|  |  |  |
| --- | --- | --- |
|  | **POLITECHNIKA KRAKOWSKA im. T. Kościuszki**  Wydział Mechaniczny  **Katedra Pojazdów Szynowych i Transportu** |  |

Kierunek studiów: **Środki Transportu i Logistyka**

Specjalność: **Logistyka i Spedycja**

STUDIA STACJONARNE

**PRACA DYPLOMOWA**

MAGISTERSKA

inż. **Jakub Tomczyk**

Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów

The application of artificial neural networks in predicting random events during the transportation of goods

Promotor:

dr inż. **Augustyn Krzysztof Lorenc**

Kraków, rok akad. 2022/2023



Spis treści

[1. Cel i zakres prasy 10](#_Toc136849526)

[2. Wstęp 11](#_Toc136849527)

[3. Podstawy projektowe 12](#_Toc136849528)

[**3.1. Sztuczna inteligencja w logistyce** 12](#_Toc136849529)

[**3.2. Sztuczne sieci neuronowe** 13](#_Toc136849530)

[**3.3. Analiza i inżynieria danych** 14](#_Toc136849531)

[4. Założenia projektowe 15](#_Toc136849532)

[**4.1. Dane** 4](#_Toc136849533)

[**4.2. Oprogramowanie** 4](#_Toc136849534)

[5. Wykonanie projektu 6](#_Toc136849535)

[**5.1. Biblioteki Pythona** 6](#_Toc136849536)

[**5.1.1. NumPy** 6](#_Toc136849537)

[**5.1.2. Matplotlib** 7](#_Toc136849538)

[**5.1.3. Pandas** 7](#_Toc136849539)

[**5.1.4. TensorFlow** 8](#_Toc136849540)

[**5.2. Przygotowanie i analiza danych** 9](#_Toc136849541)

[**5.2.1. Temperatura** 11](#_Toc136849542)

[**5.2.2. Wilgotność** 15](#_Toc136849543)

[**5.2.3. Nasłonecznienie** 18](#_Toc136849544)

[**5.3. Sieć neuronowa** 22](#_Toc136849545)

[**5.4. Graficzne przedstawienie wyników** 4](#_Toc136849546)

[**5.4.1. Temperatura** 4](#_Toc136849547)

[**5.4.2. Wilgotność** 5](#_Toc136849548)

[**5.4.3. Nasłonecznienie** 6](#_Toc136849549)

[**5.5. Analiza wyników** 6](#_Toc136849550)

[6. Wnioski 7](#_Toc136849551)

[Literatura 8](#_Toc136849552)

[Summary 9](#_Toc136849553)

[Spis załączników 10](#_Toc136849554)

# 1. Cel i zakres prasy

Celem niniejszej pracy magisterskiej jest zaprojektowanie sztucznej sieci neuronowej, która będzie wykorzystana do prognozowania losowych zdarzeń podczas transportu towarów. Praca ma na celu dostarczenie rozwiązania dedykowanego firmom logistycznym i przewoźnikom, którzy transportują towary, na których stan mogą negatywnie wpływać zmiany temperatury, wilgotności i ekspozycji na słońce.

Praca składa się z dwóch głównych części: teoretycznej i praktycznej. Część teoretyczna skupia się na przedstawieniu teoretycznych aspektów poruszonych w pracy. Przedstawione zostały również przykłady praktycznych zastosowań sztucznej inteligencji w wiodących firmach logistycznych, co pozwoli na lepsze zrozumienie kontekstu i znaczenia projektu. Ponadto, dokładnie opisana została sztuczna sieć neuronowa, w tym zasady działania, różne typy warstw i funkcje aktywacji. Przedstawione zostaną również technologie wykorzystane przy projektowaniu sieci neuronowej, takie jak biblioteki w języku Python.

Część praktyczna pracy skupia się na zamodelowaniu konkretnej sieci neuronowej, która będzie w stanie przewidywać losowe zdarzenia podczas transportu towarów. Wykorzystane zostały dane dotyczące temperatury, wilgotności i ekspozycji na słońce. Następnie, przeprowadzony został trening i testowanie sieci neuronowej, a wyniki predykcji zostały zwizualizowane. Wizualizacja predykcji pozwoliła ocenić jakość predykcji.

W trakcie pracy magisterskiej wykorzystano język programowania Python 3, który zapewnia wiele narzędzi i bibliotek do implementacji sztucznych sieci neuronowych. Python 3 jest szeroko stosowany w dziedzinie uczenia maszynowego i zapewnia intuicyjny i efektywny sposób programowania sieci neuronowych jak również wiele innych bibliotek przy pracy z danymi.

# 2. Wstęp

Branża logistyczna odgrywa kluczową rolę w globalnej gospodarce. Wraz ze wzrostem globalizacji i e-commerce, zapotrzebowanie na usługi logistyczne stale rośnie, co niesie ze sobą coraz większe wyzwania. Jednym z najważniejszych wyzwań, przed którymi stoi branża logistyczna, jest zapewnienie niezawodności transportu towarów. Proces transportu towarów jest podatny na wiele losowych zdarzeń, które mogą wpływać na jego niezawodność, takie jak opóźnienia w dostawach, zdarzenia losowe na drodze, awarie sprzętu czy problemy związane z przechowywaniem towarów w odpowiednich warunkach podczas transportu. Przewóz żywności wymaga szczególnej uwagi ze względu na ograniczenia w zakresie temperatury, wilgotności oraz ekspozycji na słońce, aby utrzymać odpowiednie warunki przechowywania i zapobiec pogorszeniu jakości produktów. Dlatego precyzyjne prognozowanie zmian tych wartości podczas transportu staje się kluczowym elementem dla zapewnienia jakości i bezpieczeństwa żywności.

Aby sprostać tym wyzwaniom, coraz więcej firm logistycznych zaczyna wykorzystywać sztuczne sieci neuronowe. Są one jednym z filarów sztucznej inteligencji, umożliwiają analizę dużych zbiorów danych oraz generowanie prognoz na ich podstawie. Dzięki swojej zdolności do uczenia się na podstawie wzorców i adaptacji do zmieniających się warunków, sieci neuronowe mogą przewidywać przyszłe zdarzenia na podstawie dostępnych danych historycznych.

W niniejszej pracy magisterskiej skoncentrowano się na zaprojektowaniu sztucznej sieci neuronowej, która będzie w stanie przewidzieć zmianę wartości temperatury, wilgotności oraz ekspozycji na słońce. Owa sieć pozwoli na lepsze zarządzanie i monitorowanie transportu towarów, z uwzględnieniem specyficznych wymagań, zwłaszcza w przypadku transportu żywności.

Zainteresowanie sztucznymi sieciami neuronowymi w branży logistycznej rośnie w tempie ekspotencjalnym. Według raportu firmy Allied Market Research, globalny rynek sztucznej inteligencji w branży logistycznej wzrośnie z 1,7 miliardów dolarów w 2018 roku do 12 miliardów dolarów w 2027 roku, co stanowi roczny wzrost o 24% w okresie prognozowanym.

# 3. Podstawy projektowe

W tym rozdziale omówione zostały główne zagadnienia związane z tematem pracy. W pierwszym podrozdziale opisano ogólne zasady wykorzystania sztucznej inteligencji w logistyce, wraz z przykładami zastosowań w praktyce. Następnie przedstawiono podstawy teoretyczne sztucznych sieci neuronowych, w tym ich architekturę, proces trenowania oraz różne rodzaje sieci. W kolejnym podrozdziale poruszono tematykę analizy i inżynierii danych, w tym procesy przetwarzania i czyszczenia danych, a także budowanie modeli predykcyjnych.

## **3.1. Sztuczna inteligencja w logistyce**

Sztuczna inteligencja (AI) to dziedzina informatyki, która umożliwia maszynom i systemom komputerowym podejmowanie decyzji i wykonywanie zadań, które dotychczas wykonywane były tylko przez ludzi. W logistyce, gdzie skuteczne planowanie, zarządzanie łańcuchem dostaw i koordynacja działań są kluczowe dla efektywności i konkurencyjności przedsiębiorstw, sztuczna inteligencja odgrywa coraz większą rolę. Wśród najważniejszych zastosowań sztucznej inteligencji w logistyce należy wymienić:

* Predykcja popytu: dokładne przewidywanie zapotrzebowania na produkty, lub usługi, co pozwala przedsiębiorstwom na optymalizację procesów produkcyjnych i dostaw, zmniejszenie kosztów magazynowania oraz poprawę jakości obsługi klienta.
* Planowanie tras: dzięki sztucznej inteligencji możliwe jest skuteczne planowanie tras dostaw, uwzględniające wszelkie dostępne dane, takie jak ruch uliczny, informacje pogodowe, ograniczenia czasowe.
* Zarządzanie magazynami: optymalizację procesów magazynowych, w tym na szybkie i precyzyjne określenie najlepszych lokalizacji dla magazynów, zoptymalizowanie procesów kompletacji zamówień, a także przewidywanie zapotrzebowania na poszczególne produkty.
* Automatyzacja wielu procesów logistycznych, co pozwala na znaczne zmniejszenie kosztów i zwiększenie efektywności działań.

Przykłady konkretnych zastosowań sztucznej inteligencji w logistyce to m.in.:

* Wprowadzenie systemu AI w procesie planowania tras pozwoliło firmie UPS na zaoszczędzenie 8 milionów litrów paliwa rocznie oraz zmniejszenie emisji CO2 o ponad 20 tysięcy ton.
* Amazon wykorzystuje sztuczną inteligencję do skutecznego zarządzania magazynami, w tym do przewidywania zapotrzebowania na poszczególne produkty, optymalizacji procesów pakowania i kompletacji zamówień, a także do monitorowania stanu magazynowego.
* Maersk, jedna z największych firm transportowych na świecie, wykorzystuje sztuczną inteligencję do optymalizacji procesów logistycznych, w tym do predykcji popytu na transport morski, planowania tras oraz zarządzania magazynami.
* DHL wykorzystuje sztuczną inteligencję do optymalizacji procesów transportowych, w tym do planowania tras oraz do zarządzania flotą pojazdów.
* Coca-Cola wykorzystuje sztuczną inteligencję do predykcji popytu na produkty oraz do optymalizacji procesów magazynowych i transportowych.

Wprowadzenie sztucznej inteligencji do logistyki pozwala na znaczne zwiększenie efektywności działań, poprawę jakości obsługi klienta oraz znaczne obniżenie kosztów. W kontekście transportu towarów, zastosowanie sztucznych sieci neuronowych do przewidywania losowych zdarzeń pozwoli na jeszcze dokładniejsze i precyzyjniejsze planowanie działań, co przyczyni się do zwiększenia efektywności transportu towarów oraz minimalizację strat.

