**Politechnika Krakowska**
**im. Tadeusza Kościuszki**

Wydział Fizyki, Matematyki i Informatyki

**Piotr Weszka**

Numer albumu: 104901

# Uczenie maszynowe jako narzędzie do klasyfikacji zdarzeń w fizyce cząstek elementranych

# Machine learning as tool for classification in high energy physics

**Praca magisterska**
**na kierunku Fizyka techniczna**

Praca wykonana pod kierunkiem:
**Dr Radosław Kycia**

**Uzgodniona ocena:**.....

.....
podpisy promotora i recenzenta

**Kraków 2018**

# Spis treści

## Część teoretyczna



## Część praktyczna

# I Część teoretyczna

# 1. Wstęp

Dziedziną która jest w czołówce, jeśli chodzi o ilość produkowanych danych jest fizyka cząstek elementarnych. W październiku 2017 roku w samym laboratorium CERN w ciągu miesiąca zgromadzono 12.3 petabajta danych [5]. Obróbka takiej ilości informacji zmusza nas do poszukiwania nowych rozwiązań w dziedzinie przetwarzania sygnału. Oprócz optymalizacji sprzętowych równie ważna jest optymalizacja algorytmów oraz rozwój technologiczny umożliwiający implementację tychże metod. Analiza wielkiej ilości danych o wielu wymiarach wymaga bowiem odpowiednich zasobów. Zwłaszcza gdy potrzebujemy zaklasyfikować zdarzenia jako istotne lub nieistotne w przeciągu ułamków sekund. Uczenie maszynowe jest metodą która w ostatnich latach zdobyła uznanie zarówno wśród profesjonalistów jak i początkujących analityków. Stało się to głównie za sprawą upowszechnienia się otwartych bibliotek oferujących implementacje algorytmów uczenia maszynowego.

# 2. Cel, zakres i metodyka pracy

W niniejszej pracy zostanie opisane opracowanie i implementacja algorytmu służącego do klasyfikacji zdarzeń pochodzących z eksperymentu OPERA. Dane te zwierają informację o kaskadach cząstek, których inicjatorem były neutrina. Bardzo lekkie i niezwykle trudne do wykrycia cząstki zaliczane do leptonów. Celem jest osiągnięcie dokładności na poziomie 90% poprawnych klasyfikacji spośród sygnałów tła.

Do realizacji celu tej pracy posłużę się skryptem w języku Python oraz modułem *scikit-learn*. Biblioteka ta zawiera implementację wielu algorytmów uczenia maszynowego w tym model losowego lasu opierający się o algorytm drzew decyzyjnych. Aby określić dokładność estymowania zbiór dostępnych danych zostanie zostanie podzielony na podzbiór trenujący oraz podzbiór walidacyjny. Za pomocą podzbioru walidacyjnego zostanie wyrysowana krzywa zwana Krzywą Operacyjną Odbiornika. Pozwala ona zobaczyć o ile lepsze są przewidywania naszego modelu względem całkowicie losowego przyporządkowania klas. Sprawdzony zostanie wpływ zmiany liczby drzew w modelu losowego lasu oraz maksymalnej liczby rozgałęzień na dokładność modelu a także na czas jaki jest potrzebny na trening.

Do wizualizacji samych zdarzeń zostanie wykorzystany inny pythonowy moduł – *matplotlib* pozwalający na tworzenie wykresów w trzech wymiarach przestrzennych.

Wczytywanie oraz manipulowanie danymi zostanie przeprowadzone przy użyciu metod biblioteki *pandas*.

# 3. Paradygmaty trenowania w algorytmach uczenia maszynowego

## 3.1 Historia komputerów

Obliczenia towarzyszą ludziom od zarania dziejów. Na początku liczby służyły do oszacowania ilości inwentarza, zmierzenia wydajności plonów czy określenia cyklu menstruacyjnego u kobiet. W miarę rozwoju ludzkości rosło zapotrzebowanie na coraz bardziej skomplikowane narzędzia. Czasem wyprzedzały one swoją epokę, czasem były odpowiedzią na bieżące potrzeby. Niemniej jednak ich złożoność rosła w szybkim tempie, szczególnie w czasach nowożytnych. Doprowadziło to do momentu w którym człowiek musiał znaleźć bardziej efektywne metody wykonywania obliczeń. Wystarczająco szybkie aby problem można było rozwiązać w skończonym czasie – krótszym niż czas jaki potrzebowałby na to zespół ludzi, a jednocześnie odpowiednio dokładne aby można z nich było zrobić praktyczny użytek. Mówimy tu o czasach w których automatyczne obliczenia symboliczne nie były nawet w sferze marzeń ówczesnych naukowców oraz wynalazców. Dlatego naturalnym sposobem radzenia sobie z problemami jakie dostarczały skomplikowane równania były obliczenia numeryczne. Dawały one wynik przybliżony jednak na tyle dokładny iż można go było zastosować w opisywanych modelach. Obliczenia numeryczne miały tę istotną zaletę, że możliwe było zaimplementowania ich na wszelakiego rodzaju maszynach liczących. Maszynach dla których pojęcie nieskończoności czy zera było nieosiągalne, ale skończone liczby i operacje na nich były bardzo proste do wykonania. Pomijając rozwiązania takie jak maszyna Schickarda czy suwak logarytmiczny - siedemnastowieczne wynalazki, niezwykle eleganckie w swojej prostocie, musimy wspomnieć o prawdziwej rewolucji która dokonała się w dziedzinie maszyn liczących, a zaczęła się w końcu XIX stulecia. Na uwagę zasługuje Kaptometr – pierwsze urządzenie które do wprowadzenia danych używało klawiszy. Wielki sukces odniósł również Analizator Różnicowy – jeden z pierwszych mechanicznych komputerów analogowych stosowanych na wielką skalę. Był on w stanie w sposób przybliżony rozwiązywać równania

różniczkowe. Przydatny między innymi dla wojska do obliczania trajektorii pocisków [50].

Dużo bardziej zaawansowaną formą były komputery lampowe (tak zwane komputery pierwszej generacji) oparte na lampach elektronowych, programowane poprzez zmianę połączeń kablowych, później poprzez karty perforowane.

16 grudnia 1947 roku to data która ma ogromne znaczenie dla ludzkości. To właśnie tego dnia w laboratoriach firmy Bell Telephone skonstruowano pierwszy działający tranzystor. Dzieło warte Nobla, którego zresztą twórcy otrzymali w 1956 roku [21]. Od tego momentu rozwój nauki przebiegał na tyle szybko, że zaledwie po 62 latach od tego doniosłego momentu możliwe było wysłanie samochodu w kierunku Marsa czy zbudowanie samolądujących rakiet [33]. Rzeczą na którą warto zwrócić uwagę jest fakt, że również polscy naukowcy mają niemały wkład w rozwój układów scalonych. Sztandarowym przykładem jest tutaj Jan Czochralski [52]. Ten poznański chemik opracował stosowaną do dzisiaj metodę otrzymywania monokryształów krzemu z których tworzy się te układy. Niewielkie płytki zawierające miliony tranzystorów, co do zasady działania, prawie takich samych jak ten stworzony ponad pół wieku temu w Bell Telephone Laboratories.

Szybkie i energetycznie wydajne komputery są obecne w prawie każdym domu. Jednak niezależnie czy mamy do czynienia z domowym „pecetem” czy ogromnym klastrem obliczeniowym. I jeden i drugi będzie działał tak dobrze, jak dobry jest zestaw instrukcji przekazany mu przez człowieka. Dlatego zaszła potrzeba wymyślenia sposobu na to, aby komputer sam potrafił zobaczyć to, co mogło umknąć programiście. Tę potrzebę wyrażają nieustające próby stworzenia sztucznej inteligencji. Jednym z jej przejawów jest uczenie maszynowe. Można je zaaplikować szczególnie tam gdzie mamy do czynienia z dużą ilością informacji do przetworzenia.

Wśród różnych podejść do zagadnienia uczenia maszynowego wyróżnia się trzy paradygmaty trenowania. Są to uczenie nadzorowane lub inaczej uczenie z

nauczycielem, uczenie nienadzorowane (uczenie bez nauczyciela) oraz uczenie przez wzmacnianie [39].

## 3.2 Uczenie nadzorowane

W uczeniu nadzorowanym dane możemy powiązać w pary: wektor danych wejściowych oraz znana wartość wyjściowa zwana również sygnałem nadzorującym. Poszukiwanym jest natomiast odwzorowanie łączące te dwa sygnały. Innymi słowy szukamy funkcji które spełni relację: $Y=f(x)$, gdzie $x$ to dane wejściowe, $Y$ to wartość wyjściowa zaś $f$ to szukana funkcja [51]. Aby zaimplementować uczenie nadzorowane potrzebny jest zbiór trenujący zawierający właśnie dane wejściowe i gotową odpowiedź. Na takim zbiorze korzystając z wybranej metody uczenia uzyskujemy wartości wag które, po wpuszczeniu na przykład do sieci neuronowej pozwolą na znalezienie optymalnego modelu do klasyfikacji nowych danych. Istotną częścią tego procesu jest również walidacja. Polega na sprawdzeniu skuteczności modelu na danych co do których mamy informację o poprawnej wartości wyjściowej ale nie pokazujemy jej sieci przez co mamy pewność że odpowiedź modeli na nowe dane będzie taka jak na zbiór walidacyjny.

Takie podejście pozwala na przetworzenie dużych ilości danych za pomocą małych zasobów. Mówiąc o uczeniu nadzorowanym warto wspomnieć o następujących algorytmach:

- Maszyna wektorów nośnych (ang. Support vector machine) – algorytm pozwala na wyznaczanie hiperpłaszczyzny lub całego ich zbioru w przestrzeni o dużej liczbie wymiarów (również nieskończenie wymiarowej). Rozważając przypadek jednowymiarowy szukamy linii, której odległość do punktów poszczególnych zbiorów będzie największa. Im odległość większa tym mniejszy błąd klasyfikacji. Algorytm znalazł zastosowanie w zagadnieniach związanych z klasyfikacją, regresją czy szukaniem właściwości odstających [39, 13].

- Naiwny klasyfikator bajesowski (ang. Naive Bayes classifier) – algorytm oparty na twierdzeniu Bayesa o prawdopodobieństwie warunkowym. Jest stosowany jako jedno z podejść do zadań z dziedziny klasyfikacji [36].

- Sieci neuronowe – znajdują swoją realizację z każdym z paradygmatów trenowania, ale pierwsza ich realizacja zwana perceptronem była próbą odtworzenia biologicznego neuronu. Proces uczenia, czy to perceptronu, czy też wielowarstwowego perceprtonu, polega na aktualizacji wag poszczególnych neuronów poprzez porównywanie wartości przewidzianej na danym etapie z wartością prawdziwą. Gdy wartość estymowana jest bliska wartości rzeczywistej proces uczenia możemy uznać za zakończony, a wytrenowaną sieć za gotową do pracy [48].

## 3.3 Uczenie nienadzorowane

W uczeniu nienadzorowanym, w przeciwieństwie do trenowaniu z nauczycielem, nie jest znana prawidłowa odpowiedź. Algorytm stara się sam przeprowadzić klasyfikację danych poprzez znalezienie wzorów które nie są bezpośrednio widoczne dla człowieka i na tej podstawie przyporządkowania odpowiedniej kategorii dla danych wejściowych. Symbolicznie można to zapisać jako $Y=f(x)$, gdzie $Y$ to dane wyjściowe, $x$ – dane wejściowe zaś $f$ to szukane odwzorowanie. W przypadku treningu bez nauczyciela znane jest tylko $x$ nieznane zaś $Y$ oraz $f$.

Wśród wielu algorytmów uczenia nienadzorowanego do najważniejszym możemy zaliczyć [39, 3]:

- K najbliższych sąsiadów (ang. k nearest neighbour) - decyzja o przynależności danego punktu w przestrzeni właściwości jest podejmowana wyłącznie na podstawie najbliższych sąsiadów klasyfikowanego punktu [36].

- autoenkodery – algorytm szczególnie przydatny w redukcji wymiarowości danych czy pozbywaniu się szumu. Ten rodzaj sieci próbuje zreprodukować dane wejściowe na swoim wyjściu. Wynik jest porównywany z oryginałem i na tej podstawie aktualizowane parametry modelu [11].

- Sieć głębokiego zaufania (Deep belief network) – Inny przejaw sztucznych sieci neuronowych. Sieci tego typu charakteryzują się lokalną kierunkowością. Poszczególne warstwy sieci mogą być ze sobą połączone. Ograniczeniem jest brak połączeń pomiędzy poszczególnymi jednostkami danej warstwy. Takie sieci znajdują zastosowanie w rozpoznawaniu obrazów czy też sekwencji wideo [43].

