University of Padova

Department of Management Engineering
Dipartimento di Tecnica e Gestione dei sistemi industriali

Sustainable Mobility

Supervisors:
Marta Disegna

Master Candidate:
Nima Karimi

Summer 2023

Nima Karimi
Sustainable Mobility
Documentation, September 05, 2023
Supervisors: Marta Disegna

University of Padova
Dipartimento di Tecnica e Gestione dei sistemi industriali
Department of Management Engineering
Padova

Abstract

This thesis presents a comprehensive study on developing a machine learning model to predict mobility patterns in various weather conditions and group structures. The primary focus of this project is to assess the effectiveness of machine learning models in forecasting mobility behavior, considering different weather scenarios, social contexts, and route characteristics. Additionally, the research aims to identify the key factors influencing mobility behavior and explore the potential application of machine learning models in personalized mobility prediction and sustainable transportation planning.

To achieve these objectives, a range of machine learning models will be evaluated for their accuracy and precision in forecasting mobility patterns. The research will investigate how different models perform in predicting mobility behavior under varying weather conditions and social contexts. Moreover, an in-depth analysis will be conducted to identify the factors that significantly impact the accuracy of these models.

By examining the accuracy and precision of various machine learning models, this research will provide valuable insights into their effectiveness in mobility forecasting. Furthermore, it will uncover the factors that play a crucial role in influencing the accuracy of these models. The findings of this study will contribute to the advancement of machine learning applications in transportation planning and assist in developing personalized mobility prediction systems for sustainable transportation.

Keywords: machine learning, mobility patterns, weather conditions, group structures, accuracy, precision, personalized mobility prediction, sustainable transportation planning

Astratto

Questa tesi presenta uno studio approfondito per lo sviluppo di un modello di apprendimento automatico per prevedere i modelli di mobilità in diverse condizioni meteorologiche e strutture di gruppo. Il focus principale di questo progetto è

valutare l'efficacia dei modelli di apprendimento automatico nella previsione del comportamento di mobilità, considerando diversi scenari meteorologici, contesti sociali e caratteristiche del percorso. Inoltre, la ricerca mira a identificare i fattori che influenzano il comportamento di mobilità ed esplorare la possibilità di utilizzare i modelli di apprendimento automatico per la previsione della mobilità personalizzata e la pianificazione del trasporto sostenibile.

Per raggiungere questi obiettivi, verranno valutati diversi modelli di apprendimento automatico per la loro accuratezza e precisione nella previsione dei modelli di mobilità. La ricerca indagherà su come diversi modelli si comportano nella previsione del comportamento di mobilità in diverse condizioni meteorologiche e contesti sociali. Inoltre, verrà condotta un'analisi approfondita per identificare i fattori che influenzano in modo significativo l'accuratezza di questi modelli.

Attraverso l'esame dell'accuratezza e della precisione dei vari modelli di apprendimento automatico, questa ricerca fornirà preziose intuizioni sulla loro efficacia nella previsione della mobilità. Inoltre, svelerà i fattori che giocano un ruolo cruciale nell'influenzare l'accuratezza di questi modelli. I risultati di questo studio contribuiranno all'avanzamento delle applicazioni di apprendimento automatico nella pianificazione dei trasporti e aiuteranno nello sviluppo di sistemi di previsione della mobilità personalizzata per il trasporto sostenibile.

Parole chiave: apprendimento automatico, modelli di mobilità, condizioni meteorologiche, strutture di gruppo, accuratezza, precisione, previsione della mobilità personalizzata, pianificazione del trasporto sostenibile.

Acknowledgement

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction

With the rapid expansion of cities and the escalating worries regarding the sustainable use of our natural resources, the quest for achieving transportation systems that are not only efficient but also eco-friendly has garnered significant attention and become an urgent global challenge. Sustainable mobility, often referred to as sustainable transportation, aims to develop and implement strategies that promote the efficient movement of people and goods while minimizing negative environmental and social impacts.

Furthermore, machine learning, a field of artificial intelligence, involves the development and application of computational models that can automatically learn and improve from data without explicit programming. By analyzing large and diverse datasets, machine learning algorithms can identify patterns, make predictions, and gain valuable insights that aid decision-making processes. In the context of sustainable mobility, machine learning can provide innovative solutions to optimize transportation systems, improve accessibility, and minimize environmental impact.

Optimizing transportation systems through machine learning involves the application of data analytics to enhance operational efficiency, reduce congestion, and improve the overall performance of transportation networks. By processing large volumes of data collected from sources such as traffic sensors, GPS devices, and social media platforms, machine learning algorithms can generate real-time traffic predictions, optimize routing, and dynamically adapt transportation services. These advancements lead to improved travel experiences, reduced travel times, and more efficient allocation of resources.

