<https://www.anaconda.com/products/distribution#windows>

<https://analityk.edu.pl/jupyter-notebook-edytor-tekstu-dla-python/>

<https://www.tutorialcup.com/python/jupyter-notebook.htm>

https://stooq.pl/q/d/?s=wig

DataExtractor- łączy wszystkie dane- do wzbogacenia pliku z danymi.

right = pd.read\_csv('zmienne/'+file, index\_col='Data',engine='python') -> wczytanie pliku csv z katalogu zmienne. Right to jest zmienna. +file to wczytanie wszystkich plików z katalogu zmienne. Następne to argumenty.

Main.table to nasze wczytane dane.

Main.table.merge-> funkcja odpowiadająca za lączenie. Przedrostek oznacza, co łączymy.

**Ekstrakcja lokalizuje i identyfikuje odpowiednie dane, a następnie przygotowuje je do przetworzenia lub przekształcenia**. Ekstrakcja umożliwia łączenie wielu różnych rodzajów danych, a ostatecznie ich wydobywanie na potrzeby analizy biznesowej.

**Ekstrakcja danych** obejmuje pobieranie danych z różnych źródeł i przekształcanie ich w przydatny format do dalszego przetwarzania lub analizy. Jest to pierwszy krok potoku Extract-Transform-Load ([ETL](https://en.wikipedia.org/wiki/Extract,_transform,_load)) w procesie inżynierii danych.

Jako analityk danych może być konieczne połączenie danych dostępnych w wielu formatach plików, takich jak JSON, XML, CSV i SQL.

Użyliśmy bibliotek Pythona, takich jak pandas, json i żądania, aby odczytać dane z różnych źródeł i załadować je do notatnika Jupyter jako ramkę danych pandas.

**1. Pliki CSV**

Odnosi się to do pliku „wartości rozdzielanych przecinkami”, który służy do przechowywania danych w formacie tabelarycznym, podobnym do arkusza kalkulacyjnego. Każdy wiersz w pliku jest obserwacją (lub rekordem), a każdy rekord ma jedną lub więcej cech oddzielonych przecinkami.

Zanim utworzymy ramkę danych pandy z danych, ważne jest, aby najpierw zajrzeć do pliku za pomocą wbudowanej biblioteki odczytu Pythona. Na co zwrócić uwagę:

* Czy będziemy potrzebować dodatkowych argumentów z pd.read\_csv()
* Jeśli pierwszy wiersz zawiera nazwy zmiennych. Jeśli nie, użyjemy pd.read\_csv(file, header=None).
* Jeśli na górze są jakieś puste wiersze, musimy pominąć za pomocą pd.read\_csv(file, skip\_rows=n).

#os Różne interfejsy systemu operacyjnego. Ten moduł zapewnia przenośny sposób korzystania z funkcji zależnych od systemu operacyjnego. Moduły os w Python-dostarczają licznych narzędzi do radzenia sobie z nazwami plików, ścieżkami, katalogami. Jest bardzo szeroko stosowanym modułem, który jest przydatny podczas przetwarzania plików z różnych miejsc w systemie . Jest używany do różnych celów, takich jak łączenie, normalizowanie i pobieranie nazw ścieżek w Pythonie. Pozwala uzyskać ścieżkę folderów i nazwy plików.

#pandas to pakiet Pythona typu open source, który jest najczęściej używany do zadań związanych z nauką o danych/analizą danych i uczeniem maszynowym . Jest zbudowany na innym pakiecie o nazwie Numpy, który zapewnia obsługę tablic wielowymiarowych.

**Pandas** to biblioteka Pythona do obsługi danych sekwencyjnych i tabelarycznych. Zawiera wiele narzędzi do zarządzania, analizowania i manipulowania danymi w wygodny i wydajny sposób. Możemy myśleć o jego strukturach danych jak o tabelach baz danych lub arkuszach kalkulacyjnych.

Pandas jest zbudowany na bazie biblioteki Numpy i ma dwie podstawowe struktury danych, a mianowicie. Seria (1-wymiarowa) i DataFrame (2-wymiarowa). Może obsługiwać zarówno dane homogeniczne, jak i heterogeniczne, a niektóre z jego wielu możliwości to:

* Narzędzia ETL (narzędzia do ekstrakcji, transformacji i ładowania)
* Postępowanie z brakującymi danymi (NaN)
* Radzenie sobie z plikami danych (csv, xls, db, hdf5 itp.)
* Narzędzia do manipulacji szeregami czasowymi

W ekosystemie Pythona Pandas to najlepszy wybór do pobierania, manipulowania, analizowania i przekształcania danych finansowych.

#pd.read\_csv Wczytaj plik z wartościami rozdzielanymi przecinkami (csv) do DataFrame. Obsługuje również opcjonalnie iterację lub dzielenie pliku na kawałki.

**index\_col *int, str, sekwencja int / str lub False, opcjonalne, domyślneNone***

Kolumny, które mają być używane jako etykiety wierszy DataFrame, podane jako nazwa ciągu lub indeks kolumny. Jeśli podano sekwencję int / str, używany jest MultiIndex.

