AGH
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca magisterska

Wiktor Bieda

kierunek studiów: informatyka stosowana
specjalność: grafika komputerowa i przetwarzanie obrazów

Analiza atrybucji z wykorzystaniem technik uczenia głębokiego

Opiekun: **dr hab. inż. Szymon Łukasik, prof. AGH**

Kraków, listopad 2020

Oświadczenie studenta

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. — Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

.....

(czytelny podpis)

Kraków, listopad 2020

Tematyka pracy magisterskiej i praktyki dyplomowej Wiktora Bieda, studenta drugiego roku studiów drugiego stopnia na kierunku informatyka stosowana, specjalności grafika komputerowa i przetwarzanie obrazów.

Temat pracy magisterskiej: **Analiza atrybucji z wykorzystaniem technik uczenia głębokiego**

Opiekun pracy:	dr hab inż. Szymon Łukasik, prof. AGH
Recenzenci pracy:	
Miejsce praktyki dyplomowej:	WFiIS AGH, Kraków

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
   - zapoznanie się z ideą analizy atrybucji,
   - przygotowania oprogramowania,
   - dyskusja i analiza wyników,
   - sporządzenie sprawozdania z praktyki.
4. Kontynuacja obliczeń związanych z tematem pracy magisterskiej.
5. Zebranie i opracowanie wyników obliczeń.
6. Analiza wyników obliczeń numerycznych.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie:

....

(podpis kierownika katedry) (podpis opiekuna)

Ocena promotora

Ocena recenzenta

Serdeczne podziękowania dla dr hab. inż. Szymona Łukasika za wyrozumiałość oraz wsparcie podczas pisania niniejszej pracy. Jak również dla firmy Synerise za udostępnienie danych oraz pomoc w ich przetworzeniu.

Spis treści

1 Wstęp **9**

2 Wprowadzenie **10**
2.1 Czym jest analiza atrybucji? 10
2.2 Standardowe modele atrybucji 11
2.3 Pozyskiwanie danych 12

3 Preliminaria metodologiczne **13**
3.1 Głębokie uczenie 13
3.1.1 Czym jest głębokie uczenie? 13
3.1.2 Sztuczna sieć neuronowa 15
3.1.3 Propagacja sygnału wejściowego 17
3.1.4 Funkcja strat 19
3.1.5 Metoda gradientu prostego 19
3.2 Rekurencyjne sieci neuronowe 21
3.2.1 Czym jest LSTM? 22
3.3 Wykorzystanie głębokich sieci neuronowych do zagadnienia atrybucji 23
3.4 Wykorzystanie kart graficznych w procesie uczenia sieci 24

4 Część praktyczna **24**
4.1 Analiza danych 24
4.2 Struktura projektu 27
4.3 Przetwarzanie danych 29
4.3.1 Struktura do przechowywania danych w języku python 29
4.3.2 Elementy odstające 30
4.3.3 Przetworzenie ścieżek 31
4.3.4 Usunięcie wejść bezpośrednich 32
4.3.5 Utworzenie wektorów wejściowych 33
4.3.6 Podział danych 36
4.4 Ocena skuteczności modeli 36

5 Wyniki **37**
5.1 Modele First Touch i Last Touch 37
5.2 Model Dense 42
5.3 Model LSTM 46
5.4 Porównanie modeli 50

6 Podsumowanie **50**
6.1 Osiągnięte cele 50

6.2 Napotkane problemy 50
6.3 Możliwy rozwój 51
7 Bibliografia **52**

1 Wstęp

W dzisiejszym świecie jesteśmy ze wszystkich stron otoczeni reklamami, lecz rzadko zastanawiamy się o tym, co decyduje jakie reklamy są nam wyświetlane. Jak nietrudno się domyślić to jaka reklama nam się pokaże, zależy od reklamodawcy, czyli osoby która płaci poszczególnym podmiotom za możliwość wyświetlenia nam swoich reklam. Oczywistym faktem jest że reklamodawca wyświetla nam reklamę w jakimś konkretnym celu, najczęściej tym celem jest zachęcenie nas do zakupu poszczególnego produktu. Lecz co decyduje o tym, jakimi kanałami są nam wyświetlane reklamy? Aby lepiej przedstawić problem wyobraźmy sobie następującą sytuację: otwieramy własną firmę dostarczającą jakiś produkt, ale początkowo nie cieszy się on zbyt dużym zainteresowaniem i chcąc to zmienić postanawiamy zainwestować pieniądze w reklamy. Decydujemy się na inwestycje w dwa kanały: pierwszy - wyświetlanie reklam użytkownikom na portalu społecznościowym Facebook, drugi - wyświetlanie reklam użytkownikom w wyszukiwarce internetowej Google. Załóżmy, że na oba kanały przeznaczyliśmy taką samą ilość środków. Po pewnym czasie okazuje się, że wyświetlanie reklam przyniosło skutki i sprzedaż naszego produktu znacząco się zwiększyła, a co za tym idzie zwiększyły się również nasze przychody. I w tym miejscu pojawia się pytanie, czy podział budżetu na reklamę w formie 50% Facebook, 50% Google, był optymalny? Czy może jeśli byśmy zainwestowaliśmy więcej środków w jeden z kanałów, a mniej w drugi, to zainteresowanie naszym produktem wzrosło by jeszcze bardziej, a co za tym idzie odpowiednio zwiększyłyby się nasze przychody. W prawdziwym życiu mamy najczęściej do czynienia z większą ilością kanałów możliwych do wykorzystania, co sprawia że problem ten staje się nietrywialny.

Użytkownikowi, przed dokonaniem przez niego zakupu najczęściej wyświetlana jest nie pojedyncza reklama, ale ich większa liczba. Zbiór tych reklam w odpowiedniej kolejności, zgodnie z kolejnością ich wyświetlania nazywany jest w dalszej części pracy **ścieżką**. Celem pracy jest opracowanie algorytmów z wykorzystaniem technik uczenia głębokiego do analizy atrybucji, czyli wpływu poszczególnych pozycji ze ścieżki, na decyzję użytkownika o zakupie

produktu. Oprócz stworzenia i przetestowania odpowiednich rozwiązań wykorzystujących uczenie głęboki podjęto się również porównania tych podejść z istniejącymi klasycznymi technikami wykorzystywanymi w tym zagadnieniu, a mianowicie technice First Touch i technice Last Touch. Pojęcia te zostaną szczegółowo opisane w kolejnych rozdziałach.

Struktura pracy prezentuje się następująco: w kolejnym rozdziale "Wprowadzenie" omówiono istniejące już rozwiązania dla problemu atrybucji. W rozdziale "Preliminaria metodologiczne" omówiono kilka kwestii teoretycznych związanych z uczeniem maszynowym. Następnym rozdziałem jest "Część praktyczna" w której opisano po kolei: przygotowanie danych, trenowanie modeli, zaprezentowanie wyników. Pracę podsumowuje rozdział w którym odniesiono się do osiągniętych celów, napotkanych problemów, oraz możliwości dalszego rozwoju pracy.

2 Wprowadzenie

2.1 Czym jest analiza atrybucji?

Należy zacząć od wyjaśnienia czym jest atrybucja. Atrybucja to zgodnie z definicją "przypisywanie komuś lub czemuś pewnych cech" [1]. Czyli odnosząc się do tematu pracy analiza atrybucji, będzie równoznaczna z analizą wpływu poszczególnych interakcji w ramach kanałów reklamowych (które są możliwe do wykorzystania w promocji produktu), na dokonanie przez użytkownika transakcji.

Rysunek 1. Przykładowa ścieżka użytkownika.

W celu lepszego zilustrowania problemu posłużono się powyższym rysunkiem. Jest na nim przedstawiona przykładowa ścieżka, która doprowadziła do zakupu produktu przez użytkownika. Ścieżka jest to zbiór interakcji reklamowych, którym został poddany dany użytkownik. Każdy użytkownik posiada własną ścieżkę. Prezentowana ścieżka składa się z 3 interakcji, które zostały przeprowadzone w różnych odstępach czasowych i prawdopodobnie w pewnym stopniu zadecydowały o zakupie. Lecz jak stwierdzić, która z interakcji wpłynęła na zakup w największym stopniu? Obecnie najczęściej wykorzystywanymi technikami są metody Pierwszego Kontaktu (ang. First Touch) i Ostatniego Kontaktu (ang. Last Touch), które są omówione w kolejnym rozdziale. [2]

2.2 Standardowe modele atrybucji

Jak zostało wspomniane w poprzednim podrozdziale najpopularniejszymi metodami analizy atrybucji są: First Touch i Last Touch. Charakteryzują się one tym, że całą zasługę za dokonanie transakcji przypisują tylko pojedynczej akcji ze ścieżki. Czyli odpowiednio dla First Touch jest to pierwsza interakcja z użytkownikiem, dla Last Touch ostatnia. Odnosząc się do ilustracji z poprzedniego rozdziału metoda First Touch przypisuje całkowitą zasługę za

dokonanie transakcji wysłaniu wiadomości email do użytkownika, kompletnie pomijając kolejne interakcje z użytkownikiem, takie jak wyświetlenie reklamy w wyszukiwarce google, oraz wyświetleniu reklamy na portalu społecznościowym Facebook. Jak można się domyślić metoda Last Touch zadziała w analogiczny sposób, z tą różnicą że przypisuje całą zasługę ostatniej interakcji (wyświetleniu reklamy na portalu społecznościowym Facebook) a pomija wszystkie poprzednie.

Podejścia te zawdzięczają swoją popularność głównie dwóm powodom. Pierwszym jest trywialna implementacja tych rozwiązań. Drugim natomiast jest łatwość w zrozumieniu i interpretacji. Metoda First Touch ma jeszcze tą dodatkową zaletę, szczególnie przydatną dla rozwijających się marek, że pozwala określić którym kanałem są pozyskiwani nowi klienci. Oczywiście podejścia te nie są idealne. Główną wadą opisywanych modeli jest to, że ignorują znaczną część danych, która w niektórych przypadkach może okazać się kluczowa dla podjęcia decyzji o dokonaniu transakcji przez użytkownika. [2]

Modelami używanymi do analizy, które wykorzystują wszystkie dane ze ścieżki są:

- model liniowy, który przypisuje identyczne wagi wszystkim akcjom ze ścieżki,
- model o rozkładzie czasowym, który zwiększa wagę wraz z kolejnymi elementami ze ścieżki (im później nastąpiła dana interakcja, tym większa waga jest jest przypisywana),
- model uwzględniający pozycje, przypisuje on największą wagę do pierwszej i ostatniej interakcji ze ścieżki.

Ponadto istnieją też inne rozwiązania wykorzystujące wszystkie dane ze ścieżki oparte o uczenie głębokie.

