Fitbit Project Report

Contents

[GitHub URL 1](#_Toc146211908)

[Abstract 1](#_Toc146211909)

[Introduction 1](#_Toc146211910)

[Dataset 1](#_Toc146211911)

[Implementation Process 1](#_Toc146211912)

[Importing 1](#_Toc146211913)

[Preparing, cleaning data 2](#_Toc146211914)

[Feature Engineering. 2](#_Toc146211915)

[Time Series Analysis: Import, Clean and Preprocessing. 2](#_Toc146211916)

[Time Series Analysis: Splitting the Data 2](#_Toc146211917)

[Time Series Analysis: Hyperparameter Tuning 2](#_Toc146211918)

[Time Series Analysis: Train and Predict 2](#_Toc146211919)

[Regression Analysis: Import, Clean, Preprocessing, Train and Predict. 2](#_Toc146211920)

[Regression Analysis: Coefficients, Hyperparameter Tuning, Train and Predict (again). 3](#_Toc146211921)

[Regression Analysis: Feature selection based on statistical significance. 3](#_Toc146211922)

[Analysis and Results 3](#_Toc146211923)

[Conclusions 3](#_Toc146211924)

[What could be added to this project? 3](#_Toc146211925)

# GitHub URL

[Crowmium1/Fitbit-Project: customer segmentation analysis with 7 user data (github.com)](https://github.com/Crowmium1/Fitbit-Project)

# Abstract

This project takes Fitbit user data, cleans it and transforms it into feature columns. Descriptive and inferential statistical analysis is done on this data. Functions (for trending and plotting data) inside classes in a separate module were imported over to the main file. Customer segmentation analysis was done on the data. Time series analysis was also then done on the data, where forecasted results were predicted and evaluated.

# Introduction

# Dataset

The dataset accepts two excel files from the ‘fitbit fitness tracker database’; dailyActivity\_merged and sleepDay\_merged. These excel files are loaded, cleaned and transformed in MySQL. A new excel file is created and put into the project folder for further use in VS code.

# Implementation Process

## Importing

The dataset accepts two excel files from the ‘fitbit fitness tracker database’; dailyActivity\_merged and sleepDay\_merged. These excel files are loaded, cleaned and transformed in MySQL. A new excel file is created and put into the project folder for further use in VS code. There are 7 unique user IDs in both these excel files, while 5 user IDs are shared.

The excel file was read into VS Code and made into a Dataframe using Pandas.

## Preparing, cleaning data

The relevant libraries and modules are imported.

The Dataframe was cleaned and feature variables were created. Feature labels were converted into their correct data types. The new Dataframe was grouped and sorted by ActivityDate and UserID. The Dataframe includes shared information only.

## Feature Engineering.

The ‘data analyzer’ module was instantiated as an object used to create plots for comparing feature labels.

A function for classifying the time sedentary into categories was created as part of the customer segmentation analysis. Another function for classifying the number of steps taken by categories was created.

A new aggregate Dataframe was created which combined the ‘Mean Steps’, ‘Calories’ and ‘Sleep’ by ‘DayOfWeek’. The ‘activity category’ and ‘sedentary category’ functions were applied to this Dataframe and made into feature labels.

## Time Series Analysis: Import, Clean and Preprocessing.

The relevant libraries and modules are imported.

The data is resampled and plotting as calories burnt over time.

A time series decomposition model is instantiated for Calories and plotted over time.

Lag features were created for the past 7 days. These lag features capture past information and are used to improve the performance of time series models.

## Time Series Analysis: Splitting the Data

The data is split into training and test sets.

The Sarima model is instantiated.

A function was created for evaluating the Sarima model based on root mean squared error is instantiated.

## Time Series Analysis: Hyperparameter Tuning

The SARIMA hyperparameter grid were defined;

* **p**: The number of lag observations included in the model, also called the lag order. This is the AR (Autoregressive) part.
* **d**: The number of times that the raw observations are differenced, also known as the degree of differencing. This is the I (Integrated) part.
* **q**: The size of the moving average window, also known as the order of the moving average. This is the MA (Moving Average) part.

The **order\_grid** and **seasonal\_order\_grid** specify the configurations that will be tried out for both non-seasonal and seasonal components of a SARIMA model, and **best\_rmse**, **best\_order**, and **best\_seasonal\_order** are variables meant to store the best model's performance and configuration.

This code manually loops through the hyperparameter combinations and selects the best SARIMA model based on RMSE. You can't set hyperparameters for GridSearchCV with SARIMA as statsmodels doesn't have a set\_params method like some other scikit-learn estimators. Model Interpretability is not applicable for SARIMA.

## Time Series Analysis: Train and Predict

These hyperparameters are used to train the Sarima model. It then evaluates the model on the test data and calculates performance metrics (mean absolute error, R^2 and root mean squared error).

SARIMA models don't have feature importance’s, so that section is not applicable.

## Regression Analysis: Import, Clean, Preprocessing, Train and Predict.

Import relevant libraries and modules.

Filter User Data by ID for the aggregated Dataframe.

Define the feature and target variables.

Split the data.

Create a pipeline with preprocessing and modelling steps, where the missing data is imputed, the data is scaled, and a linear regression model is applied to the data. Fit the pipeline on the training data.

Predict on the test data and calculate performance metrics.

## Regression Analysis: Coefficients, Hyperparameter Tuning, Train and Predict (again).

Get coefficients from the Linear Regression model. Get the feature names from the column transformer. Create a DataFrame of the features and coefficients. Print the Dataframe and plot these coefficients.

Hyperparameter Tuning: Define a parameter grid for tuning. Instantiate the grid search CV and fit to the training and test data. Get the best estimator and best parameter values using the results.

Predict on the test data and calculate performance metrics using the best estimator variable.

Predict the target variable using the feature matrix and the trained regression model as **pred\_y**. Calculate residuals by subtracting the predicted values from the actual values **y**.

## Regression Analysis: Feature selection based on statistical significance.

Using statsmodels get the p-values for each feature (and the intercept) from the fitted model and print these results.

P-values are used to test the null hypothesis that each coefficient (corresponding to each feature and the intercept) has no effect on the target variable **y**. A small p-value (commonly below 0.05) indicates that you can reject the null hypothesis, implying that the feature is statistically significant in predicting the target variable.

* If the p-value for a feature is small (e.g., < 0.05), the feature is usually considered to be statistically significant.
* If the p-value is large (e.g., > 0.05), it suggests that the feature may not be a good predictor for the target variable within the given dataset.

# Analysis and Results

# Conclusions

# What could be added to this project?

1. Additional Feature Engineering: Rolling Statistics
2. Exploratory Data Analysis (EDA): Visualize autocorrelation plot
3. Visualize autocorrelation plot
4. Model Selection: Try an Exponential Smoothing model (Holt-Winters)
5. Hyperparameter Tuning: For Exponential Smoothing, there are alpha, beta, and gamma parameters that can be tuned
6. Ensemble Models: Combine SARIMA and Exponential Smoothing forecasts
7. Cross-Validation: Use Walk-Forward Validation for time series data
8. Model Evaluation: Calculate additional metrics like MAPE and SMAPE
9. Forecast Visualization: Plot the forecasted values with confidence intervals
10. Model Monitoring and Maintenance: Implement a retraining schedule
11. Error Analysis: Analyze model errors to identify patterns and improve the models
12. Advanced Techniques: Explore Bayesian structural time series (BSTS) or deep learning models
13. Alerting and Reporting: Implement alerts or reporting mechanisms