Příloha 1: Python skript – stažení satelitních dat.

Python skript

Celý skript v Pythonu je k dispozici ke stažení na GitHub.
Části kódu jsou uvedeny níže.

GitHub NoiseMap

Struktura kódu

0. Technické

    Rychlejší GPU

1. Úvodní

    1.1 Instalace a importy

        1.1.1 Instalace balíčků

        1.1.2 Import balíčků

    1.2 Připojení externích služeb

        1.2.1 Google Drive

        1.2.2 Google Earth Engine

2. Satelitní data

    2.1 Vstupní data a parametry

        2.1.1 Nastavení vstupních parametrů

        2.1.2 Nastavení vstupních dat

    2.2 LANDSAT 8

        2.2.1 Indexy

```
def apply_scale_factors(image):
    optical_bands = image.select('SR_B.').multiply(0.0000275).add(-0.2)
    thermal_bands = image.select('ST_B.*').multiply(0.00341802).add(149.0)
    return image.addBands(optical_bands, None, True).addBands(thermal_bands, None, True)
```

```
def rename_bands(image):
    return image.select(
        ['SR_B5', 'SR_B6', 'SR_B4', 'SR_B3', 'SR_B2', 'ST_B10'],
        ['NIR', 'SWIR1', 'red', 'green', 'blue', 'TIR1']
    )
```

NDVI přes GEE

```
def calculate_ndvi(image):
    return image.normalizedDifference(['NIR', 'red']).rename('NDVI')

```

Příloha 1: Python skript – stažení satelitních dat.

```
def calculate_ndvi_tiles(tile):
    landsat_collection_tile = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile.geometry()) \
        .map(calculate_ndvi)

    ndvi_median = landsat_collection_tile.median().rename('NDVI_Median').clip(tile.geometry())
    ndvi_max = landsat_collection_tile.max().rename('NDVI_Max').clip(tile.geometry())

    return ndvi_median, ndvi_max

ndvi_median_tiles = []
ndvi_max_tiles = []

for tile in tiles_ee:
    median, max_val = calculate_ndvi_tiles(tile)
    ndvi_median_tiles.append(median)
    ndvi_max_tiles.append(max_val)

print("NDVI Median and Max calculated and stored in separate lists.")

```

```
ndvi_median_collection = ee.ImageCollection(ndvi_median_tiles)
ndvi_max_collection = ee.ImageCollection(ndvi_max_tiles)
```

```
ndvi_median_mosaic = ndvi_median_collection.mosaic().rename('NDVI_Median')
print("Mosaic NDVI Median created.")

ndvi_max_mosaic = ndvi_max_collection.mosaic().rename('NDVI_Max')
print("Mosaic NDVI Max created.")
```

Příloha 1: Python skript – stažení satelitních dat.

NDBI přes GEE

```
def calculate_ndbi(tile, startDate, endDate):
    landsat_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile)

    ndbi = landsat_collection \
        .map(lambda image: image.normalizedDifference(['SWIR1', 'NIR']).rename('NDBI')) \
        .median() \
        .clip(tile)

    return ndbi
```

```
ndbi_tiles = [calculate_ndbi(tile, startDate, endDate) for tile in tiles_ee]
print("NDBI calculated.")
```

```
ndbi_mosaic = ee.ImageCollection(ndbi_tiles).mosaic()
```

MNDWI přes GEE

```
def calculate_mndwi(tile, startDate, endDate):
    landsat_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile)

    mndwi = landsat_collection \
        .map(lambda image: image.normalizedDifference(['green', 'SWIR1']).rename('MNDWI')) \
        .median() \
        .clip(tile)

    return mndwi
```

```
mndwi_tiles = [calculate_mndwi(tile, startDate, endDate) for tile in tiles_ee]
print("MNDVW calculated.")
```

```
mndwi_mosaic = ee.ImageCollection(mndwi_tiles).mosaic()
```

Příloha 1: Python skript – stažení satelitních dat.

UI přes GEE

```python
def calculate_ui(tile, startDate, endDate):
    landsat_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile)

    ui = landsat_collection \
        .map(lambda image: image.expression(
            '((NIR - (red + green)) / (NIR + (red + green)))',
            {
                'NIR': image.select('NIR'),
                'red': image.select('red'),
                'green': image.select('green')
            }).rename('UI')
        ) \
        .median() \
        .clip(tile)

    return ui
```

```python
ui_tiles = [calculate_ui(tile, startDate, endDate) for tile in tiles_ee]
print("UI calculated.")
```

```python
ui_mosaic = ee.ImageCollection(ui_tiles).mosaic()
```