2.3 Pozyskiwanie danych

W poprzednich rozdziałach często używano sformułowania ścieżka użytkownika, odnoszącego się do wszystkich reklam jakie zostały wyświetlone danemu użytkownikowi lecz nie wyjaśniono jak utworzyć takie ścieżki. Odpowiadają za to kody UTM (ang. Urchin tracking module). Urchin Software Corporation była to firma, która została przejęta przez Google w 2005 roku, a jej oprogramowanie stanowiło podstawy narzędzia Google Analytics - zdecydowanie najpopularniejszego narzędzia służącego do zarządzania reklamami, wykorzystywanego przez

ponad 80% stron [3]. Kod UTM jest tekstem, który można dodać do adresu internetowego po znaku zapytania na przykład *https://www.agh.edu.pl?utm_source=facebook&utm_medium=cpc* i który pozwala nam monitorować w jaki sposób ruch trafił na naszą stronę. Czyli dla przykładu reklamy wyświetlane na portalu Facebook będą kierować do naszej strony przez adres z *utm_source=facebook* w środku, a reklamy wyświetlane na Google będą miały ustawiony odpowiednio parametr *utm_source=google*. [4] Ponadto oprócz parametru *utm_source* możliwy jest także do ustawienia parametr *utm_medium* mogący być bardziej ogólną wartością, na przykład email, co odpowiada kategori wejść na stronę przez kliknięcie linku w mailu. Przekładając to na prostszy język *utm_source* można traktować jako "kto" sprowadził ruch na naszą stronę, a *utm_source* jako "jak" to się stało, przez jaki kanał. [5]

Niestety nie zawsze jest możliwość stwierdzenia, co sprawiło że użytkownik trafił na naszą stronę. Na przykład jeśli użytkownik wpisze ręcznie adres naszej strony, to wejście takie zostanie potraktowane jako wejście bezpośrednie (ang. direct). Tak samo jeśli użytkownik zapisze sobie naszą stronę w zakładce (ang. bookmark), też nie ma możliwości określić skąd się o niej dowiedział. [6]

3 Preliminaria metodologiczne

3.1 Głębokie uczenie

3.1.1 Czym jest głębokie uczenie?

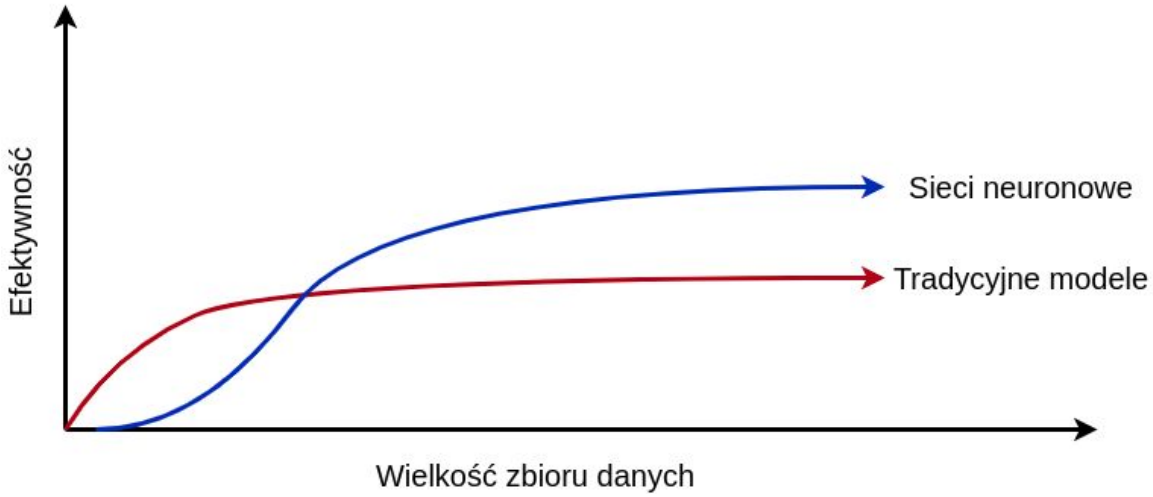
Głębokie uczenie (ang. Deep Learning) jest to pod-dziedzina uczenia maszynowego (ang. Machine Learning), które z kolei jest pod-dziedziną sztucznej inteligencji (ang Artificial Intelligence). Sztuczna inteligencja jest to ogólny termin odnoszący się do technik, które pozwalają programom komputerowym na naśladowanie zachowania ludzkiego. Uczenie maszynowe jest to zbiór algorytmów, które umożliwiają to właśnie naśladowanie. Podział ten zaprezentowano na poniższym rysunku.

Rysunek 2. Podział obszarów poszczególnych dziedzin sztucznej inteligencji.

Wzorem dla głębokiego uczenia była struktura ludzkiego mózgu. Tak jak mózg w ludzkim ciele po otrzymaniu jakiegoś bodźca analizuje go porównując ze znanymi mu już informacjami, tak samo sieć neuronowa może być wytrenowana do wykonywania różnych zadań. [7] Najpopularniejszymi zadaniami wykonywanymi przez sieć neuronową są:

- klasteryzacja , czyli podział elementów ze zbioru na klastry, czyli takie grupy, w obrębie których elementy są jak najbardziej podobne do siebie, jednocześnie będąc jak najbardziej różne od elementów z innych grup
- klasyfikacja, jest to przyporządkowanie na podstawie jakiś informacji, na przykład zdjęcia, odpowiedniej kategorii, na przykład obiektu jaki znajduje się na tym obrazku
- regresja, czyli próba predykcji jakiejś wartości liczbowej, na przykład próba przewidzenia ceny mieszkania, na podstawie jego wymiarów takich jak powierzchnia, dzielnica itd.

Dzięki swojej unikalności głębokie uczenie jest szeroko wykorzystywane w dzisiejszym świecie: samochody autonomiczne, chatboty, asystenci głosowi (Alexa, Siri), Google Translate, systemy rekomendacji filmów (Netflix, Youtube) i w wielu innych. [8]

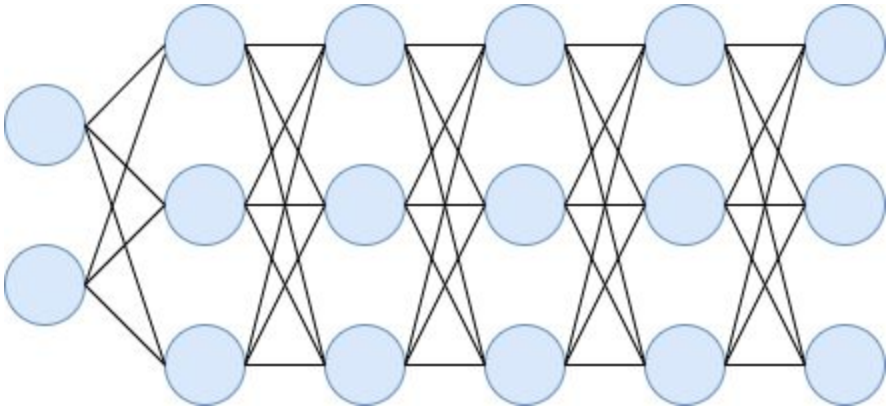


Rysunek 3. Efektywność poszczególnych technik w zależności od ilości danych. [8]

Jedną z głównych przyczyn, z powodu których uczenie maszynowe stało się tak popularne są możliwości technologiczne, które pozwalają w prosty sposób gromadzić ogromne ilości danych. Mając dostęp do tak dużych zbiorów z danymi sieć neuronowa jest w stanie rozpoznawać ogólne wzorce, co w znaczącym stopniu wpływa na jej skuteczność i przez co w większości przypadków staje się lepszym rozwiązaniem od tradycyjnych algorytmów, co zilustrowano na rysunku 3. Dzieje się tak ponieważ tradycyjne modele posiadają pewne ograniczenia w tym, z jakiej ilości danych są w sensowny sposób skorzystać, natomiast głębokie sieci neuronowe dzięki swojej strukturze, pozwalają na rozpoznanie większej ilości zależności w dużych zbiorach danych, co skutkuje zwiększoną skutecznością. [9]

3.1.2 Sztuczna sieć neuronowa

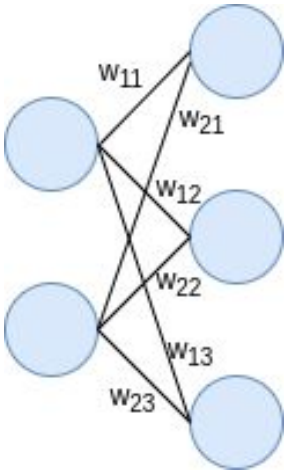
Sztuczne sieci neuronowe (ang. Artificial Neural Networks) składają się z neuronów, zwanych też węzłami, które są po prostu graficzną reprezentacją dla zwykłych wartości liczbowych. Połączenia między neuronami są nazywane wagami, i one też nie są niczym innym jak po prostu liczbami. [10] Gdy wprowadzimy do sieci jakieś zadanie, poprzez podanie danych wejściowych i danych wyjściowych można rozpocząć trening modelu. Polega on na takim dopasowaniu poszczególnych wag, aby sieć jak najlepiej wykonywała powierzone jej zadanie. Wagi te będą różniły się dla różnych zbiorów danych [8]



Rysunek 4. Sztuczna sieć neuronowa.

Sieć neuronowa zazwyczaj składa się z kilku warstw. Pierwsza warstwa nazywana jest warstwą wejściową (ang. input layer), a ostatnia wyjściową. [10] Warstwa wejściowa otrzymuje dane wejściowe w postaci wektora **X**, na podstawie których się uczy. Liczba neuronów w warstwie wejściowej musi być taka sama jak w wektorze **X**, tak aby każdy neuron reprezentował jedną wartość z wektora **X**. Ostatnia warstwa w sieci jest nazywana warstwą wyjściową (ang. output layer). W zależności od tego do czego służy model, składa się z jednego, lub większej ilości neuronów. W celu uzyskania wartości **y**, sieć wykonuje określone działania matematyczne. Działania te wykonywane są w warstwach pomiędzy warstwą wejściową a warstwą wyjściową. Warstwy te nazywane są warstwami ukrytymi (ang. hidden layers). [8]

W celu lepszego zilustrowania działania sieci neuronowej, na poniższym rysunku przedstawiono małą sieć składającą się tylko z dwóch warstw. Wejściowej, która posiada 2 neurony, i wyjściowej posiadającej 3 neurony.



Rysunek 5. Połączenia pomiędzy warstwami.

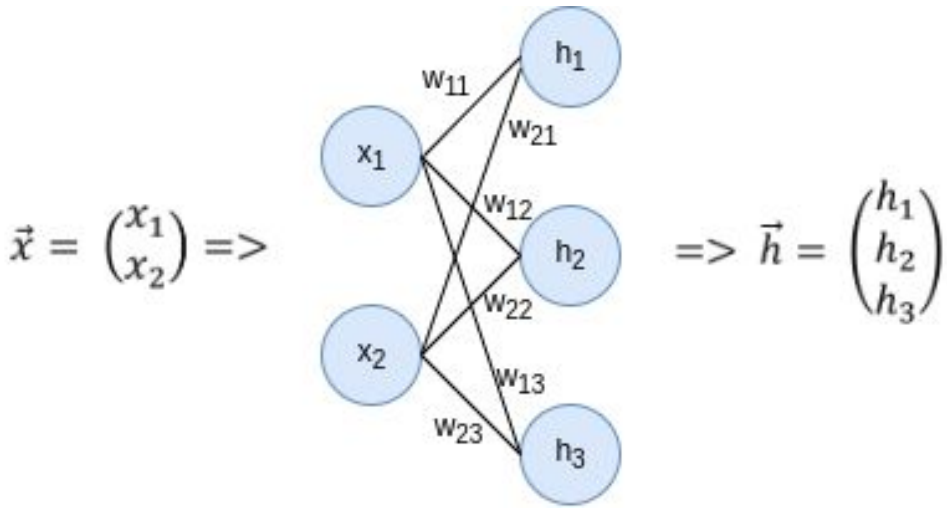
Jak można zauważyć na rysunku 5, każde połączenie między neuronami ma przyporządkowaną wagę **w**. Wagi te posiadają indeksy, pierwszą wartością indeksu jest numer neuronu warstwy z której połączenie wychodzi, a drugą wartością indeksu, jest numer neuronu w warstwie do której prowadzi połączenie. Zbiór wszystkich wag, można przedstawić w postaci macierzy. Macierz taką określa się symbolem **W** i nazywa się macierzą wag (ang. weight matrix). Prezentuje się ona następująco:

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

Liczba wierszy w macierzy **W** odpowiada liczbie neuronów warstwy z której wychodzą połączenia, natomiast liczba kolumn jest równa liczbie neuronów w warstwie do której prowadzą połączenia. [8]

3.1.3 Propagacja sygnału wejściowego

Dla określonych danych wejściowych **x**, sieć neuronowa oblicza wartości wektora **h**, co zilustrowano na poniższej ilustracji.



Rysunek 6. Propagacja.

Wartości wektora **h** obliczane są poprzez wymnożenie transponowanej macierzy **X** oraz macierzy z wagami **W**. Przedstawienie tych obliczeń w postaci równań prezentuje się następująco:

$$\vec{x}^T * W = (x_1, \quad x_2)\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$= (x_1 w_{11} + x_2 w_{21}, \quad x_1 w_{12} + x_2 w_{22}, \quad x_1 w_{13} + x_2 w_{23})$$

$$= (z_1 \quad z_2 \quad z_3) = \vec{z}$$

$$\vec{h} = \sigma(\vec{z})$$

Wektor **h** jest otrzymywany poprzez użycie funkcji aktywacji (ang. Activation Function) na wektorze **z**. Funkcja ta przedstawiona jest w powyższym wzorze znakiem sigmy a jej zadaniem jest umożliwienie sieci nauczenie się zależności nieliniowych. Istnieje wiele typów funkcji aktywacyjnych, między innymi: sigmoid, ReLU, Tah, Softmax. Bez użycia funkcji aktywacyjnej sieć neuronowa działałaby w ten sam sposób co zwykły model regresji liniowej.

