Analyse von Straßenoberflächen mit Beschleunigungsdaten

Olivia Lawinski

Matrikelnr: 586732

IKT-Kurs, Wintersemester 2023/2024

23. September 2024

1.1. Einleitung

Projektübersicht:

In diesem Projekt werden Straßenoberflächen mithilfe von Beschleunigungsdaten analysiert, die über ein Smartphone erfasst wurden. Das Hauptziel besteht darin, unterschiedliche Oberflächenqualitäten wie Asphalt, Pflastersteine oder beschädigte Straßen zu erkennen und zu klassifizieren.

Motivation und Relevanz:

Die Qualität von Straßenbelägen beeinflusst nicht nur den Fahrkomfort, sondern auch die Sicherheit und Effizienz des Verkehrs. Unebene oder beschädigte Straßen können das Unfallrisiko erhöhen und die Fahrzeugabnutzung beschleunigen. Daher ist es wichtig, Methoden zur Überwachung der Straßenqualität zu entwickeln, um präventive Maßnahmen zu ermöglichen und den Verkehrsfluss zu optimieren.

Problemstellung:

Die unterschiedlichen Oberflächen wie Asphalt, Pflastersteine oder unebene Straßen wirken sich verschieden auf das Fahrerlebnis aus. Eine detaillierte Erfassung und Analyse dieser Belagsarten ist entscheidend, um Rückschlüsse auf den Zustand der Straßen und potenzielle Risiken zu ziehen.

1.2. Verwendete Daten

Datenquellen:

Bachelorarbeit von Minh Do: *Straßenklassifikation mittels Smartphone*

Projektteam IQ-Trans der HTW: Entwicklung eines IoT-Echtzeitmonitoringsystems zur Qualitätssicherung von Ebersperma während des Transports, basierend auf Echtzeitsensordaten. Dieses System berücksichtigt Umweltfaktoren wie Temperatur, Licht und Erschütterungen während der Fahrt, um den Zustand der transportierten Ware zu überwachen. Eine wichtige Variable dabei ist die Straßenqualität, die Einfluss auf die Stabilität der Ware während des Transports hat.

Smartphones bieten heutzutage eine hohe Rechenleistung und enthalten Sensoren wie Gyroskope, GNSS, Beschleunigungs- und Orientierungssensoren . Daher sind sie gut geeignet, um eine mobile und günstige Lösung für die Klassifizierung von Straßenoberflächen zu bieten.

Datenerhebung:

Das Smartphone wurde während der Messfahrten auf der mittleren Armlehne des Fahrzeugs platziert, mit dem Rücken nach unten und parallel zum Boden. Die Achsen des Smartphones wurden so ausgerichtet, dass sie mit den Fahrzeugachsen übereinstimmen. In dieser Position wurde das Gerät fest fixiert (siehe Abb. 2).

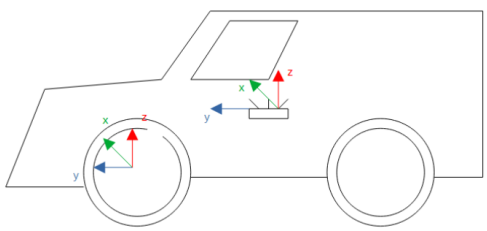


Abb. 2: Platzierung des Smartphones im Fahrzeug mit übereinstimmenden Achsen (X,Y,Z)

Datenbeschreibung:

Für jede Fahrt wurde eine Textdatei generiert, die die erfassten Sensordaten im folgenden Format speichert:

UnixTime in ms; Messgerät; Sensor-ID; Messwerte
1629745197750;LOCAL_PHONE;11;20210823
1629745197749;LOCAL_PHONE;10;205957
1629745196615;LOCAL_PHONE;AB;52.5593763,13.6304612
1629745197750;LOCAL_PHONE;11;20210823
1629745198027;LOCAL_PHONE;20;1.16,-0.59,-0.56
1629745198028;LOCAL_PHONE;20;-0.32,0.39,0.16
1629745198029;LOCAL_PHONE;20;0.3,-0.5,-1.04

Die aufgezeichneten Daten umfassen Zeitstempel, Sensor-IDs und Beschleunigungswerte, die zur Analyse der Straßenqualität verwendet werden.