ISA přes GEE

```python
def calculate_isa(tile, startDate, endDate):
    landsat_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile.geometry())

    ndvi = landsat_collection.map(calculate_ndvi).median().rename('NDVI').clip(tile.geometry())
    ndbi = calculate_ndbi(tile, startDate, endDate)

    # rozdil NDBI - NDVI
    isa = ndbi.subtract(ndvi).rename('ISA')

    return isa
```

```python
isa_tiles = [calculate_isa(tile, startDate, endDate) for tile in tiles_ee]
print("ISA calculated.")
```

Příloha 1: Python skript – stažení satelitních dat.

```
isa_mosaic = ee.ImageCollection(isa_tiles).mosaic()
```

Thermal data přes GEE

```
def download_tir(tile, startDate, endDate):
    landsat_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2') \
        .filterDate(startDate, endDate) \
        .map(apply_scale_factors) \
        .map(rename_bands) \
        .filterBounds(tile)

    tir = landsat_collection.select('TIR1').max().clip(tile)

    return tir
```

```
tir_tiles = [download_tir(tile, startDate, endDate) for tile in tiles_ee]
tir_mosaic = ee.ImageCollection(tir_tiles).mosaic()
```

2.2.2 Výstupy

```
ndvi_median_mosaic = ndvi_median_collection.mosaic().rename('NDVI_Median')
print("Mosaic median NDVI created.")
ndvi_max_mosaic = ndvi_max_collection.mosaic().rename('NDVI_Max')
print("Mosaic max NDVI created.")
ndbi_mosaic = ee.ImageCollection(ndbi_tiles).mosaic().rename('NDBI')
print("Mosaic NDBI created.")
mndwi_mosaic = ee.ImageCollection(mndwi_tiles).mosaic().rename('MNDWI')
print("Mosaic MNDWI created.")
ui_mosaic = ee.ImageCollection(ui_tiles).mosaic().rename('UI')
print("Mosaic UI created.")
isa_mosaic = ee.ImageCollection(isa_tiles).mosaic().rename('ISA')
print("Mosaic ISA created.")
tir_mosaic = ee.ImageCollection(tir_tiles).mosaic().rename('TIR_Max')
print("Mosaic TIR created.")
```

```
landsat_combined = ee.Image.cat([ndvi_median_mosaic.rename('NDVI_Median'),
                                 ndvi_max_mosaic.rename('NDVI_Max'),
                                 ndbi_mosaic.rename('NDBI'),
                                 mndwi_mosaic.rename('MNDWI'),
                                 ui_mosaic.rename('UI'),
                                 isa_mosaic.rename('ISA'),
                                 tir_mosaic.rename('TIR_Max')])
print("The combination of several images (frames) from different indexes into one multi-band image is complete.")
```

Příloha 1: Python skript – stažení satelitních dat.

```
landsat_samples = landsat_combined.sampleRegions(
        collection=Body1400_ee,  # Nactena bodova vrstva
        scale=30,
        geometries=True
)
print("Landsat 8 samples extracted.")
```

2.3 MODIS

```
def sample_lst(image):
    return image.sampleRegions(
        collection=Body1400_ee,
        scale=1000,
        geometries=True
    )
```

MODIS A1

```
modis = ee.ImageCollection('MODIS/061/MOD11A1').filter(ee.Filter.date(startDate, endDate))
modLSTday = modis.select('LST_Day_1km')

# prevedeni z Kelvinu na Celsia
modLSTk = modLSTday.map(lambda image: image.multiply(0.02).copyProperties(image, ['system:time_start']))

modisMax = modLSTk.max()
```

```
sampled_lst = modLSTk.map(sample_lst).flatten()
```

MODIS C3

```
modisMonth = ee.ImageCollection('MODIS/061/MOD21C3').filter(ee.Filter.date
  (startDate, endDate))
modLSTmonth = modisMonth.select('LST_Day')

# prevedeni z Kelvinu na Celsia
modLSTc2 = modLSTmonth.map(lambda image: image.multiply(0.02).subtract(273.15)
                                       .copyProperties(image, ['system:time_start']))

modisMean2 = modLSTc2.mean()
```

```
sampled_lst2 = modLSTc2.map(sample_lst).flatten()
```

Příloha 1: Python skript – stažení satelitních dat.

2.4 SENTINEL 2

2.4.1 Indexy

NDVI přes GEE

```python
def calculate_s2_ndvi(tile, startDate, endDate):
    sentinel_collection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
        .filterDate(startDate, endDate) \
        .map(rename_bands) \
        .filterBounds(tile)

    ndvi = sentinel_collection \
        .map(lambda image: image.normalizedDifference(['NIR', 'red']).rename('S2_NDVI')) \
        .median() \
        .clip(tile)

    return ndvi
```

NDBI přes GEE

```python
def calculate_s2_ndbi(tile, startDate, endDate):
    sentinel_collection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
        .filterDate(startDate, endDate) \
        .map(rename_bands) \
        .filterBounds(tile)

    ndbi = sentinel_collection \
        .map(lambda image: image.normalizedDifference(['SWIR1', 'NIR']).rename('S2_NDBI')) \
        .median() \
        .clip(tile)

    return ndbi

```

Příloha 1: Python skript – stažení satelitních dat.