Uwaga: index\_col=Falsemoże być użyte do wymuszenia na pandach, aby *nie* używały pierwszej kolumny jako indeksu, np. gdy masz zniekształcony plik z ogranicznikami na końcu każdej linii.

+file to nazwa tego pliku

os.listdir(path); ścieżka- katalog, jaki należy przeszukać

Metoda Pythona **listdir()** zwraca listę zawierającą nazwy wpisów w katalogu podanym przez path. Lista jest ułożona w dowolnej kolejności. Nie zawiera specjalnych wpisów „.” i '..', nawet jeśli są obecne w katalogu.

#files.remove('glowna\_etfw20.csv')  
#files.remove('proba.ipynb') To kasuje plik, ale jeżeli nie jest wygenerowany, to tego się nie robi, bo python wygeneruje błąd. Tylko w sytuacji jak już jest plik żeby nie dopisywało kolejnych danych. Obie te linijki mogą być skasowane, bo to pozostałości po kodzie, który był użyty wcześniej.

Wszystko, co musisz zrobić, aby usunąć plik, to **zwrócić się do systemu operacyjnego. remove() z odpowiednią nazwą pliku i ścieżką** (Python domyślnie wskazuje bieżący katalog, więc nie musisz określać ścieżki, jeśli plik, który chcesz usunąć, znajduje się w katalogu domyślnym).

#.head  
Zwróć pierwsze n wierszy. Ta funkcja zwraca pierwszych n wierszy dla obiektu na podstawie pozycji. Przydaje się do szybkiego testowania, czy Twój obiekt ma w sobie odpowiedni typ danych.

Scal obiekty DataFrame lub nazwane Series za pomocą sprzężenia w stylu bazy danych.

#merge   
Nazwany obiekt Series jest traktowany jako DataFrame z pojedynczą nazwaną kolumną. Łączenie odbywa się na kolumnach lub indeksach. W przypadku łączenia kolumn w kolumnach indeksy DataFrame zostaną zignorowane . W przeciwnym razie w przypadku łączenia indeksów w indeksach lub indeksów w kolumnie lub kolumnach indeks zostanie przekazany dalej. Podczas wykonywania scalania krzyżowego nie są dozwolone żadne specyfikacje kolumn do scalania.

**how *{'left', 'right', 'outer', 'inner', 'cross'}, domyślnie 'inner'*** Rodzaj scalania do wykonania.

-left: używaj tylko kluczy z lewej ramki, podobnie do lewego sprzężenia zewnętrznego SQL; zachowaj kolejność kluczy.

-right: używaj tylko kluczy z prawej ramki, podobnie do prawego sprzężenia zewnętrznego SQL; zachowaj kolejność kluczy.

-outer: użyj unii kluczy z obu ramek, podobnie do pełnego zewnętrznego sprzężenia SQL; sortuj klucze leksykograficznie.

-inner: użyj przecięcia kluczy z obu ramek, podobnie do wewnętrznego sprzężenia SQL; zachowaj kolejność lewych klawiszy.

-cross (krzyż): tworzy iloczyn kartezjański z obu ramek, zachowuje kolejność lewych klawiszy.

**left\_index *bool, domyślnie False***

* Użyj indeksu z lewego elementu DataFrame jako klucza (kluczy) łączenia. Jeśli jest to MultiIndex, liczba kluczy w drugiej ramce DataFrame (indeks lub liczba kolumn) musi być zgodna z liczbą poziomów.

**right\_index *bool, domyślnie False***

* Użyj indeksu z prawej ramki DataFrame jako klucza łączenia. Te same zastrzeżenia co left\_index.

**copy *bool, domyślna prawda***

* Jeśli False, jeśli to możliwe, unikaj kopiowania.

**suffixes**, sufiks — ciąg lub krotka sufiksów do sprawdzenia. Przyrostki.

Currency Model LSTM

Aby zainstalować Tensorflow, potrzebujemy pip dla Pythona. Zwykle pip jest już zainstalowany, jeśli wersja Pythona to 3 lub wyższa.

1. Tworzenie wirtualnego środowiska Tensorflow: pip install --user virtualenv
2. Po skonfigurowaniu środowiska wirtualnego użyj następującego polecenia, aby zainstalować pakiet Tensorflow pip: pip install --upgrade tensorflow

Gdybyśmy użyli na początku pip install tensorflow, to wyskoczyłby nam błąd. Dlaczego ?

Ponieważ u nas Python nie ma dostępu do plików serwera. Dlatego trzeba to zrobić z uwagi na konkretnego, danego użytkownika. Dopisujemy zatem –user, by zainstalować biblioteki tylko dla danego użytkownika.

Istnieje wiele metod, których można użyć do zainstalowania TensorFlow, takich jak użycie pip do zainstalowania kół dostępnych w PyPI. Instalacja TensorFlow przy użyciu pakietów conda oferuje szereg korzyści, w tym kompletny system zarządzania pakietami, szerszą obsługę platform, bardziej usprawnione działanie GPU i lepszą wydajność procesora. Pakiety te są dostępne za pośrednictwem repozytorium Anaconda, a ich instalacja jest tak prosta, jak uruchomienie „conda install tensorflow” lub „conda install tensorflow-gpu” z interfejsu wiersza poleceń.