## **3.2. Sztuczne sieci neuronowe**

Sztuczne sieci neuronowe to modele matematyczne inspirowane funkcjonowaniem ludzkiego mózgu. Sieci neuronowe składają się z wielu połączonych ze sobą węzłów, zwanych neuronami, które przetwarzają informacje. Sieci te są zdolne do nauki i dostosowywania się do nowych danych, co czyni je bardzo skutecznymi w przewidywaniu losowych zdarzeń. Sztuczne sieci neuronowe mogą mieć różną architekturę i różne funkcje aktywacji. Najpopularniejsze rodzaje sieci neuronowych to:

* Perceptron wielowarstwowy (MLP),
* Sieci rekurencyjne (RNN),
* Sieci konwolucyjne (CNN).

Perceptron wielowarstwowy składa się z jednej lub wielu warstw neuronów, gdzie każdy neuron jest połączony z każdym neuronem z poprzedniej i następnej warstwy. MLP wykorzystywany jest w zadaniach klasyfikacji, przewidywania i regresji.

Sieci rekurencyjne wykorzystywane są w zadaniach przetwarzania języka naturalnego, tłumaczenia maszynowego oraz przetwarzania czasowego. RNN posiada połączenia zwrotne między neuronami, co pozwala na przetwarzanie sekwencji danych.

Sieci konwolucyjne wykorzystywane są w zadaniach przetwarzania obrazów i dźwięków. CNN wykorzystuje filtrację i warstwy konwolucyjne, które pozwalają na ekstrakcję cech z obrazów i dźwięków.

Przykłady zastosowań sieci neuronowych to:

* przewidywanie popytu na produkty w handlu detalicznym,
* klasyfikacja dokumentów w przetwarzaniu języka naturalnego,
* diagnozowanie chorób na podstawie badań medycznych,
* przewidywanie wyników meczów w sporcie,
* rozpoznawanie twarzy w systemach zabezpieczeń,
* sterowanie robotami w przemyśle.

W kontekście transportu towarów, sieci neuronowe mogą być wykorzystane nie tylko do przewidywania losowych zdarzeń podczas transportu towarów, takich jak zbyt wysoka temperatura, bądź wilgotność, lecz również do przewidywania opóźnień w dostawach, szacowania czasu potrzebnego na dostarczenie towarów czy optymalizacji tras transportowych.

## **3.3. Analiza i inżynieria danych**

Analiza i inżynieria danych to proces przetwarzania i wykorzystywania danych w celu uzyskania wiedzy oraz wyciągnięcia wniosków. Analiza danych umożliwia poznanie charakterystyki danych, ich wzorców oraz identyfikację problemów. Inżynieria danych, czyli proces projektowania, tworzenia i utrzymania struktury danych, pozwala na wykorzystanie tych informacji w procesie uczenia maszynowego.

W kontekście sztucznych sieci neuronowych, analiza i inżynieria danych są niezwykle ważne. Sieci neuronowe uczą się na podstawie zbioru danych, więc jakość i ilość danych mają bezpośredni wpływ na jakość modelu. Wprowadzenie niedokładnych lub błędnych danych może prowadzić do nieprawidłowych wniosków i wyników.

Przykłady zastosowania analizy i inżynierii danych w kontekście AI i sztucznych sieci neuronowych to:

* Przygotowanie danych do uczenia maszynowego, w tym czyszczenie, normalizacja i transformacja danych.
* Identyfikacja istotnych cech danych, które mają wpływ na wynik modelu.
* Optymalizacja procesu uczenia maszynowego, poprzez wykorzystanie odpowiednich algorytmów i technik.
* Monitorowanie i ocena modelu, w celu wykrycia błędów i poprawy dokładności modelu.

Podsumowując, analiza i inżynieria danych są kluczowe w procesie tworzenia i uczenia sztucznych sieci neuronowych. Poprawne przetwarzanie i wykorzystanie danych pozwala na uzyskanie dokładniejszych i bardziej efektywnych modeli sztucznej inteligencji, co jest niezwykle ważne w kontekście transportu towarów i innych dziedzin związanych z logistyką.

# 4. Założenia projektowe

Założeniem projektowym pracy magisterskiej "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" jest opracowanie efektywnego modelu sztucznej inteligencji, który będzie mógł przewidywać losowe zdarzenia podczas transportu towarów.

W celu osiągnięcia tego celu, w pracy zostaną wykorzystane sztuczne sieci neuronowe oraz analiza i inżynieria danych, aby zapewnić jak najdokładniejsze przewidywanie oraz minimalizację ryzyka związanego z transportem towarów.

Kolejnym założeniem projektowym jest wykorzystanie realnych danych transportowych, aby model sztucznej inteligencji był oparty na rzeczywistych danych i mógł być łatwiej zastosowany w praktyce. W ramach pracy zostaną przetworzone odpowiednie dane, aby były gotowe do wykorzystania w procesie uczenia maszynowego.

Ostatecznym celem projektu jest stworzenie skutecznego narzędzia, które będzie mogło być wykorzystane w praktyce w branży logistycznej, aby przewidywać losowe zdarzenia podczas transportu towarów i minimalizować ryzyko związanego z tym procesem.

## 

## **4.1. Dane**

W pracy magisterskiej "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" dane wejściowe są w dostarczone w formie pliku o formacie CSV (Comma Separated Values).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | **MeasurementDateTime** | **SensorId** | **ModuleCode** | **MeasurementInterval** | **ModuleName** | **Number of Records** | **Unit** | **Value** |
| 0 | 21.06.2020 19:03 | 602267e77f522d0007fb01cc | Temperature | 00:05:00 | SHT3X\_Temperature | 1 | °C | 21.5625 |
| 1 | 21.06.2020 19:08 | 602267e77f522d0007fb01cc | Temperature | 00:05:00 | SHT3X\_Temperature | 1 | °C | 21.5625 |
| 2 | 21.06.2020 19:13 | 602267e77f522d0007fb01cc | Temperature | 00:05:00 | SHT3X\_Temperature | 1 | °C | 21.5625 |
| 3 | 21.06.2020 19:18 | 602267e77f522d0007fb01cc | Temperature | 00:05:00 | SHT3X\_Temperature | 1 | °C | 21.5625 |
| 4 | 21.06.2020 19:23 | 602267e77f522d0007fb01cc | Temperature | 00:05:00 | SHT3X\_Temperature | 1 | °C | 21.5625 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Dane te zawierają informacje pomiarowe o różnych aspektach związanych z warunkami wewnątrz chłodni:

* ‘MeasurementDateTime’ – data i godzina zarejestrowania pomiaru,
* ‘SensorId’ – identyfikator sensora,
* ‘ModuleCode’ – rodzaj informacji zarejestrowanej przed sensor,
* ‘MeasurementInterval’ – odstęp czasowy pomiędzy pomiarami,
* ‘ModuleName’ – rodzaj sensora,
* ‘Number of Records’ – ilość rejestrów w jednym cyklu pomiarowym,
* ‘Value’ – zarejestrowana wartość.

Dane zostały zapisane w pliku CSV, który będzie wykorzystany jako wejście do modelu sztucznej inteligencji.

W celu zapewnienia poprawności i kompletności danych, zostały one poddane procesowi weryfikacji oraz walidacji. Weryfikacja danych polegała na sprawdzeniu, czy wszystkie wpisy są poprawnie sformatowane i czy nie zawierają błędów. Walidacja danych polegała na sprawdzeniu, czy dane są zgodne z rzeczywistością oraz czy nie występują w nich nieprawidłowości.

W pracy zostaną wykorzystane zaawansowane techniki analizy danych, takie jak uczenie maszynowe, aby opracować skuteczny model przewidywania losowych zdarzeń podczas transportu towarów. Wykorzystanie sztucznej inteligencji w połączeniu z rzeczywistymi danymi transportowymi ma na celu zapewnienie jak największej dokładności oraz minimalizacji ryzyka związanego z transportem towarów.

## **4.2. Oprogramowanie**

W pracy magisterskiej "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" do tworzenia modeli sztucznej inteligencji wykorzystany zostanie język programowania Python 3. Python to język interpretowany, wysokiego poziomu, który cieszy się popularnością w środowisku data science i uczenia maszynowego.

Jedną z najważniejszych zalet Pythona jest jego prostota i czytelność kodu. Dzięki temu programiści mogą tworzyć czytelne i łatwe do zrozumienia modele, które są intuicyjne dla użytkowników końcowych. Ponadto, Python posiada bogatą bibliotekę modułów i narzędzi, które ułatwiają pracę z danymi i uczeniem maszynowym. Wśród nich znajdują się m.in. NumPy, Pandas, Scikit-Learn, TensorFlow i Keras.

Python 3 jest również bardzo popularny w środowisku sztucznych sieci neuronowych. Dzięki narzędziom takim jak TensorFlow i Keras, można łatwo i szybko tworzyć i trenować modele sieci neuronowych. Python umożliwia tworzenie zarówno prostych, jak i skomplikowanych modeli, które można dopasować do specyficznych potrzeb projektu.

Python ma również zalety związane z jego otwartością i wsparciem społeczności. Dostępność otwartoźródłowych narzędzi i bibliotek umożliwia łatwe dzielenie się kodem i rozwiązaniami z innymi programistami na całym świecie. Dodatkowo, Python 3 jest aktywnie rozwijany i aktualizowany, co oznacza, że użytkownicy mogą liczyć na stałe udoskonalenia i nowe funkcjonalności.

Podsumowując, język Python 3 jest bardzo dobrym wyborem dla projektów związanych z uczeniem maszynowym i sztucznymi sieciami neuronowymi. Prostota i czytelność kodu, bogata biblioteka modułów i narzędzi oraz otwartoźródłowe podejście do rozwoju sprawiają, że Python 3 jest jednym z najpopularniejszych języków w środowisku data science i uczenia maszynowego.

# 5. Wykonanie projektu

Rozdział "Wykonanie projektu" to centralna część pracy magisterskiej "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów". W tym rozdziale przedstawiony zostanie proces tworzenia i trenowania modelu sztucznej sieci neuronowej oraz analiza wyników uzyskanych w procesie eksperymentu. Rozdział "Wykonanie projektu" jest kluczowy dla całej pracy magisterskiej, ponieważ to w nim zostanie przedstawiony proces tworzenia i trenowania modelu sztucznej sieci neuronowej oraz analiza uzyskanych wyników.

## **5.1. Biblioteki Pythona**

Podstawowy opis bibliotek wykorzystanych w pracy.

### **5.1.1. NumPy**

NumPy to biblioteka języka Python, która umożliwia wykonywanie operacji matematycznych i naukowych na tablicach oraz macierzach. Dzięki wykorzystaniu NumPy można w prosty sposób wykonywać zaawansowane obliczenia numeryczne w Pythonie.

Głównym elementem NumPy są wielowymiarowe tablice (ndarray). NumPy umożliwia tworzenie, modyfikowanie oraz manipulowanie tymi tablicami. Zaletą NumPy jest również to, że umożliwia on szybką i wydajną pracę z dużymi ilościami danych.