1.3. Datenvorbereitung und -exploration

Datenbereinigung:

Um die Daten gezielt für die Analyse der Straßenqualität zu nutzen, wurden unnötige Spalten entfernt. Da die App ursprünglich entwickelt wurde, um Daten von mehreren Sensoren aufzuzeichnen, befinden sich in der Rohdatei auch Messwerte, die für die Klassifizierung der Straßenbeläge irrelevant sind. Für die Analyse sind jedoch nur die Beschleunigungsdaten mit der Sensor-ID „20“ relevant, da hier die X-, Y- und Z-Beschleunigungswerte gespeichert werden.

Es wird nur die Z-Achse in betracht gezogen, da diese die vertikale Beschleunigung erfasst, was aufgrund der Positionierung des Smartphones die relevanteste Achse für die Bewertung der Straßenqualität ist. Da die Unix-Zeit für die eigentliche Analyse nicht mehr notwendig war, wurde diese Spalte ebenfalls verworfen. Darüber hinaus wurden GPS-Daten unter der Sensor-ID „AB“ erfasst, jedoch lagen nur wenige Datenpunkte vor. Diese wurden hauptsächlich zur Visualisierung der Fahrtrouten verwendet.

Datenexploration:

In diesem Schritt wurden die gesammelten Daten auf potenzielle Ausreißer untersucht und fehlerhafte Datenpunkte gefiltert. Durch erste Visualisierungen konnte ein besseres Verständnis der Datenstruktur gewonnen werden. Beispielsweise wurden Histogramme und Scatterplots verwendet, um die Verteilung der Beschleunigungswerte zu visualisieren und potenzielle Unregelmäßigkeiten zu erkennen.

Feature Engineering:

Um die Daten besser für maschinelles Lernen zu nutzen, wurden zusätzliche Merkmale (Features) generiert. Aus der Messwertspalte wurden die X-, Y- und Z-Beschleunigungswerte extrahiert. Zudem wurde ein „Sliding Window“-Verfahren angewendet, um in regelmäßigen Abständen statistische Merkmale aus den Daten zu berechnen. Zu den berechneten Features gehören unter anderem:

- Durchschnittliche Beschleunigung
- Standardabweichung
- Minimum und Maximum der Beschleunigung *(aus Zeitgründen nicht mehr geschafft)*
- Frequenzbasierte Merkmale (z. B. Peaks oder Schwingungen) *(aus Zeitgründen nicht mehr geschafft)*

Diese Merkmale liefern eine detaillierte Grundlage, um verschiedene Straßenbeläge anhand der aufgezeichneten Beschleunigungsdaten zu klassifizieren.

1.4. Vorgehensweise und Sprints

1. Sprint-Dokumentation: Erste Versuche und Datenexploration

Ziele:
Da die Messdaten in drei verschiedenen Ordnern sowie mehreren Unterordnern gespeichert waren, bestand das erste Ziel darin, die Daten in drei Kategorien zu unterteilen: *bumpy_roads*, *cobblestone_street* und *flat_streets*. Anschließend wurden die Daten in eine CSV-Datei umgewandelt, um erste Datenbereinigungen und explorative Analysen durchzuführen.

Durchgeführte Arbeit:
Beim ersten Versuch wurden die Unterordner nicht aufgelöst, was die Verarbeitung unnötig erschwert hat. Erst nachträglich wurden die Unterordner aufgelöst, was die Strukturierung der Daten deutlich verbessert hat.

Ergebnisse:
Die Unterteilung der Daten in klare Kategorien legte den Grundstein für die spätere Analyse und Modellierung.

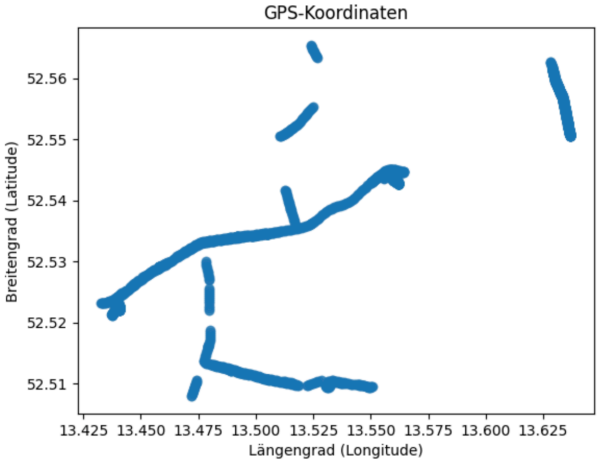
Learnings:
„Learning by doing“: Durch das Ausprobieren verschiedener Ansätze und das Verständnis auftretender Fehler konnte ich viel über die Struktur der Daten lernen.

