Weekly Reporting

agrotech live | wigglelabs

2025-04-14

Abstract

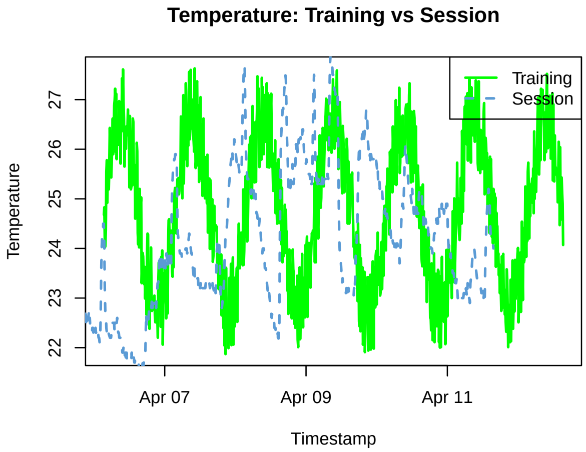
In an ongoing effort to achieve food sovereignty, Wiggle Labs has developed Agrotech Live to monitor the soil health of different plants and crops. This tool collects four data points; Temperature, Moisture, Light and Conductivity from sensors placed near a subject. The training data for this program are the ideal conditions for the subject, and the performance of the experiment is based on how close the collected sensor (testing) data is to the input care training data. Input data is identical in structure the testing data (collected during a session), except it represents only perfect conditions for the subject. A score is generated (along with other statistical results) periodically to communicate how well the experiment is performing.

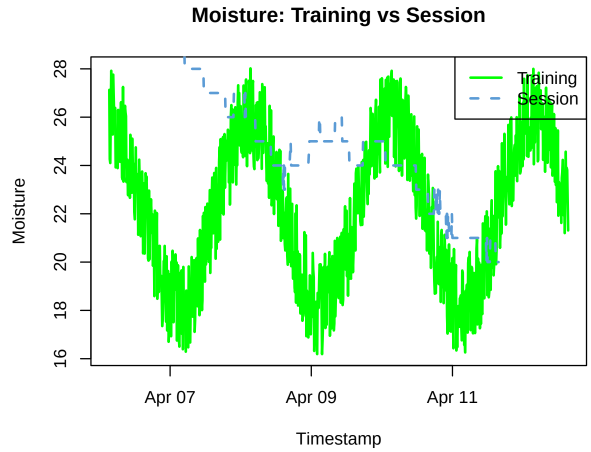
This report covers the past seven days of session data. If there are missing records, then the closest recorded days will be included.

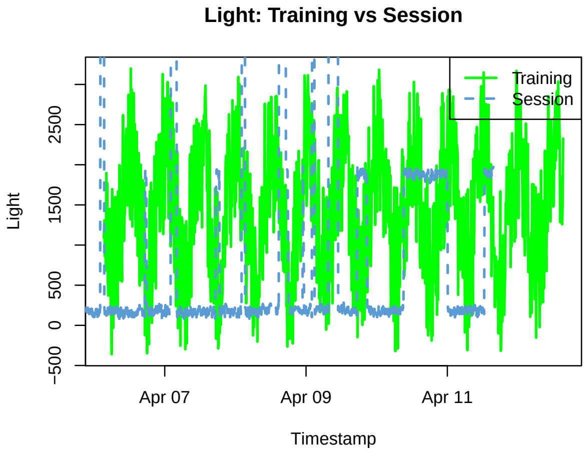
Stage 1: Time Series, Correlation, Covariance and Heatmap