MNDWI přes GEE

```python
def calculate_s2_mndwi(tile, startDate, endDate):
    sentinel_collection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
        .filterDate(startDate, endDate) \
        .map(rename_bands) \
        .filterBounds(tile)

    mndwi = sentinel_collection \
        .map(lambda image: image.normalizedDifference(['green', 'SWIR1']).rename('S2_MNDWI')) \
        .median() \
        .clip(tile)

    return mndwi

```

UI přes GEE

```python
def calculate_s2_ui(tile, startDate, endDate):
    sentinel_collection = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
        .filterDate(startDate, endDate) \
        .map(rename_bands) \
        .filterBounds(tile)

    ui = sentinel_collection \
        .map(lambda image: image.expression(
            '((NIR - (Red + Green)) / (NIR + (Red + Green)))',
            {
                'NIR': image.select('NIR'),
                'Red': image.select('red'),
                'Green': image.select('green')
            }).rename('S2_UI')
        ) \
        .median() \
        .clip(tile)

    return ui

```

Příloha 1: Python skript – stažení satelitních dat.

ISA přes GEE

```python
def calculate_s2_isa(tile, startDate, endDate):
    # Výpočet NDVI a NDBI
    ndvi = calculate_s2_ndvi(tile, startDate, endDate)
    ndbi = calculate_s2_ndbi(tile, startDate, endDate)

    # Výpočet ISA (rozdíl NDBI - NDVI)
    isa = ndbi.subtract(ndvi).rename('S2_ISA')

    return isa

```

2.4.2 Výstupy

```python
ndvi_tile = calculate_s2_ndvi(Body1400_ee, startDate, endDate)
ndbi_tile = calculate_s2_ndbi(Body1400_ee, startDate, endDate)
mndwi_tile = calculate_s2_mndwi(Body1400_ee, startDate, endDate)
ui_tile = calculate_s2_ui(Body1400_ee, startDate, endDate)
isa_tile = calculate_s2_isa(Body1400_ee, startDate, endDate)
```

```python
S2_combined = ee.Image.cat([ndvi_tile, ndbi_tile, mndwi_tile, ui_tile,
isa_tile])
print("Sentinel-2 indices combined into a multi-band image.")
```

```python
# vzorkova
s2_samples = S2_combined.sampleRegions(
    collection=Body1400_ee,
    scale=10,
    geometries=True
)
```

Příloha 1: Python skript – stažení satelitních dat.

2.5 SENTINEL 5P

Obecné

```python
try:
    CO = ee.ImageCollection('COPERNICUS/S5P/NRTI/L3_CO').filterDate(startDate, endDate).select("CO_column_number_density").mean().rename('CO')
    HCHO = ee.ImageCollection('COPERNICUS/S5P/NRTI/L3_HCHO').filterDate(startDate, endDate).select("tropospheric_HCHO_column_number_density").mean().rename('HCHO')
    NO2 = ee.ImageCollection('COPERNICUS/S5P/NRTI/L3_NO2').filterDate(startDate, endDate).select("tropospheric_NO2_column_number_density").mean().rename('NO2')
    SO2 = ee.ImageCollection('COPERNICUS/S5P/NRTI/L3_SO2').filterDate(startDate, endDate).select("SO2_column_number_density").mean().rename('SO2')
    CH4 = ee.ImageCollection('COPERNICUS/S5P/OFFL/L3_CH4').filterDate(startDate, endDate).select("CH4_column_volume_mixing_ratio_dry_air").mean().rename('CH4')
    O3 = ee.ImageCollection('COPERNICUS/S5P/NRTI/L3_O3').filterDate(startDate, endDate).select("O3_column_number_density").mean().rename('O3')

    print("Sentinel 5P layers prepared.")

except Exception as e:
    print(f"Error loading or processing Sentinel-5P ImageCollection: {e}")
```

```python
# Vzorkovani a prevod do DataFrame
results = []

for image, name in zip([CO, HCHO, NO2, SO2, CH4, O3], ['CO', 'HCHO', 'NO2', 'SO2', 'CH4', 'O3']):
    samples = image.sampleRegions(
        collection=Body1400_ee,
        scale=1000,
        geometries=True
    )
    task = ee.batch.Export.table.toDrive(
        collection=samples,
        description=f'Sentinel_5P_sample_export_{name}',
        folder='GEE_output',
        fileNamePrefix=f'sentinel5P_samples_{name}',
        fileFormat='CSV'
    )
    task.start()
    results.append((task, name))
```

Příloha 1: Python skript – stažení satelitních dat.