Accessibility is a key aspect of sustainable mobility, aiming to ensure that transportation services are available and equitable for all individuals, regardless of their physical abilities, income levels, or geographic location. Machine learning algorithms can help address accessibility challenges by analyzing data on travel demand, demographics, and infrastructure characteristics. This enables the identification of underserved areas, optimization of transit routes, and the design of transportation services that cater to the needs of diverse populations.

Environmental sustainability is another critical dimension of sustainable mobility. Transportation accounts for a significant portion of global greenhouse gas emissions and is a major contributor to air pollution. Machine learning can play a crucial role in minimizing the environmental impact of transportation by developing predictive models that estimate emissions, optimize energy consumption, and facilitate the integration of clean and renewable energy sources. Additionally, machine learning techniques can aid in the design of eco-routing algorithms, which suggest the most environmentally friendly travel routes based on real-time data.

In summary, the combination of machine learning and sustainable mobility presents a promising framework to address the complex challenges faced by transportation systems. By harnessing the power of machine learning algorithms, transportation stakeholders can optimize operations, improve accessibility, and mitigate environmental impact. As the demand for sustainable and efficient mobility solutions continues to grow, machine learning can provide valuable insights and tools to shape the future of transportation, leading to more sustainable, accessible, and environmentally friendly mobility systems.

1.1 Machine Learning

Can computers learn like humans do from experience? Yes, through machine learning (ML). It enhances system performance by using computational methods to learn from experience, mainly in the form of data. The key task in machine learning is developing algorithms that build models from data. By inputting experience data into these algorithms, we obtain models capable of predicting outcomes for new observations. In the context of computer science, machine learning focuses on learning algorithms [1].

[2] provides a more formal definition: “A computer program is said to learn from experience E for some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“ The term "model" is broadly used to denote outcomes learned from data. In some literature (e.g. [3]), "model" might mean a global outcome, while "pattern" signifies a local outcome.

The procedure of constructing models from data using machine learning algorithms is referred to as learning or training. The data employed during this phase is known as training data, where each sample serves as a training example, and the complete collection of training examples is the training set. The model derived from this process, representing the underlying data rules, is also termed a hypothesis, while the actual underlying rules are considered as facts or ground-truth. Thus, the fundamental goal of machine learning is to identify or approximate the ground-truth rules [1].

A wide range of machine-learning algorithms has been developed to address the diverse data and problem types in various machine-learning scenarios [4, 5]. These algorithms can be understood as exploring a vast space of potential programs, guided by training experience, to discover the most optimal performance. Variation among these algorithms arises from their methods of representing candidate programs, such as decision trees or mathematical functions, and their strategies for navigating this program space, including optimization techniques and evolutionary search methods [6].

1.1.1 Different Machine Learning algorithms

Machine learning encompasses a wide range of learning types and algorithms. The major categories are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [7].

Supervised learning algorithms are trained on labeled data, which includes inputs and desired outputs. Common supervised tasks are classification, predicting categorical outputs like spam or not spam, and regression, predicting continuous values like price. Algorithms learn a mapping from inputs to outputs by examining many examples. Supervised learning can achieve high predictive accuracy but relies on large labeled training sets which can be costly to obtain. Popular supervised algorithms include logistic regression, support vector machines, neural networks, decision trees, and random forests.

In contrast, unsupervised learning operates on unlabeled data, finding hidden patterns and intrinsic structure within it. Clustering algorithms are a key example, grouping data points that are similar. Other unsupervised techniques like dimensionality reduction and association rule mining also derive insights from the data itself without external labeling. As labeling is not required, unsupervised methods can more easily scale to new problem domains, but the models may not have clear accuracy measures.

In summary, machine learning employs various approaches to train models that can analyze data, recognize patterns, make predictions, or perform tasks. Supervised learning offers predictive accuracy but requires labeled data. Unsupervised learning can find hidden insights in any data without labeling but has less defined performance measures. Together these technologies enable machine learning systems to perform tasks not explicitly programmed.

1.1.2 Overfitting

In supervised machine learning, an issue called overfitting arises, leading to poor generalization from observed data to new, unseen data. This results in a model performing well on the training set but poorly on the testing set. Overfitting occurs because the model becomes too specific to the training data and struggles with variations present in the testing data. Overfitted models tend to memorize noise and details of the training set rather than learning the underlying patterns [8].

Occam's Razor, or the principle of parsimony, advocates using models that contain only what's necessary for effective modeling. Overfitting occurs when models or procedures include more terms than needed, violating parsimony. Two types of overfitting are identified: using excessively flexible models and incorporating irrelevant components. Overfitting is undesirable due to resource wastage, potential prediction errors, worse decisions in feature selection, degraded predictions, and reduced portability of models. Portable models, adhering to Occam's Razor, are preferred for their broader applicability and reproducibility across locations [9, 10].

1.1.3 Spliting the dataset

In both statistical and machine learning model development, a common practice is to divide the dataset into two parts: training and testing [4]. The training set is utilized to estimate the model's unknown parameters, while the model's accuracy is assessed using the testing dataset. This division is essential to prevent overfitting, where a model becomes too tailored to the data, potentially leading to poor predictions in new situations. By reserving a subset of the data for testing, the model's performance can be evaluated before actual deployment, safeguarding against issues arising from overfitting.