Do najważniejszych funkcji i metod biblioteki NumPy należą m.in.:

* Tworzenie tablic: NumPy umożliwia tworzenie tablic różnych kształtów i typów danych, w tym tablic jednowymiarowych, dwuwymiarowych, trójwymiarowych i więcej.
* Indeksowanie i wycinanie: NumPy umożliwia łatwe indeksowanie i wycinanie tablic, co pozwala na szybkie dostęp do poszczególnych elementów tablicy.
* Operacje matematyczne: NumPy umożliwia wykonywanie różnych operacji matematycznych na tablicach, takich jak dodawanie, odejmowanie, mnożenie, dzielenie i wiele innych.
* Agregowanie danych: NumPy umożliwia agregowanie danych na podstawie różnych funkcji, takich jak suma, średnia, maksimum czy minimum.

W projekcie "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" biblioteka NumPy została wykorzystana do przetwarzania i analizy danych numerycznych. Dzięki wykorzystaniu NumPy można w prosty sposób wykonywać operacje na dużych zbiorach danych, co jest kluczowe w kontekście sztucznej inteligencji i uczenia maszynowego.

### **5.1.2. Matplotlib**

Matplotlib to biblioteka języka Python, która umożliwia tworzenie wykresów, histogramów, diagramów i innych wizualizacji danych. Matplotlib umożliwia tworzenie wysokiej jakości wykresów w prosty sposób, dzięki czemu jest bardzo popularna wśród naukowców, inżynierów i analityków danych.

Do najważniejszych funkcji i metod biblioteki Matplotlib należą m.in.:

* Tworzenie wykresów: Matplotlib umożliwia tworzenie różnych typów wykresów, takich jak wykresy liniowe, słupkowe, kołowe i punktowe.
* Personalizacja wykresów: Matplotlib umożliwia personalizowanie wykresów w zależności od potrzeb, takie jak zmiana kolorów, stylów linii i etykiet osi.
* Wyświetlanie danych w czasie rzeczywistym: Matplotlib umożliwia wyświetlanie danych w czasie rzeczywistym, co jest bardzo przydatne w przypadku monitorowania procesów w czasie rzeczywistym.
* Wyświetlanie wielu wykresów na raz: Matplotlib umożliwia wyświetlanie wielu wykresów na jednej płaszczyźnie, co pozwala na łatwe porównywanie różnych zestawów danych.

W projekcie "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" biblioteka Matplotlib została wykorzystana do przedstawienia wyników przeprowadzonych analiz. Dzięki wykorzystaniu Matplotlib możliwe jest łatwe i szybkie wyświetlanie danych w postaci graficznej, co pozwala na szybkie zrozumienie i interpretację wyników.

### **5.1.3. Pandas**

Pandas to biblioteka języka Python, która umożliwia łatwe i efektywne manipulowanie danymi w formacie tabelarycznym. Biblioteka pandas pozwala na łatwe wczytywanie i zapisywanie danych w różnych formatach, takich jak CSV, Excel, SQL, JSON i wiele innych.

Do najważniejszych funkcji i metod biblioteki pandas należą m.in.:

* Wczytywanie i zapisywanie danych: pandas umożliwia łatwe wczytywanie i zapisywanie danych w różnych formatach, co pozwala na łatwe przetwarzanie i analizowanie danych.
* Manipulowanie danymi: pandas umożliwia manipulowanie danymi w różny sposób, takim jak sortowanie, filtrowanie, grupowanie i łączenie.
* Przetwarzanie brakujących danych: pandas umożliwia łatwe przetwarzanie brakujących danych, co pozwala na łatwe uzupełnienie i analizowanie danych.
* Wizualizacja danych: pandas umożliwia wizualizowanie danych w formie wykresów i grafów, co pozwala na łatwe zrozumienie i interpretację danych.

W projekcie "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" biblioteka pandas została wykorzystana do przetwarzania danych i przygotowania ich do analizy. Dzięki wykorzystaniu pandas możliwe jest łatwe wczytanie danych w formacie CSV oraz przeprowadzenie różnych operacji na danych, takich jak filtrowanie i grupowanie. Biblioteka pandas pozwala na łatwe i efektywne przetwarzanie danych w formacie tabelarycznym, co jest niezbędne w procesie analizy danych w projekcie AI i sztucznych sieci neuronowych.

### **5.1.4. Keras**

Biblioteka Keras jest jednym z najpopularniejszych narzędzi do budowania i trenowania modeli sieci neuronowych w języku Python. Zaprojektowana w celu zapewnienia prostoty i intuicyjnego interfejsu, Keras umożliwia użytkownikom tworzenie zaawansowanych modeli uczenia maszynowego bez konieczności zgłębiania skomplikowanych detali implementacyjnych. Poniżej przedstawiam rozszerzony opis biblioteki Keras i jej kluczowych modułów.

Modele: Keras umożliwia budowanie modeli sieci neuronowych za pomocą klasy Sequential lub funkcjonalnego interfejsu API. Klasa Sequential pozwala na tworzenie sekwencyjnych modeli, gdzie warstwy są dodawane jedna po drugiej. Funkcjonalne API pozwala na budowanie bardziej złożonych modeli, takich jak modele z wieloma wejściami i wyjściami lub modele z warstwami o nieliniowych połączeniach.

Warstwy: Keras oferuje różnorodne typy warstw, które mogą być stosowane w modelach sieci neuronowych. Warstwa Dense (w pełni połączona) jest najbardziej podstawową warstwą, w której każdy neuron jest połączony z każdym neuronem z poprzedniej warstwy. Istnieją również inne warstwy, takie jak Convolutional Layers (warstwy konwolucyjne), Recurrent Layers (warstwy rekurencyjne) i wiele innych, które umożliwiają modelom uczenia maszynowego analizowanie różnych typów danych i wykrywanie złożonych wzorców.

Optymalizatory: Keras dostarcza różne optymalizatory, które mogą być wykorzystane podczas procesu treningu modelu. Optymalizatory, takie jak RMSprop, Adam, czy SGD (stochastic gradient descent), dostosowują wagi sieci neuronowej w celu minimalizacji funkcji kosztu. Użytkownik może dostosować parametry optymalizatora, takie jak współczynnik uczenia, czy tempo uczenia adaptacyjnego, w celu zoptymalizowania procesu uczenia.

Funkcje kosztu: określają, jak dobrze model przewiduje dane na podstawie rzeczywistych wartości. Keras zapewnia różne funkcje kosztu, takie jak mean squared error (MSE), binary cross-entropy, czy categorical cross-entropy, które są odpowiednie dla różnych typów problemów, takich jak regresja, klasyfikacja binarna lub wieloklasowa.

Metryki: służą do oceny wydajności modelu. Metryki, takie jak accuracy (dokładność), precision (precyzja) , recall (czułość), czy F1-score, dostarczają informacji na temat jakości predykcji modelu na podstawie prawdziwych wartości danych testowych.

Biblioteka Keras umożliwia elastyczne i łatwe tworzenie, trenowanie i ocenianie modeli sieci neuronowych. Jej intuicyjny interfejs oraz wsparcie dla różnorodnych warstw, optymalizatorów i funkcji kosztu czynią z niej potężne narzędzie do eksperymentowania i rozwijania zaawansowanych modeli uczenia maszynowego.

W projekcie "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" wykorzystano następujące moduły:

* keras.models.Sequential: reprezentuje liniową sekwencję warstw, która pozwala na budowanie modeli sieci neuronowych warstwa po warstwie. W dostarczonym kodzie został utworzony model sekwencyjny, do którego można kolejno dodawać warstwy,
* keras.layers.Dense: w pełni połączona warstwa, w której każdy neuron jest połączony ze wszystkimi neuronami z poprzedniej warstwy.

## **5.2. Przygotowanie i analiza danych**

W projekcie "Zastosowanie sztucznych sieci neuronowych w przewidywaniu losowych zdarzeń podczas transportu towarów" biblioteka Keras została wykorzystana do tworzenia i szkolenia modelu sztucznej sieci neuronowej. Dzięki wykorzystaniu Keras możliwe jest łatwe tworzenie zaawansowanych modeli AI i sztucznych sieci neuronowych, co jest niezbędne do przewidywania losowych zdarzeń podczas transportu towarów. Keras umożliwia szybkie szkolenie modelu, co pozwala na osiągnięcie wysokiej wydajności i dokładności modelu.

- Krok pierwszy, wczytanie bibliotek oraz danych:

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

from matplotlib import cm

df = pd.read\_csv('measurements\_join\_eda.csv', sep=';', decimal=',')

Do wykonania zadań związanych z przygotowaniem oraz analizą danych wykorzystano trzy biblioteki: pandas, NumPy oraz Matplotlib. Przy użyciu metody read\_csv z biblioteki pandas przypisano dataset do zmiennej df. Wskazano seperator oraz znak oddzielający części całkowite od ułamkowych.

- Krok drugi, Exploratory Data Analysis:

for col in df.columns:

    print('Column name:', col)

    print('\t Count of distinct values: ', len(df[col].unique()))

    print('\t Has null: ', df[col].isnull().values.any())

    if len(df[col].unique()) < 10:

        print('\t Distinct Values in column', df[col].unique())

Stworzono pętlę, która tworzy krótki raport dotyczący odrębnych oraz NULL wartości dla każdej z kolumn. Dodano warunek, aby pętla nie tworzyła listy odrębnych wartości dla kolumn: „MeasurementDateTime” oraz „SensorId”, ponieważ ich liczebność jest bardzo duża i nie dostarczy nam żadnych wartościowych informacji.

Column name: MeasurementDateTime

Count of distinct values: 46683

Has null: False

Column name: SensorId

Count of distinct values: 25

Has null: False

Column name: ModuleCode

Count of distinct values: 6

Has null: False

Distinct Values in column ['Temperature' 'VOC' 'Light' 'Humidity' 'Acceleration' 'Pressure']

Column name: MeasurementInterval

Count of distinct values: 2

Has null: False

Distinct Values in column ['00:05:00' '00:10:00']

Column name: ModuleName

Count of distinct values: 6

Has null: False

Distinct Values in column ['SHT3X\_Temperature' 'BME680\_VOC' 'ISL29035\_Light' 'SHT3X\_Humidity'

'LIS2DH\_Acceleration' 'BME680\_Pressure']

Column name: Number of Records

Count of distinct values: 1

Has null: False

Column name: Unit

Count of distinct values: 6

Has null: False

Column name: Value

Count of distinct values: 938

Has null: False

Z raportu wywnioskowano, że zbiór danych nie zawiera żadnych NULL wartości. Kolumny „ModuleCode”, „ModuleName”, „Unit” zawierają dokładnie tyle samo odrębnych wartości. Bazując na tej obserwacji oraz analizie tabeli wywnioskowano, że owe kolumny reprezentują tę samą informację. Należy usunąć dwie z trzech kolumn, aby zminimalizować rozmiar tabeli oraz zwiększyć przejrzystość. Wartości w kolumnie „Number of Records” dla wszystkich rejestrów wynoszą „1”. Nie jest to wartościowa informacja w kontekście analizy oraz sieci neuronowej. Kolumna „MeasurementInterval” również została usunięta, ponieważ aby przedstawić zmianę rejestrowanych wartości w czasie wykorzystano kolumnę „MeasurementDateTime”.

df.drop(['Number of Records', 'ModuleCode', 'ModuleName', 'MeasurementInterval'], inplace=True, axis=1)

Usunięto kolumny. Przy użyciu metody .info() uzyskano informacje odnośnie ilości rejestrów oraz typie danych.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 468953 entries, 0 to 468952

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 468953 non-null object

1 SensorId 468953 non-null object

2 Unit 468953 non-null object

3 Value 468953 non-null float64

dtypes: float64(1), object(3)

memory usage: 14.3+ MB

Zbiór danych posiada 468953 zarejestrowanych odczytów. Kolumny „MeasurementDateTime”, „SensorId”, “Unit” zawierają dane obiektowe. Jako dane obiektowe biblioteki pandas należy rozumieć tekst, lub mieszane numeryczne i nienumeryczne wartości. Kolumna „Value” zawiera dane w formacie liczb rzeczywistych.

