

Politechnika Warszawska

W Y D Z I A Ł M A T E M A T Y K I
I N A U K I N F O R M A C Y J N Y C H



Praca dyplomowa inżynierska

na kierunku Inżynieria i Analiza Danych

Przewidywanie aktywności sejsmicznej z zastosowaniem sieci
neuronowych

Michał Gromadzki

Numer albumu 313356

Julia Kaznowska

Numer albumu 313379

Krzysztof Wodnicki

Numer albumu 313538

promotor

dr Elżbieta Sienkiewicz

WARSZAWA 2024

Streszczenie

Przewidywanie aktywności sejsmicznej z zastosowaniem sieci neuronowych

Trzęsienia ziemi są niszczycielską siłą, która może prowadzić do wielu ludzkich tragedii. Efektywne przewidywanie takich wydarzeń może pomóc minimalizować szkody i lepiej przygotowywać się na katastrofy. Celem tej pracy jest sprawdzenie skuteczności przewidywania trzęsień ziemi za pomocą metod uczenia maszynowego. Szczególny nacisk został położony na zastosowanie rekurencyjnych sieci neuronowych i modeli typu transformer. Skuteczność badanych metod została zweryfikowana i daje obiecujące efekty na przyszłość. Wyniki zostały przedstawione w przykładowej, praktycznej aplikacji. Otrzymane wyniki mogą mieć pozytywne konsekwencje na działania prewencyjne i szybkie reagowanie w przypadku trzęsień ziemi.

Słowa kluczowe: sieci neuronowe, rekurencyjne sieci neuronowe, transformer, trzęsienie ziemi, klasyfikacja, szeregi czasowe

Abstract

Seismic activity prediction with neural networks

Earthquakes are a destructive force that can cause many tragedies. Effectively predicting such events can help minimize damages and better prepare for the disasters. The aim of this work is to try to predict the earthquakes using machine learning methods. The emphasis has been put on the use of recurrent neural networks and transformer-type models. The effectiveness of the method has been verified and gives promising results for the future research. Results have been presented as a sample, practical application. They may have a positive effect regarding preventive action and rapid response in the event of an earthquake.

Keywords: neural networks, recurrent neural network, transformer, earthquake, classification, time series

Spis treści

1. Wstęp	10
1.1. Wprowadzenie	10
1.2. Słownik	11
1.3. Podział pracy	12
2. Wymagania	14
2.1. Wymagania funkcjonalne	14
2.1.1. Przypadki użycia (Use cases)	14
2.1.2. Historie użytkowników (User stories)	15
2.2. Wymagania niefunkcjonalne	15
2.3. Analiza ryzyka	16
3. Propozycja rozwiązania	18
3.1. Opis rozwiązania	18
3.2. Architektura systemu	18
3.3. Opis modułów	20
3.4. Projekt interfejsu graficznego	20
4. Dane	22
4.1. Grupowanie danych	23
4.2. Eksploracja danych	24
4.3. Zmienne tektoniczne	28
4.4. Dodawanie etykiet	32
4.5. Przygotowanie danych	34
4.5.1. Magnitude	34
4.5.2. MagType	36
4.5.3. Longitude	36
4.5.4. Latitude	37
4.5.5. Depth	38
4.5.6. Dist	39

SPIS TREŚCI

4.5.7. Plate	41
4.5.8. Dist_region	42
4.5.9. Plate_region	44
4.5.10. Lon_cent	45
4.5.11. Lat_cent	46
5. Metodologia	47
5.1. Tworzenie zbioru uczącego do modelu	47
5.1.1. Iteracja po regionach	48
5.1.2. Dodanie zmiennej Distance	51
5.1.3. Wybór trzęsień w określonym promieniu od środka danego regionu	51
5.1.4. Sortowanie wybranych trzęsień rosnąco po dacie	51
5.1.5. Dodanie zmiennej Diff_days oraz przygotowanie zmiennych Distance i Diff_days	52
5.1.6. Dodanie zmiennych z określonej liczby wcześniejszych trzęsień - tworzenie sze- regu czasowego	55
5.1.7. Podział na zbiór treningowy, walidacyjny i testowy	58
5.1.8. Stworzenie 3 tablic dla każdego ze zbiorów - szereg czasowy, informacje regio- nalne, etykiety	58
5.1.9. Przekształcenie tablic do odpowiednich wymiarów	60
5.1.10. Zapisanie szeregów czasowych do plików .npy	61
5.2. Modele uczenia maszynowego	62
5.2.1. Opis modeli	62
5.2.2. Architektury modeli	67
5.2.3. Trenowanie	77
5.3. Ewaluacja modeli	79
5.3.1. Modele trenowane bez parametru <i>class_weights</i>	79
5.3.2. Modele trenowane z parametrem <i>class_weights</i>	85
5.3.3. Analiza wyników	91
6. Wyniki - aplikacja	92
6.1. Opis aplikacji	92
6.2. Dependencje	92
6.3. Instrukcja instalacji	93
6.3.1. Konfiguracja aplikacji	93
6.3.2. Uruchomienie aplikacji	95

6.4.	Instrukcja obsługi	95
6.4.1.	Home	95
6.4.2.	App	95
6.4.3.	Admin panel	101
6.5.	Testy	103
6.5.1.	Mapa	103
6.5.2.	Baza trzęsień ziemi	104
6.5.3.	API	105
6.6.	Dokumentacja techniczna	106
6.6.1.	Baza danych	106
6.6.2.	Modele django	106
6.6.3.	API	108
7.	Podsumowanie	110
	Bibliografia	111

1. Wstęp

1.1. Wprowadzenie

Według danych USGS[1] w 2023 roku na świecie wystąpiło 147 343 trzęsień ziemi. Oznacza to, że zjawisko to jest rejestrowane średnio co 3.5 minuty.

Aktualne systemy ostrzegania (np. ShakeAlert obsługiwany przez USGS) powiadamiają użytkowników o trzęsieniu ziemi dopiero kilka sekund po tym, jak trzęsienie się rozpocznie. Bardziej nowatorskie metody i badania[2] pozwalają na wykrycie trzęsień o dużych magnitudach nawet do dwóch godzin przed rozpoczęciem. Osiągane jest to dzięki czujnikom GPS o wysokiej czułości.

Kolejnym naturalnym krokiem w rozwoju badań jest opracowanie metody wczesnego ostrzegania przed trzęsieniami ziemi - na przykład na tydzień czy miesiąc przed wydarzeniem. Udoskonalenie narzędzi predykcyjnych wiążałoby się z możliwością wdrożenia skutecznych procedur wczesnego reagowania, a co za tym idzie ratowania ludzkiego życia i zminimalizowania strat.

W odpowiedzi na powyższe wyzwania zaczęły powstawać prace naukowe próbujące przewidzieć trzęsienia ziemi, przy wykorzystaniu modeli np. typu Random Forest[3]. Niniejsza praca ma na celu rozszerzenie tej koncepcji na bardziej zaawansowane modele. Przygotowane zostały modele oparte o rekurencyjne sieci neuronowe i architekturę transformer oraz prosta aplikacja pokazująca praktyczne możliwości tego podejścia.

Drugi rozdział tej pracy - Wymagania - opisuje początkowe oczekiwania względem realizowanego rozwiązania. Wymagania dotyczące modelu i działania aplikacji rozwijały się wraz z postępami pracy, ale podstawowe potrzeby pozostały bez zmian. Rozdział Propozycja rozwiązania jest odpowiedzią na postawione w poprzednim rozdziale wymagania. Zostały tam opisane pierwsze idee dotyczące modelowania i implementacji aplikacji. Mimo rozwoju pomysłu i wielu zmian, podstawowa koncepcja pozostała taka sama. Następne rozdziały skupią się już na merytorycznych efektach pracy i szczegółowym opisie konkretnych zagadnień. W rozdziale Dane została przeprowadzona szczegółowa eksploracja dostępnego zbioru danych. Opisane zostały również jego przekształcenia. Kwestia przekształceń danych jest kontynuowana w rozdziale Metodologia, który zaczyna się od opisu konstrukcji zbioru uczącego. Rozdział ten opisuje też zastosowane modele i analizuje ich wyniki. Ostatni rozdział Wyniki - aplikacja jest w całości

1.2. SŁOWNIK

poświęcony opisowi aplikacji. Zawarte są w nim również instrukcja wdrożeniowa i testy.

1.2. Słownik

Region	Jednostkowy obszar, o wielkości 1 stopień szerokości na 1 stopień długości geograficznej, dla którego będzie wykonywana predykcja.
Tensorflow	Biblioteka służąca do implementacji sieci neuronowych w Pythonie.
LSTM	Architektura rekurencyjnej sieci neuronowej.
Transformer	Popularna sieci neuronowa, często stosowana w NLP, jak i szeregach czasowych.
Szereg czasowy	Ciąg informacji uporządkowanych w czasie.
tf.data.Dataset	Klasa w bibliotece Tensorflow, umożliwia efektywne zarządzanie wejściowymi danymi do modelu.
tf.keras.layers.Dense	Warstwa w bibliotece Tensorflow, implementuje standardową warstwę sieci neuronowej, w której każdy neuron jest połączony ze wszystkimi neuronami w poprzedniej warstwie.
tf.keras.layers.Embedding	Warstwa w bibliotece Tensorflow, służy do mapowania dyskretnych kategorii na ciągle wektory liczbowe.
tf.keras.layers.MultiHeadAttention	Warstwa w bibliotece Tensorflow, implementuje mechanizm atencji, stosowane w transformerach[4].
tf.keras.layers.LayerNorm	Warstwa w bibliotece Tensorflow, normalizuje aktywacje w warstwie sieci, wspomaga proces uczenia.
Koder	Część transformera, odpowiada za przetwarzanie kontekstu.
Dekoder	Część transformera, odpowiada za przetwarzanie sekwencji.
Połączenia rezydualne	Technika używana w głębszych sieciach neuronowych, w której wejście do danej warstwy jest dodawane do wyjścia tej samej warstwy. Przeciwdziała problemowi zanikającego gradientu, wspomaga trenowanie modeli.

1.3. Podział pracy

Nr	Zadanie	Osoba
1.0	Abstrakt	Krzysztof Wodnicki
1.1.1	Wprowadzenie	Julia Kaznowska
1.1.2	Słownik	Michał Gromadzki
1.2	Wymagania	Krzysztof Wodnicki
1.3	Propozycja rozwiązania	Julia Kaznowska
1.4	Dane	Michał Gromadzki
1.5	Metodologia	
1.6	Wyniki - aplikacja	Julia Kaznowska
1.7	Podsumowanie	Krzysztof Wodnicki
1.8	Bibliografia	

Tabela 1.1: Podział pracy - tekst pracy

1.3. PODZIAŁ PRACY

Nr	Zadanie	Osoba
Modelowanie		
2.1	Pobieranie danych	Michał Gromadzki
2.2	Analiza danych	
2.3	Przygotowanie danych	
2.4	Stworzenie szeregów czasowych	
2.5	Implementacja modeli	
2.6	Trenowanie modeli	
2.7	Ewaluacja modeli	
2.8	Implementacja inferencji na stronie	
Aplikacja		
3.1	Konfiguracja bazy danych	Julia Kaznowska
3.2	Filtrowanie mapy	
3.3	Nawigacja mapy	
3.4	Obsługa listy trzęsień ziemi	
3.5	Wyświetlanie szczegółów trzęsień ziemi	
3.6	Konfiguracja aplikacji django	Krzysztof Wodnicki
3.7	Menu nawigacji	
3.8	Konfiguracja mapy	
3.9	Panel ustawień	
3.10	Konfiguracja API	
3.11	Konfiguracja predykcji	

Tabela 1.2: Podział pracy

2. Wymagania

W początkowych etapach pisania pracy inżynierskiej zostały ustalone wymagania dotyczące modelów i aplikacji.

2.1. Wymagania funkcjonalne

Przygotowywana aplikacja powinna umożliwiać użytkownikom przeglądanie, analizę i przewidywanie trzęsień ziemi. W tym celu zostanie udostępniony interfejs z mapą, na której możliwy będzie wybór interesującego regionu i sprawdzenie związków z nim informacji. Pojedyncze wydarzenia będą wyświetlać się na mapie w postaci pinezek, które także będzie można wybrać, aby zapoznać się ze szczegółami danego wydarzenia. Możliwy będzie także wybór miejsca na mapie, pozwalający na zapoznanie się z prognozą modelu dotyczącą wystąpienia tam trzęsienia ziemi. Ponadto administratorzy powinni mieć dostęp do narzędzi niezbędnych do utrzymania systemu.

2.1.1. Przypadki użycia (Use cases)

Typowe użycie aplikacji wyglądałoby w następujący sposób:

1. Użytkownik otwiera aplikację. Na ekranie pojawia się mapa, którą może przybliżać, oddalać, przesuwać i wykonywać inne standardowe akcje.
2. Użytkownik ma dostęp do historii trzęsień ziemi w postaci listy, a na mapie pojawiają się pinezki oznaczające poszczególne wydarzenia.
3. Użytkownik może wybrać konkretne wydarzenie z mapy. Dzięki temu pojawią się szczegóły dotyczące wydarzenia.
4. Użytkownik wybiera obszar, dla którego chciałby przeprowadzić predykcję. Obszar to przestrzeń wyznaczona przez linie siatki szerokości i długości geograficznej. Predykcja mówi nam, czy w przyszłym miesiącu w tym obszarze wystąpi trzęsienie ziemi. Dzięki temu panel historii wydarzeń zostaje zastąpiony informacjami o możliwym trzęsieniu ziemi.

2.2. WYMAGANIA NIEFUNKCJONALNE

Typowe wykorzystanie aplikacji przez administratora wyglądałoby następująco:

1. Administrator otwiera panel administracyjny. Pojawiają się tam informacje o aktualnej wersji danych oraz lista załadowanych modeli wraz z podstawowymi szczegółami.
2. Administrator wybiera opcję załadowania nowych danych, przesyła aktualny plik i uzupełnia metadane.
3. Administrator może załadować do aplikacji nowy, gotowy model. Trenowanie modelu nie jest częścią aplikacji, odbywa się w innym miejscu.

2.1.2. Historie użytkowników (User stories)

Aby poprawnie zaprojektować i zaimplementować system, niezbędne jest zrozumienie motywacji wszystkich stron, które będą z niego korzystać. Poniższe historie przybliżają motywacje użytkowników i administratorów.

1. Jako użytkownik chciałbym wyświetlić historię trzęsień ziemi, aby wiedzieć jakie trzęsienia występowały w przeszłości.
2. Jako użytkownik chciałbym wyświetlić szczegóły trzęsienia ziemi, aby dowiedzieć się więcej o tym wydarzeniu.
3. Jako użytkownik chciałbym sprawdzić predykcję dla wybranego regionu, aby dowiedzieć się czy w najbliższym czasie w danym regionie wystąpi trzęsienie ziemi.
4. Jako administrator chciałbym załadować nowe dane, aby w systemie były aktualne informacje.
5. Jako administrator chciałbym dodać nowy model, aby wykonywać jak najlepsze predykcje.

2.2. Wymagania niefunkcjonalne

Oprócz zapewnienia konkretnych funkcjonalności system powinien spełniać szereg wymagań, które zapewnią użytkownikom i administratorom możliwość praktycznego wykorzystania systemu. Lista wymagań niefunkcjonalnych została przedstawiona w Tabeli 2.1.

Obszar wymagań	Nr wymagania	Opis
Użyteczność	1	Wszystkie funkcjonalności aplikacji dostępne dla użytkownika muszą zmieścić się na jednym ekranie o rozdzielczości 1920x1080.
	2	Możliwość wykonania predykcji na regionach, które dostarczają wystarczającą ilość danych historycznych.
Niezawodność	3	<i>Accuracy</i> modelu powinno przekraczać 80%.
Wydajność	4	Załadowanie danych i wyświetlanie ich na stronie powinno trwać nie dłużej niż 5 sekund.
	5	Wyświetlenie szczegółów powinno trwać nie dłużej niż 5 sekund.
Wsparcie	6	W panelu administratora wyświetla się lista wgranych modeli.

Tabela 2.1: Lista nie-funkcjonalnych wymagań

2.3. Analiza ryzyka

Analiza ryzyka została przeprowadzona zgodnie z metodologią SWOT. Wyniki zapisane zostały w tabeli 2.2.

SWOT	Szanse	Zagrożenia
Wewnętrze	Doświadczenie w eksplorowanych dziedzinach	Ograniczenia czasowe, inne zobowiązania
Zewnętrzne	Brak podobnych rozwiązań na rynku	Brak gwarancji skuteczności modelu, mała dostępność literatury

Tabela 2.2: SWOT

Zagrożenia wewnętrzne:

- Ograniczenia czasowe - wysokie prawdopodobieństwo.

Aby temu zapobiec będziemy ściśle trzymać się harmonogramu i przestrzegać terminów.

- Inne zobowiązania - średnie prawdopodobieństwo.

W razie wyjątkowego obciążenia któregoś z członków zespołu jego zadania tymczasowo

2.3. ANALIZA RYZYKA

zostaną podzielone na pozostałą część zespołu.

Zagrożenia zewnętrzne:

- Niska skuteczność modelu - niskie prawdopodobieństwo.

Jeśli predykcje modelu będą niezadowalające możliwe jest znalezienie dodatkowych, większych zbiorów danych lub przeformułowanie problemu w taki sposób, aby możliwe było przeprowadzenie skuteczniejszych predykcji.

- Mała dostępność literatury - średnie prawdopodobieństwo.

Jeśli literatura dotycząca przewidywania trzęsień ziemi będzie niewystarczająca, możliwe jest skontaktowanie się z ekspertami - zarówno z dziedziny uczenia maszynowego jak i geologii.

3. Propozycja rozwiązania

We wstępnej fazie rozwoju projektu zaproponowane zostało poniżej opisane rozwiązanie. W trakcie rozwoju modelu i aplikacji niektóre aspekty uległy zmianom, natomiast główna idea pozostała taka sama.

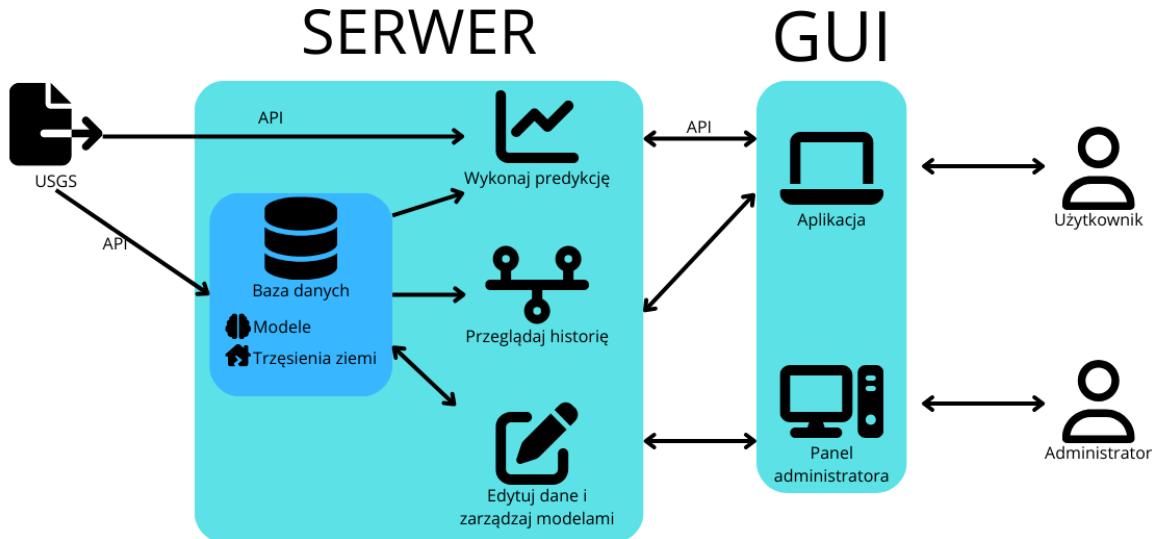
3.1. Opis rozwiązania

Rozwiązanie będzie składać się z dwóch głównych części - modelowania oraz strony internetowej. Wypróbowane zostaną różne techniki modelowania, z różnymi danymi wejściowymi. Przygotowana strona internetowa będzie interfejsem graficznym do wyświetlania predykcji oraz przeglądania danych historycznych.

3.2. Architektura systemu

Rozwój modeli będzie odbywał się poza dostarczoną aplikacją. Z poziomu panelu administratora będzie natomiast możliwe zarządzanie modelami i wprowadzanie zmian w danych związanych z modelem. Możliwe jest także wykorzystanie zapisanego w bazie modelu do predykcji. Użytkownik może to zrobić z poziomu aplikacji, za pośrednictwem API. Do wykonania predykcji konieczne jest pobranie danych z wybranego czasu i miejsca, a co za tym idzie odwołanie się do API USGS lub danych zapisanych w lokalnej bazie.

3.2. ARCHITEKTURA SYSTEMU



Rysunek 3.1: Główne elementy architektury

Powyżej opisane główne funkcjonalności i architektura systemu zostały graficznie przedstawione na schemacie na rysunku 3.1. Są to jednak tylko najważniejsze elementy systemu. Możliwe, że uwzględnione zostaną też inne, pomniejsze funkcjonalności, które nie będą się bezpośrednio wpisywać w powyższy schemat.

3.3. Opis modułów

Moduły zostały opisane w tabeli 3.1

Moduł	Wejście	Wyjście	Opis działania
Moduł predykcji	Region predykcji, model	Predykcja	Moduł pozyskuje historię wstrząsów z bazy danych lub API USGS i przekształca ją do postaci przyjmowanej przez model. Następnie z bazy danych wczytywany jest model i wykonywana predykcja, która jest zwracana jako wyjście modułu.
Moduł historii	Filtryle lokalizacji, czasu, magnitudy, etc.	Lista wstrząsów po filtrowaniu	Moduł przyjmuje listę filtrów i na ich podstawie wysyła zapytanie do bazy danych, którego wynik jest zwracany jako wyjście modułu.

Tabela 3.1: Moduły

3.4. Projekt interfejsu graficznego

Interfejs graficzny będzie zrealizowany w postaci strony internetowej. Dostępne będą dwie warstwy - warstwa użytkownika oraz warstwa administratora.

Warstwa administratora pozwoli na zarządzanie modelami, w tym załadowanie nowych modeli i usuwanie starych. Przeglądanie modeli będzie zrealizowane w postaci tabelki. Ponadto administrator będzie mógł aktualizować bazę danych trzęsień i wykonywać na niej wszystkie standardowe operacje - dodawać, usuwać i aktualizować rekordy.

W warstwie użytkownika możliwe będzie przeglądanie mapy z pinezkami oznaczającymi poszczególne trzęsienia ziemi. Skrócone informacje o nich będą widoczne w bocznym panelu. Będzie

3.4. PROJEKT INTERFEJSU GRAFICZNEGO

można również wyświetlać szczegółowe dotyczące konkretnego trzęsienia ziemi poprzez kliknięcie na daną pinezkę bądź wybranie trzęsienia z panelu bocznego. Użytkownik będzie mógł także zobaczyć predykcję modelu dla regionu. Dostępne będą również dodatkowe opcje związane z obsługą mapy i wyświetlaniem trzęsień.

4. Dane

Dane zostały pobrane poprzez API strony USGS[1] za pomocą biblioteki requests w języku Python[5]. Następnie zostały zapisane w formacie .csv, w taki sposób, że jeden wiersz w pliku zawiera informacje o jednym zjawisku. Pobrane i zapisane zostały wszystkie dostępne dane do dnia 1 października 2023 roku włącznie. Należy zauważyć, że API strony USGS udostępnia nie tylko dane trzęsień ziemi, ale również eksplozji górniczych, osuwisk, a nawet wybuchów wulkanów. Z tego powodu zostały wyróżnione 2 zestawy danych:

1. Pełen zestaw danych liczący 4.45 miliona zjawisk oraz 29 kolumn je opisujących. Braki danych stanowią około 21% danych.
2. Odfiltrowany zestaw danych liczący 4.33 miliona trzęsień ziemi oraz 6 następujących kolumn:
 - time (date) - Czas trzęsienia ziemi
 - longitude (float) - Długość geograficzna
 - latitude (float) - Szerokość geograficzna
 - depth (float) - Głębokość epicentrum trzęsienia
 - mag (float)- Skala trzęsienia
 - magType (string) - Rodzaj skali trzęsienia ziemi

Braki danych stanowią około 1.3% danych. W Tabeli 4.1 przedstawiony jest przykładowy widok odfiltrowanego zestawu danych.

4.1. GRUPOWANIE DANYCH

time	longitude	latitude	depth	mag	magType
1940-01-06 20:04:35.170	25.81	35.37	15.00	5.84	mw
1940-01-06 09:15:39.210	151.50	45.08	25.00	6.07	mw
1940-01-05 09:42:55.570	-116.37	33.17	6.00	3.42	ml
1940-01-05 07:20:50.460	-119.44	32.93	6.00	3.97	ml
1940-01-04 21:44:55.390	37.93	40.41	15.00	5.51	mw

Tabela 4.1: Przykładowy widok tabeli z danymi

Do dalszego modelowania został wykorzystany jedynie zbiór danych odfiltrowanych. Pełen zestaw danych został zapisany w charakterze kopii zapasowej.

4.1. Grupowanie danych

Wszystkie operacje na danych zostały wykonane za pomocą biblioteki Pandas[6], o ile nie zostało napisane inaczej. Trzęsienia zostały pogrupowane w kwadraty o wielkości 1° szerokości na 1° długości geograficznej nazywane później regionami. W Tabeli 4.2 został przedstawiony przykładowy widok zestawu danych wraz z nowymi kolumnami.

time	longitude	latitude	...	longitude_disc	latitude_disc	pos
1950-01-07 17:22:50.050	-119.49	34.32	...	-120.00	34.00	34_-120
1950-01-07 10:37:33.670	-116.82	31.98	...	-117.00	31.00	31_-117
1950-01-05 15:00:31.860	-119.24	34.55	...	-120.00	34.00	34_-120
1950-01-05 14:45:48.730	-119.32	33.57	...	-120.00	33.00	33_-120
1950-01-05 13:51:29.690	-116.89	34.36	...	-117.00	34.00	34_-117

Tabela 4.2: Przykładowy widok pogrupowanych danych

Zostały dodane 3 nowe kolumny:

- longitude_disc (int) - Zdyskretyzowana długość geograficzna, co 1°
- latitude_disc (int) - Zdyskretyzowana szerokość geograficzna, co 1°
- pos (string) - Tekstowy identyfikator każdego z regionów

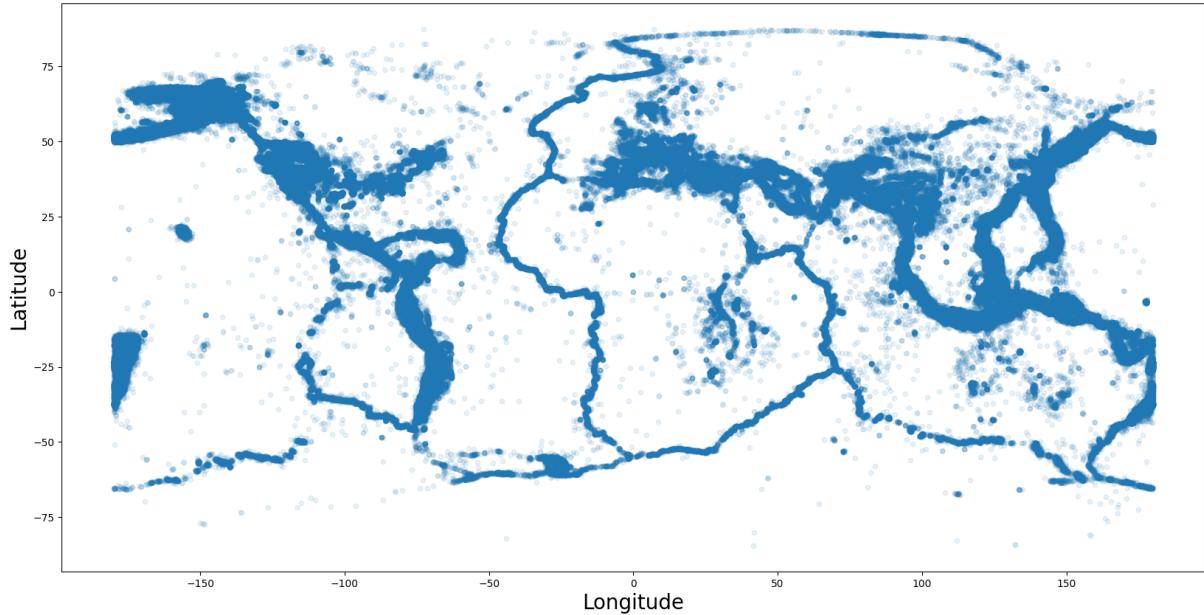
W ten sposób został otrzymany podział trzęsień na 13 434 różnych regionów.

4.2. Eksploracja danych

Wszystkie wizualizacje zostały przygotowane za pomocą biblioteki Matplotlib[7], o ile nie zostało napisane inaczej. Wszystkie trzęsienia z brakującymi informacjami zostały usunięte. Należy również podkreślić, że przed rokiem 1950 było rejestrowane niewiele trzęsień - z tego powodu na wizualizacjach będą występować jedynie trzęsienia od 1951 roku. Ostatecznie wizualizowane będzie 4 136 500 trzęsień w 13 137 regionach.

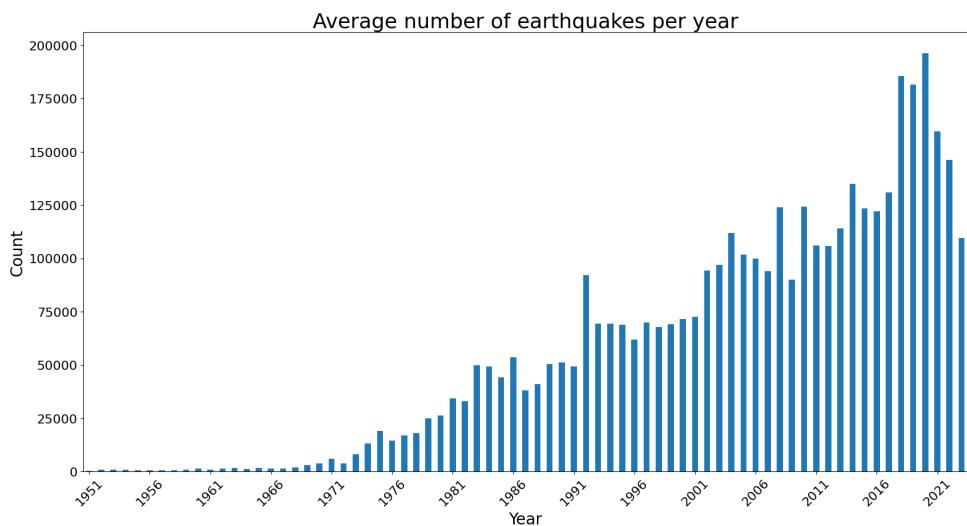
Na Rysunku 4.1 przedstawiony jest rozkład trzęsień na mapie. Najwięcej trzęsień jest rejestrowanych na zachodnim brzegu USA oraz w Oceanii. Dodatkowo widać znaczne zwiększenie liczby trzęsień wzdłuż granic płyt tektonicznych.

4.2. EKSPLORACJA DANYCH



Rysunek 4.1: Rozkład trzęsień na mapie

Na Rysunku 4.2 przedstawiona jest średnia liczba trzęsień na rok. Liczba trzęsień z roku na rok wyraźnie rośnie.

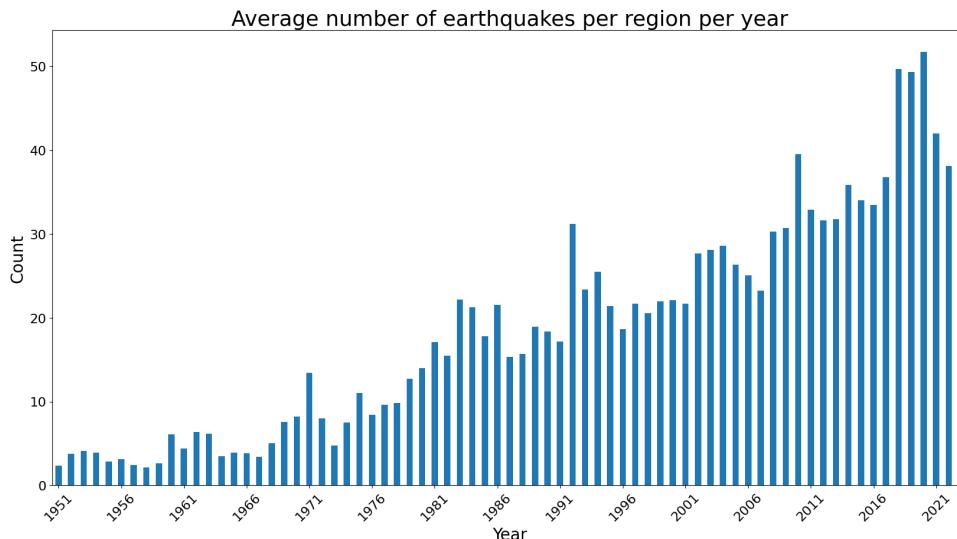


Rysunek 4.2: Średnia liczba trzęsień na rok

Poniżej zostały przedstawione dwie potencjalne przyczyny tego zjawiska:

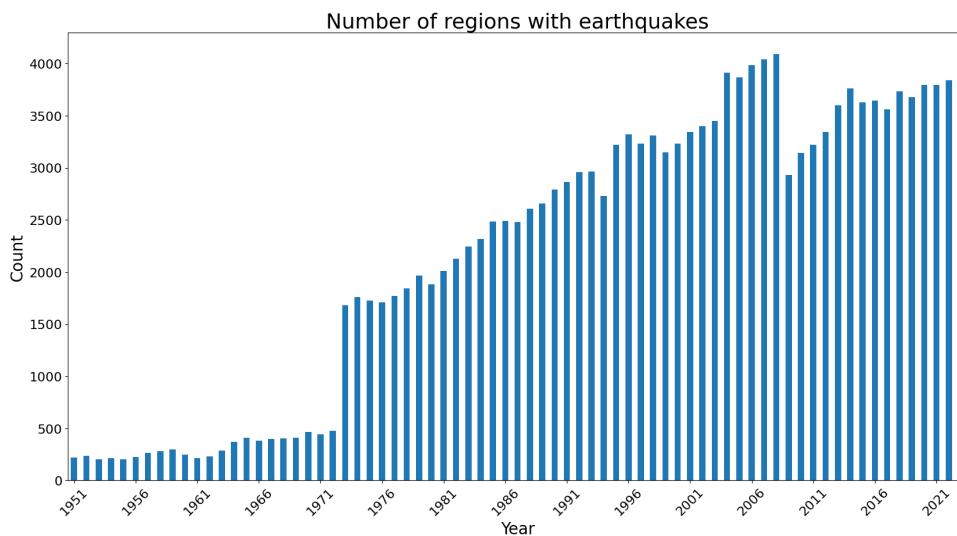
- W regionach już badanych poprawiona jest jakość sejsmografów, a co za tym idzie coraz więcej trzęsień zostaje wykrytych - Rysunek 4.3
- Pojawia się coraz więcej nowych, badanych, regionów, co również prowadzi do zwiększonej liczby wykrywanych trzęsień - Rysunek 4.4

Na Rysunku 4.3 przedstawiona jest średnia liczba trzęsień w regionie na rok. Można stwierdzić, że liczba ta z roku na rok rośnie, co przyczynia się do rosnącej liczby wykrywanych trzęsień.



Rysunek 4.3: Średnia liczba trzęsień w regionach na rok

Na Rysunku 4.4 przedstawiona jest liczba regionów z przynajmniej jednym trzęsieniem. Wyraźnie widać, że liczba badanych regionów rośnie. Dodatkowo widoczny jest skokowy wzrost w 1973 roku. Można stwierdzić, że zwiększająca się liczba badanych regionów również przyczynia się do rosnącej liczby wykrywanych trzęsień.

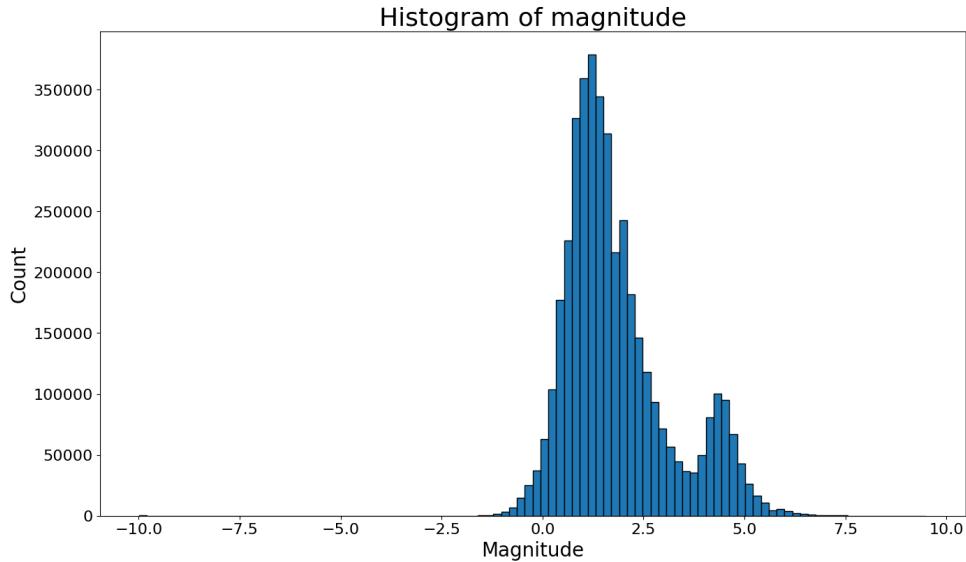


Rysunek 4.4: Liczba regionów z trzęsieniami na rok

Podsumowując, zarówno poprawianie jakości sejsmografów w już badanych regionach jak i badanie nowych regionów przyczyniło się do rosnącej z roku na rok liczby trzęsień widocznej na Rysunku 4.2.