The simplest and probably the most common strategy to split such a dataset is to randomly sample a fraction of the dataset. For example, 80% of the rows of the dataset can be randomly chosen for training, and the remaining 20% can be used for testing. Various strategies are showcased in [11] for efficiently and optimally dividing the dataset.

1.1.4 Validation

Furthermore, it's a frequent practice to reserve a segment of the training set for validation. This validation subset serves purposes such as refining model performance through hyper-parameter or regularization parameter selection (e.g. the number of hidden units—layers and layer widths—in a neural network [12]). Validation datasets also can be used for regularization by early stopping (stopping training when the error on the validation data set increases, as this is a sign of over-fitting to the training data set) [13].

To ensure greater result stability and maximize the use of valuable data for training, datasets can be repeatedly divided into multiple training and validation subsets. This practice, referred to as cross-validation, is employed. Additionally, an independent test dataset held apart from cross-validation is typically utilized to validate the model's performance. [14] provides an overview of various cross-validation techniques.

1.2 Thesis Structure

Chapter 2
Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 3
Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you

information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 4

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 5

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2 Data Analysis

> *Information is the oil of the 21st century, and analytics is the combustion engine.*
>
> — **Peter Sondergaard**
> (Founder of The Sondergaard Group, LLC.)

This chapter discusses data analysis methodologies. Collaborating with Mowi Space, a platform tailored for mountain biking enthusiasts and winter sports enthusiasts, this collaboration has yielded a dynamic digital platform catering to outdoor sports. The MOWI website offers real-time data and interactive 3D maps for offline exploration. It delivers current trail conditions, weather updates, lift operations, and local events, ensuring informed and safe adventures. Notably, the Live Track feature allows real-time monitoring of family or friends during mountain activities, fostering connectivity and safety. This collaboration signifies a pivotal advancement in outdoor experience planning, merging technology with nature's allure.

As a result of getting some user's data going through various tracks, it is possible to convert the GPS data into CSV files.

2.1 Converting the GPS data

This Python code snippet performs data processing on GPS data from GPX files and converts it into a more analyzable CSV format. The code utilizes several libraries for various functionalities.

2.1.1 Imports

The script begins by importing necessary Python libraries:

```python
import gpxpy
import gpxpy.gpx
import numpy as np
import haversine as hs
import pandas as pd
import os
import gpxpy
import pandas as pd
from tqdm import tqdm
import json
```

These libraries are used for working with GPX files, numerical calculations, data manipulation, and progress tracking during processing.

2.1.2 Functions

The code defines several important functions:

`gpx_to_csv`

This function converts GPX data to CSV format and calculates various metrics.

```python
def gpx_to_csv(gpx_file_path, csv_file_path):
    with open(gpx_file_path, "r") as gpx_file:
        gpx = gpxpy.parse(gpx_file)

    route_info = []
    for track in gpx.tracks:
        for segment in track.segments:
            for point in segment.points:
                route_info.append({
                    "time": point.time,
                    "latitude": point.latitude,
                    "longitude": point.longitude,
                    "altitude": point.elevation
                })

    route_df = pd.DataFrame(route_info)

    route_df["altitude_diff"] = route_df["altitude"].diff()
```

```
    route_df["relative_elevation"] = route_df["altitude_diff"].cumsum()

    distances = [np.nan]
    speed = [np.nan]

    for i in range(1, len(route_df)):
        distances.append(haversine_distance(
            lat1=route_df.iloc[i - 1]["latitude"],
            lon1=route_df.iloc[i - 1]["longitude"],
            lat2=route_df.iloc[i]["latitude"],
            lon2=route_df.iloc[i]["longitude"]
        ))

        # #* speed
        time_diff = (route_df.iloc[i].time - route_df.iloc[i - 1].time).seconds
        distances_i = distances[i]

        # Handling division by zero
        if time_diff == 0:
            speed_i = 10  # Assign an appropriate default value
        else:
            speed_i = distances_i / time_diff

        speed.append(speed_i)

    route_df["distance"] = distances
    route_df["cum_distance"] = route_df["distance"].cumsum()/1e3
    route_df["speed"] = speed

    number_of_lifts = lift_checker(route_df)
    if number_of_lifts > 0:
        report.append({
            "file": csv_file_path[11:],
            "n": number_of_lifts,
            "sum_of_n": route_df["lift_path"].sum()/2
        })
        print("----------------------------------")
        print(f"The number of lifts detected on {csv_file_path[11:]} is {number_of_lifts}")
        print("----------------------------------")
```

```python

    route_df = route_df.fillna(0)  # replace NANs with zero
    ######
    route_df.to_csv(csv_file_path, index=False)
    return route_df
```

It works as follows:

1. The function first parses the input GPX file using the *gpxpy* library and extracts key data like time, latitude, longitude and elevation into a Python dictionary for each point along the route.