Jeśli sieć składa się z większej ilości warstw, to dla kolejnych warstw postępujemy analogicznie, z tym że wektor **h** z wcześniejszej warstwy, będzie mnożony przez wektor wag **W** w celu obliczenia nowego wektora. Ostatni wektor jest nazywany wektorem wyjściowym **y**. [11]

3.1.4 Funkcja strat

Po otrzymaniu wektora wyjściowego **y** porównywany jest on z wektorem z wartościami oczekiwanymi **ŷ**. W celu określenia jak predykowane przez model wartości **y** są bliskie od wartości rzeczywistych **ŷ,** wykorzystywana jest funkcja strat (ang. Loss function). Jeśli przewidywania są złe funkcja strat funkcja strat zwraca większą wartość. Jeśli przewidywania są dobre, funkcja ta zwraca mniejsze wartości.

$$L(W) = -\sum_{i=0}^{N} \hat{y}_i * log(y_i) \tag{1}$$

$$L(W) = -\sum_{i=0}^{N} \hat{y}_i * log(y_i) \tag{2}$$

Powyżej przedstawiono wzory dla dwóch popularnych funkcji strat: średniej entropii krzyżowej (ang. Cross Entropy) (1) oraz średniego błędu kwadratowego (ang. Mean Squared Error) (2). Funkcje te dobrze spełniają swoją rolę, a do tego są proste do zaimplementowania. [12]

3.1.5 Metoda gradientu prostego

Metoda gradientu prostego (ang. Gradient Descent) polega na wykorzystywaniu gradientu w celu poprawy wartości wag sieci neuronowej. W celu przedstawienia działania najlepiej posłużyć przykładem najprostszej sieci neuronowej, przedstawionym na poniższym rysunku.

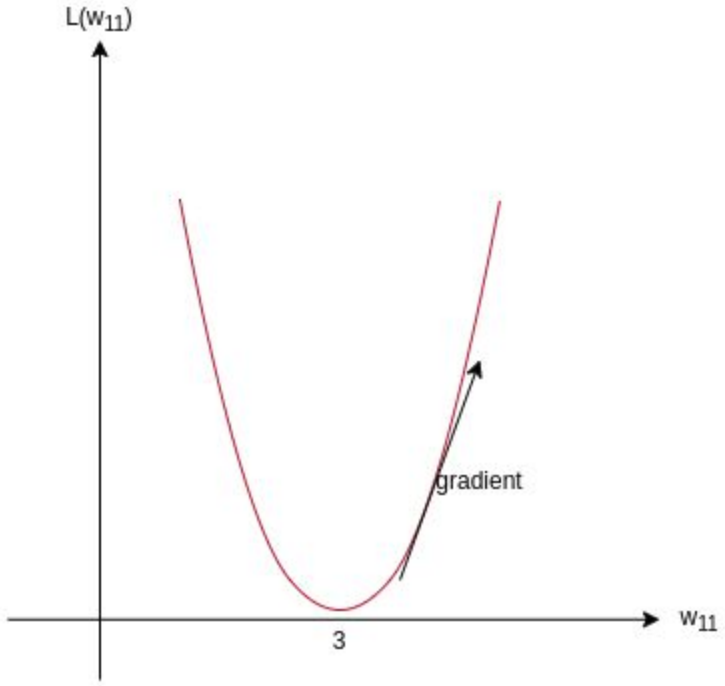
Rysunek 7. Prosta sieć neuronowa.

Sieć neuronowa otrzymuje dane wejściowe **x** i oczekiwane dane wyjściowe **ŷ**. Zadaniem sieci jest obliczenie wagi **w**. Załóżmy, że początkowa waga **w** wynosi 10, **x** wejściowe wynosi 4, a **ŷ** rzeczywiste wynosi 12. Wobec tego predykowane **y** można obliczyć następująco:

$$y = x * w = 4 * 10 = 20$$

$$\hat{y} = 12 \neq 20$$

Jak można łatwo zauważyć predykcja jest błędna. W celu znalezienia nowej wartości dla wagi wykorzystywany jest gradient funkcji strat.



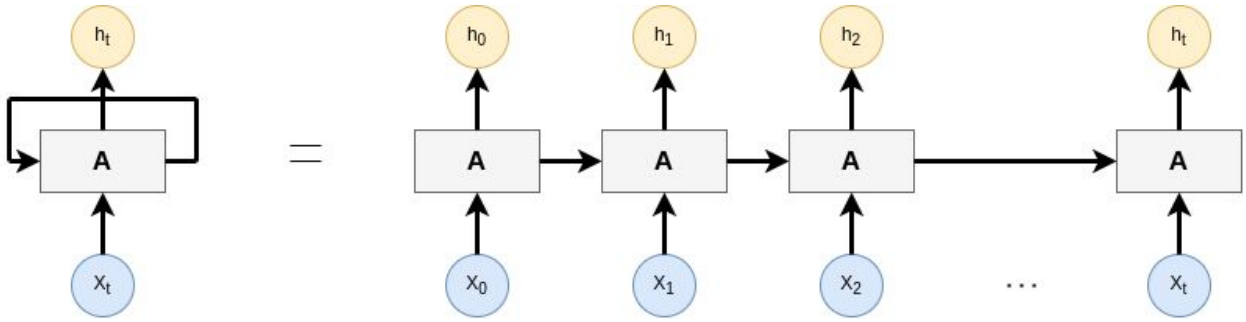
Rysunek 8. Kwadratowa funkcja strat.

Gradient są to pochodne cząstkowe funkcji strat. Nowe wagi są modyfikowane w kierunku przeciwnym do obliczonego gradientu, aż do osiągnięcia satysfakcjonującego rezultatu. Ponieważ omawiany przykład jest bardzo prosty i mamy do czynienia tylko z jedną wagą, łatwo to zilustrować, co przedstawiono na powyższym rysunku. Mimo tego, nawet jeśli ma się do czynienia z większą ilością parametrów zasada pozostaje ta sama. [12]

3.2 Rekurencyjne sieci neuronowe

Rekurencyjne sieci neuronowe (ang. Recurrent Neural Network) jest to termin określający sieci neuronowe, które posiadają możliwość przechowywania danych w wewnętrznym stanie (pamięci). W RNN rekurencja występuje, ponieważ ta sama funkcja jest wywoływana dla każdej danej wejściowej, podczas gdy dane wyjściowe w bieżącym kroku, zależą od poprzednich wywołań.

W przeciwieństwie do zwykłych sieci neuronowych, sieci rekurencyjne wykorzystują wewnętrzny stan, do przetwarzania sekwencji wejściowej. Dlatego też rekurencyjne sieci neuronowe wykorzystywane są do zadań w których istnieje zależność między kolejnymi danymi w sekwencji. Jako przykład można podać analizę tekstu. Poszczególne słowa w zdaniu zależą od siebie nawzajem.



Rysunek 9. Uproszczony model rekurencyjnej sieci neuronowej.

Na powyższym zdjęciu zilustrowano działanie rekurencji poprzez rozwinięcie tego co się dzieje w środku dla każdej sekwencji. Najpierw brana jest pierwsza wartość z sekwencji X(0) i na jej podstawie obliczane jest h_0, które wraz z kolejną wartością sekwencji X(1) są danymi wejściowymi w kolejnym kroku. Dzięki temu zabiegowi sieć neuronowa zapamiętuje kontekst podczas trenowania. Wzór dla bieżącego stanu wygląda następująco:

$$h_t = f(h_{t-1}, x_t) \tag{3}$$

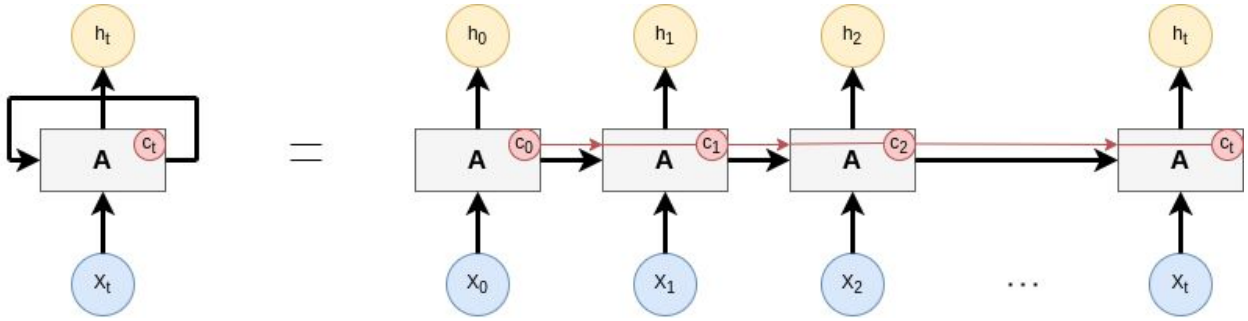
A przy wykorzystaniu funkcji aktywacyjnej, wzór przekształca się do postaci:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{4}$$

Zaletami używania rekurencyjnych sieci neuronowych jest to, że model można trenować na sekwencjach danych, przez co brana pod uwagę jest kolejność elementów w sekwencji. Wadą natomiast jest większe skomplikowania sieci, przez co trenowanie rekurencyjnej sieci neuronowej może być bardzo trudnym zadaniem. [13]

3.2.1 Czym jest LSTM?

Long Short-Term Memory (LSTM) jest to termin określający zmodyfikowaną wersję rekurencyjnych sieci neuronowych, którego zaletą jest możliwość zapamiętywania które były dużo wcześniej w sekwencji. W zwykłych sieciach rekurencyjnych im większa długość danych w sekwencji, tym gorzej sieć radzi sobie z zapamiętywaniem danych które były wcześniej. Zjawisko to nazywane jest "Problemem znikającego gradientu" (ang. Vanishing gradient problem). [14]



Rysunek 10. Uproszczony model sieci LSTM.

Kluczową różnicą między prostymi sieciami rekurencyjnymi a LSTM, jest to, że LSTM posiada dodatkowe połączenie, przedstawione na powyższym rysunku czerwoną linią. Czyli oprócz stanu **h** opisanego w poprzednim podrozdziale, LSTM posiada również dodatkowy stan

c, który przechowuje pewną informację, która odpowiednio regulowana w poszczególnych krokach przechodzi przez całą sekwencję. [15]

3.3 Wykorzystanie głębokich sieci neuronowych do zagadnienia atrybucji

Głębokie uczenie dzięki swojej strukturze, czyli warstwom składającym się z neuronów pozwala na odczytywanie z danych wzorców i zależności bliskich temu, co byłby w stanie odczytać człowiek. Dzięki temu liczne branże technologiczne mogą powiększyć swoje zyski, poprzez zaimplementowanie rozwiązań wykorzystujących uczenie głębokie. [16]

Celem pracy była analiza atrybucji, czyli analiza wpływu poszczególnych interakcji na dokonanie przez użytkownika transakcji. Zazwyczaj nie ma możliwości spytania użytkownika, która interakcji z nim, wpłynęła w jakim stopniu na dokonanie przez niego zakupu, przez co te informacje nie znajdują się w zbiorze danych. Zawsze natomiast wiadomo ile wyniosła transakcja, której dokonał użytkownik. Dlatego właśnie autor pracy zdecydował się wykorzystać sieci neuronowe, do przewidywania kwoty transakcji. Kierując się rozumowaniem, że jeśli głęboka sieć neuronowa, dobrze przewiduje kwoty transakcji, na podstawie tego jakim interakcją został poddany użytkownik, to znaczy że dobrze nauczyła się zależności między interakcjami. Dlatego w celu określenia atrybucji, postanowiono wykorzystać nauczoną głęboką sieć do tego jak będzie wyglądało przewidywanie ceny jeśli usunięte zostaną wszystkie wystąpienia danej interakcji. Ważnym jest pamiętać, że sieć uczona jest na wszystkich interakcjach, a dopiero do określenia atrybucji usuwane są poszczególne interakcje. I jeśli na przykład usunięcie wszystkich reklam wyświetlanych na Facebooku będzie skutkować znaczącym obniżeniem przewidywanej przez głęboką sieć neuronową kwoty transakcji, to znaczy że “reklama na Facebooku” ma duże znaczenie. Wpływy poszczególnych interakcji przeskalowano tak, aby ich suma wynosiła 100% i przedstawiono w rozdziale “ 5 Wyniki”.