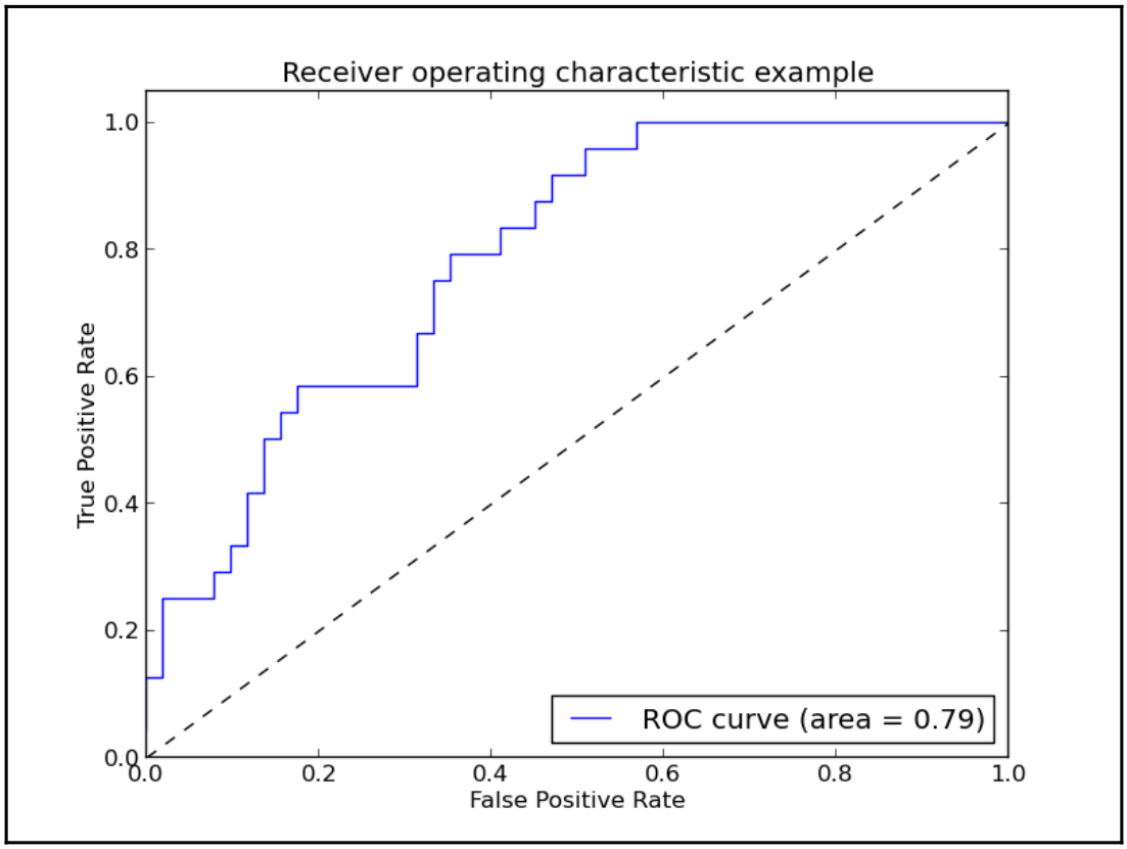
## 3.4 Uczenie przez wzmacnianie

W uczeniu przez wzmacnianie próbuje się stworzyć system który poprzez oddziaływanie ze środowiskiem stara się zwiększyć swoją skuteczność. Informacja o tym jak dobrze system dopasował się do środowiska przekazuje się za pomocą sygnału nagrody i na tej podstawie zmienia się stan systemu. Podejście to przypomina nieco algorytm uczenie nadzorowanego ale jest to dosyć zgrubne porównanie. W uczeniu nadzorowanym bowiem tym co zostaje przekazane do sieci są dokładne, prawdziwe wartości których porównanie z przewidywaniami pokazuje o ile zmienić parametry modelu. W uczeniu wspomaganym jedyną informacją zwrotną jaką otrzymuje model jest to czy całość systemu podąża we właściwym kierunku czy też nie, jest to tzw. wartość funkcji nagrody. Bazując na informacji ze środowiska system może podjąć kilka kroków stosując metodykę prób i błędów wybiera wówczas ten zwrot gdzie nagroda będzie największa. Jedną z aplikacji uczenia przez wzmacnianie można znaleźć w silnikach gier, na przykład jako przeciwnika w rozgrywce szachowej z komputerem [39].

## 3.5 Krzywa ROC

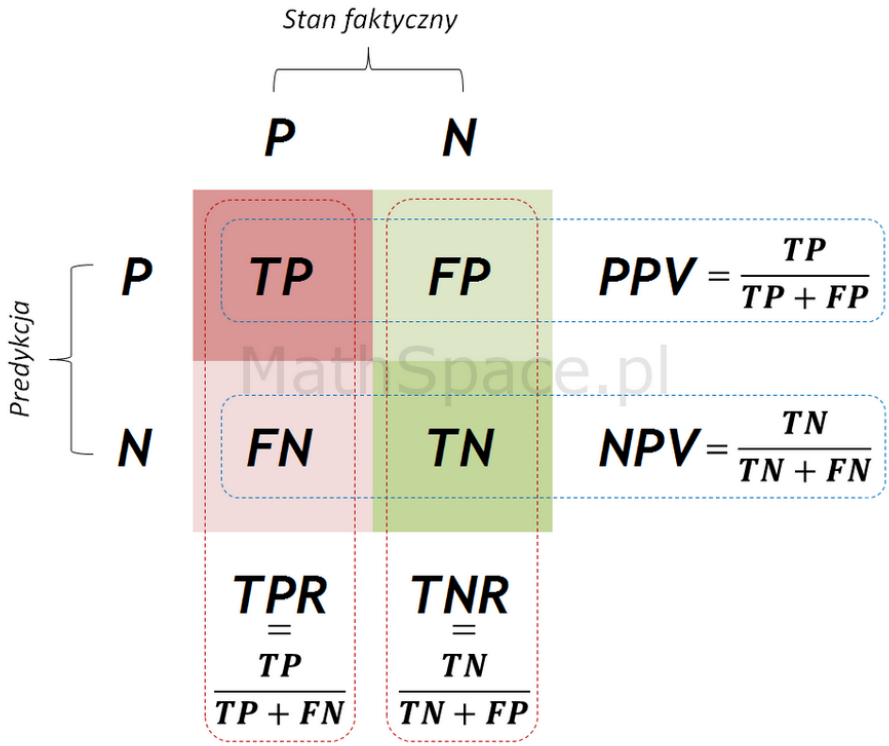
W niniejszym paragrafie zostanie przedstawiony opis Krzywej odbiornika operatora (ROC – reciver operating curve) oraz interpretacja pola pod tą krzywą – AUROC (Arean under ROC).

Przykładowa krzywa została zaprezentowana niebieską linią ciągłą na rysunku 1. Jest ona określana na podstawie współczynników znanych z macierzy błędu przedstawionej na rysunku 2. True Positive Rate, zwany również czułością (ang. sensitivity) jest stosunkiem poprawnych predykcji pozytywnych do sumy niepoprawnych predykcji negatywnych i poprawnych pozytywnych. False Positive Rate, zwany również specyficznością (ang. specificity) jest stosunkiem




*Rys 1: Przykładowa krzywa ROC*

poprawnych predykcji negatywnych do sumy poprawnych predykcji negatywnych i niepoprawnych predykcji pozytywnych. Tworząc krzywą na osi y odkłada się wartość równą 1-specificity [14, 19].




*Rys 2 Macierz błędu [19].*

Za pomocą tej krzywej możemy porównać dwa klasyfikatory binarne lub zbadać wydajność pojedynczego klasyfikatora względem klasyfikatora losowego (oznaczonego na rysunku 1 linią przerywaną. Przez klasyfikator losowy rozumie się przyznawanie kategorii losowo z prawdopodobieństwem 50 %.

Inną ważną informacją jaką możemy uzyskać krzywej ROC jest wartość pola pod tą krzywą. Można ją interpretować jako dokładność badanego modelu (pole całkowite) lub określić o ile lepszy jest on od klasyfikatora losowego którego AUROC wynosi 0.5 [14].

# 4. Drzewa decyzyjne

## 4.1 Początki

Drzewa klasyfikacyjno-regresyjne (ang. CART – classifiacation and regression trees) nie były pierwszymi które zostały użyte w kontekście uczenia maszynowego. Były za to pierwszymi które zostały opisane za pomocą ścisłego formalizmu teorii prawdopodobieństwa oraz ujęte w ramach statystyki. Wczesne ślady ich pochodzenia prowadzą nas w lata 60 XX wieku, a konkretnie do roku 1963 gdzie automatyczny wykrywacz interakcji (AID – automatic interaction detection) został wykorzystany do znajdowania relacji w danych pochodzących z ankiet. Choć bardzo użyteczny nie posiadał żadnego opisu teoretycznego. Skutkowało to, mimo poprawności w działaniu, brakiem zaufania, zwłaszcza jeśli chodzi o przetrenowanie które może prowadzić do błędnych wniosków przy tworzeniu modeli w oparciu o małe próby.  Głównie z tego powodu już w kolejnej dekadzie dwudziestego wieku drzewa zostały niemal całkowicie zapomnianą i nieużywaną powszechnie metodą. Jednak nie wszyscy badacze porzucili to podejście. Richard Olsen wraz Jerome Friedmanem bazując na publikacji Covera i Harta z 1967 traktującej o innym algorytmie uczenia maszynowego, klasyfikacji wykorzystującej najbliższe sąsiedztwo uznali, iż jeżeli możliwe jest poznanie błędu estymacji dla algorytmu najbliższego sąsiedztwa, możliwe też będzie to dla drzew decyzyjnych gdyż każdy węzeł końcowy drzewa można traktować jako dynamicznie tworzony klasyfikator najbliższego sąsiedztwa. To właśnie te badania przyczyniły się do powstania algorytmu który z czasem przekształcił się w drzewa klasyfikacyjno regresyjne. Badacze Ci prowadzili badania w Centrum Liniowego Akceleratora Stanforda (SLAC). W tym samym czasie niezależne badania w Los Angeles prowadzili Leo Breiman oraz Charles Stone. Całość wysiłków tych czterech autorów została wydana w 1984 i datę tę można uznać za narodzony algorytmu drzew klasyfikacyjno regresyjnych [6].

Intuicyjną ewolucją tego algorytmu jest, zapostulowany na przełomie XX i XXI wieku, model losowego lasu. W tym podejściu o wynikach klasyfikacji decyduje nie jedno drzewo ale cały ich zbiór [42].

## 4.2 Działanie klasyfikatora

Na początek wprowadzone zostanie kilka terminów które ułatwią zrozumienie zagadnień związanych z drzewami decyzyjnymi oraz z modelem losowego lasu [40, 41]:

- Węzeł główny (Root node) – jest początkiem całego drzewa, jest dzielony na dwa lub więcej podzbiorów. Rodzic dla wszystkich pozostałych węzłów.
- Liść (Leaf node) – węzeł końcowy, nie może już być podzielony na mniejsze segmenty.
- Rodzic, dziecko (Parent/ child node)– węzeł nazywamy rodzicem jeśli rozgałęzia się na co najmniej dwa węzły. Dziecko to węzeł który posiada rodzica.
- Podział (Splitting) – utworzenie poddrzew z węzła głównego lub innego rodzica na podstawie zadanych kryteriów.
- Gałąź, poddrzewo (Branch, SubTree) – powstaje poprzez podział drzewa/węzła
- Przycinanie (pruning) – jest procesem odwrotnym do podziału. Pomaga w usunięciu niechcianych gałęzi zawierających mało istotne informacje które mogą prowadzić do przetrenowania drzewa.