2. It then converts this dictionary into a Pandas DataFrame to enable easier data manipulation.

3. Additional columns are created in the DataFrame to calculate elevation difference, cumulative elevation gain, distance between points, cumulative distance, and speed based on the time difference between points.

4. Potential divide-by-zero errors are handled when calculating speed.

5. A lift detection function is called to analyze the elevation profile and count the number of detected lifts along the route.

6. The number of detected lifts is tracked in a report.

7. Missing data in the DataFrame is filled with zeros.

8. Finally, the processed DataFrame is written out to a CSV file to save the updated route data.

9. The code returns the final DataFrame containing the enriched route data with statistics like speed, distance, elevation, and lift counts.

`haversine_distance`

In the previous function, the implementation leverages the functionality of two additional functions. Firstly, an auxiliary function is employed to compute the haversine distance, which quantifies the geographical distance between two distinct sets of latitude and longitude coordinates. This computation is facilitated through the utilization of the `haversine` library.

```python
def haversine_distance(lat1, lon1, lat2, lon2) -> float:
    distance = hs.haversine(
        point1=(lat1, lon1),
        point2=(lat2, lon2),
        unit=hs.Unit.METERS
    )
    return np.round(distance, 2)
```

`lift_checker`

Furthermore, an additional vital function comes into play. This function is dedicated to the identification of lift occurrences in the dataset. After loading the dataset of the lifts, two for loop is used. One iterates over the track and the other iterates over lifts. Then, lists are detected when the location of a track aligns with the location of a lift. In the event a lift is detected, this function augments the existing DataFrame of GPS data with two supplementary columns. The first column, designated as `lift?`, is equipped with boolean values (0 and 1) to indicate the presence of a lift at specific points. The second column, titled `lift_path`, serves as an indicator for lift pathways, allowing for the demarcation of paths corresponding to lift usage:

```python
def lift_checker(df):
    with open("lift_dataset.json", "r") as f:
        lifts = json.load(f)
    number_of_lifts = 0
    df["lift?"] = 0  # ? set the "lift?" column to zero
    df["lift_path"] = 0

    for i in range(len(df)):
        for lift in lifts:
            lift_location = lift["geoLocation"]["
                coordinatesLineString"]
            if df["latitude"][i] == lift_location[0] and df["
                longitude"][i] == lift_location[1]:
                number_of_lifts += 1
                df.loc[i, "lift?"] = 1
                df.loc[i-1:i, "lift_path"] = 1
    return number_of_lifts
```

`convert_all_gpx_to_csv`

This function processes all GPX files in a directory and converts them to CSV:

```python
def convert_all_gpx_to_csv(gpx_dir, csv_dir):
    gpx_files = [filename for filename in os.listdir(
        gpx_dir) if filename.endswith(".gpx")]
    progress_bar = tqdm(total=len(gpx_files), desc="
        Converting GPX files")
    for filename in gpx_files:
        gpx_file_path = os.path.join(gpx_dir, filename)
        csv_file_path = os.path.join(csv_dir, filename.
            replace(".gpx", ".csv"))
        gpx_to_csv(gpx_file_path, csv_file_path)
        progress_bar.update(1)
    progress_bar.close()
```

2.1.3 Main Execution

Eventually, the *convert_all_gpx_to_csv* function is invoked by providing it with the relevant directories for input GPX files (*gpx_dir*) and output CSV files (*csv_dir*). As the function iterates through each GPX file, it performs the necessary conversions and progress is visually indicated through status updates. After the conversion process concludes, a report is generated to catalog tracks that contain a minimum of one lift. This report is structured as a DataFrame and subsequently saved as a CSV file named *report.csv* within the designated data directory. The overall result is a seamless conversion of raw GPS data into a more structured and informative format, followed by the generation of a comprehensive report for further analysis.

```python
# Usage
gpx_dir = "./data/gpx_train/"
csv_dir = "./data/csv_train/"
convert_all_gpx_to_csv(gpx_dir, csv_dir)
print("Converting is finished.")

# Generating a report of the tracks
# that have at least one lift
report_df = pd.DataFrame(report)
report_df.to_csv("./data/report.csv", index=True)
```

2.2 Utilizing the CSV data

This array contains column names for the Pandas DataFrame that has been generated and calculated from analyzing GPS route data. Each element represents the name of a column:

```
['time', 'latitude', 'longitude', 'altitude',
'altitude_diff', 'relative_elevation', 'distance',
'cum_distance', 'speed', 'lift?', 'lift_path']
```

Note that the `cum_distance` corresponds to the cumulative distance, `lift?` refers to whether transportation is used or not, and `lift_path` is 1 wherever the user is on a lift.

To visualize the dataset, just to have a sense, the figure 2.1 is generated by *sweetviz* library.