2.6 VIIRS

Světelné znečištění

Stažení dat z EOG

```
# celorovni prumer
#url = 'https://eogdata.mines.edu/nighttime_light/annual/v22/2022/VNL_v22_npp-j01_2022_global_vcmslcfg_c202303062300.average.dat.tif.gz'
```

VNL v2.2 pro rok 2022

Rozbalení dat

```
#response = requests.get(url, stream=True)
#if response.status_code != 200:
#    raise Exception(f"Failed to download file. Status code: {response.status_code}")
```

Práce s daty

```
tiff_path = os.path.splitext(gz_file)[0]
```

```
from osgeo import gdal
bbox = gdfCR.total_bounds # ohranicujici bbox
gdal.Warp('/content/drive4/MyDrive/DP/SVDNB_v10_11_2022_CR_vcmslcfg_avg.tif', tiff_path, outputBounds=bbox)
```

```
from rasterio.mask import mask
with rasterio.open(tiff_path) as src:
    # Extrahovani souradnic bodu
    coords = [(x, y) for x, y in zip(gdfBody.geometry.x, gdfBody.geometry.y)]
    values = [val[0] for val in src.sample(coords)]  # Hodnot rastru jednotlive body
    print(f"Extracted raster values: {values[:5]}")

# Přidání extrahovaných hodnot do původní bodové vrstvy
gdfBody['raster_value'] = values
print(gdfBody.head())
```

Příloha 2: R skript – práce s daty a tvorba modelu.

R skript

Celý skript v R je k dispozici ke stažení na GitHub.

Části kódu jsou uvedeny níže.

GitHub NoiseMap

Struktura kódu

DP_hluk - sloučení dat

Vstupní data

Načtení s práce s daty

```
```{R}
#| echo: false
landsat8 <- read.csv(file.path(input_dir, "landsat8/L8_2022_08/landsat8_samples.csv"))
modis <- read.csv(file.path(input_dir, "modis/modis_2022_08/modis_lst_day.csv"))
sentinel2 <- read.csv(file.path(input_dir, "sentinel2/S2_2022_08/sentinel2_samples.csv"))
sentinel5P_CO <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_CO.csv"))
sentinel5P_HCHO <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_HCHO.csv"))
sentinel5P_CH4 <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_CH4.csv"))
sentinel5P_NO2 <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_NO2.csv"))
sentinel5P_O3 <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_O3.csv"))
sentinel5P_SO2 <- read.csv(file.path(input_dir, "sentinel5P/S5P_2022_08/sentinel5P_samples_SO2.csv"))
viirs <- read.csv(file.path(input_dir, "viirs/viirs_2022_08/lightPollution_values.csv"))

message("Data uploaded successfully.")

```
```

Landsad 8

MODIS

Sentienl 2

Sentinel 5P

VIIRS

```
library(data.table)

dta0 <- as.data.table(dta0)  # prevedeni na dataFrame
dta <- copy(dta0[, setdiff(names(dta0), c("ID", "noiseLevel", "sourceLDEN")), with = FALSE])
dta[, LDEN := as.factor(LDEN)]
```

Příloha 2: R skript – práce s daty a tvorba modelu.

DP_hluk_02 - analýza a úprava dat

Vstupní data

Histogramy

Log-transformace

```
dta$VIIRS_log <- log(dta$VIIRS + 1)
dta$CO_log <- log(dta$CO + 1)
dta$CH4_log <- log(dta$CH4 + 1)
dta$HCHO_log <- log(dta$HCHO + 1)
dta$NO2_log <- log(dta$NO2 + 1)
```

Histogramy po transformaci (pouze nenulové hodnoty)

```
library(ggplot2)
library(tidyr)
library(patchwork)

numeric_variables <- c("L8_ISA", "S2_ISA", "L8_MNDWI", "S2_MNDWI",
                       "L8_NDBI", "S2_NDBI","L8_UI", "S2_UI",
                       "L8_NDVI_Median", "L8_NDVI_Max","S2_NDVI_Median",  "LST",
                       "CO", "CO_log","HCHO", "HCHO_log",
                       "CH4", "CH4_log", "NO2", "NO2_log",
                       "O3", "SO2", "VIIRS","VIIRS_log")

dta_long <- pivot_longer(dta, cols = all_of(numeric_variables),
                         names_to = "Variable", values_to = "Value")

variable_groups <- split(numeric_variables, ceiling(seq_along(numeric_variables) / 4))

histogram_list <- lapply(variable_groups, function(vars) {
  dta_subset <- dta_long[dta_long$Variable %in% vars, ]

  ggplot(dta_subset, aes(x = Value, fill = as.factor(LDEN))) +  #
    geom_histogram(bins = 30, fill = "green", color = "black", alpha = 0.7) +
    facet_wrap(~ Variable, scales = "free_x", ncol = 2) +  #
    labs(x = "Hodnota", y = "Frekvence", fill = "LDEN") +
    theme_minimal()
})

for (plot in histogram_list) {
  print(plot)
}
```

Příloha 2: R skript – práce s daty a tvorba modelu.

