**DLsP Projektaufgabe  
Prozessüberwachung einer Waschmaschine**

Dokumentation

an der  
FH Aachen  
FB Wirtschaftswissenschaften  
Industrial Engineering  
Dozent Prof. Dr.-Ing. Stephan Kallweit  
15.03.2022

Vorgelegt von  
Tony Peter Ettrich  
Matr.Nr.: 3093498

# Inhaltsverzeichnis

[Inhaltsverzeichnis II](#_Toc98268588)

[1 Prozessbeschreibung 1](#_Toc98268589)

[2 Datenerfassung 1](#_Toc98268590)

[3 Datentypen 2](#_Toc98268591)

[4 Datenmenge 2](#_Toc98268592)

[5 Datenprotokolle 2](#_Toc98268593)

[6 Analyse 2](#_Toc98268594)

[7 Datenbankanbindung 5](#_Toc98268595)

[8 Datenzugänglichkeit 5](#_Toc98268596)

[9 Benutzeroberfläche 6](#_Toc98268597)

# Prozessbeschreibung

Die verschiedenen Betriebszustände einer Waschmaschine sollen diagnostiziert werden. Hierzu wird ein IoT-Device mit einer IMU (Inertial Measurement Unit) verwendet. Die Vibrationen sind signifikant für den Betriebszustand. Die Daten sollen aufbereitet, umgewandelt, komprimiert, klassifiziert und dargestellt werden.

Zur Datenerfassung wird das IoT-Device an der Waschmaschine angebracht. Es erfasst die Vibrationen der Waschmaschine. Die Daten werden als UTF8-String via MQTT an einen Broker übermittelt. Der Broker ist mit einem MongoDB-Server verbunden. Der Broker konvertiert die Daten in ein JSON-Dict und schreibt sie in eine Collection, die auf dem MongoDB-Server liegt.

Die Daten können von Jedermann mit den Zugangsdaten vom Server abgerufen werden. Die abgerufenen Daten werden per fast Fourier Transformation und Power-Spectral-Density analysiert und visualisiert. Das Ergebnis soll in einem GUI ausgegeben werden können.

# Datenerfassung

Zur Datenerfassung wird das IoT-Device an der Waschmaschine angebracht. Das IoT-Device besteht aus einem ESP8266 Microcontroller (Wemos D1 Mini) und einem MPU6050 Beschleunigungssensor und Gyroskop. Via I2C-Bus liest der Microcontroller die Daten des MPU6050. Zur Programmierung des IoT-Devices wird Micropython verwendet.

Ein Bild, das Text, drinnen enthält.

Automatisch generierte Beschreibung

Abb. 2‑1 IoT-Device an der Waschmaschine

Die Waschmaschine hat eine maximale Drehzahl von 1.200 Umdrehungen pro Minute. Zur Sicherheit sollte die Messung mit der zehnfachen Frequenz stattfinden, also mit mindestens 12.000 Messungen pro Minute bzw. 200 Hz.

# Datentypen

Jeder Datensatz hat eine Größe von 121 Byte und beinhaltet 6 Variablen:

1. *dev* Name des IoT-Device String mit 6 Zeichen
2. *dt* DateTime String mit 19 Zeichen
3. *time* Unix epoch time Integer mit 10 Zeichen
4. *x\_a* X-Beschleunigung float
5. *y\_a* Y-Beschleunigung float
6. *z\_a* Z-Beschleunigung float

# Datenmenge

Pro Messung werden 100 Datensätze erfasst. Jede Messung erzeugt 12.100 Byte bzw. 11,82 kB an Daten.

Alle 30 Sekunden findet eine Messung statt. Pro Stunde werden bis zu 1.452.000 Byte bzw. 1,38 mB Daten erzeugt.

Pro Tag würde eine Datenmenge von bis zu 33,23 mB anfallen.