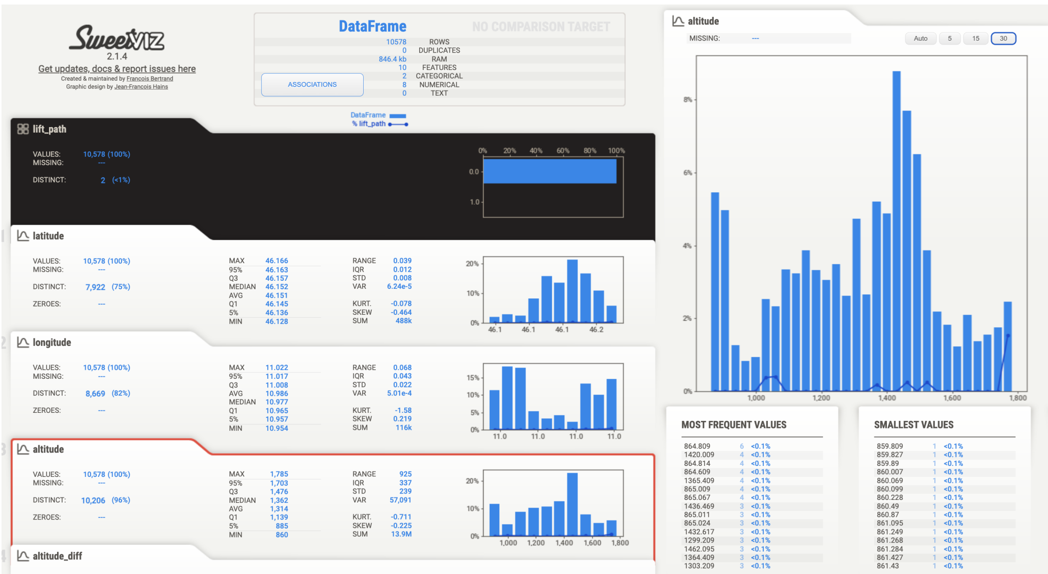

Fig. 2.1.: Data visualization: *(a)* left hand side, all the columns description, *(b)* right hand side, altitude figure and the lift path bar chart

2.2.1 Data correlation

Additionally, within the scope of this study, an accurate evaluation of data correlation will be conducted. It refers to the relationship between two or more variables, describing how changes in one variable may correspond to changes in another. Correlation analysis helps uncover patterns and dependencies in data.