4.2. EKSPLORACJA DANYCH

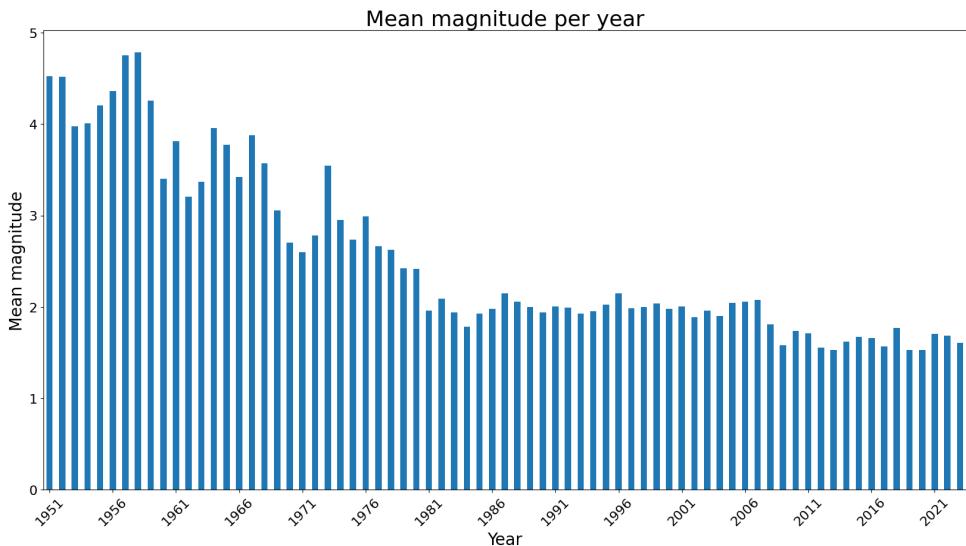
Na Rysunku 4.5 przedstawiony jest histogram skali trzęsień ziemi. Rysunek sugeruje, że wartości magnitudy mają rozkład, w przybliżeniu, dwu-modalny.



Rysunek 4.5: Histogram skali trzęsień

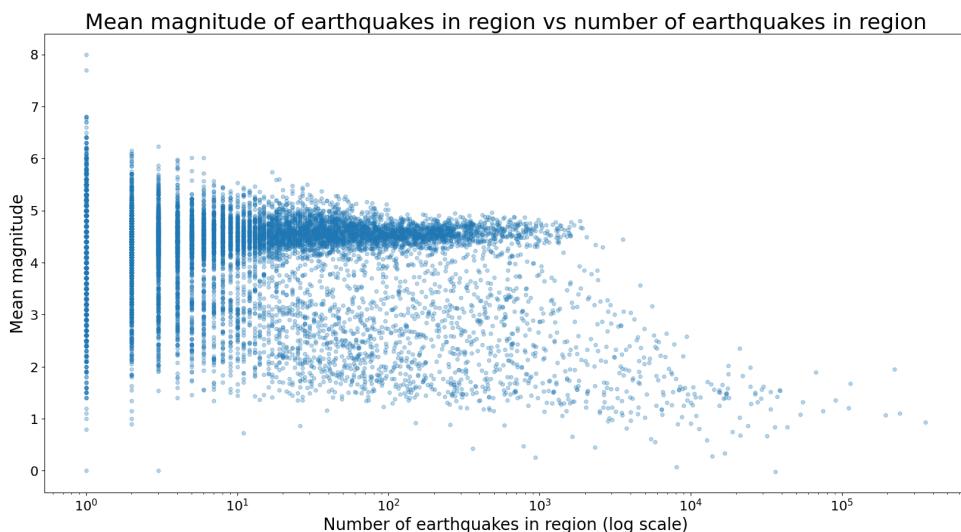
Taki rozkład bardzo często wskazuje na istnienie dwóch grup w danych. Aby lepiej zrozumieć pochodzenie takiego rozkładu zostały przygotowane Rysunki 4.6 oraz 4.7.

Na Rysunku 4.6 przedstawiona jest średnia skala trzęsienia na rok. Wyraźnie widać, że z roku na rok średnia magnituda maleje. Jest to najprawdopodobniej spowodowane rosnącą dokładnością zbierania danych. W przeszłości rejestrowane były jedynie największe trzęsienia. W dzisiejszych czasach kolekcjonowane są również trzęsienia o mniejszej skali, co powoduje zmniejszanie się średniej magnitudy.



Rysunek 4.6: Średnia skala trzęsienia na rok

Można postawić hipotezę, że w wykorzystywanych danych istnieją dwa rodzaje regionów. Pierwszy rodzaj to regiony, w których systemy wykrywania trzęsień ziemi nie są dobrze rozwinięte. W takich regionach będą rejestrowane jedynie największe trzęsienia. Drugi rodzaj regionów to takie, w których systemy wykrywania trzęsień są bardzo dobrze rozwinięte. W takich regionach rejestrowana jest dużo większa liczba trzęsień, zarówno dużych, jak i małych. Regiony drugiego rodzaju odpowiadają za pierwszą modę na Rysunku 4.5, równą około 1.5. Natomiast pierwszy rodzaj regionów odpowiada za drugą modę, równą około 4.5. Aby potwierdzić hipotezę został przygotowany wykres średniej magnitudy od liczby trzęsień w danym regionie, przedstawiony na Rysunku 4.7.



Rysunek 4.7: Średnia skala trzęsień w regionie do liczby trzęsień w regionie

Na podstawie Rysunku 4.7 można stwierdzić, że faktycznie istnieją dwa rodzaje regionów. Pierwszy rodzaj to regiony do około 2 000 trzęsień o średniej magnitudzie, w przybliżeniu, 4.5. Drugi rodzaj to obszary liczące nawet 300 000 trzęsień o średniej magnitudzie około 1.5. Rysunek 4.7 potwierdza postawioną wcześniej hipotezę.

4.3. Zmienne tektoniczne

Na podstawie podobieństwa rozkładu trzęsień na Rysunku 4.1 z granicami płyt tektonicznych na Rysunku 4.8 została podjęta decyzja o dodaniu do zbioru danych informacji na temat granic płyt tektonicznych. Do każdego trzęsienia zostały dodane zmienne określające odległość trzęsienia od najbliższej granicy płyt tektonicznych. Dane o granicach płyt tektonicznych zostały pobrane ze strony Kaggle[8][9][10]. W Tabeli 4.3 został przedstawiony przykładowy widok pobranych danych.

4.3. ZMIENNE TEKTONICZNE

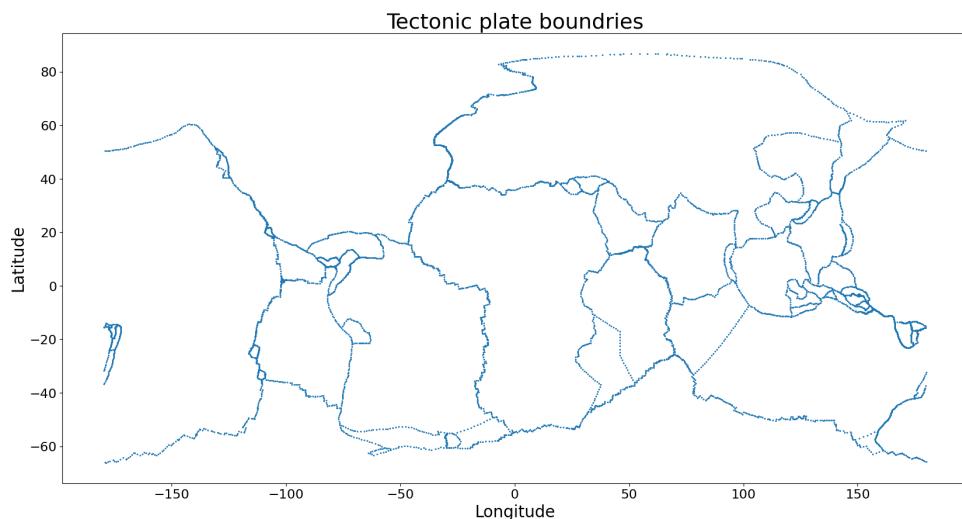
plate	lat	lon
am	30.75	132.82
am	30.97	132.97
am	31.22	133.20
am	31.52	133.50
am	31.88	134.04

Tabela 4.3: Widok tabeli z granicami płyt tektonicznych

Tabela z granicami płyt tektonicznych zawiera 3 kolumny:

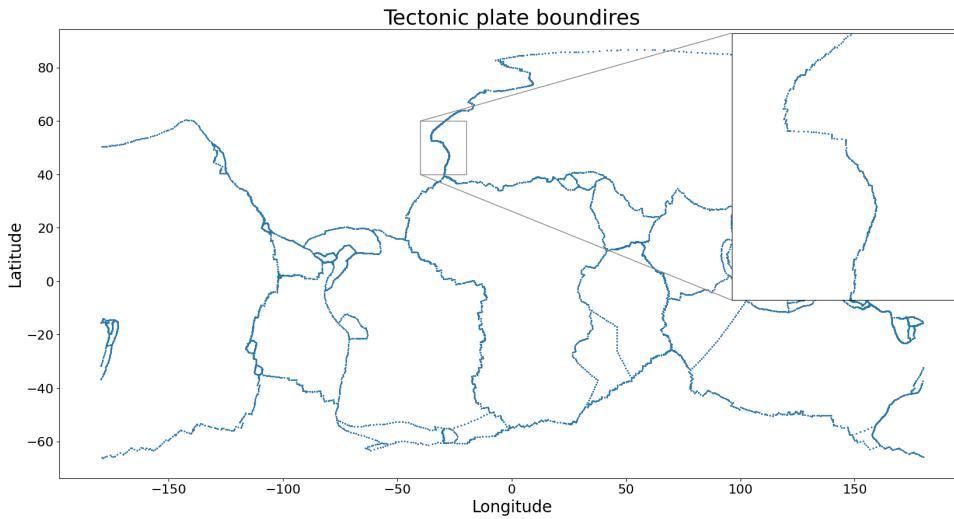
- plate (string) - Identyfikator tekstowy płyty tektonicznej
- lat (float) - Szerokość geograficzna punktu leżącego na granicy
- lon (float) - Długość geograficzna punktu leżącego na granicy

Granice płyt tektonicznych zostały przedstawione na Rysunku 4.8.

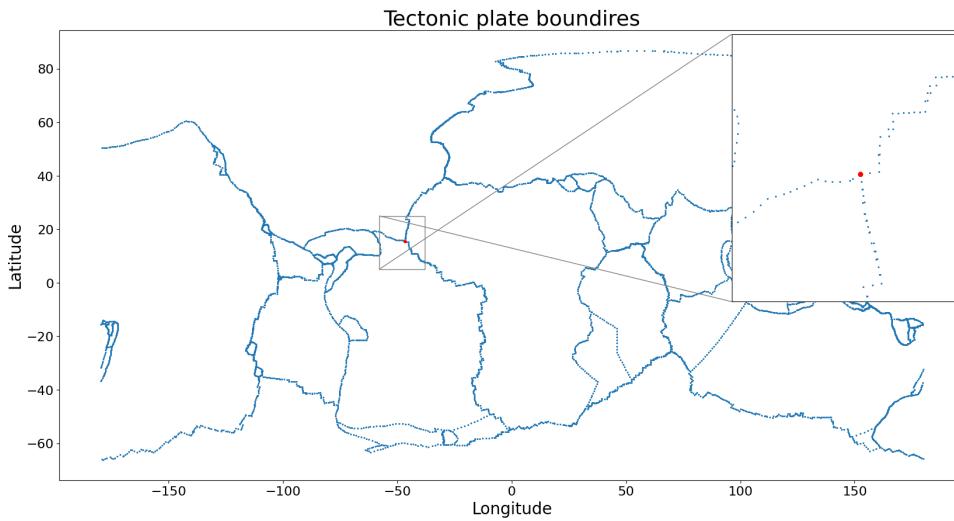


Rysunek 4.8: Granice płyt tektonicznych na mapie

Należy podkreślić, że każdy punkt na mapie występuje w tabeli tyle razy, ile płyt tektonicznych ma w jego miejscu granicę. Na Rysunku 4.9 został przedstawiony przykład.



(a) Przykłady punktów odpowiadających dwóm wierszom w tabeli



(b) Przykład punktu odpowiadającego trzem wierszom w tabeli

Rysunek 4.9: Przykłady rodzajów punktów

Przybliżone punkty przedstawione na Rysunku 4.9a leżą na granicy dwóch płyt tektonicznych, zatem każdemu punktowi odpowiadają dwa wiersze w tabeli - po jednym dla każdej z płyt. Czerwony punkt na Rysunku 4.9b leży na granicy trzech płyt tektonicznych, zatem odpowiadają mu trzy wiersze w tabeli. Pozostałe punkty na tym rysunku leżą na granicy dwóch płyt, więc odpowiadają im po dwa wiersze. W opisywanym zbiorze występują jedynie te dwa przypadki - nie ma punktów, które leżą na granicy czterech lub więcej płyt.

Algorytm znajdowania najbliższej granicy płyt tektonicznych od danego trzęsienia wygląda następująco:

1. Dla każdego punktu w zbiorze danych z granicami płyt tektonicznych obliczana jest odległość od lokalizacji danego trzęsienia

4.3. ZMIENNE TEKTONICZNE

2. Znajdowana jest minimalna odległość
3. Znajdowane są wszystkie tekstowe identyfikatory płyt oddalonych o znalezioną wcześniej minimalną odległość
4. Identyfikatory są sortowane alfabetycznie i konkatenowane w pojedynczy identyfikator granicy
5. Zwrotnie jest minimalna odległość (w kilometrach) oraz nowy identyfikator granicy

Powyższy algorytm zostaje wykonany oddziennie dla każdego trzęsienia. Odległość między trzęsieniami a granicami płyt tektonicznych była obliczana za pomocą formuły Haversine[11]. Należy podkreślić, że identyfikatory granicy występują w dwóch postaciach:

- x_y (string) - Gdzie x i y to identyfikatory tekstowe graniczących płyt (dla granicy między dwiema płytami)
- x_y_z (string) - Gdzie x , y , z to identyfikatory tekstowe graniczących płyt (dla granicy między trzema płytami)

Przykładowy widok zestawu danych po dodaniu informacji o najbliższej granicy płyt tektonicznych znajduje się w Tabeli 4.4.

time	longitude	latitude	...	dist	plate
1973-01-01 01:05:56.150	-117.59	34.19	...	19.30	na_pa
1973-01-01 04:46:09.800	150.63	-9.21	...	55.73	WL_au
1973-01-01 05:20:59.780	-122.12	48.31	...	314.74	jf_na
1973-01-01 06:22:29.800	-173.96	-15.01	...	18.24	NI_TO
1973-01-01 08:58:11.460	-155.36	19.44	...	3522.5	jf_pa

Tabela 4.4: Widok tabeli po dodaniu informacji o płytach tektonicznych do trzęsień ziemi

Zostały dodane 2 nowe kolumny:

- dist (float) - Odległość trzęsienia (w kilometrach) od najbliższej granicy płyt tektonicznych
- plate (string) - Identyfikatory najbliższej granicy płyt tektonicznych

Następnie, w analogiczny sposób, do każdego regionu zostały dodane zmienne określające odległość środka regionu od najbliższej granicy płyt tektonicznych. Należy podkreślić, że dla wszystkich trzęsień w jednym regionie wartości tych kolumn będą takie same. W Tabeli 4.5

został przedstawiony widok tabeli po dodaniu kolumn zawierających informacje o najbliższej granicy płyt tektonicznych od środka danego regionu.

time	longitude	latitude	...	lat_center	lon_center	plate_region	dist_region
1973-01-01 01:05:56.150	-117.59	34.19	...	34.50	-117.50	na_pa	16.69
1973-01-01 04:46:09.800	150.63	-9.21	...	-9.50	150.50	WL_au	21.12
1973-01-01 05:20:59.780	-122.12	48.31	...	48.50	-122.50	jf_na	296.11
1973-01-01 06:22:29.800	-173.96	-15.01	...	-15.50	-173.50	NI_TO	84.32
1973-01-01 08:58:11.460	-155.36	19.44	...	19.50	-155.50	jf_pa	3527.67

Tabela 4.5: Widok tabeli po dodaniu informacji o płytach tektonicznych do poszczególnych regionów

Zostały dodane 4 nowe kolumny:

- lon_center (float) - Długość geograficzna środka danego regionu
- lat_center (float) - Szerokość geograficzna środka danego regionu
- dist_region (float) - Odległość środka regionu (w kilometrach) od najbliższej granicy płyt tektonicznych
- plate_region (string) - Identyfikator najbliższej granicy płyt tektonicznych

4.4. Dodawanie etykiet

Do każdego trzęsienia została dodana etykieta. Zadaniem modelu będzie przewidzenie tej etykiety. Etykiety zostały dodane zgodnie z algorytmem opisany poniżej:

1. Iteracja po regionach
2. Trzęsienia w danym regionie są sortowane rosnąco po dacie

4.4. DODAWANIE ETYKIET

3. Iteracja po dacie, z dokładnością co do dnia, w danym regionie
4. Dla danej daty znajdywana jest data przesunięta o 1 miesiąc do przodu
5. Wybierane są wszystkie trzęsienia w danym regionie w przedziale od daty w danej iteracji (wyłącznie) do znalezionej daty przesuniętej o 1 miesiąc (włącznie)
6. Jeżeli wśród wybranych trzęsień pojawiło się trzęsienie o magnitudzie większej lub równej 5 zostaje ustalone *label* = 1 a w przeciwnym przypadku *label* = 0
7. Wszystkim trzęsieniom w danym regionie o danej dacie przypisywana jest etykieta o wartości zmiennej *label*

W Tabeli 4.6 został przedstawiony przykładowy widok zbioru danych po dodaniu etykiet.

time	longitude	latitude	...	time_disc	label
1973-01-01 01:05:56.150	-117.59	34.19	...	1973-01-01	0
1973-01-01 04:46:09.800	150.63	-9.21	...	1973-01-01	0
1973-01-01 05:20:59.780	-122.12	48.31	...	1973-01-01	0
1973-01-01 06:22:29.800	-173.96	-15.01	...	1973-01-01	0
1973-01-01 08:58:11.460	-155.36	19.44	...	1973-01-01	0

Tabela 4.6: Widok tabeli po dodaniu etykiet

Zostały dodane 2 nowe kolumny:

- *time_disc* (date) - Zdyskretyzowana data trzęsienia, co 1 dzień
- *label* (int) - Etykieta

Algorytm tworzy etykiety o następującej charakterystyce:

- Dane trzęsienie otrzymuje etykietę 1 jeżeli w tym samy regionie, w którym się ono znajduje, w przeciągu jednego miesiąca od jego wystąpienia pojawiło się trzęsienie o skali większej lub równej 5
- Jeżeli występowały tylko trzęsienia o skali mniejszej niż 5 otrzymuje etykietę 0
- Jeżeli nie wystąpiło żadne trzęsienie również otrzymuje etykietę 0

Taki sposób konstrukcji etykiet pozwala na przewidywanie czy w danym regionie w przeciągu miesiąca od wystąpienia danego trzęsienia wystąpi kolejne trzęsienie mogące powodować zniszczenia. Wartość graniczna magnitudy równa 5 została wybrana na podstawie artykułu[12]. Taka

wartość pozwala na przewidywanie relatywnie dużych trzęsień, jednocześnie nie powodując zbyt dużego niezbalansowania klas, które mogłyby wystąpić przy większych wartościach granicznych. Ostatecznie 3 851 797 trzęsień otrzymało etykietę 0, a 248 209 etykietę 1.

4.5. Przygotowanie danych

Jednym z najważniejszych wymagań modelu jest działanie w jak największej liczbie regionów. Z tego powodu (oraz na podstawie Rysunku 4.4) została podjęta decyzja o ograniczeniu danych do trzęsień od 1973 roku. Podsumowując, jako dane do modelu będzie wykorzystywane 4 100 006 trzęsień od początku 1973 roku do 1. października 2023 roku. Jest to 98.7% trzęsień dostępnych na stronie USGS.

Dane zostały podzielone na zbiory treningowy, walidacyjny oraz testowy w proporcjach 85 : 12 : 3. Jest to równoważne następującemu podziałowi po latach:

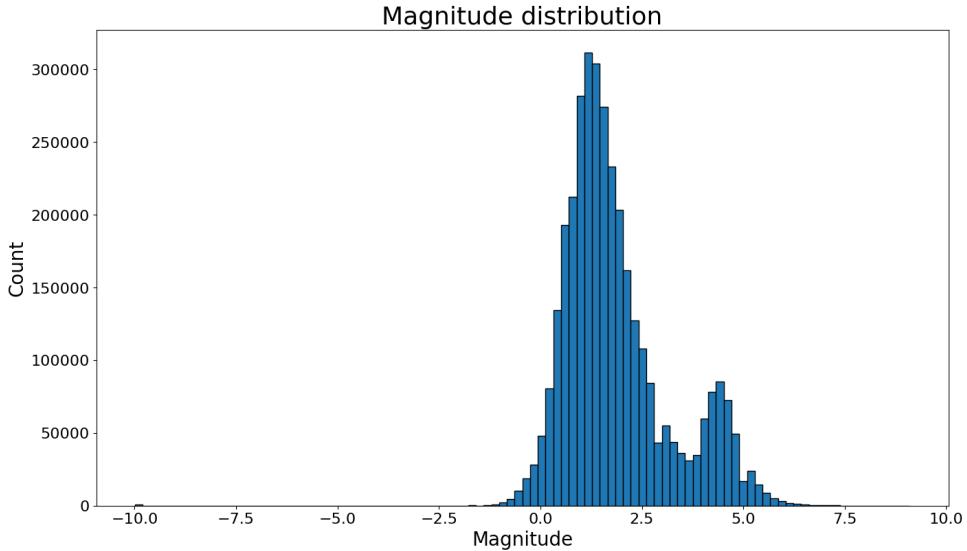
- zbiór treningowy - od 1973 do końca 2020 roku, zawiera 3 488 698 trzęsień
- zbiór walidacyjny - od 2021 do końca 2022 roku, zawiera 501 751 trzęsień
- zbiór testowy - od 2023 roku, zawiera 109 557 trzęsień

Przygotowanie danych przedstawione poniżej zostało wykonane na podstawie rozkładów zmiennych w zbiorze treningowym.

4.5.1. Magnitude

Magnitude jest zmienną opisującą skalę trzęsienia. Na Rysunku 4.10 został przedstawiony rozkład tej zmiennej.

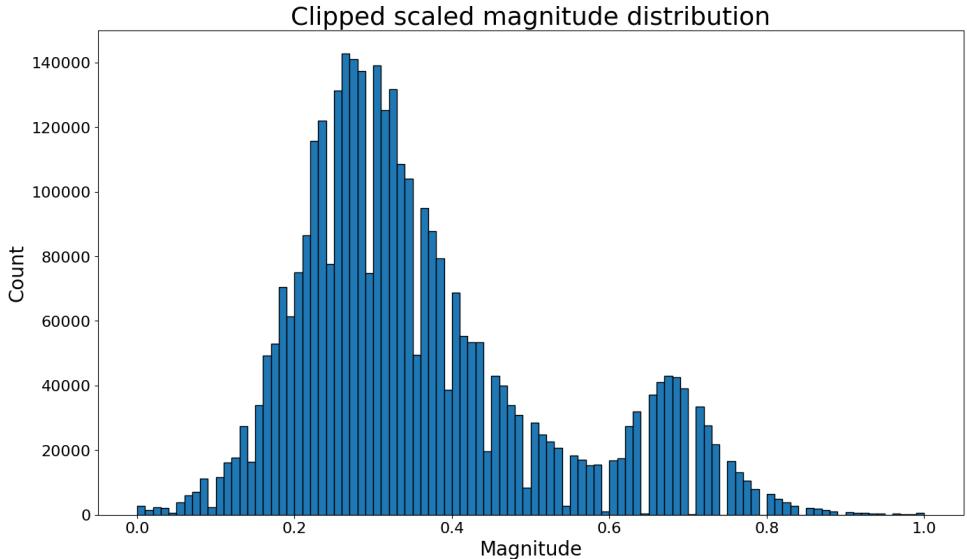
4.5. PRZYGOTOWANIE DANYCH



Rysunek 4.10: Rozkład zmiennej Magnitude

Na podstawie Rysunku 4.10 można stwierdzić, że zmienna Magnitude ma rozkład dwumodalny, dodatkowo pojawiają się wartości odstające. Aby obsłużyć wartości odstające, wartości zostały przycięte do przedziału od -1 do 7 . W tym przedziale mieści się ponad 99.9% wartości.

Ostatecznie zmienna zostaje przeskalowana, za pomocą MinMaxScalera[13], do zakresu od 0 do 1 . Na Rysunku 4.11 został przedstawiony finalny rozkład zmiennej Magnitude.

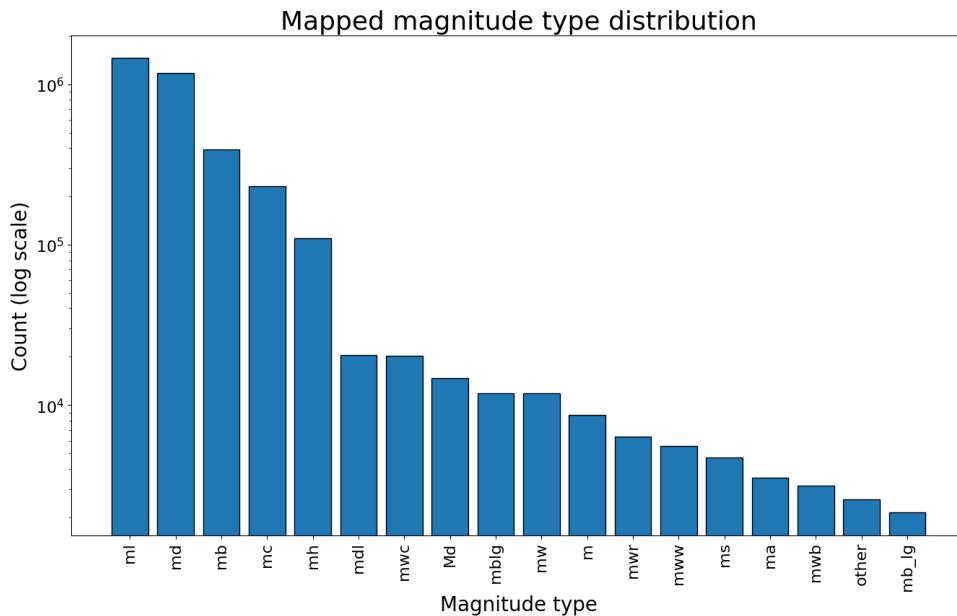


Rysunek 4.11: Rozkład zmiennej Magnitude po przycięciu i przeskalowaniu

Duża część wartości jest podawana z dokładnością do jednego miejsca po przecinku. Jest to przyczyną wgłębień widocznych na Rysunku 4.11.

4.5.2. MagType

MagType to zmienna określająca rodzaj skali trzęsienia ziemi. Zmienna MagType liczy 27 różnych klas. 96.7% danych zawiera się w 5 najbardziej popularnych kategoriach, natomiast w 17 najbardziej popularnych kategoriach zawiera się już ponad 99.9% danych. Na tej podstawie została podjęta decyzja o zmapowaniu pozostałych 10 klas do jednej klasy - "other". Na Rysunku 4.12 został przedstawiony rozkład zmiennej MagType po zmapowaniu.



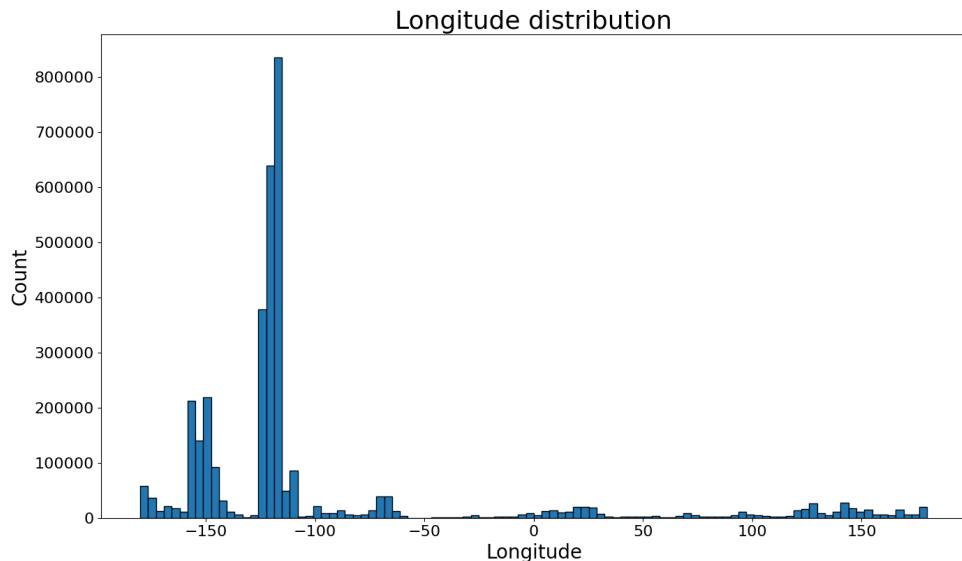
Rysunek 4.12: Rozkład zmiennej MagType po zmapowaniu

Ostatecznie poszczególne kategorie zostały zmapowane do kolejnych liczb naturalnych, a mapowanie zostało zapisane do pliku .csv.

4.5.3. Longitude

Longitude to zmienna określająca długość geograficzną danego trzęsienia. Jej rozkład został przedstawiony na Rysunku 4.13.

4.5. PRZYGOTOWANIE DANYCH

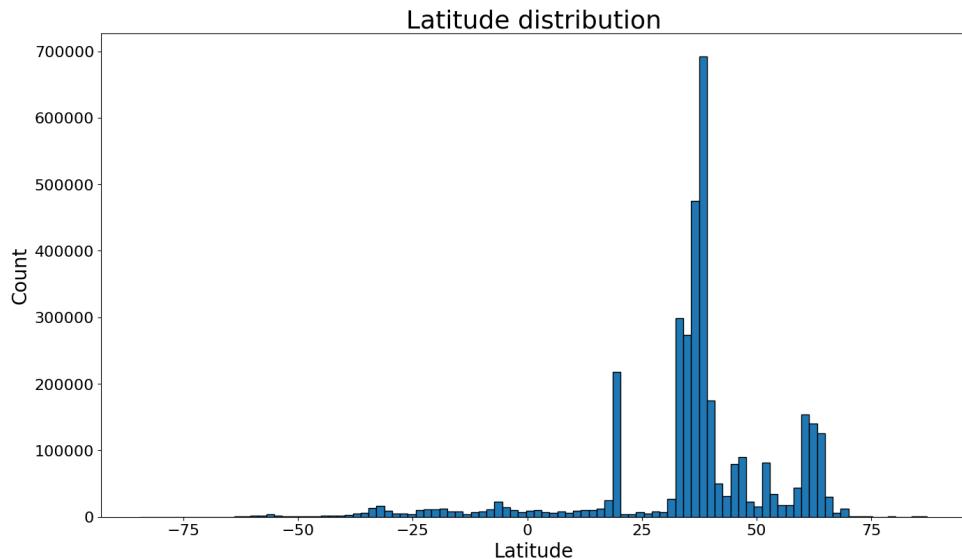


Rysunek 4.13: Rozkład zmiennej Longitude

Zmienna ta ma jasno określoną wartość minimalną oraz maksymalną, dlatego za pomocą MinMaxScalera została przeskalowana do zakresu od 0 do 1.

4.5.4. Latitude

Latitude to zmienna określająca szerokość geograficzną danego trzęsienia. Jej rozkład został przedstawiony na Rysunku 4.14.

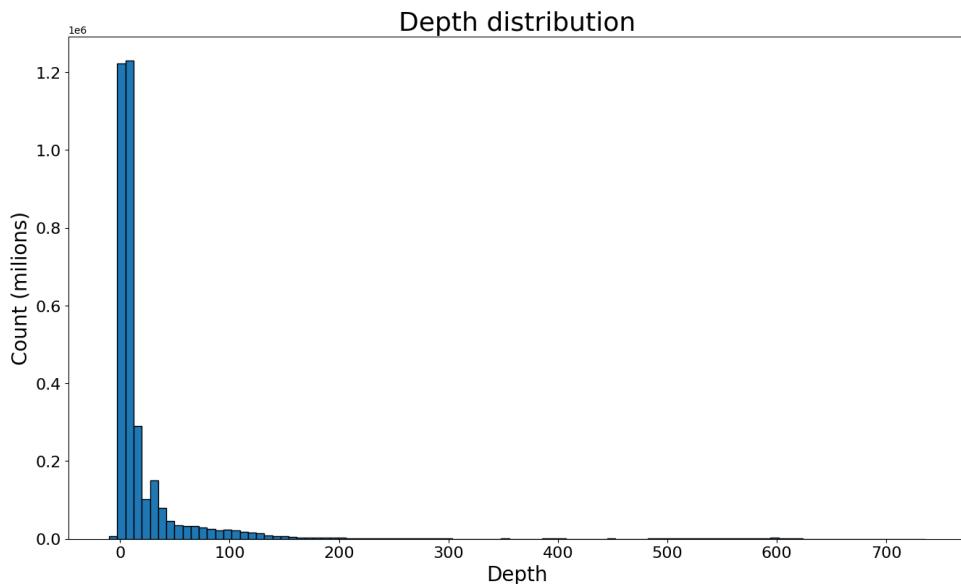


Rysunek 4.14: Rozkład zmiennej Latitude

Zmienna ta również ma jasno określoną wartość minimalną oraz maksymalną, dlatego za pomocą MinMaxScalera została przeskalowana do zakresu od 0 do 1.

4.5.5. Depth

Depth to zmienna określająca głębokość epicentrum trzęsienia ziemi. Jej rozkład został przedstawiony na Rysunku 4.15.

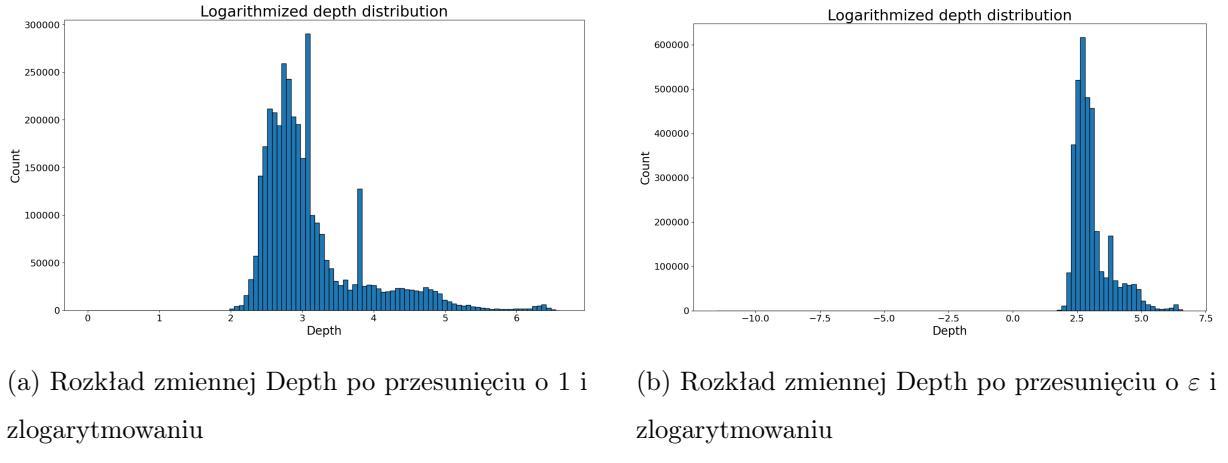


Rysunek 4.15: Rozkład zmiennej Depth

Na podstawie Rysunku 4.15 można stwierdzić, że większość wartości zmiennej mieści się w przedziale od -1 do 200 . Pojawiają się też jednak wartości dużo większe. Aby zawęzić rozkład została podjęta decyzja o zlogarytmowaniu zmiennej. Przed zlogarytmowaniem należy upewnić się, że zmienna ma jedynie wartości dodatnie. W tym celu zmienna została przesunięta o wartość bezwzględną z wartością minimalną w prawo. Zmodyfikowana zmienna posiada zatem wartość minimalną równą 0 . Następnie zostały rozważone dwie możliwości:

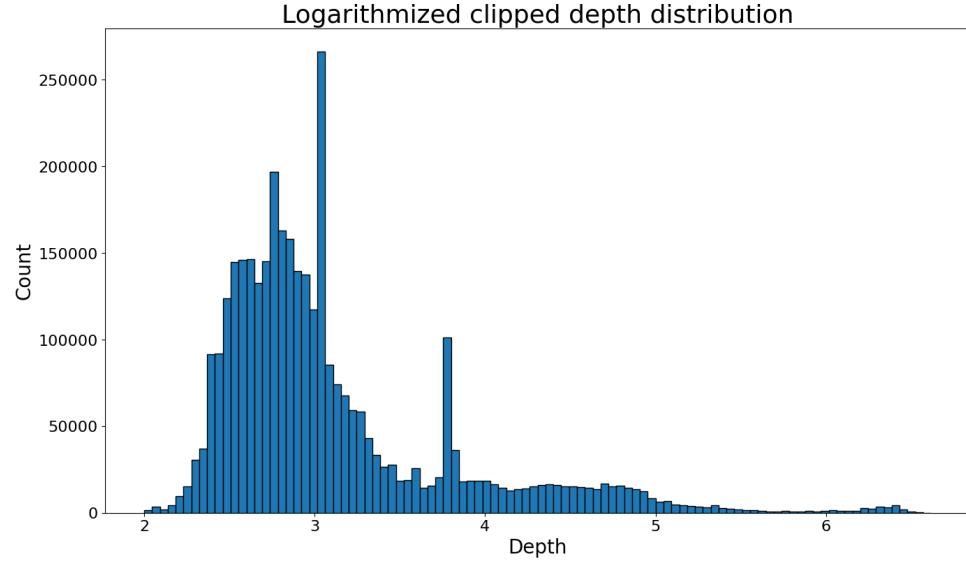
- Przesunięcie o 1 w prawo i zlogarytmowanie - Rysunek 4.16a
- Przesunięcie o $\varepsilon = 10^{-5}$ w prawo i zlogarytmowanie - Rysunek 4.16b

4.5. PRZYGOTOWANIE DANYCH



Rysunek 4.16: Rozkłady zmiennej Depth po przesunięciu i zlogarytmowaniu

Na podstawie Rysunku 4.16 można stwierdzić, że zlogarytmowanie faktycznie zawęziło rozkład zmiennej. Ostatecznie bardziej skuteczne okazało się przesunięcie o 1 w prawo i zlogarytmowanie przedstawione na Rysunku 4.16a. Następnie, aby dalej zawęzić rozkład, zmienna została przycięta do wartości minimalnej równej 2, a górna granica zmiennej pozostała niezmieniona. Rozkład zmiennej po przycięciu został przedstawiony na Rysunku 4.17.