Boxplot

```
library(ggplot2)
library(tidyr)
library(patchwork)

numeric_variables <- c("L8_ISA", "S2_ISA", "L8_MNDWI", "S2_MNDWI",
                       "L8_NDBI", "S2_NDBI","L8_UI", "S2_UI",
                        "L8_NDVI_Median", "L8_NDVI_Max","S2_NDVI_Median",  "LST",
                       "CO", "CO_log","HCHO", "HCHO_log",
                       "CH4", "CH4_log", "NO2", "NO2_log",
                       "O3", "SO2", "VIIRS","VIIRS_log")

dta_long <- pivot_longer(dta, cols = all_of(numeric_variables),
                         names_to = "Variable", values_to = "Value")

variable_groups <- split(numeric_variables, ceiling(seq_along(numeric_variables) / 4))

plot_list <- lapply(variable_groups, function(vars) {
  dta_subset <- dta_long[dta_long$Variable %in% vars, ]

  ggplot(dta_subset, aes(x = as.factor(LDEN), y = Value)) +
    geom_boxplot(outlier.color = "red", fill = "lightblue") +
    facet_wrap(~ Variable, scales = "free_y", ncol = 2) +  # 2 sloupce
    labs(#title = "Boxploty proměnných vs LDEN",
         x = "LDEN (kategorie)", y = "Hodnota") +
    theme_minimal()
})

for (plot in plot_list) {
  print(plot)
}
```

Identifikace odlehlých hodnot

```
find_outliers <- function(data, column) {
  Q1 <- quantile(data[[column]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[column]], 0.75, na.rm = TRUE)
  IQR_value <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value

  # Výběr outlierů
  outliers <- data[data[[column]] < lower_bound | data[[column]] > upper_bound, ]

  return(outliers)
}

outliers_list <- lapply(names(dta)[sapply(dta, is.numeric)], function(col) find_outliers(dta, col))

names(outliers_list) <- names(dta)[sapply(dta, is.numeric)]
```

Příloha 2: R skript – práce s daty a tvorba modelu.

Grubbsův test

```
library(outliers)

apply_grubbs <- function(column) {
  if (length(na.omit(column)) > 2) {
    return(grubbs.test(column))
  } else {
    return(NA)
  }
}

grubbs_results <- lapply(dta[, sapply(dta, is.numeric), with = FALSE], apply_grubbs)
grubbs_results
```

Jitter plot

Uložení dta bez outliers

ANOVA vybraných proměnných

```
anova_vars <- c("L8_NDVI_Max", "S2_NDVI_Median", "L8_ISA", "S2_ISA", "CH4_log", "VIIRS")

# Spuštění ANOVA pro každou proměnnou
anova_results <- lapply(anova_vars, function(var) {
  model <- aov(dta_no_outliers[[var]] ~ as.factor(dta_no_outliers$LDEN), data = dta_no_outliers)
  summary(model)
})

# Výpis výsledků
names(anova_results) <- anova_vars
anova_results
```

Boxploty pro vybrané proměnné (bez odledlých hodnot)

Kontrola outliers

Uložení dat bez outliers

Odstranění outliers

Boxploty pro vybrané proměnné (bez odledlých hodnot)

Příloha 2: R skript – práce s daty a tvorba modelu.

Korelační matice

```r
if (!require(corrplot)) install.packages("corrplot")
library(corrplot)
library(data.table)

# Výběr proměnných pro analýzu
selected_vars <- dta_no_outliers[, .(L8_NDVI_Max, S2_ISA, VIIRS)]

# Výpočet korelační matice (Pearsonova a Spearmanova)
cor_matrix_pearson <- cor(selected_vars, method = "pearson", use = "complete.obs")
cor_matrix_spearman <- cor(selected_vars, method = "spearman", use = "complete.obs")

# Vykreslení korelační matice pomocí corrplot
par(mfrow = c(1, 2))  # Dvě grafické oblasti vedle sebe

# Pearsonova korelační matice
corrplot(cor_matrix_pearson, method = "color",
         #col = colorRampPalette(c("blue", "white", "red"))(200),
         addCoef.col = "black", # pridani hodnoty korelace
         tl.col = "black", tl.cex = 1.2, tl.srt = 45, # nastaveni popisku
         cl.pos = "b", # umisteni legendy
         title = "Pearsonova korelační matice", mar = c(0, 0, 2, 0))

# Spearmanova korelační matice
corrplot(cor_matrix_spearman, method = "color",
         #col = colorRampPalette(c("blue", "white", "red"))(200),
         addCoef.col = "black",
         tl.col = "black", tl.cex = 1.2, tl.srt = 45,
         cl.pos = "b",
         title = "Spearmanova korelační matice", mar = c(0, 0, 2, 0))
```