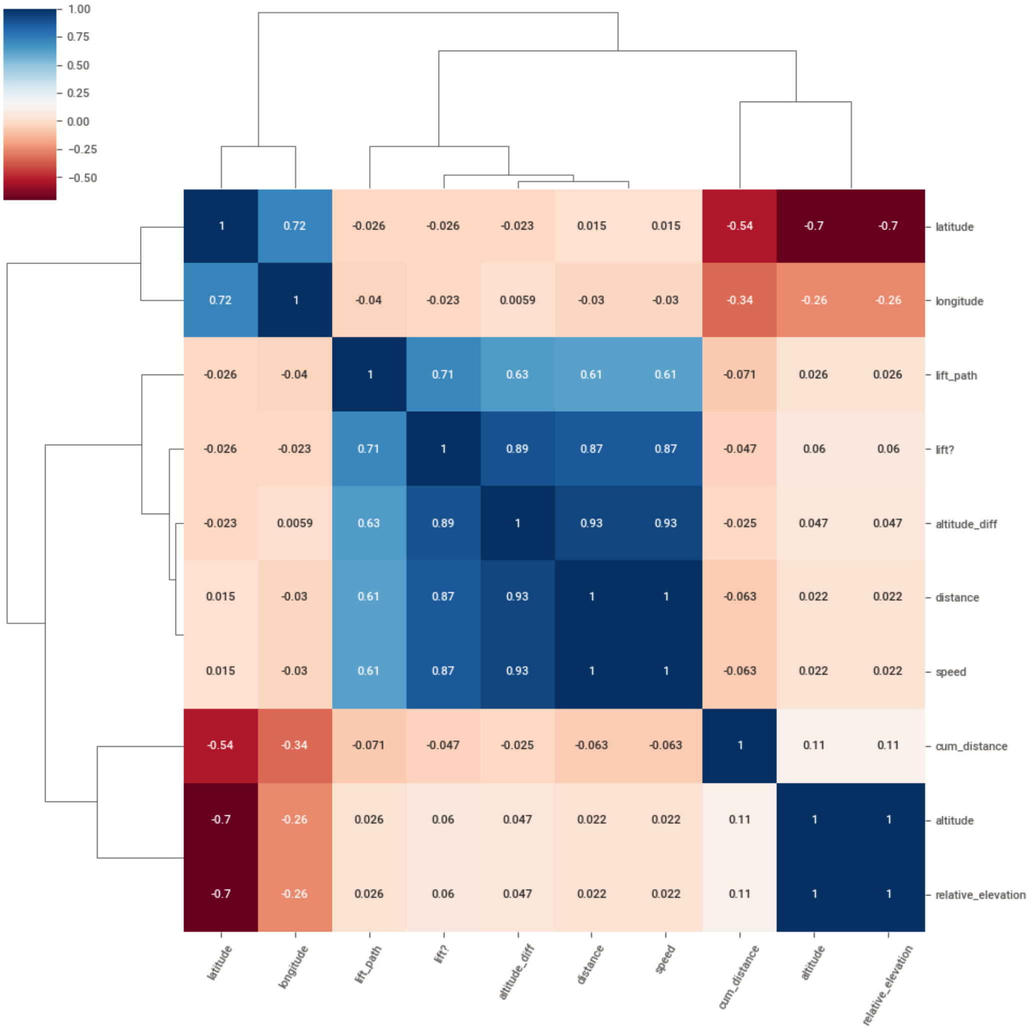

Fig. 2.2.: Data correlation

Based on the provided illustration in 2.2, it is evident that there exists a direct relationship or positive correlation between the lift variable and altitude difference, distance, as well as speed. This suggests that as there is a lift, there is a corresponding increase in altitude difference, distance, and speed. Unarguably, it is pretty evident that any mode of transportation will be faster than biking or skiing down.

2.3 Conclusion

To summarize, the main focus of this chapter was to process raw GPS data by loading it and transforming it into Pandas data frames. The data is then used for several calculations such as altitude, altitude difference, distance, cumulative distance, speed, lift detection, and lift path identification. The chapter also includes visualizing all the columns of the data frames and analyzing and data correlation. Finally, the variables with high correlation are selected as features to train ML models in **Chapter** 3

Modeling

3

This chapter provides an overview of common machine learning modeling techniques for prediction and classification tasks. Key concepts relevant to the model building process are first introduced, followed by sections describing logistic regression, random forest, and neural network models.

3.1 Modeling Concepts and Terminology

Machine learning models learn relationships and patterns from sample data in order to make predictions or decisions. The data used for training is known as the training set, containing numerous examples the model can learn from. Each data point or example is represented using features, also called predictor variables or independent variables. These are the input variables describing an observation. The output being predicted is called the target variable or dependent variable.

Features can be categorical, ordinal, or continuous/numerical. The target variable is typically categorical for classification tasks or continuous for regression tasks. Feature selection and engineering is an important part of the modeling process, ensuring the model has relevant and informative attributes to train on. Feature scaling through techniques like normalization is often necessary. The training data should be representative of the real-world use cases.

To shed a light on this Do plants grow faster in natural or artificial research? The type of light the plant grows under The plans growth rate

3.1.1 Coding Setup

Starting the project requires the creation of a new environment with conda. It is strongly advised to refer to the Github page [15] for the complete project and a list of hundreds of packages to be installed `environment.yml`.

As the packages installed correctly, at the top of the code they are imported. In this project, sklearn library is used:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import linear_model

data = pd.read_csv("path_to_csv")
```

A brief look at the data set:

	time	latitude	longitude	altitude	altitude_diff	relative_elevation	distance	cum_distance	speed	lift?	lift_path
0	2022-10-02 09:46:55+00:00	46.156385	10.955038	1524.084	0.000	0.000	0.00	0.00000	0.000000	0	0.0
1	2022-10-02 09:46:56+00:00	46.156384	10.955040	1521.207	-2.877	-2.877	0.19	0.00019	0.190000	0	0.0
2	2022-10-02 09:46:59+00:00	46.156368	10.955054	1521.017	-0.190	-3.067	2.08	0.00227	0.693333	0	0.0
3	2022-10-02 09:47:04+00:00	46.156375	10.955060	1521.209	0.192	-2.875	0.91	0.00318	0.182000	0	0.0
4	2022-10-02 09:47:06+00:00	46.156371	10.955066	1521.279	0.070	-2.805	0.64	0.00382	0.320000	0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
10573	2022-10-02 16:57:54+00:00	46.144351	10.965109	928.883	0.274	-595.201	0.68	57.21197	0.340000	0	0.0
10574	2022-10-02 16:57:56+00:00	46.144371	10.965139	928.873	-0.010	-595.211	3.21	57.21518	1.605000	0	0.0
10575	2022-10-02 16:57:59+00:00	46.144381	10.965139	928.897	0.024	-595.187	1.11	57.21629	0.370000	0	0.0
10576	2022-10-02 16:58:01+00:00	46.144385	10.965150	929.043	0.146	-595.041	0.96	57.21725	0.480000	0	0.0
10577	2022-10-02 16:58:02+00:00	46.144383	10.965153	929.070	0.027	-595.014	0.32	57.21757	0.320000	0	0.0

10578 rows × 11 columns

Fig. 3.1.: The loaded DataFrame

In the next step the dependent and independent variables are defined.