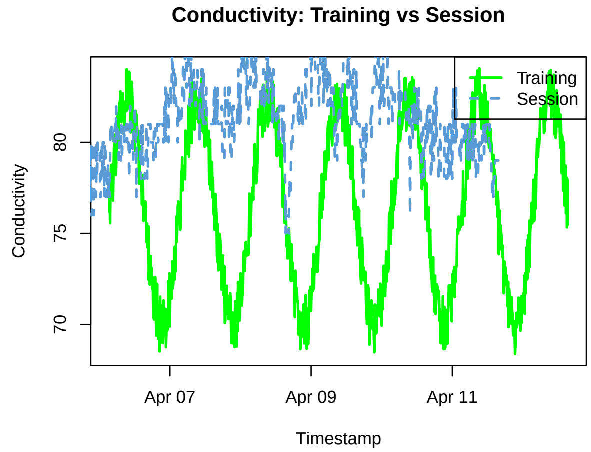
Time Series Comparison

This section examines the relationships between sensor variables from the past week and the initial input data we're comparing the session data to.









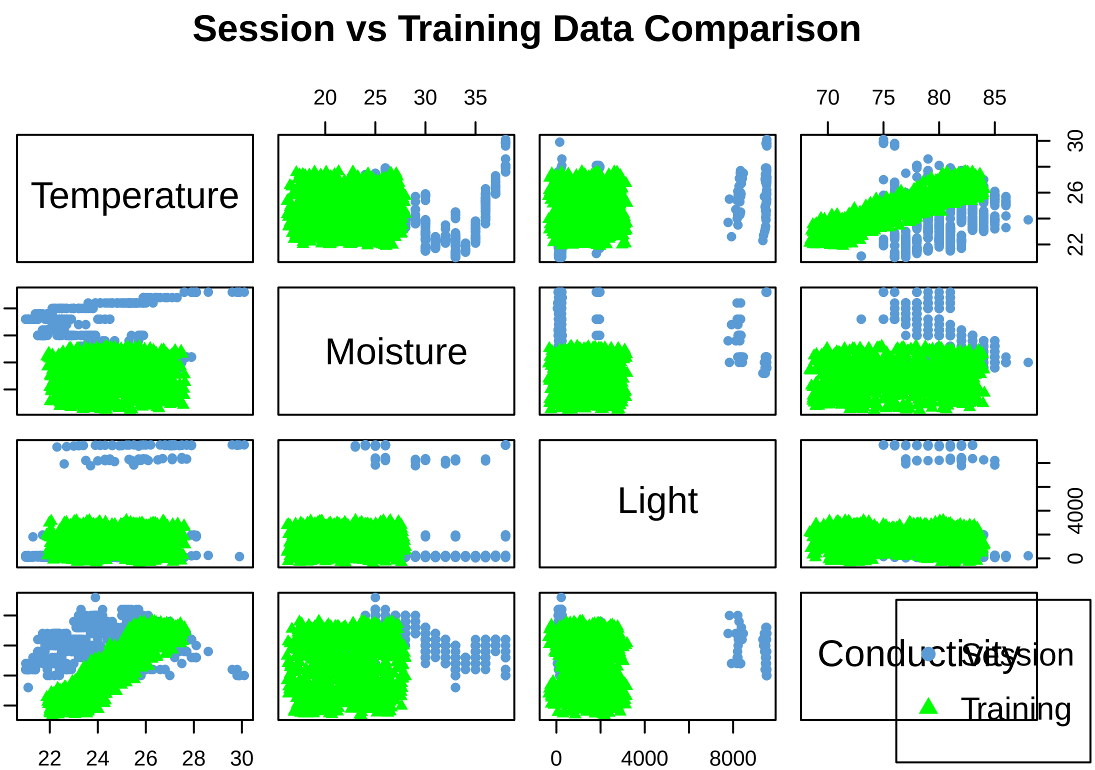
Correlation and Covariance Matrices

In this correlation matrix, score of 1.0 or -1.0 represents a perfect (positive or negative) self-correlation and values closer to 0 show less to no correlation. The matrix above reflects the same relationships as the covariance matrix above.

```
## CROSS-COVARIANCE MATRIX (Session vs Training)

##                Temperature      Moisture        Light Conductivity
## Temperature      0.7666743    0.66692249     