# Datenprotokolle

Das IoT-Device übermittelt die Daten per MQTT-Protokoll als UTF8-String an den Broker. Es hat sich herausgestellt, dass das MQTT-Protokoll zu langsam ist, um Daten in einer Geschwindigkeit von 200 Hz zu senden. Die Frequenz zur Messung wurde auf 100 Hz reduziert. Da die immer noch fünfmal so hoch wie die maximale Drehzahl der Waschmaschine ist, könnten trotzdem geeignete Messwerte ohne Aliasing erfasst werden.

# Analyse

Abb. 2‑1 zeigt den Plot aller Messdaten. Es wird deutlich, dass es unterschiedliche Zustände gibt.

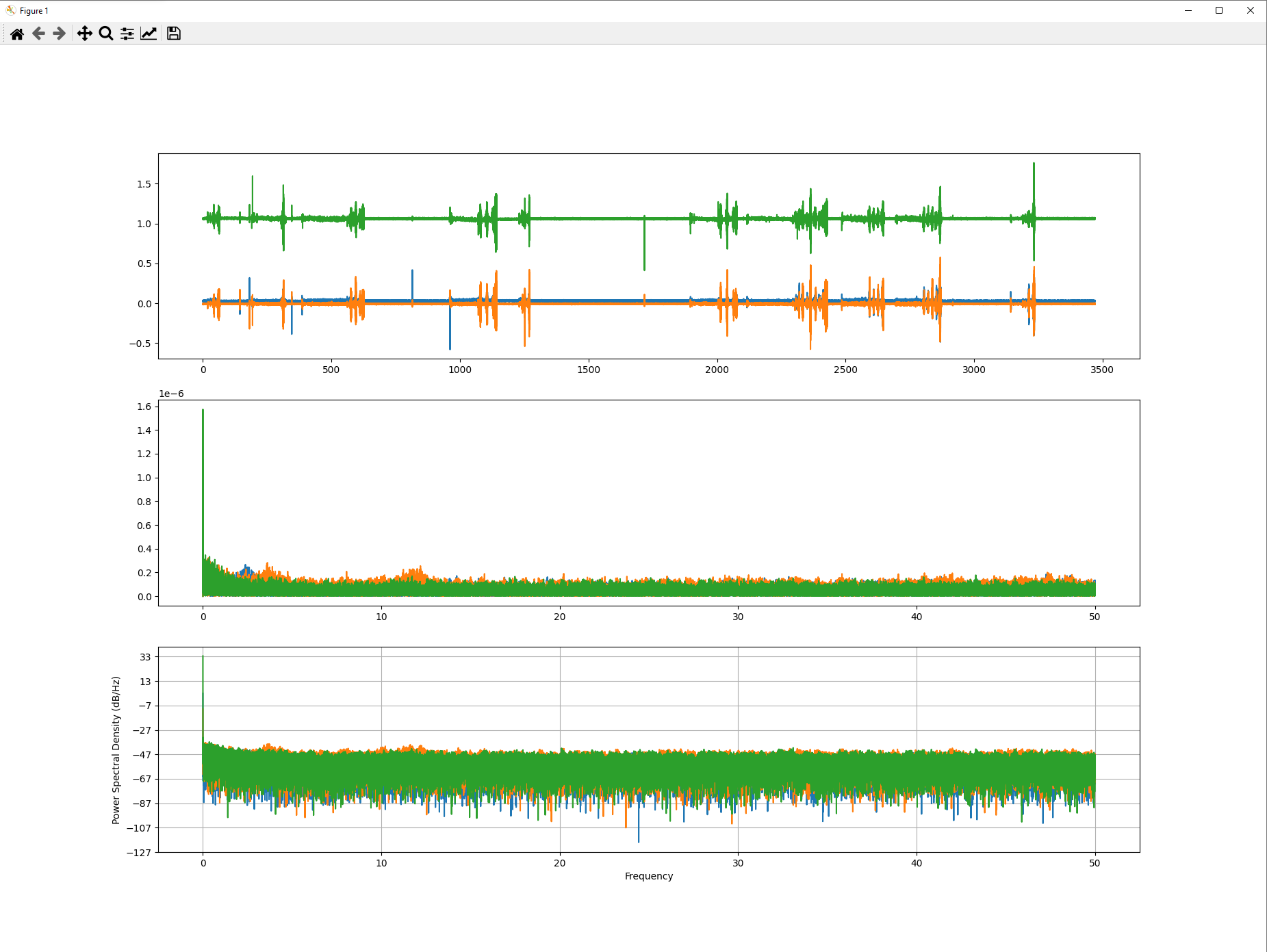
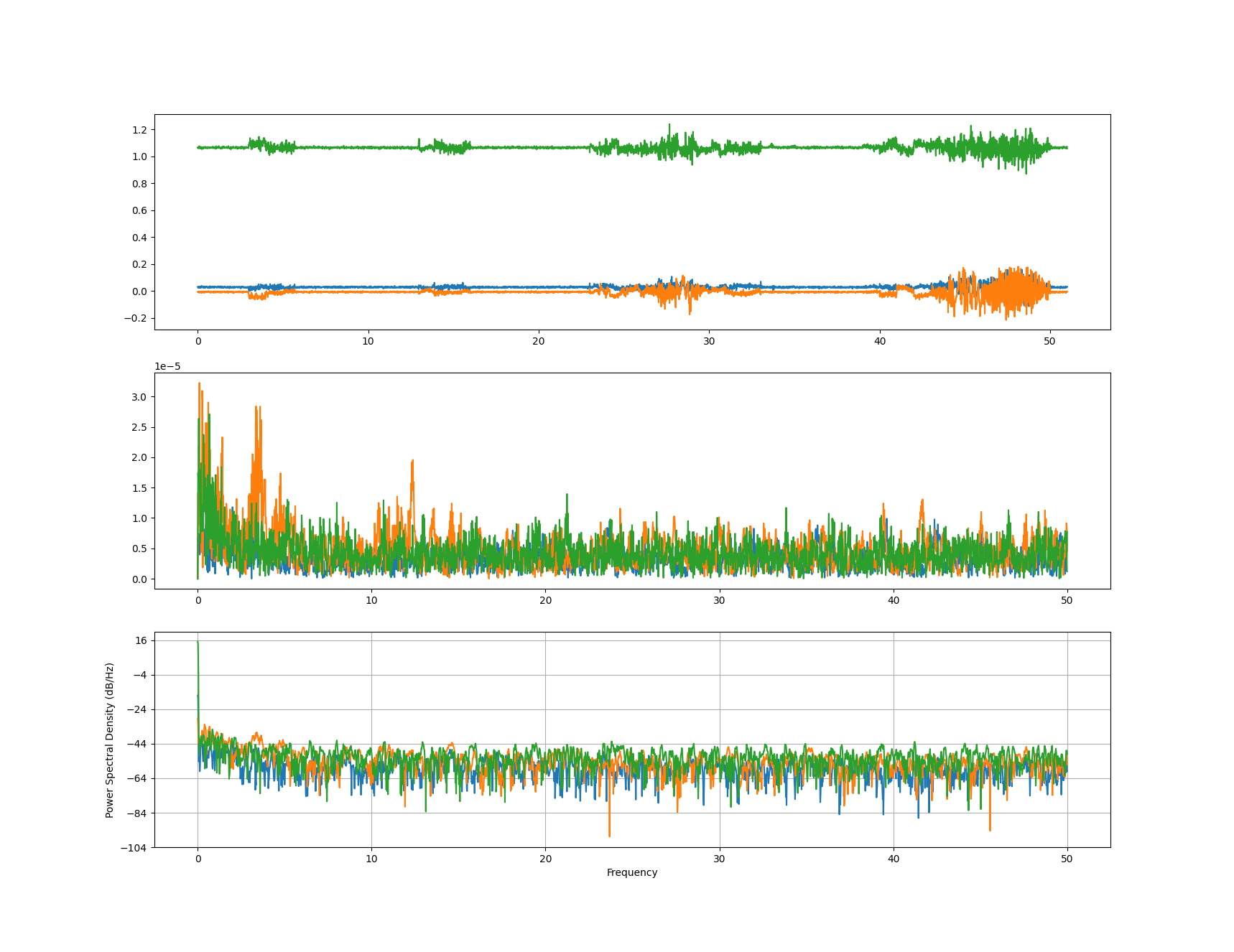