3.4 Wykorzystanie kart graficznych w procesie uczenia sieci

Do trenowania modeli sieci neuronowych w większości przypadków wykorzystuje się karty graficzne (ang. Graphics processing unit) zamiast procesorów (ang. Central processing unit). Jest to spowodowane tym, że karty graficzne są dużo wydajniejsze w mnożeniu macierzy. A właśnie ta operacja jest najczęściej wykonywana przy trenowaniu modelu sieci neuronowej. Główną przyczyną takiego stanu rzeczy jest to, że procesory są zoptymalizowane pod względem czasu oczekiwania (ang. latency), a karty graficzne pod względem przepustowości pamięci (ang. Memory bandwidth). Najszybsze procesory posiadają przepustowość na poziomie 50GB/s, podczas gdy w najszybszych kartach graficznych wynosi ona 750GB/s. Więc jeśli obliczenia wymagają znacznej ilości pamięci, to mimo to, że procesor mam mniejszy czas oczekiwania, i szybciej pobiera kolejne pakiety danych, to i tak nie niweluje to tego, że karta graficzna potrafi pobrać na raz znacznie większy pakiet danych z pamięci RAM. Trafnym porównaniem jest określenie procesora jako Ferrari, a karty graficznej jako autobusu, danymi zaś niech będą pasażerowie. Jeśli naszym zadaniem jest przewieźć 100 pasażerów z punktu A, do punktu B, to pomimo tego, że Ferrari jest znacznie szybsze, to może ono zabrać maksymalnie 5 pasażerów, więc będzie musiało pokonać tę trasę aż 20 razy. Podczas gdy, mimo tego że autobus jest znacznie wolniejszy, to fakt, że jest w stanie pomieścić aż kilkudziesięciu pasażerów, powoduje że ogólny czas wszystkich przejazdów będzie znacznie krótszy w przypadku autobusu. [17]

4 Część praktyczna

4.1 Analiza danych

Dla celów realizacji pracy pozyskano przykładowy zbiór stanowiący bazę rzeczywistych ścieżek konwersji. Dane były dostępne w postaci CSV (ang. Comma-separated values), czyli w postaci pliku tekstowego, w którym kolejne wartości są oddzielone przecinkami. Każda linia, to nowy rekord z danymi. Cały plik *conversion_paths.csv* zajmuje 220MB i składa się z 459917 rekordów, a każdy z rekordów posiada 9 atrybutów.

	utm_source	utm_medium	days_till_conversions	revenue
0	['synerise','bing.com']	['e-mail','referral']	[13,13]	64.90
1	['facebook']	['cpc']	[18]	19.99
2	['(direct)']	['(none)']	[0]	30.75
3	['facebook']	['cpc']	[14]	25.99
4	['(direct)']	['(none)']	[0]	199.00

Rysunek 10. Tabela z informacjami.

W powyższej tabeli przedstawiono 5 losowych rekordów z danymi, wraz z kolumnami, które zostały wykorzystane w projekcie. Pierwsza kolumna to ***utm_source***, jak wspomniane zostało we wcześniejszych rozdziałach decyduje ona "kto" sprowadził ruch na naszą stronę. Kolejną kolumną jest ***utm_medium***, która zawiera ścieżki z informacjami "jak" ruch został przekierowany do strony. Kolejna kolumna ***days_till_conversions*** informuje o tym kiedy dane wejścia na stronę miały miejsce. Jej wartości są liczbami całkowitymi określającymi ile dni przed zakupem miało miejsce to wejście. Ostatnią kolumną jest ***revenue*** informująca o tym, jaką wartość miała transakcja dokonana przez użytkownika. Wszystkie przedstawione kolumny poza revenue, składają się z list, ponieważ każdy użytkownik może wejść na stronę kilkukrotnie przez różne kanały. Czyli odczytując z tabeli, użytkownik z id 0 odwiedził stronę dwukrotnie, raz przez link w emailu, drugi raz poprzez referral (pojęcie to zostanie objaśnione w dalszej części pracy). Oba wejścia miały miejsce na 13 dni przed dokonaniem przez niego transakcji o wartości 64,90.

Źródło oraz medium	**Liczba wystąpień par**
('(direct)', '(none)')	1896145
('(direct)', 'sms_text')	251772
('google', 'cpc')	132504
('facebook', 'cpc')	132045
('synerise', 'organic')	131781
('google', 'web_push')	131398

('bing.com', 'referral')	87186
('facebook.com', 'referral')	84155
('google', 'e-mail')	80622
('google.com', 'referral')	72756
('synerise', 'e-mail')	54295
('(direct)', 'cpc')	53591
('google', '(none)')	173
('synerise', '(none)')	78
('facebook', '(none)')	64
('(direct)', 'organic')	24
('(direct)', 'web_push')	22
('(direct)', 'e-mail')	16

Rysunek 11. Tabela z liczbą wystąpień dla poszczególnych par.

W celu analizy ścieżek dla poszczególnych został utworzony skrypt w języku python, który obliczył liczbę wystąpień poszczególnych par *utm_source* i *utm_medium*. Jak można zauważyć z powyższej tabeli bardzo dużo jest wejść bezpośrednich, które nie są zbyt przydatne w analizie i które zostały usunięte, o czym będzie mowa w kolejnych rozdziałach. Odczytując tabelę widać, że mamy do czynienia z pięcioma źródłami przez które ruch został sprowadzony do strony:

- Direct
- Google
- Facebook
- Bing
- Synerise

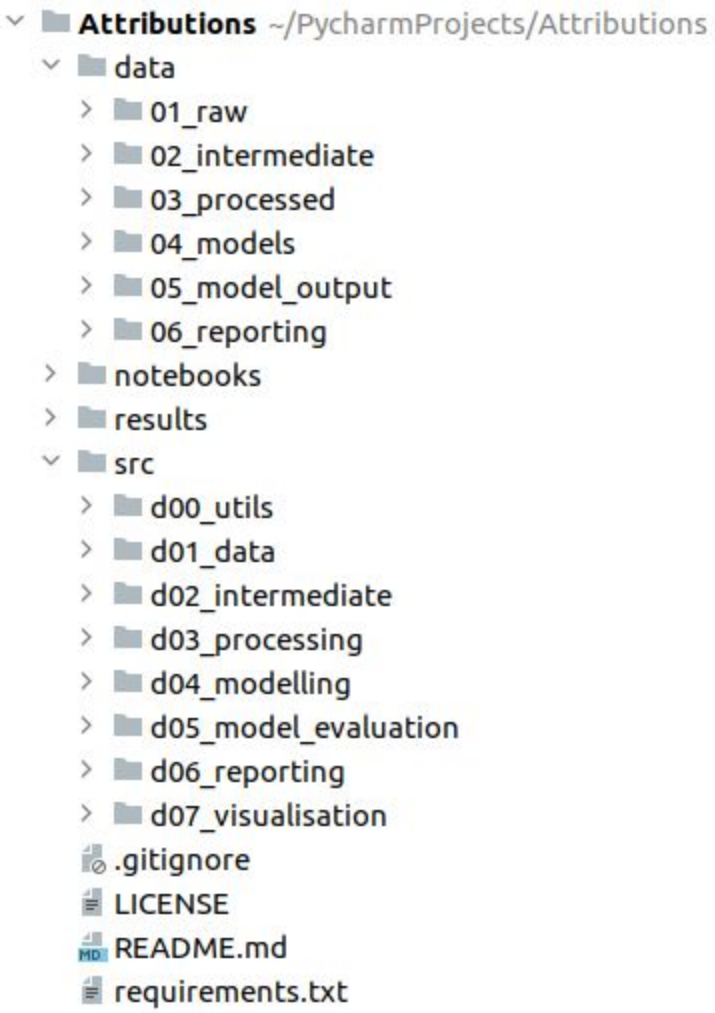
Oraz z następującymi mediami:

- none - są to wejścia bez przypisanego żadnego medium

- referral - wartość podstawowa, która jest automatycznie, gdy nic innego nie zostało ustawione.
- cpc - skrót dla kategorii ruchu płatnego
- email - ruch przez linki w mailach
- organic - ruch za który nie trzeba płacić, jeśli strona jest wartościowa, inne strony podają jej adres [18]
- web-push - notyfikacje

4.2 Struktura projektu

Częstym problemem występującym wśród osób zajmujących się analizą danych jest brak podstaw programistycznych odnośnie pisania czytelnego kodu. O ile nie ma to zbytniego wpływu na małe projekty, to w przypadku większych może się to wiązać z ogromnymi problemami w przyszłości, ponieważ wraz z nawarstwieniem się ilości kodu zmiana w jednym miejscu, może popsuć działanie programu w kilku innych. W celu zapobiegania tym problemom przy tworzeniu algorytmu opisywanego w tej pracy postanowiono wykorzystać zrozumiałą i przejrzystą strukturę projektu. [19]



Rysunek 12. Struktura projektu w programie PyCharm.

Struktura projektu prezentuje się w następujący sposób. Główny folder Attributions w którym znajduje się projekt jest podzielony na podfoldery. Opis, co znajduje się w poszczególnych folderach prezentuje się następująco

- data:
    - 01_raw - surowe dane w pliku o formacie csv
    - 02_intermediate - dane przekonwertowane do odpowiednich obiektów języka python, oczyszczone z wartości odstających i wejść bezpośrednich

    - 03_processed - przetworzenie ścieżek odpowiadających poszczególnym użytkownikom do postaci wektorów, które można wprowadzić jako dane wejściowe do sieci neuronowej
    - 04_models - pliki z wagami dla wyszkolonych modeli
    - 05_model_output - dane wyjściowe otrzymywane z modeli
    - 06_reporting - różnego rodzaju raporty
- notebooks - skrypty w postaci .ipynb, głównie służące do testowania małych zmian
- results - ostateczne dane wyjściowe wykorzystywane w pracy
- src:
    - d00_utils - funkcje wykorzystywane na przedziale całego projektu
    - d01_data - skrypty służące do wczytania danych
    - d02_intermediate - skrypty służące do przekonwertowania danych i oczyszczenia z niepożądanych wartości
    - d03_processing - skrypty służące do przetwarzania danych do postaci wymaganej jako dane wejściowe do modelu
    - d04_modelling - skrypty służące do wytrenowania modeli
    - d05_model_evaluation - skrypty służące do analizy skuteczności modeli
    - d06_reporting - skrtypty obliczające dane podsumowujące
    - d07_visualization - skrypty służące do wizualizacji danych

4.3 Przetwarzanie danych

4.3.1 Struktura do przechowywania danych w języku python

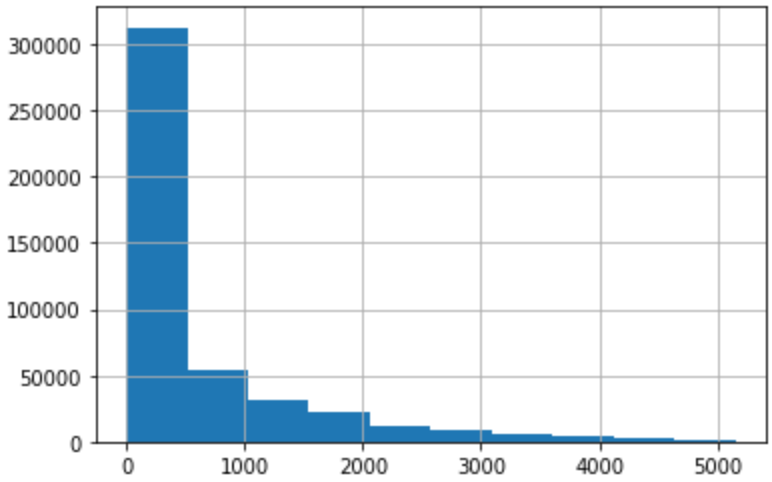
Do wczytania i obróbki danych postanowiono wykorzystać bibliotekę pandas. Pandas jest to potężne, elastyczne i łatwe do użytkowania narzędzie z otwartym kodem źródłowym, które pozwala na sprawne manipulowanie danymi. Biblioteka ta udostępnia bardzo przydatną klasę - DataFrame. Klasa ta umożliwia wczytanie danych do struktury w języku python, na przykład z pliku csv i wykonywanie na nich różnorodnych operacji matematycznych. Ma ona strukturę tabeli. [20]