35.63236     2.609230
## Moisture         0.4706329   -1.66406751   -314.54838     1.439805
## Light         -254.8970735 1094.39801970 211438.39331  -572.978846
## Conductivity     0.7580297   -0.06816996   -211.56020     2.281201

##
## CROSS-CORRELATION MATRIX (Session vs Training)

##               Temperature     Moisture       Light Conductivity
## Temperature    0.32144663  0.133821283  0.02644675   0.35529199
## Moisture       0.06919658 -0.117091443 -0.08186901   0.06875134
```

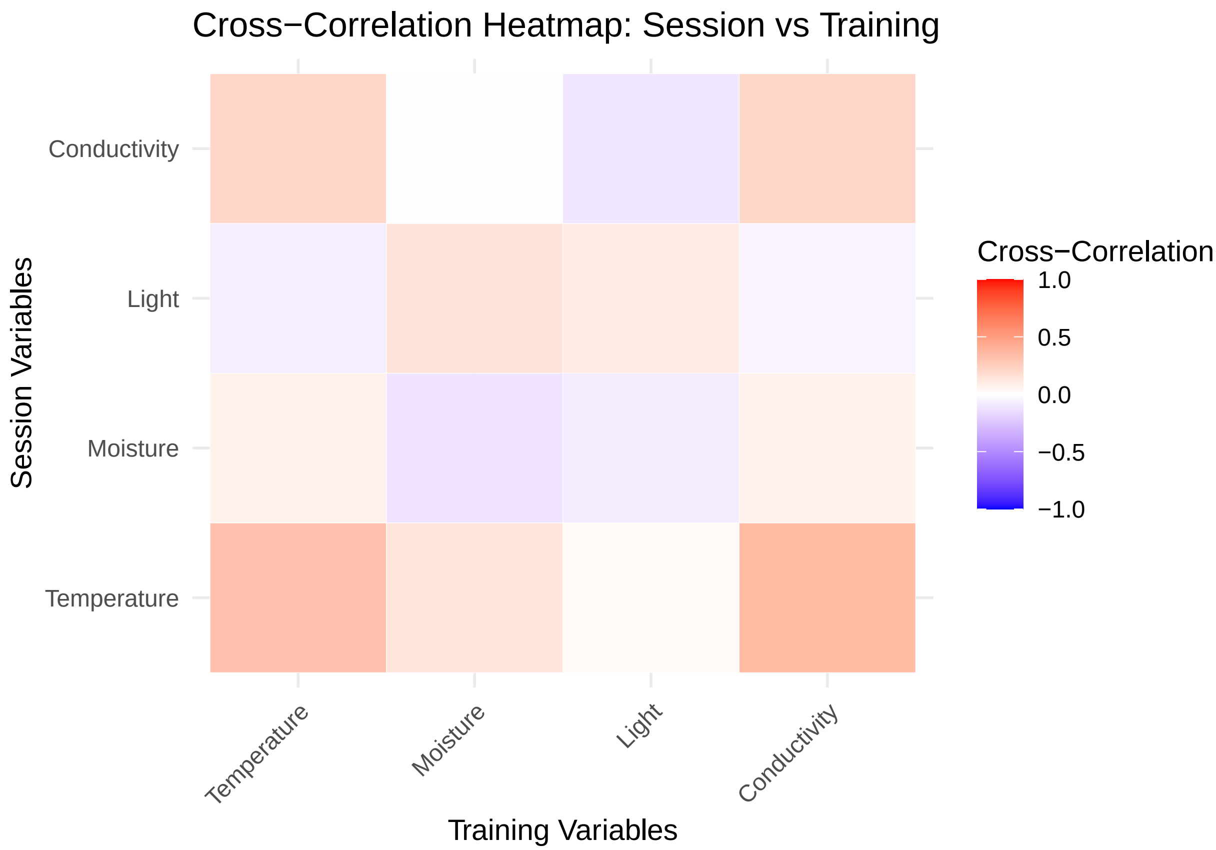
```
## Light          -0.07154050  0.146998962  0.10505112  -0.05222768
## Conductivity   0.21355602 -0.009191164 -0.10550895   0.20872006
```

Plotting variables against each other.