- Krok trzeci, zmiana typu danych:

df['MeasurementDateTime'] = pd.to\_datetime(df['MeasurementDateTime'], dayfirst=True)

df = df.sort\_values(by='MeasurementDateTime')

W tym kroku zmieniono typ danych w kolumnie „MeasurementDateTime” z obiektowych na datowe oraz posortowano tabelę, aby zapisy z sensorów zaczynały się od najstarszych. Na tym etapie pracy tabela wygląda w przedstawiony poniżej sposób:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MeasurementDateTime** | **SensorId** | **Unit** | **Value** |
| 343551 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | Lux | 0.000000 |
| 340979 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | hPa | 236.355896 |
| 351243 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | % | 98.669998 |
| 351767 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | °C | 2.500000 |
| 289563 | 2020-06-07 13:00:00 | 602267e87f522d0007fb01e3 | % | 33.930000 |
| ... | ... | ... | ... | ... |

### **5.2.1. Temperatura**

Na początku przeanalizowano zmianę temperatury. W związku z tym wyfiltrowano odczyty dla których wartość w kolumnie „Unit” to „°C”.

df\_temp = df.loc[df.Unit == '°C']

df\_temp.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 123871 entries, 351767 to 1768

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 123871 non-null datetime64[ns]

1 SensorId 123871 non-null object

2 Unit 123871 non-null object

3 Value 123871 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 4.7+ MB

Zaobserwowano, że spośród wszystkich 468953 odczytów, 123871 zawiera informacje o temperaturze wewnątrz pojemnika chłodzącego. Sprawdzono, czy każdy z 25 sensorów przeprowadzał zapis temperatury, poprzez odczyt średniej zarejestrowanej temperatury dla każdego sensora.

df\_temp\_group = df\_temp.groupby(['SensorId']).mean()

Value

SensorId

602267e77f522d0007fb01ca 4.932607

602267e77f522d0007fb01cb 4.931107

602267e77f522d0007fb01cc 20.935456

602267e77f522d0007fb01cd 4.935001

602267e77f522d0007fb01ce 4.834896

602267e77f522d0007fb01cf 4.132562

602267e77f522d0007fb01d0 3.962662

602267e77f522d0007fb01d1 1.999463

602267e77f522d0007fb01d2 16.007420

602267e77f522d0007fb01d3 20.920984

602267e77f522d0007fb01d5 1.876689

602267e87f522d0007fb01d6 20.091500

602267e87f522d0007fb01d7 2.152099

602267e87f522d0007fb01d8 2.353918

602267e87f522d0007fb01da 20.045310

602267e87f522d0007fb01db 19.908002

602267e87f522d0007fb01dc 2.542276

602267e87f522d0007fb01dd 3.175481

602267e87f522d0007fb01df -2.406760

602267e87f522d0007fb01e1 -2.208603

602267e87f522d0007fb01e2 20.627365

602267e87f522d0007fb01e3 20.610280

602267e87f522d0007fb01e4 -1.355914

602267e87f522d0007fb01e6 -3.767295

602267e87f522d0007fb01e7 23.161892

Średnia temperatura waha się od -4 do 5 stopni Celcjusza. Jest jednak 9 sensorów, dla których zarejestrowana temperatura jest niepokojąco wysoka, zważywszy na pożądane warunki przewozu, wedle których temperatura powinna utrzymywać się poniżej 0. Wywnioskowano, że te sensory rejestrowały temperaturę w momencie składowania pustych pojemników i nie dostarczają one wartościowych informacji na temat temperatury podczas przewozu towarów, a nawet mogę negatywnie wpłynąć na wyniki predykcji. Zdecydowano się usunąć zapisy temperatury z tych sensorów.

df\_temp = df\_temp[~df\_temp['SensorId'].isin(['602267e77f522d0007fb01cc', '602267e77f522d0007fb01d2', '602267e77f522d0007fb01d3', '602267e87f522d0007fb01d6', '602267e87f522d0007fb01da', '602267e87f522d0007fb01db', '602267e87f522d0007fb01e2', '602267e87f522d0007fb01e3', '602267e87f522d0007fb01e7'])]

df\_temp.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 77968 entries, 351767 to 468952

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 77968 non-null datetime64[ns]

1 SensorId 77968 non-null object

2 Unit 77968 non-null object

3 Value 77968 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 3.0+ MB

Pozostało 77968 zapisów temperatury. Wcześniej posortowanej tabeli pod względem chronologii nadano nowe indeksy.

df\_temp = df\_temp.reset\_index()

df\_temp.drop('index', axis=1, inplace=True)

df\_temp.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MeasurementDateTime** | **SensorId** | **Unit** | **Value** |
| 0 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | °C | 2.5000 |
| 1 | 2020-06-07 13:01:00 | 602267e87f522d0007fb01e4 | °C | 3.1250 |
| 2 | 2020-06-07 13:02:00 | 602267e77f522d0007fb01d1 | °C | 1.8750 |
| 3 | 2020-06-07 13:02:00 | 602267e77f522d0007fb01d0 | °C | 3.1250 |
| 4 | 2020-06-07 13:03:00 | 602267e77f522d0007fb01d5 | °C | 3.4375 |
| ... | ... | ... | ... | ... |

Wizualizacja przebiegu zmiany temperatury w czasie.

plt.figure(figsize=(15, 4))

plt.title("Temperature all data points", fontsize=16)

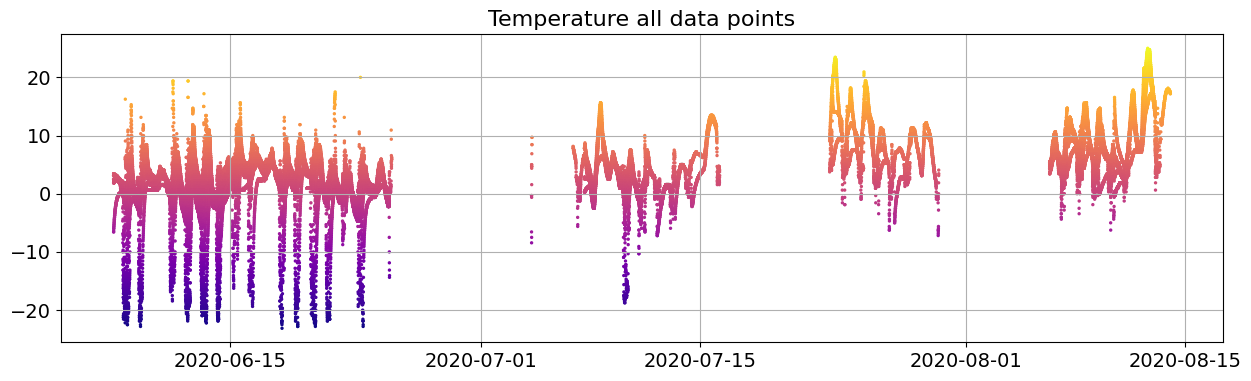
plt.scatter(df\_temp['MeasurementDateTime'], df\_temp['Value'], c = df\_temp['Value'], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Widać wyraźnie 4 okresy użytkowania pojemnika. Do predykcji wykorzystano jedynie zapisy z pierwszego okresu użytkowania, ponieważ wiadomo, że w tym okresie pojemnik był wykorzystywany do przewozu towarów mrożonych, gdzie temperatura powinna wynosić mniej niż 0 stopni Celcjusza. W dwóch ostatnich okresach użytkowania pojemnik był prawdopodobnie wykorzystywany do przewozu towarów fresh, gdzie temperatura powinna być zachowana w przedziale od 2 do 6 stopni Celcjusza. Jest to jednak jedynie założenie.

df\_temp = df\_temp[df\_temp['MeasurementDateTime'] < '2020-07-01']

plt.figure(figsize=(15, 4))

plt.title("Temperature all data points", fontsize=16)

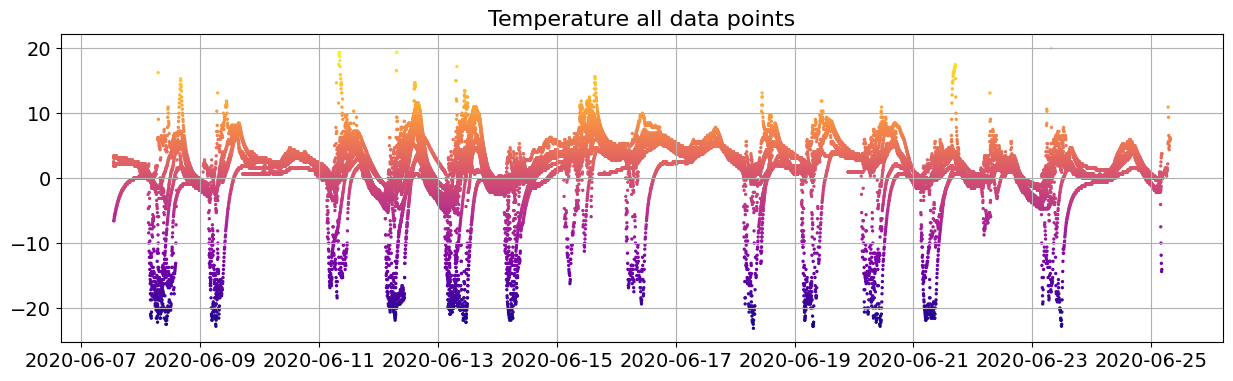
plt.scatter(df\_temp['MeasurementDateTime'], df\_temp['Value'], c = df\_temp['Value'], cmap = 'plasma', s = 2, lw = 1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Na załączonym wyżej wykresie widać wyraźnie cykle transportowe. Dwa pierwsze cykle transportowe zostały użyte jako treningowy zbiór danych dla siecii neuronowej.