2. Sprint-Dokumentation: GPS-Daten erkunden und visualisieren

Ziel:
Die GPS-Daten sollten für eine Visualisierung der Fahrtrouten genutzt werden.

Durchgeführte Arbeit:
Leider wurde hier versehentlich der falsche, noch unaufbereitete Datensatz verwendet. Dennoch habe ich mir zuerst alle möglichen Informationen über die Daten ausgeben lassen, wie z.B. die Anzahl fehlender Werte pro Spalte, den Datentyp und Nullwerte. Anschließend habe ich nach dem Sensor-Typ „AB“ gefiltert, da hier die GPS-Daten gespeichert sind, und mir die Koordinaten grafisch anzeigen lassen. Die Visualisierung war zunächst verwirrend, machte aber im späteren Vergleich mehr Sinn (siehe beigefügte Karten). Zudem wurde eine Heatmap erstellt, um die Dichte der Fahrtrouten darzustellen.

Ergebnisse:
Eine erste visuelle Darstellung der GPS-Daten half, die Struktur der Daten besser zu verstehen.



Jupyter Notebook

Aus Streamlit-App

3. Sprint-Dokumentation: Erster Versuch mit Modelltesten (RandomForest & GridSearch)

Ziel:
Testen eines ersten Modells mit dem *RandomForestClassifier* und Durchführung einer Hyperparameteroptimierung mit GridSearch.

Durchgeführte Arbeit:
Hier wurde versehentlich die falsche CSV-Datei genutzt, was zu langen Rechenzeiten und unzufriedenstellenden Ergebnissen führte. Nach einer Filterung der Sensor-ID „20“ und Extraktion des Z-Werts wurde ein erster Versuch mit dem *RandomForestClassifier* durchgeführt. Dabei zeigte das *classification_report*, dass einige Klassen unterrepräsentiert sind, was die Modellleistung beeinträchtigt. Durch den Einsatz von SMOTE (Oversampling) konnte das Problem nur teilweise behoben werden. Auch nach der Anwendung von GridSearchCV für das Hyperparameter-Tuning blieb die Modellleistung unzureichend.

Ergebnisse:
Obwohl der Prozess langwierig war, lieferte der erste Versuch wertvolle Erkenntnisse, insbesondere über das Ungleichgewicht der Klassen.

4. Sprint-Dokumentation: Modelltest KNN, SMOTE & GridSearch

Ziel:
Testen des K-Nearest-Neighbor (KNN) Modells und Optimierung mit SMOTE und GridSearch.

Durchgeführte Arbeit:
Ähnlich wie beim RandomForest wurde der KNN-Algorithmus mit den gleichen Methoden getestet. Dabei zeigte sich erneut, dass SMOTE das Ungleichgewicht nur bedingt ausgleichen konnte.

Ergebnisse:
Die Resultate mit KNN waren ähnlich wie beim RandomForest: Verbesserungen waren vorhanden, jedoch reichte das Ergebnis nicht aus.

5. Sprint-Dokumentation: Zweiter Versuch mit Modelltesten (RandomForest & GridSearch)

Ziel:
Erneuter Test des *RandomForestClassifier* mit korrekt aufbereitetem Datensatz.

Durchgeführte Arbeit:
Mit der richtigen CSV-Datei konnte das Modell bessere Ergebnisse liefern. Dennoch zeigte die Confusion Matrix weiterhin Schwierigkeiten bei der Klassifikation der *bumpy_roads*, deren Genauigkeit bei Null lag. Nach Anwendung von Oversampling verbesserten sich die Ergebnisse leicht, jedoch war die Modellleistung immer noch nicht zufriedenstellend. GridSearch brachte ebenfalls keine wesentliche Verbesserung.

Ergebnisse:
Die Modellgenauigkeit konnte leicht verbessert werden, jedoch bleibt das Problem der unzureichenden Klassifikation der *bumpy_roads* bestehen.

