Value : 0.1213
##
##             Sensitivity : 0.7027
##             Specificity : 0.7500
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.5217
##              Prevalence : 0.6981
##          Detection Rate : 0.4906
##    Detection Prevalence : 0.5660
##       Balanced Accuracy : 0.7264
##
##        'Positive' Class : 0
##
```

- The model correctly predicted 38 out of 53 instances of churn (positive class) and 12 out of 23 instances of non-churn (negative class).
- The accuracy of the model is 0.717, which means that the model correctly predicted 71.7% of the instances.
- The Kappa coefficient measures the agreement between the model and the actual outcomes, and its value is 0.4027. A value closer to 1 indicates better agreement.
- The sensitivity of the model is 0.75, which means that it correctly identifies 75% of the instances of churn.
- The specificity of the model is 0.7027, which means that it correctly identifies 70.27% of the instances of non-churn.
- The positive predictive value (PPV) of the model is 0.5217, which means that among the instances predicted as churn, only 52.17% were actually churned.
- The negative predictive value (NPV) of the model is 0.8667, which means that among the instances predicted as non-churn, 86.67% were actually non-churned.

Overall, the model seems to perform moderately well in predicting the churn/non-churn outcomes as it has accuracy of 77 percent. However, there is still room for improvement in the performance, especially in terms of PPV and specificity.