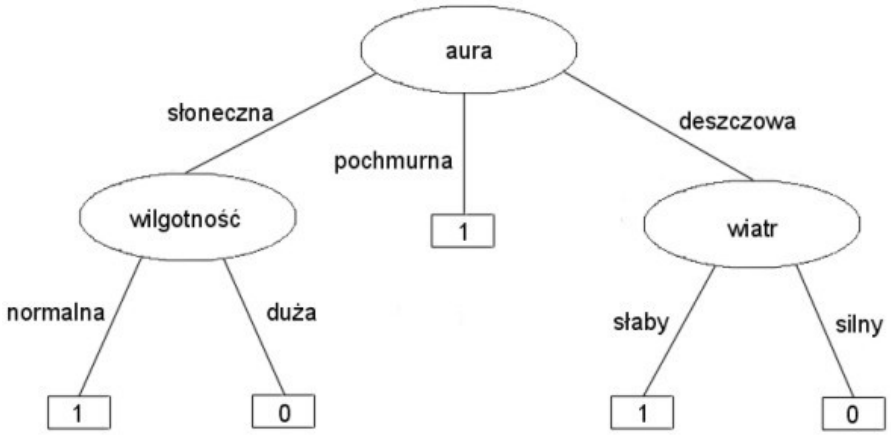
Budowę drzewa rozpoczynamy od węzła głównego, który jest naszym zbiorem do klasyfikacji (lub jego próbką). Wybrana zostaje kategoria która znowu zostaje podzielona na kolejne. Proces ten jest kontynuowany aż do momentu podziału lub gdy uzyskanej gałęzi nie można już podzielić. Problemem jest ustalenie właściwych kryteriów wyboru. Jeśli mamy kilkanaście parametrów które mają prowadzić nas do przyporządkowania danej próbki do jednej z dwóch klas to istotnym jest którą kategorię będziemy dzielić. Celem jest osiągnięcie czystego zbioru, to znaczy zbioru który możemy zaklasyfikować tylko do jednej

klasy. W tym celu obliczany jest zysk informacyjny dla każdej kategorii. Zysk informacyjny jest to zmiana (zmniejszenie) entropii informacyjnej. Entropia jest miarą niepewności (losowości) danych. Pojęcie to zostało wprowadzone w 1948 roku przez Clauda Shannona. Zdefiniowane jest jako:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_r p(x_i),$$

gdzie *H(X)* jest entropią zmiennej losowej *X* o zbiorze wartość $\{x_1, x_2, \ldots, x_n\}$, zaś $p(x_i)$ jest prawdopodobieństwem zajścia zdarzenia $x_i$, natomiast *r* jest podstawą logarytmu. W teorii informacji najczęściej jako podstawę logarytmu stosuje się liczby 2, e oraz 10, odpowiadające im nazwy jednostek informacji to bit, nat i dit [49].

Podczas poruszania się w dół drzewa od węzła głównego zaczynamy z punktu o którym nie wiemy nic, kiedy dotrzemy do liścia wiadomo jest dokładnie do jakiej klasy należy dany punkt. Uściślając więc, poruszając się w dół drzewa na każdym poziomie wymagana jest jak największa redukcja entropii. Taki test jest powtarzany dla każdego atrybutu na każdym węźle. Cecha z największym zyskiem informacyjnym umieszczana jest jako rodzic dla danej gałęzi. Przykładowy sposób budowania drzewa został przedstawiony na rysunku



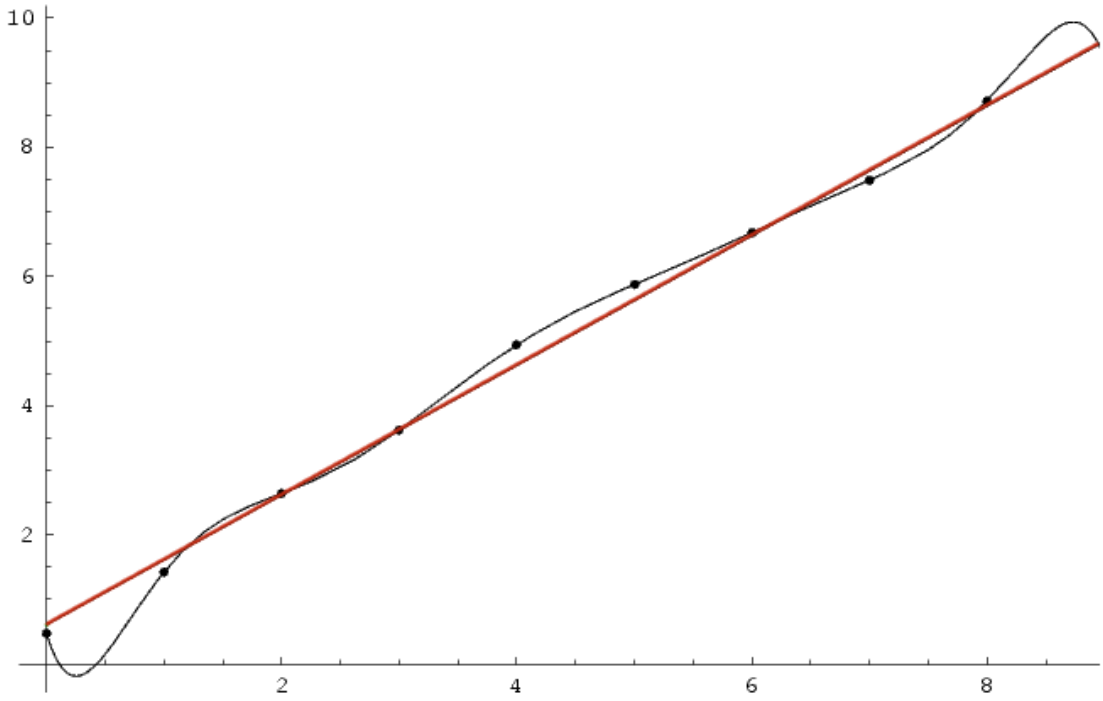

*Rys 3: Przykładowe drzewo decyzyjne odpowiadające na pytanie czy użytkownik wyjdzie na spacer [10].*

3.

Inną liczbą definiującą poprawność klasyfikacji jest zanieczyszczenie Giniego (Gini impurity) informujące jak często węzeł decyzyjny prowadzi do błędnej klasyfikacji [24].

## 4.3 Metody zespołowe – model losowego lasu

Pojedyncze drzewo decyzyjne potrafi poradzić sobie z wieloma złożonymi problemami. Jednakże w wielu przypadkach może okazać się, że doskonała dokładność modelu jest jedynie osiągana na zbiorze trenującym. Taką przypadłość nazwano przeuczeniem lub przetrenowaniem (ang. overfitting). Dzieje się tak ponieważ model podczas treningu traci zdolność do generalizacji i stara się dopasować do nieistniejących regularności będących w rzeczywistości szumem. Może to być spowodowane zbyt długim procesem uczenia lub korzystaniem z mocno zaszumionych danych, lub też ze zbioru który posiada silnie skorelowane cechy [9]. Aby zobrazować istotę przeuczenia posłużymy się przykładem. Na rysunku nr 4 widzimy punkty pomiarowe do których został




*Rys 4: Przykład punktów które mogą być dopasowane za pomocą wielomianiu oraz funkcji liniowej (kolor zielony). Mimo iż wielomian przechodzi przez wszystkie punkty nie można nazwać go najlepszym dopasowaniem [53].*

dopasowany wielomian oraz funkcja liniowa. Mimo iż wielomian przechodzi przez wszystkie punkty lepszym dopasowaniem jest linia prosta. Widać to zwłaszcza przy końcach przedziału które w przypadku wielomianu przyjmują wartości odbiegające od trendu wartości. Taki właśnie efekt uzyskamy gdy algorytm przez nas szkolony świetnie nauczy się zbioru trenującego, ale nie nabędzie zdolności do predykcji nowych danych.

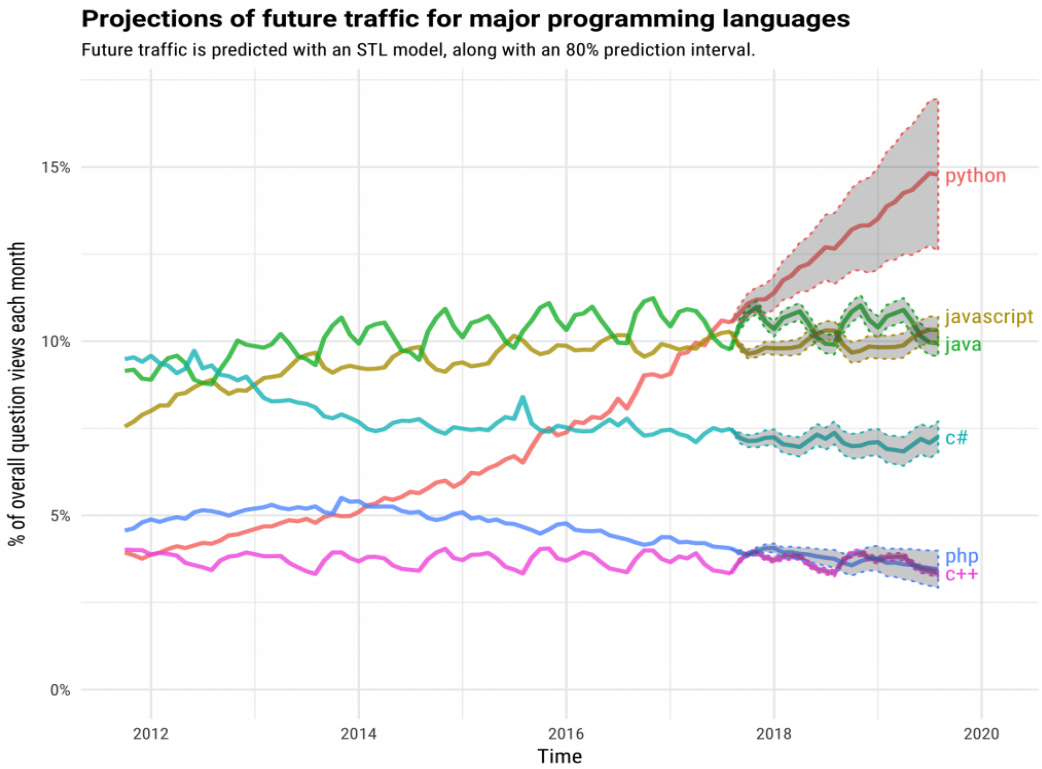
Jednym ze sposobów na poradzenie sobie z tym problemem jest dopasowanie maksymalnej ilości podziałów dla danego drzewa. Skutkuje to bardziej globalnym podziałem drzewa i brakiem podziału na wysoce wyspecjalizowane klasy aczkolwiek może to powodować spadek wydajności.

Aby skompensować te straty możemy stworzyć kilka drzew decyzyjnych i sprawdzić jak zakwalifikowany zostanie dane zdarzenie. Za wartość prawdziwą predykcji bierzemy wtedy tę która została przyznana przez największą liczbę drzew. Tak właśnie działa model lasu losowego [22].
Powoduje to jednak pewne problemy. Jednym z nich jest deterministyczność drzew, te same dane wejściowe spowodują pojawienie się tej samej wartości wyjściowej za każdym razem. Można jednak podzielić zbiór trenujący losując kilka mniejszych co spowoduje powstanie osobnego zbioru dla każdego drzewa, a nawet dla każdego węzła. Procedura ta nazywana jest baggingiem. Jednak w dalszym ciągu możemy spodziewać się że nasze drzewa będą budowane w podobny lub nawet czasem ten sam sposób ponieważ zysk informacyjny dla danej cechy wciąż będzie wysoki. Aby uniknąć tego podobieństwa między drzewami, również w sposób losowy, wybierany jest zbiór cech które będą brane pod uwagę w węzłach danego drzewa. W ten sposób otrzymujemy las losowych drzew decyzyjnych [22, 24]

# 5. Język Python w kontekście uczenia maszynowego

## 5.1 Opis języka Python

Python jest wysokopoziomowym, wieloparadygmatowym, interpretowanym, dynamicznie typowanym językiem programowania [30]. Doskonale sprawdza się jako narzędzie do Rapid Application Development metodologii pozwalającej programiście na podjęcie szerokich działań w dziedzinie prototypowania w krótkim czasie od podjęcia działań. Jest szeroko stosowany jako jako język do łączenie istniejących komponentów razem. Python jest intuicyjny oraz posiada łatwą do przysposobienia składnię. Wspiera modułowość i jest dostępny bezpłatnie na wiele platform oraz co najważniejsze jest udostępniany z otwartym kodem. Powstał we wczesnych latach 90 XX




*Rys 5: Wykres przedstawiający procent zapytań na forum stackoverflow.com dotyczący poszczególnych języków programowania na przestrzeni lat [45].*

wieku, ale znacząco na popularności zyskał w drugiej dekadzie dwudziestego pierwszego stulecia. Na rysunku nr 5 widzimy statystykę ze strony stackoverflow.com – popularnego forum dla programistów zrzeszającego zarówno ekspertów jak i amatorów. Wykresy przedstawiają procentowy udział zapytań dotyczący poszczególnych języków. Jak łatwo zauważyć python w okolicach 2017 roku przegonił w tym zestawieniu język Java. W modelu nie zostały uwzględnione CSS, HTM, Andorid , Jquery [45].