Tp = 4750

plt.figure(figsize=(15, 4))

plt.title("Temperature of first {} data points".format(Tp), fontsize=16)

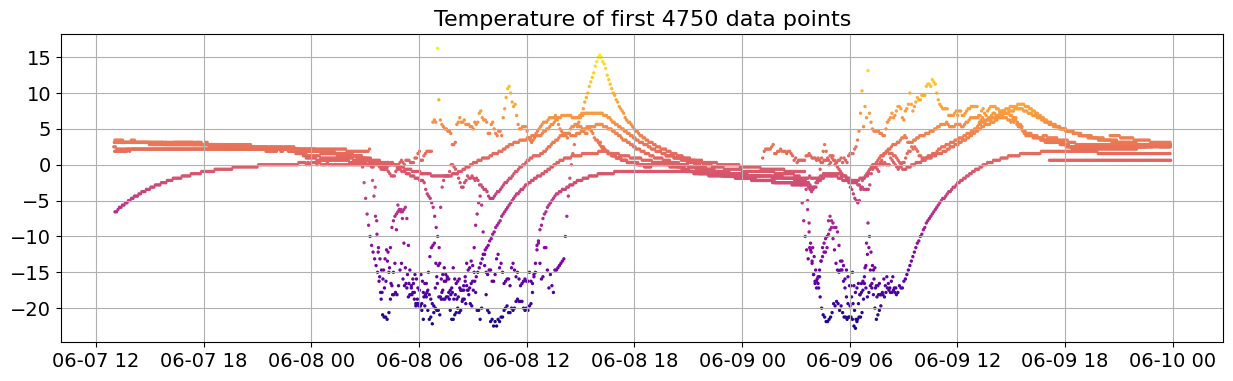
plt.scatter(df\_temp['MeasurementDateTime'][:Tp], df\_temp['Value'][:Tp], c = df\_temp['Value'][:Tp], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



### **5.2.2. Wilgotność**

Po przeanalizowaniu zarejestrowanych temperatur, przeanalizowano zmianę wilgotności podczas transportu. W związku z tym wyfiltrowano odczyty dla których wartość w kolumnie „Unit” to „%”.

df\_hum = df.loc[df.Unit == '%']

df\_hum.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 97133 entries, 351243 to 1480

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 97133 non-null datetime64[ns]

1 SensorId 97133 non-null object

2 Unit 97133 non-null object

3 Value 97133 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 3.7+ MB

Zaobserwowano, że spośród wszystkich 468953 odczytów, 97133 zawiera informacje o wilgotności wewnątrz pojemników chłodzących. Podobnie jak w przypadku analizy temperatury, sprawdzono, czy każdy z 25 sensorów przeprowadzał zapis poziomu wilgotności, poprzez odczyt średniej zarejestrowanej wilgotności dla każdego sensora.

df\_hum\_group = df\_hum.groupby(['SensorId']).mean()

print(df\_hum\_group)

Value

SensorId

602267e77f522d0007fb01ca 85.074490

602267e77f522d0007fb01cb 81.510436

602267e77f522d0007fb01cc 30.961556

602267e77f522d0007fb01cd 77.808280

602267e77f522d0007fb01ce 78.984658

602267e77f522d0007fb01cf 77.777028

602267e77f522d0007fb01d0 73.450989

602267e77f522d0007fb01d1 74.316431

602267e77f522d0007fb01d2 67.012326

602267e77f522d0007fb01d3 30.189449

602267e87f522d0007fb01df 82.938846

602267e87f522d0007fb01e1 92.752230

602267e87f522d0007fb01e2 31.220814

602267e87f522d0007fb01e3 30.724314

602267e87f522d0007fb01e4 90.101383

602267e87f522d0007fb01e6 89.319674

602267e87f522d0007fb01e7 38.344582

17 ze wszystkich 25 sensorów rejestrowało poziom wilgotności wewnątrz pojemników. 5 nich wykazuje znacznie niższą średnią wilgotność od pozostałych. Usunięto odczyty z tych sensorów, aby nie zaburzyły wyników predykcji.

df\_hum = df\_hum[~df\_hum['SensorId'].isin(['602267e77f522d0007fb01cc', '602267e77f522d0007fb01d3', '602267e87f522d0007fb01e2', '602267e87f522d0007fb01e3', '602267e87f522d0007fb01e7'])]

df\_hum.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 65301 entries, 351243 to 468776

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 65301 non-null datetime64[ns]

1 SensorId 65301 non-null object

2 Unit 65301 non-null object

3 Value 65301 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 2.5+ MB

Po usunięci niepokojących danych zostało 65301 z 97133 wszystkich rejestrów wilgotności. Nadano nowe indexy zapisom.

df\_hum = df\_hum.reset\_index()

df\_hum.drop('index', axis=1, inplace=True)

df\_hum.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MeasurementDateTime** | **SensorId** | **Unit** | **Value** |
| 0 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | % | 98.669998 |
| 1 | 2020-06-07 13:01:00 | 602267e87f522d0007fb01e4 | % | 83.459999 |
| 2 | 2020-06-07 13:01:00 | 602267e77f522d0007fb01d2 | % | 60.059998 |
| 3 | 2020-06-07 13:02:00 | 602267e77f522d0007fb01d0 | % | 65.909996 |
| 4 | 2020-06-07 13:02:00 | 602267e77f522d0007fb01d1 | % | 72.540001 |
| ... | ... | ... | ... | ... |

Wizualizacja przebiegu zmiany wilgotności w czasie.

plt.figure(figsize=(15, 4))

plt.title("Humidity all data points", fontsize=16)

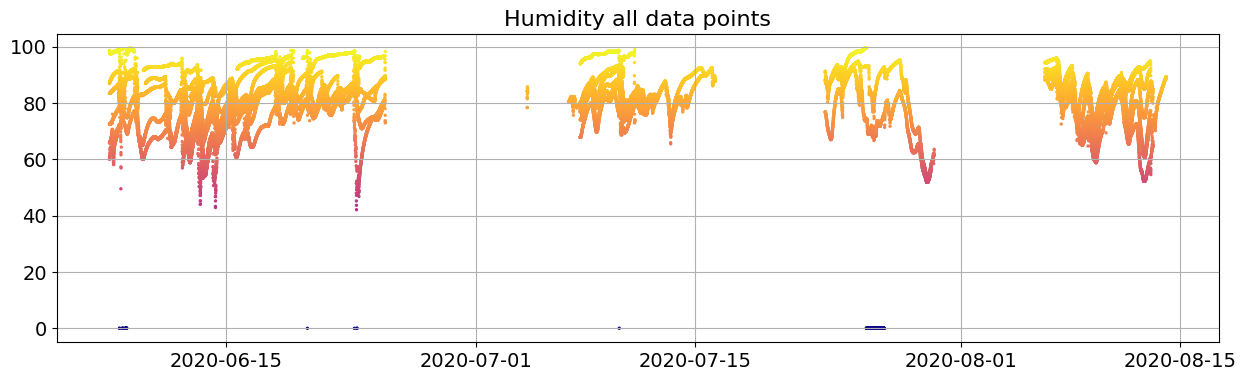
plt.scatter(df\_hum['MeasurementDateTime'], df\_hum['Value'], c = df\_hum['Value'], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Widać wyraźnie 4 okresy użytkowania pojemnika. Do predykcji wykorzystano jedynie zapisy z pierwszego okresu użytkowania, ponieważ wiadomo, że w tym okresie czas, wiadomo był cel wykorzystania pojemników – przewóz towarów mrożonych.

df\_hum = df\_hum[df\_hum['MeasurementDateTime'] < '2020-07-01']

plt.figure(figsize=(15, 4))

plt.title("Humidity all data points", fontsize=16)

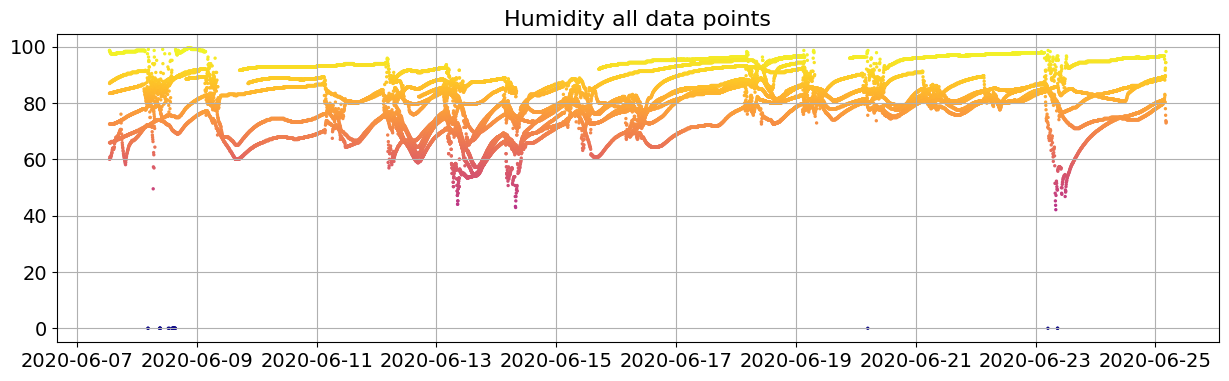
plt.scatter(df\_hum['MeasurementDateTime'], df\_hum['Value'], c = df\_hum['Value'], cmap = 'plasma', s = 2, lw = 1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Na załączonym wyżej wykresie widać wyraźnie cykle transportowe. Dwa pierwsze cykle transportowe zostały użyte jako treningowy zbiór danych dla siecii neuronowej.

Tp = 4750

plt.figure(figsize=(15, 4))

plt.title("Humidity of first {} data points".format(Tp), fontsize=16)

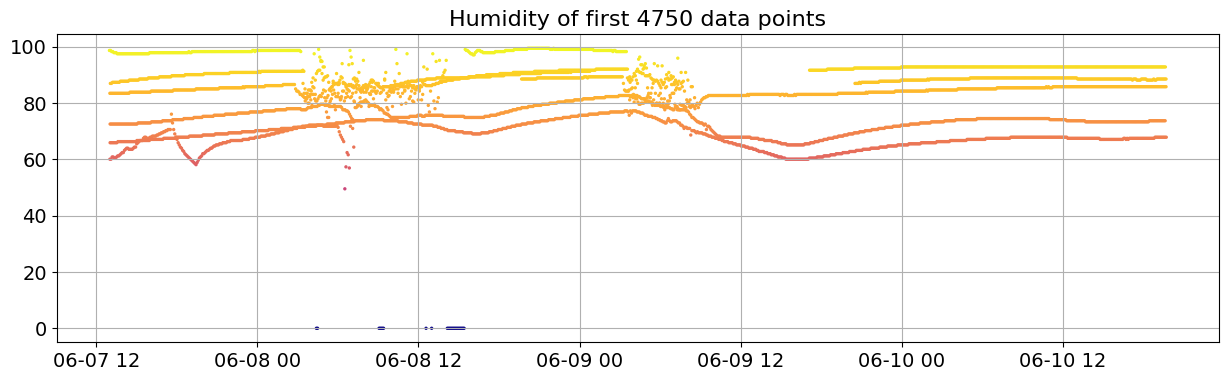
plt.scatter(df\_hum['MeasurementDateTime'][:Tp], df\_hum['Value'][:Tp], c = df\_hum['Value'][:Tp], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