Uložení výstupních dat (upravených)

Příloha 2: R skript – práce s daty a tvorba modelu.

DP_hluk_03 - model RF

Vstupní data

Načtení dat

Selekce dat

Rozdělení dat

```
set.seed(123)
n <- nrow(dtaM)
train_index <- sample(1:n, 0.7 * n)  # 70 % train data
remaining_index <- setdiff(1:n, train_index)
valid_index <- sample(remaining_index, 0.5 * length(remaining_index))  # 15 % valid data
test_index <- setdiff(remaining_index, valid_index)  # 15 % test data

train_data <- dtaM[train_index]
valid_data <- dtaM[valid_index]
test_data <- dtaM[test_index]
```

Vyvážení tříd v datech

```
if (!require(smotefamily)) install.packages("smotefamily")
library(smotefamily)

train_data$LDEN <- as.factor(train_data$LDEN)

# Oddělení nezávislých proměnných (X) a cílové proměnné (target)
X <- train_data[, !names(train_data) %in% "LDEN", with = FALSE]
target <- train_data$LDEN

smote_data <- SMOTE(X, target, K = 5, dup_size = 5)

# Konverze výstupu na data.table a přejmenování
balanced_data <- as.data.table(smote_data$data)
names(balanced_data)[ncol(balanced_data)] <- "LDEN"

# Konverze LDEN zpět na faktor
balanced_data$LDEN <- as.factor(balanced_data$LDEN)

# Kontrola vyváženosti tříd po SMOTE
table(balanced_data$LDEN)
```

Model

Trénink modelu RF (před sloučením kategorií)

```
# Def RF model
rf_model <- randomForest(
  LDEN ~ .,
  data = balanced_data,
  mtry = 3,
  ntree = 300,
  nodesize = 15,
  importance = TRUE,
  na.action = na.omit
)

print(rf_model)
varImpPlot(rf_model)
```

Příloha 2: R skript – práce s daty a tvorba modelu.

```r
# Sloucení kategorií 70 dB až 90 dB do jedné
library(dplyr)

balanced_data <- balanced_data %>%
  mutate(LDEN = recode(LDEN,
                       "70" = "70+",
                       "75" = "70+",
                       "80" = "70+",
                       "85" = "70+",
                       "90" = "70+"))

# Ověření distribuce tříd po sloučení
table(balanced_data$LDEN)
```

Trénink modelu RF (po sloučení kategorií)

```r
rf_model <- randomForest(
  LDEN ~ .,
  data = balanced_data,
  mtry = 3,
  ntree = 500,
  nodesize = 15,
  importance = TRUE,
  na.action = na.omit
)

print(rf_model)
varImpPlot(rf_model)
```

Příloha 2: R skript – práce s daty a tvorba modelu.

Vyhodnocení modelu

```
# predikce test data
predicted <- predict(rf_model, test_data)
actual <- as.numeric(test_data$LDEN)

# predikce train data
train_predicted <- predict(rf_model, train_data)
train_actual <- as.numeric(train_data$LDEN)

# prevod na number
train_predicted <- as.numeric(train_predicted)
train_actual <- as.numeric(train_actual)
predicted <- as.numeric(predicted)
actual <- as.numeric(actual)
valid_predicted <- as.numeric(valid_predicted)
valid_actual <- as.numeric(valid_actual)

# vypocet metrik
train_rmse <- sqrt(mean((train_predicted - train_actual)^2))
train_mae <- mean(abs(train_predicted - train_actual))
train_rsq <- cor(train_predicted, train_actual)^2

cat("RMSE on Training Data:", train_rmse, "\n")
cat("MAE on Training Data:", train_mae, "\n")
cat("R-squared on Training Data:", train_rsq, "\n")

rmse <- sqrt(mean((predicted - actual)^2))   # Penalizuje velke chyby vice nez male
mae <- mean(abs(predicted - actual)) # Jednoduse meri prumer absolutni chyby
rsq <- cor(predicted, actual)^2    # Udava kolik variability promenné LDE je vysvetleno

cat("RMSE on Test Data:", rmse, "\n")
cat("MAE on Test Data:", mae, "\n")
cat("R-squared on Test Data:", rsq, "\n")

valid_predicted <- as.numeric(predict(rf_model, valid_data))
valid_actual <- as.numeric(valid_data$LDEN)

valid_rmse <- sqrt(mean((valid_predicted - valid_actual)^2))
valid_mae <- mean(abs(valid_predicted - valid_actual))
valid_rsq <- cor(valid_predicted, valid_actual)^2

cat("RMSE on Validation Data:", valid_rmse, "\n")
cat("MAE on Validation Data:", valid_mae, "\n")
cat("R-squared on Validation Data:", valid_rsq, "\n")
```

Příloha 2: R skript – práce s daty a tvorba modelu.