Up until this point (excluding the matrecies) we have been treating the session and training data as independent data sets for comparison purposes. Instead of running similarity algorithms within each variable, both the independent (`session_data`) and the dependent (`training_data`) must be used together in order to produce valuable insights about their relationship.

Cross Correlation Heat Map



Stage 2: Similarities and RSME

In this stage the goal is to measure the similarities and differences between our session and training data. From there, we'll ba able to classify and label the features for K means clustering in Stage 3.

Preprocessing

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:reshape2':
##
##     dcast, melt

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

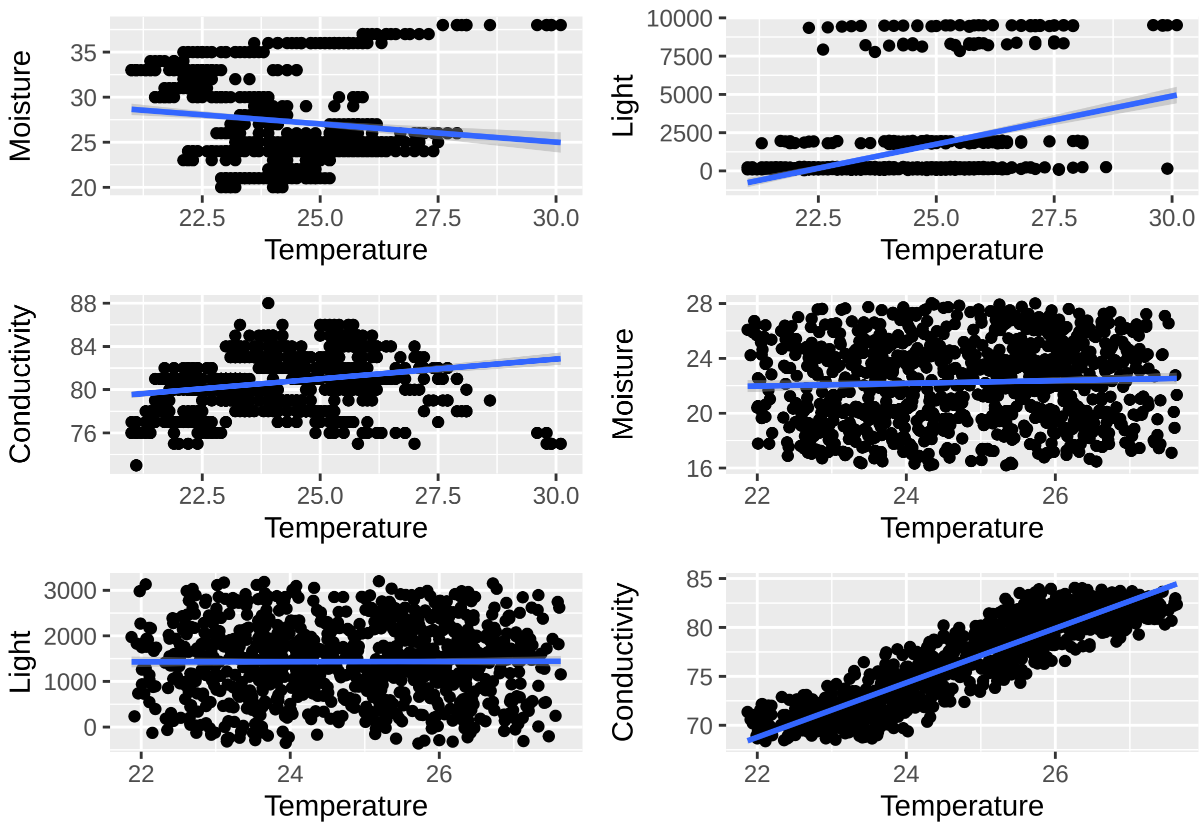
```r
library(dplyr)

training_dt <- as.data.table(training_data)
session_dt <- as.data.table(session_data)
# head()
```