```
from sklearn.model_selection import train_test_split

# Select relevant features (columns)
features = ["distance", "speed", "altitude_diff"]

# Define the target column
target = "lift?"

# Split the data into features (X) and target (y)
X = data[features]
y = data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

By running the following command, the size of the independent variable X is determined:

```
# Display the shape of the training and testing sets
print("x_train shape:", X_train.shape)
print("x_test shape:", X_test.shape)
```

```

# output:
  x_train shape: (8462, 3)
  x_test shape: (2116, 3)
```

Taking into account the dependency of relying only on a particular CSV file, it becomes apparent that the integration of all the diverse DataFrames produced in Chapter 2.2 would result in a more comprehensive model training and validation process.

```
# Load data from multiple CSV files
csv_dir = 'csv_folder'
file_path = os.listdir(csv_dir)

data_frames = [pd.read_csv(f'./data/csv_train/{file}')
               for file in file_path]
combined_data = pd.concat(data_frames, ignore_index=False)

df = combined_data[combined_data['speed'] == np.inf]
combined_data.describe()
```

The output of the the above code would be the Fig. 3.2.

	latitude	longitude	altitude	altitude_diff	relative_elevation	distance	cum_distance	speed	lift?	lift_path
count	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000	246395.000000
mean	46.183239	10.960967	1266.506680	-0.012680	262.499626	20.348112	26.923287	3.852743	0.000244	0.000467
std	0.078900	0.085127	424.490643	8.237209	426.892697	62.215384	19.580102	3.565807	0.015603	0.021599
min	45.884029	10.510484	30.000000	-947.250000	-1159.211000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	46.142949	10.942188	1004.839000	-0.827000	-1.674500	4.640000	11.217755	1.762172	0.000000	0.000000
50%	46.156417	10.989996	1247.733000	-0.011000	198.208000	7.030000	23.207870	3.065000	0.000000	0.000000
75%	46.183007	11.015397	1544.059500	0.607000	532.154500	12.520000	39.206765	5.215893	0.000000	0.000000
max	46.551682	11.115855	2625.868000	833.319000	1867.893000	12552.920000	107.941840	739.370000	1.000000	1.000000

Fig. 3.2.: The details of all the DataFrames concated together

It can be seen in Fig. 3.2 that there are 11 columns and 246,395 rows in the `combined_data`. Again, the dependent variables and independent variable are extracted from the DataFrame. To avoid unnecessary duplication, the code for this part is not repeated, as it closely resembles the mentioned codes above.

According to the Fig. 3.3, all the dependent variables are significant.

3.2 Approaches used

As the project is classification [16], the following approaches are used.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lift?   R-squared:                       0.511
Model:                            OLS   Adj. R-squared:                  0.511
Method:                 Least Squares   F-statistic:                 8.599e+04
Date:                Sun, 27 Aug 2023   Prob (F-statistic):               0.00
Time:                        18:41:17   Log-Likelihood:             7.6371e+05
No. Observations:              246395   AIC:                        -1.527e+06
Df Residuals:                  246391   BIC:                        -1.527e+06
Df Model:                           3
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const            -0.0013   3.25e-05    -41.491      0.000      -0.001      -0.001
distance       5.052e-05   3.63e-07    139.106      0.000    4.98e-05    5.12e-05
altitude_diff     0.0012    2.7e-06    460.182      0.000       0.001       0.001
speed             0.0002   6.27e-06     23.954      0.000       0.000       0.000
==============================================================================
Omnibus:                   756968.658   Durbin-Watson:                   1.713
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      111685333421.084
Skew:                          46.871   Prob(JB):                         0.00
Kurtosis:                    3299.949   Cond. No.                         97.7
==============================================================================
```

Fig. 3.3.: OLS Regression Results on the DataFrames

3.2.1 Logistic Regression

Logistic regression is a common statistical technique adapted for machine learning. It is suited for binary classification tasks where the target variable has two possible classes. Logistic regression models the probability of an observation belonging to each class. The logistic function ensures the probabilities are between 0 and 1 [17].

Regression coefficients are learned during training to associate each feature with the log odds of the target. Important considerations when training logistic regressions include handling class imbalance and avoiding overfitting. Regularization methods like L1 and L2 can help prevent overfitting. Logistic regression is easy to implement, fast to train, and interpretable, but may not achieve best-in-class accuracy.