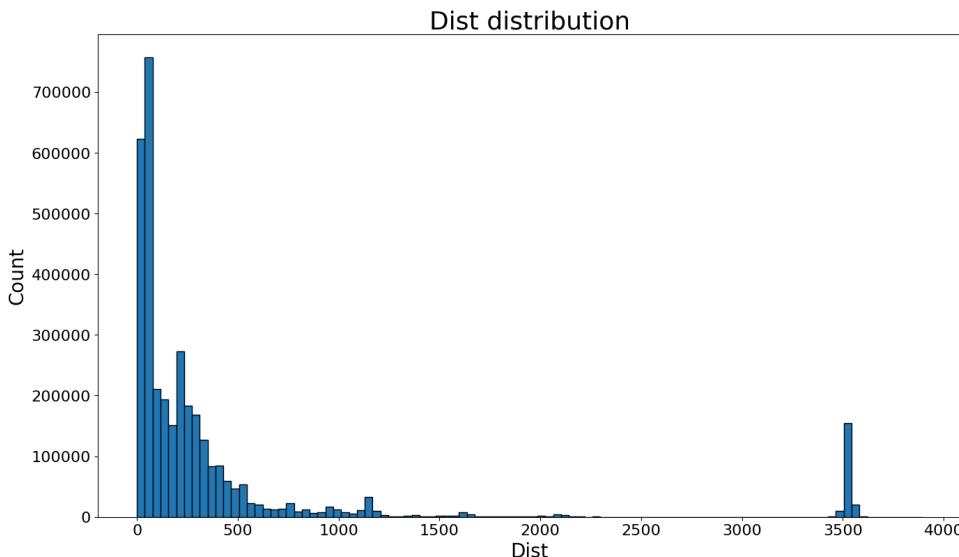


Rysunek 4.17: Rozkład zmiennej Depth po zlogarytmowaniu i przycięciu

Ostatecznie zmienna została przeskalowana do przedziału od 0 do 1 za pomocą MinMaxScaler-a.

4.5.6. Dist

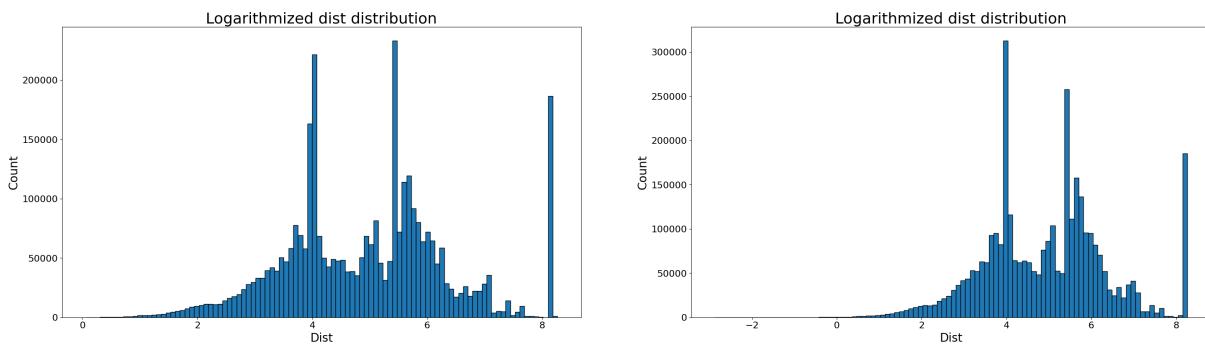
Dist to zmienna określająca odległość danego trzęsienia od najbliższej granicy płyt tektonicznych. Jej rozkład został przedstawiony na Rysunku 4.18.



Rysunek 4.18: Rozkład zmiennej Dist

Zmienna Dist posiada bardzo szeroki rozkład. Aby go obsłużyć została podjęta decyzja o zlogarytmowaniu zmiennej. Minimalna wartość zmiennej jest równa 0, dlatego, podobnie jak w przypadku zmiennej Depth, zostały rozważone dwie możliwości:

- Przesunięcie o 1 w prawo i zlogarytmowanie - Rysunek 4.19a
- Przesunięcie o $\varepsilon = 10^{-5}$ w prawo i zlogarytmowanie - Rysunek 4.19b



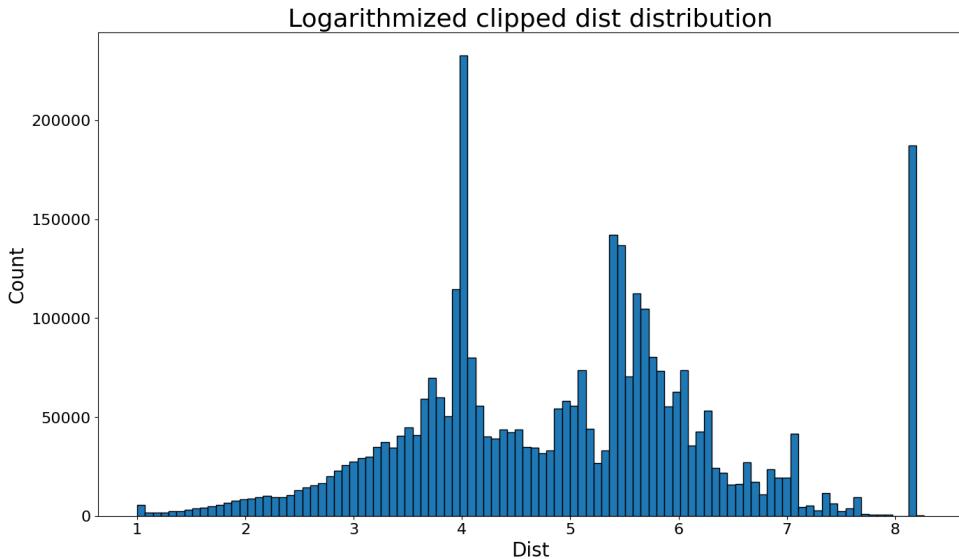
(a) Rozkład zmiennej Dist po przesunięciu o 1 i zlogarytmowaniu

(b) Rozkład zmiennej Dist po przesunięciu o ε i zlogarytmowaniu

Rysunek 4.19: Rozkłady zmiennej Dist po przesunięciu i zlogarytmowaniu

Na podstawie Rysunku 4.19 można stwierdzić, że logarytmowanie faktycznie zawęziło rozkład zmiennej. Ostatecznie bardziej skuteczne okazało się przesunięcie o 1 w prawo i zlogarytmowanie przedstawione na Rysunku 4.19a. Następnie zmienna została przycięta do wartości minimalnej równej 1, a górna granica rozkładu pozostała niezmieniona. Rozkład zmiennej po przycięciu został przedstawiony na Rysunku 4.20.

4.5. PRZYGOTOWANIE DANYCH

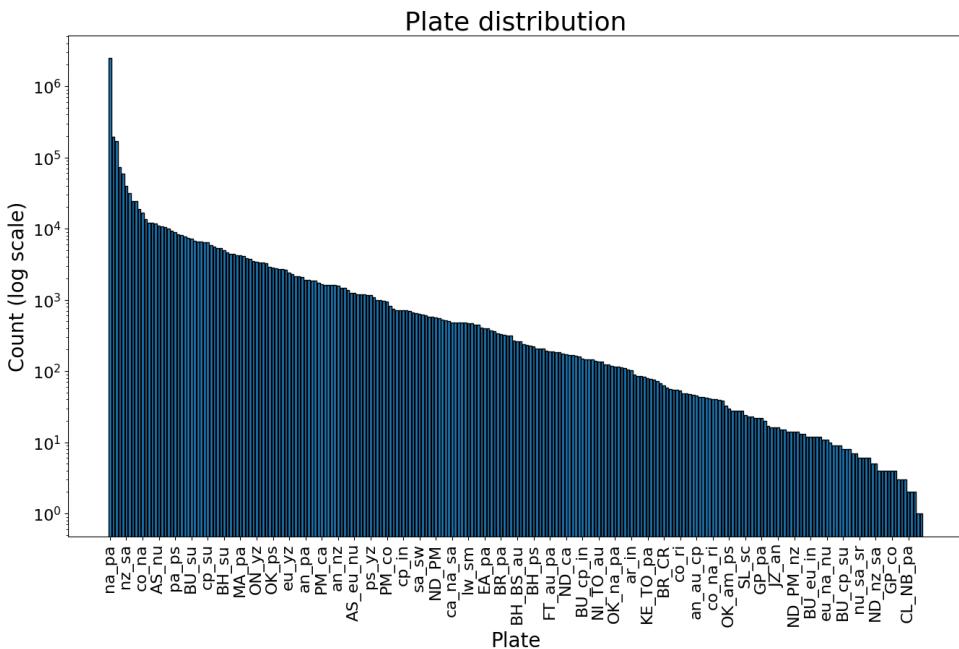


Rysunek 4.20: Rozkład zmiennej Dist po zlogarytmowaniu i przycięciu

Ostatecznie zmienna została przeskalowana do zakresu od 0 do 1 za pomocą MinMaxScalera.

4.5.7. Plate

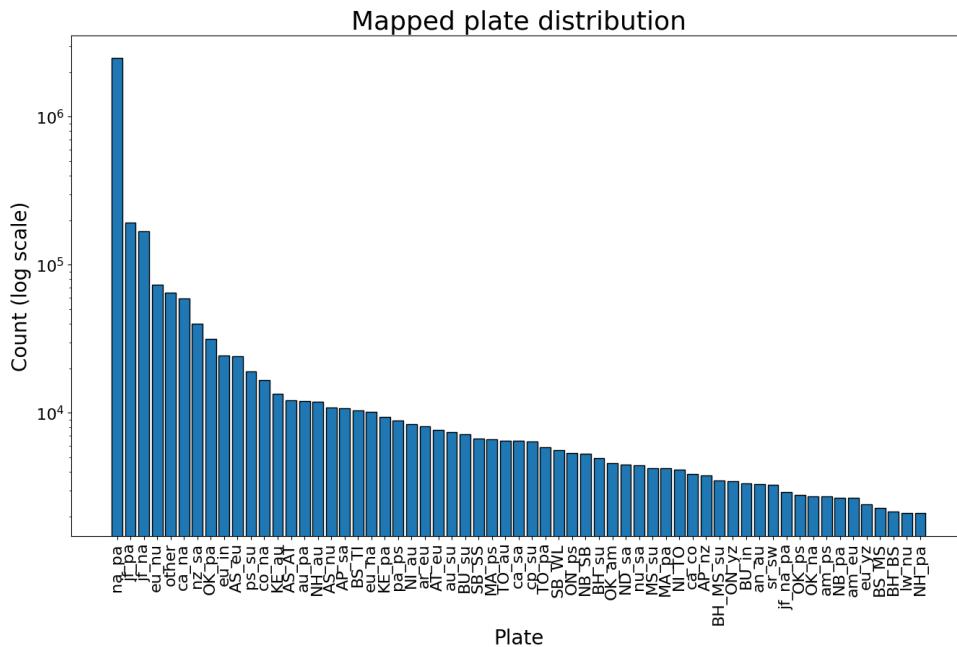
Plate to zmienna zawierająca tekstowy identyfikator najbliższej granicy płyt tektonicznych. Jej rozkład został przedstawiony na Rysunku 4.21.



Rysunek 4.21: Rozkład zmiennej Plate

Należy podkreślić, że zmienna ta posiada aż 250 różnych kategorii. Ze względu na czytelność, na osi X powyższego wykresu został podpisany jedynie co 5 słupek. Aby zmniejszyć liczbę kate-

gorii zostało zachowane jedynie 60 najbardziej licznych klas a pozostałe zostały zmapowane do jednej - "other". Na Rysunku 4.22 został przedstawiony rozkład po zmapowaniu.



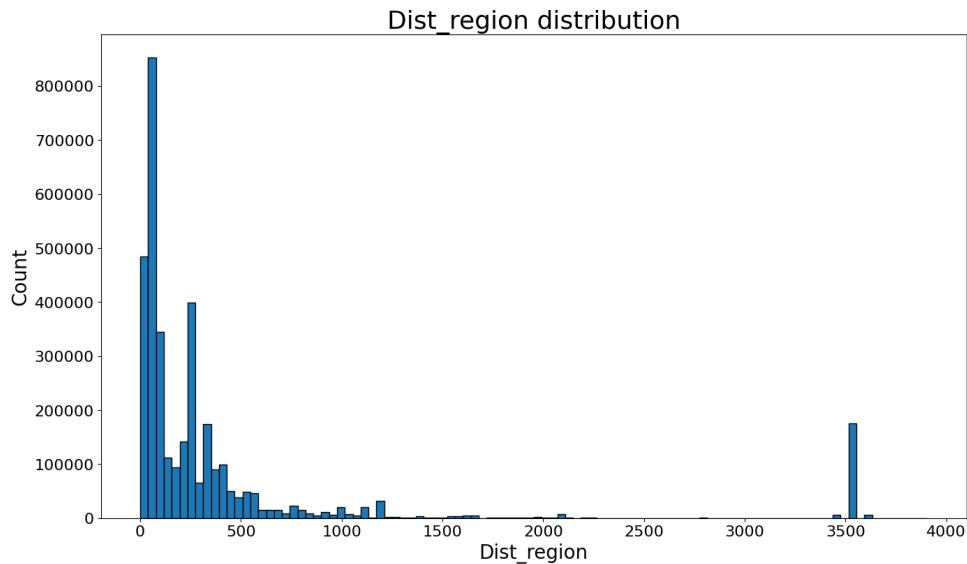
Rysunek 4.22: Rozkład zmiennej Plate po zmapowaniu

Ostatecznie poszczególne kategorie zostały zmapowane do kolejnych liczb naturalnych, a mapeowanie zostało zapisane do pliku .csv.

4.5.8. Dist_region

Dist_region to zmienna określająca odległość środka regionu, w którym znajduje się dane trzęsienia, od najbliższej granicy płyt tektonicznych. Na Rysunku 4.23 został przedstawiony rozkład zmiennej.

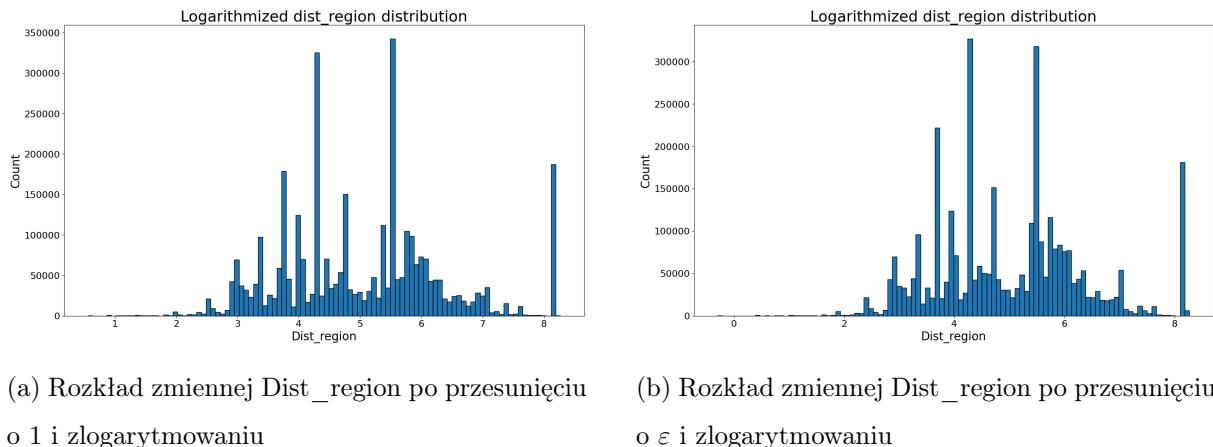
4.5. PRZYGOTOWANIE DANYCH



Rysunek 4.23: Rozkład zmiennej Dist_region

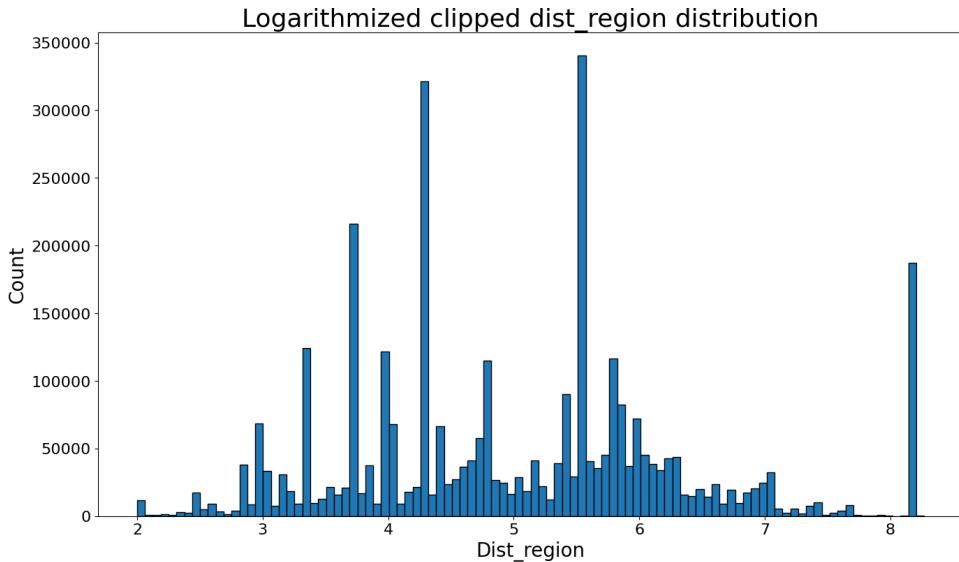
Rozkład zmiennej Dist_region jest bardzo podobny do rozkładu zmiennej Dist, dlatego zostanie przygotowany w ten sam sposób. Zostały rozważone dwa przesunięcia:

- Przesunięcie o 1 w prawo i zlogarytmowanie - Rysunek 4.24a
- Przesunięcie o $\varepsilon = 10^{-5}$ w prawo i zlogarytmowanie - Rysunek 4.24b



Rysunek 4.24: Rozkłady zmiennej Dist_region po przesunięciu i zlogarytmowaniu

Bardziej skuteczne okazało się przesunięcie o 1 w prawo i zlogarytmowanie przedstawione na Rysunku 4.24b. Następnie zmienna została przycięta do wartości minimalnej równej 2, a górna granica zmiennej pozostała niezmieniona. Rozkład zmiennej po przycięciu został przedstawiony na Rysunku 4.25.

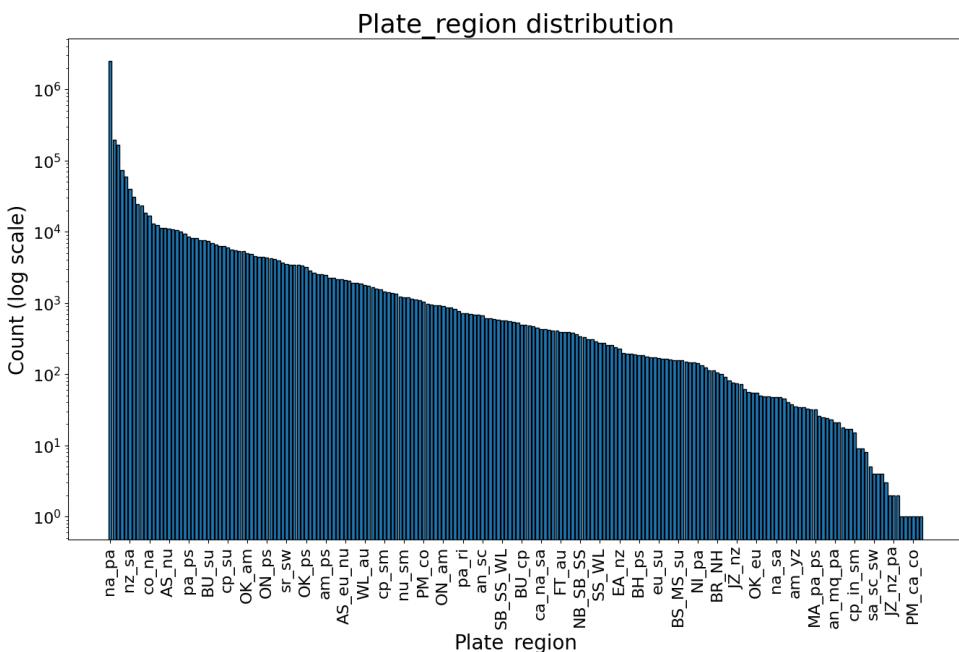


Rysunek 4.25: Rozkład zmiennej Dist_region po zlogarytmowaniu i przycięciu

Ostatecznie zmienna została przeskalowana do zakresu od 0 do 1 za pomocą MinMaxScalera.

4.5.9. Plate_region

Plate_region to zmienna zawierająca tekstowy identyfikator najbliższej granicy płyt tektonicznych. Jej rozkład został przedstawiony na Rysunku 4.26.



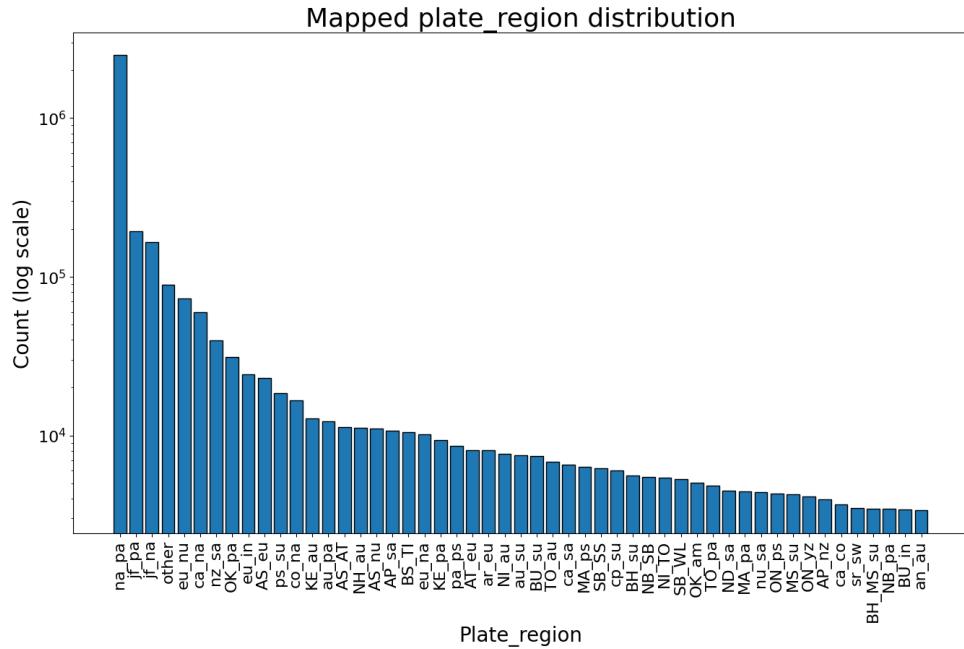
Rysunek 4.26: Rozkład zmiennej Plate_region

Należy podkreślić, że zmienna ta posiada aż 208 różnych kategorii. Na osi X powyższego wykresu została wypisana jedynie co 5 kategoria. Aby zmniejszyć liczbę kategorii zostało zacho-

4.5. PRZYGOTOWANIE DANYCH

wane jedynie 50 najbardziej licznych klas a pozostałe zostały zmapowane do jednej - "other".

Na wykresie 4.22 został przedstawiony rozkład po zmapowaniu.

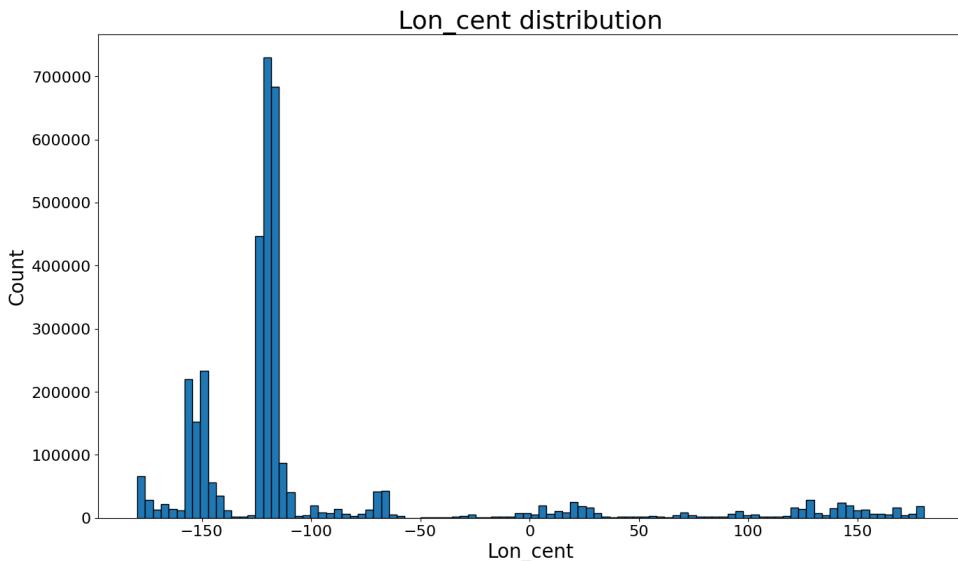


Rysunek 4.27: Rozkład zmiennej Plate_region po zmapowaniu

Ostatecznie poszczególne kategorie zostały zmapowane do kolejnych liczb naturalnych, a mapeowanie zostało zapisane do pliku .csv.

4.5.10. Lon_center

Lon_center to zmienna określająca długość geograficzną środka danego regionu. Jej rozkład został przedstawiony na Rysunku 4.28.

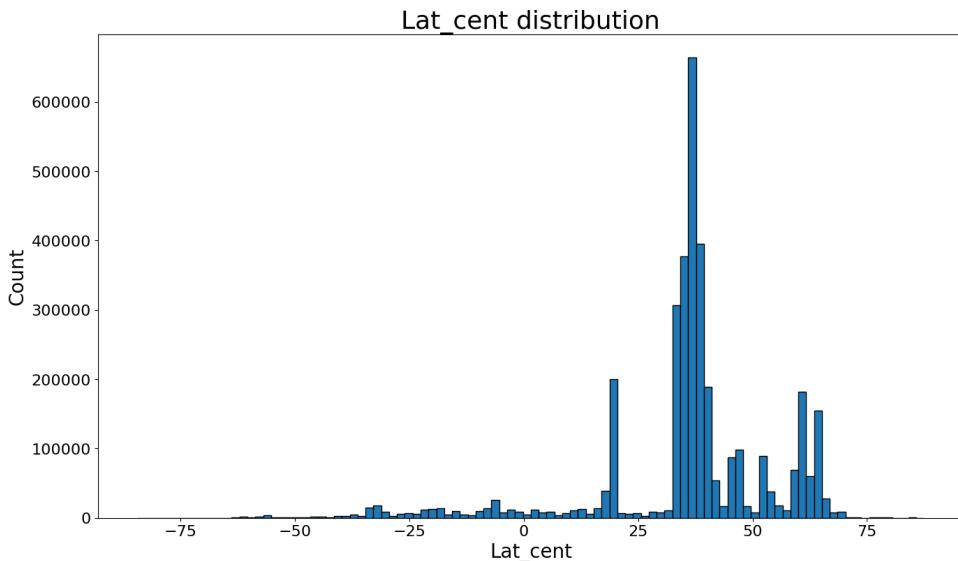


Rysunek 4.28: Rozkład zmiennej Lon_center

Zmienna ta ma jasno określoną wartość minimalną oraz maksymalną, dlatego została przeskalowana do zakresu od 0 do 1 przy pomocy MinMaxScalera.

4.5.11. Lat_center

Lat_center to zmienna określająca szerokość geograficzną środka danego regionu. Jej rozkład został przedstawiony na Rysunku 4.29.



Rysunek 4.29: Rozkład zmiennej Lat_center

Zmienna ta ma jasno określoną wartość minimalną oraz maksymalną, dlatego została przeskalowana do zakresu od 0 do 1 przy pomocy MinMaxScalera.

5. Metodologia

5.1. Tworzenie zbioru uczącego do modelu

Tworzenie zbioru uczącego zostało wykonane za pomocą bibliotek Pandas[6], NumPy[14] oraz Tensorflow[15]. Przewidywanie aktywności sejsmicznej zostało potraktowane jako problem klasyfikacji binarnej szeregów czasowych. Algorytm tworzenia szeregów czasowych na podstawie przygotowanego zbioru danych został przedstawiony poniżej:

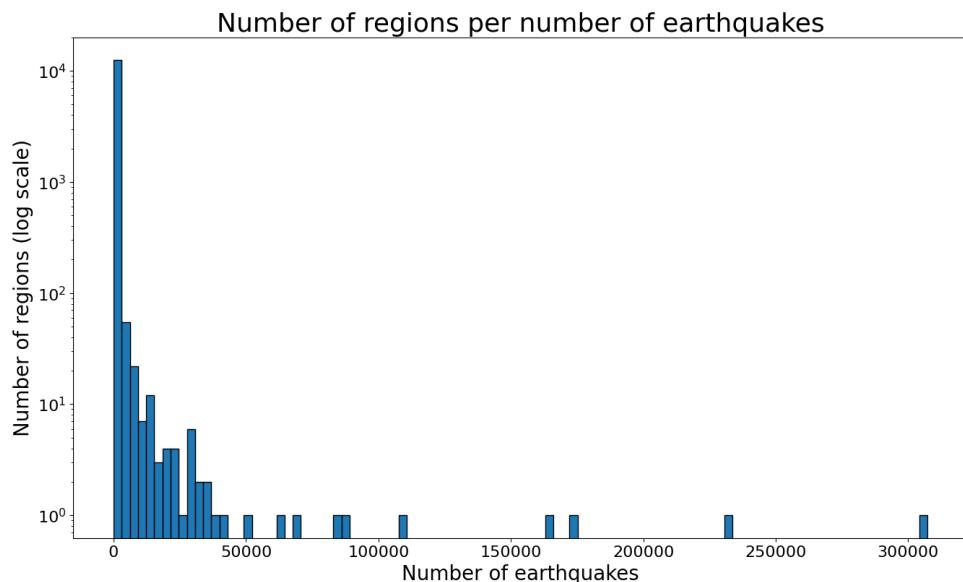
1. Iteracja po regionach 5.1.1
2. Dodanie zmiennej Distance 5.1.2
3. Wybór trzęsień w określonym promieniu od środka danego regionu 5.1.3
4. Sortowanie wybranych trzęsień rosnąco po dacie 5.1.4
5. Dodanie zmiennej Diff_days oraz przygotowanie zmiennych Distance i Diff_days 5.1.5
6. Dodanie zmiennych z określonej liczby wcześniejszych trzęsień - tworzenie szeregu czasowego 5.1.6
7. Podział na zbiór treningowy, walidacyjny i testowy 5.1.7
8. Stworzenie 3 tablic dla każdego ze zbiorów - szereg czasowy, informacje regionalne, etykiety 5.1.8
9. Przekształcenie tablic do odpowiednich wymiarów 5.1.9
10. Zapisanie tablic do plików .npy 5.1.10

Dokładny opis każdego z kroków algorytmu tworzenia szeregów czasowych został przedstawiony w następujących sekcjach.

5.1.1. Iteracja po regionach

Szeregi czasowe będą konstruowane dla każdego z regionów oddzielnie. Podczas iteracji po regionach należy zwrócić uwagę na liczbę trzęsień w danym regionie. W późniejszych etapach algorytmu zbyt mała liczba trzęsień w regionie będzie powodować generowanie niepełnych szeregów czasowych.

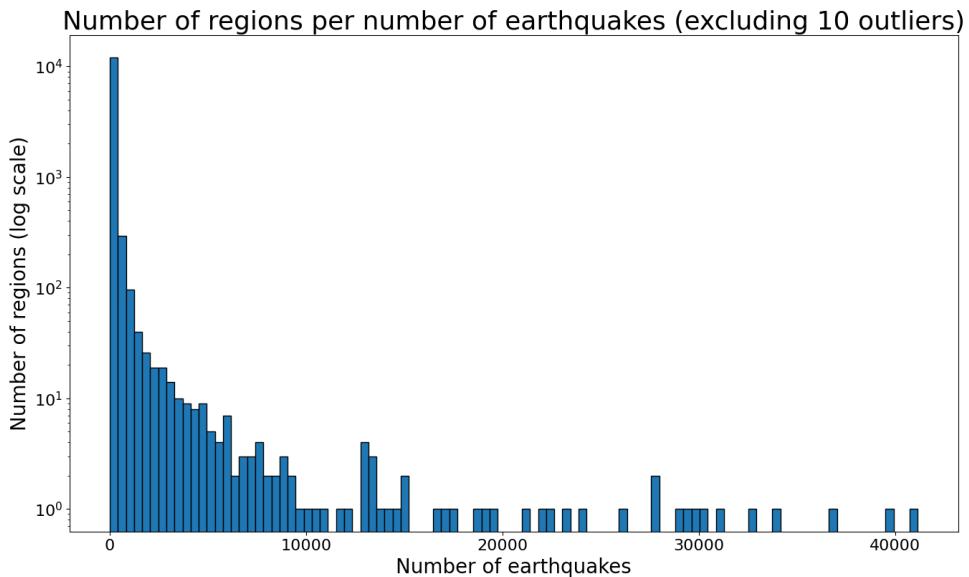
Do poniższych wizualizacji zostały wykorzystane jedynie dane znajdujące się w zbiorze treningowym. Na Rysunku 5.1 została przedstawiona liczba regionów o danej liczbie trzęsień.



Rysunek 5.1: Liczba regionów o danej liczbie trzęsień

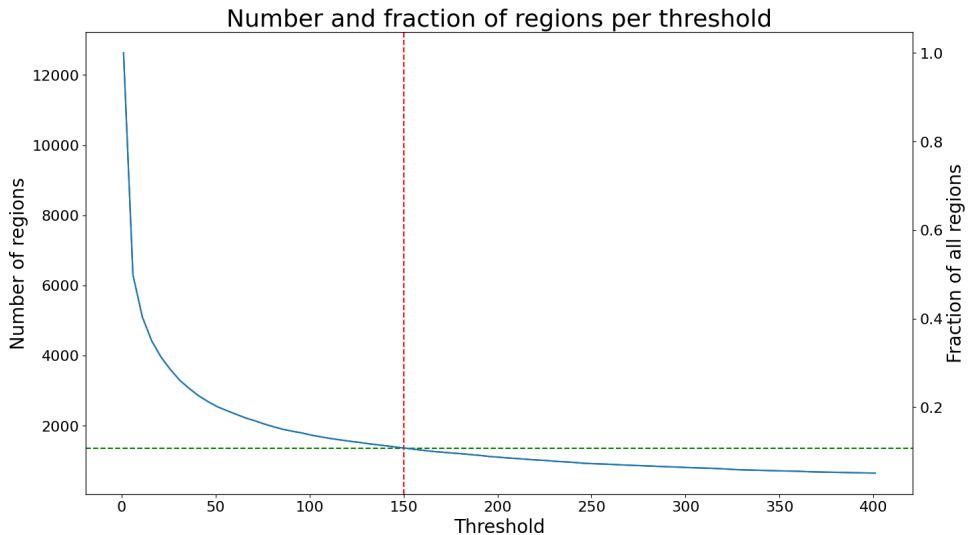
W dużej części regionów znajduje się niewiele trzęsień, jednak istnieją również regiony zawierające setki tysięcy trzęsień. Aby lepiej zwizualizować rozkład liczby regionów o danej liczbie trzęsień został przygotowany Rysunek 5.2. Nie zawiera on 10 regionów o największej liczbie trzęsień.

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU



Rysunek 5.2: Liczba regionów o danej liczbie trzęsień bez 10 największych wartości

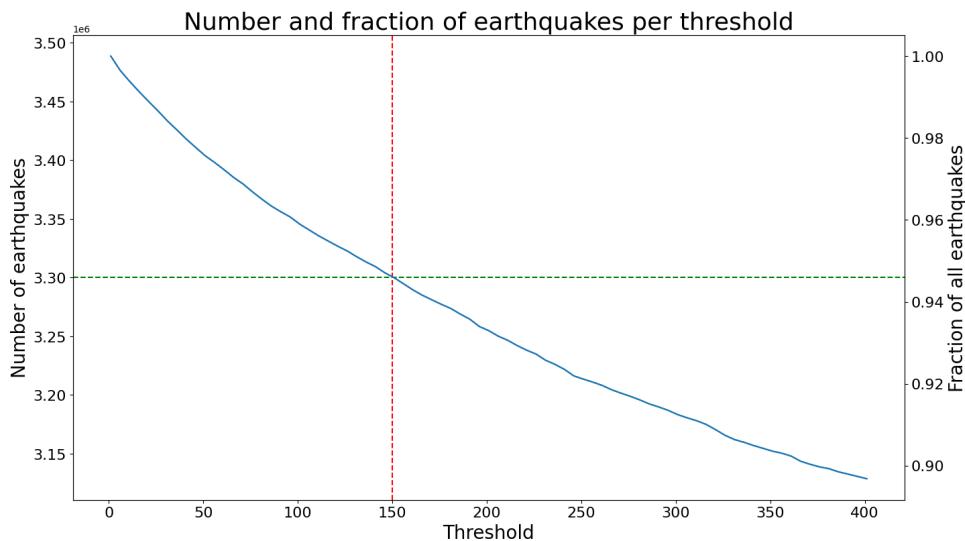
Aby zapobiec generowaniu niepełnych szeregów czasowych została podjęta decyzja o odfiltrowaniu regionów z niewystarczającą liczbą trzęsień. Aby odpowiednio wybrać graniczą liczbę trzęsień, potrzebną, aby region nie został odfiltrowany, zostały przygotowane Rysunki 5.3 oraz 5.4.



Rysunek 5.3: Liczba oraz procent regionów do threshold

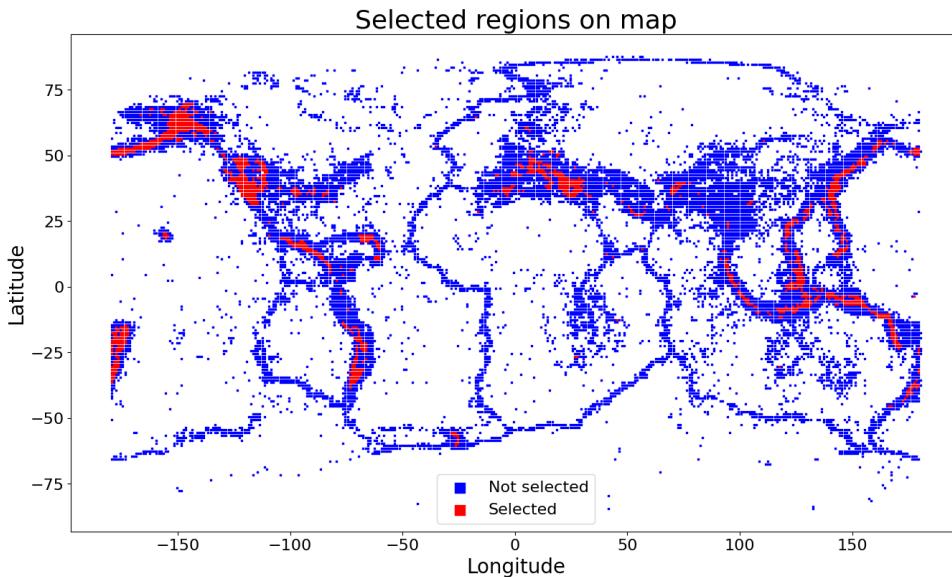
Na Rysunku 5.3 została przedstawiona liczba oraz część wszystkich regionów zawierająca przynajmniej *Threshold* trzęsień. Na podstawie Rysunku 5.3 można stwierdzić, że około 80% regionów zawiera od 1 do 50 trzęsień, natomiast około 90% regionów zawiera do 150 trzęsień.

