Tree Regression

Regression model evaluation metrics

The **MSE**, **MAE**, **RMSE**, and **R-Squared** metrics are mainly used to evaluate the prediction error rates and model performance in regression analysis.

- **MAE (Mean absolute error)** represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.
- **MSE (Mean Squared Error)** represents the difference between the original and predicted values extracted by squared the average difference over the data set.
- **RMSE (Root Mean Squared Error)** is the error rate by the square root of MSE.
- **R-squared (Coefficient of determination)** represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

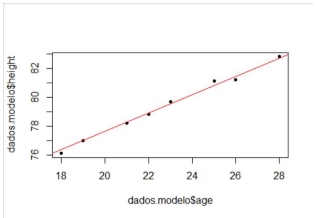
Linear Regression and Regression Trees - An Example

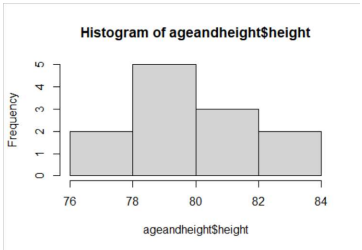
```
library(readxl)
library(rpart)
library(rpart.plot)
#Upload the data
ageandheight <- read_excel("ageandheight.xlsx", sheet = "Hoja2")
lmHeight = lm(height~age, data = ageandheight) #Create the linear regression model
summary(lmHeight) #Review the results
#Height=64.93+0.64*age
amostra.modelo <-sample (1:nrow(ageandheight),as.integer(0.7*nrow(ageandheight)))
dados.modelo <- ageandheight[amostra.modelo,]
dados.teste <- ageandheight[-amostra.modelo,]

plot(dados.modelo$age,dados.modelo$height, pch=20)
abline(lmHeight$coefficients[1],lmHeight$coefficients[2], col='red')
summary(dados.modelo)
hist(ageandheight$height)

arv.regr <- rpart(height ~ age,ageandheight)
arv.regr
prev.arv <- predict(arv.regr,dados.teste)
prev.arv

arv.regr <- rpart(height ~., ageandheight)
arv.regr
rpart.plot(arv.regr, digits = 3)
prev.arv <- predict(arv.regr,dados.teste)
prev.arv
```






```
#. View the Regression Trees plot
rpart.plot(arv.regr, digits = 3)
rpart.plot(arv.regr, digits = 3, fallen.leaves = TRUE, type = 3, extra = 101)

 # Mean absolute error - MAE
MAE <-mean(abs(prev.arv - dados.teste$height))
print("MAE:"); MAE
# Root Mean Squared - RMSE
RMSE <- sqrt(mean((prev.arv - dados.teste$height)^2))
print("RMSE:");RMSE
```

Cross-Validation

Cross-validation algorithm:

- Reserve a small sample of the data set
- Build (or train) the model using the remaining part of the data set
- Test the effectiveness of the model on the reserved sample of the data set. If the model works well on the test data set, then it’s good.

Cross-Validation Methods

- **Hold Out** approach consists of randomly splitting the data into two sets: one set is used to train the model and the remaining other set sis used to test the model. The process works as follow:
    1. Build (train) the model on the training data set
    2. Apply the model to the test data set to predict the outcome of new unseen observations
    3. Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values.
- **K-fold cross-validation -** The k-fold cross-validation method evaluates the model performance on different subset of the training data and then calculate the average prediction error rate. The algorithm is as follow:
    1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
    2. Reserve one subset and train the model on all other subsets
    3. Test the model on the reserved subset and record the prediction error
    4. Repeat this process until each of the k subsets has served as the test set.
    5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

    K-fold cross-validation (CV) is a robust method for estimating the accuracy of a model.

    The most obvious advantage of k-fold CV compared to LOOCV is computational. A less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV (James et al. 2014).

    Typical question, is how to choose right value of k?
- **Repeated K-fold cross-validation-** The process of splitting the data into k-folds can be repeated a number of times, this is called repeated k-fold cross validation. The final model error is taken as the mean error from the number of repeats.
- **Leave One Out Cross Validation** method works as follow:
    1. Leave out one data point and build the model on the rest of the data set
    2. Test the model against the data point that is left out at step 1 and record the test error associated with the prediction
    3. Repeat the process for all data points
    4. Compute the overall prediction error by taking the average of all these test error estimates recorded at step 2.