DP_hluk_04 - model GB

Gradient Boosting Model

Načtení dat

Rozdělení dat

Optimalizace hyperparametrů Gradient Boosting

```
control <- trainControl(method = "cv", number = 5)

# Grid hledani hyperparametru
tune_grid <- expand.grid(
  n.trees = c(100, 200, 500, 1000, 2000, 5000),
  interaction.depth = c(1, 3, 5, 7),
  shrinkage = c(0.01, 0.05, 0.1),
  n.minobsinnode = c(10, 20, 30)
)
```

Trénink modelu Gradient Boosting s optimalizovanými parametry

```
# trenovani - optimalni parametry
set.seed(123)
gbm_model <- gbm(
  formula = LDEN ~ .,
  data = train_data,
  distribution = "gaussian",
  n.trees = gbm_tuned$bestTune$n.trees,
  interaction.depth = gbm_tuned$bestTune$interaction.depth,
  shrinkage = gbm_tuned$bestTune$shrinkage,
  n.minobsinnode = gbm_tuned$bestTune$n.minobsinnode,
  verbose = FALSE
)
```

Důležitost proměnných

```
library(ggplot2)

# Získání důležitosti proměnných
importance_df <- summary(gbm_model)

# Přejmenování sloupců pro lepší přehlednost
colnames(importance_df) <- c("Variable", "Importance")

# Vykreslení grafu

custom_colors <- c("#bcbd22","#17becf", "#2ca02c", "#d62728", "#9467bd","#ff7f0e")

ggplot(importance_df, aes(x = reorder(Variable, Importance), y = Importance, fill = Variable)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = round(Importance, 1)),  # Přidání hodnot
            hjust = 1, size = 5) +
  coord_flip() +
  scale_fill_manual(values = custom_colors) +
  theme_minimal() +
  labs(title = "Relativní důležitost pro metodu Gradient Boosting",
       x = "Proměnná", y = "Relativní vliv (%)") +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))
```

Příloha 2: R skript – práce s daty a tvorba modelu.

Trénink modelu na upravených parametrech

```r
# trenovani - upravené parametry
set.seed(123)
gbm_model <- gbm(
  formula = LDEN ~ .,
  data = train_data,
  distribution = "gaussian",
  n.trees = 1000,
  interaction.depth = 5,
  shrinkage = 0.05,
  n.minobsinnode = 50,
  verbose = FALSE
)
```

Vyhodnocení modelu

```r
test_predicted <- predict(gbm_model_updated, test_data, n.trees = gbm_tuned$bestTune$n.trees)
test_actual <- as.numeric(test_data$LDEN)

train_predicted <- predict(gbm_model_updated, train_data, n.trees = gbm_tuned$bestTune$n.trees)
train_actual <- as.numeric(train_data$LDEN)

valid_predicted <- predict(gbm_model_updated, valid_data, n.trees = gbm_tuned$bestTune$n.trees)
valid_actual <- as.numeric(valid_data$LDEN)


test_predicted <- as.numeric(test_predicted)
train_predicted <- as.numeric(train_predicted)
valid_predicted <- as.numeric(valid_predicted)


train_rmse <- sqrt(mean((train_predicted - train_actual)^2))
train_mae <- mean(abs(train_predicted - train_actual))
train_rsq <- cor(train_predicted, train_actual)^2

cat("RMSE na trénovacích datech:", train_rmse, "\n")
cat("MAE na trénovacích datech:", train_mae, "\n")
cat("R-squared na trénovacích datech:", train_rsq, "\n\n")

test_rmse <- sqrt(mean((test_predicted - test_actual)^2))
test_mae <- mean(abs(test_predicted - test_actual))
test_rsq <- cor(test_predicted, test_actual)^2

cat("RMSE na testovacích datech:", test_rmse, "\n")
cat("MAE na testovacích datech:", test_mae, "\n")
cat("R-squared na testovacích datech:", test_rsq, "\n\n")

valid_rmse <- sqrt(mean((valid_predicted - valid_actual)^2))
valid_mae <- mean(abs(valid_predicted - valid_actual))
valid_rsq <- cor(valid_predicted, valid_actual)^2

cat("RMSE na validačních datech:", valid_rmse, "\n")
cat("MAE na validačních datech:", valid_mae, "\n")
cat("R-squared na validačních datech:", valid_rsq, "\n")
```

Příloha 2: R skript – práce s daty a tvorba modelu.