Abb. 6‑1 Plot der Messdaten

Abb. 6‑2 visualisiert die Messdaten eines Waschmaschinen-Zyklus. Es können vier unterschiedliche Zustände erkannt werden: (1) Leerlauf, (2) leichter Waschgang, (3) starker Waschgang und (4) Schleudergang.



(1)

(4)

(3)

(2)

Abb. 6‑2 Plot der Messdaten eines Waschmaschinen-Zyklus

Ein näherer Blick auf die Messdaten der markierten Bereiche stärkt den Eindruck. Wie in Abb. 6‑2 zu sehen ist, unterscheiden sich die Ausschläge der Messdaten der vier Betriebszustände. Der Leerlauf zeigt einen ruhigen Verlauf, der Schleudergang zeigt die stärksten Ausschläge.

|  |  |
| --- | --- |
| (1) Leerlauf |  |
| (2) leichter W. |  |
| (3) starker W. |  |
| (4) Schleudern |  |

Abb. 6‑3 Plot der Messdaten der vier Betriebszustände

In Abb. 6‑4 werden die Fast Fourier Transformationen der vier Betriebszustände dargestellt. Sie unterscheiden sich deutlich. Leerlauf und leichter Waschgang sehen sich zwar ähnlich, jedoch sind die Ausschläge des leichten Waschgangs im Verlauf stärker. Der starke Waschgang und der Schleudergang unterscheiden sich deutlich von den beiden vorherigen und voneinander.

|  |  |
| --- | --- |
| (1) Leerlauf |  |
| (2) leichter W. |  |
| (3) starker W. |  |
| (4) Schleudern |  |

Abb. 6‑4 Plot der Fast Fourier Transformationen der vier Betriebszustände

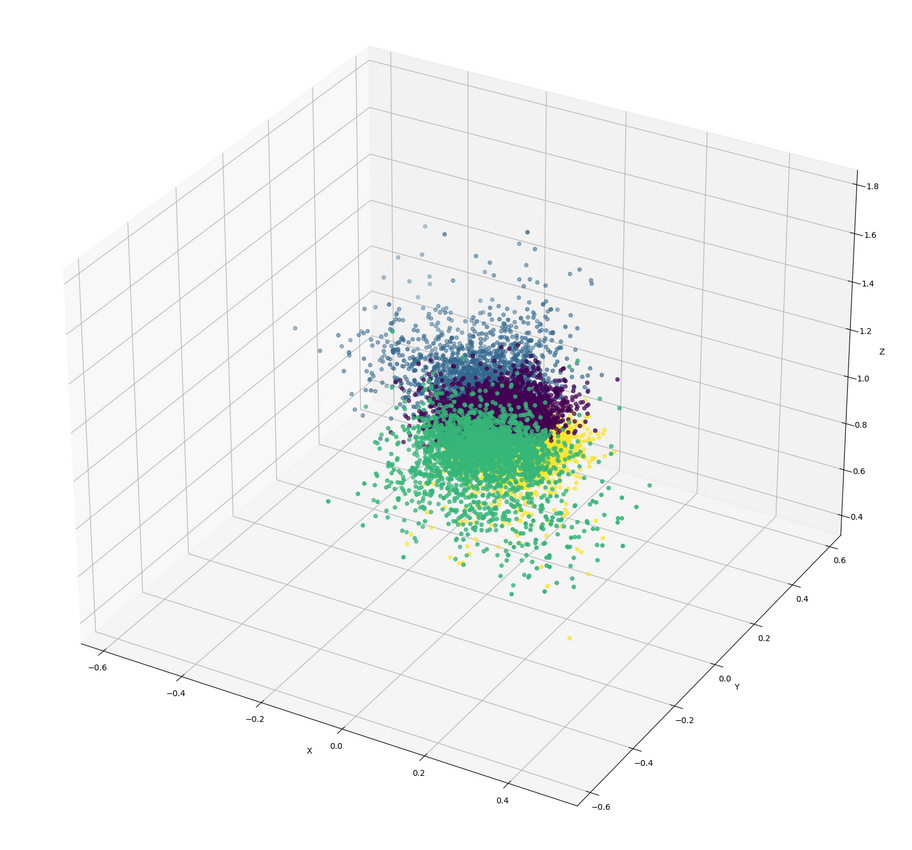
Die in Abb. 6‑5 dargestellten Power-Spectral-Density verstärkt den Eindruck, dass sich vier Betriebszustände unterscheiden lassen.