```python
from sklearn.linear_model import LogisticRegression


model_lr = linear_model.LogisticRegression(
multi_class="multinomial", solver="lbfgs", max_iter=120,
    verbose=True)
model_lr.fit(X_train, y_train)
```

3.2.2 Random Forest

Random forest is an ensemble method that trains multiple decision trees on subsets of data and features, combining their predictions through voting or averaging. By training on subsets, the decision trees exhibit greater diversity, reducing overfitting. Random forests achieve strong predictive accuracy by aggregating across many decision trees to smooth out individual errors [18].

Tuning key hyperparameters like number of trees, maximum depth, and minimum samples per leaf can improve random forest performance. Feature importance scores can be calculated to understand impact on predictions. Random forests are accurate and robust to noise, but lose interpretability compared to simpler models. They can be prone to overfitting with noisy or complex data.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


# Initialize and train the Random Forest model
model_rf = RandomForestClassifier(n_estimators=100,
    random_state=42)
model_rf.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = model_rf.predict(X_test)

# Evaluate the model"s performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

3.2.3 Cross validation

```
from sklearn.model_selection import cross_val_score

# Create a Random Forest classifier
model_rf = RandomForestClassifier(n_estimators=100,
    random_state=42)

# Perform cross-validation with 5 folds
cross_val_scores = cross_val_score(model_rf, X, y, cv=5,
    scoring="accuracy")
```

```python

# Print the cross-validation scores
print("Cross-validation Scores:", cross_val_scores)
print("Mean Accuracy:", cross_val_scores.mean())
```

3.2.4 Neural Networks

Artificial neural networks are computing systems inspired by animal brains. They contain interconnected nodes called neurons arranged in layers. Input features are fed into input neurons, transformed through hidden layers of neurons via weighted connections, and output from output neurons. Neural nets learn by adjusting connection weights during training to minimize prediction error [19].

Deep neural networks contain more hidden layers enabling modeling of complex nonlinear relationships in data. Key considerations include network topology and hyperparameter selection. Neural networks can model complex interactions between features that other models may miss. However, they require substantial data for training and are difficult to interpret compared to other techniques.

```python
from sklearn.neural_network import MLPClassifier

# Create neural network model
model_nn = MLPClassifier(hidden_layer_sizes=(
  6,), activation="relu", solver="adam", random_state=1)

# Perform 5-fold cross validation
scores = cross_val_score(model_nn, X, y, cv=5)
print("Cross-validation scores: ", scores)

# Train model on training set
model_nn.fit(X_train, y_train)

# Evaluate model performance on test set
print("Test set score: ", model_nn.score(X_test, y_test))
```

3.3 Summary

This section has provided an overview of key machine learning modeling concepts like features, target variables, and training data. Introductory explanations of three major algorithms - logistic regression, random forests, and neural networks - were also presented. These descriptions aim to establish essential foundational knowledge before diving into modeling details and experiments in subsequent chapters.

3.4 model Section 1

3.5 Conclusion

4 Results

4.1 Concepts Section 1

4.2 Concepts Section 2 with a very very long title that illustrates how long section titles are handled in the footer

4.3 Concepts Section 3

4.4 Conclusion

5 Conclusion

5.1 System Section 1

5.2 System Section 2

5.3 Future Work

Bibliography

[1]Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021 (cit. on pp. 2, 3).

[2]Tom M Mitchell. *Machine learning*. 1997 (cit. on p. 2).

[3]David Hand, Heikki Mannila, and Padhraic Smyth. "Principles of data mining. 2001". In: *MIT Press. Sections* 6.3 (2001), pp. 2–6 (cit. on p. 2).

[4]Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009 (cit. on pp. 3, 4).

[5]Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012 (cit. on p. 3).

[6]Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260 (cit. on p. 3).

[7]Ethem Alpaydin. *Introduction to Machine Learning*. 3rd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2014 (cit. on p. 3).

[8]Xue Ying. "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168.2 (Feb. 2019), p. 022022 (cit. on p. 4).

[9]Douglas M. Hawkins. "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1 (2004), pp. 1–12 (cit. on p. 4).

[10]JA Cook and J Ranstam. "Overfitting". In: *Journal of British Surgery* 103.13 (2016), pp. 1814–1814 (cit. on p. 4).

[11]V. Roshan Joseph and Akhil Vakayil. "SPlit: An Optimal Method for Data Splitting". In: *Technometrics* 64.2 (2022), pp. 166–176 (cit. on p. 5).

[12]Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007 (cit. on p. 5).

[13]Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67 (cit. on p. 5).

[14]Michael W Browne. "Cross-Validation Methods". In: *Journal of Mathematical Psychology* 44.1 (2000), pp. 108–132 (cit. on p. 5).

[15]Nima Karimi. *M.Sc. Thesis Repository*. `https://github.com/nimakarimi76/my-thesis-MSc` (cit. on p. 17).

[16]FY Osisanwo, JET Akinsola, O Awodele, et al. “Supervised machine learning algorithms: classification and comparison”. In: *International Journal of Computer Trends and Technology (IJCTT)* 48.3 (2017), pp. 128–138 (cit. on p. 19).

[17]David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013 (cit. on p. 20).

[18]Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32 (cit. on p. 21).

[19]Phil Picton and Phil Picton. *What is a neural network?* Springer, 1994 (cit. on p. 22).

List of Figures

List of Tables

List of Listings

Example Appendix A

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

A.1 Appendix Section 1

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.1.: This is a caption text.

A.2 Appendix Section 2

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.2.: This is a caption text.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Padova, September 05, 2023

Nima Karimi