Na Rysunku 5.4 została przedstawiona liczba oraz część wszystkich trzęsień znajdujących się w regionach o przynajmniej *Threshold* trzęsieniach.



Rysunek 5.4: Liczba oraz procent trzęsień do threshold

Ostatecznie wartość graniczna *Threshold* została ustalona na 150. Na podstawie Rysunków 5.3 i 5.4 można stwierdzić, że pomimo odfiltrowania około 90% regionów zostało zachowane prawie 95% trzęsień. Na Rysunku 5.5 został przedstawiony rozkład wybranych regionów na mapie.



Rysunek 5.5: Rozkład wybranych regionów na mapie

Na Rysunku 5.5 czerwone kwadraty oznaczają wybrane regiony, natomiast niebieskie kwadraty oznaczają niewybrane regiony. Podsumowując, szeregi czasowe będą konstruowane dla 1 362 regionów, co jest równoważne z tworzeniem predykcji dla danych regionów.

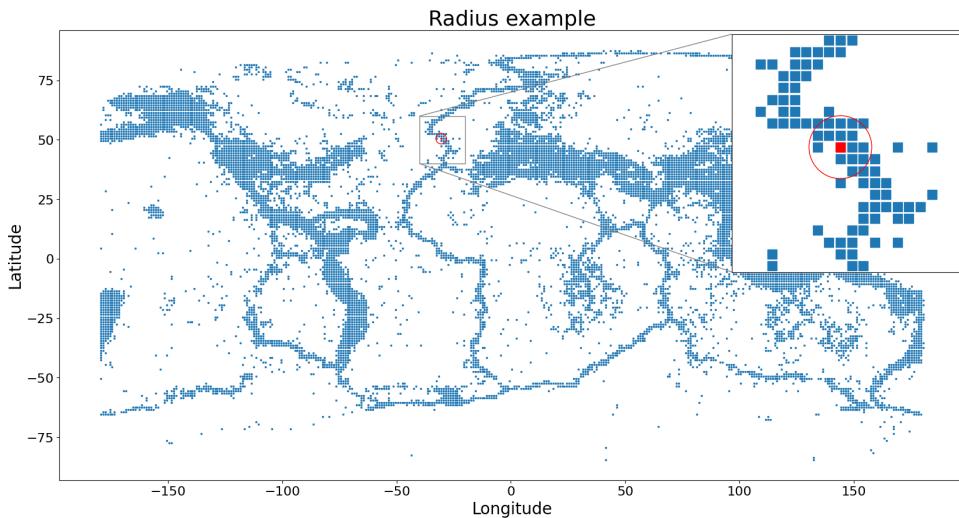
5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU

5.1.2. Dodanie zmiennej Distance

Zmienna Distance wyraża odległość (w kilometrach) trzęsienia od środka danego regionu. Została ona policzona za pomocą formuły Haversine[11]. Należy podkreślić, że zmienna ta jest liczona dopiero na tym etapie, ponieważ jej wartości są różne dla tego samego trzęsienia - w zależności od środka regionu, względem którego jest obliczana.

5.1.3. Wybór trzęsień w określonym promieniu od środka danego regionu

Szeregi czasowe dla trzęsień w danym regionie będą uwzględniać nie tylko trzęsienia z danego regionu, lecz również w określonym promieniu od środka danego regionu znajdujące się poza danym regionem. Zmienna Distance, dodana w kroku 5.1.2, zostanie wykorzystana do wybrania trzęsień znajdujących się w określonym promieniu od środka danego regionu. Takie podejście powoduje, że niektóre trzęsienia będą występować w różnych szeregach czasowych. Wydaje się to logiczne, ponieważ pojedyncze trzęsienie ma wpływ nie tylko na region, w którym wystąpiło, lecz również na sąsiadujące regiony. Ostatecznie promień został ustalony na 300 kilometrów. Na Rysunku 5.6 każdy kwadrat odpowiada jednemu regionowi. Została przedstawiona też wizualizacja pokazująca promień 300 kilometrów dla jednego z regionów.



Rysunek 5.6: Okrąg o promieniu 300 kilometrów dla przykładowego regionu

5.1.4. Sortowanie wybranych trzęsień rosnąco po dacie

Takie sortowanie znacząco ułatwia tworzenie szeregów czasowych przedstawione w kroku 5.1.6. W Tabeli 5.1 został przedstawiony przykładowy widok posortowanej tabeli zawierającej informacje o trzęsieniach z jednego regionu wraz z trzęsieniami znajdującymi się w promieniu 300 kilometrów od środka tego regionu.

time	longitude	latitude	...	pos	label	...	distance
1973-07-27 17:48:13.100	106.93	-8.9740	...	-9_106	-1	...	85.48
1974-03-15 07:37:13.500	107.02	-9.9180	...	-10_107	0	...	69.46
1974-03-25 17:31:45.000	108.93	-8.2800	...	-9_108	-1	...	207.97
...
2023-08-13 04:46:35.770	107.13	-10.0480	...	-11_107	-1	...	73.11
2023-09-05 16:35:53.061	108.86	-8.5443	...	-9_108	-1	...	184.08
2023-09-30 08:33:43.391	105.48	-9.0912	...	-10_105	-1	...	226.10

Tabela 5.1: Przykładowa tabela zawierająca posortowane trzęsienia

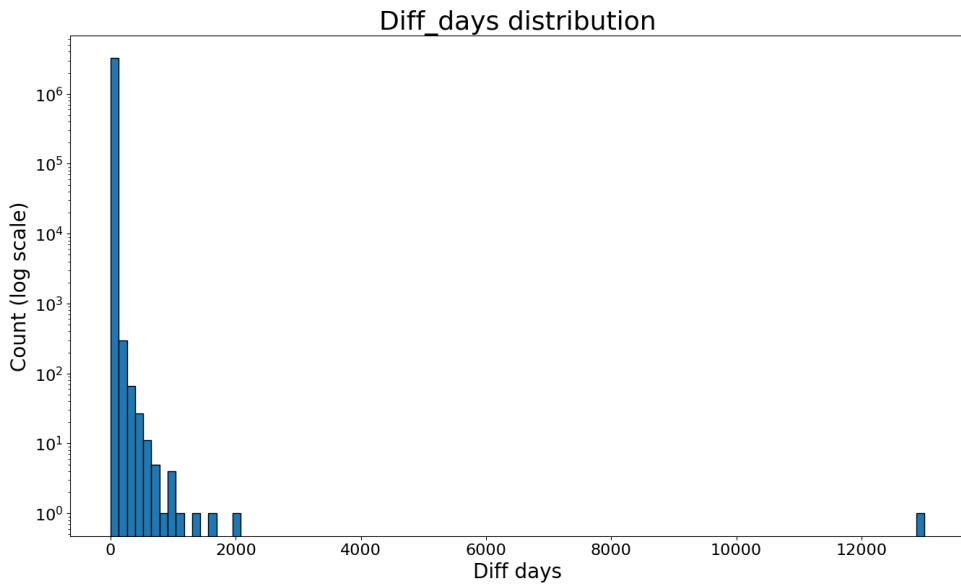
Aby utworzyć szereg czasowy o długości 2 w tak posortowanej tabeli wystarczy do każdego trzęsienia dodać nowe kolumny z informacjami na temat trzęsienia znajdującego się w rekordzie poprzedzającym dane trzęsienie. Aby utworzyć szereg czasowy o długości *block_size* wystarczy dodać kolumny z informacjami o trzęsieniach z rekordów od 1 do *block_size* – 1 nad danym trzęsieniem. API biblioteki Pandas[6] posiada pojedynczą funkcję pozwalającą na wykonanie opisanej powyżej operacji - *shift()*. Funkcja *shift()* pozwala na przesunięcie wartości kolumny o dowolną liczbę wierszy w góre lub w dół tabeli. Należy również podkreślić, że trzęsieniom spoza danego regionu, lecz leżącym w promieniu do 300 kilometrów od środka tego regionu, etykiety zostały tymczasowo ustalone na –1. Dzięki temu można odróżnić szeregi czasowe dla danego regionu od szeregów czasowych spoza tego regionu. Dla ustalonego regionu chcemy wykonywać predykcję tylko dla szeregów z tego regionu. Szeregi mogą jednak zawierać trzęsienia spoza tego regionu (w promieniu 300 kilometrów).

5.1.5. Dodanie zmiennej *Diff_days* oraz przygotowanie zmiennych *Distance* i *Diff_days*

Zmienne *Distance* i *Diff_days* zostały dodane dopiero podczas tworzenia szeregów czasowych. Wynika to z faktu, że ich wartości są różne w zależności od tego, do którego szeregu czasowego przynależą.

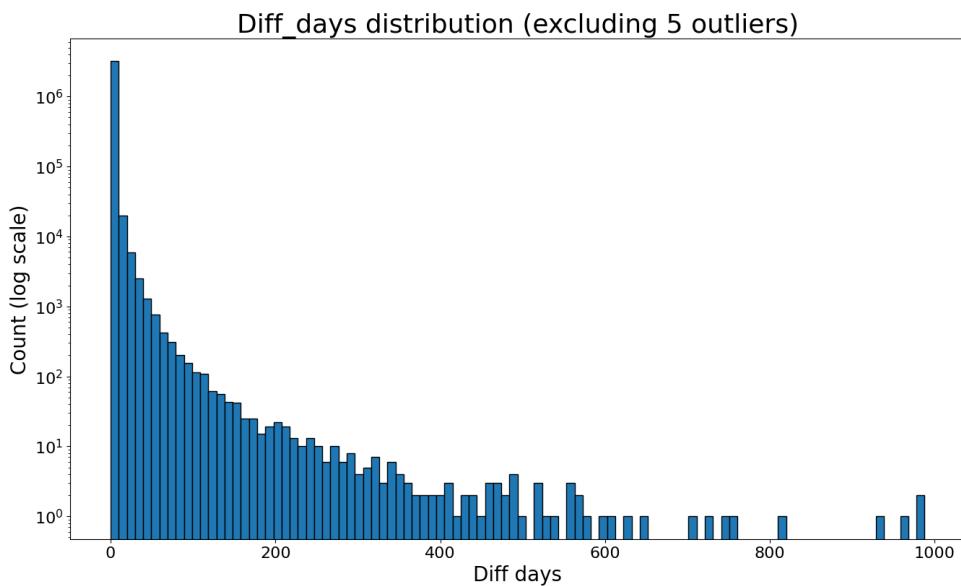
Zmienna *Diff_days* to różnica w dniach między czasem wystąpienia kolejnych trzęsień w tym samym regionie (wraz z trzęsieniami znajdującymi się w promieniu 300 kilometrów od środka tego regionu). Jej rozkład został przedstawiony na Rysunku 5.7.

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU



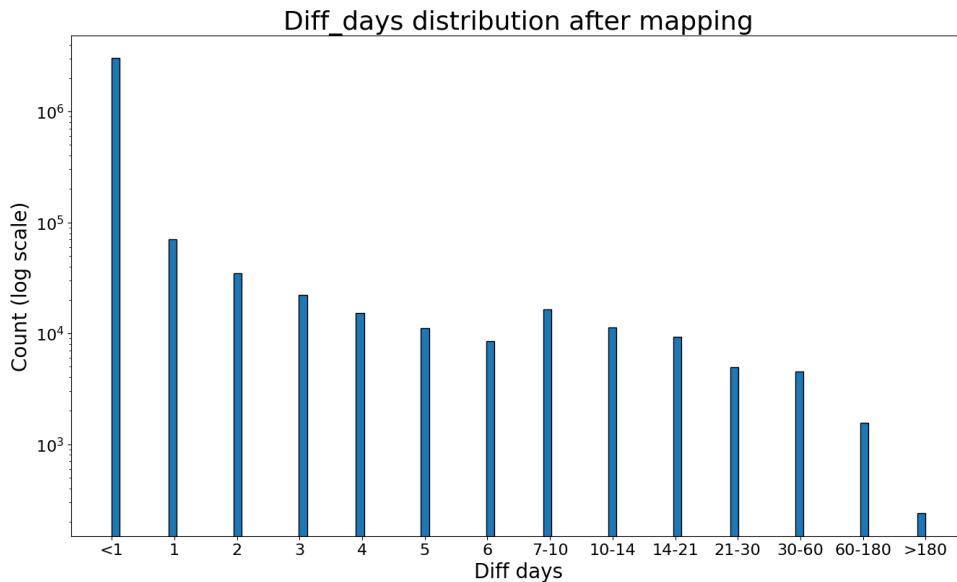
Rysunek 5.7: Rozkład zmiennej Diff_days

Na podstawie Rysunku 5.7 można stwierdzić, że większość wartości mieści się w przedziale od 0 do 1 000. Dodatkowo pojawia się jedna wartość równa 13 011, która znaczaco odbiega od rozkładu pozostałych. Aby lepiej zwizualizować rozkład zmiennej, został również przygotowany wykres z wykluczeniem 5 największych wartości. Został on przedstawiony na Rysunku 5.8.



Rysunek 5.8: Rozkład zmiennej Diff_days bez 5 największych wartości

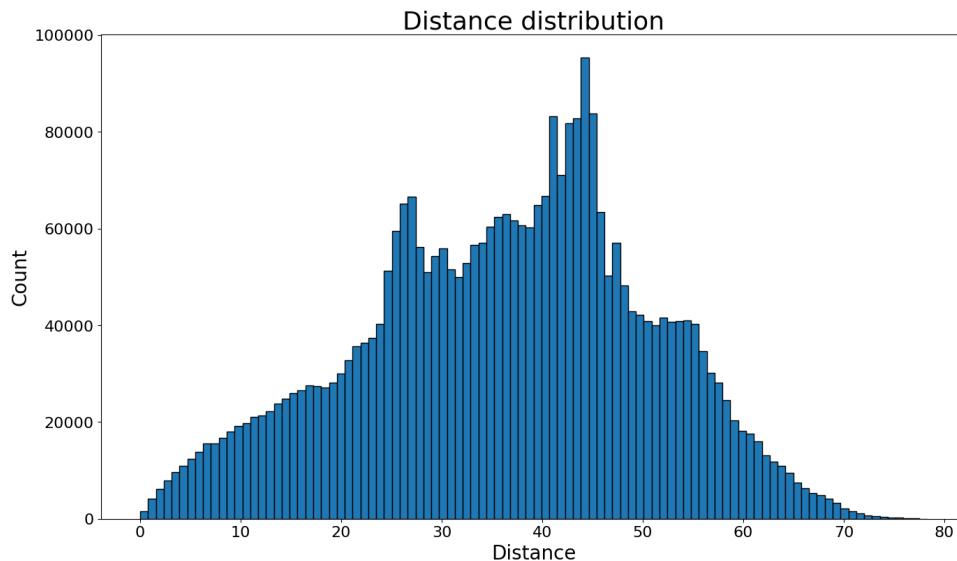
Należy podkreślić, że zmienna Diff_days jest zmienną dyskretną. Jej wartości to jedynie liczby naturalne. Aby przygotować zmienną Diff_days została podjęta decyzja o zmapowaniu jej do 14 klas. Finalny rozkład zmiennej został przedstawiony na Rysunku 5.9.



Rysunek 5.9: Rozkład zmiennej Diff_days po zmapowaniu

Takie mapowanie pozwala na zniwelowanie wpływu wartości odstających, jednocześnie zachowując różnorodność danych w bardziej licznych przedziałach.

Zmienna Distance to odległość (w kilometrach) trzęsienia od środka danego regionu. Jej rozkład został przedstawiony na Rysunku 5.10.



Rysunek 5.10: Rozkład zmiennej Distance

Należy podkreślić, że zmienna Distance mieści się w przedziale od 0 do 78.28. Taki rozkład powoduje, że przeskalowanie zmiennej do przedziału od 0 do 1 jest równoważne podzieleniu wszystkich wartości zmiennej przez wartość maksymalną, w tym przypadku przez 78.28. Zmienna Distance została przeskalowana do zakresu od 0 do 1.

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU

5.1.6. Dodanie zmiennych z określonej liczby wcześniejszych trzęsień - tworzenie szeregu czasowego

Zostało wybrane 9 zmiennych, które będą wchodzić w skład szeregów czasowych:

- Magnitude - Skala trzęsienia
- Depth - Głębokość epicentrum
- Latitude - Szerokość geograficzna
- Longitude - Długość geograficzna
- Dist - Odległość od najbliższej granicy płyt tektonicznych
- Distance - Odległość od środka regionu
- Plate - Identyfikator granicy płyt tektonicznych
- Diff_days - Różnica (w dniach) między czasem wystąpienia kolejnych trzęsień
- magType - Rodzaj skali trzęsienia ziemi

Aby stworzyć szereg czasowy o długości *block_size* należy dodać kolumny z informacjami o trzęsieniach znajdujących się w rekordach od 1 do *block_size* – 1 nad danym trzęsieniem. Na Rysunku 5.2 został przedstawiony widok przykładowej tabeli zawierającej szereg czasowy o długości 2 dla jednego z regionów.

30 kolumn tabeli 5.2 zostało opisanych w Tabeli 5.3.

time	longitude	latitude	depth	mag	...	mag_1	depth_1	...
1974-03-15 07:37:13.500	107.03	-9.92	0.51	0.76	...	0.81	0.49	...
1974-09-04 07:04:50.700	107.57	-9.03	0.52	0.80	...	0.70	0.39	...
1977-08-28 13:14:17.800	107.54	-9.71	0.39	0.68	...	0.84	0.39	...
...
2022-04-04 23:51:49.680	107.17	-9.39	0.28	0.66	...	0.79	0.30	...
2022-07-14 10:39:37.467	107.13	-9.84	0.23	0.62	...	0.68	0.23	...
2023-03-29 13:21:33.992	107.01	-9.24	0.23	0.68	...	0.62	0.24	...

Tabela 5.2: Tabela zawierająca szereg czasowy o długości 2 dla jednego z regionów

Kolumna	Typ danych	Opis
time	date	Czas trzęsienia ziemi
longitude	float	Długość geograficzna
latitude	float	Szerokość geograficzna
depth	float	Głębokość epicentrum trzęsienia
mag	type	Skala trzęsienia
magType	int	Rodzaj skali trzęsienia
time_disc	date	Zdyskretyzowana data trzęsienia, co 1 dzień
longitude_disc	int	Zdyskretyzowana długość geograficzna, co 1°
latitude_disc	int	Zdyskretyzowana szerokość geograficzna, co 1°
pos	string	Tekstowy identyfikator regionu
lat_cent	float	Szerokość geograficzna środka regionu
lon_cent	float	Długość geograficzna środka regionu
plate_region	int	Identyfikator granicy płyt tektonicznych
dist_region	float	Odległość środka regionu (w kilometrach) od najbliższej granicy płyt tektonicznych
dist	float	Odległość trzęsienia (w kilometrach) od najbliższej granicy płyt tektonicznych

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU

plate	int	Identyfikator najbliższej granicy płyt tektonicznych
label	int	Etykieta - 4.4
latitude_new	float	Zpreprocesowana szerokość geograficzna trzęsienia
longitude_new	float	Zpreprocesowana długość geograficzna trzęsienia
distance	float	Odległość (w kilometrach) trzęsienia od środka danego regionu
diff_days	int	Różnica (w dniach) między czasem wystąpienia trzęsienia oraz wcześniejszego trzęsienia
mag_1	float	Skala poprzedniego trzęsienia
depth_1	float	Głębokość epicentrum poprzedniego trzęsienia
latitude_new_1	float	Zpreprocesowana szerokość geograficzna poprzedniego trzęsienia
longitude_new_1	float	Zpreprocesowana długość geograficzna poprzedniego trzęsienia
dist_1	float	Odległość poprzedniego trzęsienia (w kilometrach) od najbliższej granicy płyt tektonicznych
distance_1	float	Odległość (w kilometrach) poprzedniego trzęsienia od środka regionu
plate_1	int	Identyfikator najbliższej granicy płyt tektonicznych, od poprzedniego trzęsienia
diff_days_1	int	Różnica (w dniach) między czasem wystąpienia trzęsienia poprzedniego oraz jeszcze wcześniejszego
magType_1	int	Rodzaj skali poprzedniego trzęsienia

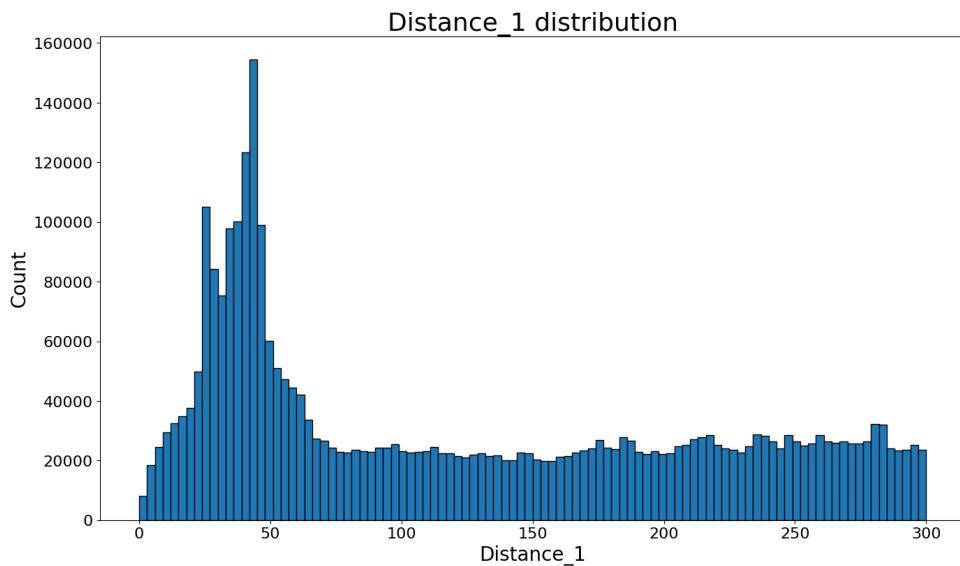
Tabela 5.3: Opis kolumn Tabeli 5.2

Wybierając odpowiednie podzbiory kolumn można otrzymać: szereg czasowy o długości 2, informacje o regionie, w którym miało miejsce trzęsienie, oraz etykietę, która będzie przewidywana przez model. Ostatecznie testowane będą szeregi czasowe o długości $block_size = 64$.

Należy podkreślić, że pierwsze $block_size$ wierszy będzie zawierać braki danych. Dzieje się tak, ponieważ dla pierwszych $block_size$ trzęsień nie ma wystarczającej liczby przeszłych trzęsień, aby utworzyć pełny szereg czasowy. Przykładowo, dla pierwszego trzęsienia w danym regionie nie ma wcześniejszych trzęsień, na podstawie których można by stworzyć szereg czasowy o długości $block_size$. Właśnie na tym etapie zostaje wykorzystana, ustalona w kroku 5.1.1, minimalna liczba trzęsień równa $Threshold = 150$. Taka liczba zapewnia, że dla każdego z regionów powstanie przynajmniej $Threshold - block_size = 150 - 64 = 86$ szeregów czasowych. Wiersze zawierające niepełne szeregi czasowe zostały usunięte.

Należy również pokreślić, że rozkład zmiennych kolumnach z sufiksem 1 są zbliżone do roz-

kładów oryginalnych zmiennych. Wyjątkiem od tej zasady jest rozkład zmiennej Distance_1. Na Rysunku 5.11 został przedstawiony rozkład zmiennej Distance_1.



Rysunek 5.11: Rozkład zmiennej Distance_1

Rozkład zmiennej Distance_1 różni się od rozkładu zmiennej Distance, przedstawionego na Rysunku 5.10, ponieważ do zmiennej Distance zaliczają się jedynie te trzęsienia, które odbyły się w danym regionie. Do zmiennej Distance_1 natomiast zaliczają się również trzęsienia znajdujące się w promieniu 300 kilometrów od środka danego regionu. Zmienna zawiera się w przedziale od 0 do 300. Aby przeskalać ją do przedziału od 0 do 1, należy podzielić wszystkie wartości przez wartość maksymalną, w tym przypadku przez 300.

5.1.7. Podział na zbiór treningowy, walidacyjny i testowy

Tabela na Rysunku 5.2 zostaje podzielona na 3 tabele reprezentujące zbiór treningowy, walidacyjny i testowy:

- zbiór treningowy - od 1973 do końca 2020 roku
- zbiór walidacyjny - od 2021 do końca 2022 roku
- zbiór testowy - od 2023 roku

5.1.8. Stworzenie 3 tablic dla każdego ze zbiorów - szereg czasowy, informacje regionalne, etykiety

Z każdej z tabel reprezentujących zbiór treningowy, walidacyjny i testowy, zostały wybrane 3 podzbiory kolumn, tworzące:

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU

- Szereg czasowy - kolumny:

- mag_1
- depth_1
- latitude_new_1
- longitude_new_1
- dist_1
- distance_1
- plate_1
- diff_days_1
- magType_1
- mag
- depth
- latitude_new
- longitude_new
- dist
- distance
- plate
- diff_days
- magType

- Informacje regionalne - kolumny:

- lat_cent
- lon_cent
- dist_region
- plate_region

- Etykiety - kolumna label

W Tabelach 5.4, 5.5 oraz 5.6 zostały przedstawione odpowiednio szereg czasowy, informacje regionalne oraz etykiety.

mag	depth	latitude_new	...	mag_1	depth_1	latitude_new_1	...
0.76	0.51	0.43	...	0.81	0.49	0.44	...
0.80	0.52	0.44	...	0.70	0.39	0.44	...
...
0.61	0.39	0.44	...	0.84	0.39	0.43	...
0.76	0.39	0.44	...	0.61	0.39	0.44	...

Tabela 5.4: Przykładowa tabela zawierająca szereg czasowy o długości 2

lat_cent	lon_cent	dist_region	plate_region
0.44	0.80	0.38	25.00
0.44	0.80	0.38	25.00
...
0.44	0.80	0.38	25.00
0.44	0.80	0.38	25.00

Tabela 5.5: Przykładowa tabela zawierająca informacje regionalne

label
0
0
...
0
0

Tabela 5.6: Przykładowa tabela zawierająca etykiety

Ostatecznie powstało 9 tablic, po 3 dla każdego ze zbiorów: treningowego, walidacyjnego oraz testowego.

5.1.9. Przekształcenie tablic do odpowiednich wymiarów

Większość modeli stosowanych do klasyfikacji szeregów czasowych wymaga określonego wymiaru danych wejściowych oraz przewidywanych etykiet. Szeregi czasowe, informacje regionalne oraz etykiety należy przekształcić do tablicy o wymiarach, odpowiednio, $(n_samples, block_size, n_features)$, $(n_samples, n_features_reg)$ oraz $(n_samples, 1)$, gdzie:

5.1. TWORZENIE ZBIORU UCZĄCEGO DO MODELU

- $n_samples$ - liczba szeregów czasowych w danym regionie (indywidualna dla każdego regionu)
- $block_size$ - liczba trzęsień uwzględnianych w szeregach czasowych, $block_size = 64$
- $n_features$ - liczba zmiennych opisujących każde trzęsienie w szeregu czasowym, $n_features = 9$
- $n_features_reg$ - liczba zmiennych opisujących dany region, $n_features = 4$

5.1.10. Zapisanie szeregów czasowych do plików .npy

Wszystkie 9 tablic, o odpowiednich wymiarach, zostaje zapisanych do oddzielnych plików w formacie .npy w tym samym folderze. Pliki nazywane są w następującej konwencji:

- x_split_idx - Szeregi czasowe
- $x_split_region_idx$ - informacje regionalne
- y_split_idx - Etykiety

gdzie:

- split - może przyjąć wartości „train”, „val” lub „test”, dla odpowiadających zbiorów
- idx - numer regionu, wartości od 0 do 1 361 dla kolejnych regionów

Wynikiem działania programu jest po 9 plików w formacie .npy dla każdego z 1 362 regionów znajdujących się w tym samym folderze. Następnie 3 pliki zawierające szereg czasowy, informacje regionalne oraz etykiety dla tego samego regionu oraz zbioru zostają zamienione na tensory oraz zapisane do pojedynczego obiektu klasy tf.data.Dataset, który również zapisywany jest do pliku w oddzielnym folderze. W ten sposób zostało otrzymane 4 086 plików zawierających obiekty tf.data.Dataset[15], po 1 dla każdego z 3 zbiorów dla każdego z 1 362 regionów. Ostatecznie pliki z różnych regionów, ale o tych samych zbiorach, są łączone w 3 finalne pliki, po 1 dla każdego ze zbiorów - treningowy, walidacyjny, testowy. Finalnie zbiory zawierają następującą liczbę trójelek (szereg czasowy, informacje regionalne, etykiety) uczących:

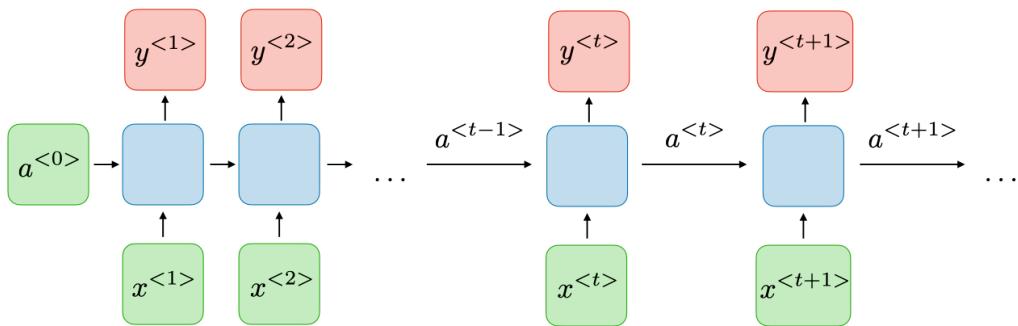
- zbiór treningowy - 3 294 436
- zbiór walidacyjny - 465 661
- zbiór testowy - 101 326

5.2. Modele uczenia maszynowego

5.2.1. Opis modeli

Sieć rekurencyjna

Model sieci rekurencyjnej jest rodzajem sieci neuronowej, która ma zdolność do przetwarzania danych sekwencyjnych lub czasowych poprzez przechowywanie informacji o poprzednich iteracjach. Każda jednostka w sieci rekurencyjnej przyjmuje dane wejściowe oraz stan wewnętrzny, który jest aktualizowany na podstawie wszystkich poprzednich stanów. To pozwala na uwzględnienie kontekstu historycznego i zapamiętanie zależności między elementami sekwencji. Sieci rekurencyjne są szeroko stosowane w przetwarzaniu języka naturalnego, generowaniu tekstu, przetwarzaniu szeregów czasowych i innych zadaniach, gdzie występuje sekwencyjna struktura danych. Na Rysunku 5.12 został przedstawiony uogólniony schemat działania sieci.



Rysunek 5.12: Schemat działania sieci rekurencyjnej. Źródło: Standford.edu

Dla każdego kroku w czasie t , aktywacja $a^{<t>}$ oraz wyjście $y^{<t>}$ są wyrażone w następujący sposób:

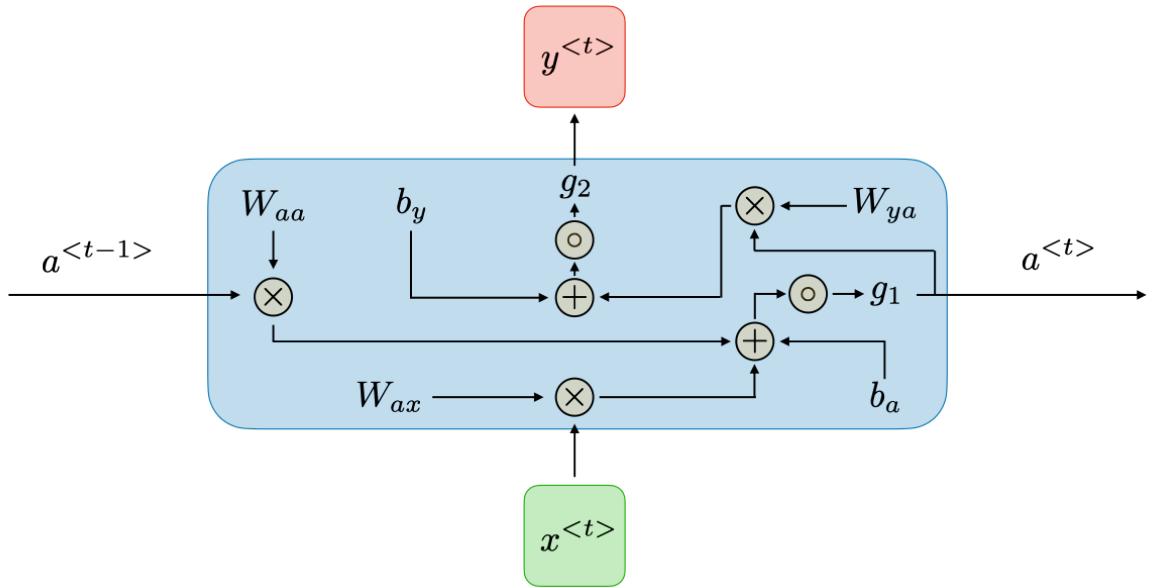
$$a^{<t>} = g_1(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ya} \cdot a^{<t>} + b_y)$$

gdzie W_{ax} , W_{aa} , W_{ya} , b_a , b_y są trenowanymi parametrami, a g_1 oraz g_2 funkcjami aktywacji.

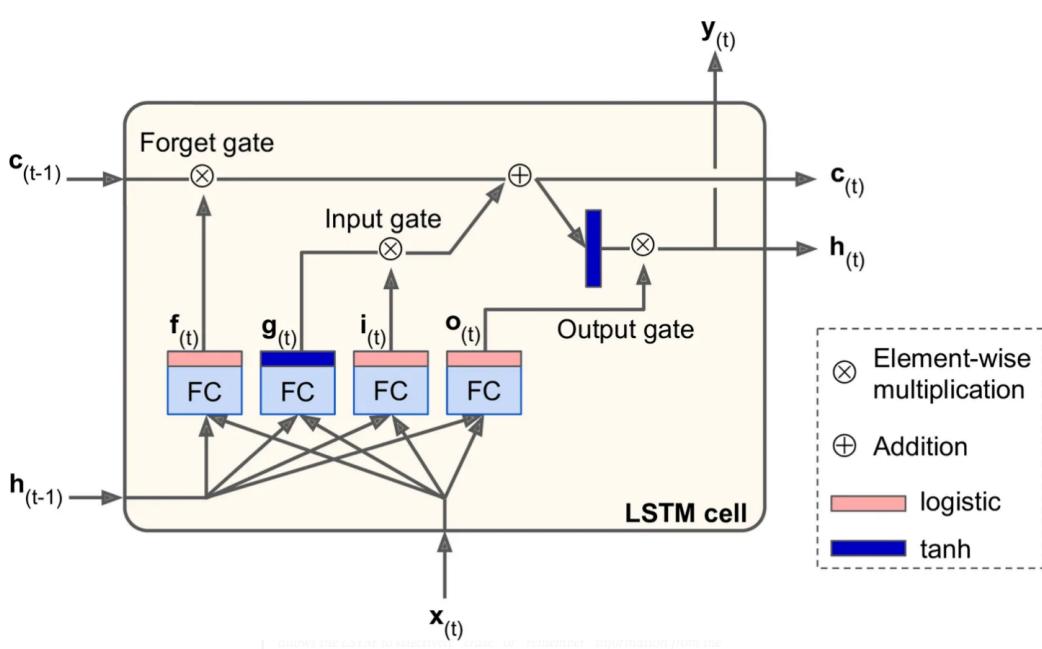
Na Rysunku 5.13 został przedstawiony szczegółowy schemat działania sieci, dla jednego kroku w czasie.

5.2. MODELE UCZENIA MASZYNOWEGO



Rysunek 5.13: Schemat działania sieci rekurencyjnej dla jednego kroku w czasie. Źródło: Stanford.edu

W pracy szczególny nacisk został położony na architekturę LSTM. Jest to architektura sieci neuronowej, zaprojektowana specjalnie do przetwarzania i modelowania długoterminowych zależności w sekwencyjnych danych, eliminując one problemy związane ze znikającym gradientem, które występują w standardowych rekurencyjnych sieciach neuronowych. Na Rysunku 5.14 został przedstawiony schemat działania sieci LSTM dla jednego kroku w czasie.



Rysunek 5.14: Schemat działania sieci LSTM dla jednego kroku w czasie. Źródło: oreilly.com

Opis elementów schematu sieci LSTM[16]:

- Wejście (Input) - W każdym kroku w czasie t sieć przyjmuje wektor wejściowy x_t , który reprezentuje obecną obserwację
- Stan Ukryty (Hidden State) - Sieć przechowuje ukryty wektor stanu h_t , który działa jako "pamięć" sieci.
- Stan Komórki (Cell State) - Sieć przechowuje wektor stanu komórki c_t , którego zadaniem jest przechowywanie informacji długoterminowych.
- Bramki (Gates) - Sieć LSTM wykorzystuje 3 rodzaje bramek, które kontrolują przepływ informacji przez sieć.