4.3.2 Elementy odstające

Po wczytaniu danych, kolejnym krokiem było usunięcie elementów odstających. Elementy odstające są to, jak wskazuje nazwa elementy które w znacznym stopniu odbiegają od pozostałych, co może doprowadzić do zaburzeń w działaniu algorytmu. Do wykrycia elementów odstających postanowiono posłużyć się kolumną z dochodem. Technika jaka została wykorzystana do usunięcia niechcianych elementów to zmodyfikowana **Reguła Trzech Sigm**. Polega ona na usunięciu wszystkich elementów, które leżą poza obszarem 3 odchyleń standardowych od średniej.

$$z = (x - \textit{wartość średnia}) / \textit{odchylenie standardowe} \quad (5)$$

W powyższym wzorze $\boldsymbol{x}$ to wartość transakcji dla danego klienta, wartość średnia dla kolumny z transakcjami wyniosła 689,46, a odchylenie standardowe 1274,35. W celu zmniejszenia ilości usuwanych elementów zdecydowano się na wykorzystanie wariacji reguły trzech sigm, poprzez zmodyfikowanie współczynnika $\boldsymbol{z}$ z wartości 3, na wartość 3.5. Wobec tego ze zbioru danych zostały usunięte wszystkie rekordy, dla których wartość transakcji przekroczyła 5149. Klientów, którzy dokonali zakupu za kwotę przekraczającą 5149 było 5463, co stanowi 1.19% całego zbioru. Usunięcie tych rekordów pozwoliło na pozbycie się ze zbioru, bardzo dużych wartości transakcji (największa wyniosła prawie 300 tysięcy), które mogły bardzo zaburzać wyniki, szczególnie przy wykorzystaniu techniki RMSE (ang Root Mean Square Error), która potęgowała by te wartości.



Rysunek 15. Na osi pionowej przedstawiona jest liczba transakcji, a na osi poziomej ich kwota.

Powyżej przedstawiono histogram dla wartości transakcji po oczyszczeniu z elementów odstających. Jak można zauważyć ma on silną prawostronną skośność, lecz w przypadku sieci neuronowych nie stanowi to problemu. Wynika z niego, że znaczna większość klientów dokonuje zakupu za mniejsze kwoty, a tylko nieliczni wydają więcej pieniędzy. Dodatkowym argumentem, przemawiającym za usunięciem transakcji o większych wartościach było przeświadczenie że prawdopodobnie klienci, którzy decydują wydać się znaczną kwotę na jakiś produkt, podejmują tą decyzję na podstawie innych czynników niż wyświetlanej reklamy.

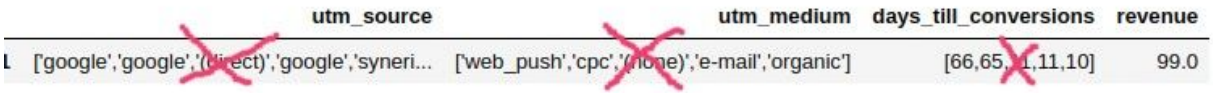
4.3.3 Przetworzenie ścieżek

Kolejnym krokiem podjętym podczas przetwarzania danych była obróbka poszczególnych komórek ze struktury. Ponieważ ścieżki były zapisane w postaci ciągów znaków, które postanowiono przekonwertować do struktury listy w języku python. Wykorzystano do tego kolejnej przydatnej funkcjonalności klasy DataFrame a mianowicie metody apply(). Jako argument przyjmuje ona funkcje i wykonuje ją dla każdego rekordu struktury. Dzięki temu po zaimplementowaniu metody do konwertowania łańcucha znaków do listy można było z łatwością wykonać ją dla wszystkich rekordów na raz. Należy wspomnieć o tym, że przechowywanie zmiennych obiektów w postaci list w komórkach obiektu DataFrame

nie jest dobrą praktyką, ponieważ elementy tego obiektu powinny być niezmienne, aby uniknąć niektórych problemów wynikających z tym jak działa język python, lecz mimo to zdecydowano się takie rozwiązanie, ponieważ w na dalszym etapie prac wymagane było częsta modyfikacja tych ścieżek.

4.3.4 Usunięcie wejść bezpośrednich

Ostatnim etapem oczyszczania danych było usunięcie bezpośrednich wejść, które stanowiły aż 53% danych, lecz niestety, było one niezbyt użyteczne dla modelu który tworzyłem. Jako wejścia bezpośrednie traktowane były wszystkie wejścia z parametrem ‘direct’ w kolumnie *utm_source*, oraz wszystkie z parametrem ‘none’ w kolumnie *utm_medium*. Należało pamiętać o tym, aby przy usuwaniu ścieżki z listy usunąć też odpowiadające temu wejściu inne wartości, takie jak czas. Jeśli ścieżka składała się z kilku elementów, to nie usuwano całego rekordu lecz tylko elementy odpowiadające wejściu bezpośredniemu, co zaprezentowano na poniższym rysunku.



Rysunek 16. Usunięcie wejść bezpośrednich, wraz z odpowiadającymi im danymi.

Usuwanie danych zawsze wiąże się z pewnym ryzykiem, ponieważ jest szansa, że można by było je wykorzystać w jakiś sposób, na przykład na ich podstawie spróbować odczytać różne trendy, w stylu wzmożony ruch w niektórych okresach roku, lecz nie było to rozważane w tej pracy. Po oczyszczeniu liczba źródeł i medium wraz z liczbą wystąpień prezentuje się następująco:

Źródło oraz medium	**Liczba wystąpień par**
('google', 'cpc')	132504
('facebook', 'cpc')	132045

('synerise', 'organic')	131781
('google', 'web_push')	131398
('bing.com', 'referral')	87186
('facebook.com', 'referral')	84155
('google', 'e-mail')	80622
('google.com', 'referral')	72756
('synerise', 'e-mail')	54295

Rysunek 17. Tabela z liczbą wystąpień dla poszczególnych par po oczyszczeniu zbioru.

4.3.5 Utworzenie wektorów wejściowych

Mając już oczyszczone dane, kolejnym krokiem było przetworzenie ścieżek użytkownika do takiej formy, którą można wprowadzić jako dane wejściowe do modelu sieci neuronowej. Elementy ścieżek użytkowników należy traktować jako dane kategoryczne, ponieważ mają one określoną liczbę możliwych stanów. Dla *utm_source* są to odpowiednio: Google, Facebook, Synerise, Bing, a dla *utm_medium*: cpc, e-mail, organic, referral, sms_text, web_push. Niektóre algorytmy, taki jak na przykład drzewa decyzyjne potrafią sobie poradzić z danymi kategorycznymi, lecz wiele innych algorytmów wśród których są między innymi sieci neuronowe wymagają danych wejściowych w postaci numerycznej. W związku z tym należało przekształcić dane ścieżki w postaci danych kategorycznych do formy numerycznej. Istnieją dwie możliwości takiego przekształcenia. Pierwszym z nich jest zwykłe zastąpienie kategori liczbami. Czyli na przykład jeśli mamy do czynienia z trzema kategoriami wykształcenia: podstawowe, średnie i wyższe, to możemy zastąpić odpowiednio wykształcenie podstawowe wartością 1, średnie wartością 2, a wyższe wartością 3. W takim przypadku, byłoby to dobre rozwiązanie, ponieważ wykształcenie jest uporządkowane i na przykład wykształcenie średnie, jest wyżej niż podstawowe, a niżej niż wyższe. Odpowiada to liczbie 2, która jest większa niż 1, a mniejsza niż 3. Jednak w przypadku ścieżek klientów nie ma możliwości wykorzystania tego

rozwiązania, bo nie istnieją hierarchiczne zależności między poszczególnymi kanałami. Dlatego zdecydowano się wykorzystać technikę One-Hot Encoding. Polega ona na zakodowaniu każdej kategori do wektora w postaci binarnej, w którym wszystkie wartości są zerami, poza jedną, która jest jedynką. [15]

```
ENCODING_DIC_UTM_SOURCE = {
    'google':       [1, 0, 0, 0],
    'facebook':     [0, 1, 0, 0],
    'synerise':     [0, 0, 1, 0],
    'bing':         [0, 0, 0, 1],
}

ENCODING_DIC_UTM_MEDIUM = {
    'cpc':      [1, 0, 0, 0, 0, 0],
    'e-mail':   [0, 1, 0, 0, 0, 0],
    'organic':  [0, 0, 1, 0, 0, 0],
    'referral': [0, 0, 0, 1, 0, 0],
    'sms_text': [0, 0, 0, 0, 1, 0],
    'web_push': [0, 0, 0, 0, 0, 1],
}
```

Rysunek 18. Słowniki do zakodowania wartości kategorycznych.

W celu przekształcenia ścieżek z postaci kategorycznej do wektorów zostały utworzone specjalne słowniki kodujące odpowiednio dla ścieżek zawierających źródła, jak i dla tych zawierających media, co przedstawiono na powyższym rysunku. Dla lepszego wyjaśnienia najlepiej posłużyć się przykładem, załóżmy że mamy następującą ścieżkę dla danego użytkownika, przedstawioną na poniższym rysunku.

utm_source	utm_mdeium	days_till_conversions	revenue
[google, facebook]	[cpc, referral]	[5, 2]	99

Rysunek 19. Ścieżka dla danego użytkownika.

Kodujemy odpowiednio:

‘google’ -> [1, 0, 0, 0]

‘facebook’ -> [0, 1, 0, 0]

Czyli cała ścieżka ze źródłami będzie prezentowała się następująco:

[‘google’, ‘facebook’] -> [[1, 0, 0, 0], [0, 1, 0, 0]]. Prezentowana ścieżka jest teraz reprezentowana jako tablica dwuwymiarowa, pierwszym wymiarem jest liczba źródeł w ścieżce, tutaj odpowiednio 2, drugim wymiarem jest natomiast długość wektora kodującego dane źródła w tym przypadku 4. O ile długość wektora kodującego jest stała dla danej ścieżki i wynosi odpowiednio 4 dla *utm_source* i 6 dla *utm_medium*, to liczba elementów w ścieżce jest zmienna i zależna od tego ilu interakcją został poddany dany użytkownik. Może wynosić 1, a równie dobrze może wynosić 100. Ponieważ jako dane wejściowe w wykorzystanych modelach (opisanych w kolejnych rozdziałach) wymagana jest stała wielkość wektora. Zdecydowano się na określić ją na sztywno i dopasować do niej wszystkie ścieżki. Zdecydowano się na 4 różne wielkości tego wektora: 5, 10, 20, 40 i dla każdej z nich osobno wyszkolono i oszacowano skuteczność modeli, w celu sprawdzenia jaka długość będzie najodpowiedniejsza. Czyli nawiązując do wcześniejszego przykładu, w którym użytkownikowi przedstawiono dwie reklamy jedną na Google, drugą na portalu Facebook, mielibyśmy długość wektora równą 2. Więc aby osiągnąć wymaganą długość wektora, na jego początku zostały dodane zera. Analogicznie jeśli wektor był zbyt długi, ponieważ użytkownikowi wyświetlono znaczną liczbę reklam, brano tylko n ostatnich reklam (gdzie n odpowiednio przyjmowało 5, 10, 20, 40).

