Urban Development Project – Machine Learning Part

2024-05-15

Remember to set directory to folder 'directory'

Libraries setup

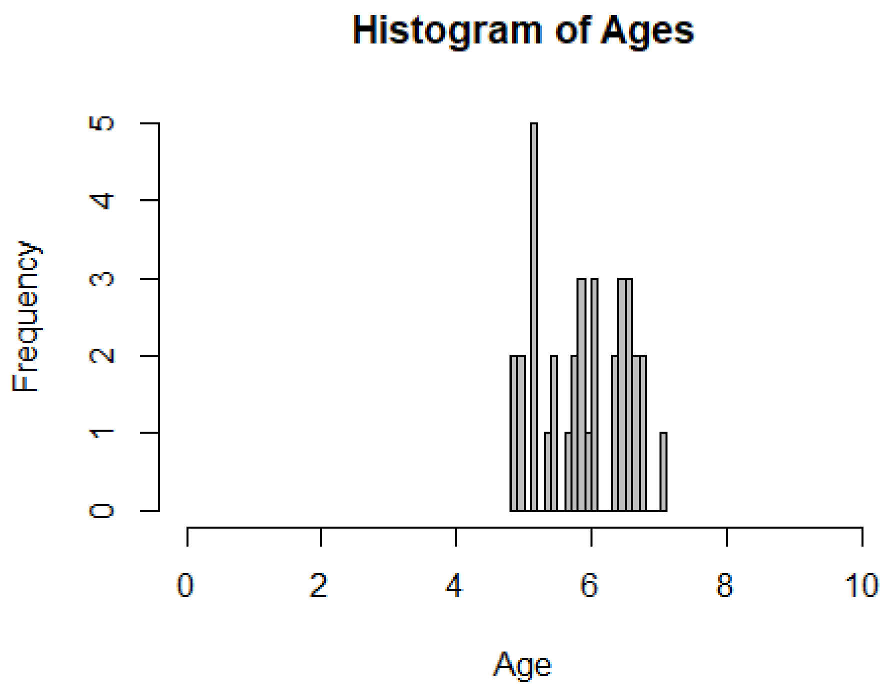
Data engineering - creating meaningful features [Done manually in excel; check file 'denhaagVariables' to see modifications]

Raw datasets - correlation matrices

Filtering for features that correlate strongly with each major indicator

```
# Read correlation matrix with significant correlations
significant_correlations <- read.xlsx("intermediary files/haag2021cor.xlsx",
rowNames = TRUE)

# Keeping only the correlations above the threshold in the new table:
cormatrix_refined
threshold <- 0.2 # Insignificant correlations are 0 thus below threshold

# Filter based on columns
filtered_matrix <-
significant_correlations[abs(significant_correlations$Social_cohesion_21) >=
threshold, ]
# Transpose
transposed_matrix <- t(filtered_matrix)
transposed_matrix <- as.data.frame(transposed_matrix)
# Filter columns again
cormatrix_refined <-
transposed_matrix[abs(transposed_matrix$Social_cohesion_21) >= threshold, ]


# Find the major indicator index
major_indicator_index <- which(colnames(cormatrix_refined) ==
"Social_cohesion_21")
# Create data frame with the features with strong correlations with the major
indicator
refined_features <- as.data.frame(cormatrix_refined[major_indicator_index])
rownames(refined_features) <- rownames(cormatrix_refined)

# Correlation matrix with strong correlations
savepath <- "intermediary files/Social_cohesion_21strongcorrelations.xlsx"
# Variable that correlate strongly with major indicator (dulicate for later
use)
savepath5 <- "intermediary files/Social_cohesion_21strongVariables.xlsx"
```

```
savepath11 <- "intermediary files/Social_cohesion_21Variables.xlsx"

write.xlsx(cormatrix_refined, savepath, rowNames = TRUE)
write.xlsx(refined_features, savepath5, rowNames = TRUE)
write.xlsx(refined_features, savepath11, rowNames = TRUE)


# Distribution of major indicator
hist(data$Social_cohesion_21, breaks = 20, main = "Histogram of Ages", xlab = 
"Age", col = "gray", xlim = c(0, 10))
```




```
# Check dataframe: refined_features
```

Hierarchical clustering on correlation matrices

```
# correlation matrix with variables with strong correlations with major indicator

getcormatrix <- "intermediary files/Social_cohesion_21strongcorrelations.xlsx"

cormatrix_refined <- read.xlsx(getcormatrix, rowNames = TRUE)
```

```
# Making the diagonal values 0
cormatrix_refined[abs(cormatrix_refined) == 1] <- 0
# Compute the distance matrix using Euclidean distance
dist_matrix <- dist(cormatrix_refined, method = "euclidean")
# Perform hierarchical clustering
hc <- hclust(dist_matrix, method = "average")
# Plot the dendrogram
plot(hc, labels = rownames(cormatrix_refined), main = "Hierarchical Clustering
Dendrogram", sub = "", xlab = "")
```

```
# Get the heights at which merges occur to choose number of clusters
merge_heights <- hc$height
# Plot the heights to find the "elbow" to choose number of clusters
plot(merge_heights, type = 'b', xlab = "Number of merges", ylab = "Merge height",
  main = "Elbow Plot")
```

```
# CHOOSE NUMBER OF CLUSTERS
k <- 20
# CHOOSE NUMBER OF CLUSTERS
clusters <- cutree(hc, k = k)
# Create data frame with clusters
clusters <- data.frame(row.names = row.names(cormatrix_refined), cluster = clusters)
# Descending order
clusters <- clusters %>% arrange(desc(cluster))
```

```
# Check data frame clusters


# GIVE TARGET VARIABLE
target_variable <- cormatrix_refined$Social_cohesion_21
# GIVE TARGET VARIABLE



# Create a data frame with variables, cluster numbers, and correlation coefficients with major indicator
# Add the correlation coefficient with the target variable, as a vector, to the clusters data frame
correlation <- as.numeric(vector())
for (i in 1:nrow(cormatrix_refined)) { # Correlation matrix rows
 for (z in 1:nrow(clusters)) { # Clusters data frame rows
  if (row.names(cormatrix_refined)[i] == row.names(clusters)[z]) { # Find matches and pass the coefficient
   correlation[z] <- round(target_variable[i], 2)
  }
 }
}
clusters$correlation <- correlation


# Store clusters data frame
storepath <- "intermediary files/Social_cohesion_21Clusters2.xlsx"
write.xlsx(clusters, storepath, rowNames = TRUE)


# Check data frame: clusters
```

Extracting vif values from clusters

```
# Read clusters, strong correlations matrix and full data
variables <- "intermediary files/Social_cohesion_21Clusters.xlsx"
correlations_matrix <- "intermediary
files/Social_cohesion_21strongcorrelations.xlsx"
citydata <- "source data/source_data_hague.xlsx"

features <- read.xlsx(variables, colNames = TRUE)
correlationsfull <- read.xlsx(correlations_matrix, rowNames = TRUE)
datafull <- read.xlsx(citydata, rowNames = TRUE)

# Refine full data to only for the selected variables that correlate strongly
with the major indicator
column_indices <- match(features[,1], names(datafull))
data <- datafull[, column_indices]


# CHOOSE MAJOR INDICATOR
target_variable <- "Social_cohesion_21"
# CHOOSE MAJOR INDICATOR

# Initialize list
vif_values <- list()

# Get number of clusters
number_of_clusters_in_dataset <- max(features$cluster)

# Iterate over clusters to split the data, run regressions, and get the vifs
for (cluster_number in 1:number_of_clusters_in_dataset) {

  # Find the right indeces to split data by cluster and add the major
indicator, if it is not already included in the cluster
  cluster_indices <- which(features$cluster == cluster_number)
  target_variable_index <- which(features[,1] == target_variable)
  logical <- 0
  # Flag will become 1 for the one cluster that included the major indicators
so that we won't add it again
  for (index in 1:length(cluster_indices)) {
    if (cluster_indices[index] == target_variable_index) {
      logical <- logical + 1
    }
  }

  # If flag = 0 then add the index of the major indicator
  if (logical == 1) {
    data_indeces <- cluster_indices
  } else {
    data_indeces <- c(cluster_indices, target_variable_index)
  }
```

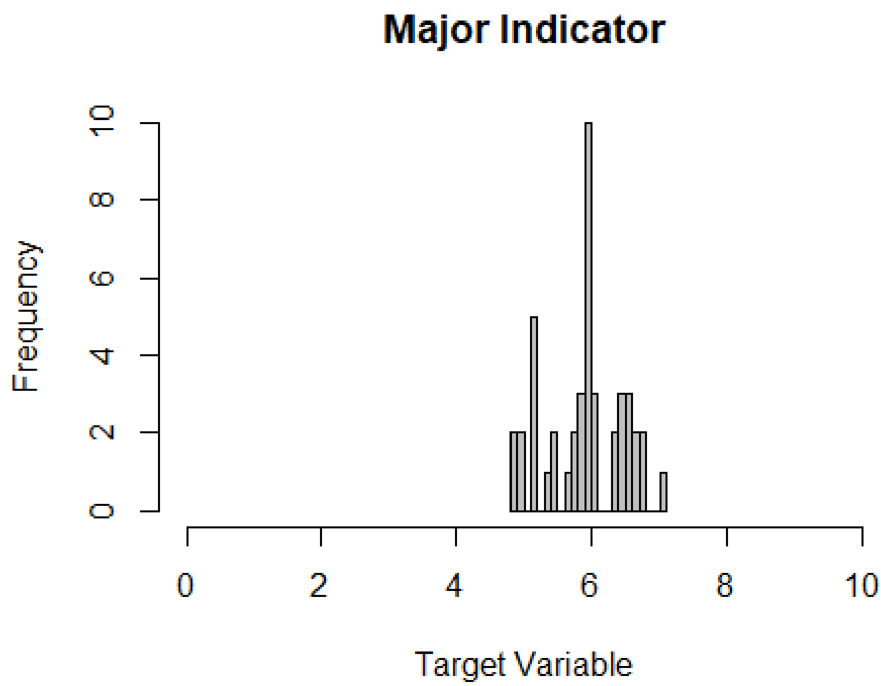
```
  data_cluster <- data[, data_indeces]
  # data_cluster is the data for the variables in the current cluster and the
target variable.


  # regression will only work with at least 1 indipendent variable and the
major indicator
  if (length(data_indeces) > 2) {
    # Fit a linear regression model
    lm_model <- lm(data_cluster$Social_cohesion_21 ~., data = data_cluster)

    # Calculate VIF
    vif_values[[cluster_number]] <- vif(lm_model)

  } else { # if cluster has only 1 variable
    vif_values[[cluster_number]] <- "Too small cluster, buddy!"
  }
}

# Initialize aggregate vif data frame for all clusters/variables
vif_values_df <- data.frame()

for (element in 1:number_of_clusters_in_dataset) { # iterate over clusters
  # Convert current vif list to a data frame
  vif_values_current <- data.frame(vif_values[[element]])
  vif_values_current$cluster <- element
  # column names
  colnames(vif_values_current) <- c("vif_value", "cluster")


  # add current vifs to the data frame with the previous ones with every
iteration
  vif_values_df <- rbind(vif_values_df, vif_values_current)

}
# Make them pretty
vif_values_df$vif_value <- as.numeric(vif_values_df$vif_value)

## Warning: NAs introduced by coercion

vif_values_df$vif_value <- round(vif_values_df$vif_value, 1)
# Sort out NAs
vif_values_df <- vif_values_df[!is.na(vif_values_df$vif_value), ]

# Save vifs
savepath <- "intermediary files/Social_cohesion_21VIF.xlsx"
write.xlsx(vif_values_df, savepath, rowNames = TRUE)
```

```
# Check dataframe: vif_values_df
```