|  |  |
| --- | --- |
| (1) Leerlauf |  |
| (2) leichter W. |  |
| (3) starker W. |  |
| (4) Schleudern |  |

Abb. 6‑5 Plot der Power-Spectral-Density der vier Betriebszustände

# Klassifizierung

Zunächst werden die vorhandenenen Daten nach den 4 unterschiedlichen Betriebszuständen klassifiziert. Die Klassifizierung findet mittels KMeans statt. KMeans wird zunächst mit allen bisherigen Daten via KMeans(n\_clusters 0 4).fit(df\_train) trainiert. Abb. 7 1 veranschaulicht die Klassifizierung. Die Ergebnisse werden dem Dataframe als Feature status übergeben.



Zur Klassifizierung des aktuellen Betriebszustands soll das Multi-Layer Perceptron Classifier genutzt werden. Als Features werden die Beschleunigungswerte und der Timestamp übergeben. Zielwert ist der vorher mit KMeans festgestellte Zustand status. Via train\_test\_split() werden Trainings- und Testdatensätze erstellt. Sobald die Vorhersagen generiert sind, kann die Leistung des Modells bewertet werden. Hierzu wird die Konfusionsmatrix ausgegeben. Die Ausgabe zeigt die Leistung des Modells für Trainingsdaten. Wie Abb. 7 2 zeigt, liegen die Genauigkeit und der F1-Score zwischen etwa 0,90 und 1,00. Die Vorhersage ist also relativ genau.

Ein Bild, das Text, Anzeige, Gerät enthält.

Automatisch generierte Beschreibung

Die Leistung des Modells soll zudem anhand der Testdaten bewertet werden. Die Ausgabe in Abb. 7 3 zeigt die Leistung des Modells bei Testdaten. Die Genauigkeit und die F1-Werte liegen zwischen 0,92 und 1,00. Der Zustand der Waschmaschine kann also relativ präzise bestimmt werden.

Ein Bild, das Text, Gerät, Anzeige, draußen enthält.

Automatisch generierte Beschreibung

# Datenbankanbindung

Es kann sowohl ein lokaler MongoDB-Server oder eine MongoDB Atlas Cloud-Server genutzt werden. Es bietet sich an, einen Cloud-Server zu nutzen. Die Daten können jederzeit via Code 7‑1 aus der MongoDB Atlas Cloud-Datenbank exportiert werden.

cd C:\Program Files\MongoDB\Server\4.2\bin

mongoexport --uri mongodb+srv://test:test@cluster1337.kv1ih.mongodb.net/ProjectData --collection Data --out=C:\Users\Tony\Desktop\WaMaData.json

Code 7‑1 Daten aus MongoDB Atlas Cloud-Datenbank per Mongoexport exportieren

Die Daten werden als JSON-Datei auf dem Desktop abgelegt.