Models Evaluation Metrics

- Get a reliable estimate of a model's expected forecast error

Regression Model performance metrics

Regression model's performance assessment is based in determining the accuracy of the model on predicting the outcome for new unseen observations – Test Data - not used to build the model.

The basic strategy consists in:

- Build the model on a training data set
- Apply the model on a new test data set to make predictions
- Compute the prediction errors

Statistical metrics for quantifying the overall quality/performance of regression models include:

- **R-squared (R2),** representing the squared correlation between the observed outcome values and the predicted values by the model. The higher the adjusted R2, the better the model.
- **Root Mean Squared Error (RMSE),** which measures the average prediction error made by the model in predicting the outcome for an observation. That is, the average difference between the observed known outcome values and the values predicted by the model. The lower the RMSE, the better the model performance.
- **Mean Absolute Error (MAE),** an alternative to the RMSE that is less sensitive to outliers. It corresponds to the average absolute difference between observed and predicted outcomes. The lower the MAE, the better the model performance.

Machine Learning Regression Models Evaluation Metrics

- **Root Mean Squared Error (RMSE)** measures the average magnitude of the error by taking the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i are the observations, $\hat{y}$ the predicted values of a variable, and n the number of observations available for analysis

RMSE is a good measure of accuracy, but only to compare prediction errors of different models or model configurations for a particular variable and not between variables, as it is scale-dependent.

- **Mean Absolute Error (MAE)** is the sum of the absolute differences between predictions and actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Coefficient of Determination (R2) -** is the criterion generally used in linear regression to test the adjustment of the model.

```
# Mean absolute error - MAE
MAE <-mean(abs(prev.arv - dados.teste$height))
print("MAE:"); MAE
# Root Mean Squared - RMSE
RMSE <- sqrt(mean((prev.arv - dados.teste$height)^2))
print("RMSE:");RMSE
```

Machine Learning Classification Models Evaluation Metrics

- **Confusion matrix for binary classification**
- **What is a Confusion Matrix?**
  - A confusion matrix is a summary of prediction results on a classification problem.
  - The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

		Predicted	
Actual		Negative	Positive
	Negative	TN	FP
	Positive	FN	TP

  - The confusion matrix shows the ways in which your classification model is confused when it makes predictions.
  - It gives insight not only into the errors being made by the classifier but mainly the types of errors that are being made.
  - It is this breakdown that overcomes the limitation of using classification accuracy alone.

Machine Learning Classification Models Evaluation Metrics

- **Accuracy**: the ability of a binary classifier to accurately classify both positives and negatives
- **Precision**: gives the proportion of predicted Positives are truly Positive
- **Sensitivity/recall:** The ability of a binary classifier to detect true positives
- **The Recall** gives the proportion of actual Positives is correctly classified
- **Specificity:** The ability of a binary classifier to detect true negatives
- **F1 score** is a number between 0 and 1 and is the harmonic mean of precision and recall

Metric	
Accuracy	$\text{Accuracy} = \frac{TP + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$
Precision (P)	$\text{Precision} = \frac{TP}{\text{TP} + \text{FP}}$
Sensitivity/Recall(R)	$\text{Sensitivity/recall} = \frac{TP}{\text{TP} + \text{FN}}$
Specificity	$\text{Specificity} = \frac{TN}{\text{TN} + \text{FP}}$
F1	$\text{F1} = \frac{2 * P * R}{\text{P} + \text{R}}$

Machine Learning Classification Models Evaluation Metrics

- Receiver Operating Characteristic (ROC) Curve
  - Are commonly used to characterize the sensitivity/specificity tradeoffs for a binary classifier.
  - Most machine learning classifiers produce real-valued scores that correspond with the strength of the prediction that a given case is positive.
  - Turning these real-valued scores into yes or no predictions requires setting a threshold; cases with scores above the threshold are classified as positive, and cases with scores below the threshold are predicted to be negative.