Handling Missing Values

```
# Get raw data
excelwithallvariables <- "source data/source_data_hague.xlsx"
data <-read.xlsx(excelwithallvariables, rowNames = TRUE)


# Ensure columns are numeric
for (i in 1:ncol(data)) {
  data[, i] <- as.numeric(data[, i])
}

# Create a function that checks if a column contains integers
are_all_integers <- function(x) {
  all(x == floor(x), na.rm = TRUE)
}
# The NAs must be replaced with integers for the features that contain only
integer values. This ensure realistic replacements


# Create outout data frame
data_clean <- data

# If skewness in a column is < 0.5 then it replaces NAs with the mean of the
column. If the variable contains integers the it rounds the mean to be an
integer.
# If skewness in a column is >= 0.5 and < 1 then it replaces NAs with the
mean of the k-NN (k is set to 5) of the column. If the variable contains
integers the it rounds the k-NN mean to be an integer.
# If skewness in a column is > 1 then it replaces NAs with the median of the
column. If the variable contains integers the it rounds the median to be an
integer.
# When a column has values that are all the same 1. it souldn't, 2. instead
of NA the skewness is treated like it's 0.

# Search each column for it's skewness, and then for being an integer
variable
for (i in 1:ncol(data)) {
  # To avoid errors for variables with 0 skewness fill in 0 manually
  skewness_current <- skewness(data[,i], na.rm = TRUE)
  if (is.na(skewness_current) == TRUE) {
    skewness_current <- 0
  }
  if (abs(skewness_current) < 0.5) { # Low skewness
    if (are_all_integers(data[,i]) == TRUE) { # Integer values
      # Replace NAs with integer mean
      data_clean[,i][is.na(data[,i])] <- round(mean(data[,i]))
```

```
    } else {
      # Replace NAs with mean
      data_clean[,i][is.na(data[,i])] <- mean(data[,i], na.rm = TRUE)
    }

  } else if (abs(skewness_current) < 1) { # Medium skewness
    z <- as.numeric(i) # corret variable index type to avoid error
    if (are_all_integers(data[,i]) == TRUE) {
      # Replace NAs with integer 5-NN mean
      data_clean[,i] <- round(kNN(data, variable = z, k = 5)[,i])
    } else {
      # Replace NAs with 5-NN mean
      data_clean[,i] <- kNN(data, variable = z, k = 5)[,i]
    }
  } else {
    if (are_all_integers(data[,i]) == TRUE) { # High skewness
      # Replace NAs with integer medium
      data_clean[,i][is.na(data[,i])] <- round(median(data[,i], na.rm =
TRUE))
    } else {
      # Replace NAs with medium
      data_clean[,i][is.na(data[,i])] <- median(data[,i], na.rm = TRUE)
    }
  }
}


# Save clean dataset
savepath <- "intermediary files/haag2021clean.xlsx"
write.xlsx(data_clean, savepath, rowNames = TRUE)


# Check statistics and distribution of target variable to decide what model
to use.
data_clean <-read.xlsx(savepath, rowNames = TRUE)

# Summary statistics for numerical data
summary(data_clean$Social_cohesion_21)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.800   5.500   5.931   5.931   6.425   7.100

# Histogram target variable
hist(data_clean$Social_cohesion_21, breaks = 20, main = "Major Indicator",
xlab = "Target Variable", col = "gray", xlim = c(0, 10)) # Adjust xlim with
both scale edges
```




```
# Skewness of target variable
skewness(data_clean$Social_cohesion_21, na.rm = TRUE)

## [1] -0.1130332

# Compare dataframes: data_clean , datafull
```

THE HAGUE - FEATURE SELECTION 1/2

```
# FOR THE FIRST RUN NUMBER SHOULD BE EQUIAL TO LENGTH OF FEATURES TO INCLUDE
ALL VARIABLES. Then you may lower the number to take the top informative
features and rerun the code chunk.
number_of_variables <- 155




# Read variables that correlate strongly with major indicator and read full
data
variables <- "intermediary files/Social_cohesion_21strongVariables.xlsx"
citydata <- "intermediary files/haag2021clean.xlsx"
features <- read.xlsx(variables, colNames = TRUE)
colnames(features)[1] <- "variable"
datafull <- read.xlsx(citydata, rowNames = TRUE)
# Refine full data only for the selected variables
column_indices <- match(features$variable, names(datafull))
data_fixed <- datafull[, column_indices]
length(column_indices) # To help decide about the right number of variables
```

```
## [1] 155
```

```
# Subset data
subset_indexes <- c(1:number_of_variables)
data_fixed <- data_fixed[, subset_indexes]
```

```
# RANDOM FOREST
# Initialize metric vector for LOOCV. Their means will be the final metric.
predictionsrf <- vector("numeric", length = nrow(data_fixed))
accuraciesrf <- vector("numeric", length = nrow(data_fixed))
mserf <- vector("numeric", length = nrow(data_fixed))
maerf <- vector("numeric", length = nrow(data_fixed))
maperf <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  # Create the training set by excluding the ith observation
  train_data <- data_fixed[-i, ]

  # Create the test set with only the ith observation
  test_data <- data_fixed[i, ]

  # Fit a Random Forest model
  rffeatures <- randomForest(train_data$Social_cohesion_21 ~ ., data =
train_data, ntree = 500, mtry = 5, importance = TRUE)

  if (i == 1) { # Initialize importance matrix
    importance_matrix <- as.numeric(matrix(NA, nrow = ncol(data_fixed) - 1,
ncol = 1))
    # Store feature importance
    importance_matrix <- importance(rffeatures)[,1]
  } else {
    # Store recurrent feature importances
    importance_matrix <- cbind(importance_matrix, importance(rffeatures)[,1])
  }

  # Make predictions on train dataset
  predictionsrf[i] <- predict(rffeatures, test_data, type = "response")

  # MSE
  mserf[i] <- (predictionsrf[i] - test_data$Social_cohesion_21)^2

  # MSE
  maerf[i] <- MAE(predictionsrf[i], test_data$Social_cohesion_21)
```

```
  # MAPE
  maperf[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsrf[i]) /
test_data$Social_cohesion_21) * 100)

  # Accuracy on train set
  accuraciesrf[i] <- Accuracy(round(predictionsrf[i],1),
round(test_data$Social_cohesion_21, 1))
}

# Handle Metrics - take their means
importances <- apply(importance_matrix, 1, mean)
importances <- data.frame(importance = importances)
importances <- importances %>% arrange(desc(importance))
accuracyrf <- mean(accuraciesrf)
mserf <- mean(mserf)
maerf <- mean(maerf)
maperf <- mean(maperf)
# Calculate Rsquared values
rsquaredrf <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsrf)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredrfAdj <- 1 - ((1 - rsquaredrf) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))


# Show metrics
importances
```

```
##
importance
## I_feel_at_home_with_the_people_percent_21
5.703649574
## people_interact_in_a_pleasant_manner_percent_21
5.359478134
## safety_score_21
5.358591881
## Pleasant_living_score_21
5.358510263
## I_live_in_a_nice_neghborhood_where_people_help_each_other_21
5.269239686
## turkish_21
4.952659747
## people_hardly_know_each_other_percent_21
4.644330271
## percentage_of_sports_associations_member_ship_17
4.416033119
## Dutch_21
4.285030870
## nuisance_from_local_residents_percent_21
4.263876664
## satisfied_with_municipality_for_quality_of_life_and_safety_21
```

```
4.033456969
## non_working_job_seekers_total_17
3.870745857
## antillians_21
3.816494962
## surinamese_21
3.706860549
## moroccan_21
3.657158726
## children_in_childcare_19
3.632363062
## odor_nuisance_percent_21
3.526548605
## percentage_high_risk_for_anxiety_disorder_or_depression_20
3.517963871
## percentage_have_experienced_a_lot_of_stress_in_past_four_weeks_20
3.501793032
## percentage_who_do_volunteer_work_20
3.443589448
## drig_trafficking_percent_21
3.439070878
## with_partner_persons_21
3.378910016
## children_receiving_out_of_school_care_in_a_child_center_19
3.323110836
## percentage_who_feel_seriously_lonely_20
3.222944379
## average_valie_of_homes_in_general_21
3.092564001
## dissatisfied_with_municipality_for_quality_of_life_and_safety_21
3.029094209
## rubbish_on_street_21
3.020214669
## satisfied_with_maintencance_public_gardens_and_parks_percent_21
2.933287480
## satisfied_with_maintenance_of_sidewalks_streets_and_squares_percent_21
2.847881862
## percentage_who_have_difficulty_making_ends_meet_20
2.839693064
## MuseumWithin10Km_97
2.806026165
## average_SES_WOA_partial_score_of_financial_educational_level_19
2.792431202
## average_total_social_score_economic_status_SES_WOA_2019
2.698241822
## confused_persons_percent_21
2.687993859
## percentage_who_provide_informal_care_20
2.671939866
## social_nuisance_percent_21
```

```
2.583726920
## average_calue_of_apartment_homes_21
2.482540514
## average_value_of_single_family_homes_21
2.470030584
## persentage_smoke_20
2.435976565
## non_westerns_21
2.434924319
## percentage_drink_no_alcoho_or_one_glass_per_day_20
2.419254511
## AttractionsWithin20Km_112
2.337731928
## percentage_rental_properties_21
2.272913993
## average_personal_yearly_income_individuals_in_euros_21
2.207926403
## drug_use_percent_21
2.200118153
## environmental_nuisance_21
2.169288718
## average_age_population_21
2.125164726
## niusance_from_harassing_people_on_te_street_percent_21
2.106993807
## education_level_low_21
2.029739358
## score_physical_quality_of_living_environment_percent_21
2.020034961
## percentage_overweight_20
1.964111313
## average_disposable_part_household_income_21
1.946038140
## victimization_of_property_crimes_percent_21
1.912931469
## males_21
1.846781426
## females_21
1.828667902
## education_level_high_21
1.820464759
## perventage_high_income_households_21
1.738423019
## victimization_total_percent_21
1.736872992
## children_in_out_of_school_care_with_childminder_19
1.708182128
## daubed_walls_or_buildings_percent_21
1.704409507
## aggressive_driving_behavior_pecent_21
```

```
1.678575564
## I_have_a_lot_of_contact_with_locl_residents_percent_21
1.678041130
## destroyed_street_furniture_percent_21
1.657250205
## AverageSES_WOA_score_subscore_of_educational_level_19
1.647752746
## living_together_without_children_households_21
1.632449821
## happiness_index_18
1.629879960
## Average_SES_WOAscore_partial_score_of_employment_history_19
1.560079175
## age_65_years_or_older_persons_21
1.554782480
## noise_pollution_percent_21
1.547519098
## child_place_in_daycare_21
1.538447631
## children_in_day_care_in_child_center_19
1.532659453
## single_parent_family_21
1.508076309
## drunk_people_on_street_percent_21
1.500849500
## neisance_total_21
1.481412059
## percentage_using_care_total_health_insurance_act_18_to_64_years_21
1.403375624
## percentage_people_with_pgysical_disabilities_20
1.397096152
## very_satisfied_with_street_lighting_in_neighborhood_percent_21
1.370518267
## percentage_people_with_good_or_very_good_general_health_20
1.321754313
## energy_label_a_and_higher_21
1.293209521
## age_20_to_64_persons_21
1.255596828
## gray_pressure_percentage_close_to_age_64_21
1.248189389
## percentage_low_income_households_21
1.238411661
## crimes_total_21
1.219968407
## waste_notifications_21
1.199454624
## victimization_of_proerty_crimes_home_burglary_percent_21
1.179286021
## parking_problems_percent_21
```

```
1.158419158
## unmarried_21
1.156347547
## households_21
1.122528260
## HotelEtcWithin5Km_49
1.054149164
## nuisance_vagrants_21
1.051102027
## percentage_using_care_total_health_insurance_act_21
1.027312709
## nuisance_caused_by_young_people_hanging_around_21
1.023547537
## childcare_centers_21
1.001291204
## married_21
0.977078892
## very_satisfied_with_playgrounds_for_chldren_percent_21
0.973351277
## percentage_who_exercise_at_least_once_a_week_20
0.965276320
## noise_pollution_21
0.947038938
## nuisance_related_to_alcohol_drugs_21
0.938679164
## Housing_density_houses_per_hectare_21
0.929183466
## DepartmentStoreWithin5Km_33
0.928184014
## household_density_21
0.871279461
## residential_funtion_homes_21
0.850957123
## HospitalsInclWithin5Km_12
0.846469694
## LargeSupermarketWithin1Km_25
0.844515606
## gross_population_density_21
0.768083059
## youth_nuissance_21
0.761086764
## stock_of_homes_21
0.751347848
## DistanceToTrainStationAllTypes_90
0.732965250
## AttractionsWithin10Km_111
0.722139723
## victimization_of_ciolent_crimes_percent_21
0.697494697
## Havo_wvoWithin3Km_73
```

```
0.694665440
## DistanceToImportantTransferStation_91
0.672941718
## physical_deterioration_percent_21
0.669614429
## traffic_nuisance_percent_21
0.630220517
## percentage_who_meet_the_exercise_guideline_20
0.627539979
## PrimarySchoolWithin1Km_61
0.626983076
## westerns_21
0.624491457
## DistanceToMainRoadEntrance_89
0.616973203
## nuisance_caused_by_cofused_person_21
0.598292705
## average_home_occupancy_21
0.563896927
## notifications_of_animals_and_dog_feces_21
0.540471148
## DistanceToAttraction_110
0.533521067
## child_places_in_out_of_school_child_care_21
0.524666162
## education_level_secondary_21
0.496737477
## speeding_occurs_percent_21
0.457258119
## out_of_school_child_care_centers_21
0.430769226
## nuisance_from_catering_establishments_percent_21
0.397573444
## apartments_percentage_21
0.397404939
## DistanceToHavo_vwoSchool_72
0.375709616
## DistanceToSauna_108
0.351130402
## CafeEtcWithin1Km_37
0.348011076
## average_private_cars_per_adress_21
0.346208760
## distance_to_VMBO_school_km_13
0.322438397
## primary_schools_21
0.318458178
## residential_funtion_office_21
0.305540919
## DistanceToCafeEtc_36
```

```
0.284928458
## homes_for_single_families_percentage_21
0.277057838
## public_drunkenness_21
0.220258557
## DistanceToGPPost_9
0.207040318
## distance_to_HAVO_VMO_school_in_km_13
0.180277492
## notifications_for_companies_or_events_21
0.147274252
## DistanceToShopForOtherDailyFood_28
0.145206174
## notifications_streets_and_street_furniture_21
0.116938680
## secondary_schools_21
0.115907674
## divorced_21                                                              -
0.008521957
## single_person_households_21                                              -
0.017355529
## DistanceToHospitalInclOutpatientClinics_11                               -
0.035968664
## accomodation_companies_21                                                -
0.062487506
## points_of_sale_stores_21                                                 -
0.081853400
## neisance_general_occurs_percent_21                                       -
0.087444233
## DistanceToCafeteriaEtc_40                                                -
0.296689666
## DistanceToLargeSupermarket_24                                            -
0.319104653
## DistanceToFireStation_114                                                -
0.399844156
## percentage_of_significantly_wormer_area_in_the_neighborhood_21          -
0.418959085
```

```
print(paste(round(accuracyrf, 2), "Accuracy RF"))
```

```
## [1] "0.2 Accuracy RF"
```

```
print(paste(round(mserf, 5), "MSE RF"))
```