W opisany w poprzednim akapicie sposób zostały utworzone dane wejściowe, z których każdy użytkownik miał przyporządkowany dwuwymiarowy wektor, zapisano je jako nowe kolumny ***utm_source_embedded_2d***, oraz ***utm_medium_embedded_2d***. Wykorzystano je jako danych wejściowych do modeli, w których pierwszą warstwą był LSTM. Dodatkowo, aby móc również trenować modele w których pierwszą warstwą jest Dense, utworzono dodatkową kolumnę ze spłaszczonymi wektorami, czyli na przykład wektor o wymiarach (6,10) został spłaszczony do wektora o jednym wymiarze (60), tak powstały kolumny ***utm_source_embedded_1d*** oraz ***utm_medium_embedded_1d.***

Ostatnią operacją jako została wykonana na wektorach było utworzenie nowych kolumn z zakodowaną również informacją o tym, kiedy poszczególna reklama została wyświetlona. Nawiązując do rysunku ... , kodowanie wyglądało następująco:

‘google’ -> [1, 0, 0, 0, 5]

‘facebook’ -> [0, 1, 0, 0, 2]

Gdzie zielonym kolorem zaznaczono czas w dniach jaki minął od wyświetlenia reklamy do dokonania transakcji przez użytkownika. Tak utworzone kolumny zapisano odpowiednio w kolumnach ***utm_source_embedded_1d_with_time*** oraz ***utm_source_embedded_2d_with_time***

4.3.6 Podział danych

Kolejnym, a zarazem ostatnim krokiem który został wykonany, był podział danych na dwa zbiory. Jeden służący do trenowania modeli, drugi do ich testowania. Często praktyką stosowaną przy takim problemie jest podział zbioru w stosunku 80% danych jako zbiór treningowych, oraz 20% jako zbiór do testowania. Właśnie taki podział postanowiono wykorzystać w tym projekcie. Często elementy do poszczególnych grup podziału wybiera się losowo, lecz ze względu na to, że dane w zbiorze były uporządkowane zgodnie z datą, kiedy odbyła się transakcja, aby dane były odpowiednio reprezentowane w obu zbiorach postanowiono co 5 element przenieś do zbioru testowego. Finałowy podział wyglądał następująco: 193555 rekordów do zbioru treningowego, a 48389 do zbioru służącego do walidacji i testowania.

4.4 Ocena skuteczności modeli

Mając przygotowane dane, kolejnym etapem było określenie w jaki sposób, poszczególne modele będą ze sobą porównywane w celu sprawdzenia, który z nich sprawuje się najlepiej. Należy pamiętać, że model został nauczony na danych treningowych, natomiast do określania skuteczności jego działania posługiwano się danymi testowymi. Do oszacowania skuteczności modelu postanowiono wykorzystać technikę średniego błędu kwadratowego. (ang Root Mean Square Error). Predykcja wyglądała następująco, dla każdego rekordu z danych testowych pobierana była ścieżka, i na jej podstawie model przewidywał, jaka będzie kwota transakcji. Następnie kwotę tę porównywano z prawdziwą wartością jaką klient faktycznie wydał. Operację

tę powtarzano dla każdego rekordu. A wyniki zgodnie z techniką RMSE, były podnoszone do kwadratu. Ostatnim krokiem było zsumowanie wszystkich wyników i obliczenie pierwiastka.

$$RMSE = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (y_i - \hat{y})^2} \tag{6}$$

Technika średniego błędu kwadratowego przedstawiona jest za pomocą powyższego wzoru, gdzie $\mathbf{y_i}$ to wartość predykowana przez model dla rekordu i, a $\hat{\mathbf{y}}$ to rzeczywista wartość dla tego rekordu.

Rozwiązywanie problemów metodą uczenia mszynowego najlepiej rozpocząć od stworzenie rozwiązania naiwnego, po to, aby potem można było porównywać z nim kolejne rozwiązania i sprawdzać czy są lepsze, jeśli tak to o ile. Dlatego też jako rozwiązanie naiwne postanowiono zastosować prostę technikę, a mianowicie, zawsze przewidywanie kwoty transakcji jako wartości średniej z wszystkich transakcji wynoszącej 590. Średni błąd kwadratowy obliczony tym sposobem wyniosiósł **RMSE = 867,34** i właśnie do tej wartości były porównywane kolejne modele. [21]

5 Wyniki

5.1 Modele First Touch i Last Touch

Kolejnym etapem było utworzenie prostych modeli omówionych w podrozdziale "standardowe modele atrybucji". Są one lepsze od modelu naiwnego w predykcji kwoty transakcji, ze względu na to, że zamiast tak jak w modelu naiwnym produkowania jednej średniej dla wszystkich klientów, zostało utworzonych kilka różnych średnich wartości, dla każdego kanału z osobna.

Źródło	Wartość dla First Touch	Wartość dla Last Touch
Google	581,53	568,48
Bing	780,63	744,07
Synerise	540,19	539.20
Facebook	606,57	538,40

Rysunek 20. Przedstawienie średnich wartości transakcji dla poszczególnych źródeł.

Źródło	Wartość dla First Touch	Wartość dla Last Touch
Organic	539,55	539,82
Referral	775,94	727,65
Web_push	559,88	533.07
E-email	536,60	539.94
Cpc	538,40	528.30

Rysunek 21. Przedstawienie średnich wartości transakcji dla poszczególnych medium.

Na powyższych tabelach przedstawiono średnie wartości dla technik First Touch i Last Touch w zależności od tego, czy wykorzystywano ścieżki ze źródłami, czy z medium. Czyli predykowanie polegało na przewidywanie jednej z tych wartości w zależności od tego, który element wystąpił odpowiednio pierwszy lub ostatni w ścieżce wyniki prezentują się następująco:

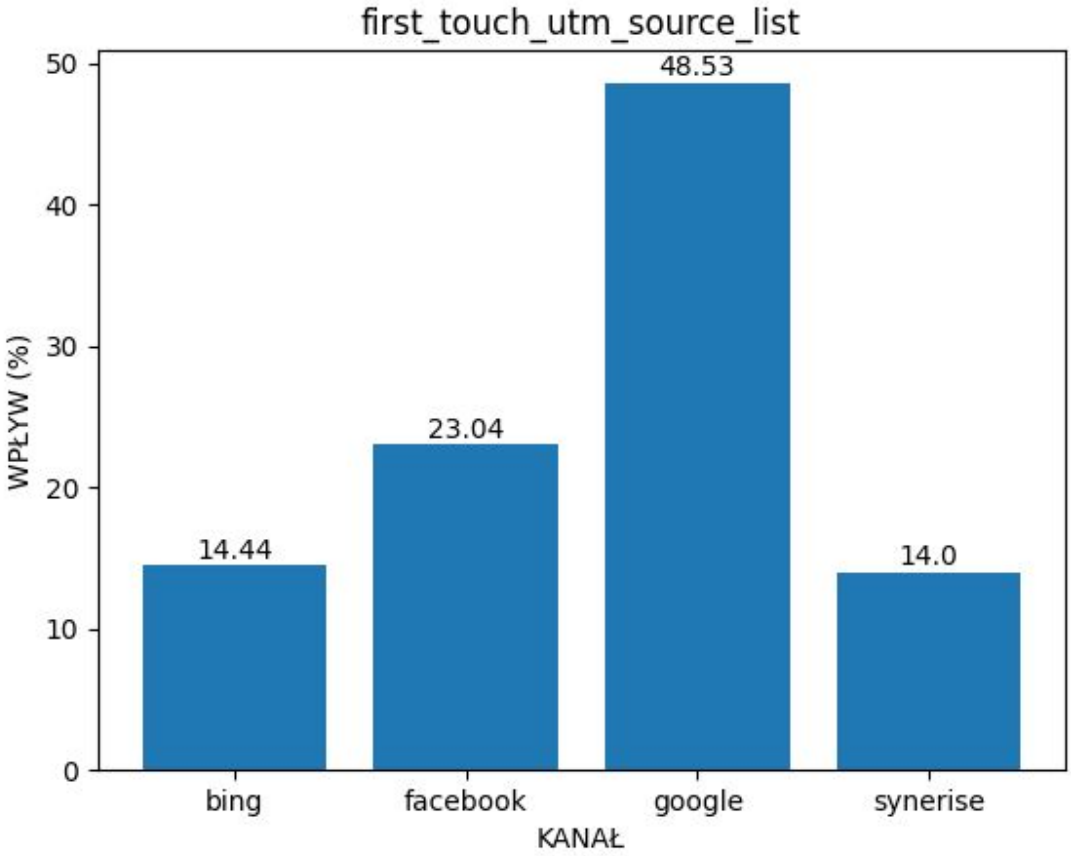
Wykorzystana ścieżka	Użyta technika	**RMSE**
utm_source	First Touch	**865,30**

utm_source	Last Touch	**865,32**
utm_medium	First Touch	**861,86**
utm_medium	Last Touch	**862,66**

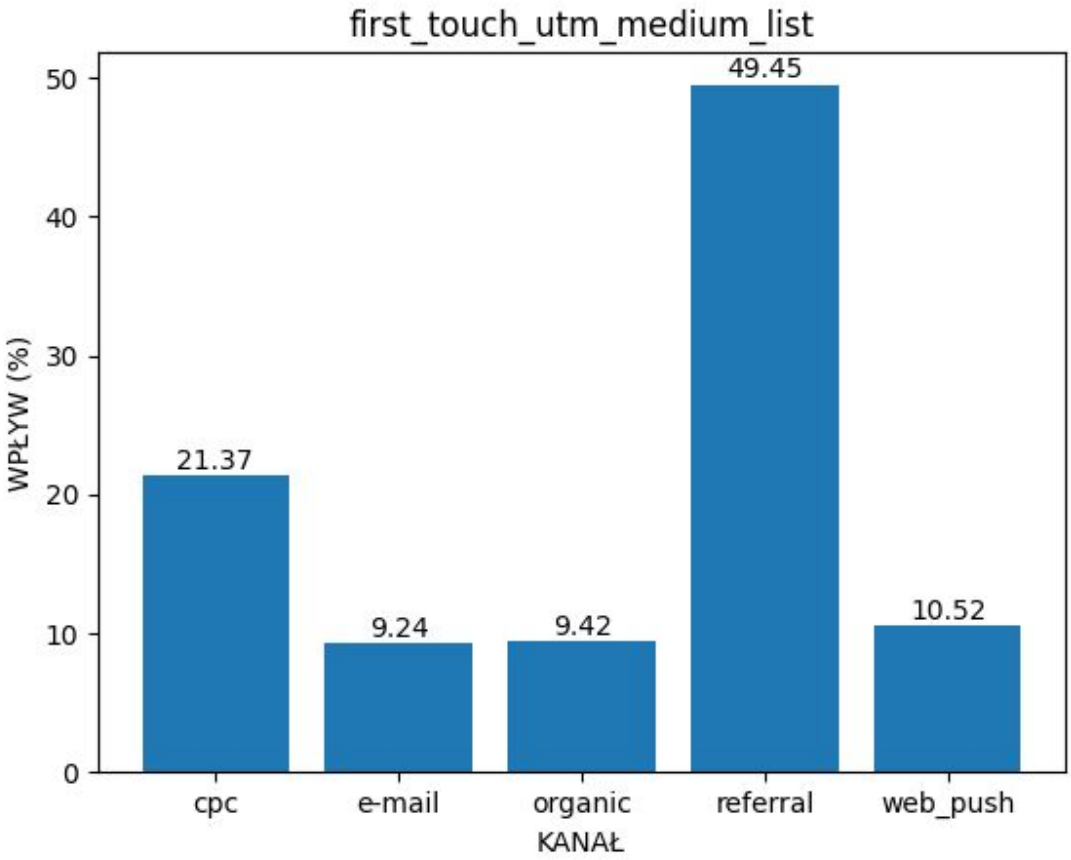
Rysunek 22. Przedstawienie średniego błędu kwadratowego w zależności od użytej techniki.

Jak można zauważyć na powyższej tabeli techniki First Touch oraz Last Touch dały zbliżone do siebie rezultaty. Przy użyciu ścieżek *utm_source* średni błąd kwadratowy RMSE był o około **2** mniejszy niż w przypadku opisanej w poprzednim rozdziale metody naiwnej, przy użyciu której został otrzymany wynik **867,34**. Natomiast przy użyciu ścieżek *utm_medium* błąd ten był aż o **5** mniejszy, co jest już znaczną poprawą w stosunku do metody naiwnej.

Aby określić wpływ poszczególnych kanałów na podjęcie decyzji przez użytkownika o dokonaniu transakcji zdecydowano się zaimplementować technikę opisaną w publikacji Deep Neural Net with Attention for Multi-channel Multi-touch Attribution. Polega ona na usuwaniu wszystkich wystąpień danego kanału ze ścieżek klientów i sprawdzeniu jak zmienia się suma pozostałych transakcji. Jeśli na przykład po usunięciu kanału **Google** suma ta znacząco się zmniejszy, to znaczy że kanał ten jest istotny. Jeśli natomiast suma ta zmieni się tylko nieznacznie, to wynika z tego, że kanał jest mało znaczący. Wyniki przeskalowano odpowiednio, aby ich suma dawała 100%.