DP_hluk_05 - model Neuronové sítě

Neuronové sítě pro predikci LDEN

Načtení dat

Normalizace dat

```
# min/max pro denormalizaci
min_ldn <- min(dtaM$LDEN)
max_ldn <- max(dtaM$LDEN)

# Normalizace dat (0-1 škála)
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
dtaM_norm <- as.data.frame(lapply(dtaM, normalize))
```

Rozdělení vstupních dat

Optimalizace hyperparametrů

```
hidden_layers <- list(c(5, 3), c(10, 5), c(20, 10, 5), c(10, 5, 3), c(20, 10, 5), c(30, 15, 5))

# Funkce na vyhodnocení neuronové sítě
evaluate_nn <- function(hidden_structure) {
  set.seed(123)
  nn_model <- neuralnet(
    LDEN ~ .,
    data = train_data,
    hidden = hidden_structure,
    linear.output = TRUE,
    stepmax = 1e6
  )

  test_predicted <- compute(nn_model, test_data[, setdiff(names(test_data), "LDEN")])$net.result

  # Oprava denormalizace
  denormalize <- function(x, min_val, max_val) {
    return(x * (max_val - min_val) + min_val)
  }

  test_predicted <- denormalize(test_predicted, min_ldn, max_ldn)
  actual <- denormalize(test_data$LDEN, min_ldn, max_ldn)  # Oprava denormalizace skutečných hodnot!

  # Výpočet metrik
  rmse <- sqrt(mean((test_predicted - actual)^2))
  mae <- mean(abs(test_predicted - actual))
  rsq <- cor(test_predicted, actual)^2

  return(list(model = nn_model, rmse = rmse, mae = mae, rsq = rsq))
}

# Otestování různých struktur sítě
results <- lapply(hidden_layers, evaluate_nn)

# Výběr nejlepšího modelu
best_model_index <- which.min(sapply(results, function(x) x$rmse))
best_model <- results[[best_model_index]]$model

cat("Nejlepší struktura neuronové sítě:", hidden_layers[[best_model_index]], "\n")
cat("RMSE nejlepšího modelu:", results[[best_model_index]]$rmse, "\n")
```

Příloha 2: R skript – práce s daty a tvorba modelu.

Vyhodnocení modelu

```
train_predicted <- compute(best_model, train_data[, setdiff(names(train_data), "LDEN")])$net.result
test_predicted <- compute(best_model, test_data[, setdiff(names(test_data), "LDEN")])$net.result
valid_predicted <- compute(best_model, valid_data[, setdiff(names(valid_data), "LDEN")])$net.result

train_predicted <- denormalize(train_predicted, min_ldn, max_ldn)
test_predicted <- denormalize(test_predicted, min_ldn, max_ldn)
valid_predicted <- denormalize(valid_predicted, min_ldn, max_ldn)

train_actual <- denormalize(train_data$LDEN, min_ldn, max_ldn)
test_actual <- denormalize(test_data$LDEN, min_ldn, max_ldn)
valid_actual <- denormalize(valid_data$LDEN, min_ldn, max_ldn)

# výpočet metrik
train_rmse <- sqrt(mean((train_predicted - train_actual)^2))
train_mae <- mean(abs(train_predicted - train_actual))
train_rsq <- cor(train_predicted, train_actual)^2

cat("RMSE na trénovacích datech:", train_rmse, "\n")
cat("MAE na trénovacích datech:", train_mae, "\n")
cat("R-squared na trénovacích datech:", train_rsq, "\n\n")

test_rmse <- sqrt(mean((test_predicted - test_actual)^2))
test_mae <- mean(abs(test_predicted - test_actual))
test_rsq <- cor(test_predicted, test_actual)^2

cat("RMSE na testovacích datech:", test_rmse, "\n")
cat("MAE na testovacích datech:", test_mae, "\n")
cat("R-squared na testovacích datech:", test_rsq, "\n\n")

valid_rmse <- sqrt(mean((valid_predicted - valid_actual)^2))
valid_mae <- mean(abs(valid_predicted - valid_actual))
valid_rsq <- cor(valid_predicted, valid_actual)^2

cat("RMSE na validačních datech:", valid_rmse, "\n")
cat("MAE na validačních datech:", valid_mae, "\n")
cat("R-squared na validačních datech:", valid_rsq, "\n")
```

Kontrola rozsahu predikce