Komórka LSTM zawiera trzy bramki[16]:

- Bramka Zapominania (Forget Gate) - Decyduje, które informacje należy zapomnieć z poprzedniego stanu komórki, biorąc pod uwagę aktualne wejście. Bramka zapomnienia przyjmuje wcześniejszy wektor stanu h_{t-1} oraz obecne wejście x_t , a zwraca wektor liczb od 0 do 1. Wektor liczb mówi ile wcześniejszego stanu komórki "zapomnieć", a ile pozostawić. Bramka zapomnienia wyraża się wzorem:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

gdzie:

- σ - funkcja aktywacji sigmoid
- W_f - macierz wag bramki zapomnienia
- $[h_{t-1}, x_t]$ - konkatenacja wcześniejszego stanu ukrytego oraz obecnego wejścia
- b_f - wektor biasu bramki zapomnienia
- f_t - wektor bramki zapomnienia dla obecnego kroku w czasie

- Bramka Wejścia (Input Gate) - Decyduje, które nowe informacje należy dodać do stanu komórki. Bramka wejścia przyjmuje wcześniejszy stan ukryty h_{t-1} oraz obecne wejście x_t , a zwraca wektor liczb od 0 do 1. Wektor liczb mówi ile obecnego wejścia dodać do stanu komórki. Bramka wejścia wyraża się wzorami:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

5.2. MODELE UCZENIA MASZYNOWEGO

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \circ c_{t_1} + i_t \circ \tilde{c}_t$$

gdzie:

- σ - funkcja aktywacji sigmoid
 - W_i, W_c - macierz wag bramki wejścia
 - $[h_{t-1}, x_t]$ - konkatenacja wcześniejszego stanu ukrytego oraz obecnego wejścia
 - b_i, b_c - wektory biasu bramki wejścia
 - i_t - wektor bramki wejścia dla obecnego kroku w czasie
 - \tilde{c}_t - wektor kandydat dla stanu komórki dla obecnego kroku w czasie
 - \circ - mnożenie Hadamarda
- Bramka Wyjścia (Output Gate) - Decyduje, które informacje należy przekazać do wyjścia, na podstawie aktualnego stanu komórki. Bramka wyjścia przyjmuje wcześniejszy wektor stanu h_{t-1} oraz obecny stan komórki c_t , a zwraca wektor liczb od 0 do 1. Wektor liczb mówi ile obecnego stanu komórki zwrócić jako obecny stan ukryty h_t . Bramka wyjścia wyraża się wzorem:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(c_t)$$

gdzie:

- σ - funkcja aktywacji sigmoid
- W_o - macierz wag bramki wyjścia
- $[h_{t-1}, x_t]$ - konkatenacja wcześniejszego stanu ukrytego oraz obecnego wejścia
- b_o - wektory biasu bramki wyjścia
- o_t - wektor bramki wyjścia dla obecnego kroku w czasie
- h_t - wektor stanu ukrytego dla obecnego kroku w czasie

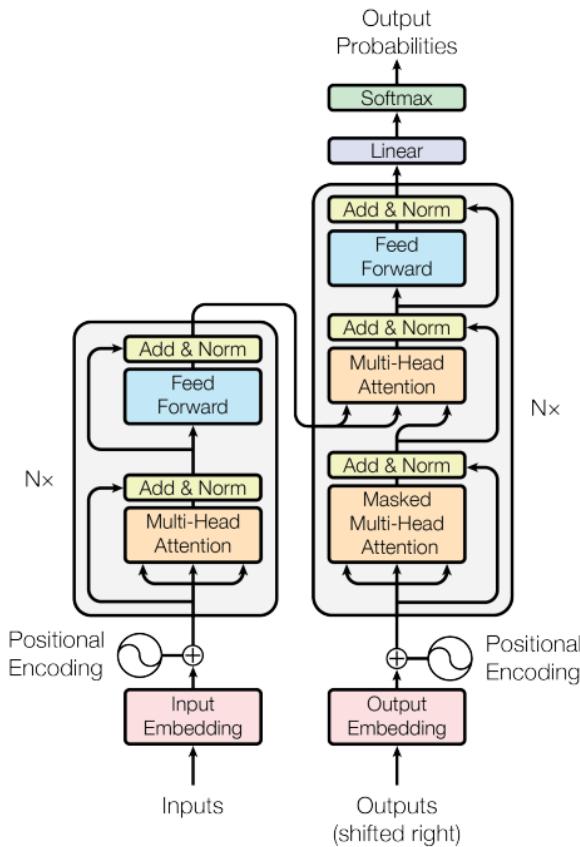
Transformer

Transformer to innowacyjny model sieci neuronowej[4], który zrewolucjonizował dziedzinę przetwarzania języka naturalnego. Zamiast polegać na rekurencyjnych lub konwolucyjnych warstwach, jak w tradycyjnych architekturach, Transformer opiera się na mechanizmie atencji do

skupienia się na różnych częściach sekwencji wejściowej, bezpośrednio przetwarzając całe sekwencje naraz. Głównymi komponentami modelu są:

- Koder - Przetwarza kontekst na podstawie którego Dekoder generuje sekwencję. Składa się z ustalonej liczby bloków, a każdy blok składa się z dwóch elementów - mechanizmu self-attention oraz 2-warstwowego MLP.
- Dekoder - Przetwarza sekwencję oraz generuje następne jej elementy na podstawie informacji z Kodera. Składa się z ustalonej liczby bloków, a każdy blok składa się z trzech elementów - mechanizmu self-attention, mechanizmu cross-attention oraz 2-warstwowego MLP.
- Pozycyjne kodowanie - Dodaje informacje o pozycji, danego wektora w całej sekwencji, do wszystkich wektorów wejściowych

W pracy wprowadzającej transformer[4] został on wykorzystany do translacji maszynowej. W takiej implementacji Koder odpowiadał za analizowanie podanej przez użytkownika sentencji. Dekoder natomiast autoregresyjnie generował następny fragment tekstu na podstawie informacji przekazanych przez Koder oraz już wygenerowanej części tłumaczenia. Na Rysunku 5.15 została przedstawiona oryginalna architektura transformera, zgodna z powyższym opisem.



Rysunek 5.15: Architektura transformera. Źródło: [4]

5.2. MODELE UCZENIA MASZYNOWEGO

Transformer rozpoczyna działanie od pozycyjnego kodowania wejściowych danych sekwencyjnych. Do zwektoryzowanej reprezentacji każdego kroku w czasie zostaje dodany wektor, o tych samych wymiarach, który zawiera informacje o położeniu danego kroku w całej sekwencji. Jest to niezbędna operacja, ponieważ transformer nie zawiera rekurencji lub konwolucji. Dodatkowo nowo-wprowadzony mechanizm atencji działa na zbiorach.

Następnie pozycyjne zakodowane dane zostają przekazane do mechanizmu atencji. Mechanizm ten odpowiada za wymianę informacji między obserwacjami w tej samej sekwencji. Dodatkowo zwektoryzowana reprezentacja każdego kroku zostaje liniowo zrzutowana na ustaloną liczbę krótkich wektorów. Mechanizm atencji zostaje wykorzystany dla każdego wektora oddzielnie. Dzięki temu model ma możliwość interpretacji elementów sekwencji w różnych kontekstach.

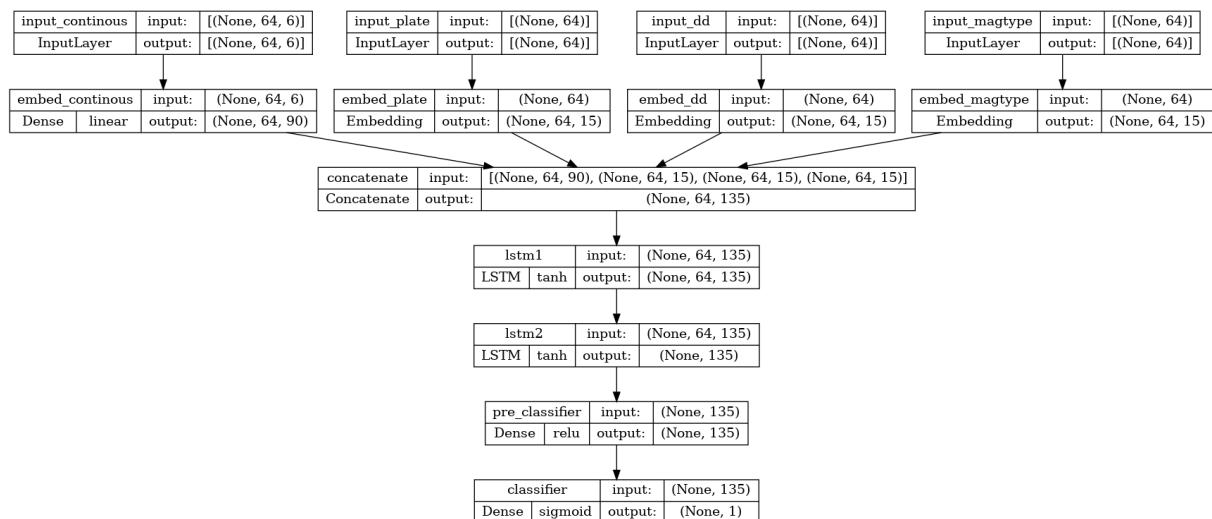
Następnie sekwencja przetworzona przez mechanizm atencji zostaje przekazana do 2-warstwowego MLP. To właśnie tam odbywa się większość obliczeń oraz proces ekstrakcji cech. Powyżej opisany proces analizy danych sekwencyjnych zostaje wykonany ustaloną liczbę razy. Dzięki temu model zaczyna od najbardziej ogólnych cech, a kończy na najbardziej szczegółowych.

5.2.2. Architektury modeli

Wszystkie modele zostały zaimplementowane oraz trenowane przy pomocy biblioteki Tensorflow[15]. Zostało rozważone 5 architektur modeli wykorzystywanych do klasyfikacji danych sekwencyjnych, w tym szeregow czasowych.

Sieć rekurencyjna

Pierwszym testowanym modelem była rekurencyjna sieć neuronowa. Architektura sieci została przedstawiona na Rysunku 5.16.

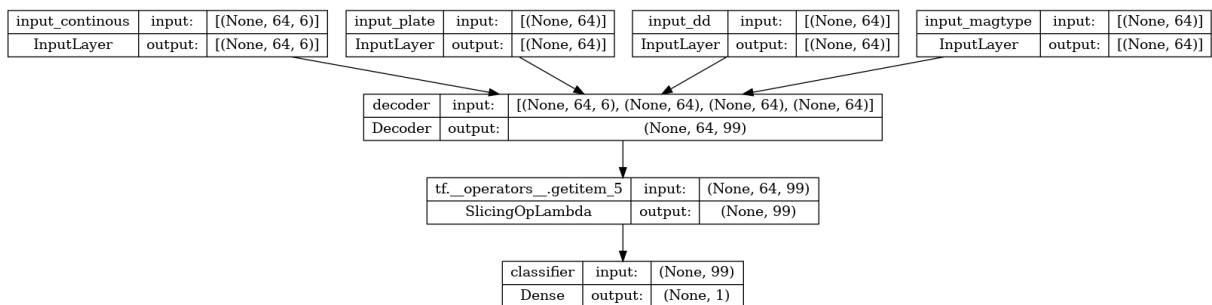


Rysunek 5.16: Architektura sieci rekurencyjnej

Na początku poszczególne zmienne z szeregu czasowego zostają zwektoryzowane. Na tym etapie należy rozdzielić zmienne dyskretne od ciągłych, ponieważ są one wektoryzowane w inny sposób. Wszystkie 6 zmiennych ciągłych jest zawarte w tensorze o wymiarach (*None, block_size, 6*), gdzie *None* to wymiar batch. Zmienne ciągłe są zanurzone do przestrzeni o wymiarze $\frac{n_units}{9} \cdot 6$ za pomocą pojedynczej warstwy tf.keras.layers.Dense[15]. Każda zmienna dyskretna jest z kolei traktowana oddzielnie. Poszczególne klasy każdej ze zmiennych są zanurzane do przestrzeni o wymiarze $\frac{n_units}{9}$ za pomocą warstwy tf.keras.layers.Embedding[15]. Ostatecznie cały szereg czasowy jest konkatenowany do tensora o wymiarze (*None, block_size, n_units*). Tak zwektoryzowany szereg czasowy zostaje przepuszczony przez 2 warstwy sieci rekurencyjnej LSTM[17] o *n_units* jednostkach. Ostatecznie wyjściowy tensor o wymiarach (*None, n_units*) zostaje przepuszczony przez 2-warstwowy MLP o *n_units* oraz 1 neuronach. Ostatnia warstwa składa się z pojedynczego neuronu z funkcją aktywacji sigmoid. Taka architektura warstwy wyjściowej gwarantuje, że model, dla każdego szeregu czasowego, będzie zwracał 1 liczbę reprezentującą prawdopodobieństwo wystąpienia etykiety 1 (trzęsienia o skali większej niż 5, w przeciągu miesiąca od ostatniego trzęsienia w danym szeregu czasowym). W tak zaprojektowanym modelu jedna zmienna, *n_units*, odpowiada za liczbę parametrów w modelu. Należy podkreślić, że zmienna *n_units* musi być podzielna przez 9. W testowanym modelu zostało ustalone *n_units* = 135, model posiada 313 366 parametrów.

Transformer - bez informacji regionalnych - mały

Wszystkie następne modele bazowały na transformerze z pracy[4] Vaswaniego. Następną testowaną architekturą był transformer dekoder . Na Rysunku 5.17 została przedstawiona generalna architektura.

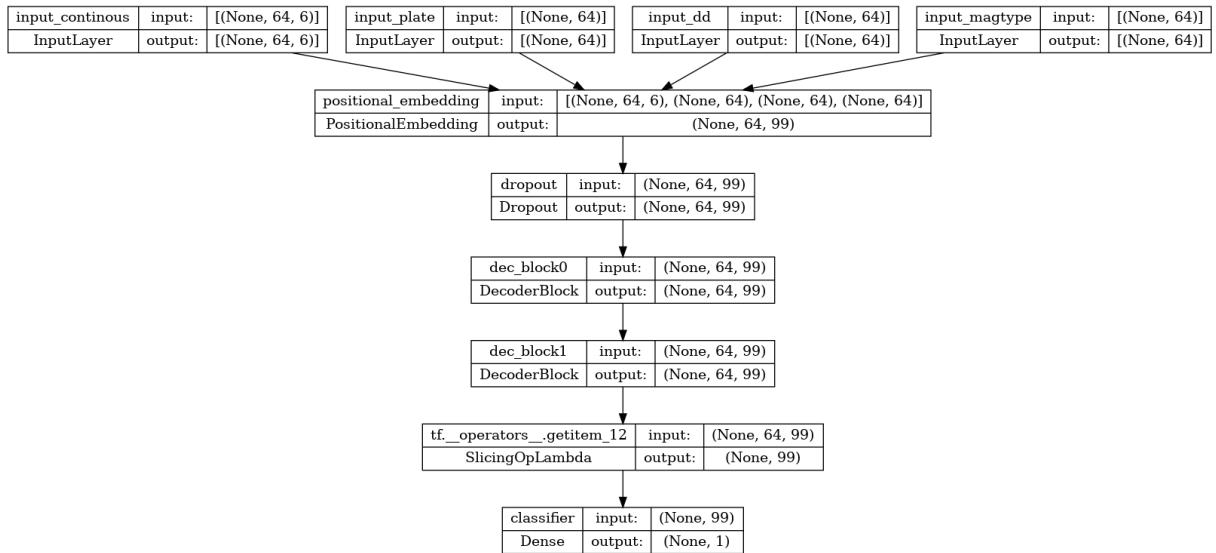


Rysunek 5.17: Architektura transformera

Podobnie jak w sieci rekurencyjnej, szereg czasowy zostaje rozdzielony na zmienne ciągłe i dyskretne. Następnie podzielony szereg czasowy zostaje przekazany do dekodera. Właśnie tam odbywa się większość obliczeń. Dekoder zwraca zwektoryzowany oraz przetworzony szereg cza-

5.2. MODELE UCZENIA MASZYNOWEGO

sowy o wymiarach ($None$, $block_size$, d_{model}). Następnie, z przetworzonego szeregu czasowego zostaje wybrany ostatni element, otrzymując w ten sposób tensor o wymiarach ($None$, d_{model}). Ostatecznie tensor zostaje przetworzony przez pojedynczą warstwę `tf.keras.layers.Dense` o 1 neuronie, z funkcją aktywacji sigmoid. Podobnie jak w sieci neuronowej, model zwraca 1 liczbę dla każdego szeregu czasowego, reprezentującą prawdopodobieństwo wystąpienia etykiety 1. Na Rysunku 5.18 została przedstawiona architektura modelu z rozszerzonym widokiem dekodera.



Rysunek 5.18: Architektura transformera z rozszerzonym widokiem dekodera

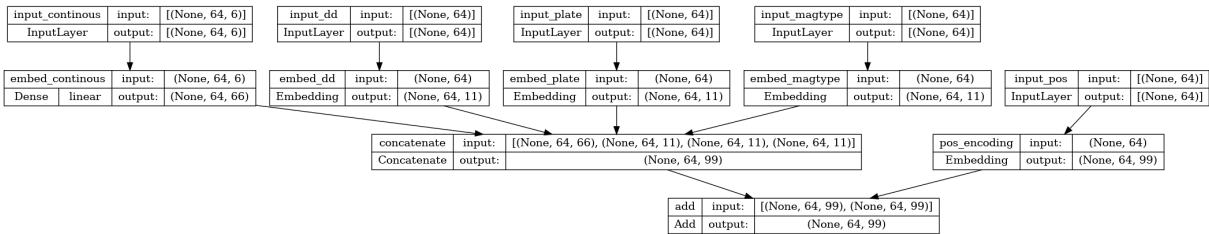
Dekoder składa się z 2 głównych elementów:

- PositionalEmbedding - pozycyjne zanurzanie szeregu czasowego w przestrzeń o wymiarze d_{model} - Rysunek 5.19
- DecoderBlock - Główna część transformera, tam odbywa się większość obliczeń - Rysunek 5.20

Należy podkreślić, że liczba instancji DecoderBlock jest jednym z parametrów modelu. W testowanym modelu zostało ustalone $num_layers = 2$. Dodatkowo, pomiędzy warstwą PositionalEmbedding a pierwszym DecoderBlock znajduje się warstwa `tf.keras.layers.Dropout`, której zadaniem jest zapobieganie przetrenowaniu modelu.

Na Rysunku 5.19 została przedstawiona architektura warstwy PositionalEmbedding.

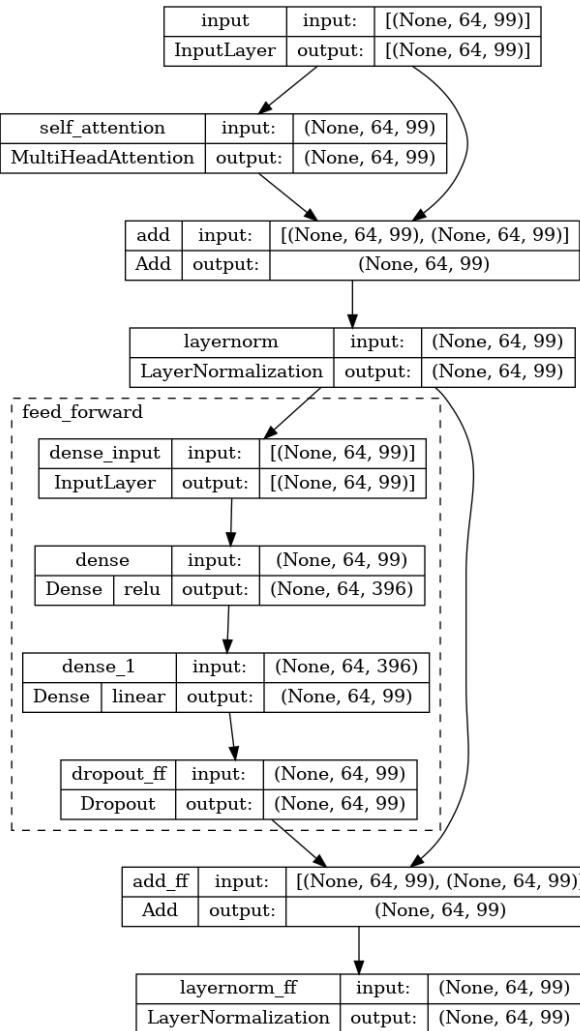
5. METODOLOGIA



Rysunek 5.19: Architektura pozycyjnej wektoryzacji trzęsień

Architektura warstwy PositionalEmbedding jest zbliżona do początkowych warstw sieci rekurencyjnej. Szeregi czasowe są zanurzane w przestrzeń o wymiarze d_{model} . Następnie do każdego trzęsienia w szeregu czasowym zostaje dodany wektor o wymiarze $(1, d_{model})$ reprezentujący pozycję danego trzęsienia w szeregu czasowym. Ostatecznie tensor o wymiarach $(None, block_size, d_{model})$ zostaje przekazany do następnych warstw transformera.

Na Rysunku 5.20 została przedstawiona architektura pojedynczego bloku dekodera.



Rysunek 5.20: Architektura pojedynczego bloku w dekoderze

5.2. MODELE UCZENIA MASZYNOWEGO

Na wejściu warstwa DecoderBlock otrzymuje pozycyjnie zanurzony szereg czasowy w przestrzeni o d_{model} wymiarach. Na początku szereg czasowy zostaje przepuszczony przez warstwę tf.keras.layers.MultiHeadAttention[15]. Zadaniem tej warstwy jest wymiana informacji między trzęsieniami w szeregu czasowym. Dodatkowo, wyjściowy tensor został połączony z wejściowym za pomocą połączenia rezydualnego. Następnie tensor zostaje znormalizowany przy pomocy tf.keras.layers.LayerNorm[15]. W kolejnym kroku tensor o wymiarach $(None, block_size, d_{model})$ zostaje przepuszczony przez 2-warstwowy MLP o d_{ff} oraz d_{model} neuronach. Ponownie, wektor wyjściowy z MLP jest połączony z wejściowym za pomocą połączenia rezydualnego. Ostatecznie tensor wyjściowy zostaje znormalizowany. Należy podkreślić, że wymiar wektora wejściowego do warstwy DecoderBlock oraz wyjściowego jest taki sam. Taka zależność pozwala ułożyć wiele takich bloków jeden po drugim. Należy również podkreślić, że połączenia rezydualne oraz normalizacja nie są konieczne do działania modelu, a jedynie poprawiają proces trenowania modelu.

Architektura opisanego modelu posiada 5 parametrów określających wielkość modelu:

- d_{model} - wymiar przestrzeni zanurzania szeregów czasowych, musi być podzielne przez 9,
 $d_{model} = 99$
- d_{ff} - liczba neuronów w pierwszej warstwie MLP w DecoderBlock, $d_{ff} = 4 \cdot d_{model}$ [4]
- num_layers - liczba warstw DecoderBlock w dekoderze, $num_layers = 2$
- num_heads - liczba głów w warstwie tf.keras.layers.MultiHeadAttention, $num_heads = 2$
- $dropout$ - prawdopodobieństwo ustawienia danej liczby na 0, $dropout = 0.1$ [4]

Model o takich wymiarach posiada 324 842 parametrów.

Transformer - bez informacji regionalnych - duży

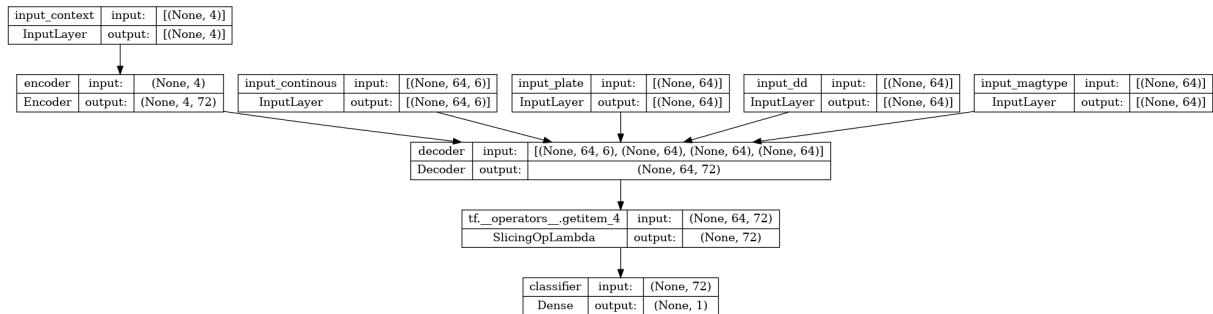
Architektura tego modelu jest identyczna jak poprzedniego, zmieniają się jedynie parametry. W tym modelu:

- $d_{model} = 180$
- $d_{ff} = 4 \cdot d_{model}$
- $num_layers = 2$
- $num_heads = 2$
- $dropout = 0.1$

Model o takich wymiarach posiada 1 057 181 parametrów.

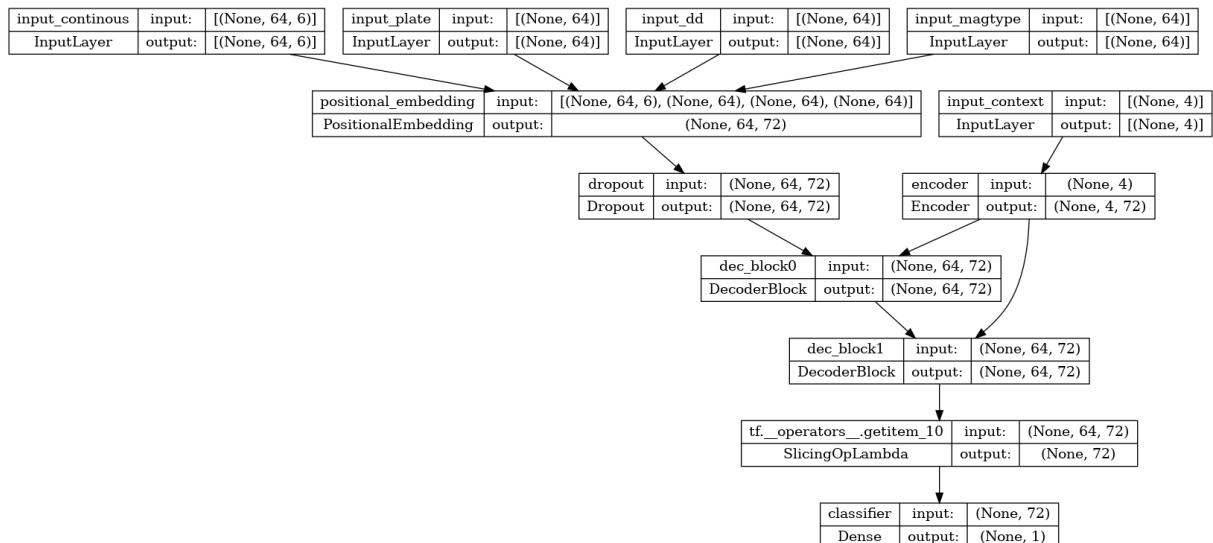
Transformer - z informacjami regionalnymi - mały

Ostatnią testowaną architekturą był transformer koder-dekoder. Na Rysunku 5.21 została przedstawiona generalna architektura.



Rysunek 5.21: Architektura transformera

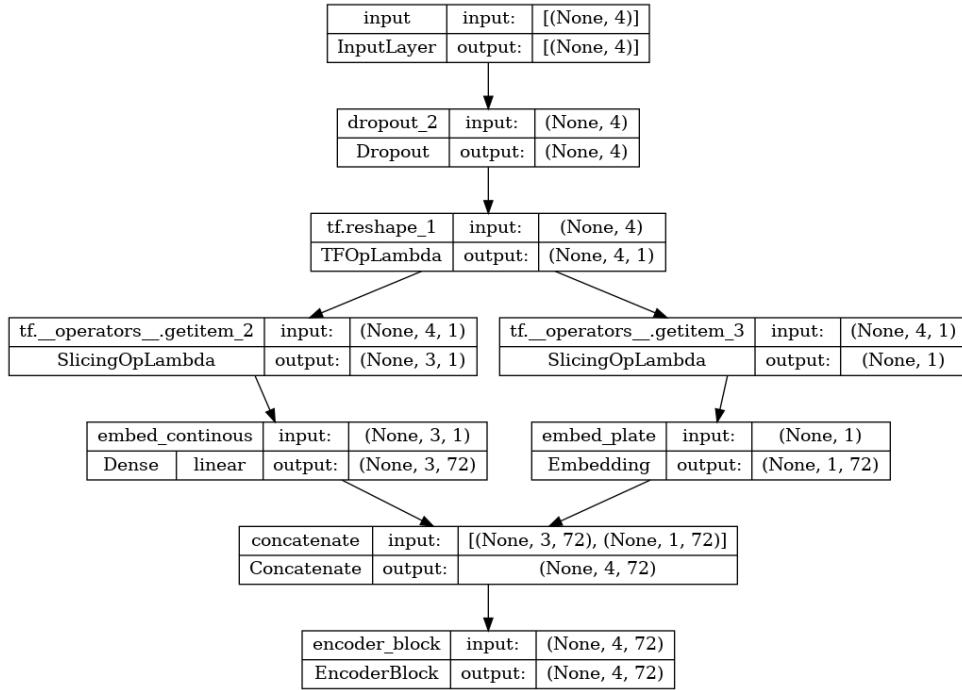
W opisywanym modelu szeregi czasowe będą obsługiwane w ten sam sposób co w transformerze dekoderze. Dodatkowo, jako dane wejściowe, zostały zawarte informacje regionalne. Są one przetwarzane przez koder, a następnie dodawane do dekodera za pomocą cross atencji. Na Rysunku 5.22 została przedstawiona dokładniejsza architektura modelu.



Rysunek 5.22: Architektura transformera z rozszerzonym widokiem dekodera

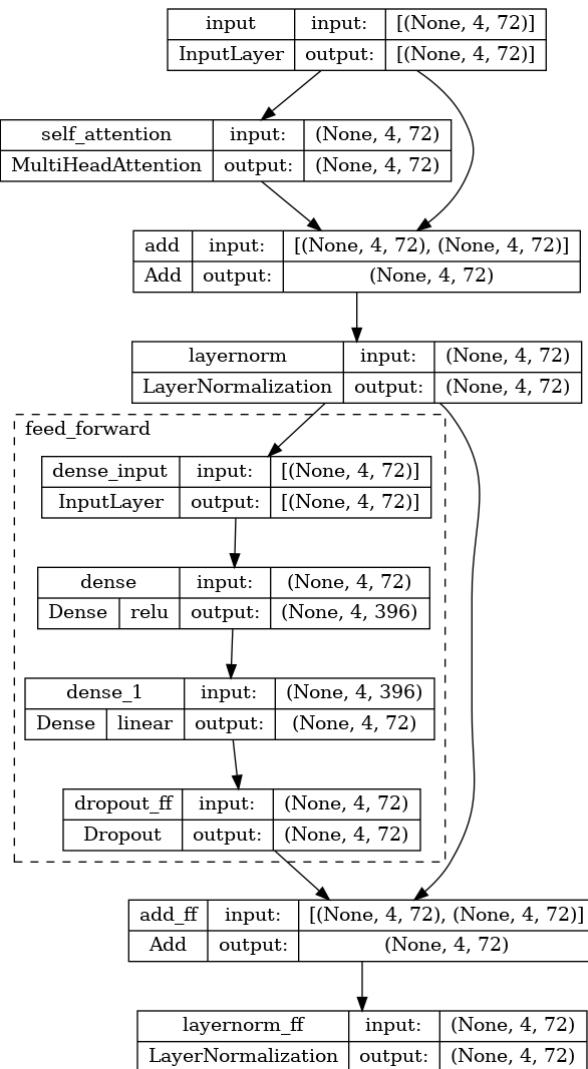
Nowym elementem tego modelu jest koder(encoder), który jest wykorzystywany do przetwarzania informacji regionalnych. Następnie informacje te są wykorzystywane w każdej instancji warstwy DecoderBlock. Na Rysunku 5.23 została przedstawiona architektura kodera.

5.2. MODELE UCZENIA MASZYNOWEGO



Rysunek 5.23: Architektura kodera

Na początku informacje regionalne są przekształcane do tensora o wymiarach (*None*, 4, 1), a następnie zmienne ciągłe są oddzielane od dyskretnej. Zmienne ciągłe są zanurzane do przestrzeni d_{model} wymiarowej za pomocą pojedynczej warstwy `tf.keras.layers.Dense`. Zmienna dyskretna jest zanurzona do przestrzeni d_{model} wymiarowej za pomocą warstwy `tf.keras.layers.Embedding`. Ostateczne reprezentacje wektorowe zmiennych są konkatenowane w jeden tensor o wymiarach (*None*, 4, d_{model}) i przekazane do warstwy `EncoderBlock`. Na Rysunku 5.24 została przedstawiona architektura warstwy.

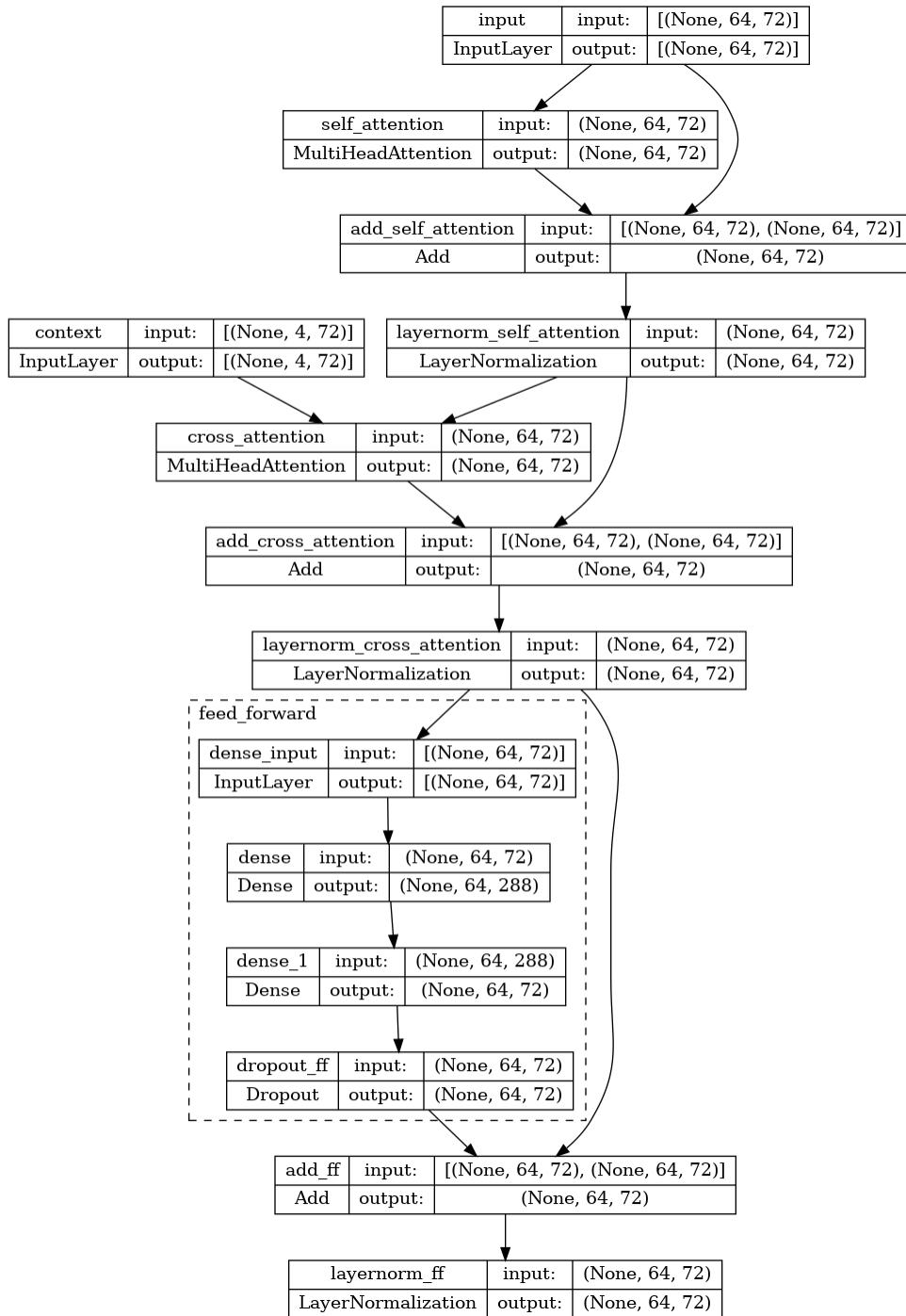


Rysunek 5.24: Architektura pojedynczego bloku w koderze

Architektura warstwy EncoderBlock jest taka sama, jak warstwy DecoderBlock z Rysunku 5.20, różni się jedynie parametrem d_{model} . Należy podkreślić, że liczba instancji EncoderBlock w koderze jest jednym z parametrów opisywanej architektury modelu. Została ona ustalona na $num_enc_layers = 1$.

Na Rysunku 5.25 została przedstawiona architektura warstwy DecoderBlock.

5.2. MODELE UCZENIA MASZYNOWEGO



Rysunek 5.25: Architektura pojedynczego bloku w dekoderze

Architektura warstwy DecoderBlock jest zbliżona do warstwy EncoderBlock z Rysunku 5.24. Została dodana warstwa `tf.keras.layers.MultiHeadAttention` wykorzystywana do dodania przetworzonych przez koder informacji regionalnych do szeregu czasowego.

Architektura opisanego modelu posiada 6 parametrów określających wielkość modelu:

- d_{model} - wymiar przestrzeni zanurzania szeregów czasowych, musi być podzielne przez 9,
 $d_{model} = 72$

- d_{ff} - liczba neuronów w pierwszej warstwie MLP w DecoderBlock, $d_{ff} = 4 \cdot d_{model}$
- num_dec_layers - liczba warstw DecoderBlock w dekoderze, $num_dec_layers = 2$
- num_enc_layers - liczba warstw EncoderBlock w koderze, $num_enc_layers = 1$
- num_heads - liczba głów w warstwie MultiHeadAttention, $num_heads = 2$
- $dropout$ - prawdopodobieństwo ustawienia danej liczby na 0, $dropout = 0.1$

Model o takich wymiarach posiada 347 129 parametrów.

Transformer - z informacjami regionalnymi - duży

Architektura tego modelu jest identyczna jak poprzedniego, zmieniają się jedynie parametry.

W tym modelu:

- $d_{model} = 126$
- $d_{ff} = 4 \cdot d_{model}$
- $num_dec_layers = 2$
- $num_enc_layers = 1$
- $num_heads = 2$
- $dropout = 0.1$

Model o takich wymiarach posiada 1 042 861 parametrów.

W Tabeli 5.7 zostały zawarte nazwy wszystkich modeli, wraz z ich liczbą parametrów. Nazwy modeli uwzględniających informacje regionalne kończą się sufiksem `_enc_dec`, natomiast modele bez informacji regionalnych `_dec`.

Nazwa modelu	Liczba parametrów
LSTM	313 366
Transformer _dec _small	324 842
Transformer _dec _large	1 057 181
Transformer _enc _dec _small	347 129
Transformer _enc _dec _large	1 042 861

Tabela 5.7: Tabela z liczbą parametrów dla wszystkich modeli

5.2. MODELE UCZENIA MASZYNOWEGO

5.2.3. Trenowanie

Wszystkie modele były trenowane z $batch_size = 1024$. Modele należy podzielić na 2 grupy pod względem trenowania, podział został przedstawiony w tabeli 5.8.