### **5.2.3. Nasłonecznienie**

Jak ostatnie, przeanalizowano poziom nasłonecznienia wewnątrz pojemnika podczas transportu. W związku z tym wyfiltrowano odczyty dla których wartość w kolumnie „Unit” to „Lux”.

df\_lux = df.loc[df.Unit == 'Lux']

df\_lux.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 123699 entries, 343551 to 1192

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 123699 non-null datetime64[ns]

1 SensorId 123699 non-null object

2 Unit 123699 non-null object

3 Value 123699 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 4.7+ MB

Zaobserwowano, że spośród wszystkich 468953 odczytów, 123699 zawiera informacje o nasłonecznieniu wewnątrz pojemników chłodzących. Sprawdzono, czy każdy z 25 sensorów przeprowadzał zapis poziomu wilgotności, poprzez odczyt średniej zarejestrowanej wilgotności dla każdego sensora.

df\_lux\_group = df\_lux.groupby(['SensorId']).mean()

print(df\_lux\_group)

Value

SensorId

602267e77f522d0007fb01ca 1.707234

602267e77f522d0007fb01cb 1.666969

602267e77f522d0007fb01cc 1.204482

602267e77f522d0007fb01cd 0.544603

602267e77f522d0007fb01ce 0.591245

602267e77f522d0007fb01cf 0.411700

602267e77f522d0007fb01d0 0.724208

602267e77f522d0007fb01d1 0.956370

602267e77f522d0007fb01d2 13.119999

602267e77f522d0007fb01d3 5.964296

602267e77f522d0007fb01d5 2.008216

602267e87f522d0007fb01d6 8.368784

602267e87f522d0007fb01d7 15.472803

602267e87f522d0007fb01d8 15.767505

602267e87f522d0007fb01da 1.848674

602267e87f522d0007fb01db 8.358942

602267e87f522d0007fb01dc 21.288149

602267e87f522d0007fb01dd 11.997642

602267e87f522d0007fb01df 0.109870

602267e87f522d0007fb01e1 0.020666

602267e87f522d0007fb01e2 0.077108

602267e87f522d0007fb01e3 0.000000

602267e87f522d0007fb01e4 0.161315

602267e87f522d0007fb01e6 0.528550

602267e87f522d0007fb01e7 0.000000

Każdy z 25 sensorów rejestrował poziom nasłonecznienia wewnątrz pojemników. 5 nich wykazuje znacznie wyższe średnie nasłonecznienie od pozostałych. Usunięto odczyty z tych sensorów, aby nie zaburzyły wyników predykcji.

df\_lux = df\_lux[~df\_lux['SensorId'].isin(['602267e77f522d0007fb01d2', '602267e87f522d0007fb01d7', '602267e87f522d0007fb01d8', '602267e87f522d0007fb01dc', '602267e87f522d0007fb01dd'])]

df\_lux.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 111277 entries, 343551 to 1192

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MeasurementDateTime 111277 non-null datetime64[ns]

1 SensorId 111277 non-null object

2 Unit 111277 non-null object

3 Value 111277 non-null float64

dtypes: datetime64[ns](1), float64(1), object(2)

memory usage: 4.2+ MB

Po usunięci niepokojących danych zostało 111277 z 123699 wszystkich rejestrów nasłonecznienia.

df\_lux = df\_lux.reset\_index()

df\_lux.drop('index', axis=1, inplace=True)

df\_lux.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MeasurementDateTime** | **SensorId** | **Unit** | **Value** |
| 0 | 2020-06-07 12:59:00 | 602267e87f522d0007fb01e6 | Lux | 0.000000 |
| 1 | 2020-06-07 13:00:00 | 602267e87f522d0007fb01e3 | Lux | 0.000000 |
| 2 | 2020-06-07 13:00:00 | 602267e87f522d0007fb01e2 | Lux | 0.000000 |
| 3 | 2020-06-07 13:01:00 | 602267e77f522d0007fb01d3 | Lux | 17.039999 |
| 4 | 2020-06-07 13:01:00 | 602267e87f522d0007fb01db | Lux | 23.519999 |
| ... | ... | ... | ... | ... |

Wizualizacja przebiegu zmiany nasłonecznienia w czasie.

plt.figure(figsize=(15, 4))

plt.title("Exposure all data points", fontsize=16)

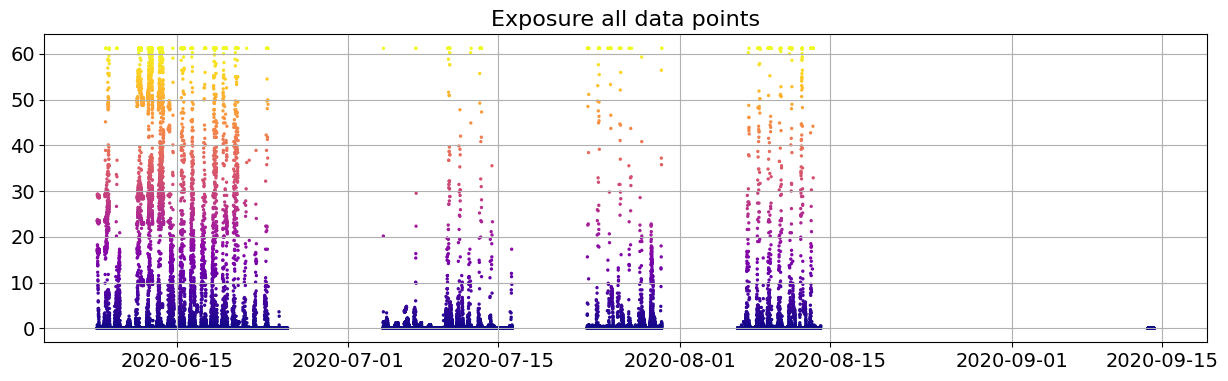
plt.scatter(df\_lux['MeasurementDateTime'], df\_lux['Value'], c = df\_lux['Value'], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Widać wyraźnie 4 okresy użytkowania pojemnika. Do predykcji wykorzystano jedynie zapisy z pierwszego okresu użytkowania, ponieważ wiadomo, że w tym okresie czas, wiadomo był cel wykorzystania pojemników – przewóz towarów mrożonych.

df\_lux = df\_lux[df\_lux['MeasurementDateTime'] < '2020-07-01']

plt.figure(figsize=(15, 4))

plt.title("Exposure all data points", fontsize=16)

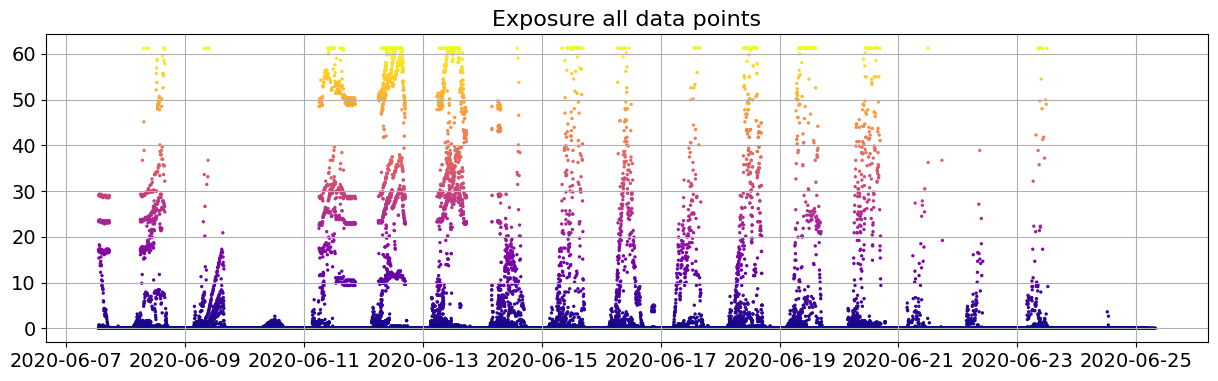
plt.scatter(df\_lux['MeasurementDateTime'], df\_lux['Value'], c = df\_lux['Value'], cmap = 'plasma', s = 2, lw = 1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



Na załączonym wyżej wykresie widać wyraźnie cykle transportowe. Dwa pierwsze cykle transportowe zostały użyte jako treningowy zbiór danych dla sieci neuronowej.

Tp = 4750

plt.figure(figsize=(15, 4))

plt.title("Exposure of first {} data points".format(Tp), fontsize=16)

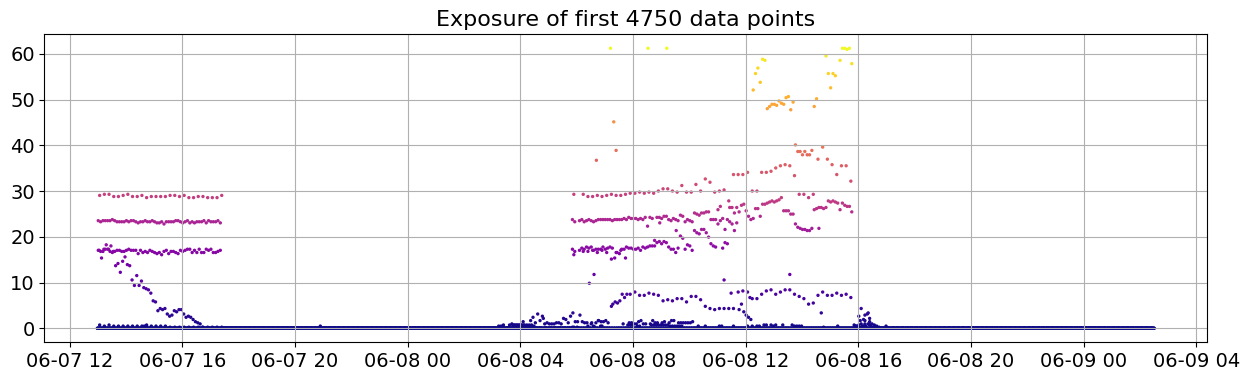
plt.scatter(df\_lux['MeasurementDateTime'][:Tp], df\_lux['Value'][:Tp], c = df\_lux['Value'][:Tp], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()



## **5.3. Sieć neuronowa**

W zależności od badanego parametru, w wierszu 22 zmieniano symbol w kolumnie „Unit”. „°C” odpowiada temperaturze, „%” odpowiada wilgotności, „Lux” odpowiada nasłonecznieniu. Zmieniano również nagłówki wykresów w wierszach: 45, 55 oraz 139. W wierszu 39 również w zależności od badanego parametru wykluczano inne sensory.

