PART 1

Please submit a report to answer the questions in part 1. The report's format (Python output screenshots with one- or two-line explanation) must be a pdf file. Moreover, you need to submit your Python code (Jupyter notebook). Zip your report and Python code (Jupyter notebook) as a single zip file titled 'Part 1'.

Why are those best and most experienced employees leaving prematurely? Have fun with data in "*HR_data.xlsx*" and explore the **possible causes of people leaving the company**. Variables in the dataset include:

- Employee satisfaction level
- Last evaluation
- Number of projects
- Average monthly hours
- Time spent at the company
- Whether they have had a work accident
- Whether they have had a promotion in the last 5 years
- Department
- Salary
- Whether the employee has left

Write a summary report about your analysis result (3-4 charts with analysis should be OK). The report should contain the following: Your findings on the possible causes of employee turnover, supported by data and charts.

PART 2

You do not need to submit a report to answer the questions in part 2. Just submit your jupyter notebook with your code and answers.

The data set is real data structure used in the project. In the input folder, there are 4 files train/test/params/dates:
· train.csv and test.csv are training and test data set respectively
· params.rds installed constant parameters used in further analysis
· dates.rds are the the manufacturing dates of machines.

Train data is 2188 * 23, Test data is 30 * 22, each row represents 1 machine.
There are 21 variables and 1 machine id, called: serial_longurl, in both train and test.
There is 1 additional target column (y) in train data, called: obpf, and it is continuous.

This part of the test consists of 2 parts:
- Basic Data preparation
- Load and run XGBoost model

####### **Q1.** #######

- Load Data from input folder.

####### **Q2.** #######
In all columns, there is c1_xx, c2_xx, c3_xx, c4_xx.
But we only need mean value of such columns.

- Create new Column "c_fa" to compute mean of columns: "c1_fa", "c2_fa", "c3_fa", "c4_fa"
- Create new Column "c_fr" to compute mean of columns: "c1_fr", "c2_fr", "c3_fr", "c4_fr"
- Create new Column "c_imp" to compute mean of columns: "c1_imp", "c2_imp", "c3_imp", "c4_imp"
- Then remove all columns like c1_xx, c2_xx, c3_xx, c4_xx

####### **Q3.** #######
Not all variables are used in the model. Below is a list of selected columns. Additionally, we split x and y for both train and test (y for test is not available)

```
imp_cols = ("mounted_hf_fa_fr", "mounted_hf_imp", "mounted_hf_res_fr", "mounted_lf_fa_fr",
"mounted_lf_imp", "mounted_lf_res_fr","initial_wedge_height", "vout", "wedge_stroke", "c_fa",
"c_fr", "c_imp")
```

- Select only above columns from train_data as train_x and convert to matrix format
- Select "obpf" column from train_data as train_y
- Select only above columns from test_data as test_x

####### **Q4.** #######
In XGBoost training, contains 3 parts:

- train_control: K-folds cross validation are used
- model_grid: To store value of hyperparameters. The input can be a vector, and it will iterate through all values in that vector. That is usually done in the model tuning stage.
- xgb_model: To train the XGBoost model.

Since our y is continuous, we use RMSE as loss function.

```
train_control <- caret::trainControl(method = "repeatedcv", number = 3, repeats = 4, verboseIter =
TRUE, allowParallel = TRUE)

model_grid <- expand.grid(nrounds = 75, max_depth = 3, eta = 0.08, gamma = 0.1, colsample_bytree =
0.6, min_child_weight = 5, subsample = 0.8)

xgb_model <- caret::train( x = train_x, y = train_y, trControl = train_control, tuneGrid = model_grid,
method = "xgbTree", verbose = T, metric = "RMSE", maximize = F)
```

Now we have XGBoost Model stored in an object called: xgb_model. Next, we will predict on test data.

- Use predict() to form predictions on test data
- Assign the predicted values to a new column in test_data, called: obpf_pred

####### **Q5.** #######
We consider the test data as a mini batch. And we want to know the batch properties, measured in % of machine's predicted value above or below certain value.

Recall we have a data set called: params, we want to check % of machines if:
1. predicted value > up_guard_band (value stored in params)
2. predicted value < lw_guard_band (value stored in params)

- Compute the percent of machines that satisfy 1, and
- Compute the percent of machines that satisfy 2.