Grupa	Model
Pierwsza	LSTM
	Transformer_dec_small
	Transformer_enc_dec_small
Druga	Transformer_dec_large
	Transformer_enc_dec_large

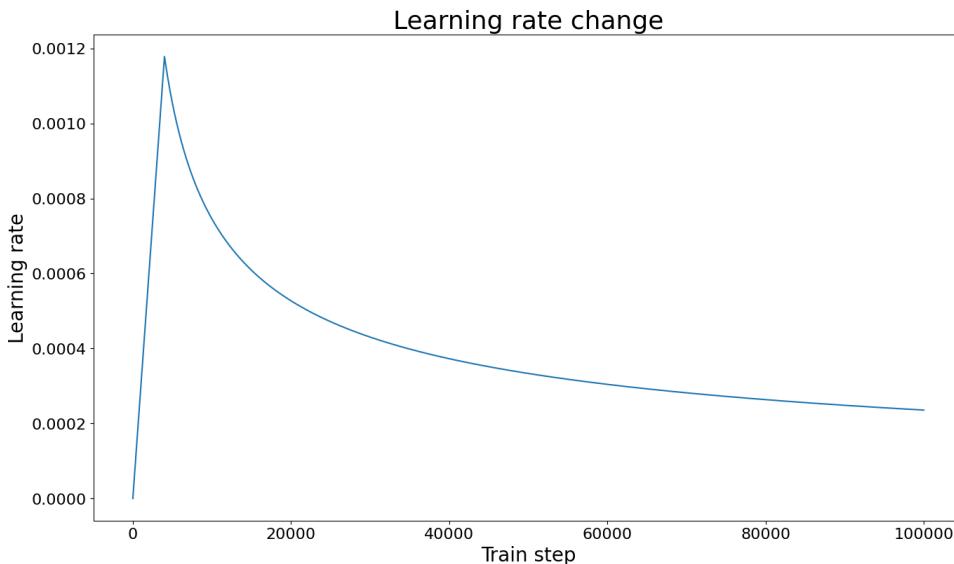
Tabela 5.8: Podział modeli pod względem trenowania

Modele były trenowane zgodnie z parametrami opisonymi w Tabeli 5.9.

Parametr	Pierwsza grupa	Druga grupa
Liczba epok	20	30
Funkcja straty	tf.keras.losses.BinaryCrossentropy[15]	tf.keras.losses.BinaryCrossentropy
Learning rate	0.001	$d_{model}^{-0.5} \cdot min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5})$, $warmup_steps = 4000$ [4] - Rysunek 5.26
Optimizer	Adam	Adam($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 1^{-9}$)[15][4]

Tabela 5.9: Trenowanie modeli

Dodatkowo wagi modeli w tej grupie były zapisywane co epokę. Na Rysunku 5.26 została przedstawiona zmiana parametru learning rate podczas trenowania.



Rysunek 5.26: Zmiana learning rate podczas trenowania

Należy podkreślić, że każdy model był trenowany 2 razy. Za pierwszym razem w sposób opisany powyżej, natomiast za drugim razem został dodany parametr *class_weight*. Zadaniem tego parametru jest zniwelowanie wpływu niezbalansowania klas na proces uczenia. Dzieje się tak poprzez skalowanie funkcji straty przez odpowiednie stałe, dla każdej z klas. Stałe zostały obliczone w następujący sposób:

Zostało ustalone:

- $neg = 3\,094\,591$ - liczba obserwacji, w zbiorze treningowym, z *label = 0*
- $pos = 199\,845$ - liczba obserwacji, w zbiorze treningowym, z *label = 1*
- $total = 3\,294\,436$ - liczba wszystkich obserwacji w zbiorze treningowym

Na podstawie powyższych stałych[18]:

- $weight_for_0 = \frac{1}{neg} \cdot \frac{total}{2} = 0.5323$
- $weight_for_1 = \frac{1}{pos} \cdot \frac{total}{2} = 8.2425$

Podczas trenowania funkcja straty dla obserwacji z etykietą 0 jest mnożona przez *weight_for_0*, natomiast obserwacje z etykietą 1 przez *weight_for_1*. Taki zabieg powoduje, że model będzie przykładał więcej uwagi do obserwacji z mniejszej klasy, niwelując wpływ niezbalansowania klas.

5.3. Ewaluacja modeli

Do wyboru najlepszego modelu zostały wykorzystane metryki opisane w Tabeli 5.10, policzone dla zbiorów testowego i walidacyjnego .

Metryka	Opis
Loss	Funkcja straty (crossentropy)
Accuracy	Część poprawnie sklasyfikowanych obserwacji
Precision	Stosunek poprawnie wykrytych trzęsień do wszystkich rekordów zakwalifikowanych jako trzęsienie
Recall	Stosunek poprawnie wykrytych trzęsień do wszystkich trzęsień
F1	Średnia harmoniczna metryk precision i recall

Tabela 5.10: Metryki

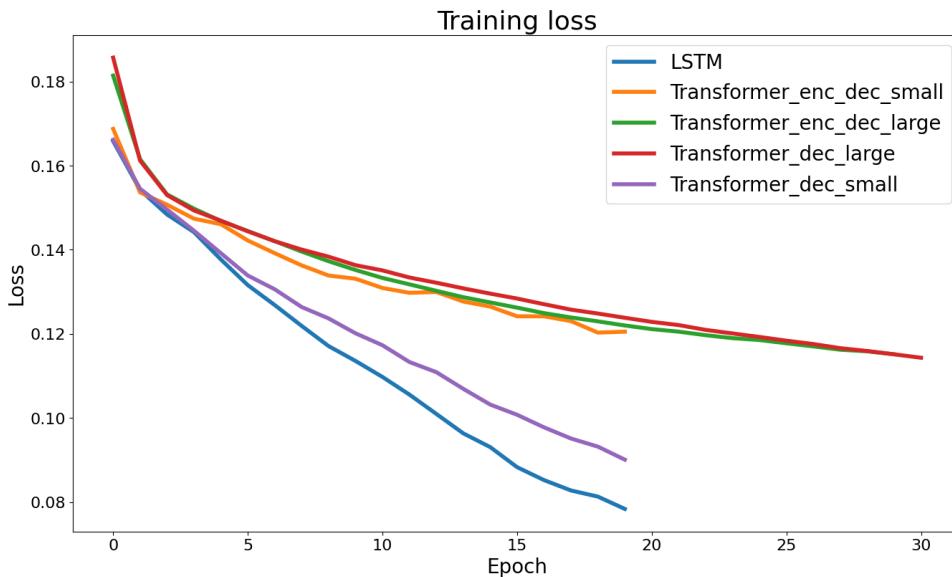
Należy podkreślić, że nie należy porównywać wartości funkcji straty na zbiorze treningowym dla modelów trenowanych z parametrem *class_weights* z funkcją straty dla modelów trenowanych bez tego parametru. Natomiast podczas ewaluacji modelu na zbiorze walidacyjnym oraz testowym, funkcja straty nie jest skalowana, a więc jej wartości są porównywalne w obu przypadkach modelów. Pozostałe metryki są porównywalne niezależnie od zbioru.

5.3.1. Modele trenowane bez parametru *class_weights*

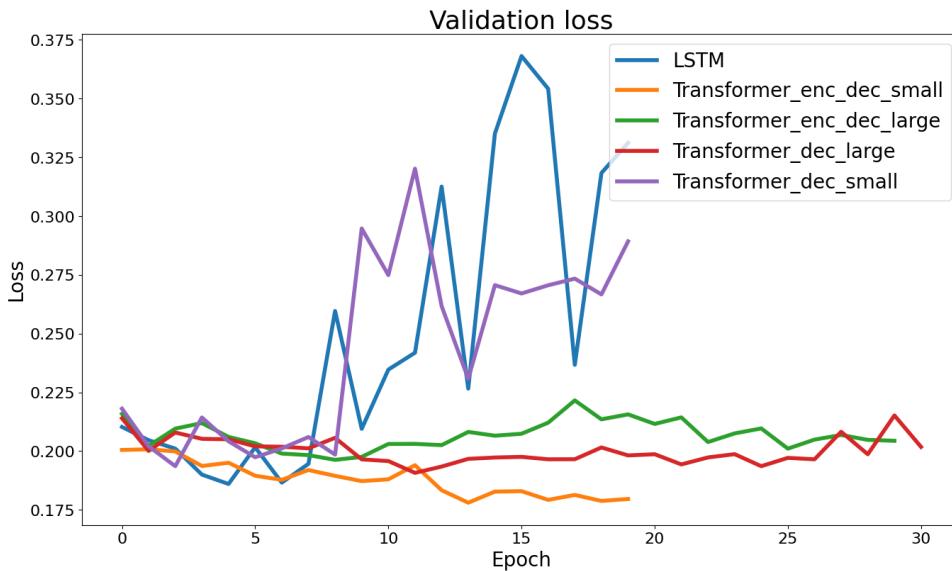
Znaczenie oraz wpływ parametru *class_weights* na proces trenowania modeli zostało wyjaśnione w 5.2.3.

Loss

Na Rysunku 5.27 została przedstawiona zmiana funkcji straty podczas uczenia. Na podstawie Rysunku 5.27b można stwierdzić, że w początkowej fazie uczenia funkcja straty maleje. Po około 8 epokach modele zaczynają się przetrenowywać. Ten efekt jest najbardziej widoczny w Modelu LSTM, który nie posiada warstwy Dropout, której zadaniem jest zapobieganie przetrenowywaniu.



(a) Loss na zbiorze treningowym



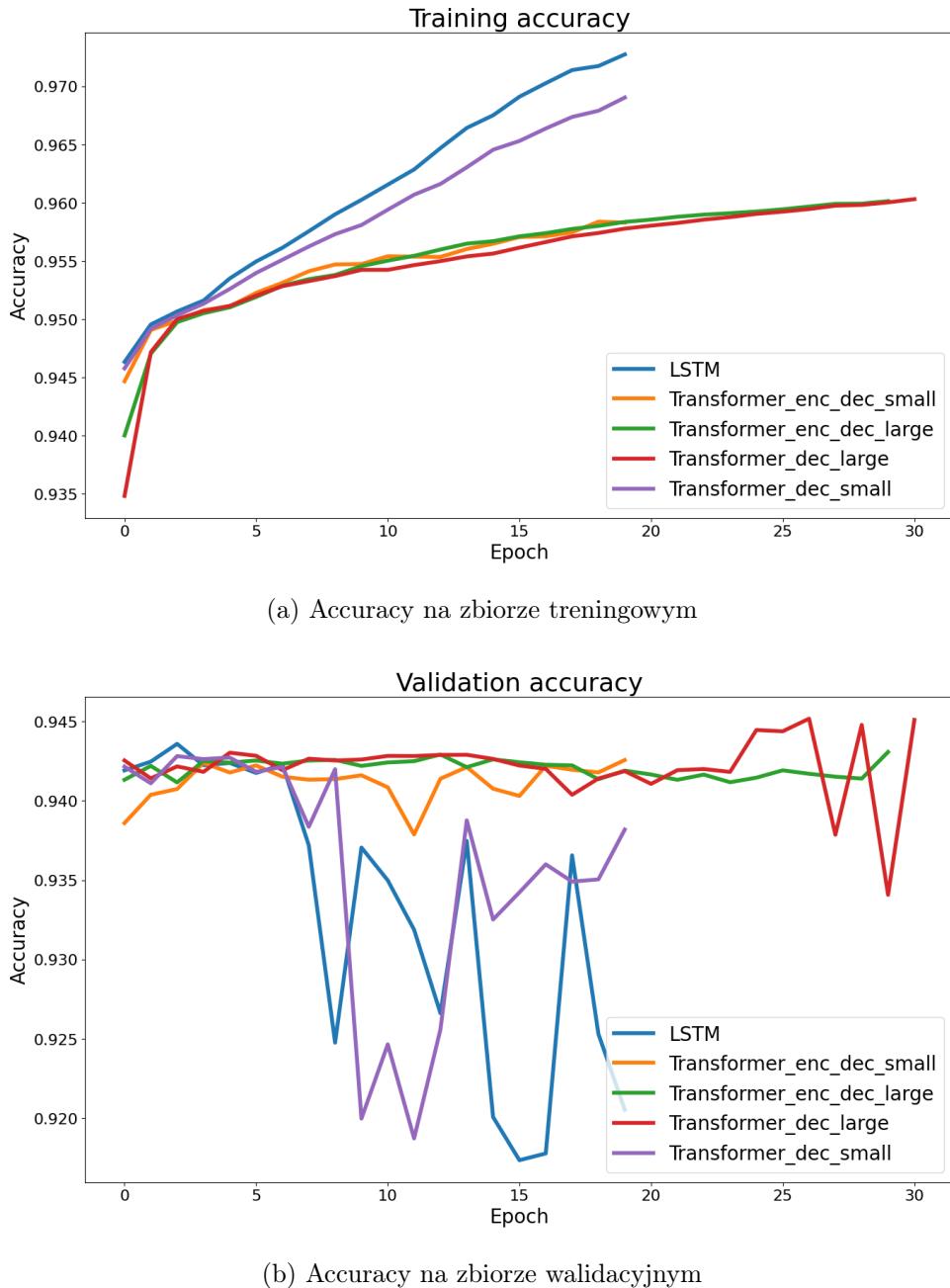
(b) Loss na zbiorze walidacyjnym

Rysunek 5.27: Zmiana loss podczas trenowania

Accuracy

Na podstawie Rysunków 5.27 i 5.28 można stwierdzić, że wykresy funkcji straty oraz accuracy są ze sobą mocno powiązane. Gdy funkcja straty maleje, accuracy rośnie w podobnym stopniu. Należy podkreślić, że bardzo wysoka wartość accuracy jest spowodowana niezbalansowanymi klasami. Z tego powodu do testowania modelów zostały również wykorzystane metryki Precision, Recall i F1, które są mniej wrażliwe na niezbalansowanie klas.

5.3. EWALUACJA MODELI

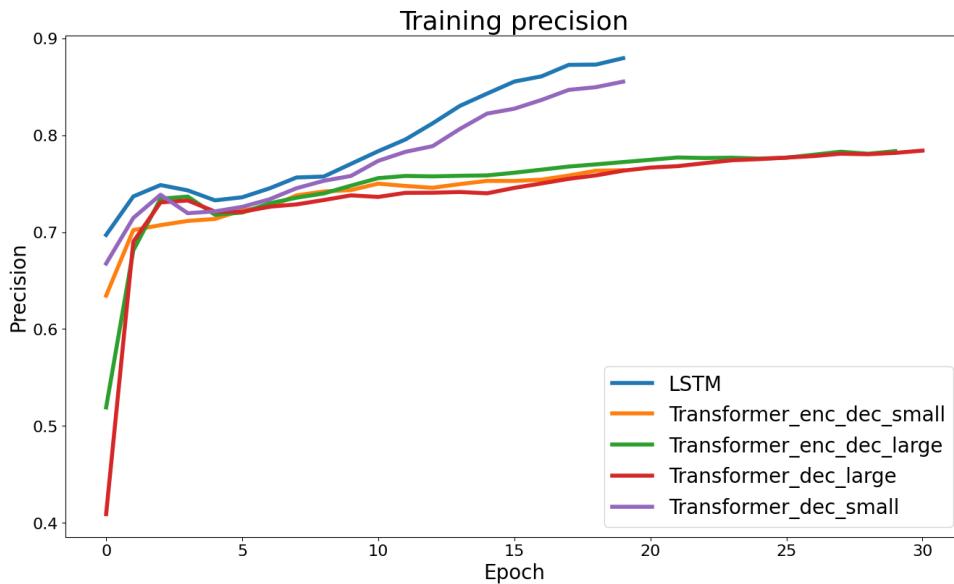


Rysunek 5.28: Zmiana accuracy podczas trenowania

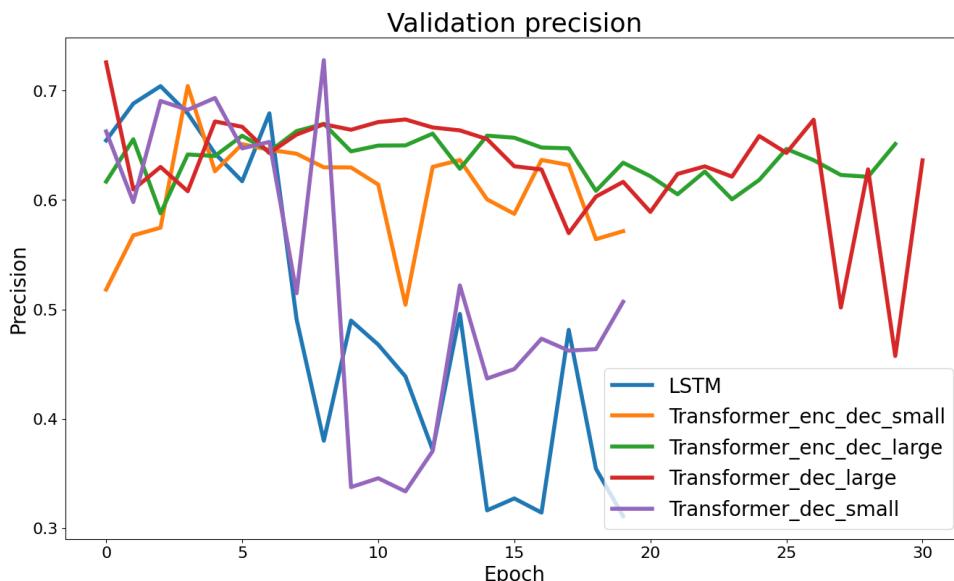
Precision

Następną badaną metryką jest precision. Jest to szczególnie ważna metryka, ponieważ przedstawiona ona stosunek liczby obserwacji true positive do sumy true positive i false positive. Oznacza to, że im większa wartość metryki, tym mniejsze prawdopodobieństwo wystąpienia false positive, a co za tym idzie, mniejsze prawdopodobieństwo wywołania paniki wśród mieszkańców danego regionu, spowodowane fałszywym ostrzeżeniem o nadchodzącym trzęsieniu ziemi.

Na Rysunku 5.29 została przedstawiona zmiana precision podczas trenowania modelów.



(a) Precision na zbiorze treningowym



(b) Precision na zbiorze walidacyjnym

Rysunek 5.29: Zmiana precision podczas trenowania

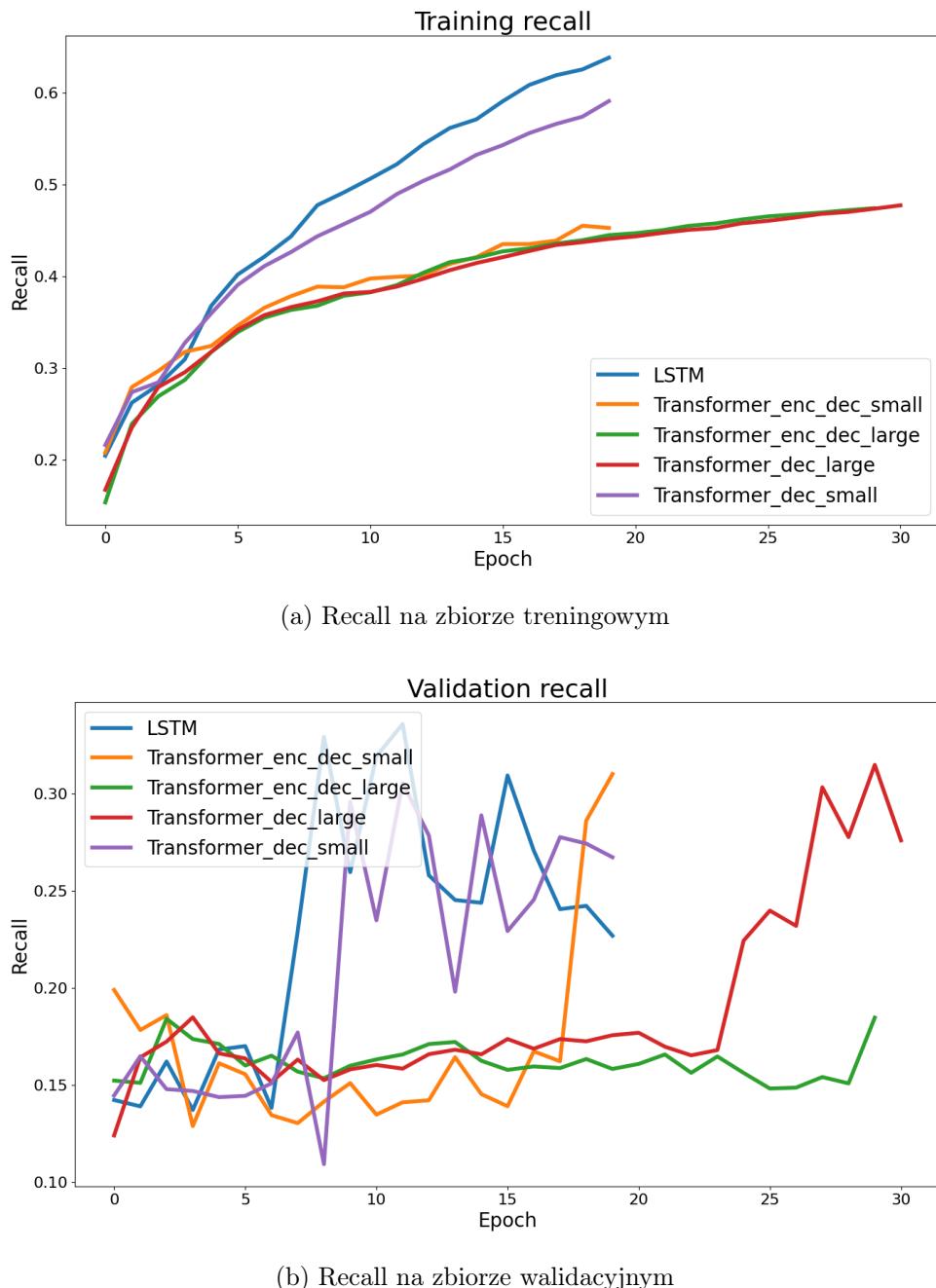
Na podstawie Rysunku 5.29b można stwierdzić, że najlepiej sprawdzają się modele Transformer_enc_dec i Transformer_enc_dec_large.

Recall

Kolejną ważną metryką jest Recall. Jest to stosunek liczby obserwacji true positive do sumy true positive i false negative. Ta metryka pokazuje jak dobrze model jest w stanie przewidywać nadchodzące trzęsienia ziemi. Na Rysunku 5.30 została przedstawiona zmiana recall podczas

5.3. EWALUACJA MODELI

trenowania modelów.

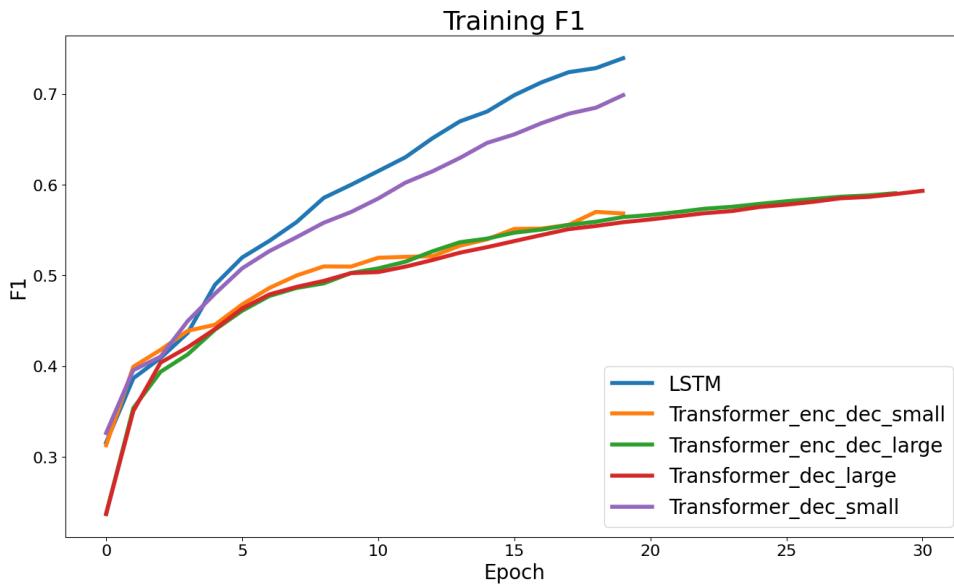


Rysunek 5.30: Zmiana recall podczas trenowania

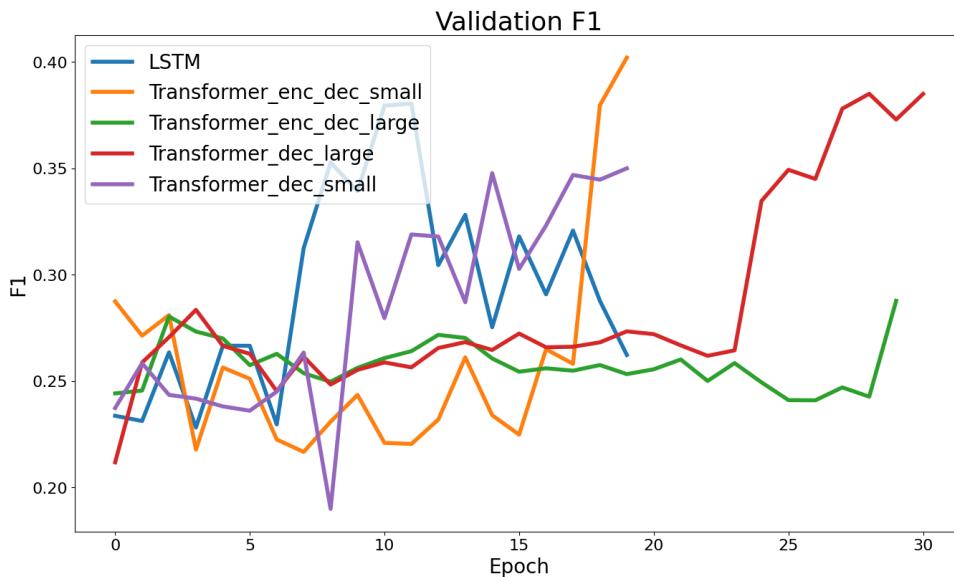
Na podstawie Rysunku 5.30b można stwierdzić, że większość modeli osiąga podobną wartość maksymalną. Najgorzej radzi sobie model Transformer_enc_dec_large.

F1

Ostatnią badaną metryką jest F1. Jest to średnia harmoniczna recall i precision. Na Rysunku 5.31 została przedstawiona zmiana F1 podczas trenowania.



(a) F1 na zbiorze treningowym



(b) F1 na zbiorze walidacyjnym

Rysunek 5.31: Zmiana F1 podczas trenowania

Na podstawie Rysunku 5.31b można stwierdzić, że najlepiej radzą sobie modele Transformer_enc_dec_small i Transformer_dec_large.

Ewaluacja na zbiorze testowym

W celu dalszej analizy wydajności modelów zostały one przetestowane na zbiorze testowym. W Tabeli 5.11 zostały przedstawione metryki modelów na zbiorze testowym. Pogrubione zostały najlepsze wyniki w danej metryce.

5.3. EWALUACJA MODELI

Nazwa modelu	Loss	Accuracy	Precision	Recall	F1
LSTM	0.1108	0.967	0.237	0.201	0.218
Transformer_dec_small	0.103	0.970	0.317	0.278	0.296
Transformer_dec_large	0.080	0.979	0.578	0.276	0.374
Transformer_enc_dec_small	0.088	0.978	0.531	0.289	0.374
Transformer_enc_dec_large	0.077	0.978	0.537	0.288	0.375

Tabela 5.11: Metryki modelów, trenowanych bez parametru *class_weights*, na zbiorze testowym

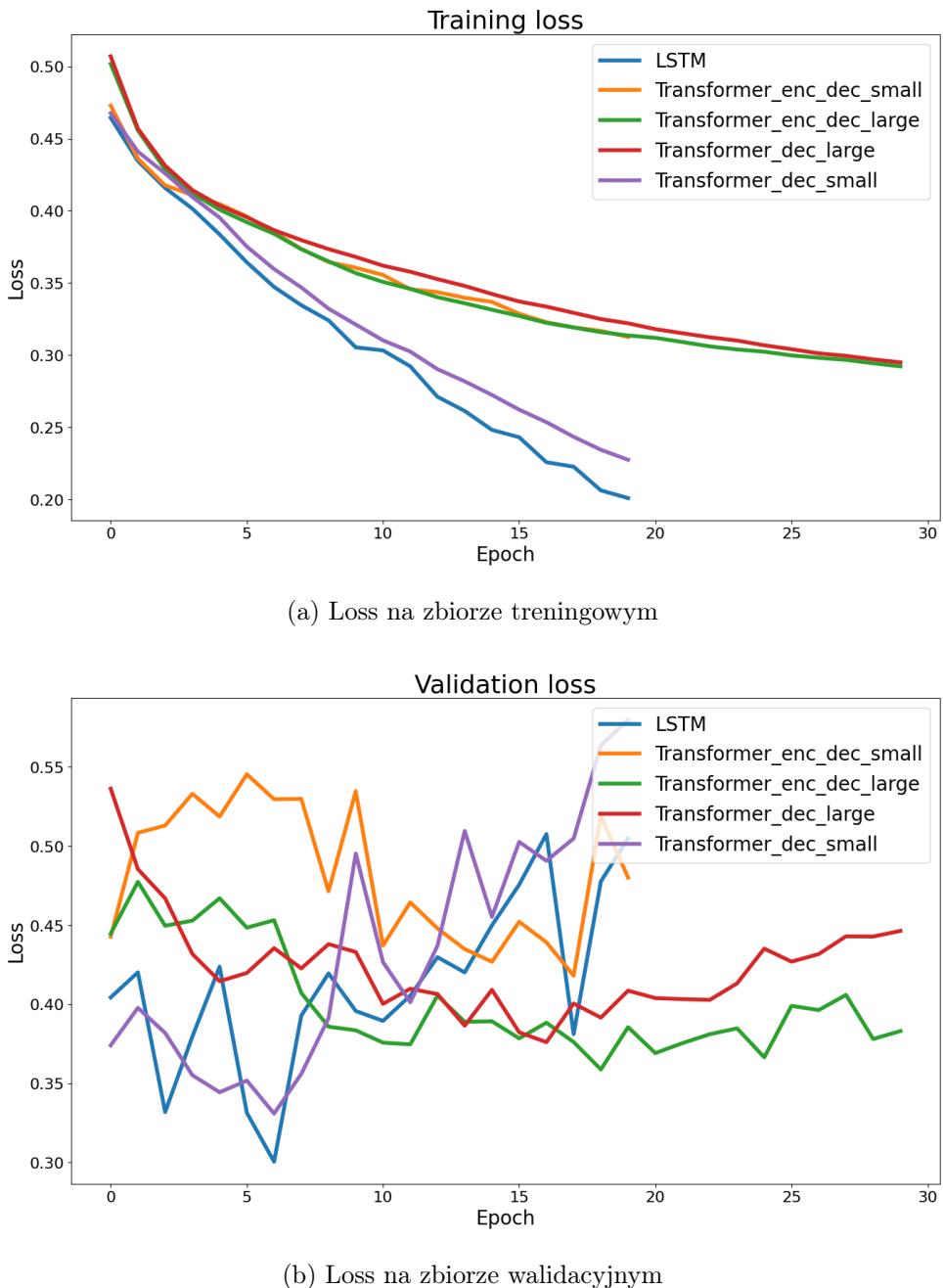
Modele trenowane bez parametru *class_weights* charakteryzują się bardzo wysoką metryką accuracy, na poziomie prawie 98%. Mimo to, poprawnie przewidują jedynie do 58% trzęsień.

Zgodnie z oczekiwaniem najlepiej sprawują się większe modele - Transformer_dec_large i Transformer_enc_dec_large.

5.3.2. Modele trenowane z parametrem *class_weights*

Loss

Na Rysunku 5.32 została przedstawiona loss podczas trenowania.



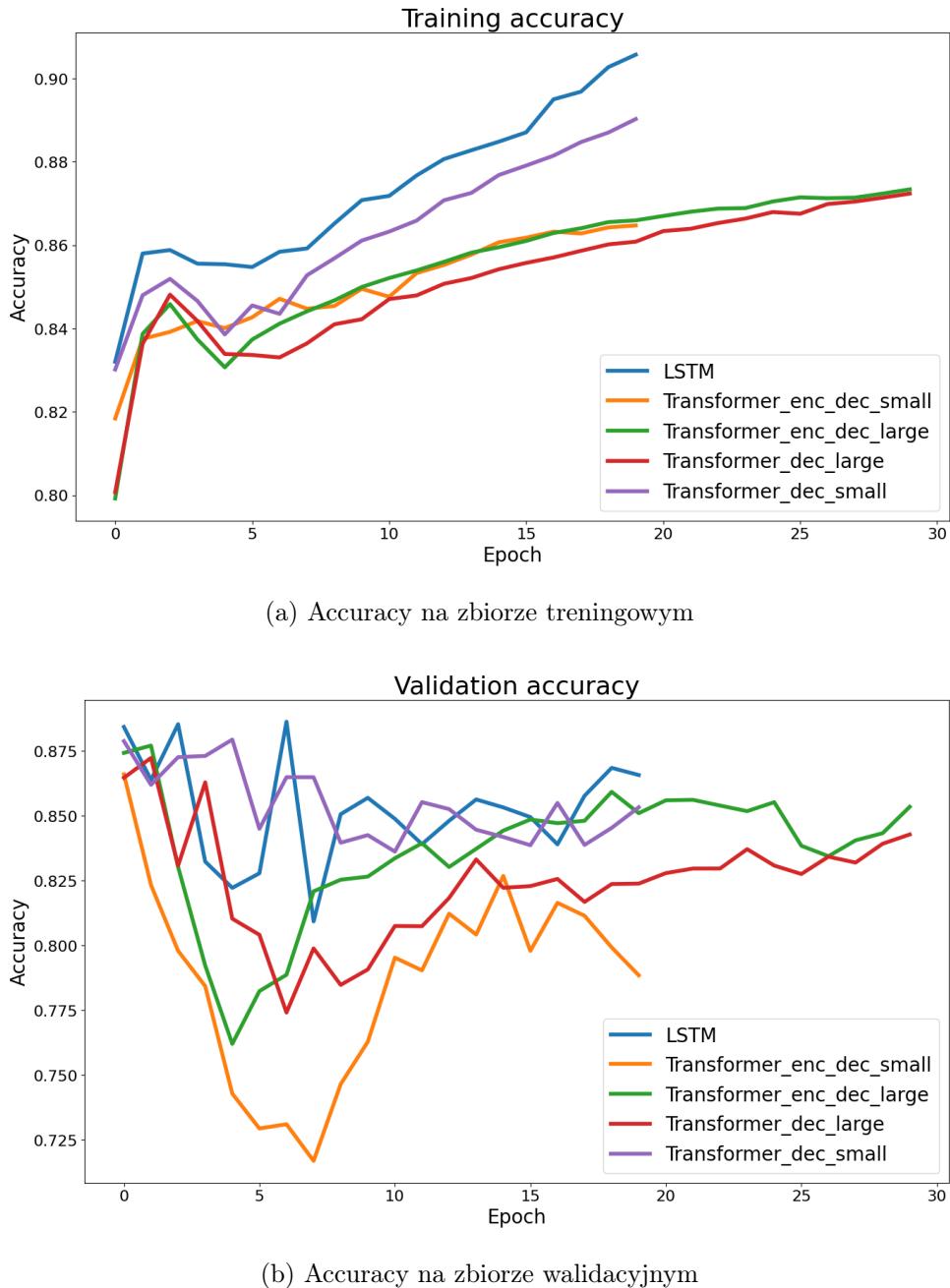
Rysunek 5.32: Zmiana loss podczas trenowania

Na podstawie Rysunku 5.32b można stwierdzić, że najlepiej radzą sobie modele Transformer_dec_large i Transformer_enc_dec_large.

Accuracy

Na Rysunku 5.33 została przedstawiona zmiana metryki accuracy w czasie trenowania.

5.3. EWALUACJA MODELI

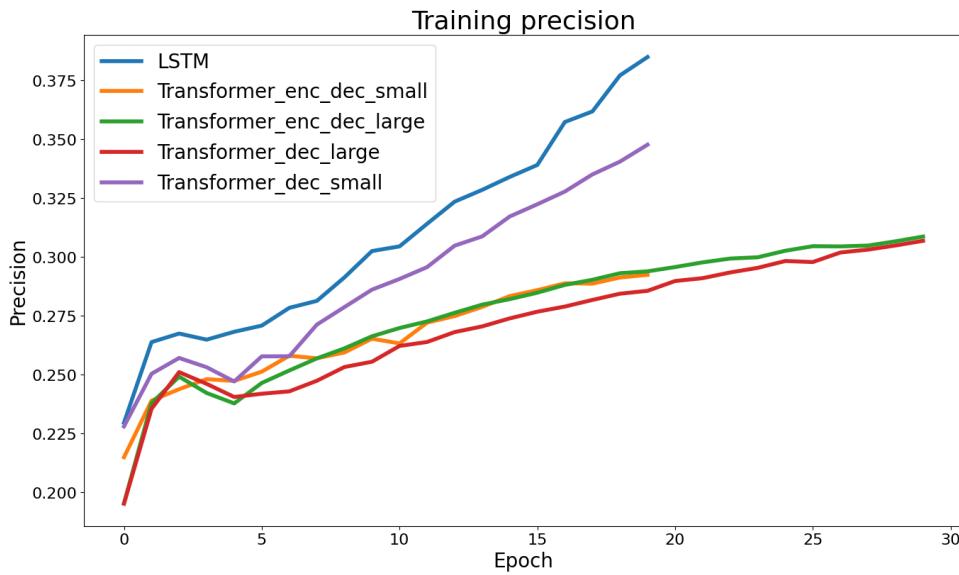


Rysunek 5.33: Zmiana accuracy podczas trenowania

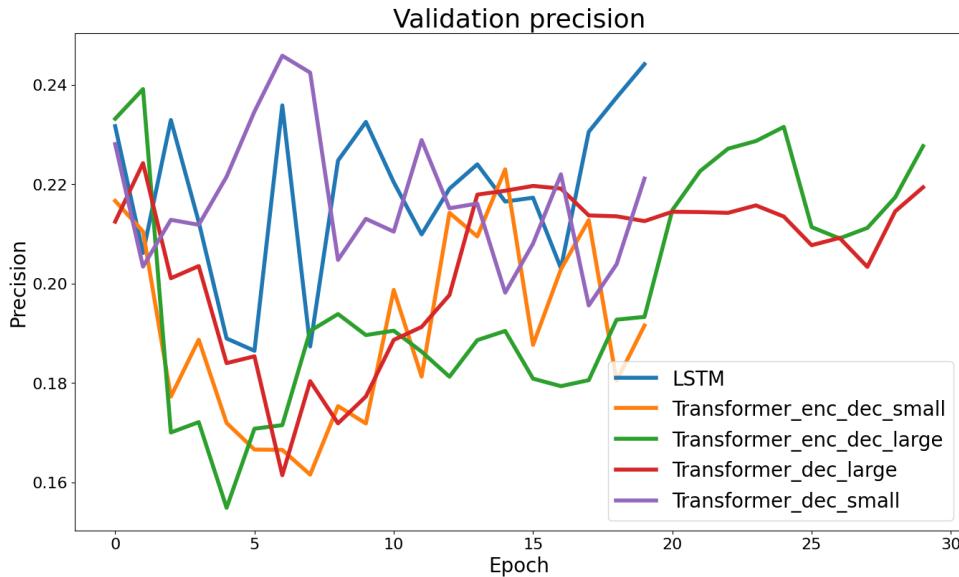
Na podstawie Rysunku 5.33b można stwierdzić, że najlepiej radzą sobie modele LSTM i Transformer_enc_dec_large.

