Analysis of Fitness Data **Project Proposal**

Created by:

MTE

Project Description

With the explosion in fitness tracker popularity, runners all of the world are collecting data with gadgets (smartphones, watches, etc.) to keep themselves motivated. They look for answers to questions like:

- How fast, long, and intense was my run today?
- Have I succeeded with my training goals?
- Am I progressing?
- What were my best achievements?
- How do I perform compared to others?

This data was exported from Runkeeper. The data is a CSV file where each row is a single training activity. In this project, you'll create import, clean, and analyze my data to answer the above questions. You can then apply the same strategy to your training data if you wish!

Project Tasks

- 1. Obtain and review raw data
- 2. Data preprocessing
- 3. Dealing with missing values
- 4. Plot running data
- 5. Running statistics
- 6. Visualization with averages
- 7. Did I reach my goals?
- 8. Am I progressing?
- 9. Training intensity
- 10. Detailed summary report

- 11. Fun facts

Task 1: Instructions

Load `pandas` and the training activities data.

- Import `pandas` under the alias `pd`.
- Use the `read_csv()` function to load the dataset (`runkeeper_file`) into a variable called `df_activities`. Parse the dates with the `parse_dates` parameter and set the index to the `Date` column using the `index_col` parameter.
- Display 3 random rows from `df_activities` using the `sample()` method.
- Print a summary of `df_activities` using the `info()` method.

Task 2: Instructions

Implement the following data preprocessing tasks:

- Delete unnecessary columns from `df_activities` with the `drop()` method, setting the `columns` parameter to the `cols_to_drop` list.
- Calculate the activity type counts using the `value_counts()` method on the `Type` column.
- Rename the 'Other' values to 'Unicycling' in the `Type` column using `str.replace()`.
- Count the missing values in each column using `isnull().sum()`.

Task 3: Instructions

Implement mean imputation for missing values.

- Calculate the sample mean for `Average Heart Rate (bpm)` for the 'Cycling' activity type. Assign the result to `avg_hr_cycle`.
- Filter the `df_activities` for the 'Cycling' activity type. Create a copy of the result using `copy()` and assign the copy to `df_cycle`.
- Fill in the missing values for `Average Heart Rate (bpm)` in `df_cycle` with `int(avg_hr_cycle)` using the `fillna()` method.
- Count the missing values for all columns in `df_run`.

Task 4: Instructions

Plot running data from 2013 through 2018.

- Subset `df_run` for data from 2013 through 2018. Take into account that observations in dataset stored in chronological order - most recent records first. Assign the result to `runs_subset_2013_2018`.
- In the plotting code, enable subplots by setting the `subplots` parameter to `True`. Don't use spaces around the `=` sign when used to indicate a keyword argument, as recommended in PEP 8 style guide for Python code.
- Show the plot using `plt.show()`.

Task 5: Instructions

Calculate annual and weekly means for `Distance (km)`, `Average Speed (km/h)`, `Climb (m)` and `Average Heart Rate (bpm)`.

- Subset `df_run` for data from 2015 through 2018. Assign the result to `runs_subset_2015_2018`.
- Count the annual averages using `resample()` with 'A' alias, and the `mean()` method for `runs_subset_2015_2018`.
- Count the average weekly statistics using `resample()` with 'W' alias, and the `mean()` method twice.
- Filter from dataset column `Distance (km)` and count the average number of trainings per week using `resample()` with the `count()` and `mean()` methods. Assign the result to `weekly_counts_average`.

Task 6: Instructions

Prepare data and create a plot.

- Select information for distance and then for heart rate from `runs_subset_2015_2018` and assign to `runs_distance` and `runs_hr`, respectively.
- Create two subplots with shared x-axis using the `plt.subplots()` method, setting the first positional parameter to 2, `sharex` to `True`, and `figsize` to `(12,8)`. Assign the output to `fig, (ax1, ax2)` variables.
- Plot distance on the first subplot, setting parameter `ax` to `ax1`.
- On the second subplot (`ax2`), add a horizontal line with `axhline()` for the average value of heart rate counted as `runs_hr.mean()`. Set `color` to `'blue'`, `linewidth` to `1`, and `linestyle` to `'-.'`.

Task 7: Instructions

Prepare data and create a plot.

- Subset `df_run` for data from 2013 through 2018 and select the `Distance (km)` column. Count annual totals with `resample()` and `sum()`. Assign the result to `df_run_dist_annual`.
- Create a plot with `plt.figure()`, setting `figsize` to define a plot of size 8.0 inches x 5.0 inches.
- Customize the plot with horizontal span from 0 to 800 km with `ax.axhspan()`. Set `color` to `'red'` and `alpha` to `0.2`.
- Show the plot with `plt.show()`.

Task 8: Instructions

Create a plot with observed distance of runs and decomposed trend.

- Import the `statsmodels.api` under the alias `sm`.
- Subset `df_run` from 2013 through 2018, select `Distance (km)` column, resample weekly, and fill NaN values with the `bfill()` method. Assign to `df_run_dist_wkly`.
- Create a plot with `plt.figure()`, defining plot size by setting `figsize` to `(12,5)`.

Task 9: Instructions

Create a customized histogram for heart rate distribution.

- Subset `df_run` from March 2015 through 2018 then select the `Average Heart Rate (bpm)` column. Assign the result to `df_run_hr_all`.
- Create a plot with `plt.subplots()`, setting `figsize` to `(8,5)`. Assign the result to `fig, ax`.
- Create customized x-axis ticks with `ax.set_xticklabels()`. Set the parameters `labels` to `zone_names`, `rotation` to `-30`, and `ha` to `'left'`.
- Show the plot with `plt.show()`.

Task 10: Instructions

Create a summary report.

- Concatenate the `df_run` DataFrame with `df_walk` and `df_cycle` using `append()`, then sort based on the index in descending order. Assign the result to `df_run_walk_cycle`.
- Group `df_run_walk_cycle` by activity type, then select the columns in `dist_climb_cols`. Sum the result using `sum()`. Assign the result to `df_totals`.
- Use the `stack()` method on `df_summary` to show a compact reshaped form of the full summary report.

Task 11: Instructions

Use FUN FACTS data to answer some fun questions.

- Calculate the instructor's average shoes per lifetime. Use number of 'Total number of km run' from FUN FACTS and divide by the number of pairs of shoes gone through.
- Calculate an estimated number of shoes gone through for Forrest Gump's route. Use 'Total number of km run' from FORREST RUN FACTS, then divide (using floor division) by the result from the previous step.

Dataset and Jupiter Notebook Files - Download from link below

https://drive.google.com/uc?export=download&id=1O--TsE3O2orEDieV7tU2pp0ndMTYekQB