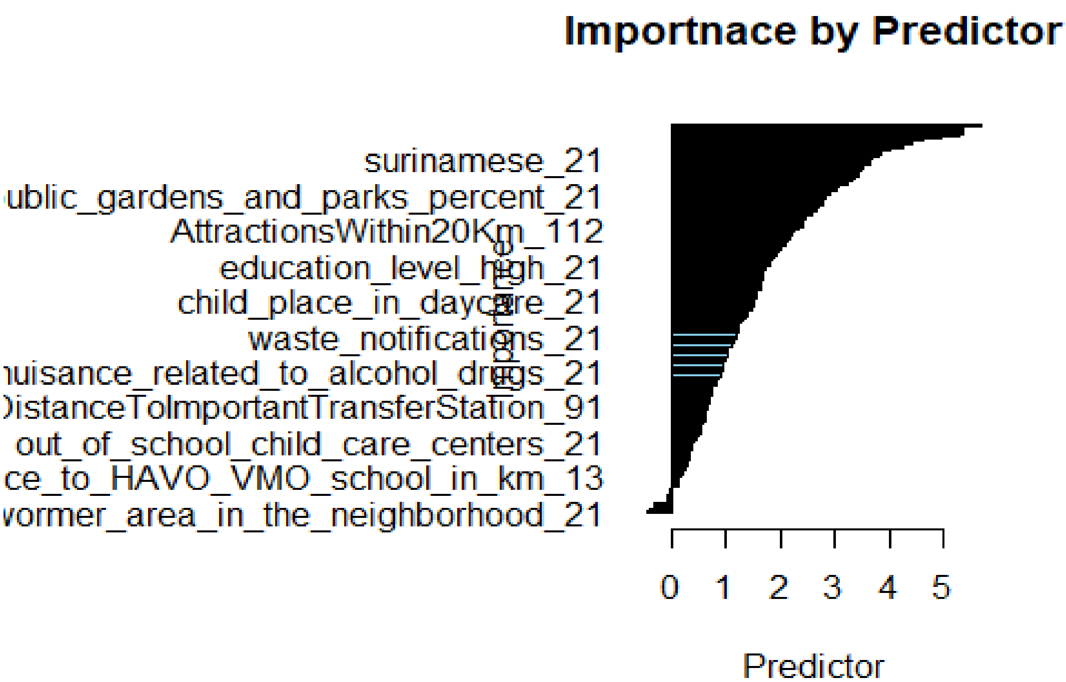
```
## [1] "0.06377 MSE RF"
```

```
print(paste(round(maerf, 5), "MAE RF"))
```

```
## [1] "0.1925 MAE RF"
```

```
print(paste(round(maperf, 5), "MAPE RF"))
```

```
## [1] "3.30369 MAPE RF"

print(paste(round(rsquaredrf, 3), "R^2 RF"))

## [1] "0.805 R^2 RF"

print(paste(round(rsquaredrfAdj, 3), "R^2 Adjusted RF"))

## [1] "1.075 R^2 Adjusted RF"

# Plot importance
par(mar=c(5, 15, 4, 2) + 0.1)
importances_reversed <- importances %>% arrange(importance)
barplot(importances_reversed$importance, names.arg =
rownames(importances_reversed), main = "Importnace by Predictor", xlab =
"Predictor", ylab = "Importance", horiz = TRUE, col = "skyblue", las = 1,
cex.names=1.0)
```



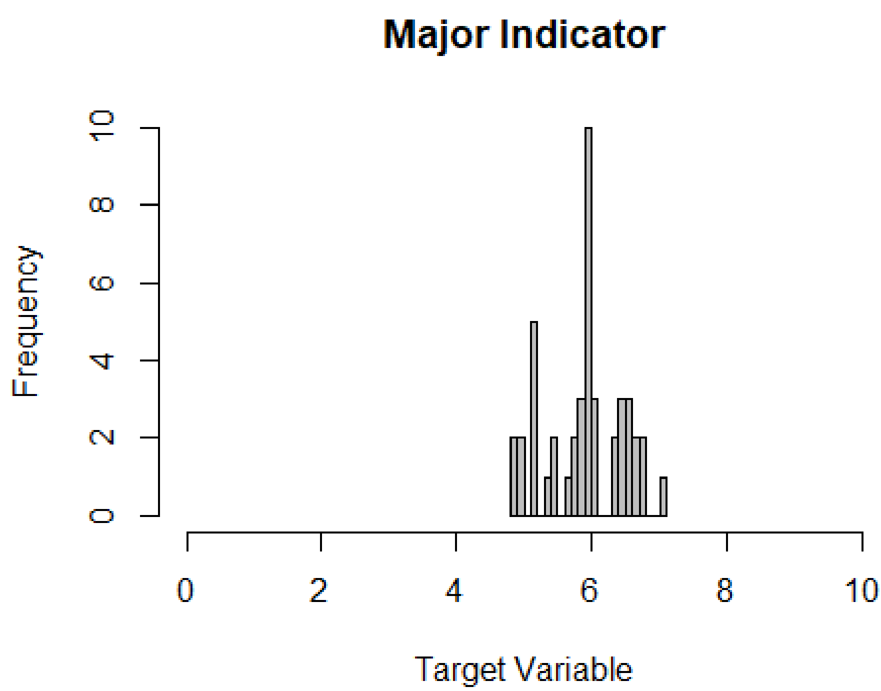

```
# Metrics data frame
rfmetrics <- data.frame(
  MAE = c(round(maerf, 3)),
  MSE = c(round(mserf, 5)),
  MAPE = c(round(maperf, 5)),
  R_squared = c(round(rsquaredrf, 3)),
  R_squared_adjusted = c(round(rsquaredrfAdj, 3)),
  row.names = c("RF metrics")
)
```

```
# Subset data - take the features that you used in the model. When you rerun
the code you will gradually lower the number of features and keep the top
picks based on the results
column_refinement <- match(rownames(importances), names(datafull))
data_fixed <- datafull[, column_refinement]
# Add target variable to the set
data_fixed <- cbind(datafull$Social_cohesion_21, datafull[,
column_refinement])
colnames(data_fixed)[1] <- "Social_cohesion_21"

# Store the subset variables to be used again
variables <- as.data.frame(c("Variables based on random forest importances",
colnames(datafull)[column_refinement]), col.names = "variable")
savepath3 <- "Social_cohesion_21Variables.xlsx"
# Store importances
savepath12 <- "Social_cohesion_21Importances.xlsx"

write.xlsx(variables, savepath3, rowNames = FALSE, colNames = FALSE)
write.xlsx(importances, savepath12, rowNames = TRUE, colNames = TRUE)


# Check dataframes: importances, rfmetrics
# You can now iterate with less features
```

THE HAGUE - Using RFE to add additional meaningful (urban) features in our feature selection

```
# Set import and export paths
file <- "intermediary files/haag2021clean.xlsx"
saveselectedvars <- "intermediary files/Social_cohesion_21RFEvariables.xlsx"
saveimportances <- "intermediary files/Social_cohesion_21RFEimportances.xlsx"

# CHOOSE TARGET VARIABLE
target_variable <- "Social_cohesion_21"


# Den Haag major indicators list; it will be excluded from the rfe formula
major_indicators <-
c("percentage_people_with_good_or_very_good_general_health_20",
"percentage_who_feel_seriously_lonely_20", "Pleasant_living_score_21",
"safety_score_21", "Social_cohesion_21")


# Import the dataset
data <- read.xlsx(file, rowNames = TRUE, colNames = TRUE)

# Separate target indicator. Create bins to run RFE
target_index <- match(target_variable, names(data))
indicator <- data[, target_index]
```

```r
# Check distribution before splitting in bins just in case something is
irregular
hist(indicator, breaks = 20, main = "Major Indicator", xlab = "Target
Variable", col = "gray", xlim = c(0, 10)) # Adjust xlim with both scale edges
```




```r
# Creating bins
quantile_breaks <- quantile(indicator, probs = seq(0, 1, by = 0.2))

# Cutting the data into these quantiles
indicator_categories <- cut(indicator, breaks = quantile_breaks,
include.lowest = TRUE, labels = FALSE)



data <- data %>%
  # Save categorical features as factors
  mutate_at(colnames(data),
            as.factor) %>%
  # Center and scale numeric features
  mutate_if(is.numeric, scale)


# Define the control using a random forest selection function
control <- rfeControl(functions = rfFuncs, # random forest
                      method = "repeatedcv", # repeated cv
```

```r
                      repeats = 5, # number of repeats
                      number = 10) # number of folds


# Features without major indicators. They will not be used in the models.
column_indices <- match(major_indicators, names(data))
x <- as.data.frame(data[, -column_indices])
# Target variable bins as target variable
y <- indicator_categories

# Training: 80%; Test: 20%
set.seed(2021)
inTrain <- createDataPartition(y, p = .70, list = FALSE)

x_train <- x[ inTrain, ]
x_test  <- x[-inTrain, ]

y_train <- y[ inTrain]
y_test  <- y[-inTrain]

# Run RFE
result_rfe1 <- rfe(x = x_train,
                   y = y_train,
                   sizes = c(1:15),
                   rfeControl = control)

# Print the results
result_rfe1

##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
##
## Resampling performance over subset size:
##
##  Variables  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD Selected
##          1 1.440   0.4525 1.245 0.3905     0.3613 0.4266
##          2 1.343   0.4851 1.179 0.3476     0.3496 0.3967
##          3 1.347   0.4722 1.185 0.3245     0.3519 0.3700
##          4 1.330   0.5091 1.168 0.3327     0.3583 0.3805        *
##          5 1.333   0.5278 1.169 0.3282     0.3416 0.3767
##          6 1.345   0.4744 1.180 0.3329     0.3303 0.3776
##          7 1.337   0.4809 1.173 0.3342     0.3345 0.3796
##          8 1.339   0.4218 1.177 0.3309     0.3193 0.3774
##          9 1.340   0.4318 1.179 0.3306     0.3334 0.3759
##         10 1.338   0.4431 1.178 0.3302     0.3021 0.3745
##         11 1.336   0.4358 1.175 0.3318     0.3134 0.3742
##         12 1.339   0.4408 1.176 0.3259     0.3392 0.3736
```