# Datenzugänglichkeit

Auf die Daten kann mit den folgenden Informationen zugegriffen werden:

|  |  |
| --- | --- |
| URI  Dataset  Collection | mongodb+srv://test:test@cluster1337.kv1ih.mongodb.net/ProjectData  ProjectData  Data |

# Benutzeroberfläche

Die Benutzeroberfläche wird mit Qt generiert. In einem ersten Schritt sollen die Daten vom MongoDB Atlas Cloud-Server abgerufen werden. Die Daten werden in einen Pandas Dataframe geschrieben, um effizient verarbeitet werden zu können.

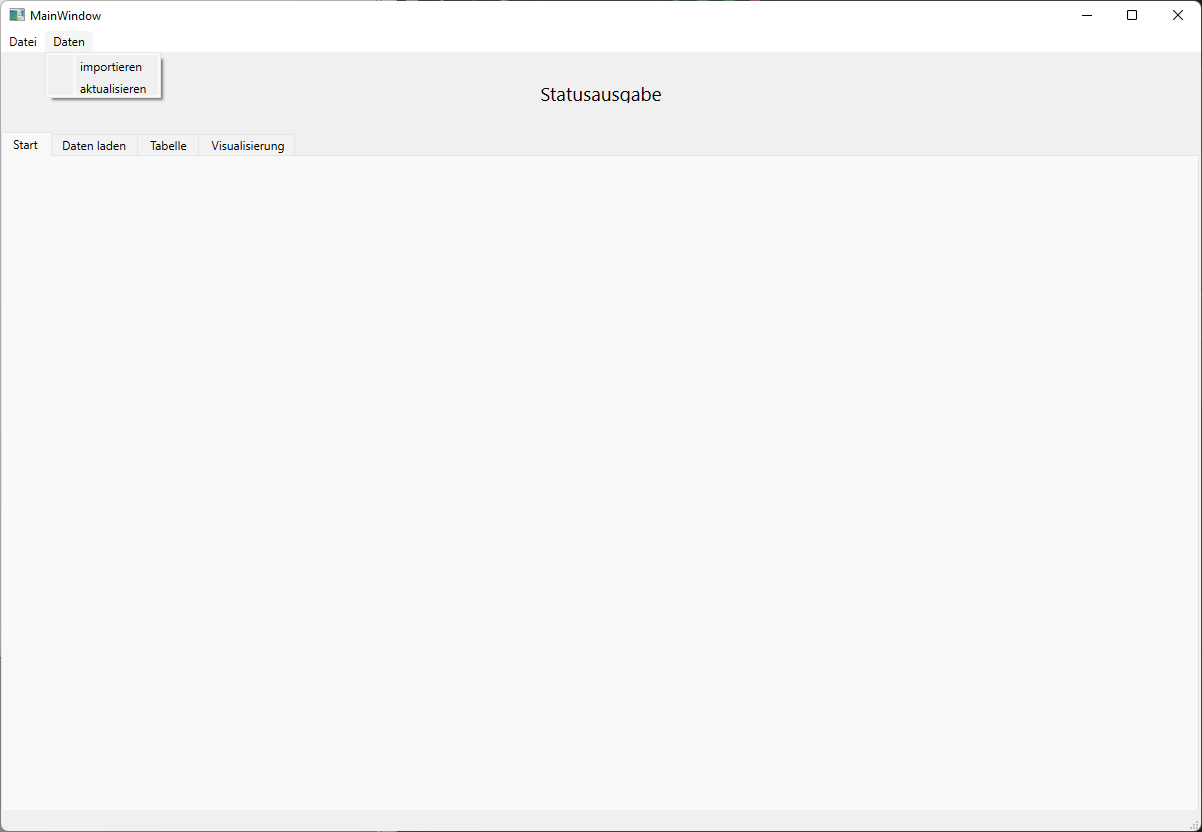


Abb. 9‑1 MainWindow des GUI - Daten importieren

Per *Daten* > *importieren* lassen sich die Daten vom MongoDB-Server abrufen (vgl. Abb. 9‑1). Die Statusausgabe gibt „Dataframe erfolgreich erstellt.“ aus, sobald das Dataframe erstellt ist (vgl. Abb. 9‑2).