Cleaning the Data

To clean up the data, we'll do some indexing by setting up keys. Then to organize it, we put the fields we want to use for our analysis into their own data frames. Converting sq_ft_lot and sale_price to numeric will ensure no problems when plotting.

Linear Regression Model & Errors

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



This plot is nice, but can only tell us so much about the data. Running a `summary()` will give us more information.

Session Summaries

AGT session data

```
##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer

##
## Temp Sesh ~ Train
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                             Temperature
## -----------------------------------------------
## Temperature                  0.286***
##                               (0.028)
##
## Constant                     17.910***
##                               (0.665)
##
## -----------------------------------------------
## Observations                    937
## R2                             0.103
## Adjusted R2                    0.102
## Residual Std. Error      1.380 (df = 935)
## F Statistic          107.745*** (df = 1; 935)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01

##
## Moisture Sesh ~ Train
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                              Moisture
## -----------------------------------------------
## Moisture                     -0.076***
##                               (0.021)
##
## Constant                     24.341***
##                               (0.588)
##
## -----------------------------------------------
## Observations                    937
## R2                             0.014
## Adjusted R2                    0.013
## Residual Std. Error      3.024 (df = 935)
## F Statistic           12.997*** (df = 1; 935)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

```
##
## Light Sesh ~ Train
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                                Light
## -----------------------------------------------
## Light                         0.035***
##                                (0.011)
##
## Constant                    1,393.937***
##                               (29.681)
##
## -----------------------------------------------
## Observations                     937
## R2                              0.011
## Adjusted R2                     0.010
## Residual Std. Error     818.684 (df = 935)
## F Statistic           10.434*** (df = 1; 935)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

```
##
## Conductivity Sesh ~ Train
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                             Conductivity
## -----------------------------------------------
## Conductivity                  0.384***
##                                (0.059)
##
## Constant                      45.580***
##                                (4.750)
##
## -----------------------------------------------
## Observations                     937
## R2                              0.044
## Adjusted R2                     0.043
## Residual Std. Error      4.388 (df = 935)
## F Statistic           42.588*** (df = 1; 935)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Residual Standard Error

Residuals represent the differences between the actual and predicted values of our dependent (response) variable, sq_ft_lot. A high RSE indicates a weak model for this prediction.

Multiple R

Multiple R, also called the correlation coefficient, measures the strength and direction of a relationship between variables. On a scale of -1 to +1, values closer to -1 or +1 represent perfect negative or positive correlation. 0 means no correlation at all.

Multiple R2 Error

R squared tells us the proportion for variance in sq_ft_lot explained by sale_price, the predictor variable. It can be on a scale of 0 to 1. A value closer to 1 represents greater variance, while closer to 0 represents less variance. Returning a whole 0 or 1 means none or perfect variance respectively.

Adjusted R2 Error

This error is a modified version of the above R-squared error that is able to accommodate for multiple predictors in a regression model.

Stage 3: Modeling, Classification and Metrics

In this next section, we will use the KNN algorithm to find the best K for a label. As this is a placeholder for future use, a good question to use this for is:

How can we classify a subjects health by having thresholds of poor, unsatisfactory, neutral, satisfactory or excellent?

To measure this hypothetical threshold, we would use how closely or different the training data is from testing. This was explored in the previous stage, and is required to be able to have a label to predict for in KNN.