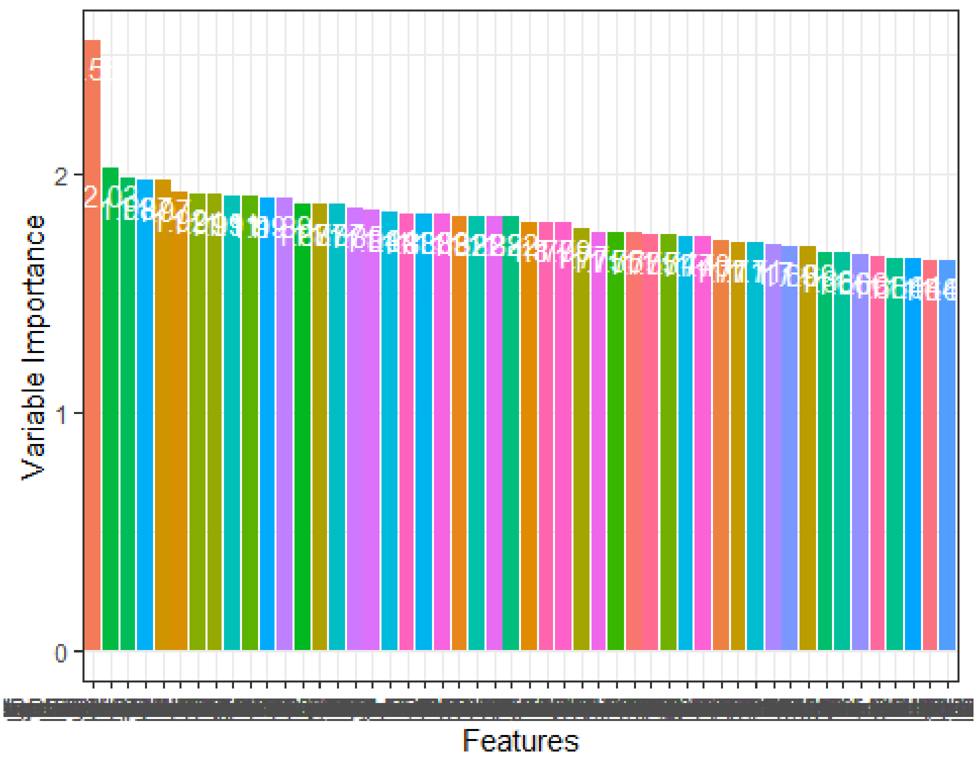
```
##         13 1.337   0.4889 1.174 0.3273     0.3305 0.3731
##         14 1.338   0.4893 1.174 0.3279     0.3411 0.3749
##         15 1.342   0.4663 1.178 0.3339     0.3492 0.3801
##        230 1.341   0.5044 1.168 0.3533     0.3967 0.4112
##
## The top 4 variables (out of 4):
##    percentage_who_provide_informal_care_20, points_of_sale_stores_21,
education_level_high_21, Havo_wvoWithin3Km_73
```

```
# Print the selected features
selected_predictors <- as.data.frame(predictors(result_rfe1))
selected_predictors[nrow(selected_predictors)+1,] <- target_variable


# Get top 50 informative feature importances. We won't need a lot of features
for our dataset size
varimp_data <- data.frame(feature = row.names(varImp(result_rfe1))[1:50],
                          importance = varImp(result_rfe1)[1:50, 1])
# Check top 50 importances
varimp_data
```

```
##                                                               feature
importance
## 1                                         AttractionsWithin20Km_112
2.556040
## 2                                            education_level_low_21
2.025258
## 3                                             GPpracticeWithin1Km_6
1.975988
## 4                                 percentage_area_of_rain_over_10cm_21
1.969365
## 5                                          DaycareCentresWithin1Km_53
1.967991
## 6                                         confused_persons_percent_21
1.918291
## 7                          DistanceToHospitalInclOutpatientClinics_11
1.910157
## 8                          DistanceToHospitalExclOutpatientClinics_15
1.908424
## 9                                         LargeSupermarketWithin1Km_25
1.905440
## 10                               DistanceToImportantTransferStation_91
1.900383
## 11 percentage_have_experienced_a_lot_of_stress_in_past_four_weeks_20
1.898376
## 12 percentage_using_care_total_health_insurance_act_0_to_17_years_21
1.893754
## 13                                             education_level_high_21
1.866689
## 14                                                DistanceToCinema_104
```

```
1.866672
## 15            niusance_from_harassing_people_on_te_street_percent_21
1.866244
## 16                           percentage_who_provide_informal_care_20
1.852863
## 17                                         points_of_sale_stores_21
1.840112
## 18                           nuisance_from_local_residents_percent_21
1.837621
## 19    very_satisfied_with_street_lighting_in_neighborhood_percent_21
1.831333
## 20                           people_hardly_know_each_other_percent_21
1.830982
## 21            score_physical_quality_of_living_environment_percent_21
1.828014
## 22   average_SES_WOA_partial_score_of_financial_educational_level_19
1.822678
## 23            I_have_a_lot_of_contact_with_locl_residents_percent_21
1.821171
## 24                                       PrimarySchoolWithin1Km_61
1.820203
## 25                                            Havo_wvoWithin3Km_73
1.818889
## 26       Average_SES_WOAscore_partial_score_of_employment_history_19
1.795590
## 27                         victimization_of_ciolent_crimes_percent_21
1.794567
## 28                                     vacant_houses_percentage_21
1.791279
## 29                                     DistanceToHavo_vwoSchool_72
1.769313
## 30                                        RestaurantWithin1Km_45
1.754876
## 31                                   DistanceToMainRoadEntrance_89
1.749399
## 32                                      AttractionsWithin10Km_111
1.747659
## 33                       victimization_of_property_crimes_percent_21
1.746266
## 34                                          DistanceToHotelEtc_48
1.742540
## 35                                     OtherDailyFoodWithin1Km_29
1.735842
## 36                                            secondary_schools_21
1.732353
## 37            average_personal_yearly_income_individuals_in_euros_21
1.714111
## 38                            destroyed_street_furniture_percent_21
1.710363
## 39                                nuisance_caused_by_cofused_person_21
```

```
1.707820
## 40                        percentage_people_with_pgysical_disabilities_20
1.698491
## 41                       percentage_of_sports_associations_member_ship_17
1.693369
## 42                                                DistanceToAttraction_110
1.689416
## 43                                                      happiness_index_18
1.663928
## 44                              I_feel_at_home_with_the_people_percent_21
1.663562
## 45                                                percentage_overweight_20
1.656473
## 46               victimization_of_proerty_crimes_home_burglary_percent_21
1.646397
## 47                                homes_for_single_families_percentage_21
1.642417
## 48             percentage_high_risk_for_anxiety_disorder_or_depression_20
1.639353
## 49                                                        WmboWithin3Km_69
1.636007
## 50                                 percentage_middle_income_households_21
1.630348

# Plot importances
ggplot(data = varimp_data,
       aes(x = reorder(feature, -importance), y = importance, fill =
feature)) +
  geom_bar(stat="identity") + labs(x = "Features", y = "Variable Importance")
+
  geom_text(aes(label = round(importance, 2)), vjust=1.6, color="white",
size=4) +
  theme_bw() + theme(legend.position = "none")
```



```
# Post prediction
postResample(predict(result_rfe1, x_test), y_test)

##       RMSE   Rsquared        MAE
## 1.40201705 0.01937057 1.23617273

# Variables the model automatically suggests. We will choose manually from
the top 50 though
selected_predictors

##                      predictors(result_rfe1)
## 1 percentage_who_provide_informal_care_20
## 2                  points_of_sale_stores_21
## 3                   education_level_high_21
## 4                      Havo_wvoWithin3Km_73
## 5                        Social_cohesion_21

# Save automatically suggested variables and top 50 importances for manual
revision
write.xlsx(selected_predictors, saveselectedvars, colNames = FALSE)
write.xlsx(varimp_data, saveimportances, colNames = TRUE, row.nmaes = FALSE)

# Check dataframes: selected_predictors, varimp_data
```

OPTIONAL: FINAL FEATURE SELECTION 2/2 After we add the urban features to an excel with the rest of the features we run random forests to keep only important urban features and examine the optimal number of features based on their importances. This step is already completed and the final features are already stored in a new folder.