Rysunek 23. Wpływ poszczególnych źródeł na decyzję o zakupie, oszacowany przy użyciu techniki First Touch.



Rysunek 24. Wpływ poszczególnych medium na decyzję o zakupie, oszacowany przy użyciu techniki First Touch.

Ponieważ przy użyciu techniki First Touch uzyskano lepsze wyniki (mniejsza RMSE), to właśnie dla niej zilustrowano na powyższych rysunkach wpływy poszczególnych kanałów. Jak można zaobserwować na rysunku 23, źródłem o największym znaczeniu jest **google** i jego udział wynosi aż **48.53%**. Z kolei medium z największym udziałem jest **referral**, a jego wpływ to **49,45%**.

5.2 Model Dense

Kolejnym model, tym razem stworzonym już przy wykorzystaniu głębokiego uczenia jest model Dense. Składa się on tylko z najprostszych warstw Dense. Po dostrojeniu modelu, poprzez dopasowanie takich parametrów jak: liczba warstw, liczba neuronów w poszczególnych warstwach, liczba epok, rozmiar wielkości partii (ang. batch size), wybraniu funkcji optymalizującej, funkcji strat oraz funkcji aktywacyjnej, otrzymano wyniki przedstawione na na poniższych tabelach.

Wykorzystana ścieżka	Liczba elementów w ścieżce	**RMSE**
utm_source	5	**861,10**
utm_source	10	**860,69**
utm_source	20	**861,33**
utm_source	40	**861,27**
utm_medium	5	**857,81**
utm_medium	10	**857,71**
utm_medium	20	**858,23**
utm_medium	40	**858,48**

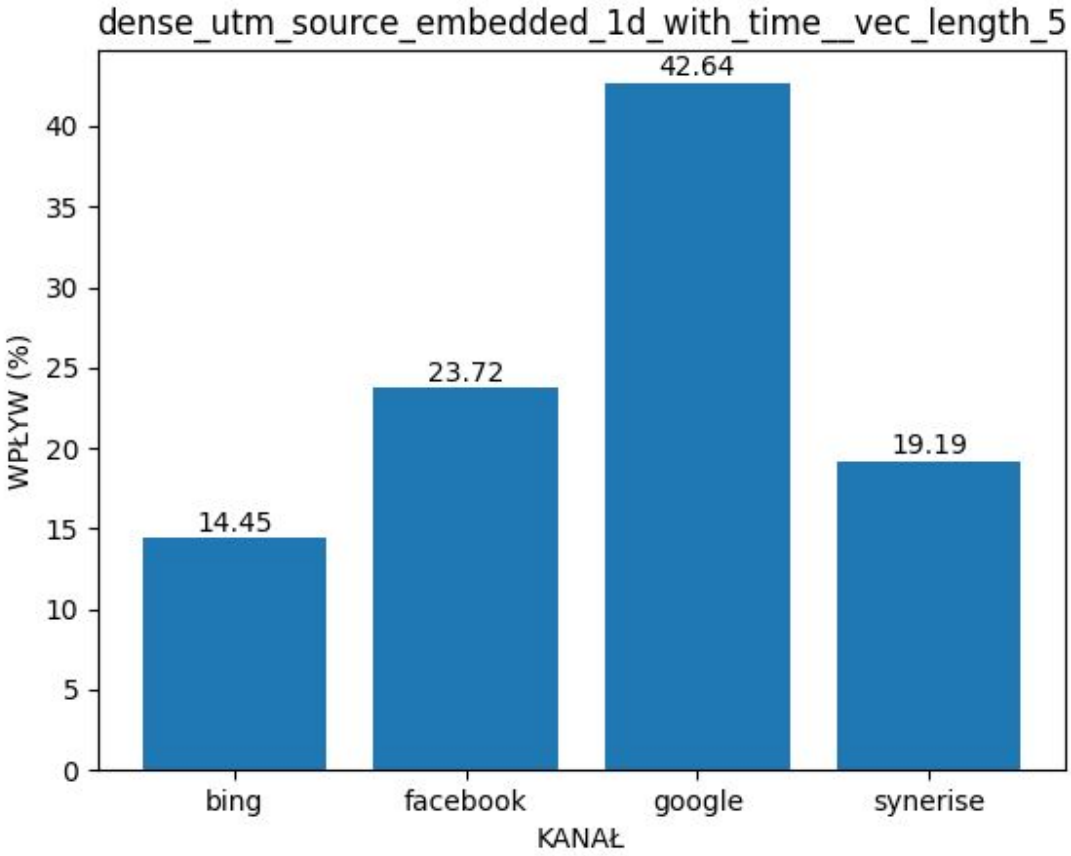
Rysunek 25. Oszacowanie skuteczności modeli, w zależności od liczby elementów w ścieżce.

Wykorzystana ścieżka	Liczba elementów w ścieżce	**RMSE**
utm_source_with_time	5	**858,48**
utm_source_with_time	10	**859,30**

utm_source_with_time	20	**660,72**
utm_source_with_time	40	**859,35**
utm_medium_with_time	5	**855.72**
utm_medium_with_time	10	**856,22**
utm_medium_with_time	20	**857,43**
utm_medium_with_time	40	**856,89**

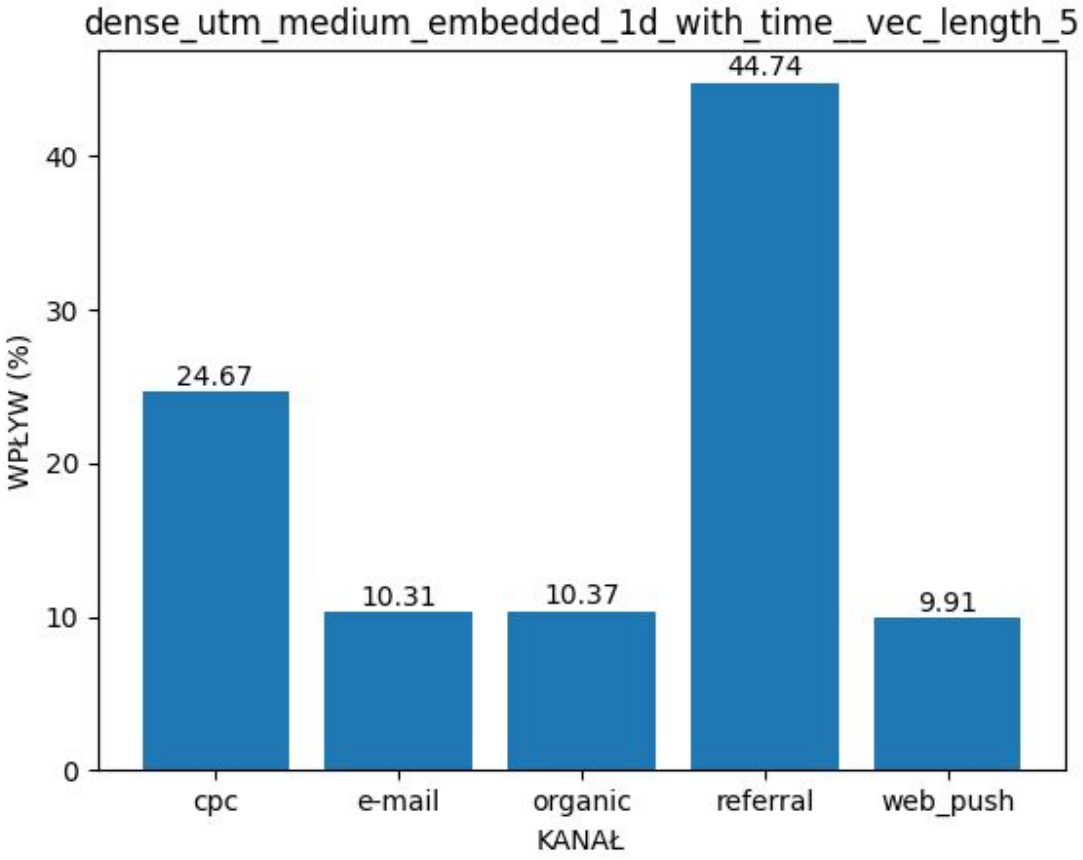
Rysunek 26. Oszacowanie skuteczności modeli, w zależności od liczby elementów w ścieżce, z uwzględnieniem czasu transakcji.

Jak można zauważycie, wykrzystanie wektorów w których uwzględniony jest również czas (rysunek 26) przyniosło średnio poprawę o **2**, względem tych bez czasu (rysunek 25). Kolorem zielonym zaznaczono najlepsze wartości odpowiednio z wykorzystaniem źródeł oraz medium. Ciekawą zależnością jest to, że model osiąga nalepsze wyniki, dla najkrótszych wektorów w których jest zakodowane tylko 5 ostatnich elementów ze ścieżki. Wynika to prawdopodobnie z tego, że użyty prosty model, ma problem z nauczeniem się zależności przy większej długości wektora wejściowego. Do zilustrowania wpływu poszczególnych kanałów posłużono się modelami które dały najlepsze rezultaty, zaznaczone na zielono na rysunku 26.



Rysunek 27. Wpływ poszczególnych źródeł na decyzję o zakupie, oszacowany przy użyciu modelu Dense.

Na rysunku 27 zilustrowano wpływ poszczególnych źródeł na decyzję o konwersji. Porównując go z wpływami oszacowany przy użyciu techniki First Touch (rysunek 23) można zauważyć, że wpływ źródła **Google** spadł prawie o **6%**, na rzecz **Synerise**, natomiast wpływ pozostałych źródeł (Bing, Facebook) utrzymał się na tym samym poziomie.



Rysunek 28. Wpływ poszczególnych medium na decyzję o zakupie, oszacowany przy użyciu modelu Dense.

Rysunek 28 przedstawia szacowany wpływ poszczególnych medium na decyzję użytkownika o dokonaniu zakupu. Porównując go z wartościami uzyskanymi techniką First Touch (rysunek 24) można zauważyć że udział medium **referral** spadł o ok. **5**% na rzecz pozostałych.

5.3 Model LSTM

Drugim z modeli stworzonym przy wykorzystaniu głębokiego uczenia jest model wykorzystujący warstwę LSTM. W przeciwieństwie do poprzedniego modelu, LSTM jako wartości wejściowe przyjmuje wektor w kształcie dwóch wymiarów. Utworzenie tego wektora opisano w rozdziale 4.3.5. Po dostrojeniu modeli otrzymano następujące wyniki:

Wykorzystana ścieżka	Liczba elementów w ścieżce	**RMSE**
utm_source	5	**860,98**
utm_source	10	**860,75**
utm_source	20	**860,31**
utm_source	40	**861,41**
utm_medium	5	**858.93**
utm_medium	10	**858,68**
utm_medium	20	**859,76**
utm_medium	40	**859,92**

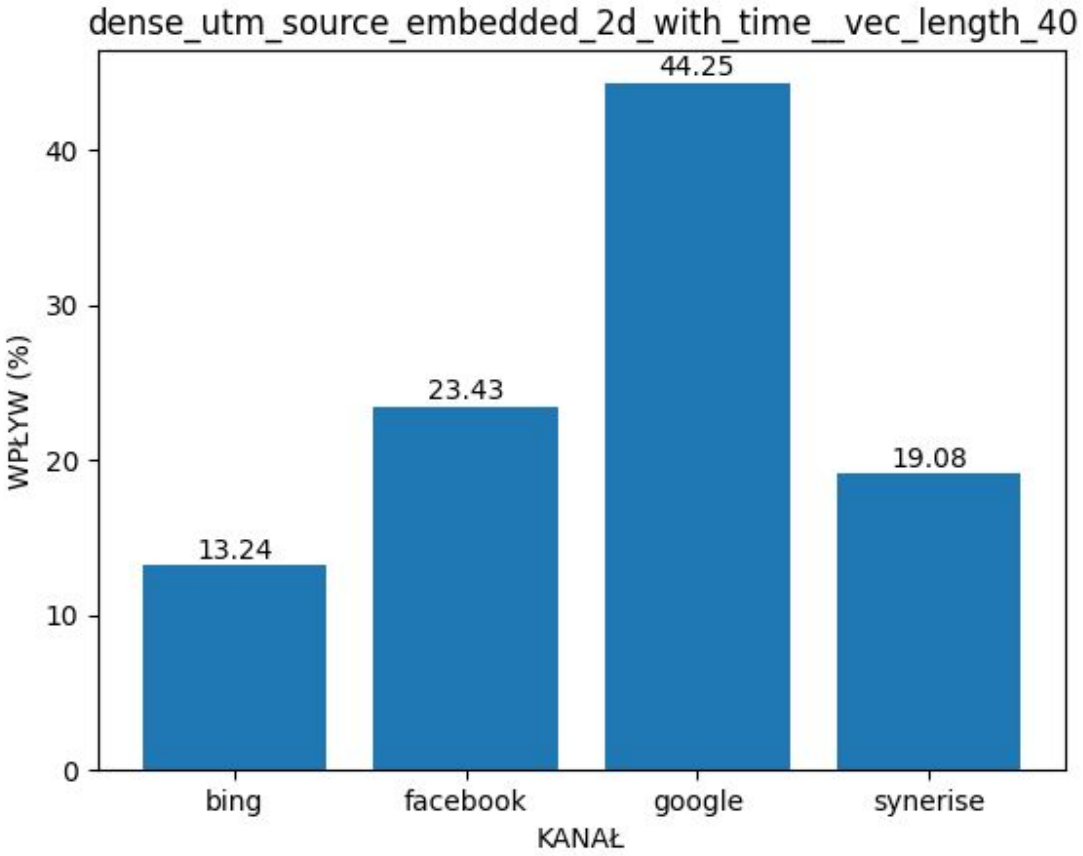
Rysunek 29. Oszacowanie skuteczności modeli, w zależności od liczby elementów w ścieżce.