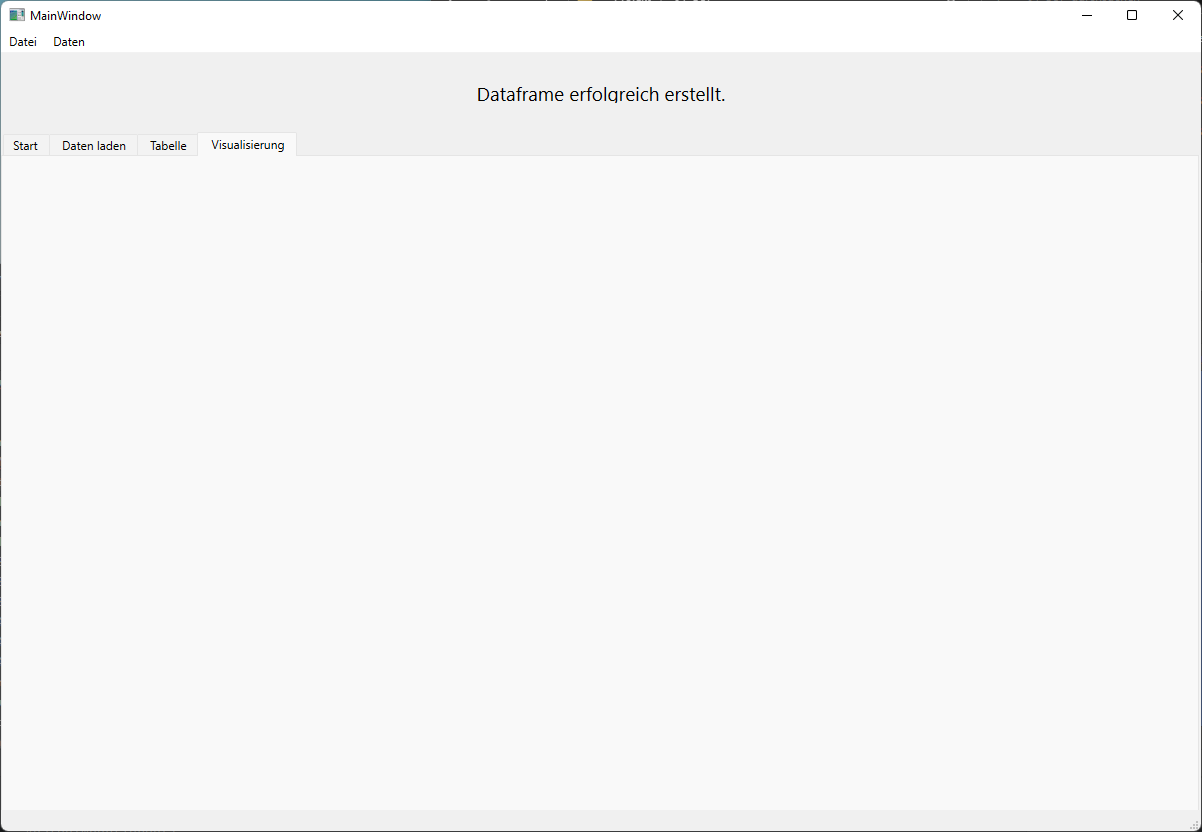


Abb. 9‑2 MainWindow des GUI - Dataframe erfolgreich erstellt

In einem nächsten Schritt sollen die Daten visualisiert werden. Der Nutzer mit per *Daten* > *aktualisieren* (vgl. Abb. 9‑1). die Möglichkeit haben, den Betrachtungszeitraum des Dataframes einzugrenzen. Dazu können Start und Ende des Betrachtungszeitraums in einem Dialogfenster mit Datum und Uhrzeit festgelegt werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abb. 9‑3 Dialogfenster zur Eingrenzung des Betrachtungszeitraums

Anhand der Daten soll das Dataframe gefiltert werden. Daraus sollen die Analysen zum Betrachtungszeitraum angestellt und visualisiert werden.