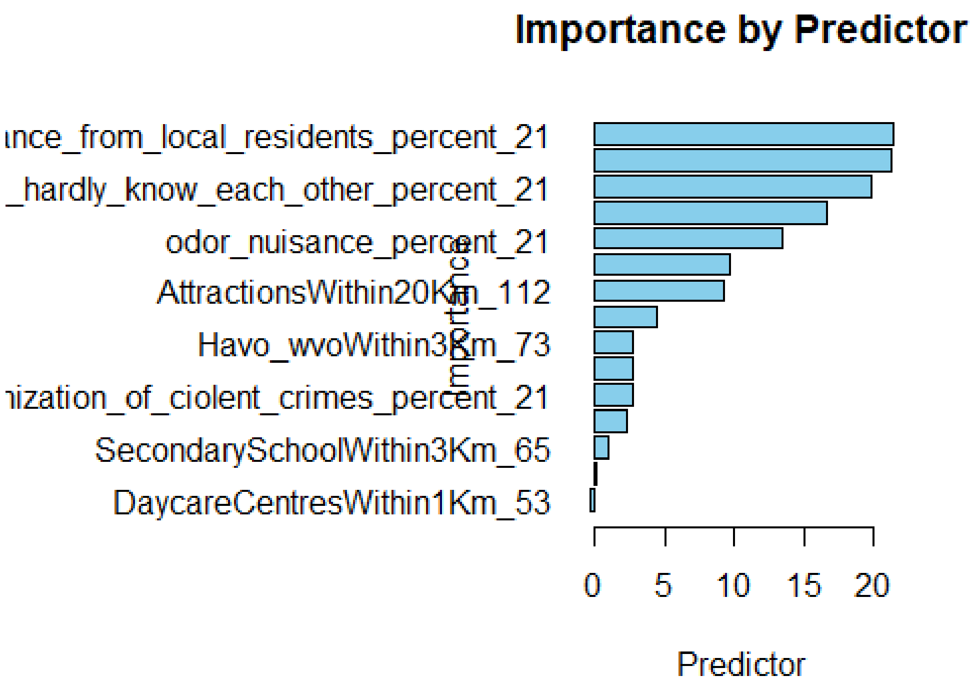
```
# SAME AS CHUNK CODE 7.
# The difference is that we manually selected variables again based on chunk
7 and chunk 8 outputs. Again, we optimize the number of variables with the
new variables list.
# For the first run, the number of variables should be equal to the number of
features we import.
number_of_variables <- 16


# Read variables and read full data
variables <- "intermediary files/Social_cohesion_21_FinalVariables.xlsx"
citydata <- "intermediary files/haag2021clean.xlsx"
features <- read.xlsx(variables, colNames = FALSE)
datafull <- read.xlsx(citydata, rowNames = TRUE)
# Refine full data to only for the selected variables
column_indices <- match(features$X1, names(datafull))
data_fixed <- datafull[, column_indices]
length(column_indices)
```

```
## [1] 16
```

```
subset_indexes <- c(1:number_of_variables)
data_fixed <- data_fixed[, subset_indexes]




# RANDOM FOREST
# Initialize an empty vector to store prediction errors for LOOCV
predictionsrf <- vector("numeric", length = nrow(data_fixed))
accuraciesrf <- vector("numeric", length = nrow(data_fixed))
mserf <- vector("numeric", length = nrow(data_fixed))
maerf <- vector("numeric", length = nrow(data_fixed))
maperf <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  # Create the training set by excluding the ith observation
  train_data <- data_fixed[-i, ]

  # Create the test set with only the ith observation
  test_data <- data_fixed[i, ]

  # Fit a Random Forest model
  rffeatures <- randomForest(train_data$Social_cohesion_21 ~ ., data =
train_data, ntree = 1000, mtry = 5, importance = TRUE)

  if (i == 1) {
```

```
    importance_matrix <- as.numeric(matrix(NA, nrow = ncol(data_fixed) - 1,
ncol = 2))
    # Store feature importance
    importance_matrix <- importance(rffeatures)[,1]
  } else {
    # Store feature importance
    importance_matrix <- cbind(importance_matrix, importance(rffeatures)[,1])
  }

  # Make predictions on train dataset
  predictionsrf[i] <- predict(rffeatures, test_data, type = "response")

  # MSE
  mserf[i] <- (predictionsrf[i] - test_data$Social_cohesion_21)^2

  # MSE
  maerf[i] <- MAE(predictionsrf[i], test_data$Social_cohesion_21)

  # MAPE
  maperf[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsrf[i]) /
test_data$Social_cohesion_21) * 100)

  # Accuracy on train set
  accuraciesrf[i] <- Accuracy(round(predictionsrf[i],1),
round(test_data$Social_cohesion_21, 1))
}
# Handle Metrics
importances <- apply(importance_matrix, 1, mean)
importances <- data.frame(importance = importances)
importances <- importances %>% arrange(desc(importance))
accuracyrf <- mean(accuraciesrf)
mserf <- mean(mserf)
maerf <- mean(maerf)
maperf <- mean(maperf)
rsquaredrf <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsrf)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredrfAdj <- 1 - ((1 - rsquaredrf) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))


# Show metrics
importances

##                                                            importance
## nuisance_from_local_residents_percent_21                   21.4061420
## percentage_of_sports_associations_member_ship_17           21.2614507
## people_hardly_know_each_other_percent_21                   19.8116523
## non_working_job_seekers_total_17                           16.6715213
## odor_nuisance_percent_21                                   13.4533357
```

```
## score_physical_quality_of_living_environment_percent_21  9.7109315
## AttractionsWithin20Km_112                                 9.2538286
## vacant_houses_percentage_21                               4.5458989
## Havo_wvoWithin3Km_73                                      2.7892438
## PerformingArtsWithin5Km_100                               2.6975322
## victimization_of_ciolent_crimes_percent_21                2.6949419
## DistanceToCinema_104                                      2.2690942
## SecondarySchoolWithin3Km_65                               1.0385306
## points_of_sale_stores_21                                  0.1719603
## DaycareCentresWithin1Km_53                               -0.2065946

print(paste(round(accuracyrf, 2), "Accuracy RF"))

## [1] "0.11 Accuracy RF"

print(paste(round(mserf, 5), "MSE RF"))

## [1] "0.04434 MSE RF"

print(paste(round(maerf, 5), "MAE RF"))

## [1] "0.16692 MAE RF"

print(paste(round(maperf, 5), "MAPE RF"))

## [1] "2.84801 MAPE RF"

print(paste(round(rsquaredrf, 3), "R^2 RF"))

## [1] "0.864 R^2 RF"

print(paste(round(rsquaredrfAdj, 3), "R^2 Adjusted RF"))

## [1] "0.784 R^2 Adjusted RF"

# Plot importance
par(mar=c(5, 15, 4, 2) + 0.1)
importances_reversed <- importances %>% arrange(importance)
barplot(importances_reversed$importance, names.arg =
rownames(importances_reversed), main = "Importance by Predictor", xlab =
"Predictor", ylab = "Importance", horiz = TRUE, col = "skyblue", las = 1,
cex.names=1.0)
```



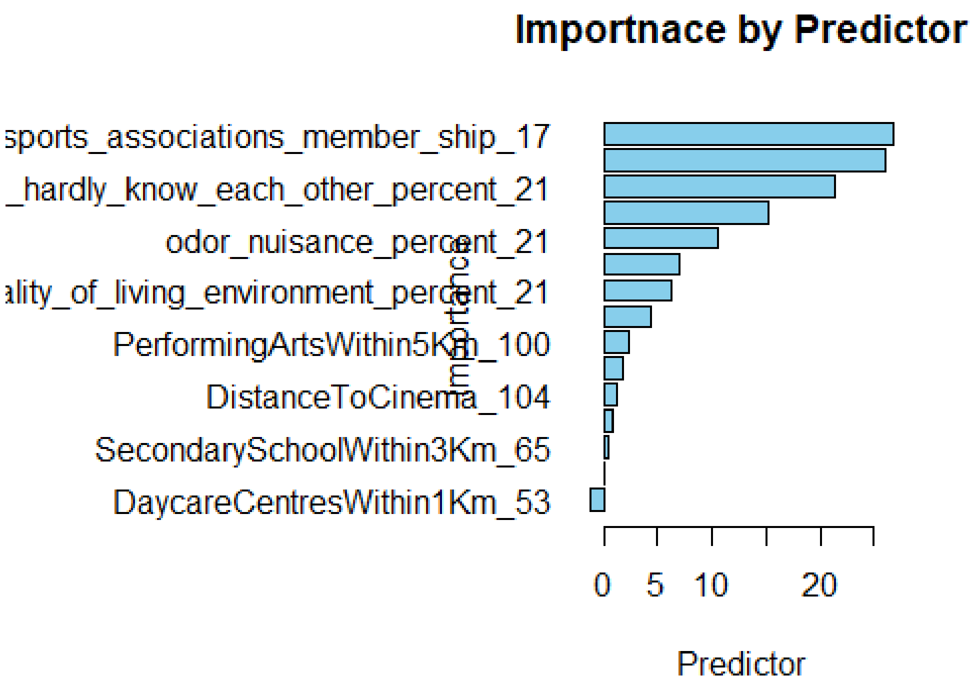

```r
# Metrics data frame
rfmetrics <- data.frame(
  MAE = c(round(maerf, 3)),
  MSE = c(round(mserf, 5)),
  MAPE = c(round(maperf, 5)),
  R_squared = c(round(rsquaredrf, 3)),
  R_squared_adjusted = c(round(rsquaredrfAdj, 3)),
  row.names = c("RF metrics")
)

# Subsetting data to get ready for a rerun
column_refinement <- match(rownames(importances), names(datafull))
data_fixed <- datafull[, column_refinement]
data_fixed <- cbind(datafull$Social_cohesion_21, datafull[,
column_refinement])
colnames(data_fixed)[1] <- "Social_cohesion_21"

# Save variables list for rerun
variables <- as.data.frame(c("Social_cohesion_21",
colnames(datafull)[column_refinement]), col.names = "variable")
savepath3 <- "intermediary files/Social_cohesion_21_FinalVariables.xlsx"
write.xlsx(variables, savepath3, rowNames = FALSE, colNames = FALSE)

# Check dataframes: importances, rfmetrics
```