Wykorzystana ścieżka	Liczba elementów w ścieżce	RMSE
utm_source	5	**860,61**
utm_source	10	**859,40**
utm_source	20	**859,11**
utm_source	40	**858,98**

utm_medium	5	**858,26**
utm_medium	10	**858,21**
utm_medium	20	**858,13**
utm_medium	40	**856,08**

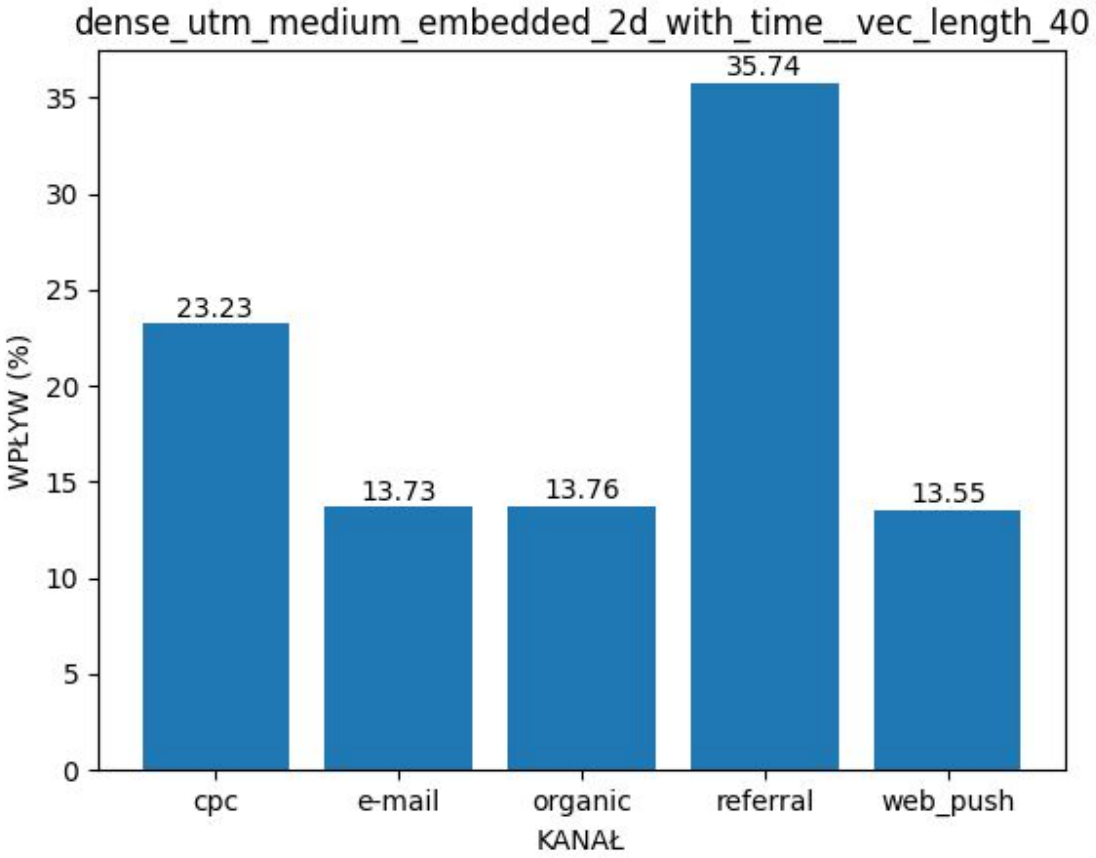
Rysunek 30. Oszacowanie skuteczności modeli, w zależności od liczby elementów w ścieżce, z uwzględnieniem czasu transakcji.

Jak można zauważyć na rysunkach 29 i 30, model LSTM podobnie jak model Dense osiąga lepsze wyniki po dodaniu czasu do wektora. LSTM ma dokładność praktycznie na tym samym poziomie co model Dense, jednak w przeciwieństwie do modelu Dense, który działał lepiej na krótszych ścieżkach, model LSTM sprawuje się lepiej przy dłuższych ścieżkach.



Rysunek 31. Wpływ poszczególnych źródeł na decyzję o zakupie, oszacowany przy użyciu modelu LSTM.

Na rysunku 31 zilustrowano wpływ poszczególnych źródeł na decyzję o konwersji, oszacowany przy wykorzystaniu modelu LSTM. Jest on niemal identyczny jak ten przewidziany przez model Dense, jedyną zmianą jest wzrost wpływu kanału **Google** o **1%**, przy jednoczesnym zmniejszeniu o **1%** wpływu ze źródła **Bing**.



Rysunek 32. Wpływ poszczególnych medium na decyzję o zakupie, oszacowany przy użyciu modelu LSTM.

Rysunek 32 przedstawia szacowany wpływ poszczególnych medium na decyzję użytkownika o dokonaniu zakupu. Porównując go z wartościami uzyskanymi przy użyciu modelu Dense (rysunek 28) można zauważyć ogromny spadek medium **referral** na rzecz pozostałych kanałów. Wynika to prawdopodobnie z tego, że model LSTM wykorzystywał dłuższe ścieżki.

5.4 Porównanie modeli

Wykorzystana ścieżka	Rozwiązanie naiwne	First Touch	Last Touch	Dense	LSTM
utm_source	867,34	865,30	865,32	858,48	858,98
utm_medium	867,34	861,86	862,66	855.72	856,08

Rysunek 33. Porównanie skuteczności różnych technik, poprzez wartość RSME (mniej = lepiej).

Na rysunku 33 porównano wyniki dla poszczególnych technik. Dla każdej z technik posłużono się modelem który dawała najlepsze rezultaty. Jak można zauważyć techniki First Touch i Last Touch dawały podobne rezultaty do siebie, lecz znacznie gorsze niż modele wykorzystujące uczenie głęboki: Dense, LSTM. Modele Dense jest tylko nieznacznie skuteczniejszy niż LSTM i oba te modele mogą być z powodzeniem wykorzystywane do analizy.

6 Podsumowanie

6.1 Osiągnięte cele

Głównym celem było stworzenie modelu z wykorzystaniem technik uczenia głębokiego, który będzie sprawował się lepiej od technik First Touch i Last Touch. Cen ten został osiągnięty, ponieważ zarówno model Dense jak i model LSTM osiągają lepsze wyniki. Jako że modele te lepiej predykują kwotę transakcji można założyć że również rozkład atrybucji dla poszczególnych kanałów jest bardziej miarodajny i to właśnie tymi wynikami powinno się kierować przy wyborze strategii marketingowej.

6.2 Napotkane problemy

Głównym problemem przy tworzeniu projektu była obróbka danych w celu doprowadzenia ich do pożądanej postaci, często wymagało to znacznej ilości operacji wykonanych na surowych danych wejściowych. Ponadto dodatkowym problem był brak danych

o użytkownikach którzy kliknęli w reklamę, ale nie dokonali transakcji. Mając również takie dane można by było znacznie lepiej oszacować wpływ poszczególnych kanałów. Kolejnym problemem była specyfika tematu pracy. Jest to dość nowe zagadnienie, więc przez niezbyt dużą liczbę artykułów w Internecie na temat wykorzystania technik uczenia głębokiego w analizie atrybucji, znacznie utrudnione było pozyskiwanie wartościowych informacji.

6.3 Możliwy rozwój

W celu zwiększenia skuteczności modeli można by było zakodować pary (źródło + medium) do wektora i sprawdzić jaki to będzie miało wpływ na predykowane przez model wartości. Ponadto można również spróbować wykorzystać daty transakcji, na przykład sprawdzić, czy są zależności między poszczególnymi miesiącami a średnią wielkością transakcji.

7 Bibliografia

[1] F. Forsterling, “Atrybucje”, GWP (2005)

[2] Grzegorz Kowalczyk, “Modelowanie atrybucji i atrybucja Data-Driven”, stan na dzień 04.11.2020, https://www.adpeak.pl/blog/modelowanie-atrybucji-data-driven/

[3] “Visitor Count Tracking Usage Distribution in the Top 1 Million Site”, https://trends.builtwith.com/analytics/visitor-count-tracking, stan na dzień 17.11.2020

[4] Jereffy Kranz, “The Complete Guide to UTM Codes: How to Track Every Link and All the Traffic From Social Media”, https://buffer.com/library/utm-guide/, stan na dzień 04.11.2020

[5] “Google Analytics UTM tagging best practices”, https://funnel.io/resources/google-analytics-utm-tagging-best-practices, stan na dzień 04.11.2020

[6] Tom Bennet, “The Complete Guide to Direct Traffic in Google Analytics”, https://moz.com/blog/guide-to-direct-traffic-google-analytics, stan na dzień 04.11.2020

[7] Ruth C. Fong, Walter J. Scheirer & David D. Cox, "Using human brain activity to guide machine learning”, Scientific Reports 2018

[8] Artem Oppermann, “What is Deep Learning and How does it work?”, https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac, stan na dzień 04.11.2020

[9] Hisham El-Amir, Mahmoud Hamdy, “Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow”, Apress 2019

[10] Wang SC. “Artificial Neural Network. In: Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, Springer 2003

[11] Dishashree Gupta, “Fundamentals of Deep Learning – Activation Functions and When to Use Them?”, https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/, stan na dzień 04.11.2020

[12] Algorithmia, “Introduction to Loss Functions”, https://algorithmia.com/blog/introduction-to-loss-functions, stan na dzień 04.11.2020

[13] Aditi Mittal, “Understanding RNN and LSTM”, https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e,

stan na dzień 04.11.2020
[14] Yoshua Bengio, Patrice Simard, Paolo Frasconi, “Learning Long-Term Dependencies with Gradient Descent Is Difficult”, IEEE Transactions on Neural Networks 1994
[15] “Understanding LSTM Networks”, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, stan na dzień 04.11.2020
[16] François Chollet. “Deep learning with Python”. Shelter Island: Manning Publications Co 2018
[17] Tim Dettmers, “Which GPU(s) to Get for Deep Learning: My Experience and Advice for Using GPUs in Deep Learning”, https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/, stan na dzień 04.11.2020
[18] “Paid vs organic; digital marketing’s big debate”, https://www.strategy-plus.net/articles/paid-advertising-organic-advertising/, stan na dzień 04.11.2020
[19] Misha Berrien, “How to Structure a Python-Based Data Science Project”, https://medium.com/swlh/how-to-structure-a-python-based-data-science-project-a-short-tutorial-for-beginners-7e00bff14f56, stan na dzień 04.11.2020
[20] Jason Browniee, “Why One-Hot Encode Data in Machine Learning?”, https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/, stan na dzień 04.11.2020
[21] Max Kuhn & Kjell Johnson, “Applied Predictive Modeling”, Springer 2016