**Student: Seif Kungulio**

**Date: 04/13/2025**

**Subject: Project 5**

**Class: DSCI 512**

**Section: 01W**

**Instructor: Dr. Nengbing Tao**

**File Name: Project5\_Kungulio\_Seif.docx**

1. Load the dataset bike.csv into memory. Then split the data into a training set containing 2/3 of the original data (test set containing remaining 1/3 of the original data).

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

1. Build a tree model using function tree().
   1. The response is count and the predictors are season, holiday, workingday, temp, atemp, humidity, windspeed, casual, and registered.

A computer screen shot of a computer code

AI-generated content may be incorrect.

* The decision tree mainly depends on the registered and casual variables, which seem to be the most direct predictors of bike rental counts.
* With 8 terminal nodes, the model reflects a moderate level of complexity while still managing to avoid overfitting.
* The Residual Mean Deviance (1875) and the distribution of residuals give insight into how well the model performs. Although there are some notable outliers (like -158 and +259.7), the residuals generally cluster around zero.
* Overall, the tree offers a simplified yet understandable view of the data relationships. However, its predictive accuracy could potentially be enhanced through hyperparameter tuning or by exploring more advanced models such as random forests or gradient boosting.
  1. Perform cross-validation to choose the best tree by calling cv.tree().

A blue and purple text

AI-generated content may be incorrect.

* 1. Plot the model results of b) and determine the best size of the optimal tree.

A graph with a red mark

AI-generated content may be incorrect.

* The plot indicates that the ideal tree size is 4.
* At this point, the deviance reaches its lowest value before leveling off or slightly rising with larger trees. This suggests the model achieves a good balance—complex enough to capture key patterns without being overly complicated or prone to overfitting.
  1. Prune the tree by calling prune.tree() function with the best size found in c).

A black background with white text and purple text

AI-generated content may be incorrect.

* 1. Plot the best tree model.

A computer screen shot of a computer

AI-generated content may be incorrect.

* The pruned decision tree, refined to 4 terminal nodes through cross-validation, forecasts a numeric target variable. It begins by splitting on the “registered” variable, with the initial split at “registered < 200.5”. Subsequent splits lead to four final predicted values: 40.02, 180.40, 332.80, and 553.20.
* This streamlined version of the tree is more interpretable, reduces the risk of overfitting, and maintains solid predictive power by emphasizing the most impactful decision points.
  1. Compute the test error using the test data set.

A screenshot of a computer program

AI-generated content may be incorrect.

* The Mean Squared Error (MSE) of 4762.857 gives a numerical indication of the average difference between the model’s predictions and the actual values in the test dataset.
* Whether this MSE is considered good or bad depends on the scale of the target variable (**count**).
  + For example, if **count** values usually fall in the thousands, this level of error might be reasonable. But if the values are much smaller, the error could be considered quite large in comparison.

1. Build a random forest model using function randomForest()
   1. The response is count and the predictors are season, holiday, workingday, temp, atemp, humidity, windspeed, casual, and registered.

A computer screen shot of a computer code

AI-generated content may be incorrect.

* The Random Forest model performs exceptionally well on the training data, accounting for 99.6% of the variance and showing a low residual error of 130.3723. However, this strong performance might suggest overfitting, so it's important to validate the model on unseen test data.
* Using 500 trees and 3 variables at each split, the model strikes a good balance between precision and computational efficiency.
* To determine which features, have the greatest impact on predictions, tools like “**importance()**” or “**varImpPlot()**” can be used to assess variable importance.
  1. Compute the test error using the test data set.

A computer screen shot of a code

AI-generated content may be incorrect.

* The test Mean Squared Error (MSE) of 177.4572 reflects strong predictive accuracy on new data, although it's slightly higher than the training error of 130.3723—indicating a minor degree of overfitting.
* Overall, the model demonstrates solid performance, with only a small drop in accuracy from training to testing, which is a normal and expected outcome in most modeling scenarios.
  1. Extract variable importance measure using importance() function.

A screenshot of a computer

AI-generated content may be incorrect.

* The variable “registered” plays the most significant role in predicting the target variable “count”, with “casual” and “workingday” also contributing notably.
* In contrast, features such as holiday and windspeed have little influence on the model’s predictive capability.
* These findings can guide model simplification by removing less impactful variables and highlight the main drivers behind the predictions.
  1. Plot the variable importance using function varImpPlot(). Which are the top 2 important predictors in this model?

A screenshot of a graph

AI-generated content may be incorrect.

* “Registered” and “casual” are the top predictors, playing a major role in improving the model's accuracy and reducing variance.
* The model relies heavily on these two variables, indicating they are the primary factors affecting the target variable, “count”.