## 5.2 Moduły – główna siła Pythona

Bezsprzeczną siła pythona jest ilość modułów które zapewniają dopasowanie języka niemal do każdego problemu. Najważniejsze z nich to [4]:

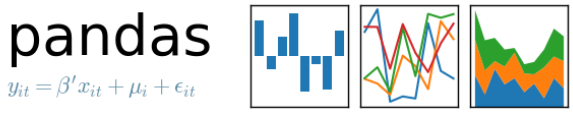
- NumPy – biblioteka zawiera implementację szybkich tablic, które są podstawowym typem danych używanym przez biblioteki naukowe i uczenia maszynowego. Pomaga w obsłudze wielowymiarowych tablic poprzez szeroką gamę matematycznych funkcji. Charakteryzuje się wysokim stopniem optymalizacji, dzięki czemu skomplikowane operacje na macierzach odbywają się w krótkim czasie [28].

*Rys 6: Logo biblioteki NumPy [28].*

- SciPy – jest pewnym rozszerzeniem funkcjonalności pakiety NumPy. Oferuje większy zbiór algorytmów naukowych z zakresu algebry liniowej, przetwarzania sygnału i obrazu, zagadnień optymalizacyjnych czy szybkiej transformaty Fouriera [4, 31].

*Rys 7: Logo SciPy [31].*

- Pandas – Pandas wypełnia wszystkie luki które



*Rys 8: Logo pandas [29].*

pozostawiły dwa powyższe moduły. Dzięki posiadaniu specjalnego strukturyzowania danych (Data Frame, Series) pozwala na obsługę złożonych tablic danych różnych typów. Za pomocą tego modułu można małym wysiłkiem ładować dane z różnych źródeł, obsługiwać brakujące elementy, zmieniać nazwy i wymiary oraz wizualizować dane [29].

- Scikit-learn – jest pythonowym rdzeniem jeśli chodzi o naukę o danych. Biblioteka dostarcza narzędzi potrzebnych wstępnego przetwarzania danych, uczenia nadzorowango i nadzorowanego, walidacji, metryk błędu [32].

*Rys 9: Znak graficzny scikit-learn [32].*

- Matplotlib – biblioteka dostarcza funkcjonalność tworzenia przejrzystych, jakościowych wykresów [27, 40].

*Rys 10: Logo biblioteki Matplotlib [27]*

## 5.3 Portal Kaggle

Kaggle jest platformą na którą warto zwrócić uwagę przy wyborze miejsca do szukania ciekawych zbiorów danych z różnych dziedzin. Dodatkowo pozwala na stawanie w szranki z innymi entuzjastami eksploracji danych i metod sztucznej inteligencji w ramach konkursów których pula nagród nieraz sięga kilkudziesięciu tysięcy dolarów. Problemy które są stawiane uczestnikom muszą spełniać restrykcyjne kryteria:

- Zagadnienie musi być trudne – ma to zachęcać firmy do przekazania swoich problemów w ręce specjalistów.

- Rozwiązanie powinno być innowacyjne – z uwagi na pierwsze kryterium opracowanie odpowiedniego podejścia wymaga sporego zaangażowania co skutkuje znalezieniem podejścia którego jeszcze nikt nie użył

- Jakość rozwiązanie powinna być mierzalna – konkursy są po to aby wyłonić zwycięzcę dlatego musi istnieć możliwość zmierzenia o ile lepsze rozwiązanie przedstawił zwycięzca względem pozostałych uczestników. Dodaje to dodatkowego charakteru zmaganiom [37].

# 6. Fizyka cząstek elementarnych

W niniejszym rozdziale zostanie przedstawiony krótki rys historyczny dotyczący poglądu na materię na przestrzeni wieków.

Próby wyjaśnienia zjawisk zachodzących w przyrodzie oraz materii podejmowali już starożytni Grecy. Uważali że materia zbudowana oraz oddziaływania można przedstawić za pomocą czterech żywiołów: ognia, wody, ziemi i powietrza oraz powiązanych z nimi cech: ciepła, zimna, wilgotności i suchości. Transformacje mogły zachodzić między żywiołami poprzez utratę lub zyskanie pewnej cechy. Tę hipotezę można uznać za pierwszą próbę unifikacji sił przyrody. Inna szkoła filozofów postulowała że cały świat składa się z niepodzielnych cząstek – atomów [35]. Pogląd ten jest znacznie bliższy faktom. Aktualne znaczenie słowa atom różni się od tego co przez to słowo rozumieli starożytni. Wierzyli oni że na pewnym poziomie możemy spotkać taką cegiełkę której już nie będzie można podzielić.

W wiekach średnich badano oddziaływania ciał naelektryzowanych, równanie stanu gazu czy rozpuszczanie się ciał stałych. Sporą sławą cieszyli się alchemicy którzy próbowali stworzyć kamień filozoficzny, artefakt który każdą substancję mógł przemienić w złoto.

W kolejnych latach zaczęto dokładnie badać otaczające nas zjawiska i sporządzać teorię te doświadczenie opisujące. Dominował nurt który próbował wyjaśnić nieintuicyjne zjawiska za pomocą ludzkiego rozumu raczej wyzbywając się wyjaśnień opartych na odwoływaniu się do sił nadprzyrodzonych.

Do końca XIX wieku fizyka zyskała teorię opisującą zachowanie obiektów w wielkiej skali. Za znany i zbadany uznawany był również elektromagnetyzm. Udało się potwierdzić istnienie elektronu. Rozumiano przepływ ciepła i zachowanie gazów. Jak powiadał Philipp von Jolly, wykładowca na monachijskim uniwersytecie : “w tym polu [fizyce], prawie wszystko zostało już odkryte i jedyne co pozostało to wypełnić kilka nieistotnych ubytków” człowiek do którego skierowane zostały te słowa nazywał się Max Planck – dzisiaj wymieniany jako jeden z ojców mechaniki

kwantowej – dziedziny fizyki która pokazała jak wiele jest jeszcze do zrobienia [1, 18].

## 6.1 Odkrycie kwantów

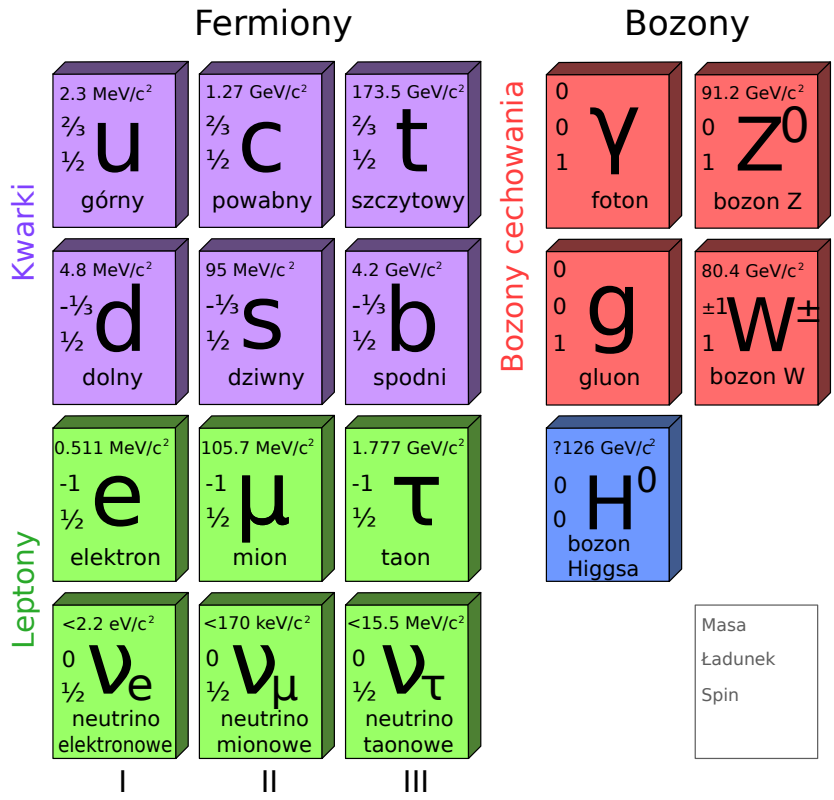
14 grudnia 1900 - ten dzień uznawany jest za narodziny mechaniki kwantowej. Nauki która z jednej strony wyjaśniła wiele sekretów przyrody oraz pozwoliła (i wciąż pozwala) na niebywały postęp technologiczny. Z drugiej strony stała się ikoną najbardziej skomplikowanej nauki i przyczyną wielu łez wylanych przez studentów fizyki. W tym dniu Max Planc na zebraniu Niemieckiego Towarzystwa Fizycznego próbując wyjaśnić widmo promieniowania ciała doskonale czarnego zapostulował istnienie kwantów i powołał do życia nową stałą fizyczną zwaną później stałą Plancka [2]. W tym czasie Planck jeszcze nie wiedział że otwiera nowy rozdział w erze badań nad otaczającym nas światem. Kolejne odkrycia następowały bardzo szybko. W ciągu kilkudziesięciu lat fizyka kwantowa odpowiedziała nie tylko na to jak promieniuje ciało doskonale czarne, ale również jakie jest prawdopodobieństwa tunelowanie cząstki, dlaczego czarne dziury jednak mogą parować, podała model kwarkowy materii oraz prognozę długości życia Słońca [1, 17].

## 6.2 Model Standardowy

Istotą kwantowej teorii pola jest wyjaśnianie efektów zachodzących w bardzo małej skali. Stara się ona odpowiedzieć jakie obiekty za pomocą jakich sił oddziałują ze sobą. Efektem tych dociekań stał się Model Standardowy – teoria fizyczna opisująca materię, zarówno tę która buduje otaczający nas świat jak i tę która powstała w bardzo wczesnym etapie ewolucji wszechświata i obecnie jest badana w laboratoriach fizyki cząstek.

Model Standardowy obejmuje trzy generacje cząstek Jednak wszystkie cząstki potrzebne do zbudowanie otaczającej nas materii są ujęte w obrębie pierwszej generacji. Wszystko co widzimy zbudowane jest z atomów, te posiadają neutronowo-protonowe jądra które z kolei składają się z kwarków górnych i dolnych. Dopełnieniem tego obrazu jest elektron oraz odpowiadające

mu  neutrino – neutrino elektronowe. Ten właśnie komplet – kwark dolny, kwark górny, elektron, neutrino elektronowe tworzą pierwszą generację cząstek.




*Rys 11: Model Standardowy [16].*

Pozostałe dwie generacje są bardzo podobne do pierwszej. Generację drugą tworzą kwark powabny i dziwny, mion i jego neutrino. Generacja trzecia to kwark wysoki i niski oraz taon i neutrino taonowe.

Oprócz cząstek mamy również siły która propagują się poprzez wymianę bozonów – są to cząstki o spinie całkowitym. Jak wspomniałem wyżej neutrony oraz protony są zbudowane z kwarków – górny oraz dolny. Kwarki są utrzymywane razem za pomocą siły kolorowej – przejawu oddziaływania silnego, polega to na wymianie gluonów. Ta interakcja jest o tyle ciekawa że jej natężenie rośnie wraz ze wzrostem odległości pomiędzy kwarkami. Nazywamy to asymptotyczną swobodą kwarków. Przyczynia się do tego, że nie możemy obserwować swobodnych kwarków czy gluonów, a jedynie taką ich kombinację która w efekcie daje kolor biały.

Elektron znajdujący się w otoczeniu jądra oddziałuje z nim elektromagnetycznie. Kwatnem tego oddziaływania są fotony. Zaś za przemianę neutronów w protony oraz rozpady promieniotwórcze odpowiada oddziaływanie słabe którego kwantami jest jeden z ciężkich bozonów: $W^+$ $W^-$, $Z^0$
Całości formalizmu dopełnia pole Higgsa wraz ze swoim bozonem zwanym przez niektórych Boską Cząstką. Pole to odpowiada za nadawanie masy bozonom W i Z oraz w pewnej części leptonom [1, 18, 44].

## 6.3 Neutrina

Koncepcja neutrina zostało po raz pierwszy zaproponowana w 1930 roku przez Wolfganga Pauliego jako wyjaśnienie ciągłego rozkładu energii promieniowania beta. Początkowe hipotezy zakładały nawet niezachowanie energii w tego typu rozpadach. Pauli chciał rozwiązać ten problem postulując istnienie neutralnej i głęboko penetrującej cząstki. Zakładał że taka hipotetyczna cząstka powinna mieć bardzo małą masę, około 0.01 masy protonu, posiadać połówkowy spin przez co podlegałaby zakazowi Pauliego. Przewidywał że moment magnetyczny takiej cząstki wynosi około 0.02 magnetonu Bohra oraz zdolność penetracyjną co najmniej dziesięć razy większą niż promieniowanie gamma. Nie opublikował jednak swoich rozważań. Zrobił to w 1933 Enrico Fermi, nazywając nową cząstkę neutrinem, odróżniając ją w ten sposób od odkrytego rok wcześniej neutronu [15].