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN

from keras.optimizers import RMSprop

from keras.callbacks import Callback

# Wczytanie danych z pliku CSV.

df = pd.read\_csv('measurements\_join\_eda.csv', sep=';', decimal=',')

# Usunięcie niepotrzebnych kolumn.

df.drop(['Number of Records', 'ModuleCode', 'MeasurementInterval', 'ModuleName'], inplace=True, axis=1)

# Zmiana typu danych na datetime.

df['MeasurementDateTime'] = pd.to\_datetime(df['MeasurementDateTime'], dayfirst=True)

# Sortowanie danych względem daty.

df = df.sort\_values(by='MeasurementDateTime')

# Wybranie zapisów, w których jednostką jest stopieć Celsjusza.

df = df.loc[df.Unit == '°C']

# Średnie wartości dla poszczególnych czujników.

df\_group = df.groupby(['SensorId']).mean()

print(df\_group)

# Wykluczenie czujników, których uwzględnienie zaburzyłoby wyniki predykcji.

df = df[~df['SensorId'].isin(['602267e77f522d0007fb01cc', '602267e77f522d0007fb01d2', '602267e77f522d0007fb01d3', '602267e87f522d0007fb01d6', '602267e87f522d0007fb01da', '602267e87f522d0007fb01db', '602267e87f522d0007fb01e2', '602267e87f522d0007fb01e3', '602267e87f522d0007fb01e7'])]

# Ustalenie indexu na nowo.

df = df.reset\_index()

df.drop('index', axis=1, inplace=True)

# Wybranie rekordów pierwszego okresu użytkowania.

df = df[df['MeasurementDateTime'] < '2020-07-01']

# Określenie liczby punktów treningowych jako Tp.

Tp = 4750

# Wykres treningowych punktów pomiarowych.

plt.figure(figsize=(15, 4))

plt.title("Temperature of first {} data points".format(Tp), fontsize=16)

plt.scatter(df['MeasurementDateTime'][:Tp], df['Value'][:Tp], c = df['Value'][:Tp], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()

# Wykres wszystkich punktów pomiarowych.

plt.figure(figsize=(15, 4))

plt.title("Temperature all data points", fontsize=16)

plt.scatter(df['MeasurementDateTime'], df['Value'], c = df['Value'], cmap = 'plasma', s = 2, lw=1)

plt.grid(True)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()

# Przygotowanie danych treningowych i testowych.

train = np.array(df['Value'][:Tp])

test = np.array(df['Value'][Tp:])

train = train.reshape(-1, 1)

test = test.reshape(-1, 1)

step = 8

# Funkcja pomocnicza do konwersji danych na macierze.

def convertToMatrix(data, step):

    X, Y = [], []

    for i in range(len(data)-step):

        d = i+step

        X.append(data[i:d,])

        Y.append(data[d,])

    return np.array(X), np.array(Y)

# Funkcja do budowy modelu RNN.

def build\_simple\_rnn(num\_units=128, embedding=4, num\_dense=32, lr=0.001):

    model = Sequential()

    model.add(SimpleRNN(units=num\_units, input\_shape=(1, embedding), activation="relu"))

    model.add(Dense(num\_dense, activation="relu"))

    model.add(Dense(1))

    model.compile(loss='mean\_squared\_error', optimizer=RMSprop(lr=lr), metrics=['mse'])

    return model

# Klasa, która informuje o zakończeniu danych ecykli obliczeń.

class MyCallback(Callback):

    def on\_epoch\_end(self, epoch, logs=None):

        if (epoch+1) % 50 == 0 and epoch > 0:

            print("Epoch number {} done".format(epoch+1))

# Dodanie elementów o długości 'step' do danych treningowych i testowych.

test = np.append(test, np.repeat(test[-1,], step))

train = np.append(train, np.repeat(train[-1,], step))

# Konwersja danych treningowych i testowych na macierze.

trainX, trainY = convertToMatrix(train, step)

testX, testY = convertToMatrix(test, step)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# Inicjalizacja modelu sieci neuronowej.

model\_temp = build\_simple\_rnn(num\_units=128, num\_dense=32, embedding=8, lr=0.001)

batch\_size = 50

num\_epochs = 2000

# Trenowanie modelu sieci neuronowej.

model\_temp.fit(trainX, trainY, epochs=num\_epochs, batch\_size=batch\_size, callbacks=[MyCallback()], verbose=0)

# Wykres wartości RMSE w zależności od epoch.

plt.figure(figsize=(7, 5))

plt.title("RMSE loss over epochs", fontsize=16)

plt.plot(np.sqrt(model\_temp.history.history['loss']), c='k', lw=2)

plt.grid(True)

plt.xlabel("Epochs", fontsize=14)

plt.ylabel("Root-mean-squared error", fontsize=14)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.show()

# Predykcje na danych treningowych i testowych.

trainPredict = model\_temp.predict(trainX)

testPredict = model\_temp.predict(testX)

predicted = np.concatenate((trainPredict, testPredict), axis=0)

index = df.index.values

# Porównanie rzeczywistych wartości do predykcji.

plt.figure(figsize=(15, 5))

plt.title("Temperature: Ground truth and prediction together", fontsize=18)

plt.plot(df['MeasurementDateTime'], df['Value'], c='blue')

plt.plot(df['MeasurementDateTime'], predicted, c='orange', alpha=0.75)

plt.legend(['True data', 'Predicted'], fontsize=15)

plt.grid(True)

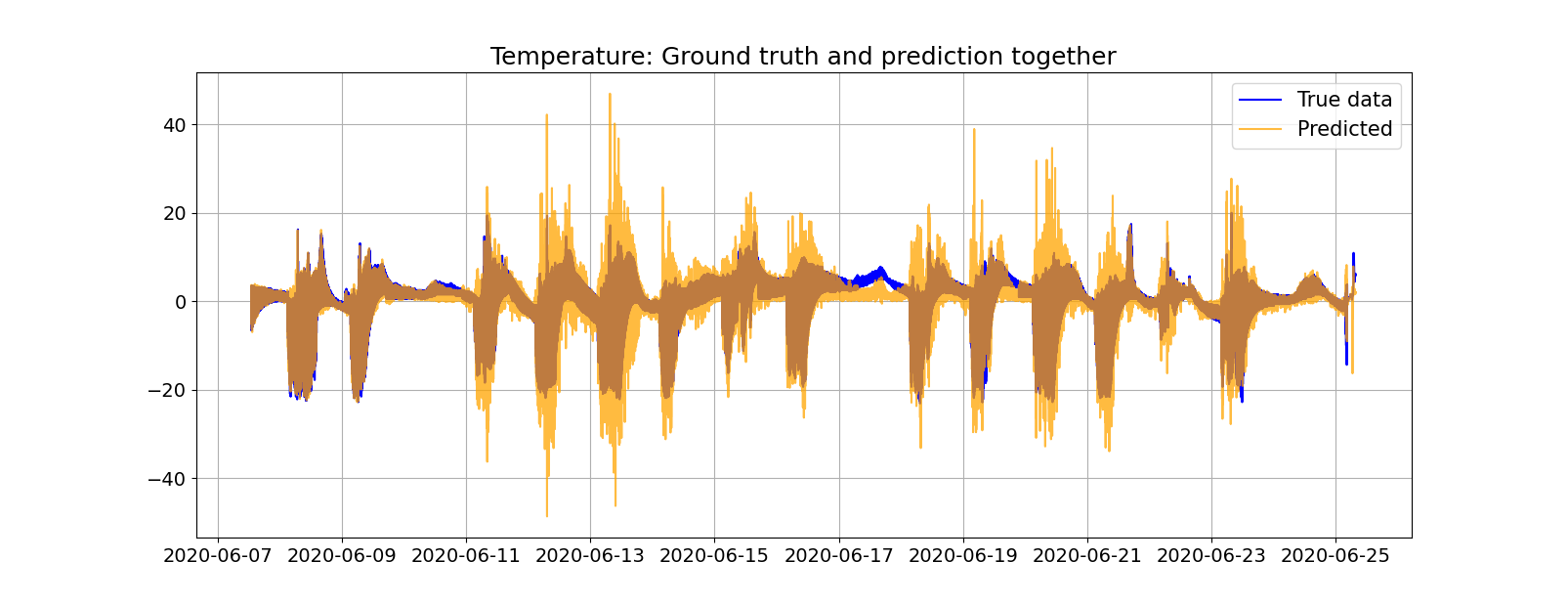
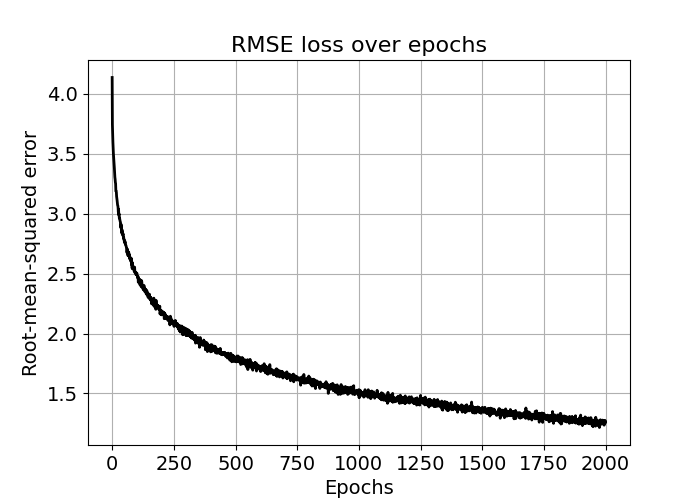
plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

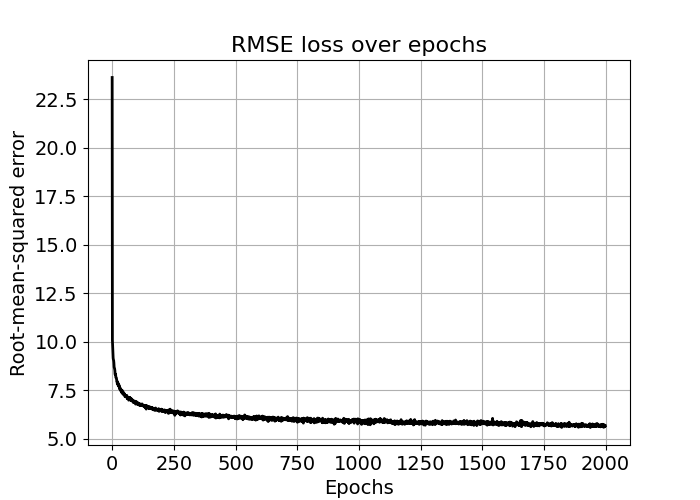
plt.show()