6. Sprint-Dokumentation: KNN, SMOTE & GridSearch, Plotten der Cross-Validation-Ergebnisse, Bias-Variance-Tradeoff

Ziele:
Im Rahmen dieses Sprints wurden der *K-nearest neighbors* (KNN)-Algorithmus optimiert, die SMOTE-Technik zur Erhöhung der Datenbalance eingesetzt und eine Hyperparameter-Optimierung mithilfe von GridSearch durchgeführt. Zudem wurden die Cross-Validation-Ergebnisse visualisiert und eine Analyse des Bias-Variance-Tradeoffs vorgenommen.
Durchgeführte Arbeit:
Zunächst wurde ein KNN-Modell mit k=5 getestet, wobei die erste erzielte Genauigkeit (Accuracy) bei 0.5493 lag. Anschließend wurde ein Parameterbereich für das GridSearch definiert, um die optimalen Hyperparameter zu ermitteln. Das Ergebnis zeigte, dass die besten Parameter {'n_neighbors': 11, 'weights': 'uniform'} waren. Dies führte zu einer verbesserten Genauigkeit von 0.6117.

Ergebnisse:
Die Durchführung einer Cross-Validation für jeden k-Wert im Bereich von 1 bis 21 ergab, dass die Genauigkeit mit steigender Anzahl von Nachbarn zunahm. Der beste Wert für die Anzahl der Nachbarn lag bei k=20.

Bias-Variance-Tradeoff:
Zum Abschluss des Sprints wurde eine Analyse des Bias-Variance-Tradeoffs durchgeführt. Die Linien für den Trainingsfehler und den Testfehler wurden geplottet, wobei festgestellt wurde, dass die Fehlerwerte mit zunehmender Anzahl von Nachbarn stetig abnahmen. Insbesondere lagen die Train- und Test-Error-Linien eng beieinander, was auf ein gutes Modell hinweist, das weder unter- noch überanpasst war.

7. Sprint-Dokumentation: Sliding Window und Modelltesting

Ziele:
In diesem Sprint wurde der Sliding-Window-Ansatz verwendet, um die Z-Beschleunigungswerte in kleinere Abschnitte zu unterteilen und daraus Merkmale für das Modelltraining zu generieren. Ziel war es, für jedes Fenster statistische Merkmale wie den Durchschnitt und die Standardabweichung zu berechnen und die erstellten Features für die Klassifizierung der Straßentypen zu verwenden.

Durchgeführte Arbeit:
Der Sliding-Window-Ansatz wurde mit folgenden Parametern implementiert:
- **Fenstergröße**: 100 Werte (entspricht ca. 1 Sekunde Messzeit)
- **Schrittweite**: 50 Werte (50 % Überlappung der Fenster)

Für jedes Fenster wurden folgende Schritte durchgeführt:
- Extraktion der Z-Beschleunigungswerte.
- Berechnung von Durchschnitt und Standardabweichung für jedes Fenster.
- Zuweisung von Labels zu den Sliding Windows basierend auf den Straßentypen.

Dabei war es entscheidend, sicherzustellen, dass die Anzahl der erzeugten Features mit der Anzahl der Labels übereinstimmte, da dies eine Grundvoraussetzung für das erfolgreiche Modelltraining ist.
Anschließend wurden die Modelle *K-nearest neighbors* (KNN) und *RandomForest* auf die neuen Features angewendet:
- KNN-Accuracy: 0.6742
- RandomForest-Accuracy: 0.6994

Probleme und Herausforderungen:
Während dieses Sprints trat ein schwerwiegendes Problem auf: Die benötigten Bibliotheken konnten trotz mehrfacher Neuinstallation nicht importiert werden. Dies führte dazu, dass ein neues virtuelles Environment erstellt werden musste. In diesem neuen Environment wurden alle erforderlichen Pakete und Abhängigkeiten (inklusive *ipykernel*) erneut installiert, sodass die Umgebung auch in Jupyter verwendet werden konnte.

Learnings:
Der Sliding-Window-Ansatz stellte sich als nützlich für die Feature-Generierung heraus, und die erzielten Modellgenauigkeiten waren vielversprechend. Das Auftreten von technischen Problemen mit den Bibliotheken und die notwendige Neueinrichtung der Entwicklungsumgebung verdeutlichten jedoch die Bedeutung eines stabilen Environments für die Entwicklung.