THE HAGUE - ML MODELS

```
# Input final dataset with manually selected features
# Run: 6 models
# Output: table with 5 metrics for each models

# Read variables and read full data
variables <- "intermediary files/Social_cohesion_21_FinalVariables.xlsx"
citydata <- "intermediary files/haag2021clean.xlsx"
features <- read.xlsx(variables, colNames = FALSE)
datafull <- read.xlsx(citydata, rowNames = TRUE)
# Refine full data to only for the selected variables
column_indices <- match(features$X1, names(datafull))
data_fixed <- datafull[, column_indices]



# RANDOM FOREST
# Initialize an empty vector to store metrics for LOOCV
predictionsrf <- vector("numeric", length = nrow(data_fixed))
accuraciesrf <- vector("numeric", length = nrow(data_fixed)) # will not be
taken into account
mserf <- vector("numeric", length = nrow(data_fixed))
maerf <- vector("numeric", length = nrow(data_fixed))
maperf <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  # Create the training set by excluding the ith observation
  train_data <- data_fixed[-i, ]

  # Create the test set with only the ith observation
  test_data <- data_fixed[i, ]

  # Fit a Random Forest model
  rffeatures <- randomForest(train_data$Social_cohesion_21 ~ ., data =
train_data, ntree = 1000, mtry = 10, importance = TRUE)

  if (i == 1) {
    # Initialize importances table
    importance_matrix <- as.numeric(matrix(NA, nrow = ncol(data_fixed) - 1,
ncol = 2))
    # Store feature importance
    importance_matrix <- importance(rffeatures)[,1]
  } else {
    # Store recurrent feature importances
    importance_matrix <- cbind(importance_matrix, importance(rffeatures)[,1])
  }
```

```
  # Make predictions on train dataset
  predictionsrf[i] <- predict(rffeatures, test_data, type = "response")

  # MSE
  mserf[i] <- (predictionsrf[i] - test_data$Social_cohesion_21)^2

  # MSE
  maerf[i] <- MAE(predictionsrf[i], test_data$Social_cohesion_21)

  # MAPE
  maperf[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsrf[i]) /
test_data$Social_cohesion_21) * 100)

  # Accuracy on train set
  accuraciesrf[i] <- Accuracy(round(predictionsrf[i],1),
round(test_data$Social_cohesion_21, 1))
}
# Handle Metrics - take means and calculate Rsquareds
importances <- apply(importance_matrix, 1, mean)
importances <- data.frame(importance = importances)
importances <- importances %>% arrange(desc(importance))
accuracyrf <- mean(accuraciesrf)
mserf <- mean(mserf)
maerf <- mean(maerf)
maperf <- mean(maperf)
rsquaredrf <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsrf)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredrfAdj <- 1 - ((1 - rsquaredrf) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))


# Show metrics
importances

##                                                          importance
## percentage_of_sports_associations_member_ship_17        26.848312871
## nuisance_from_local_residents_percent_21                26.095292543
## people_hardly_know_each_other_percent_21                21.326891741
## non_working_job_seekers_total_17                        15.160979177
## odor_nuisance_percent_21                                10.589411546
## AttractionsWithin20Km_112                                7.007740686
## score_physical_quality_of_living_environment_percent_21  6.171828238
## vacant_houses_percentage_21                              4.441264294
## PerformingArtsWithin5Km_100                              2.265909400
## Havo_wvoWithin3Km_73                                     1.812129357
## DistanceToCinema_104                                     1.223966558
## victimization_of_ciolent_crimes_percent_21               0.867907100
## SecondarySchoolWithin3Km_65                              0.394083982
```

```
## points_of_sale_stores_21                                    -0.002290663
## DaycareCentresWithin1Km_53                                  -1.147653293

print(paste(round(accuracyrf, 2), "Accuracy RF"))

## [1] "0.16 Accuracy RF"

print(paste(round(mserf, 5), "MSE RF"))

## [1] "0.04324 MSE RF"

print(paste(round(maerf, 5), "MAE RF"))

## [1] "0.16219 MAE RF"

print(paste(round(maperf, 5), "MAPE RF"))

## [1] "2.77227 MAPE RF"

print(paste(round(rsquaredrf, 3), "R^2 RF"))

## [1] "0.868 R^2 RF"

print(paste(round(rsquaredrfAdj, 3), "R^2 Adjusted RF"))

## [1] "0.79 R^2 Adjusted RF"

# Plot importance
par(mar=c(5, 15, 4, 2) + 0.1)
importances_reversed <- importances %>% arrange(importance)
barplot(importances_reversed$importance, names.arg =
rownames(importances_reversed), main = "Importnace by Predictor", xlab =
"Predictor", ylab = "Importance", horiz = TRUE, col = "skyblue", las = 1,
cex.names=1.0)
```




```
# FROM THIS POIN ON COMMENTS WILL BE SPARSE. THE FOLLOWING 5 MODELS KEEP THE
SAME STRUCTURE AS THE FIRST ONE.




# Regression
predictionsreg <- vector("numeric", length = nrow(data_fixed))
accuraciesreg <- vector("numeric", length = nrow(data_fixed))
msereg <- vector("numeric", length = nrow(data_fixed))
maereg <- vector("numeric", length = nrow(data_fixed))
mapereg <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  train_data <- data_fixed[-i, ]
  test_data <- data_fixed[i, ]
  # Fit a Random Forest model
  regfeatures <- lm(train_data$Social_cohesion_21 ~ ., data = train_data)
  predictionsreg[i] <- predict(regfeatures, test_data, type = "response")
  # Metrics
  msereg[i] <- (predictionsreg[i] - test_data$Social_cohesion_21)^2
  maereg[i] <- MAE(predictionsreg[i], test_data$Social_cohesion_21)
  mapereg[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsreg[i]) /
test_data$Social_cohesion_21) * 100)
  accuraciesreg[i] <- Accuracy(round(predictionsreg[i], 1),
```

```r
round(test_data$Social_cohesion_21, 1))
}

# Handle metrics - take means
accuracyreg <- mean(accuraciesreg)
msereg <- mean(msereg)
maereg <- mean(maereg)
mapereg <- mean(mapereg)
rsquaredreg <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsreg)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredregAdj <- 1 - ((1 - rsquaredreg) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))

print(paste(round(accuracyreg, 2), "Accuracy Reg"))

## [1] "0.25 Accuracy Reg"

print(paste(round(msereg, 5), "MSE Reg"))

## [1] "0.03236 MSE Reg"

print(paste(round(maereg, 5), "MAE Reg"))

## [1] "0.13864 MAE Reg"

print(paste(round(mapereg, 5), "MAPE Reg"))

## [1] "2.36283 MAPE Reg"

print(paste(round(rsquaredreg, 3), "R^2 Reg"))

## [1] "0.901 R^2 Reg"

print(paste(round(rsquaredregAdj, 3), "R^2 Adjusted Reg"))

## [1] "0.843 R^2 Adjusted Reg"

# SVR
predictionssvr <- vector("numeric", length = nrow(data_fixed))
accuraciessvr <- vector("numeric", length = nrow(data_fixed))
msesvr <- vector("numeric", length = nrow(data_fixed))
maesvr <- vector("numeric", length = nrow(data_fixed))
mapesvr <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  train_data <- data_fixed[-i, ]
  test_data <- data_fixed[i, ]
  # Fit an svr model
  svrfeatures <- svm(Social_cohesion_21 ~ . -Social_cohesion_21, data =
train_data, type = 'eps-regression', kernel = 'radial', epsilon = 0.1)
  predictionssvr[i] <- predict(svrfeatures, test_data, type = "response")
```

```r
  # Metrics
  msesvr[i] <- (predictionssvr[i] - test_data$Social_cohesion_21)^2
  maesvr[i] <- MAE(predictionssvr[i], test_data$Social_cohesion_21)
  mapesvr[i] <- mean(abs((test_data$Social_cohesion_21 - predictionssvr[i]) /
test_data$Social_cohesion_21) * 100)
  accuraciessvr[i] <- Accuracy(round(predictionssvr[i], 1),
round(test_data$Social_cohesion_21, 1))
}

# Metrics - means
accuracysvr <- mean(accuraciessvr)
msesvr <- mean(msesvr)
maesvr <- mean(maesvr)
mapesvr <- mean(mapesvr)
rsquaredsvr <- 1 - sum((data_fixed$Social_cohesion_21 - predictionssvr)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredsvrAdj <- 1 - ((1 - rsquaredsvr) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))

print(paste(round(accuracysvr, 2), "Accuracy SVR"))

## [1] "0.14 Accuracy SVR"

print(paste(round(msesvr, 5), "MSE SVR"))

## [1] "0.06246 MSE SVR"

print(paste(round(maesvr, 5), "MAE SVR"))

## [1] "0.18956 MAE SVR"

print(paste(round(mapesvr, 5), "MAPE SVR"))

## [1] "3.30224 MAPE SVR"

print(paste(round(rsquaredsvr, 3), "R^2 SVR"))

## [1] "0.809 R^2 SVR"

print(paste(round(rsquaredsvrAdj, 3), "R^2 Adjusted SVR"))

## [1] "0.696 R^2 Adjusted SVR"

# Ridge Regression
library(glmnet)

# Define custom MAE and Accuracy functions
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

accuracy <- function(predicted, actual) {
```

```
  mean(predicted == actual)
}

# Initialize vectors to store results
predictionsrreg <- numeric(nrow(data_fixed))
accuraciesrreg <- numeric(nrow(data_fixed))
mserreg <- numeric(nrow(data_fixed))
maerreg <- numeric(nrow(data_fixed))
maperreg <- numeric(nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  train_data <- data_fixed[-i, ]
  test_data <- data_fixed[i, , drop = FALSE]

  # Convert training data to matrix
  train_data_numeric <- model.matrix(~ . - 1, data = train_data)
  train_x <- train_data_numeric[, -which(colnames(train_data_numeric) ==
"Social_cohesion_21")]
  train_y <- train_data$Social_cohesion_21

  # Perform cross-validation to find the best lambda
  cv_ridge <- cv.glmnet(x = train_x, y = train_y, family = "gaussian", alpha
= 0)

  # Best lambda based on minimum criteria
  lambda_best <- cv_ridge$lambda.min

  # Train a Ridge Regression model
  ridge_regression <- glmnet(x = train_x, y = train_y, family = "gaussian",
alpha = 0, lambda = lambda_best)

  # Convert test data to matrix
  test_data_numeric <- model.matrix(~ . - 1, data = test_data)
  test_x <- test_data_numeric[, -which(colnames(test_data_numeric) ==
"Social_cohesion_21")]

  # Predict on test set
  predictionsrreg[i] <- predict(ridge_regression, newx = test_x)

  # Metrics
  mserreg[i] <- (predictionsrreg[i] - test_data$Social_cohesion_21)^2
  maerreg[i] <- mae(test_data$Social_cohesion_21, predictionsrreg[i])
  maperreg[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsrreg[i])
/ test_data$Social_cohesion_21) * 100)
  accuraciesrreg[i] <- accuracy(round(predictionsrreg[i], 1),
round(test_data$Social_cohesion_21, 1))
}
```

```r
# Metric means
accuracyrreg <- mean(accuraciesrreg)
mserreg_mean <- mean(mserreg)
maerreg_mean <- mean(maerreg)
maperreg_mean <- mean(maperreg)
rsquaredrreg <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsrreg)^2)
/ sum((data_fixed$Social_cohesion_21 -
mean(data_fixed$Social_cohesion_21))^2)
rsquaredrregAdj <- 1 - ((1 - rsquaredrreg) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))

print(paste(round(accuracyrreg, 2), "Accuracy Rreg"))

## [1] "0.14 Accuracy Rreg"

print(paste(round(mserreg_mean, 5), "MSE Rreg"))

## [1] "0.02624 MSE Rreg"

print(paste(round(maerreg_mean, 5), "MAE Rreg"))

## [1] "0.13737 MAE Rreg"

print(paste(round(maperreg_mean, 5), "MAPE Rreg"))

## [1] "2.3323 MAPE Rreg"

print(paste(round(rsquaredrreg, 3), "R^2 Rreg"))

## [1] "0.92 R^2 Rreg"

print(paste(round(rsquaredrregAdj, 3), "R^2 Adjusted Rreg"))

## [1] "0.872 R^2 Adjusted Rreg"

# Polynomial Regression
predictionspreg <- vector("numeric", length = nrow(data_fixed))
accuraciespreg <- vector("numeric", length = nrow(data_fixed))
msepreg <- vector("numeric", length = nrow(data_fixed))
maepreg <- vector("numeric", length = nrow(data_fixed))
mapepreg <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  train_data <- data_fixed[-i, ]
  test_data <- data_fixed[i, ]

  train_data_sub <- as.matrix(train_data[, -which(names(train_data) ==
"Social_cohesion_21"), drop = FALSE])
  # Fit a Random Forest model
  pregfeatures <- lm(train_data$Social_cohesion_21 ~ train_data_sub +
I(train_data_sub^2), data = train_data)
```

```r
  # Make predictions on train dataset
  predictionspreg[i] <- predict(pregfeatures, test_data, type = "response")

  # Merics
  msepreg[i] <- (predictionspreg[i] - test_data$Social_cohesion_21)^2
  maepreg[i] <- MAE(predictionspreg[i], test_data$Social_cohesion_21)
  mapepreg[i] <- mean(abs((test_data$Social_cohesion_21 - predictionspreg[i])
/ test_data$Social_cohesion_21) * 100)
  accuraciespreg[i] <- Accuracy(round(predictionspreg[i], 1),
round(test_data$Social_cohesion_21, 1))
}

# Metrics means
accuracypreg <- mean(accuraciespreg)
msepreg <- mean(msepreg)
maepreg <- mean(maepreg)
mapepreg <- mean(mapepreg)
rsquaredpreg <- 1 - sum((data_fixed$Social_cohesion_21 - predictionspreg)^2)
/ sum((data_fixed$Social_cohesion_21 -
mean(data_fixed$Social_cohesion_21))^2)
rsquaredpregAdj <- 1 - ((1 - rsquaredpreg) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))

print(paste(round(accuracypreg, 2), "Accuracy Preg"))

## [1] "0.16 Accuracy Preg"

print(paste(round(msepreg, 5), "MSE Preg"))

## [1] "0.33556 MSE Preg"

print(paste(round(maepreg, 5), "MAE Preg"))

## [1] "0.46569 MAE Preg"

print(paste(round(mapepreg, 5), "MAPE Preg"))

## [1] "7.92398 MAPE Preg"

print(paste(round(rsquaredpreg, 3), "R^2 Preg"))

## [1] "-0.026 R^2 Preg"

print(paste(round(rsquaredpregAdj, 3), "R^2 Adjusted Preg"))

## [1] "-0.633 R^2 Adjusted Preg"

# RESULTS IF GBM IS SKIPPED
accuracygbm <- 999
msegbm <- 999
maegbm <- 999
mapegbm <- 999
rsquaredgbm <- 999
```

```r
rsquaredgbmAdj <- 999

# GBM
predictionsgbm <- vector("numeric", length = nrow(data_fixed))
accuraciesgbm <- vector("numeric", length = nrow(data_fixed))
msegbm <- vector("numeric", length = nrow(data_fixed))
maegbm <- vector("numeric", length = nrow(data_fixed))
mapegbm <- vector("numeric", length = nrow(data_fixed))

# Perform LOOCV
for (i in 1:nrow(data_fixed)) {
  train_data <- data_fixed[-i, ]
  test_data <- data_fixed[i, ]
  # Fit the model
  set.seed(42) # for reproducibility
  gbmfeatures = gbm(train_data$Social_cohesion_21 ~ ., data = train_data,
distribution = "gaussian", n.trees = 1000, interaction.depth = 4, shrinkage =
0.01, cv.folds = 5, n.minobsinnode = 5)
  # Make predictions on train dataset
  predictionsgbm[i] <- predict(gbmfeatures, test_data, type = "response")
  # Metrics
  msegbm[i] <- (predictionsgbm[i] - test_data$Social_cohesion_21)^2
  maegbm[i] <- MAE(predictionsgbm[i], test_data$Social_cohesion_21)
  mapegbm[i] <- mean(abs((test_data$Social_cohesion_21 - predictionsgbm[i]) /
test_data$Social_cohesion_21) * 100)
  accuraciesgbm[i] <- Accuracy(round(predictionsgbm[i], 1),
round(test_data$Social_cohesion_21, 1))
}
# Metrics means
accuracygbm <- mean(accuraciesgbm)
msegbm <- mean(msegbm)
maegbm <- mean(maegbm)
mapegbm <- mean(mapegbm)
rsquaredgbm <- 1 - sum((data_fixed$Social_cohesion_21 - predictionsgbm)^2) /
sum((data_fixed$Social_cohesion_21 - mean(data_fixed$Social_cohesion_21))^2)
rsquaredgbmAdj <- 1 - ((1 - rsquaredgbm) * (nrow(data_fixed) - 1) /
(nrow(data_fixed) - ncol(data_fixed) - 1))

print(paste(round(accuracygbm, 2), "Accuracy GBM"))

## [1] "0.07 Accuracy GBM"

print(paste(round(msegbm, 5), "MSE GBM"))

## [1] "0.03791 MSE GBM"

print(paste(round(maegbm, 5), "MAE GBM"))

## [1] "0.15643 MAE GBM"

print(paste(round(mapegbm, 5), "MAPE GBM"))
```

```
## [1] "2.69128 MAPE GBM"

print(paste(round(rsquaredgbm, 3), "R^2 GBM"))

## [1] "0.884 R^2 GBM"

print(paste(round(rsquaredgbmAdj, 3), "R^2 Adjusted GBM"))

## [1] "0.815 R^2 Adjusted GBM"

# RESULTS DATA FRANE: 5 metrics x 6 models
results <- data.frame(
  MAE = c(round(maerf, 3), round(maereg, 3), round(maesvr, 3),
round(maerreg_mean, 3), round(maepreg, 3), round(maegbm, 5)),
  MSE = c(round(mserf, 5), round(msereg, 5), round(msesvr, 5),
round(mserreg_mean, 5), round(msepreg, 5), round(msegbm, 5)),
  MAPE = c(round(maperf, 5), round(mapereg, 5), round(mapesvr, 5),
round(maperreg_mean, 5), round(mapepreg, 5), round(mapegbm, 5)),
  R_squared = c(round(rsquaredrf, 3), round(rsquaredreg, 3),
round(rsquaredsvr, 3), round(rsquaredrreg, 3), round(rsquaredpreg, 3),
round(rsquaredgbm, 3)),
  R_squared_adjusted = c(round(rsquaredrfAdj, 3), round(rsquaredregAdj, 3),
round(rsquaredsvrAdj, 3), round(rsquaredrregAdj, 3), round(rsquaredpregAdj,
3), round(rsquaredgbmAdj, 3)),
  row.names = c("RF", "Reg", "SVR", "Ridge reg", "Poly reg", "GBM")
)
results

##                 MAE     MSE    MAPE R_squared R_squared_adjusted
## RF        0.16200 0.04324 2.77227     0.868              0.790
## Reg       0.13900 0.03236 2.36283     0.901              0.843
## SVR       0.19000 0.06246 3.30224     0.809              0.696
## Ridge reg 0.13700 0.02624 2.33230     0.920              0.872
## Poly reg  0.46600 0.33556 7.92398    -0.026             -0.633
## GBM       0.15643 0.03791 2.69128     0.884              0.815

# Store final output
write.xlsx(results, "output/output_hague.xlsx", rowNames = TRUE, colNames =
TRUE)
```