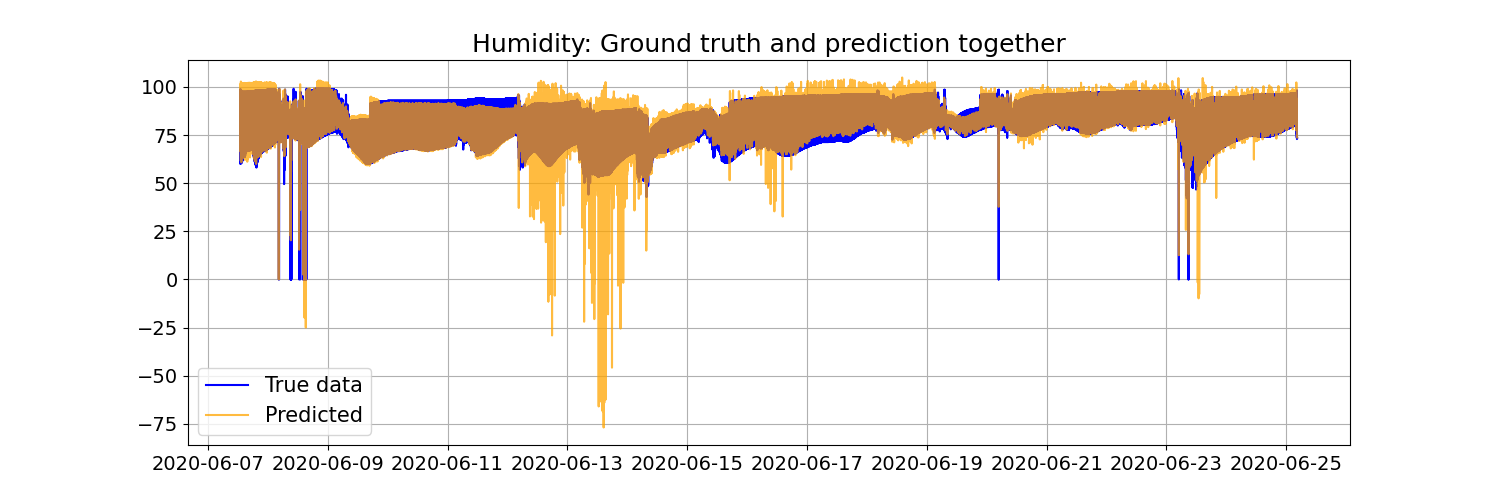
## **5.4. Graficzne przedstawienie wyników**

### **5.4.1. Temperatura**

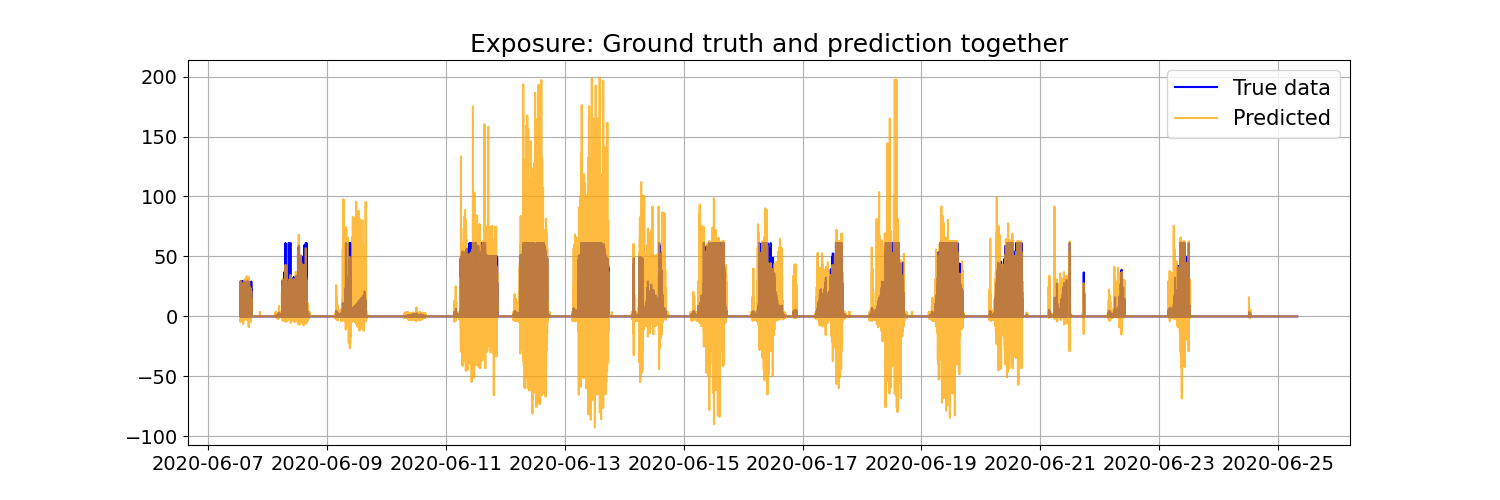
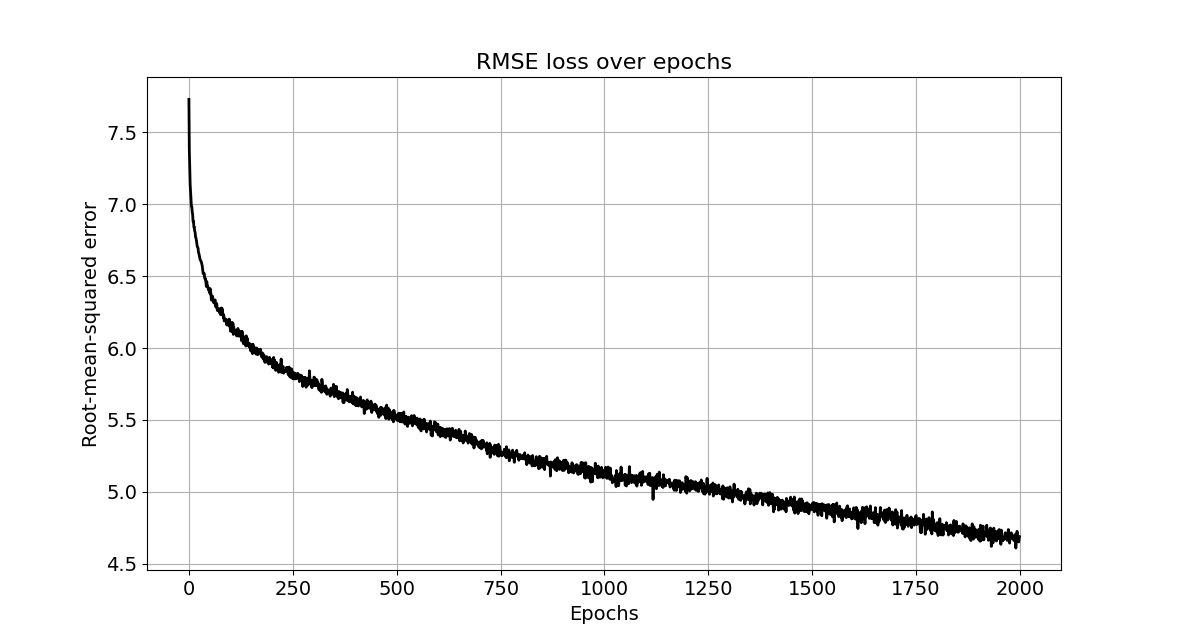


### **5.4.2. Wilgotność**





### **5.4.3. Nasłonecznienie**



## **5.5. Analiza wyników**

Na wykresach zmiany RMSE (Root mean squared error) widać, że początkowo wartości RMSE maleją wraz z rosnącą liczbą epoch treningowych. Jednak po osiągnięciu pewnego punktu, wartości RMSE stabilizują się i nie mają większych zmian, co oznacza, że dalsze szkolenie modelu nie przyniesie znaczącej poprawy wyników. Ostateczne wartości RMSE są na satysfakcjonującym poziomie, co wskazuje, że model ma dużą zdolność do dokładnego przewidywania temperatury.

Na wykresach przedstawiających prawdziwe wartości wraz z predykcjami modelu widać, że w dużej mierze owe wartości się pokrywają. W większości obszarów obie wartości są niemal identyczne, co wskazuje na dobrą jakość predykcji modelu.

# 6. Wnioski

Podsumowując, model sieci neuronowej w przedstawionym kodzie wykazuje obiecujące rezultaty i jest w stanie przewidywać temperaturę, wilgotność oraz nasłonecznienie na podstawie dostępnych danych. Analizując wykresy prawdziwych wartości i predykcji, można zauważyć pewne różnice między nimi. Jednak ogólna jakość predykcji jest dobra, ponieważ predykcje modelu wykazują tendencję do śledzenia prawdziwych wartości temperatury, wilgotności oraz nasłonecznienia.

Wyniki predykcji sugerują, że model może dobrze radzić sobie z tym konkretnym zadaniem. Jednak istnieje możliwość dalszego doskonalenia modelu poprzez dostosowanie hiperparametrów treningowych. Na przykład, zmiana liczby jednostek w warstwach RNN i Dense, współczynnika uczenia czy liczby epoch treningowych może wpłynąć na poprawę dokładności predykcji.

Warto jednak zauważyć, że zmiana hiperparametrów może wymagać znacznego czasu obliczeniowego, zwłaszcza gdy chodzi o trenowanie modelu na dużej ilości danych. Konieczne może być użycie komputera o dużej mocy obliczeniowej, lub rozważenie korzystania z platformy obliczeniowej z dostępem do GPU, aby przyspieszyć proces trenowania modelu.

Podsumowując, model sieci neuronowej ma potencjał do przewidywania temperatury na podstawie dostępnych danych, a jego jakość może być doskonalona poprzez eksperymentowanie z hiperparametrami.

# Literatura

-

# Summary

The goal of this project was to develop a neural network model to predict temperature, humidity and sun exposure based on available data. The neural network was implemented using Python and Keras, with various preprocessing steps and hyperparameter tuning. The initial dataset consisted of measurements including temperature, humidity, solar radiation, pressure, acceleration and resistance. The project began with data preprocessing, which involved removing unnecessary columns, sorting the data by measurement date, and filtering for temperature, humidity and sun exposure values. The dataset was further refined by grouping the data by sensor ID and taking the mean value for each sensor. Next, the data was prepared for training and testing the neural network. The dataset was split into a training set and a test set, and then transformed into a suitable input format for the neural network. The "convertToMatrix" function was used to create input-output pairs from the data with a specified time step. The neural network architecture chosen for this project was a simple recurrent neural network (RNN). The model consisted of a SimpleRNN layer followed by two dense layers. The model was compiled with the RMSprop optimizer and the mean squared error loss function. The model was trained using the training data, with a specified number of epochs and batch size. A custom callback function was implemented to print the progress of the training process. After training, the model's performance was evaluated using various metrics, including the root mean squared error loss. The results showed that the neural network model achieved promising results in predicting temperature. Although there were some differences between the predicted values and the true values, the overall prediction quality was good. The model was able to capture the underlying patterns and trends in the data. In conclusion, the developed neural network model demonstrated the ability to predict temperature based on the provided data. Further improvements could be made by fine-tuning the hyperparameters and exploring different network architectures. However, it should be noted that adjusting the hyperparameters could significantly increase the computational time, requiring access to a powerful computing system. This project lays the foundation for future work on temperature prediction and can be expanded to include other variables, such as humidity and solar radiation. The developed model can be further refined and deployed in real-world applications, such as weather forecasting or energy management systems, to provide accurate temperature predictions.. All of the information used in this Master’s Thesis has been acquired during studies at Rail Vehicle Engineering specialty of studies at Cracow University of Technology.

# Spis załączników

Wszystkie załączniki znajdują się w repozytorium pod następującym URL:  
<https://github.com/leaniske/RNN>.

Załącznik nr 1 – …