Rok później Hans Bethe oraz Rudolf Peierls jako pierwsi oszacowali przekrój czynny rozpraszania neutrin na nukleonach umiejscawiając go w przedziale $\sigma<10^{-44} cm^2$ dla wiązki neutrin o energii 2.3 MeV. Wywnioskowali zatem że nie ma praktycznego sposobu detekcji – aby zatrzymać neutrio o energii 1 MeV należy użyć zbiornika z wodą o długości 50 lat świetlnych [47]. Wtedy, jak i aktualnie, żadne laboratorium nie dysponowało takim sprzętem. Bethe obliczył również przekrój czynny na rozpraszanie naładowanych cząstek poprzez oddziaływanie magnetyczne. Jednakże okazało się że ustalona przez Pauliego górna granica momentu magnetycznego jest znacznie przeszacowana

przez co estymowany przekrój czynny takiego zdarzenia byłby czternaście rzędów wielkości mniejszy niż zakładano. Były to początki intensywnych badań nad tymi bardzo tajemniczymi ale też bardzo istotnymi dla fizyki wysokich energii cząstkami.

Wyróżniamy dwa typy interakcji neutrin. Mogą one sprzęgać do neutralnego bozonu pośredniczącego oddziaływań słabych – $Z^0$ – reakcja ta nazywana jest prądem neutralnym, zachowuje ona liczby kwantowe ale przenosi pęd, spin oraz energię. Mogą również sprzęgać do naładowanych bozonów cechowania oddziaływania słabego – $W^{+/-}$ - ten typ nazywany jest prądem naładowanym – może zmieniać liczby kwantowe – w tym ładunek elektryczny. Kolejnym odmiennym i bardzo ciekawą własnością neutrin jest ich spin. Okazuje się że antyneutrina nie mają spinu o przeciwnym znaku, ale o odmiennej skrętności. Wszystkie neutrina są lewoskrętne podczas gdy wszystkie nuetrina są prawoskrętne. Dlatego też neutrina są rozważane jako tak zwane fermiony Majorany co oznacza iż postuluje się, że ich antycząstki są tak naprawdę nimi samymi ale w innym stanie kwantowym [47].

## 6.4 Pochodzenie Neutrin

Neutrina kosmologiczne zwane również neutrinami reliktowymi. Pochodzą z bardzo wczesnego etapu wszechświata. Dopóki temperatura wynosiła więcej niż $10^{10}$ K neutrina były w równowadze termicznej z fotonami. Poniżej temperatury nie zachodziła już reakcja $\gamma\gamma \rightarrow \nu_\alpha \bar{\nu}_\alpha$, w związku z czym neutrina zostały „uwolnione”. Aktualnie temperatura neutrin reliktowych jest na poziomie około 1.95 K.

Innym źródłem neutrin które nie leży w obrębie układu słoneczne są supernowe typu II. Są to gwiazdy którym skończyło się paliwo do podtrzymywania reakcji nuklearnych. Ostatnim syntezowanym w ten sposób pierwiastkiem jest żelazo które nie może zostać przekształcone z żaden cięższy pierwiastek. Skutkuje to wytworzeniem się żelazowego jądra otoczonego przez coraz lżejsze elementy. Gwiazda tuż przed swoją śmiercią przypomina nieco cebulę. Kiedy masa takiego jądra przekroczy granicę Chandrasekhara staje się

bardzo niestabilne i w ciągu ułamków sekundy zapada się. Podczas kolapsu gwałtownie rośnie ciśnienie czego wynikiem jest wtłaczanie elektronów do jąder atomowych gdzie następuje wychwyt tychże elektronów na protonach, produktem tej reakcji, zwanej neutralizacją jest neutron oraz antyneutrino elektronowe:

$$e^- + p \rightarrow n + \bar{\nu}_e$$

Około 99% energii kolapsu jest emitowane w postaci neutrin wszystkich rodzajów. Inne rodzaje neutrin są produkowane w bardzo gorących obszarach na drodze reakcji $\gamma\gamma \rightarrow \nu_\alpha \bar{\nu}_\alpha$ . Na powstały w ten sposób obiekt spadają zewnętrzne warstwy gwiazdy które zderzając się falą neutrin oraz z powierzchnią gwiazdy wyzwalają wielką energię którą nazywamy wybuchem supernowej. Jeśli masa będącej w centrum gwiazdy neutronowe jest większa niż około 3 mas Słońca, to grawitacja takiego obiektu będzie silniejsza niż ciśnienie materii neutronowej – dojdzie do dalszego zapadania w wyniku którego powstanie czarna dziura [34].

Największą oraz najbliższą nam fabryką neutrin jest Słońce. W jego wnętrzu zachodzi reakcja zwana cyklem protonowym. Jest to dominujący czynnik jeśli chodzi o generację energii najbliższej nam gwiazdy. Schematycznie możemy zapisać ją następująco:

$$4\,p \rightarrow \alpha + 2\,e^+ + 2\,\nu_e + Energia$$

W wyniku połączenia czterech protonów powstaje jądro helu (cząstka alfa), dwa pozytony (antycząstki elektronu), dwa neutrina elektronowe oraz energia. Z racji na słabe interakcje z materią neutrina te dochodzą do ziemi bezpośrednio zw wnętrza naszej gwiazdy. Można powiedzieć że możliwe jest oglądanie Słońca w spektrum neutrin. Same neutrina solarne, jak się je nazywa, przysporzyły naukowcom sporą zagwozdkę. Z obliczeń uczonych wynikało że w okolicach jądra słonecznego gdzie przebiegają reakcje termojądrowe w ciągu sekundy przetwarzane jest około sześćset milionów ton zjonizowanego wodoru. Powinno to skutkować silnym strumieniem neutrin elektronowych. Jednak nie potwierdzały tego obserwacje. Rozkład odbieranych neutrin był równomierny dla każdego typu. Wyjaśnieniem tej zagadki było odkrycie zjawiska zwanego

oscylacją neutrin. Podróżując w przestrzeni neutrina mogą zmieniać się swobodnie w dowolne inne neutrino co skutkuje wyrównaniem dystrybucji wykrywanych w ziemskich obserwatoriach neutrin. Odkrycie to zostało uhonorowane Nagrodą Nobla [21].

Naturalna promieniotwórczość pierwiastków zawartych głównie w skorupie ziemi odpowiada za około 40 % energii generowanej przez naszą planetę. Reakcjom tym towarzyszy emisja neutrin których strumień jest rzędu $10^6$ $(cm^2\ s)^{-1}$ i jest zależny od grubości skorupy ziemskiej. Neutrina powstają również w górnych warstwach ziemskiej atmosferze gdzie na skutek zderzeń wysokoenergetycznych cząstek pochodzenia pozaziemskiego z atomami dochodzi do rozpadów którym towarzyszy emisja neutrin.

Ostatnim wartym wzmianki typem neutrin są te które człowiek generuje w reaktorach oraz akceleratorach cząstek podczas zderzenia ich we wnętrzu detektorów [15, 47].

## 6.5 Detekcja neutrin w eksperymencie OPERA

Eksperymnent OPERA (Oscillation Project with Emulsion-tRacking Apparatus) jest usytuowany w środkowych Włoszech w Narodowym Laboratorium Gran Sasso [26]. Jednak neutrina które są wykrywane w tym detektorze produkowane są 732 kilometry na północny-wschód Europejskim Laboratorium Fizyki Cząstek pod Genewą. Początkiem drogi protonów które posłużą jako czynnik inicjujący powstanie strumienia neutrin jest butla z wodorem. Pierwszy pierwiastek z tablicy Mendelejewa posiada jeden elektron który łatwo można odłączyć od jądra atomowego dzięki czemu uzyskujemy protony które są wstrzykiwane do liniowego akcelratora który rozpędza je do początkowej energii 50 MeV, następnie wiązka jest przyspieszana przez Booster – najmniejszy spośród kołowych akceleratorów w CERN, wiązka uzyskuje tu energię 1.4 GeV. W dalszej kolejności porcja protonów trafia do Synchrotrona Protonowego gdzie jest przyspieszana do energii 14 GeV [38] skąd wędruje do Super Synchrotrona Protonowego (SPS – Super Proton Syncrotron). Opuszczając pierścień SPSa wiązka ma energię 400 GeV i zostaje skierowana na

tarczę zbudowaną z grafitu. W wyniku reakcji protonów z jądrami atomów węgla zostają wyprodukowane piony oraz kaony które rozpadają się uwalniając neutrina [15, 23].

Zdarzenia w detektorze są wykrywane poprzez wykorzystanie technik detekcji w czasie rzeczywistym (detektory elektroniczne) oraz poprzez odtworzenie śladów zapisanych w emulsji jądrowej (ECC – Emulsion Cloud Chamber). Podstawową jednostką budulcową detektora jest cegiełka złożona z 56 płyt pomiędzy którymi znajdują się cienkie warstwy emulsji jądrowej. W całym detektorze znajduje się 150 000 takich jednostek które składają się na masę 1 250 000 kilogramów. Tak duża masa jest konieczna do osiągnięcia odpowiedniej czułości oraz liczby interakcji [23, 38].

*Rys 12: Detektor eksperymentu OPERA [46].*

# II Część praktyczna

# 7. Przebieg analizy

Pierwszym krokiem analizy jest pobranie danych z portalu *Kaggle.com*. Uzyskane dane są zapisane w pliku w formacie csv (comma seperated values) o rozmiarze 1 GB. Zawierają 9 806 386 wierszy oraz 9 kolumn. Pobrane dane zostaną wczytane przy użyciu biblioteki *pandas*. Korzystając z pakietów *mpl_toolkits* oraz *matplotlib* zostaną zwizualizowane ślady cząstek. Następnie zbiór danych zostanie podzielony na podzbiór uczący oraz testowy. Dane treningowe zostaną przekazane do funkcji pakietu *sklearn* odpowiedzialnej za szkolenie. Korzystając z części testowej zostanie wykreślona krzywa operacyjna odbiornika pozwalająca określić dokładność modelu. Wizualizacja drzewa zostanie wyeksportowana do pliku png za pomocą pakietu *graphviz*. Moduł *tqdm* pomoże w estymowaniu czasu wykonywania części kodu umieszczonych w pętlach.

## 7.2 Wizualizacja

Zbiór danych możemy podejrzeć w programie umożliwiającym ładowanie oraz formowanie plików csv w tabele. Na rysunku 13 możemy zobaczyć jak wyglądają dane. Warto wspomnieć że zostały one wygenerowane

| index | event_id | X | Y | Z | TX | TY | chi2 | signal | brick_number |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -999 | 66162.84375 | 65620.03125 | 0.0 | 0.16688987612700001 | 0.27667412161800004 | 2.98871397972 | 0.0 | 44 |
| 1 | -999 | 43900.4453125 | 67091.671875 | 0.0 | -0.179966524243 | -0.478794634342 | 1.90141999722 | 0.0 | 98 |
| 2 | -999 | 37564.0195312 | 26618.890625 | 55599.0 | -0.26781994104400003 | -0.282087057829 | 2.5102450847599997 | 0.0 | 64 |
| 3 | -999 | 39380.0273438 | 48135.4257812 | 19395.0 | 0.26519715786 | -0.577492535114 | 1.21521890163 | 0.0 | 63 |
| 4 | -999 | 65040.78125 | 44018.46875 | 12930.0 | -0.711160719395 | -0.245591521263 | 2.9736614227300002 | 0.0 | 2 |
| 5 | -999 | 55179.875 | 24942.84375 | 41376.0 | -0.173363089561 | -0.31843379139899997 | 2.48823952675 | 0.0 | 68 |
| 6 | -999 | 55329.1796875 | 58168.703125 | 6465.0 | -0.11945684254200001 | -0.07142857462169999 | 2.05435395241 | 0.0 | 60 |
| 7 | -999 | 29075.796875 | 23138.0839844 | 7758.0 | -0.410165548325 | 0.224944189191 | 2.0932071209 | 0.0 | 70 |
| 8 | -999 | 61468.56640619999 | 69761.6171875 | 1293.0 | -0.31194195151299997 | 0.31856399774599997 | 2.7136926651 | 0.0 | 30 |
| 9 | -999 | 47745.734375 | 14015.421875 | 58185.0 | 0.33418899774599997 | -0.21645274758299998 | 2.77924323082 | 0.0 | 51 |
| 10 | -999 | 46175.921875 | 53356.5117188 | 49134.0 | 0.18264508247400002 | 0.19585193693599998 | 1.09449219704 | 0.0 | 19 |
| 11 | -999 | 69591.375 | 5571.828125 | 24567.0 | 0.035342261195199995 | -0.330594301224 | 2.13952589035 | 0.0 | 66 |
| 12 | -999 | 49481.703125 | 46143.875 | 25860.0 | 0.324944198132 | 0.251060277224 | 2.99730944633 | 0.0 | 26 |
| 13 | -999 | 33727.87890620001 | 55500.921875 | 46548.0 | 0.25250184536 | 0.468154758215 | 0.766934335232 | 0.0 | 94 |
| 14 | -999 | 70301.171875 | 56099.3203125 | 65943.0 | 0.25308778882 | -0.547433018684 | 2.2599773407 | 0.0 | 43 |
| 15 | -999 | 58960.609375 | 8119.7734375 | 54306.0 | 0.358333319426 | -0.45486420393 | 2.47038221359 | 0.0 | 20 |
| 16 | -999 | 68376.421875 | 60203.48046880001 | 16809.0 | -0.0497023798525 | -0.395796120167 | 1.99797737598 | 0.0 | 6 |
| 17 | -999 | 47896.828125 | 33607.9765625 | 65943.0 | 0.0786086320877 | -0.35154390335100005 | 0.391841590405 | 0.0 | 31 |
| 18 | -999 | 70269.796875 | 11935.050781200001 | 1293.0 | -0.46309524774599997 | 0.15234375 | 2.27036046982 | 0.0 | 75 |