AMSTERDAM

FINAL FEATURE SELECTION AND MODELS [starting from clean dataset, with 5-fold cv]

```
set.seed(123)

# Clean Data set for Amsterdam
data_clean <- read.xlsx("source data/source_data_amsterdam.xlsx", rowNames =
TRUE)

# RANDOM FOREST
# Fitting a random forest and showing importance to select final features
rf_modelPW <- randomForest(data_clean$`Prettig.wonen.(1-10)`
~data_clean$Huur.gemiddeld +data_clean$`Thuisvoelen.(1-10)` +
data_clean$`Betrokkenheid.buurt.(1-10)` +
                             data_clean$`Discriminatie.(%.wel.eens)`
+data_clean$`Omgang.groepen.(1-10)` +data_clean$`Kantoren.(%)` +
                             data_clean$`Schoon.straat.(1-10)`
+data_clean$`Onderhoud.straat.(1-10)` +data_clean$`Buurt.schoon.(%)` +
                             data_clean$`Sportvestigingen./.1.000.inw.`
+data_clean$`Mensen.helpen.elkaar.(1-10)`
+data_clean$`Schoon.speelplaatsen.(1-10)` +
                             data_clean$`Zorgvoorzieningen.(1-10)`
+data_clean$`Welzijnsvoorzieningen./1.000.inw` +
                             data_clean$`Contact.in.de.buurt.(1-10)` ,data =
data_clean[,5:42], importance = TRUE, ntree = 1000)
importance(rf_modelPW)

##                                                %IncMSE IncNodePurity
## data_clean$Huur.gemiddeld                      4.517504     0.2570430
## data_clean$`Thuisvoelen.(1-10)`               30.387798     6.8834170
## data_clean$`Betrokkenheid.buurt.(1-10)`        8.711198     1.5541197
## data_clean$`Discriminatie.(%.wel.eens)`       12.450054     1.9005777
## data_clean$`Omgang.groepen.(1-10)`            23.137262     5.5999648
## data_clean$`Kantoren.(%)`                     18.433962     2.8299917
## data_clean$`Schoon.straat.(1-10)`              2.178612     0.4183244
## data_clean$`Onderhoud.straat.(1-10)`           3.617534     0.4813043
## data_clean$`Buurt.schoon.(%)`                 10.617766     1.5410975
## data_clean$`Sportvestigingen./.1.000.inw.`    12.746454     1.0217176
## data_clean$`Mensen.helpen.elkaar.(1-10)`      10.574545     1.5124955
## data_clean$`Schoon.speelplaatsen.(1-10)`       3.914227     0.3192587
## data_clean$`Zorgvoorzieningen.(1-10)`          9.865704     0.5523826
## data_clean$`Welzijnsvoorzieningen./1.000.inw`  1.275987     0.4852378
## data_clean$`Contact.in.de.buurt.(1-10)`        2.107163     0.4060410
```

```
# Making the final model
control <- trainControl(method = "cv", number = 5,savePredictions = "final")
rf_model_PW <- train(`Prettig.wonen.(1-10)`~`Thuisvoelen.(1-10)`+
```

```
`Betrokkenheid.buurt.(1-10)`+
                       `Discriminatie.(%.wel.eens)`+`Omgang.groepen.(1-
10)`+`Kantoren.(%)`+`Buurt.schoon.(%)`+`Mensen.helpen.elkaar.(1-10)`,data =
data_clean, method="rf",
                     trControl=control,
                     metric="Rsquared",
                     tuneGrid = data.frame(.mtry = c(2, 3, 4)),
                     ntree = 1000,  # Specify number of trees here
)

  ###Function to calculate Adjusted R-squared
calculate_adjusted_r_squared <- function(df, num_predictors) {
  ss_res <- sum((df$obs - df$pred)^2)
  ss_tot <- sum((df$obs - mean(df$obs))^2)
  r_squared <- 1 - (ss_res / ss_tot)

  n <- nrow(df)  # Total number of data points in the fold
  adjusted_r_squared <- 1 - ((1 - r_squared) * (n - 1) / (n - num_predictors
- 1))
  adjusted_r_squared
}
num_predictors <- 7

resamples_summary <- rf_model_PW$resample

predictions <- rf_model_PW$pred %>%
  mutate(
    obs = as.numeric(as.character(obs)),
    pred = as.numeric(as.character(pred))
  )

  ###Group by 'Resample' and calculate the metrics for each fold
metrics_per_fold <- predictions %>%
  group_by(Resample) %>%
  summarise(MSE = mean((obs - pred)^2),
            MAPE = mean(abs((obs - pred) / obs)) * 100,
            Adjusted_R_squared = calculate_adjusted_r_squared(cur_data(),
num_predictors),
  )

print(metrics_per_fold)

## # A tibble: 5 × 4
##   Resample    MSE  MAPE Adjusted_R_squared
##   <chr>     <dbl> <dbl>              <dbl>
## 1 Fold1    0.0384  2.17              0.811
## 2 Fold2    0.0544  2.41              0.645
## 3 Fold3    0.107   2.63              0.351
## 4 Fold4    0.0324  2.07              0.820
## 5 Fold5    0.0333  1.92              0.697
```

```
# Final Metrics
mae_mean<-mean(resamples_summary$MAE)
mse_mean<-mean(metrics_per_fold$MSE)
mape_mean<-mean(metrics_per_fold$MAPE)
rmse_mean <- mean(resamples_summary$RMSE)
rsq_mean <- mean(resamples_summary$Rsquared)
r2_adjusted_mean <- mean(metrics_per_fold$Adjusted_R_squared)

metrics_summary_RF <- data.frame(
  Metric = c("MAE", "MSE", "MAPE", "RMSE", "R-squared", "Adjusted R-squared"),
  Mean = c(mae_mean, mse_mean, mape_mean, rmse_mean, rsq_mean,
r2_adjusted_mean)
)

# Create result data frame
results_amsterdam <- data.frame(
  RF = c(mae_mean, mse_mean, mape_mean, rmse_mean, rsq_mean,
r2_adjusted_mean)
)
rownames(results_amsterdam) <- c("MAE", "MSE", "MAPE", "RMSE", "R-squared",
"Adjusted R-squared")
results_amsterdam <- as.data.frame(t(results_amsterdam))
```