Precision

Na Rysunku 5.34 została przedstawiona zmiana precision podczas trenowania. Wykres sugeruje, że najlepiej radzą sobie modele LSTM i Transformer_enc_dec_large, ale do potwierdzenia tej hipotezy konieczna jest też analiza wyników na zbiorze walidacyjnym.



(a) Precision na zbiorze treningowym



(b) Precision na zbiorze walidacyjnym

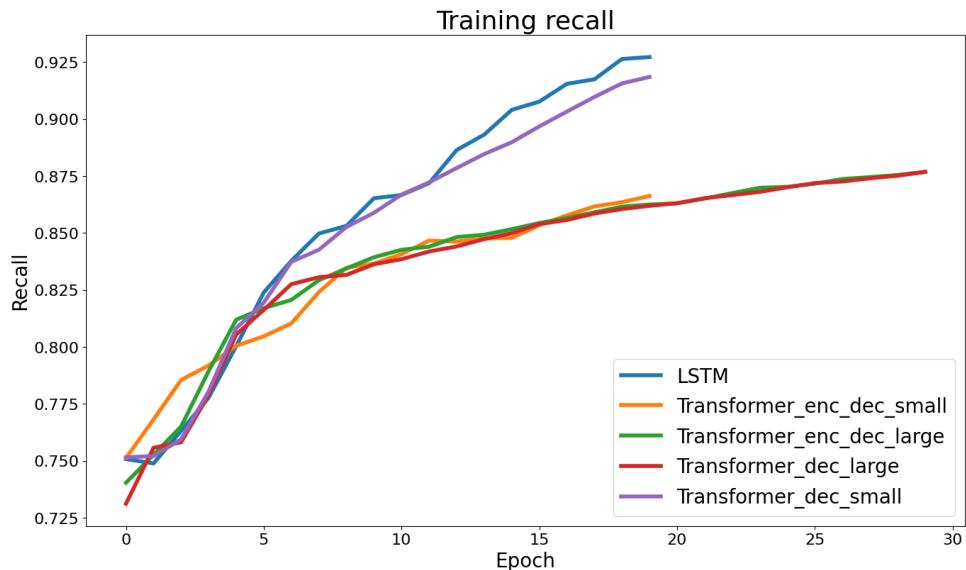
Rysunek 5.34: Zmiana precision podczas trenowania

Analiza Rysunku 5.34b pozwala potwierdzić, że najlepiej radzą sobie modele LSTM i Transformer_enc_dec_large.

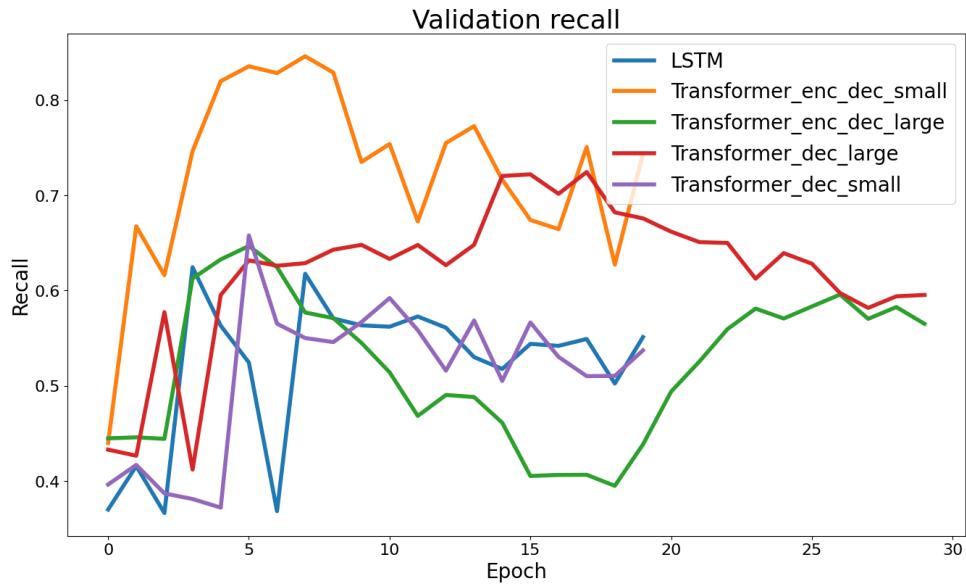
Recall

Na Rysunku 5.35 została przedstawiona zmiana metryki recall podczas trenowania.

5.3. EWALUACJA MODELI



(a) Recall na zbiorze treningowym



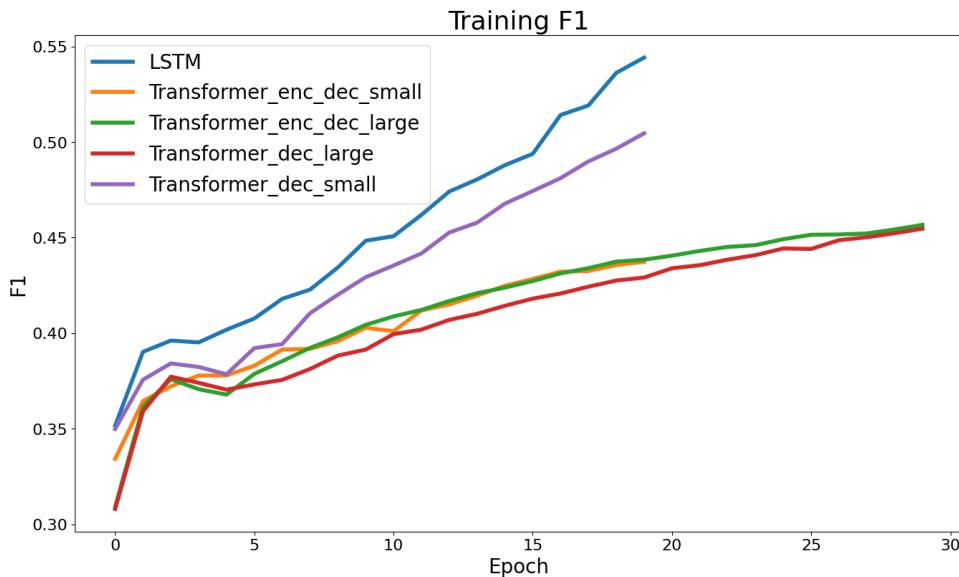
(b) Recall na zbiorze walidacyjnym

Rysunek 5.35: Zmiana recall podczas trenowania

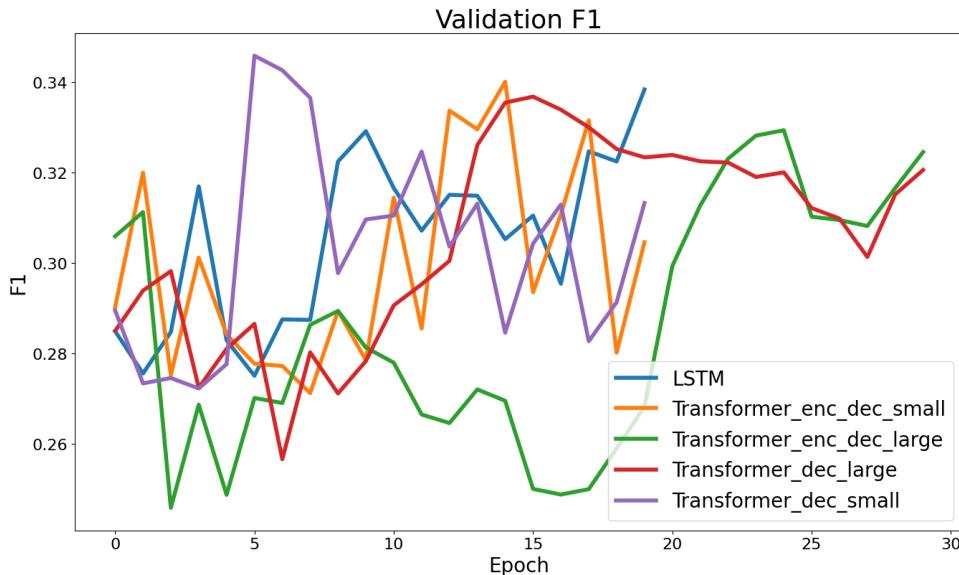
Na podstawie Rysunku 5.35b można stwierdzić, że najlepiej radzą sobie modele Transformer_enc_dec_small i Transformer_dec_large.

F1

Na Rysunku 5.36 została przedstawiona zmiana F1 podczas trenowania.



(a) F1 na zbiorze treningowym



(b) F1 na zbiorze walidacyjnym

Rysunek 5.36: Zmiana F1 podczas trenowania

Na podstawie Rysunku 5.36b można stwierdzić, że najlepiej radzą sobie modele LSTM i Transformer_dec_small

Ewaluacja na zbiorze testowym

W Tabeli 5.12 zostały przedstawione metryki modelów na zbiorze testowym. Pogrubione zostały najlepsze wyniki w danej metryce.

5.3. EWALUACJA MODELI

Nazwa modelu	Loss	Accuracy	Precision	Recall	F1
LSTM	0.281	0.892	0.141	0.748	0.237
Transformer_dec_small	0.418	0.801	0.086	0.811	0.155
Transformer_dec_large	0.338	0.872	0.122	0.755	0.210
Transformer_enc_dec_small	0.449	0.785	0.080	0.809	0.146
Transformer_enc_dec_large	0.323	0.878	0.133	0.797	0.228

Tabela 5.12: Metryki modelów, trenowanych z parametrem *class_weights*, na zbiorze testowym

Zdecydowanie najlepiej sprawuje się model LSTM.

5.3.3. Analiza wyników

Zgodnie z oczekiwaniami wykorzystanie parametru *class_weights* spowodowało, że modele dużo częściej przypisywały obserwacjom etykietę 1. Spowodowało to poprawienia metryki recall kosztem accuracy i precision. Modele trenowane z parametrem *class_weights* potrafią przewidzieć nawet do 80% trzęsień ziemi. Z drugiej strony bardzo często mylnie przewidują wystąpienia trzęsień, jedynie do 14% przewidywanych trzęsień kończy się faktycznym trzęsieniem.

Z powodu bardzo niskiej metryki precision wśród modelów trenowanych z parametrem *class_weights* została ustalona wyższość modeli trenowanych bez parametru *class_weights*. Ostatecznie został wybrany model Transformer_enc_dec_large trenowany bez parametru *class_weights*, który posiada najlepszą metrykę loss oraz F1, wśród wszystkich modeli.

6. Wyniki - aplikacja

6.1. Opis aplikacji

Backend aplikacji został utworzony we frameworku Django[19], a frontend został przygotowany za pomocą standardowych narzędzi - CSS i JS - oraz dodatkowo rozwinięty przez framework Bootstrap[20] i bibliotekę Font Awesome[21]. Aplikacja składa się z 3 zakładek - **Home**, **App** i **Admin panel**. W zakładce **Home** znajdują się informacje o projekcie. Zakładka **App** jest główną częścią aplikacji. Znajduje się w niej mapa z oznaczonymi trzęsieniami ziemi. Mapa jest obsługiwana za pośrednictwem biblioteki Leaflet[22]. Można tam również dokonać predykcji dla wybranego regionu. Zakładka **Admin panel** daje dostęp do tabel i modeli w bazie danych.

6.2. Dependencje

Aplikacja została przygotowana w Pythonie w wersji 3.11.2. Wiele funkcjonalności zależy od zewnętrznych bibliotek. Dla wygody konfiguracji zostały one umieszczone w pliku **requirements.txt**. W Tabeli 6.1 przedstawiono listę zależności wraz z wymaganymi wersjami i skróconym opisem. Dodatkowo, wymagana jest instalacja oprogramowania obsługującego przestrzenne bazy danych, opisana w punkcie 6.3.1.

6.3. INSTRUKCJA INSTALACJI

Biblioteka	Wersja
Funkcjonalności aplikacji	
Django	4.2.6
djangorestframework	3.14.0
requests	2.31.0
Wygląd aplikacji	
django-bootstrap-v5	1.0.11
fontawesomefree	6.5.1
Analiza i modelowanie	
numpy	1.26.3
pandas	2.1.2
matplotlib	3.8.2
tensorflow	2.14.0
scikit-learn	1.3.2
h5py	3.10.0
Inne	
tqdm	4.66.1
psycopg2	2.9.9
postgis	1.0.4

Tabela 6.1: Dependencje

6.3. Instrukcja instalacji

Konfigurację aplikacji wystarczy wykonać jednorazowo. Kroki zawarte w sekcji uruchomienie aplikacji trzeba wykonywać przy każdorazowym uruchomieniu aplikacji.

6.3.1. Konfiguracja aplikacji

Na początku należy zainstalować dodatkowe oprogramowanie do obsługi przestrzennych baz danych, zgodnie z instrukcją zawartą na stronie django. Proces ten jest wyjaśniony krok po kroku, dla systemów Linux, Windows oraz MacOS.

Następnie należy utworzyć bazę zgodnie z instrukcją na stronie django. Należy zmienić `<db name>` na `website_db`. Podczas zakładania bazy należy ustawić nazwę użytkownika oraz hasło

na `postgres`.

Kolejnym krokiem jest sklonowanie repozytorium i utworzenie środowiska wirtualnego:

```
git clone https://github.com/Michal1337/eq_website.git
cd eq_website
python -m venv venv
source venv/Scripts/activate
```

Następnie należy zainstalować wymagane biblioteki:

```
pip install -r requirements.txt
```

Należy stworzyć i wykonać migracje:

```
cd website
python manage.py makemigrations
python manage.py migrate
```

Aby mieć dostęp do funkcjonalności administratora, należy utworzyć superusera:

```
python manage.py createsuperuser
```

Podczas tworzenia superusera nie trzeba podawać adresu mailowego, wystarczy nazwa użytkownika i hasło.

Na koniec należy zasilić bazę danymi:

```
python manage.py load_countries
python manage.py load_api
python manage.py load_models
```

Polecenie `load_api` ładuje trzęsienia ziemi do bazy. Domyślnym przedziałem czasowym, z którego są pobierane dane to 2023-12-01 do 2023-12-31. W przypadku chęci zmiany przedziału czasowego należy użyć dodatkowych parametrów:

```
python manage.py load_api --start_date <date> --end_date <date>
```

6.4. INSTRUKCJA OBSŁUGI

W miejscu <date> należy wpisać datę w formacie YYYY-mm-dd, przykładowo 2020-01-13.

6.3.2. Uruchomienie aplikacji

Aby uruchomić aplikację należy wykonać następujące polecenie, z folderu **website**:

```
python manage.py runserver
```

Od teraz aplikacja działa pod adresem `http://127.0.0.1:8000/`. Aby zatrzymać serwer należy nacisnąć kombinację klawiszy **ctrl+c** w konsoli.

6.4. Instrukcja obsługi

6.4.1. Home

Home jest zakładką, na którą użytkownik przenosi się automatycznie po uruchomieniu aplikacji. Widnieją na niej informacje na temat projektu. Aby przenieść się do niej ponownie z innych zakładek, należy wcisnąć przycisk **Home** na panelu nawigacyjnym na górze ekranu.

6.4.2. App

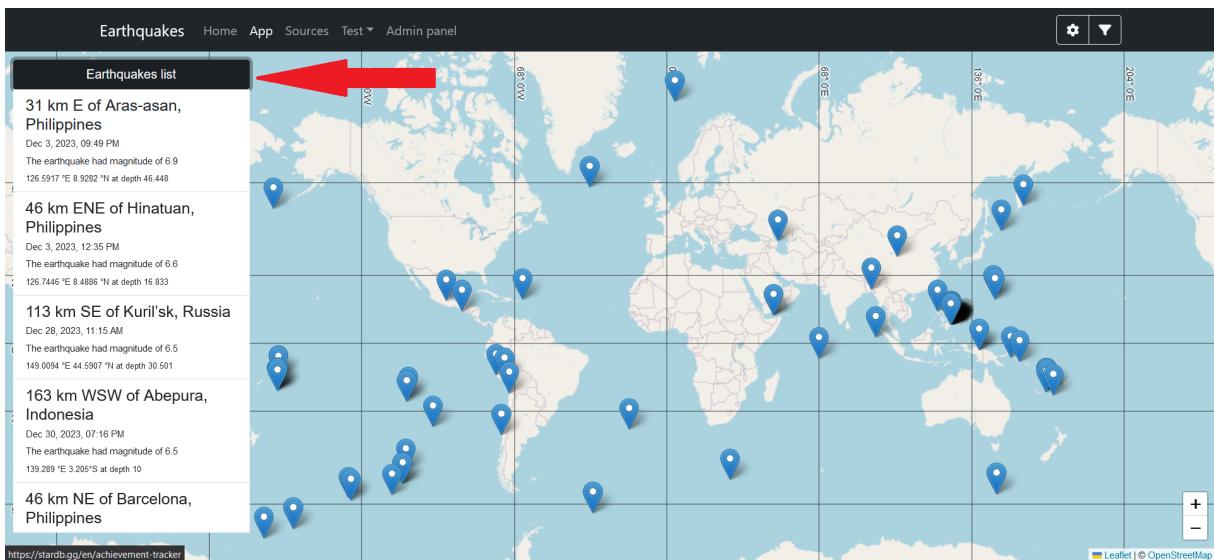
Po kliknięciu na zakładkę **App** w panelu nawigacyjnym, użytkownik może zobaczyć mapę świata z markerami wskazującymi na miejsca wystąpienia trzęsienia ziemi. W celu zachowania odpowiedniej wydajności domyślnie wyświetlane jest 100 trzęsień o największej magnitudzie spełniających podane filtry:

Filtr	Wartość filtra
Data	od 2023-12-01 do 2023-12-31
Magnituda	od 2 do 7
Głębokość	od 0 do 100

Tabela 6.2: Domyślne parametry filtrowania

W lewej górnej części mapy widnieje przycisk **Earthquakes list**. Po naciśnięciu go wyświetla się lista pokazywanych trzęsień ziemi (Rysunek 6.1). Wciśnięcie dowolnego rekordu z listy powoduje wycentrowanie mapy w miejscu trzęsienia. Przycisk **Earthquakes list** można nacisnąć ponownie, aby ukryć listę trzęsień.

6. WYNIKI - APLIKACJA



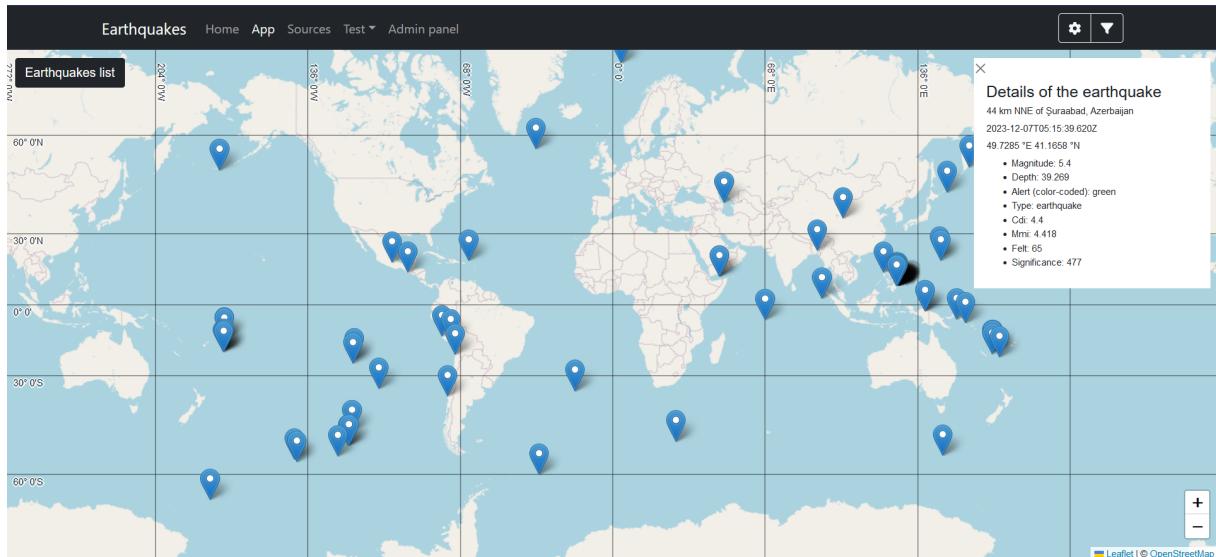
Rysunek 6.1: Lista trzęsień ziemi

Markery na mapie mogą zostać wciśnięte. W prawym górnym rogu mapy pojawiają się wtedy następujące szczegóły trzęsienia:

- Nazwa
- Timestamp
- Współrzędne geograficzne trzęsienia
- Magnitude
- Depth
- Alert
- Type
- CDI
- MMI
- Felt
- Significance

Wygląd szczegółów trzęsienia został przedstawiony na Rysunku 6.2

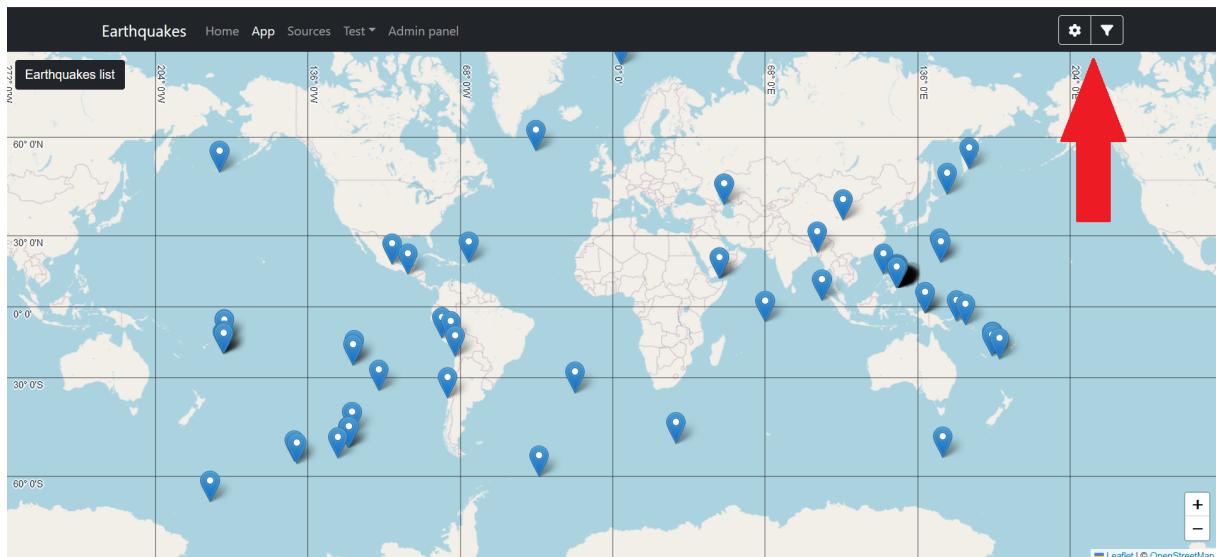
6.4. INSTRUKCJA OBSŁUGI



Rysunek 6.2: Szczegóły trzęsienia ziemi

Dokładniejsze informacje o poszczególnych polach zostały przedstawione w sekcji Modele django - earthquake. Więcej informacji na temat szczegółów trzęsienia i dodatkowych pól dostępnych w API USGS dostępne są na stronie z dokumentacją[23].

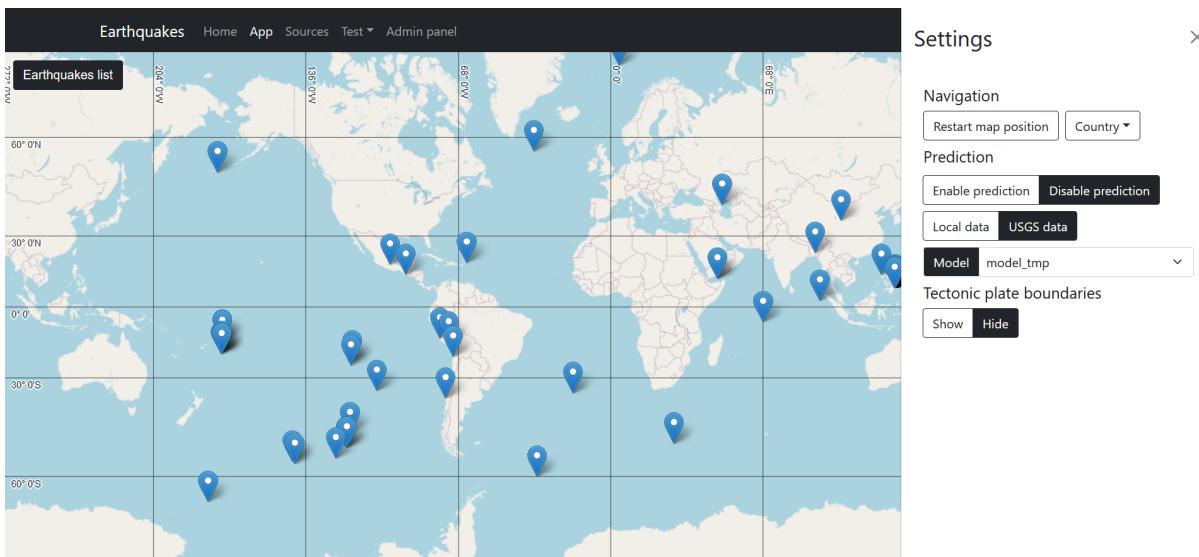
W prawym górnym rogu aplikacji, ponad mapą, znajdują się dwie zakładki - ustawienia i filtry (Rysunek 6.3).



Rysunek 6.3: Zakładki - ustawienia i filtry

Zawartość zakładki ustawień została przedstawiona na Rysunku 6.4.

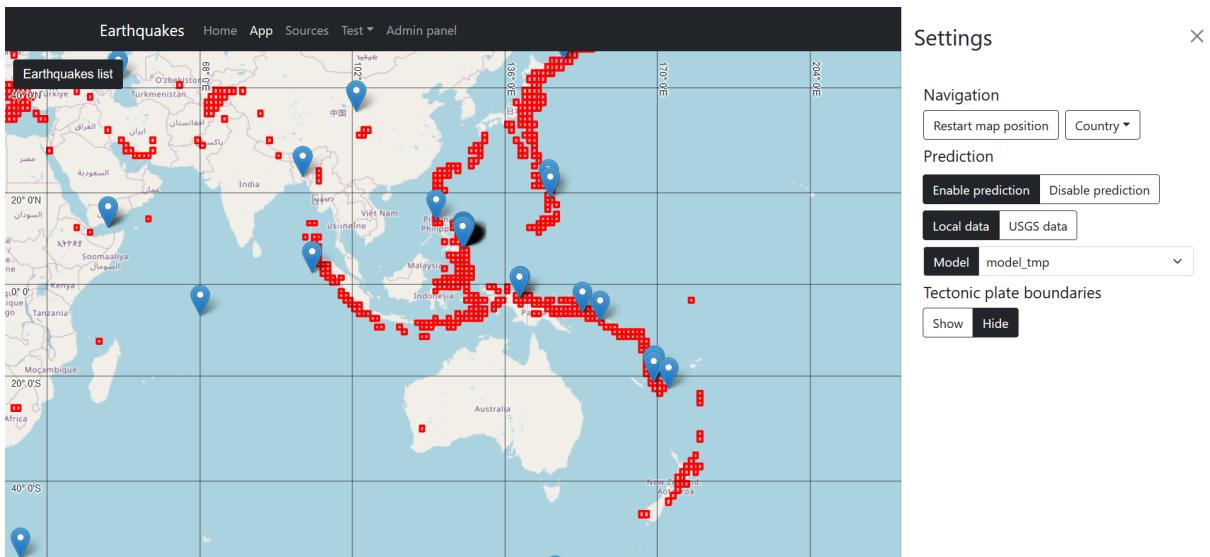
6. WYNIKI - APLIKACJA



Rysunek 6.4: Zakładka ustawień

W zakładce ustawień znajduje się sekcja **Navigation**. Widnieje w niej przycisk **Restart map position**. Wciśnięcie go powoduje wycentrowanie się mapy na współrzędnych 0° , 0° . Obok niego znajduje się lista rozwijana **Country**. Pozwala ona na wybranie najpierw kontynentu, a następnie kraju. Po wybraniu nazwy mapa centruje się na danym kraju.

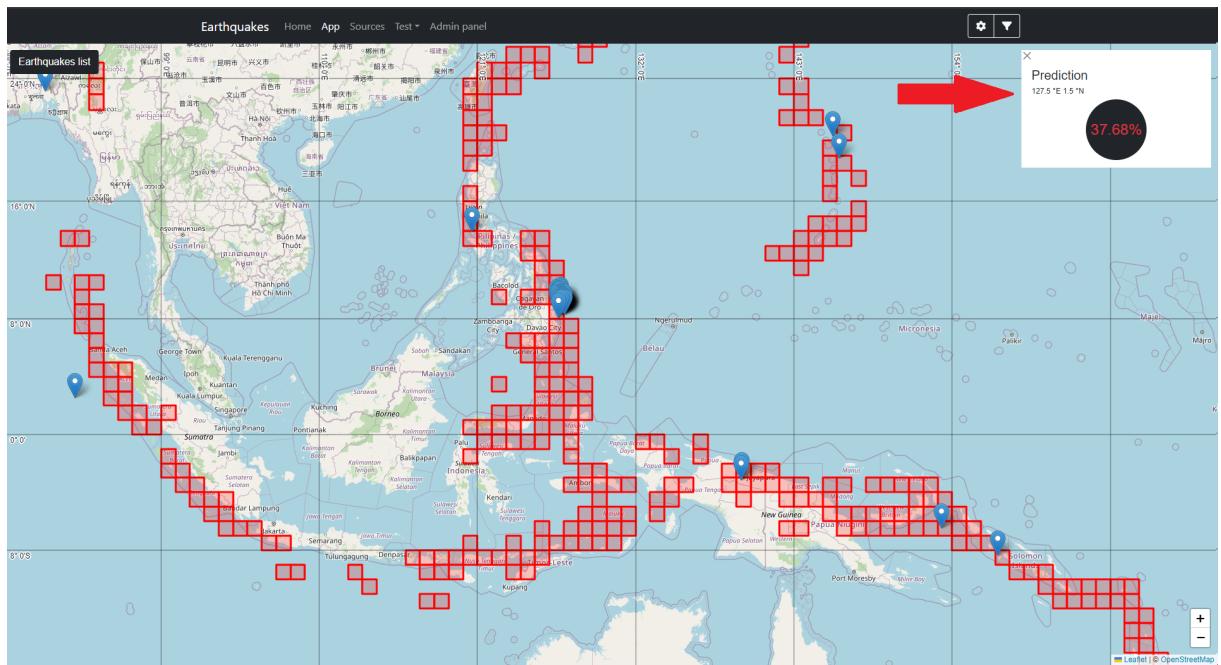
Kolejną sekcją w zakładce ustawień jest **Prediction**. Dzięki przełącznikowi **Enable / Disable prediction** użytkownik jest w stanie wejść bądź wyjść z trybu predykcji. Gdy tryb predykcji jest włączony, na mapie pojawiają się czerwone kwadraty o wymiarach $1^\circ \times 1^\circ$, przedstawione na Rysunku 6.5. Są to regiony, w których możliwe jest wykonanie predykcji.



Rysunek 6.5: Regiony predykcji

6.4. INSTRUKCJA OBSŁUGI

Po kliknięciu na taki kwadrat, w prawym górnym rogu mapy pojawiają się współrzędne i predykcja dla danego regionu (Rysunek 6.6).

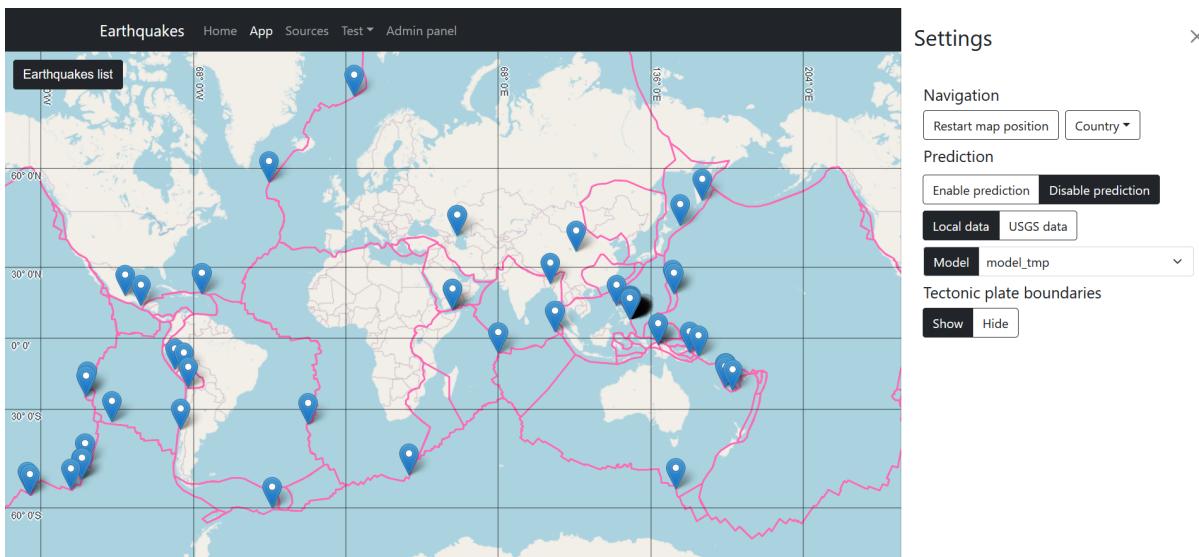


Rysunek 6.6: Wykonanie predykcji

W sekcji **Prediction** znajduje się również przełącznik **Local data / USGS data**. Dzięki niemu można wybrać jakie dane będą źródłem predykcji - dane z lokalnej bazy lub z API USGS. Pod spodem widnieje lista rozwijana Model. Można z niej wybrać który z modeli załadowanych do bazy będzie wykonywać predykcję.

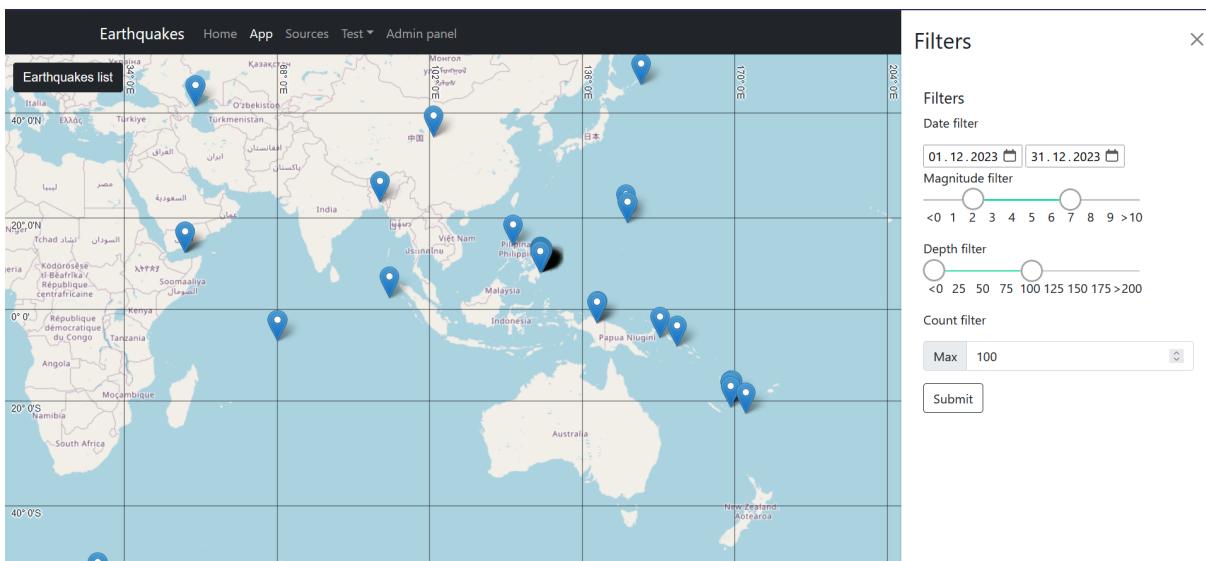
Ostatnią sekcją w zakładce ustawień jest **Tectonic plate boundaries**. Znajduje się w niej przełącznik, który pozwala na pokazywanie granic płyt tektonicznych na mapie, tak jak pokazane na Rysunku 6.7.

6. WYNIKI - APLIKACJA



Rysunek 6.7: Granice płyt tektonicznych

Drugą zakładką są filtry (rysunek 6.8). Pozwalają one na filtrowanie markerów na mapie. Istnieją 4 filtry: data, magnituda, głębokość i maksymalna liczba markerów.



Rysunek 6.8: Zakładka filtrów

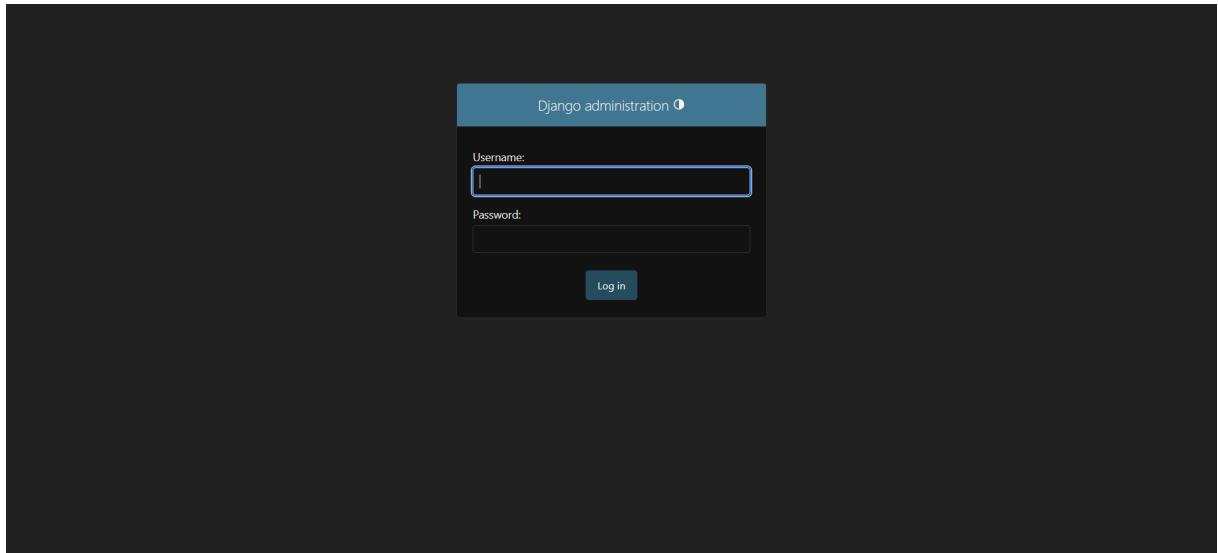
Domyślnie pokazywane jest 100 markerów. Liczbę tę można zmienić korzystając z **Count filter**. Należy zaznaczyć, że ze względu na wydajność aplikacji maksymalna liczba markerów nie powinna przekroczyć 1 000.