1.5. Entwicklung der Streamlit App

App-Entwurf:

Der Entwurf der Streamlit-App folgte einer klaren Struktur, um eine benutzerfreundliche Oberfläche zur Visualisierung und Analyse der Straßenoberflächendaten bereitzustellen. Der Hauptfokus lag darauf, die Messdaten einfach hochzuladen und die wichtigsten Informationen sowie Visualisierungen (wie Zeitreihen, Heatmaps und Straßenbelagsklassifikationen) intuitiv darzustellen. Der Workflow wurde in drei Hauptbereiche unterteilt:

1. **Daten hochladen:** Ermöglicht es den Benutzern, CSV-Dateien mit den Sensordaten hochzuladen.
2. **Datenanalyse und -visualisierung:** Integrierte Funktionen zur Visualisierung von Beschleunigungswerten, GPS-Daten sowie statistischen Merkmalen der Straßenoberflächen.
3. **Modelltesting:** Implementierung von Machine-Learning-Modellen (KNN, RandomForest) zur Klassifikation von Straßenbelägen basierend auf den Beschleunigungsdaten.

Implementierung:

Die Implementierung der Streamlit-App wurde in mehreren Schritten durchgeführt. Zunächst wurde eine Benutzeroberfläche mit Streamlit gestaltet. Anschließend wurden Funktionen zur Datenverarbeitung und -bereinigung integriert, die es den Nutzern ermöglichen, ihre Sensordaten effizient zu verarbeiten. Die wichtigsten Funktionen der Implementierung waren:

- **Datenvisualisierung**: Grafische Darstellung der Beschleunigungsdaten in Form von Zeitreihen, Histogrammen und Karten.
- **Datenfilterung und -bereinigung**: Funktionen zur Extraktion relevanter Merkmale wie Z-Beschleunigungsdaten sowie zum Entfernen von Ausreißern.
- **Modelltest**: Implementierung der Machine-Learning-Modelle (KNN, RandomForest) zur Klassifikation der Straßenoberflächen. Diese Modelle wurden mit einer übersichtlichen Benutzeroberfläche ausgestattet, um die Modellleistung (Accuracy, Confusion Matrix) direkt anzuzeigen.

1.6. Reflexion und Ausblick

Erreichte Ziele:
Das Ziel, einen gut strukturierten und bereinigten Datensatz in einer CSV-Datei zu speichern, wurde erreicht. Die Daten wurden effizient gefiltert und bereinigt, allerdings gibt es noch Potenzial für weitere Optimierungen in der Datenaufbereitung. Die Streamlit-App bietet einen guten Einblick in die Analyse und Klassifikation von Straßentypen, obwohl sie nicht alle geplanten Funktionen wie erwartet umsetzt. Einige Features funktionieren noch nicht wie gewünscht, aber die Richtung, in die sich die App entwickelt, ist klar erkennbar.

Herausforderungen:
Zu Beginn des Projekts gab es große Herausforderungen bei der Organisation der Messdaten, da diese in vielen verschiedenen Ordnern und Unterordnern gespeichert waren. Es war schwierig, einen klaren Ansatz für die Analyse zu finden und das richtige Modell für die Klassifikation

auszuwählen. Mehrmals bin ich in eine Sackgasse geraten, was jedoch zu wertvollen Lerneffekten führte. Auch technische Probleme mit den Bibliotheken und die Neuinstallation der Entwicklungsumgebung haben gezeigt, wie wichtig ein stabiles Environment für eine erfolgreiche Entwicklung ist.

In der Streamlit-App gab es einige spezifische Herausforderungen:

- Das Bild sowie der Textauf der Startseite werden auf allen Seite angezeigt
- Beim Hochladen einer CSV-Datei tritt ein Fehler auf
- In der Navigationsleiste unter "Datenanalyse" wollte ich verschiedene Visualisierungsmöglichkeiten aufzeigen, aber einige Darstellungen funktionieren nicht wie geplant

Zukünftige Arbeiten:

In Zukunft plane ich, die Streamlit-App weiter zu verbessern, insbesondere die genannten Fehler zu beheben und die Funktionalität auszubauen. Folgende Punkte stehen auf der Agenda:

- Optimierung der Bild- und Textdarstellung auf allen Seiten der App
- Behebung des CSV-Upload-Problems, um eine reibungslose Datenanalyse zu ermöglichen
- Erweiterung der Visualisierungsoptionen, um eine umfassendere Datenanalyse und Interpretation zu ermöglichen
- Weiteres Modelltraining und Feintuning, um die Klassifikationsgenauigkeit zu erhöhen (Minimum und Maximum der Beschleunigung & Frequenzbasierte Merkmale (z. B. Peaks oder Schwingungen)