REGRESSION

Multivariate regression to get vif values and select features

```
model <- lm(data_clean$`Prettig.wonen.(1-10)` ~data_clean$Huur.gemiddeld
+data_clean$`Thuisvoelen.(1-10)` + data_clean$`Betrokkenheid.buurt.(1-10)` +
              data_clean$`Discriminatie.(%.wel.eens)`
+data_clean$`Omgang.groepen.(1-10)` +data_clean$`Kantoren.(%)` +
              data_clean$`Schoon.straat.(1-10)`
+data_clean$`Onderhoud.straat.(1-10)` +data_clean$`Buurt.schoon.(%)` +
              data_clean$`Sportvestigingen./.1.000.inw.`
+data_clean$`Mensen.helpen.elkaar.(1-10)`
+data_clean$`Schoon.speelplaatsen.(1-10)` +
              data_clean$`Zorgvoorzieningen.(1-10)`
+data_clean$`Welzijnsvoorzieningen./1.000.inw` +
              data_clean$`Contact.in.de.buurt.(1-10)` ,data =
data_clean[,2:42])

summary(model)

##
## Call:
## lm(formula = data_clean$`Prettig.wonen.(1-10)` ~ data_clean$Huur.gemiddeld
+
```

```
##     data_clean$`Thuisvoelen.(1-10)` + data_clean$`Betrokkenheid.buurt.(1-
10)` +
##     data_clean$`Discriminatie.(%.wel.eens)` +
data_clean$`Omgang.groepen.(1-10)` +
##     data_clean$`Kantoren.(%)` + data_clean$`Schoon.straat.(1-10)` +
##     data_clean$`Onderhoud.straat.(1-10)` + data_clean$`Buurt.schoon.(%)` +
##     data_clean$`Sportvestigingen./.1.000.inw.` +
data_clean$`Mensen.helpen.elkaar.(1-10)` +
##     data_clean$`Schoon.speelplaatsen.(1-10)` +
data_clean$`Zorgvoorzieningen.(1-10)` +
##     data_clean$`Welzijnsvoorzieningen./1.000.inw` +
data_clean$`Contact.in.de.buurt.(1-10)`,
##     data = data_clean[, 2:42])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48611 -0.09406  0.01612  0.10812  0.53307
##
## Coefficients:
##                                                   Estimate Std. Error t
value
## (Intercept)                                      2.125e+00  6.347e-01
3.348
## data_clean$Huur.gemiddeld                        3.687e-05  1.708e-04
0.216
## data_clean$`Thuisvoelen.(1-10)`                  6.971e-01  1.043e-01
6.686
## data_clean$`Betrokkenheid.buurt.(1-10)`         -3.944e-01  1.485e-01  -
2.655
## data_clean$`Discriminatie.(%.wel.eens)`         -1.233e-02  5.360e-03  -
2.299
## data_clean$`Omgang.groepen.(1-10)`              -7.185e-02  1.284e-01  -
0.560
## data_clean$`Kantoren.(%)`                        3.910e-03  3.409e-03
1.147
## data_clean$`Schoon.straat.(1-10)`               -1.442e-01  9.085e-02  -
1.587
## data_clean$`Onderhoud.straat.(1-10)`            -7.786e-02  9.825e-02  -
0.792
## data_clean$`Buurt.schoon.(%)`                    9.859e-03  2.814e-03
3.503
## data_clean$`Sportvestigingen./.1.000.inw.`       7.893e-03  1.450e-02
0.544
## data_clean$`Mensen.helpen.elkaar.(1-10)`         8.098e-01  1.165e-01
6.952
## data_clean$`Schoon.speelplaatsen.(1-10)`         8.769e-03  5.686e-02
0.154
## data_clean$`Zorgvoorzieningen.(1-10)`            1.024e-01  4.498e-02
2.276
## data_clean$`Welzijnsvoorzieningen./1.000.inw`  2.826e-03  2.213e-03
```

```
1.277
## data_clean$`Contact.in.de.buurt.(1-10)`       -3.334e-01  1.050e-01  -
3.174
##                                               Pr(>|t|)    
## (Intercept)                                   0.001184 ** 
## data_clean$Huur.gemiddeld                     0.829522    
## data_clean$`Thuisvoelen.(1-10)`               1.81e-09 ***
## data_clean$`Betrokkenheid.buurt.(1-10)`       0.009354 ** 
## data_clean$`Discriminatie.(%.wel.eens)`       0.023765 *  
## data_clean$`Omgang.groepen.(1-10)`            0.577139    
## data_clean$`Kantoren.(%)`                     0.254379    
## data_clean$`Schoon.straat.(1-10)`             0.115983    
## data_clean$`Onderhoud.straat.(1-10)`          0.430151    
## data_clean$`Buurt.schoon.(%)`                 0.000716 ***
## data_clean$`Sportvestigingen./.1.000.inw.`    0.587510    
## data_clean$`Mensen.helpen.elkaar.(1-10)`      5.31e-10 ***
## data_clean$`Schoon.speelplaatsen.(1-10)`      0.877778    
## data_clean$`Zorgvoorzieningen.(1-10)`         0.025186 *  
## data_clean$`Welzijnsvoorzieningen./1.000.inw` 0.204858    
## data_clean$`Contact.in.de.buurt.(1-10)`       0.002049 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.1945 on 91 degrees of freedom
## Multiple R-squared:   0.87,  Adjusted R-squared:  0.8485 
## F-statistic: 40.59 on 15 and 91 DF,  p-value: < 2.2e-16

#checking VIF value
vif_valuesPW <- vif(model)
vif_values <- as.data.frame(vif_valuesPW) %>% arrange(desc(vif_valuesPW))

# Keep only the variables with a VIF below 5
low_vif_varsPW <- names(vif_valuesPW)[vif_valuesPW < 5]
print(low_vif_varsPW)

## [1] "data_clean$Huur.gemiddeld"                     
## [2] "data_clean$`Discriminatie.(%.wel.eens)`"       
## [3] "data_clean$`Kantoren.(%)`"                     
## [4] "data_clean$`Buurt.schoon.(%)`"                 
## [5] "data_clean$`Sportvestigingen./.1.000.inw.`"    
## [6] "data_clean$`Schoon.speelplaatsen.(1-10)`"      
## [7] "data_clean$`Zorgvoorzieningen.(1-10)`"         
## [8] "data_clean$`Welzijnsvoorzieningen./1.000.inw`"

# Multivariate regression for pleasant living
# Create the formula for the new model
control <- trainControl(method = "cv", number = 5,savePredictions = "final")

mr_model_PW <- train(`Prettig.wonen.(1-10)`~ `Huur.gemiddeld`+
                       `Discriminatie.(%.wel.eens)`+ `Kantoren.(%)`+
```

```
`Buurt.schoon.(%)`+
                       `Sportvestigingen./.1.000.inw.`+
                       `Zorgvoorzieningen.(1-
10)`+`Welzijnsvoorzieningen./1.000.inw`,data = data_clean,
method="lm",trControl=control, metric="Rsquared")
resamples_summary_MR <- mr_model_PW$resample
predictions_MR <- mr_model_PW$pred %>%
  mutate(
    obs = as.numeric(as.character(obs)),
    pred = as.numeric(as.character(pred))
  )

num_predictors <- 7

# group by 'Resample' and calculate the metrics for each fold
metrics_per_fold_MR <- predictions_MR %>%
  group_by(Resample) %>%
  summarise(MSE = mean((obs - pred)^2),
            MAPE = mean(abs((obs - pred) / obs)) * 100,
            Adjusted_R_squared = calculate_adjusted_r_squared(cur_data(),
num_predictors),
  )
print(metrics_per_fold_MR)

## # A tibble: 5 × 4
##   Resample    MSE  MAPE Adjusted_R_squared
##   <chr>     <dbl> <dbl>              <dbl>
## 1 Fold1    0.119   3.15              0.374
## 2 Fold2    0.104   3.08              0.248
## 3 Fold3    0.112   3.77              0.456
## 4 Fold4    0.0979  3.19              0.334
## 5 Fold5    0.0849  2.99              0.343

# Metrics Multivariate reg
mae_mean_MR<-mean(resamples_summary_MR$MAE)
mse_mean_MR<-mean(metrics_per_fold_MR$MSE)
mape_mean_MR<-mean(metrics_per_fold_MR$MAPE)
rmse_mean_MR <- mean(resamples_summary_MR$RMSE)
rsq_mean_MR <- mean(resamples_summary_MR$Rsquared)
r2_adjusted_mean_MR <- mean(metrics_per_fold_MR$Adjusted_R_squared)

# Metrics data frame
metrics_summary_MR <- data.frame(
  Metric = c("MAE", "MSE", "MAPE", "RMSE", "R-squared", "Adjusted R-
squared"),
  Reg = c(mae_mean_MR, mse_mean_MR, mape_mean_MR, rmse_mean_MR, rsq_mean_MR,
r2_adjusted_mean_MR)
)
metrics_summary_MR <- as.data.frame(t(metrics_summary_MR))
```

```r
# Final output data frame with metrics from both models
results_amsterdam[2, ] <- metrics_summary_MR[2, ]
rownames(results_amsterdam)[2] <- "multivariate reg"
```

SVR model

```r
selected_vars <- c("Prettig.wonen.(1-10)", "Huur.gemiddeld", "Thuisvoelen.(1-
10)",
                   "Betrokkenheid.buurt.(1-10)",
"Discriminatie.(%.wel.eens)",
                   "Omgang.groepen.(1-10)", "Kantoren.(%)",
"Schoon.straat.(1-10)",
                   "Onderhoud.straat.(1-10)", "Buurt.schoon.(%)",
                   "Sportvestigingen./.1.000.inw.", "Mensen.helpen.elkaar.(1-
10)",
                   "Schoon.speelplaatsen.(1-10)", "Zorgvoorzieningen.(1-10)",
                   "Welzijnsvoorzieningen./1.000.inw",
"Contact.in.de.buurt.(1-10)")

data_subset <- data_clean[, selected_vars]

# Define the rfeControl function for recursive feature elimination
rfe_control <- rfeControl(functions = caretFuncs, method = "cv", number = 5)

svr_rfe <- rfe(x = data_subset[, -which(names(data_subset) ==
"Prettig.wonen.(1-10)")],
               y = data_subset$`Prettig.wonen.(1-10)`,
               sizes = c(1:42),  # You can adjust this range based on the
number of features you have
               rfeControl = rfe_control,
               method = "svmRadial")

# Print the optimal number of features
print(svr_rfe)

##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
##  Variables   RMSE Rsquared    MAE  RMSESD RsquaredSD   MAESD Selected
##          1 0.2569   0.7539 0.1956 0.02424    0.05044 0.02311
##          2 0.2671   0.7307 0.1975 0.04524    0.06909 0.01911
##          3 0.2720   0.7450 0.1989 0.06204    0.06855 0.04132
##          4 0.2755   0.7312 0.2111 0.02404    0.04710 0.02773
##          5 0.2594   0.7627 0.1939 0.04280    0.06833 0.04174
##          6 0.2554   0.7712 0.1827 0.05953    0.07496 0.04271
```

```
##          7 0.2524   0.7781 0.1855 0.04565    0.05122 0.03908
##          8 0.2488   0.7765 0.1885 0.05662    0.06338 0.04389
##          9 0.2503   0.7728 0.1879 0.06313    0.08080 0.04686
##         10 0.2441   0.7765 0.1867 0.05023    0.07266 0.04929
##         11 0.2466   0.7732 0.1876 0.05626    0.07322 0.05195
##         12 0.2559   0.7574 0.1963 0.06017    0.09133 0.05019
##         13 0.2545   0.7646 0.1937 0.05711    0.07385 0.04146
##         14 0.2491   0.7741 0.1919 0.05937    0.06795 0.04431
##         15 0.2373   0.8066 0.1833 0.04778    0.06026 0.03359         *
##
## The top 5 variables (out of 15):
##    Thuisvoelen.(1-10), Omgang.groepen.(1-10), Betrokkenheid.buurt.(1-10),
Discriminatie.(%.wel.eens), Buurt.schoon.(%)
```

```
chosen_features <- predictors(svr_rfe)
print(chosen_features)
```

```
##  [1] "Thuisvoelen.(1-10)"                "Omgang.groepen.(1-10)"
##  [3] "Betrokkenheid.buurt.(1-10)"        "Discriminatie.(%.wel.eens)"
##  [5] "Buurt.schoon.(%)"                  "Kantoren.(%)"
##  [7] "Mensen.helpen.elkaar.(1-10)"       "Schoon.straat.(1-10)"
##  [9] "Sportvestigingen./.1.000.inw."     "Zorgvoorzieningen.(1-10)"
## [11] "Onderhoud.straat.(1-10)"           "Contact.in.de.buurt.(1-10)"
## [13] "Schoon.speelplaatsen.(1-10)"       "Huur.gemiddeld"
## [15] "Welzijnsvoorzieningen./1.000.inw"
```

```
control <- trainControl(method = "cv", number = 5, savePredictions = "final")
svr_model <- train(`Prettig.wonen.(1-10)` ~ `Thuisvoelen.(1-10)` +
`Omgang.groepen.(1-10)` +
           `Betrokkenheid.buurt.(1-10)` + `Buurt.schoon.(%)` +
`Discriminatie.(%.wel.eens)`,
                   data = data_clean,
                   method = "svmRadial",
                   trControl = control,
                   metric = "Rsquared",
                   tuneLength = 10)
print(svr_model$bestTune)
```

```
##       sigma C
## 3 0.3094615 1
```

```
summary(svr_model)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
resamples_summary_svr <- svr_model$resample
predictions_svr <- svr_model$pred %>%
  mutate(
    obs = as.numeric(as.character(obs)),
    pred = as.numeric(as.character(pred))
```

```
  )

# group by 'Resample' and calculate the metrics for each fold
num_predictors <- 5
metrics_per_fold_svr <- predictions_svr %>%
  group_by(Resample) %>%
  summarise(MSE = mean((obs - pred)^2),
            MAPE = mean(abs((obs - pred) / obs)) * 100,
            Adjusted_R_squared = calculate_adjusted_r_squared(cur_data(),
num_predictors),
  )
print(metrics_per_fold_svr)

## # A tibble: 5 × 4
##   Resample    MSE  MAPE Adjusted_R_squared
##   <chr>     <dbl> <dbl>              <dbl>
## 1 Fold1    0.0544  2.52              0.720
## 2 Fold2    0.0604  2.38              0.517
## 3 Fold3    0.0272  1.77              0.859
## 4 Fold4    0.123   3.48              0.505
## 5 Fold5    0.0788  3.02              0.503

# Metrics Multivariate reg
mae_mean_svr<-mean(resamples_summary_svr$MAE)
mse_mean_svr<-mean(metrics_per_fold_svr$MSE)
mape_mean_svr<-mean(metrics_per_fold_svr$MAPE)
rmse_mean_svr <- mean(resamples_summary_svr$RMSE)
rsq_mean_svr <- mean(resamples_summary_svr$Rsquared)
r2_adjusted_mean_svr <- mean(metrics_per_fold_svr$Adjusted_R_squared)

# Metrics data frame
metrics_summary_svr <- data.frame(
  Metric = c("MAE", "MSE", "MAPE", "RMSE", "R-squared", "Adjusted R-
squared"),
  Reg = c(mae_mean_svr, mse_mean_svr, mape_mean_svr, rmse_mean_svr,
rsq_mean_svr, r2_adjusted_mean_svr)
)
metrics_summary_svr <- as.data.frame(t(metrics_summary_svr))
results_amsterdam[3, ] <- metrics_summary_svr[2, ]
rownames(results_amsterdam)[3] <- "svr"

savepath8 <- "output/output_amsterdam.xlsx"
write.xlsx(results_amsterdam, savepath8, rowNames = TRUE)
```