```
## Session Cluster Distribution (%):

##
##        1        2
##  8.32444 91.67556

##
## Training Cluster Distribution (%):

##
##        1        2
## 53.25507 46.74493

##
## Session Cluster Centers (scaled):

##   Temperature     Moisture      Light Conductivity
## 1    1.223590 -0.009060060  3.2021365  -0.50693761
## 2   -0.111106  0.000822683 -0.2907644   0.04603159

##
## Training Cluster Centers (scaled):

##   Temperature    Moisture       Light Conductivity
## 1   0.8020489  0.09565119  0.01053205    0.8220405
## 2  -0.9137498 -0.10897247 -0.01199884   -0.9365256
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

In order to begin preprocessing and find the right k, we will only keep the relevant features and label the cluster. This analysis uses the session data as testing data and the generated ideal conditions as the training data.

We will use two approaches. First with the built in K nearest neighbor functions with R and then manually calculating the accuracy of each K values.

Method 1: Built in knn() Function

```
# Select numeric features and the cluster label
features_train <- training_data[, c("Temperature", "Moisture", "Light", "Conductivity")]
labels_train <- as.factor(training_data$Cluster)

features_test <- session_data[, c("Temperature", "Moisture", "Light", "Conductivity")]
labels_test <- as.factor(session_data$Cluster)
```

Normalizing the data will help scale larger values down to a comparable size.

```
# Normalize (scale) the features
features_train_scaled <- as.data.frame(scale(features_train))
features_test_scaled <- as.data.frame(scale(features_test))
```

For our train/test split,

```
train_features <- features_train_scaled
train_labels <- labels_train
```

Now we'll proceed with modeling KNN and evaluating its performance.

```
library(class)

# Training the KNN model
k_value <- 3  # You can experiment with different values of k
knn_model <- knn(train_features, test_features, train_labels, k = k_value)
test_features
test_labels
train_features
train_labels
# Evaluate the model
confusion_matrix <- table(Predicted = knn_model, Actual = test_labels)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

cat("Accuracy on Session Data (Test Data):", accuracy)
```

Method 2: Manual K Value Accuracy Computation

K Nearest Neighbor (KNN) works by computing the Euclidean distance between the test and training points. Then after selecting the proper k that is the shortest distance, it assigns those closest neighbors most common label to the test point.

Using `knn()` the `class` package automatically computes the Euclidean Distance between two points. We will adjust in the input parameters to ingest the train and test data we have already prepared.

```
# Inputs for manual model
train_data <- features_train_scaled
test_data <- features_test_scaled
train_labels <- as.numeric(labels_train)
test_labels <- as.numeric(labels_test)
```

```
euc_dis <- function(p1, p2) {
  sqrt(sum((p1 - p2)^2))
}
```

In this next section we're implementing the KNN Classifier manually to train the binary classifier data. At this stage, the classifier logic is being defined below.

Here is where we'll compute accuracy for the session data using manual KNN.

```r
k_values <- seq(1, 15, 2)  # or however many k's you want
accuracy_results <- c()

for (k in k_values) {
  predictions <- knn_model(train_data, train_labels, test_data, k)
  acc <- mean(predictions == test_labels)
  accuracy_results <- c(accuracy_results, acc)
  cat("k =", k, ", Accuracy =", round(acc * 100, 2), "%\n")
}
```

The accuracy of this model varies based in the k value used. Plotting them will provide a better idea of which to use to get the best accuracy.

```r
accuracy_data <- data.frame(
  k_values = k_values,
  accuracy = accuracy_results,
  dataset = "Session/Training"
)

ggplot(accuracy_data, aes(x = k_values, y = accuracy)) +
  geom_line(color = "darkgreen") +
  geom_point(size = 3, color = "darkgreen") +
  labs(title = "Manual KNN Accuracy vs. k (Agrotech Session Data)",
       x = "k (Number of Neighbors)", y = "Accuracy") +
  theme_minimal()
```

Stage 4: Forecasting

Stage 5: Discussion