*Rys 13: Kilkanaście pierwszych wierszy zbioru pobranego z portalu Kaggle.com. Na niebiesko zaznczoan została zmienna docelowa.*

przy użyciu metody Monte Carlo. Nazwy kolumn mają następujące znaczenie:

- **event_id** – numer identyfikacyjny śladu, w przypadku rekordu z sygnałem występuje wartość inna niż -999
- **X** – pozycja x śladu
- **Y** – pozycja y śladu

- **Z** – pozycja z śladu
- **TX** – kąt w płaszczyźnie XZ
- **TY** – kąt w płaszczyźnie YZ
- **chi2** – zgodność dopasowanie linii śladu do punktu początkowego śladu.
- **signal** – 1 gdy wiersz jest sygnałem (częścią kaskady), 0 w przeciwnym wypadku
- **brick_number** – numer cegiełki w której doszło do zdarzenia

W celu załadowania danych do skryptu oraz dalszej ich obróbki niezbędne jest załadowanie potrzebnych modułów.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

from mpl_toolkits.mplot3d.art3d import
Line3DCollection

from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.tree import export_graphviz
from subprocess import call

from time import time
from tqdm import tqdm
```

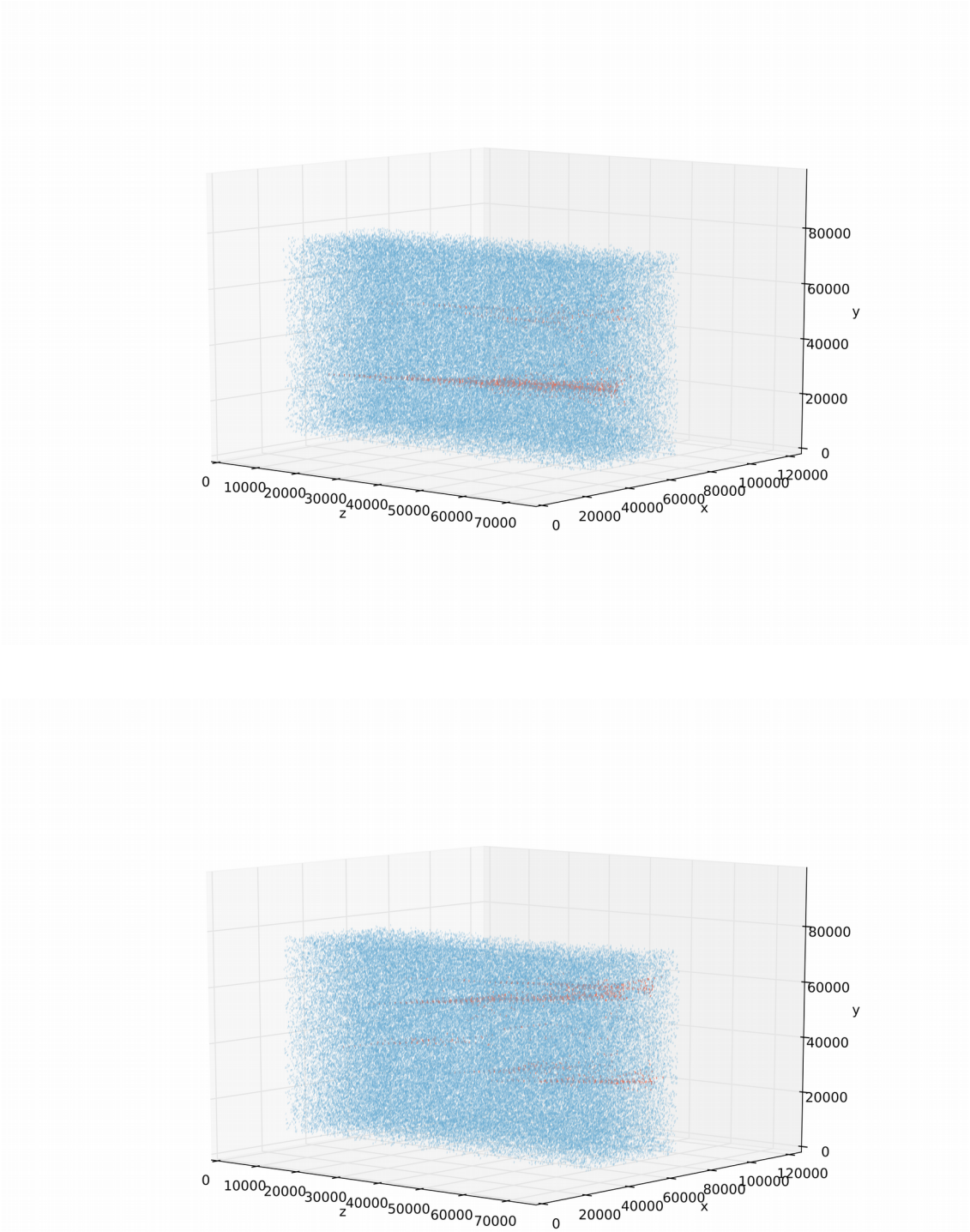
Dobrym początkiem eksploracji danych jest próba ich zrozumienia. Wyjątkowo pomocną metodą jest wizualizacja danych. Jest to dobra praktyka jeśli dane mają interpretację fizyczną i nie mają zbyt wielu stopni swobody. Próba przedstawienia na pojedynczym wykresie zbioru z wymiarowością większą niż dziesięć może nas raczej zniechęcić do dalszej pracy. Aby zwizualizować dane, została napisana funkcja która biorąc koordynaty przestrzenne oraz kąty wyznacza linię która jest obrazem toru po którym poruszała się cząstka w danej warstwie detektora. W celu narysowanie takiej linii konieczne jest podanie punktu początkowego oraz końcowego. Punkty końcowe zostały wyliczone na podstawie kątów oraz arbitralnie przyjętej grubości pojedynczego plasterka w cegiełce. Parametrem tym można sterować przez co zmienia się zagęszczenie wykresu. Osobno wyliczane są ślady dla danych które zawierają informację o kaskadzie osobno zaś dla danych tła. W przypadku obliczania linii śladów tła możemy jeszcze określić parametr step, domyślnie ustawiony na jeden. Przy zmianie tej wartości na inną liczbę całkowitą część śladów tła zostanie pominięta i nie będzie rysowana na wykresie dzięki czemu obraz sygnału stanie się wyraźniejszy. Kod wygląda następująco:

```
def plot_bg_and_mc(pbg, pmc, step=1, dZ=205):
    df = pbg
    d0 = pd.DataFrame([
                df['Z'][::step],
                df['X'][::step],
                df['Y'][::step]],
                index=['z', 'x', 'y']).T
    numtracks = d0.shape[0]
    dd = pd.DataFrame([
            df['TX'][::step]*dZ,
            df['TY'][::step]*dZ],
            index=['x', 'y']).T
    dd.insert(loc=0, column='z', value=dZ)
    d1 = d0 + dd
    C = plt.cm.Blues(0.5)
    lc_bg = Line3DCollection(list(zip(d0.values,
     d1.values)), colors=C, alpha=0.3, lw=2)
```

```
df = pmc
d0 = pd.DataFrame([
            df['Z'],
            df['X'],
            df['Y']],
            index=['z', 'x', 'y']).T

dd = pd.DataFrame([
        df['TX']*dZ,
        df['TY']*dZ],
        index=['x', 'y']).T
dd.insert(loc=0, column='z', value=dZ)
d1 = d0 + dd

C = plt.cm.Reds(0.5)
lc_mc = Line3DCollection(list(zip(d0.values,
 d1.values)), colors=C, alpha=0.7, lw=2)

fig = plt.figure(figsize=(12,8))
ax = fig.gca(projection='3d')
ax.view_init(azim=-50, elev=10)
ax.add_collection3d(lc_mc)
ax.add_collection3d(lc_bg)

ax.set_xlabel("z")
ax.set_ylabel("x")
ax.set_zlabel("y")
ax.set_xlim(0, BRICK_Z)
ax.set_ylim(0, BRICK_X)
ax.set_zlim(0, BRICK_Y)
```

W wywołaniu funkcji jako argumenty podajemy dane z konkretnej cegiełki zawierające sygnał oraz tło:

```
plot_bg_and_mc(
df[np.logical_and(df.brick_number==17, df.signal == 0)],
df[np.logical_and(df.brick_number == 17, df.signa == 1)])
plt.show()
```




*Rys 14: Przykładowe wizualizacje śladów. Niebieski kolor oznacza tło, pomarańczowy - sygnał.*

## 7.2 Parametryzacja

Implementacja algorytmu w scikit learn wymaga odpowiedniego dobrania parametrów. Część z nich w sposób znaczący wpływa na szybkość działania. Poniżej przedstawiono niektóre z nich :

**n_estimators** : liczba całkowita, domyślnie ustawiona na dziesięć. Ten argument funkcji parametryzuje ilość drzew w modelu lasu losowego.

**criterion** : kryterium zatrzymania. Domyślnym sposobem wprowadzenia węzła decycyjnego jest mierzenie zanieczyszczenie Giniego, inną możliwą opcją jest mierzenie zysku informacyjnego. Parametr ten nie był zmieniany

**max_depth** : liczba całkowita opisująca maksymalną głębokość drzewa - liczbę węzłów decyzyjnych między węzłem głównym a liściem.

**bootstrap** : parametr typu bool. Pozwala na użycie techniki bootstrapingu co ułatwia uniknięcie przetrenowania algorytmu.

**oob_scores** : parametr typu bool. Umożliwia walidację podczas trenowania poprzez obliczanie średniego błędu dla każdej próbki bez użycia próbki uczącej.

**n_jobs** : w metodach które używają zespołu algorytmu praca może być w łatwy sposób zrównoleglona. Parametr n_jobs pozwala określić ilość rdzeni które chcemy użyć do przeprowadzenia trenowania algorytmu.

**random_state** : ziarno generatora liczb pseudolosowych

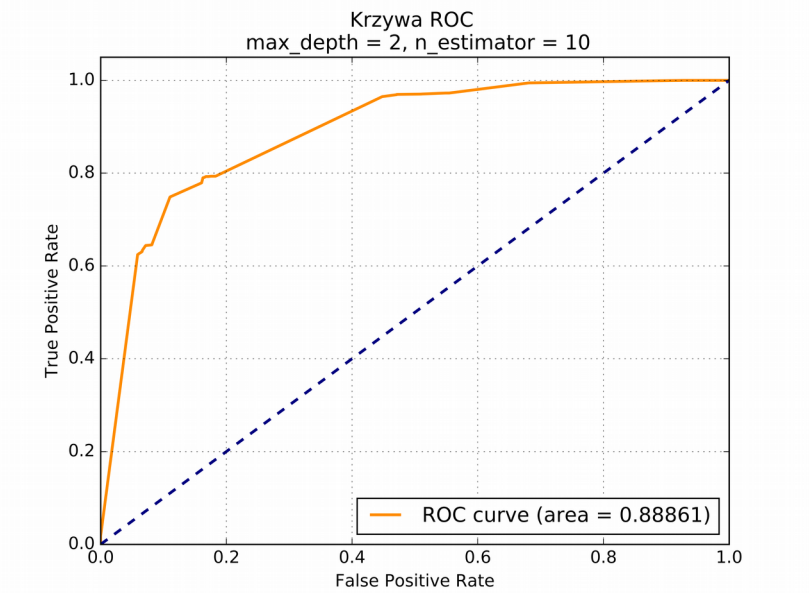
Główną metodą klasy RandomForrestClassifier jest funkcja f*it()*. Jako argumenty przyjmuje cechy danej klasyfikacji oraz jej zmienne docelowe. W naszym przypadku klasy są dwie, sygnał i tło, jest to więc klasyfikator binarny. Wcześniej, w inicjalizacji klasyfikatora zostają ustawione parametry o których wspomniano wyżej. Zestawem bardzo ułatwiającym weryfikację poprawności jest moduł *metrics* z pakietu scikit-learn. W ramach tego modułu możemy wyrysować krzywą informująca nas o ile nasz model jest lepszy od losowego przyznawania kategorii. W takim wypadku powinniśmy mieć 50% prawdopodobieństwo przyznania każdej z klas. Jednak niekoniecznie będzie to dobre oszacowanie. Taka krzywa nazywana jest Krzywą Operacyjną Odbiornika

(ROC – reciver operator curve). Ważnym parametrem jest pole pod tą krzywą – informacja jaką niesie mówi o dokładności modelu [20, 32]. Ten parametr silnie zależy od ilości drzew w module uczącym oraz maksymalnej ilości podziałów drzewa. Kod który dopasowuje dane oraz rysuję krzywą operacyjną odbiornika wygląda następująco:

```
X_train, X_test, y_train, y_test =
train_test_split(df.iloc[:,1:7], np.array(df.iloc[:,-2],
dtype=int), test_size=0.4, random_state=101)


clf = RandomForestClassifier(max_depth = 7, n_estimators =
16, n_jobs = -1, oob_score=True)

clf.fit(X_train.values, y_train)


pred = clf.predict_proba(X_test.values)

fpr, tpr, thresholds = metrics.roc_curve(y_test,
pred[:,1], pos_label=1)

roc_auc = metrics.auc(fpr, tpr)

plt.figure()

lw = 2

plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC
curve (area = %0.5f)' % roc_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=lw,
linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.grid(True, alpha=0.6)

plt.ylabel('True Positive Rate')

plt.title('Krzywa ROC \n max_depth = {}, n_estimator =
{}'.format(7, 16))
```

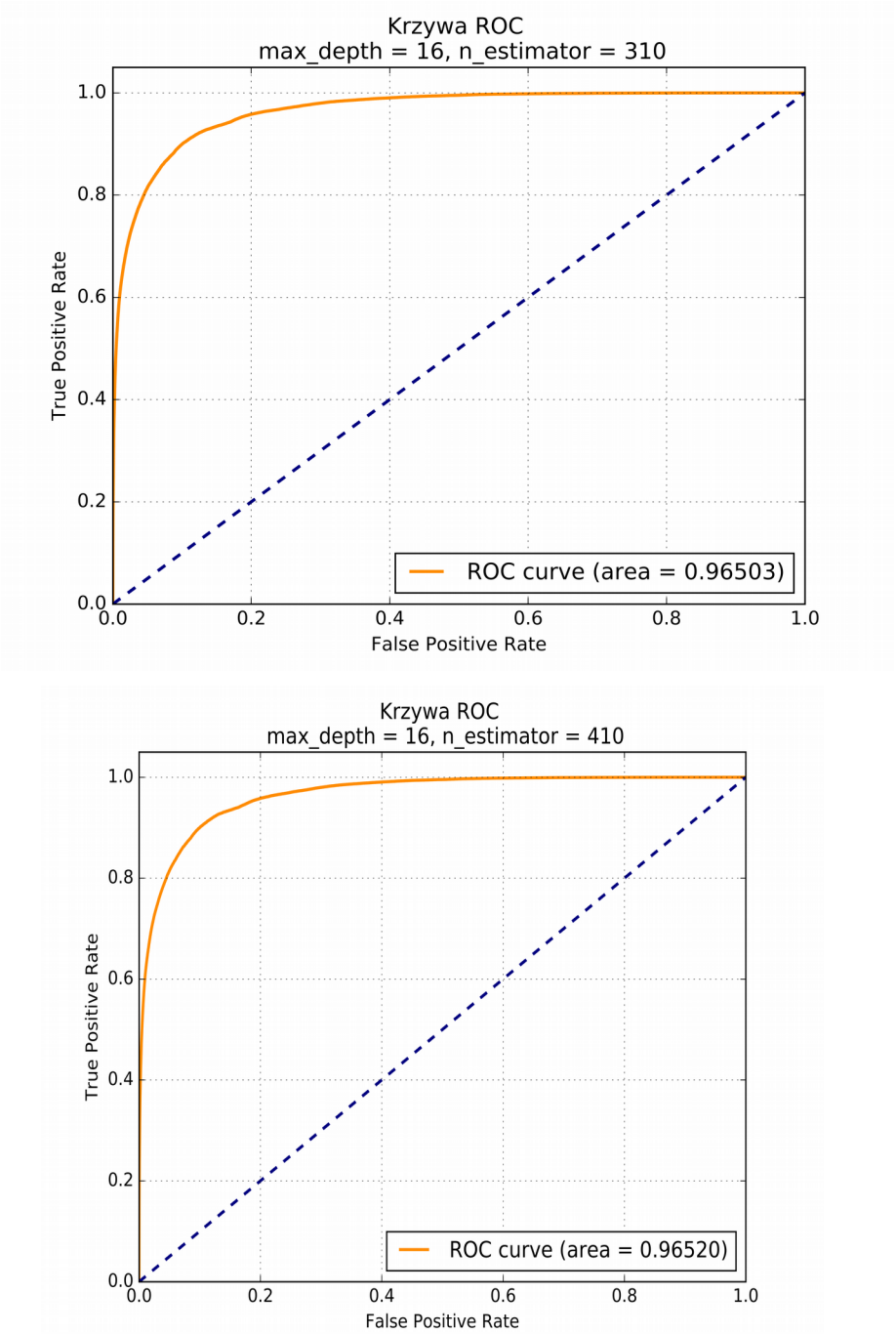
```
plt.legend(loc="lower right")
plt.show()
```

Poniżej, na rysunku 15, przedstawiono krzywą wyrysowaną dla różnych parametrów.

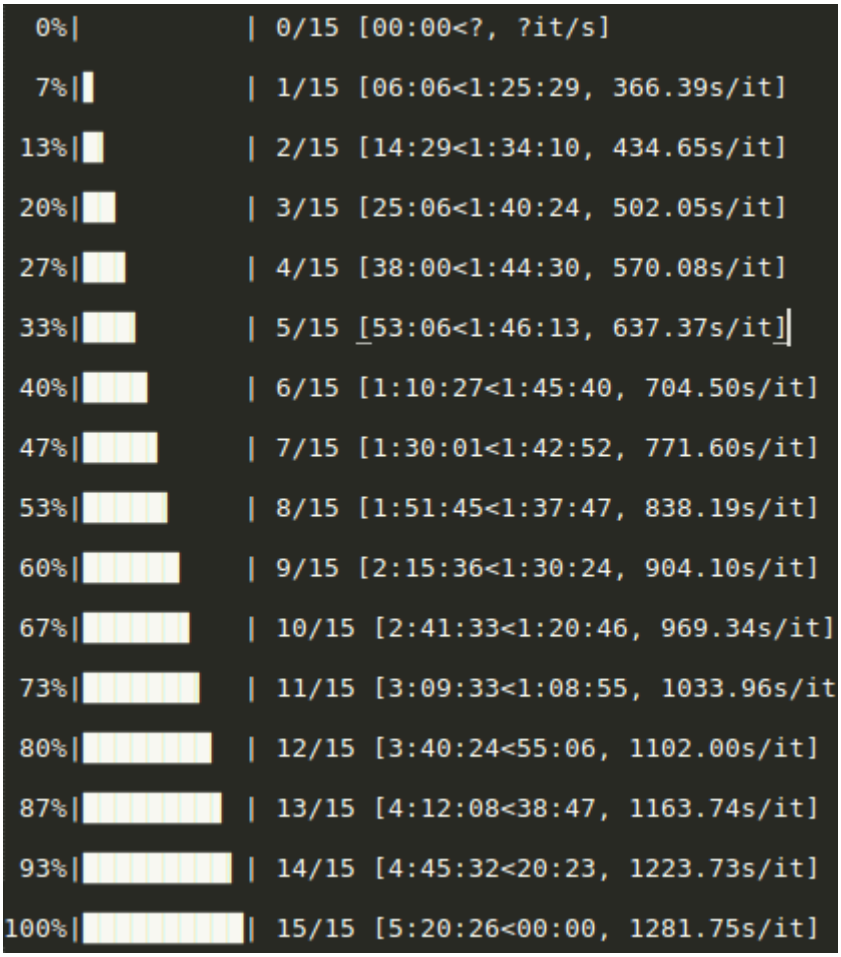



*Rys 15: Krzywa ROC dla max_depth = 2, n_estimators = 10*

Jak widać na załączonych obrazkach (rys. 16) uzyskana na zbiorze testowym dokładność jest na znacznie lepszym poziomie dla tej samej liczby drzew ale przy większej liczbie możliwych podziałów. Zwiększenie liczby drzew oraz maksymalnej liczby podziałów prowadzi do zwiększenia precyzji modelu. Zysk jednak jest niewielki  zwłaszcza jeśli wziąć pod uwagę zasoby i czas treningu. Na poniższych wykresach widać ROC dla maksymalnej liczby podziałów równej 16 natomiast różnica ilości drzew jest równa sto. Zysk precyzji, w tym przypadku pole krzywej pod wykresem jest równy 0.00029

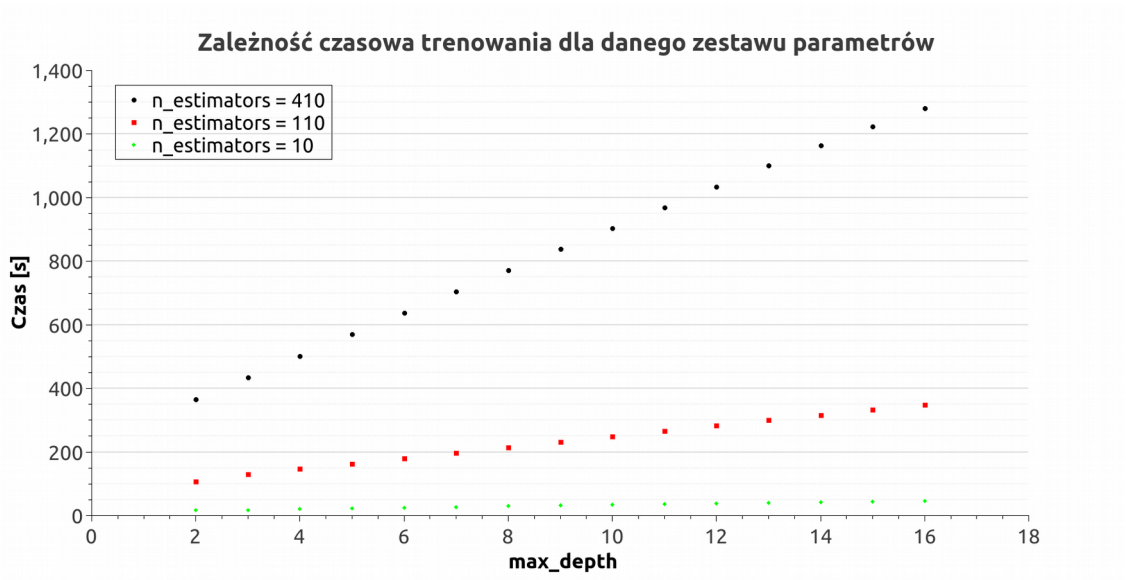



*Rys 16: Krzywe operacyjne odbiornika dla różnej ilości drzeww modelu przy stałej maksymalnej liczbie podziałów.*




*Rys 17: Zrzut ekranu prezentujący czas pojedyńczej iteracji oraz estymowany czas zakończenia wszystkich iteracji podczas trenowania algorytmu wraz ze zwiększającą się złożonością modelu.*

W ramach zbadania czasowej zależności wykonywania się algorytmu trenowanie zostało uruchomione w pętli, przy czym dla każdego obiegu pętli zestaw parametrów był inny. Na wykresie (rysunek 18) możemy zobaczyć czas wykonywania pojedynczej iteracji w zależności od maksymalnej dopuszczalnej liczby podziałów dla



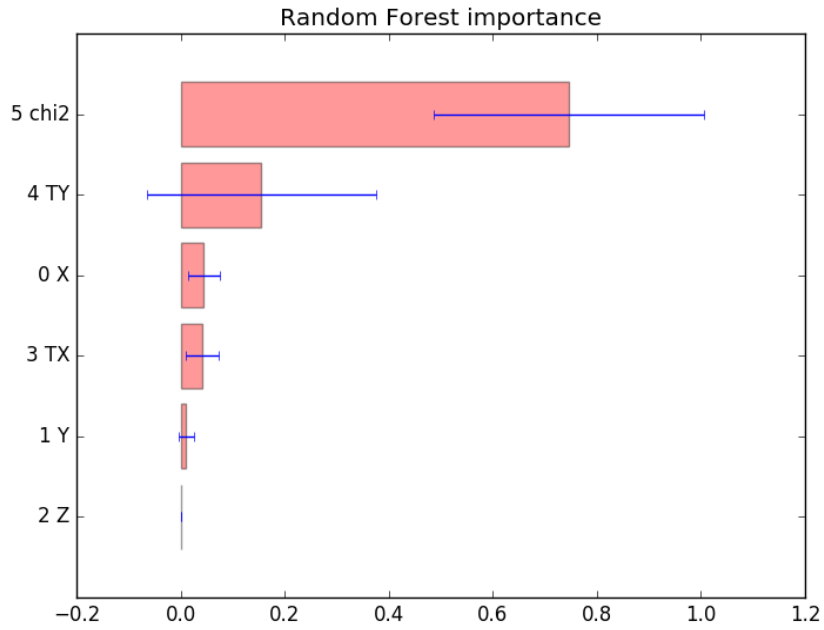

*Rys 18: Wykres prezentujący zależność wykonywania poszczególnej iteracji dla zmieniającej się liczby maksymalnych podziałów drzewa przy stałej liczbie drzewx*

ustalonej ilości drzew w lesie. Bardzo użyteczną klasą do kontroli czasu pracy algorytmu w konkretnej iteracji jest klasa *tqdm.* Każdorazowo mierzy czas wykonywania kodu i na tej podstawie estymuje czas zakończenia całego programu. Przykładowy widok z konsoli dla iteracji w której algorytm miał ustawione parametry n_estimators na 410 zaś max_depth zmieniało się w zakresie od 2 do 16 włącznie widać na  zrzucie ekranu (rysunek 17). Oczywiście czas ten jest zależny od maszyny na której kod jest uruchamiany. Dla komputera którym dysponowałem zużycie pamięci RAM oraz wykorzystanie procesora kształtowało jak na rysunku 19.


*Rys 19: Zrzut ekranu prezentujący moment zwolnienia zasobów komputera podczas zakończenia trenowania algorytmu. Zmiana w użyciu pamięci RAM to około 4 GB.*

W przypadku problemów z osiągnięciem założonej wydajności modelu warto zwrócić uwagę wykres istotności cech dla modelu. Pokazuje on jakie cechy były najbardziej istotne przy podejmowaniu decyzji. W przypadku lasu decyzyjnego jest to średnia całego dla całego zespołu. Dla omawianego klasyfikatora prezentuje się on następująco jak na rysunku 20.

Na osi *y* widzimy nazwy atrybutów które klasyfikator brał pod uwagę podczas procesu uczenia. Numery przy nazwach oznaczają numer kolumny zarówno w zbiorze uczącym jak i testowym. Słupki oznaczają średnią istotność cechy w modelu natomiast linie to odchylenie standardowe.




*Rys 20: Wykres prezentujący średnią istotność cech wraz z odchyleniem standardowym dla zespołu drzew losowych*

W przypadku algorytmu lasu losowego problematyczne jest przedstawienie całego zespołu wraz z parametrami charakteryzującymi model na każdym węźle. Taki graf jest bardzo rozległy i mocno nieczytelny. Możemy jednak zwizualizować pojedyncze drzewo z lasu. Taką opcję oferuje moduł *graphviz* oraz funkcja *export_graphviz* z pakietu *sklearn*. Pozwala na zapis dowolnego grafu w formacie dot. Dla drzew o małym stopniu rozgałęzienia możliwe jest odczytanie kryteriów przyjętych podczas podejmowanie decyzji, ilości zdarzeń jakie zostały pozostały w każdej z klas po podziale, czy ostateczną klasę w przypadku liścia [7, 23].

# 8. Podsumowanie

W tej pracy, korzystając z domowego komputera osiągnięto zdolność poprawnej klasyfikacji na poziomie 94 %. Jest to wartość średnia z ośmiu wyników uzyskanych przy użyciu cros_val_score() - funkcji która w losowy sposób miesza dane na podzbiór uczący oraz podzbiór testowy oraz sprawdza poprawność klasyfikatora. Nalepszą wartością było 94.071976%, a najgorszą 93.604546%. Określona została również zależność czasowa, a także zapotrzebowanie na zasoby.

# Bibliografia

[1] Baggot. Jim, „Teoria kwantowa. Odkrycia które zmieniły świat”, Prószyński i S-ka, 2013

[2] Biografia Maxa Plancka, https://www.britannica.com/biography/Max-Planck (data dostępu: 14 IX 2018)

[3] Bonnin. Rudolfo, „Building Machine Learning Projects with TensorFlow”, Packt Publishing, 2016

[4] Boschetti, Alberto, Massaron, Luca. „Python Data Science Essentials Second Edition”, Packt Publishing, 2016