Po ustawieniu wszystkich interesujących filtrów należy wcisnąć przycisk **Submit**. Zostaną wtedy dodatkowo sprawdzone granice aktualnie widocznej mapy. Dzięki temu po przefiltrowaniu będą widoczne jedynie te markery, które znajdują się w danych granicach.

6.4. INSTRUKCJA OBSŁUGI

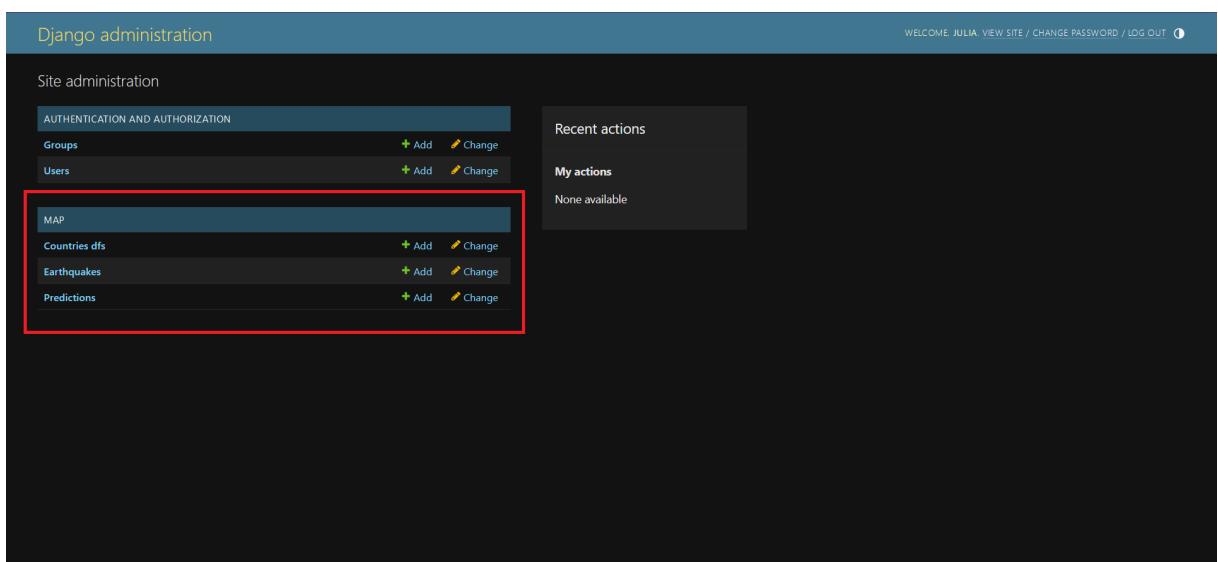
6.4.3. Admin panel

Po przejściu do zakładki **Admin panel** wyświetli się strona logowania (Rysunek 6.9). Należy zalogować się na konto superusera stworzonego podczas konfiguracji aplikacji.



Rysunek 6.9: Logowanie

Następnie aplikacja pokazuje główną stronę panelu administratora. Poszczególne źródła danych przedstawione są w sekcji **MAP** (Rysunek 6.10).



Rysunek 6.10: Strona główna panelu administratora z wyróżnioną sekcją **MAP**

Po kliknięciu w dowolną tabelę następuje przeniesienie do jej szczegółów (Rysunek 6.11). Dostępne są tam opcje dodania nowego rekordu (zielona strzałka) i filtrowania rekordów (niebieska strzałka). Możliwe jest również usuwanie rekordów (Rysunek 6.12).

6. WYNIKI - APLIKACJA

Django administration

Home > Map > Earthquakes

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MAP

Countries dfs + Add

Earthquakes + Add

Predictions + Add

Select earthquake to change

Action: ----- Go 0 of 100 selected

PLACE	TYPE	TIME	LONGITUDE	LATITUDE	MAG
3 km SSW of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:10 a.m.	-146.9205	64.4911	1.6
9 km E of Fox, Alaska	earthquake	Jan, 21, 2024, 1:10 a.m.	-147.4252	64.9467	1.6
13 km NNW of Chenega, Alaska	earthquake	Jan, 21, 2024, 1:17 a.m.	-148.085	60.1814	1.5
4 km S of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:17 a.m.	-146.9075	64.4877	1.6
8 km SW of Guánica, Puerto Rico	earthquake	Jan, 21, 2024, 1:20 a.m.	-66.96983333333333	17.91983333333333	2.91
5 km SSE of Talkeetna, Alaska	earthquake	Jan, 21, 2024, 1:23 a.m.	-150.074	62.2787	1.8
3 km SSE of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:25 a.m.	-146.8792	64.4943	1.6
34 km NW of Toyah, Texas	earthquake	Jan, 21, 2024, 1:26 a.m.	-104.022	31.557	2.0
83 km NNW of Karluk, Alaska	earthquake	Jan, 21, 2024, 1:32 a.m.	-154.9581666666667	58.27233333333333	-0.12
5 km S of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:39 a.m.	-146.9013	64.4709	1.0
6 km WSW of Volcano, Hawaii	earthquake	Jan, 21, 2024, 1:43 a.m.	-155.283172607422	19.4120006561279	1.91999996
78 km ESE of Denali Park, Alaska	earthquake	Jan, 21, 2024, 1:46 a.m.	-147.3824	63.5551	1.6

Rysunek 6.11: Szczegóły tabeli

Django administration

Home > Map > Earthquakes

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MAP

Countries dfs + Add

Earthquakes + Add

Predictions + Add

Select earthquake to change

Action: ----- Go 4 of 100 selected

PLACE	TYPE	TIME	LONGITUDE	LATITUDE	MAG
3 km SSW of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:10 a.m.	-146.9205	64.4911	1.6
9 km E of Fox, Alaska	earthquake	Jan, 21, 2024, 1:10 a.m.	-147.4252	64.9467	1.6
13 km NNW of Chenega, Alaska	earthquake	Jan, 21, 2024, 1:17 a.m.	-148.085	60.1814	1.5
4 km S of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:17 a.m.	-146.9075	64.4877	1.6
8 km SW of Guánica, Puerto Rico	earthquake	Jan, 21, 2024, 1:20 a.m.	-66.96983333333333	17.91983333333333	2.91
5 km SSE of Talkeetna, Alaska	earthquake	Jan, 21, 2024, 1:23 a.m.	-150.074	62.2787	1.8
3 km SSE of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:25 a.m.	-146.8792	64.4943	1.6
34 km NW of Toyah, Texas	earthquake	Jan, 21, 2024, 1:26 a.m.	-104.022	31.557	2.0
83 km NNW of Karluk, Alaska	earthquake	Jan, 21, 2024, 1:32 a.m.	-154.9581666666667	58.27233333333333	-0.12
5 km S of Salcha, Alaska	earthquake	Jan, 21, 2024, 1:39 a.m.	-146.9013	64.4709	1.0
6 km WSW of Volcano, Hawaii	earthquake	Jan, 21, 2024, 1:43 a.m.	-155.283172607422	19.4120006561279	1.91999996
78 km ESE of Denali Park, Alaska	earthquake	Jan, 21, 2024, 1:46 a.m.	-147.3824	63.5551	1.6

Rysunek 6.12: Usuwanie rekordów z tabeli

Po wciśnięciu dowolnego rekordu zostają pokazane jego szczegóły (Rysunek 6.13). Część z nich można edytować.

6.5. TESTY

The screenshot shows a web-based application interface for managing geological data. On the left, there's a sidebar with a search bar and several menu items: AUTHENTICATION AND AUTHORIZATION, Groups (+ Add), Users (+ Add), MAP, Countries dfs (+ Add), Earthquakes (+ Add), and Predictions (+ Add). The main content area is titled "Change earthquake" and shows a record for an earthquake that occurred at 9 km E of Fox, Alaska, on January 21, 2024, at 01:10:58. The record includes the following details:

Field	Value
Date	2024-01-21
Time	01:10:58
Longitude	-147.4252
Latitude	64.9467
Depth	6.0
Mag	1.6
MagType	ml
Place	9 km E of Fox, Alaska
Alert	(empty)
Type	earthquake
Cdi	(empty)
Mmi	(empty)
Felt	(empty)
Sig	39

At the bottom of the form, there are four buttons: "SAVE", "Save and add another", "Save and continue editing", and a red "Delete" button.

Rysunek 6.13: Szczegóły rekordu w tabeli Earthquake

6.5. Testy

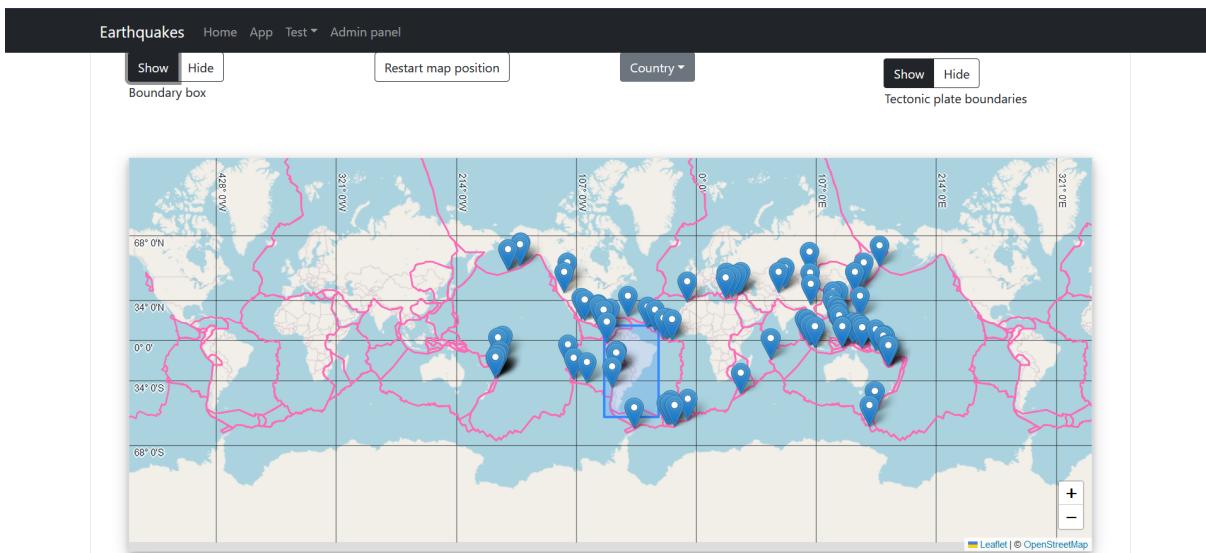
Testy poszczególnych funkcjonalności aplikacji zostały utworzone w dodatkowych, oddzielnych podstronach.

6.5.1. Mapa

Początkowo przetestowane zostały funkcjonalności mapy - przesuwanie, oddalenie i zbliżanie. Działają one poprawnie.

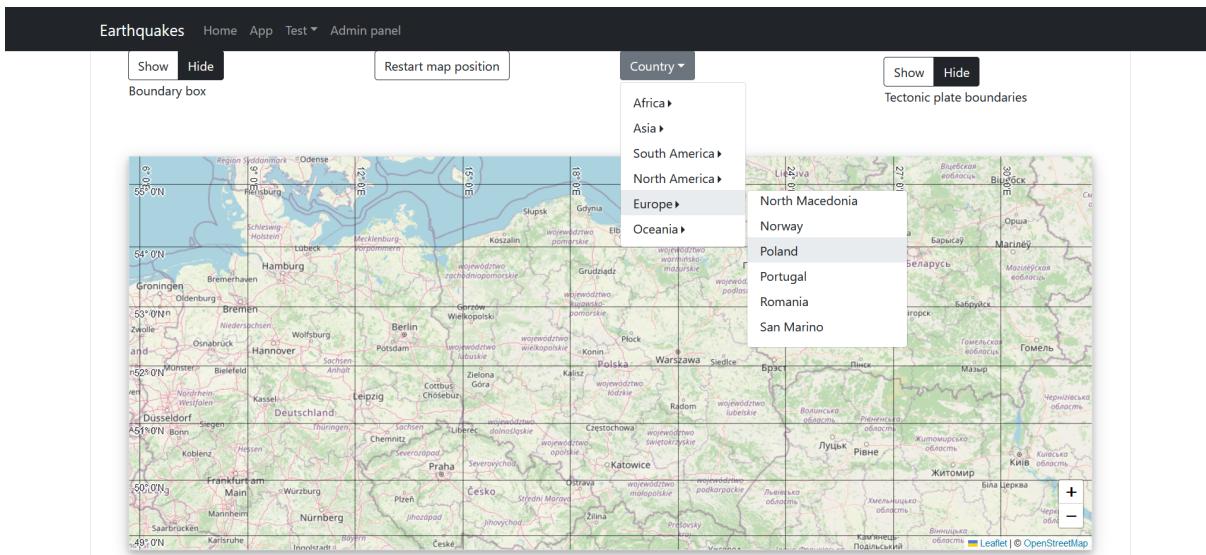
Na mapie powinny pojawiać się również pinezki oznaczające miejsca trzęsień ziemi zapisanych w naszej bazie danych. Na Rysunku 6.14 widać jak wygląda mapa z wyświetlonymi danymi. Włączone zostały na nim również funkcjonalności pokazywania granic płyt tektonicznych oraz wyszczególnionego obszaru.

6. WYNIKI - APLIKACJA



Rysunek 6.14: Testy mapy

Następnie została przetestowana funkcjonalność przenoszenia się mapą do wybranego kraju po użyciu listy rozwijanej (Rysunek 6.15).



Rysunek 6.15: Test przemieszczania po krajach

6.5.2. Baza trzęsień ziemi

Testowanie bazy polega na wypisaniu części trzęsień w niej zawartych oraz ich szczegółów. Widoczne jest to na rysunku 6.16.

6.5. TESTY

The screenshot shows a web application titled "Earthquakes". At the top, there is a navigation bar with links: "Home", "App", "Test ▾", and "Admin panel". Below the navigation bar, the main content area has a title "Earthquakes". It displays two earthquake entries:

- Alaska Peninsula**
The earthquake had magnitude of 8.2
Other features of the earthquake:
 - Alert (color-coded): yellow
 - Type: earthquake
 - Magnitude type: mww
 - Cdi: 8.1
 - Mmi: 7.355
 - Felt: 268
 - Significance: 1252157.8876°W 55.3635 °N at depth 35
Jul 29, 2021, 10:15 AM
- Kermadec Islands, New Zealand**
The earthquake had magnitude of 8.1
Other features of the earthquake:
 - Alert (color-coded): green
 - Type: earthquake
 - Magnitude type: mww
 - Cdi: 9.1Mar 4, 2021, 09:28 PM

Rysunek 6.16: Lista trzęsień ziemi

Widok administratora wygląda następująco:

The screenshot shows the "Admin panel" view of the application. The title is "Select earthquake to change". There is a search bar with a magnifying glass icon and a "Search" button. Below the search bar, there is a dropdown menu labeled "Action: -----" and a "Go" button. To the right of the search bar, it says "0 of 100 selected". A large table follows, with columns: PLACE, TYPE, TIME, LONGITUDE, LATITUDE, and MAG. The table contains the following data:

PLACE	TYPE	TIME	LONGITUDE	LATITUDE	MAG
3 km SSW of Salcha, Alaska	earthquake	Jan. 21, 2024, 1:10 a.m.	-146.9205	64.4911	1.6
9 km E of Fox, Alaska	earthquake	Jan. 21, 2024, 1:10 a.m.	-147.4252	64.9467	1.6
13 km NNW of Chenega, Alaska	earthquake	Jan. 21, 2024, 1:17 a.m.	-148.085	60.1814	1.5
4 km S of Salcha, Alaska	earthquake	Jan. 21, 2024, 1:17 a.m.	-146.9075	64.4877	1.6
8 km SW of Guánica, Puerto Rico	earthquake	Jan. 21, 2024, 1:20 a.m.	-66.96983333333333	17.91983333333333	2.91

Rysunek 6.17: Lista trzęsień ziemi z widoku administratora

6.5.3. API

Strona testowa API udostępnia przykładowe endpointy API i parametry do wysyłania zapytań (rysunek 6.18a). Na rysunku 6.18b widać przykładową odpowiedź.

```

JSON Nieprzetworzone dane Nagłówki
Zapisz Kopij Zwiń wszystkie Rozwiń wszystkie (może to chwilę zająć) Filtruj JSON
type: "FeatureCollection"
crs: { ... }
features:
  0:
    type: "Feature"
    id: 241527
    properties:
      time: "2020-06-10T21:58:50.008Z"
      depth: 10
      mag: 6
      magType: "mW"
      place: "southern Mid-Atlantic Ridge"
      alert: "green"
      type: "earthquake"
      cdi: null
      mmi: 0
      felt: null
      sig: 554
    geometry:
      type: "Point"
      coordinates:
        0: -13.0363
        1: -15.8069
  1:
    type: "Feature"
    id: 233862
    properties:
      time: "2020-05-24T04:52:24.009Z"
      depth: 10
      mag: 5.8
      magType: "mW"
      place: "southern Mid-Atlantic Ridge"

```

Example of usage:

- List all earthquakes: [/api/eqs](#)
- Set `count` parameter to `True` to only retrieve count of matching entries: [/api/eqs?count=True](#)
- Filter the list by `longitude` and `latitude`: [/api/eqs?minlatitude=-50&maxlatitude=0&minlongitude=-15&maxlongitude=0](#)
- Filter the list by `magnitude` and `depth`: [/api/eqs?minmagnitude=5&maxdepth=2](#)

(a) Przykładowe zapytania

(b) Przykładowa odpowiedź

Rysunek 6.18: Testy API

6.6. Dokumentacja techniczna

6.6.1. Baza danych

W aplikacji występuje baza danych PostgreSQL[24]. Rozszerzona jest ona funkcjonalnościami PostGIS[25], które pozwalają lepiej obsługiwać dane przestrzenne. W bazie danych występują 3 tabele: Countries dfs, Earthquakes i Predictions.

6.6.2. Modele django

CountriesDF

Model CountriesDF kontrabuuje rekordy do tabeli Countries dfs. W modelu znajdują się następujące pola:

- name (CharField) - nazwa kraju
- continent (CharField) - nazwa kontynentu, na którym kraj się znajduje
- lon (FloatField) - długość geograficzna środka kraju

6.6. DOKUMENTACJA TECHNICZNA

- lat (FloatField) - szerokość geograficzna środka kraju

Dane na temat krajów zostały pobrane ze strony World Population Review[26] i zapisane do pliku `countries.xlsx`. Przygotowanie źródła odbywa się poprzez skrypt `countries.ipynb`.

Earthquake

Model `Earthquake` kontrybuje rekordy do tabeli `Earthquakes`. W modelu znajdują się następujące pola:

- time (DateTimeField) - data i czas wystąpienia trzęsienia
- longitude (FloatField) - długość geograficzna
- latitude (FloatField) - szerokość geograficzna
- location (PointField) - lokalizacja, pole specjalne bazy przestrzennej
- depth (FloatField) - głębokość trzęsienia (w km)
- mag (FloatField) - magnituda
- magType (CharField) - algorytm użyty do obliczenia magnitudy (więcej informacji na stronie USGS)
- place (CharField) - opis miejsca, w którym odbyło się trzęsienie
- alert (CharField) - poziom ostrzeżenia, zazwyczaj wyrażony w wybranym z kolorów: zielony, żółty, pomarańczowy, czerwony. Wyznaczony na podstawie PAGER earthquake impact scale
- type (CharField) - typ trzęsienia. Zazwyczaj jest to trzęsienie ziemi, zdarzają się jednak również inne przyczyny - np. trzęsienie spowodowane działaniami w kamieniołomach
- cdi (FloatField) - największa zanotowana intensywność wydarzenia, wyznaczona przez DYFI
- mmi (FloatField) - największa przewidywana intensywność wydarzenia, wyznaczona przez ShakeMap
- felt (IntegerField) - liczba zarejestrowanych zgłoszeń o trzęsieniu w systemie DYFI
- sig (IntegerField) - liczba oznaczająca jak bardzo znaczące było wydarzenie. Większa liczba oznacza większe znaczenie. Obliczone na podstawie m.in. magnitudy, MMI, felt i CDI.

Baza jest zasilana danymi z API USGS.

Prediction

Model `Prediction` kontrybuje rekordy do tabeli `Predictions`. W modelu znajdują się następujące pola:

- `name` (`CharField`) - nazwa modelu używanego do predykcji
- `last_update_timestamp` (`DateTimeField`) - data i czas ostatniej aktualizacji modelu
- `file` (`FilePathField`) - ścieżka do modelu
- `precision` (`FloatField`) - wartość metryki `precision` modelu
- `recall` (`FloatField`) - wartość metryki `recall` modelu
- `f1` (`FloatField`) - wartość metryki `f1` modelu
- `accuracy` (`FloatField`) - wartość metryki `accuracy` modelu

Model musi znajdować się w katalogu `/assets/models`. Nie jest zapewniona dodatkowa walidacja wartości pola, dlatego należy uważać, aby nie usunąć plików, które odpowiadają modelom zapisanym w bazie. Sugerowanym formatem przechowywania modelu jest folder. Skrypt zasilający tabelę z modelami traktuje wszystkie foldery znajdujące się bezpośrednio w `/assets/models` jako foldery modeli i zapisuje je w bazie. Modele zapisane w inny sposób muszą być ręcznie wgrane do bazy za pośrednictwem panelu administratora.

6.6.3. API

W ramach aplikacji powstało również REST API na bazie Django REST framework[27]. Przygotowane API obsługuje dwa endpointy. Pierwszy z nich daje dostęp do danych o trzęsieniach przechowywanych w lokalnej bazie danych, a drugi umożliwia wykonanie predykcji.

Dostęp do danych

Endpoint jest dostępny pod adresem `/api/eqs` i został stworzony tak, aby jego interfejs był podobny do interfejsu API USGS[23]. W odpowiedzi na `GET request` zwracany jest `GeoJSON` z danymi. Przyjmowane dodatkowe parametry są opisane w tabeli 6.3.

Wykonanie predykcji

Endpoint dostępny jest pod adresem `/api/predict`. W odpowiedzi na `GET request` zwracana jest liczba oznaczająca prawdopodobieństwo wystąpienia trzęsienia dla podanych współrzędnych. Przyjmowane parametry opisane są w tabeli 6.4.

6.6. DOKUMENTACJA TECHNICZNA

Parametr	Opis
<code>minlatitude</code>	określa minimalną szerokość geograficzną
<code>maxlatitude</code>	określa maksymalną szerokość geograficzną
<code>minlongitude</code>	określa minimalną długość geograficzną
<code>maxlongitude</code>	określa maksymalną długość geograficzną
<code>mindepth</code>	określa minimalną głębokość wstrząsu
<code>maxdepth</code>	określa maksymalną głębokość wstrząsu
<code>starttime</code>	określa najwcześniejszą datę wstrząsu
<code>endtime</code>	określa najpóźniejszą datę wstrząsu
<code>minmagnitude</code>	określa minimalną magnitudę
<code>maxmagnitude</code>	określa maksymalną magnitudę
<code>limcount</code>	określa maksymalną liczbę zwracanych rekordów
<code>count</code>	zwraca słownik z kluczem <code>count</code> i liczbą wstrząsów spełniającą podane filtry

Tabela 6.3: Parametry endpointu `/api/eqs`

Parametr	Opis
<code>x</code>	długość geograficzna dla której zostanie wykonana predykcja
<code>y</code>	szerokość geograficzna dla której zostanie wykonana predykcja
<code>data</code>	'local' lub 'usgs', określa źródło wejścia predykcji
<code>model</code>	id modelu, który wykona predykcję

Tabela 6.4: Parametry endpointu `/api/predict`

Jeśli któryś z parametrów `x`. `y` nie zostanie podany, API zwróci kod `http 400`. Jeśli podane koordynaty nie będą należały do regionów, dla których możliwa jest predykcja, również zostanie zwrócony kod `http 400`.

Ponadto, do wykonania predykcji potrzebne są przynajmniej 64 wstrząsy występujące w danym regionie lub regionach sąsiadujących. Dodatkowo wymagane jest wystąpienie przynajmniej jednego wstrząsu w przeciągu ostatnich 30 dni. Jeśli te warunki nie zostaną spełnione, API zwróci kod `http 206` i informację o niewystarczających danych do wykonania predykcji.

7. Podsumowanie

Główym celem pracy było stworzenie modelu, który dostarczałby prognoz dotyczących trzęsień ziemi. Cel ten został spełniony, przygotowane modele osiągają zadowalające metryki **accuracy**, **precision**, **recall** i **F1**, zważając na trudność postawionego problemu. Dodatkowo przygotowana została aplikacja pozwalająca wykorzystać stworzone modele w praktyce. Ponadto zostały spełnione wszelkie wymagania uzgodnione na początku projektu. Wykorzystane metody opierają się na technologiach takich jak frameworki Django do budowy backendu stron internetowych i Tensorflow do tworzenia modeli. Na szczególną uwagę zasługuje wykorzystanie głębokich sieci neuronowych typu transformer oraz rekurencyjnych sieci neuronowych.

Możliwości modeli są jednak poważnie ograniczone przez dostępne dane. Do wykonania rozsądnej predykcji potrzebne są dane o przynajmniej jednym wstrząsie z ostatniego miesiąca i obszerne historyczne dane z analizowanego regionu.

Naturalnym kierunkiem rozwoju pracy jest próba skalowania modelu i danych. Model o analogicznej strukturze, ale większej liczbie parametrów, uczony na większym zbiorze danych i uwzględniający dodatkowe informacje eksperckie mógłby potencjalnie osiągać lepsze wyniki. Ważne jest też zwiększenie dokładności modelu przez zmniejszenie rozmiaru okna przestrzennego i czasowego wejścia i wyjścia. Można także rozwiązywać problem ze zmodyfikowaną definicją - na przykład rozważyć przewidywanie trzęsień jako regresję, a nie klasyfikację.

Bibliografia

- [1] U.S. Geological Survey. <https://www.usgs.gov/>. Accesed: 2023-12-18.
- [2] Quentin Bletery i Jean-Mathieu Nocquet. “The precursory phase of large earthquakes”. W: *Science* 381.6655 (2023), s. 297–301. DOI: 10.1126/science.adg2565. eprint: <https://www.science.org/doi/pdf/10.1126/science.adg2565%7D>. URL: <https://www.science.org/doi/abs/10.1126/science.adg2565%7D>.
- [3] K. Budiman i Y. N. Ifrizi. “Analysis of earthquake forecasting using random forest”. W: *Journal of Soft Computing Exploration* 2.2 (2021), s. 153–162.
- [4] Ashish Vaswani i in. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].
- [5] Guido Van Rossum i Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.
- [6] Wes McKinney i in. “Data structures for statistical computing in python”. W: *Proceedings of the 9th Python in Science Conference*. T. 445. Austin, TX. 2010, s. 51–56.
- [7] J. D. Hunter. “Matplotlib: A 2D graphics environment”. W: *Computing in Science & Engineering* 9.3 (2007), s. 90–95. DOI: 10.1109/MCSE.2007.55.
- [8] *Tectonic Plate Boundaries*. <https://www.kaggle.com/datasets/cwthompson/tectonic-plate-boundaries>. Accesed: 2024-01-28.
- [9] Peter Bird. “An updated digital model of plate boundaries”. W: *Geochemistry, Geophysics, Geosystems* 4.3 (2003). DOI: <https://doi.org/10.1029/2001GC000252>. eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2001GC000252>. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2001GC000252>.
- [10] Donald F. Argus, Richard G. Gordon i Charles DeMets. “Geologically current motion of 56 plates relative to the no-net-rotation reference frame”. W: *Geochemistry, Geophysics, Geosystems* 12.11 (2011). DOI: <https://doi.org/10.1029/2011GC003751>. eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2011GC003751>. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011GC003751>.

- [11] Wikipedia. *Haversine formula — Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Haversine%20formula&oldid=1189679575>. [On-line; accessed 28-January-2024]. 2024.
- [12] *Earthquake Measurements: Magnitude vs Intensity*. <https://www.earthquakeauthority.com/blog/2020/earthquake-measurements-magnitude-vs-intensity>. Accesed: 2024-01-28.
- [13] F. Pedregosa i in. “Scikit-learn: Machine Learning in Python”. W: *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.
- [14] Charles R. Harris i in. “Array programming with NumPy”. W: *Nature* 585.7825 (wrz. 2020), s. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [15] Martín Abadi i in. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [16] Anishnama. *Understanding LSTM: Architecture, Pros and Cons, and Implementation*. <https://medium.com/@anishnama20/understanding-lstm-architecture-pros-and-cons-and-implementation-3e0cca194094>. Accesed: 2024-01-28.
- [17] Sepp Hochreiter i Jürgen Schmidhuber. “Long short-term memory”. W: *Neural computation* 9.8 (1997), s. 1735–1780.
- [18] *Tensorflow - Classification on imbalanced data*. https://www.tensorflow.org/tutorials/structured_data/imbalanced_data. Accesed: 2024-01-28.
- [19] Django Software Foundation. *Django*. Wer. 4.2.9. 2 sty. 2024. URL: <https://djangoproject.com>.
- [20] *Bootstrap*. <https://getbootstrap.com/>. Accesed: 2024-01-28.
- [21] *Font Awesome*. <https://fontawesome.com/>. Accesed: 2024-01-28.
- [22] Volodymyr Agafonkin. *Leaflet*. <https://leafletjs.com/>. Accesed: 2024-01-28.
- [23] *USGS API Documentation*. https://earthquake.usgs.gov/fdsnws/event/1/?fbclid=IwAR1RyU_S6vga0iEkr7kODWDLHQdITIWk56P6osJKSXAC2B1mBXqvIlZva_8. Accesed: 2024-01-28.
- [24] The PostgreSQL Global Development Group. *PostgreSQL*. URL: <https://www.postgresql.org/>.
- [25] PostGIS Project Steering Committee i in. *PostGIS, spatial and geographic objects for PostgreSQL*. URL: <https://postgis.net/>.

- [26] *List of Countries by Continent.* <https://worldpopulationreview.com/country-rankings/list-of-countries-by-continent>. Accesed: 2024-01-28.
- [27] *Django REST framework.* <https://www.django-rest-framework.org/>. Accesed: 2024-01-28.

Wykaz symboli i skrótów

USGS	United States Geological Survey
GPS	Global Positioning System
API	Application Programming Interface
CSV	Comma-Separated Values
TF	Tensorflow
LSTM	Long Short-Term Memory
MLP	MultiLayer Perceptron
NLP	Natural language processing
ADAM	Adaptive Moment Estimation
CSS	Cascading Style Sheets
JS	JavaScript
JSON	JavaScript Object Notation
GeoJSON	Geographical JSON
SQL	Structured Query Language
GIS	Geographic Information System
DF	Data Frame
PAGER	Prompt Assessment of Global Earthquakes for Response
DYFI	Did You Feel It
REST API	Representational State Transfer Application Programming Interface
HTTP	Hypertext Transfer Protocol

Spis rysunków

3.1	Główne elementy architektury	19
4.1	Rozkład trzęsień na mapie	25
4.2	Średnia liczba trzęsień na rok	25
4.3	Średnia liczba trzęsień w regionach na rok	26
4.4	Liczba regionów z trzęsieniami na rok	26
4.5	Histogram skali trzęsień	27
4.6	Średnia skala trzęsienia na rok	27
4.7	Średnia skala trzęsień w regionie do liczby trzęsień w regionie	28
4.8	Granice płyt tektonicznych na mapie	29
4.9	Przykłady rodzajów punktów	30
4.10	Rozkład zmiennej Magnitude	35
4.11	Rozkład zmiennej Magnitude po przycięciu i przeskalowaniu	35
4.12	Rozkład zmiennej MagType po zmapowaniu	36
4.13	Rozkład zmiennej Longitude	37
4.14	Rozkład zmiennej Latitude	37
4.15	Rozkład zmiennej Depth	38
4.16	Rozkłady zmiennej Depth po przesunięciu i zlogarytmowaniu	39
4.17	Rozkład zmiennej Depth po zlogarytmowaniu i przycięciu	39
4.18	Rozkład zmiennej Dist	40
4.19	Rozkłady zmiennej Dist po przesunięciu i zlogarytmowaniu	40
4.20	Rozkład zmiennej Dist po zlogarytmowaniu i przycięciu	41
4.21	Rozkład zmiennej Plate	41
4.22	Rozkład zmiennej Plate po zmapowaniu	42
4.23	Rozkład zmiennej Dist_region	43
4.24	Rozkłady zmiennej Dist_region po przesunięciu i zlogarytmowaniu	43
4.25	Rozkład zmiennej Dist_region po zlogarytmowaniu i przycięciu	44
4.26	Rozkład zmiennej Plate_region	44

SPIS RYSUNKÓW

4.27 Rozkład zmiennej Plate_region po zmapowaniu	45
4.28 Rozkład zmiennej Lon_center	46
4.29 Rozkład zmiennej Lat_center	46
5.1 Liczba regionów o danej liczbie trzęsień	48
5.2 Liczba regionów o danej liczbie trzęsień bez 10 największych wartości	49
5.3 Liczba oraz procent regionów do threshold	49
5.4 Liczba oraz procent trzęsień do threshold	50
5.5 Rozkład wybranych regionów na mapie	50
5.6 Okrąg o promieniu 300 kilometrów dla przykładowego regionu	51
5.7 Rozkład zmiennej Diff_days	53
5.8 Rozkład zmiennej Diff_days bez 5 największych wartości	53
5.9 Rozkład zmiennej Diff_days po zmapowaniu	54
5.10 Rozkład zmiennej Distance	54
5.11 Rozkład zmiennej Distance_1	58
5.12 Schemat działania sieci rekurencyjnej. Źródło: Standford.edu	62
5.13 Schemat działania sieci rekurencyjnej dla jednego kroku w czasie. Źródło: Standford.edu	63
5.14 Schemat działania sieci LSTM dla jednego kroku w czasie. Źródło: oreilly.com . .	63
5.15 Architektura transformera. Źródło: [4]	66
5.16 Architektura sieci rekurencyjnej	67
5.17 Architektura transformera	68
5.18 Architektura transformera z rozszerzonym widokiem dekodera	69
5.19 Architektura pozycyjnej wektoryzacji trzęsień	70
5.20 Architektura pojedynczego bloku w dekoderze	70
5.21 Architektura transformera	72
5.22 Architektura transformera z rozszerzonym widokiem dekodera	72
5.23 Architektura kodera	73
5.24 Architektura pojedynczego bloku w koderze	74
5.25 Architektura pojedynczego bloku w dekoderze	75
5.26 Zmiana learning rate podczas trenowania	78
5.27 Zmiana loss podczas trenowania	80
5.28 Zmiana accuracy podczas trenowania	81
5.29 Zmiana precision podczas trenowania	82
5.30 Zmiana recall podczas trenowania	83

5.31 Zmiana F1 podczas trenowania	84
5.32 Zmiana loss podczas trenowania	86
5.33 Zmiana accuracy podczas trenowania	87
5.34 Zmiana precision podczas trenowania	88
5.35 Zmiana recall podczas trenowania	89
5.36 Zmiana F1 podczas trenowania	90
 6.1 Lista trzęsień ziemi	96
6.2 Szczegóły trzęsienia ziemi	97
6.3 Zakładki - ustawienia i filtry	97
6.4 Zakładka ustawień	98
6.5 Regiony predykcji	98
6.6 Wykonanie predykcji	99
6.7 Granice płyt tektonicznych	100
6.8 Zakładka filtrów	100
6.9 Logowanie	101
6.10 Strona główna panelu administratora z wyróżnioną sekcją MAP	101
6.11 Szczegóły tabeli	102
6.12 Usuwanie rekordów z tabeli	102
6.13 Szczegóły rekordu w tabeli Earthquake	103
6.14 Testy mapy	104
6.15 Test przemieszczania po krajach	104
6.16 Lista trzęsień ziemi	105
6.17 Lista trzęsień ziemi z widoku administratora	105
6.18 Testy API	106

Spis tabel

1.1	Podział pracy - tekst pracy	12
1.2	Podział pracy	13
2.1	Lista nie-funkcjonalnych wymagań	16
2.2	SWOT	16
3.1	Moduły	20
4.1	Przykładowy widok tabeli z danymi	23
4.2	Przykładowy widok pogrupowanych danych	24
4.3	Widok tabeli z granicami płyt tektonicznych	29
4.4	Widok tabeli po dodaniu informacji o płytach tektonicznych do trzęsień ziemi	31
4.5	Widok tabeli po dodaniu informacji o płytach tektonicznych do poszczególnych regionów	32
4.6	Widok tabeli po dodaniu etykiet	33
5.1	Przykładowa tabela zawierająca posortowane trzęsienia	52
5.2	Tabela zawierająca szereg czasowy o długości 2 dla jednego z regionów	56
5.3	Opis kolumn Tabeli 5.2	57
5.4	Przykładowa tabela zawierająca szereg czasowy o długości 2	60
5.5	Przykładowa tabela zawierająca informacje regionalne	60
5.6	Przykładowa tabela zawierająca etykiety	60
5.7	Tabela z liczbą parametrów dla wszystkich modeli	76
5.8	Podział modeli pod względem trenowania	77
5.9	Trenowanie modeli	77
5.10	Metryki	79
5.11	Metryki modelów, trenowanych bez parametru <i>class_weights</i> , na zbiorze testowym	85
5.12	Metryki modelów, trenowanych z parametrem <i>class_weights</i> , na zbiorze testowym	91
6.1	Dependencje	93

6.2	Domyślne parametry filtrowania	95
6.3	Parametry endpointu <code>/api/eqs</code>	109
6.4	Parametry endpointu <code>/api/predict</code>	109