[5] CERN – strona oficjalna, https://home.cern/about/computing (data dostępu: 30 VI 2018r.)

[6] Dan. Steinberg, „CART: Classification and Regression Trees”, Chapter 10, 2009

[7] Dokumentacja modułu graphviz, https://pypi.org/project/graphviz/

(data, dostępu: 16 VII 2018 r.)

[8] Dokumentacja OOB Erros for Random Forrest, http://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html

(data dostępu: 11 IX 2018)

[9] Douglas M. Hawkins. The Problem of Overfitting. J. Chem. Inf. Comput. Sci. 44, 1-12, (2012)

[10] Drzewo decyzyjne- grafika, https://www.ii.pwr.edu.pl/~kwasnicka/tekstystudenckie/apw/decyzyjne.html (data dostępu: 14 VII 2018r.)

[11] Działanie autoenkoderów, http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

(data dostępu: 12 VIII 2018)

[12] Esposito, Luigi Salvatore. Study of electron identification in the Emulsion Cloud Chambers of the OPERA experiment. PhD thesis: http://www.bo.infn.it/opera/docs/phd_thesis-BO-2005_07_08-esposito.pdf

(data dostępu: 11 IX 2018)

[13] Evgeniou, Theodoros & Pontil, Massimiliano. „Support Vector Machines: Theory and Applications". 2049. 249-257. 10.1007/3-540-44673-7_12, (2001).

[14] Fawcett. Tom, „An introduction to ROC analysis", Pattern Recognition Letters 27, 861–874, (2006)

[15] Fukugita, Masataka; Yanagida, Tsutomu. „Physics of Neutrinos and Application to Astrophysics", Springer, 2003

[16] Grafika – Model Standardowy, By Andrzej Barabasz (Chepry) - Praca własna, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20281688 (data dostępu: 10 IX 2018)

[17] Hawking Stephen. „Particle Creation by Black Holes" Comm. Math. Phys. 43 (1975), no. 3, 199--220. https://projecteuclid.org/euclid.cmp/1103899181, 1975

[18] Kibble. Tom, „Standard Model of Particle Physics", European Review. 23. 10.1017/S1062798714000520.

[19] Krzywa operacyja odbiornika, http://mathspace.pl/matematyka/receiver-operating-characteristic-krzywa-roc-czyli-ocena-jakosci-klasyfikacji-czesc-7/
(data dostępu: 14 IX 2018)

[20] Kumar, Ashish. „Learning Predictive Analytics with Python", Packt Publishing, 2016

[21] Laureaci Nagrody Nobla, https://www.nobelprize.org/prizes/uncategorized/all-nobel-prizes-in-physics/
(data dostępu: 12 IX 2018)

[22] Layton, Robert. „Learning Data Mining with Python", Packt Publishing, 2015

[23] Lenkeit, Jan. The OPERA Emulsions. Hamburg Student Seminar, http://www-opera.desy.de/publications/desyphdseminar/ (data dostępu: 01 VIII 2018r.)

[24] Lewis. Roger, An Introduction to Classification and Regression Tree (CART) Analysis, 2000

[25] Neutrino lecture writeups, https://warwick.ac.uk/fac/sci/physics/staff/academic/boyd/stuff/neutrinolectures/ (data dostępu 22 VII 2018)

[26] Oficjalna strona eksperymentu OPERA, http://operaweb.lngs.infn.it/ data dostępu: 11.09.2018 r.)

[27] Oficjalna strona MatPlotLib https://matplotlib.org/ (data dostępu: 10 IX 2018)

[28] Oficjalna strona Numpy, http://www.numpy.org/ (data dostępu: 10 IX 2018)

[29] Oficjalna strona Pandas, https://pandas.pydata.org/ (data dostępu: 10 IX 2018)

[30] Oficjalna strona Python Foundation, https://www.python.org/doc/essays/blurb/] (data dostępu: 2 VII 2018r.)

[31] Oficjalna strona SciPy, https://www.scipy.org/ (data dostępu: 10 IX 2018)

[32] Oficjalne strona Scikit-learn, http://scikit-learn.org/stable/ (data dostępu: 01 IX 2018 r.)

[33] Oficjana strona SpaceX, https://www.spacex.com/ (data dostępu: 10 IX 2018r.)

[34] Opis pozotałości supernowych: https://heasarc.gsfc.nasa.gov/docs/objects/snrs/snrstext.html (data dostępu: 16 IX 2018)

[35] Poglądy starożytnych na materię, https://www.visionlearning.com/en/library/Chemistry/1/Early-Ideas-about-Matter/49 (data dostępu: 14 IX 2018)

[36] Pratap. Dangeti, „Statistics for Machine Learning”, Pack Publishing, 2017

[37] Przewodnik po Kaggle dla początkujących, https://elitedatascience.com/beginner-kaggle (data dostępu: 5 VII 2018r.)

[38] R Acquafredda et al. „The OPERA experiment in the CERN to Gran Sasso neutrino beam”, JINST 4 P04018, 2009

[39] Raschka.Sebastian, „Python Machine Learning”, Packt Publishing, 2015

[40] Sinan. Ozdemir, „Principles of Data Science”, Pack Publishing, 2016

[41] Tan, Steinbach & Kumar, „Introduction to Data Mining”, Pearson, 2006

[42] Tin. Kam Ho, „Random Decision Forrest”, EEE Computer Society Washington, DC, USA, 1995

[43] Valentino. Zocca, „Python Deep Learning”, Pack Publishing, 2017

[44] Wilczek Franz. „Origins of Mass”, arXiv:1206.7114 [hep-ph], 2012

[45] Wpis na blogu odnośnie Python, https://stackoverflow.blog/2017/09/06/incredible-growth-python/ (data dostępu: 5 VII 2018)

[46] Zdjęcie detektora eksperymentu OPERA, https://www.sciencedaily.com/releases/2011/09/110923084425.htm (data dostępu: 11 IX 2018)

[47] Zukanovich Funchal, Renata; Schmauch, Benoit; Giesen, Gaëlle. „The Physics of Neutrinos”, arXiv:1308.1029 [hep-ph], 2013

[48] Żurada J., Barski M., Jędruch W. „Sztuczne Sieci Neuronowe”, Wydwnictwo Naukowe PWN, 1996

[49] Borda, Monica . Fundamentals in Information Theory and Coding. Springer. p. 11. 2011

[50] Ifrah, Georges, The Universal History of Numbers: From prehistory to the invention of the computer., John Wiley and Sons, p. 48, 2000

[51] Opis algorytmów uczenia maszynowego: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (data dostępu: 16 IX 2018)

[52] Paweł Tomaszewski, "Jan Czochralski i jego metoda", Oficyna Wydawnicza ATUT, Wrocław–Kcynia 2003

[53] Tetko, I. V.; Livingstone, D. J.; Luik, A. I. "Neural network studies. 1. Comparison of Overfitting and Overtraining", Journal of Chemical Information and Modeling. 35 (5): 826–833. (1995)