Jedną z kluczowych korzyści z instalacji TensorFlow przy użyciu conda zamiast pip jest wynik systemu zarządzania pakietami conda. Gdy TensorFlow jest instalowany przy użyciu conda, conda instaluje również wszystkie niezbędne i kompatybilne zależności dla pakietów. Odbywa się to automatycznie; użytkownicy nie muszą instalować żadnego dodatkowego oprogramowania za pośrednictwem menedżerów pakietów systemowych lub w inny sposób.

KERAS Czy mogę używać keras bez TensorFlow?

Nie, Keras to wysokopoziomowe API do budowania i trenowania modeli sieci neuronowych. **Keras nie jest zależny od TensorFlow** i na odwrót. Keras może używać TensorFlow jako swojego zaplecza. Może również korzystać z innych zapleczy, takich jak Theano (Theano to biblioteka Pythona i kompilator optymalizujący do manipulowania i oceny wyrażeń matematycznych, zwłaszcza tych o wartościach macierzowych. W Theano obliczenia są wyrażane przy użyciu składni NumPy-esque i kompilowane w celu wydajnego działania na architekturze CPU ((jednostka centralna) to uogólniony procesor przeznaczony do wykonywania różnorodnych zadań), lub GPU ((jednostka przetwarzania grafiki) to wyspecjalizowana jednostka przetwarzania z rozszerzonymi możliwościami obliczeń matematycznych, idealna do zadań związanych z grafiką komputerową i uczeniem maszynowym)) i CNTK (Microsoft Cognitive Toolkit, wcześniej znany jako CNTK, a czasem stylizowany jako Microsoft Cognitive Toolkit, to przestarzała platforma do głębokiego uczenia się opracowana przez Microsoft Research. Microsoft Cognitive Toolkit opisuje sieci neuronowe jako serię kroków obliczeniowych za pomocą ukierunkowanego wykresu).

DENSE – GĘSTY- Gęstą warstwę można dodać do modelu sekwencyjnego **za pomocą metody „dodaj” i określając typ warstwy jako „Gęsta”** . Warstwa gęsta to **regularna, głęboko połączona warstwa sieci neuronowej** . Jest to najpowszechniejsza i najczęściej stosowana warstwa.

FLATTEN- SPŁASZCZYĆ- Spłaszcza dane wejściowe. Nie wpływa na wielkość partii. Dlaczego jej używamy? Musi mieć postać jednowymiarowego wektora liniowego. Kształty prostokątne lub sześcienne nie mogą być danymi wejściowymi bezpośrednimi. I dlatego potrzebujemy spłaszczania i w pełni połączonych warstw. Spłaszczanie to **przekształcanie danych w jednowymiarową tablicę w celu wprowadzenia ich do następnej warstwy** .

INPUT- służy do tworzenia instancji tensora Keras.

Tensor Keras to obiekt tensor z bazowego backendu (Theano lub TensorFlow), który uzupełniamy o pewne atrybuty, które pozwalają nam zbudować model Keras po prostu znając dane wejściowe i wyjściowe modelu.

BatchNormalization- NORMALIZACJA WSADOWA- Warstwa, która normalizuje swoje dane wejściowe. Normalizacja wsadowa to **technika uczenia bardzo głębokich sieci neuronowych, która standaryzuje dane wejściowe do warstwy dla każdej minipartii** . Wpływa to na stabilizację procesu uczenia się i radykalne zmniejszenie liczby epok treningowych wymaganych do trenowania głębokich sieci.

Dropout- Stosuje Dropout do wejścia. Warstwa Dropout **losowo ustawia jednostki wejściowe na 0 z częstotliwością na każdym kroku w czasie treningu, co pomaga zapobiegać nadmiernemu dopasowaniu.** Termin „dropout” jest używany w odniesieniu **do techniki, która usuwa niektóre węzły sieci** . Wypadanie może być postrzegane jako chwilowa dezaktywacja lub ignorowanie neuronów sieci. Technikę tę stosuje się w fazie treningu, aby zredukować efekty nadmiernego dopasowania.

LSTM- Long Short-Term Memory warstwa. Long-Short-Term Memory Network lub LSTM to **odmiana rekurencyjnej sieci neuronowej (RNN), która jest dość skuteczna w przewidywaniu długich sekwencji danych, takich jak zdania i ceny akcji w określonym czasie** . Różni się od normalnej sieci ze sprzężeniem zwrotnym, ponieważ w jej architekturze występuje pętla sprzężenia zwrotnego.

Sequential- Kiedy używać modelu sekwencyjnego?

**Model ma wiele wejść lub wiele wyjść** . Każda z twoich warstw ma wiele wejść lub wiele wyjść. Musisz udostępnić warstwy. Potrzebujesz nieliniowej topologii (np. połączenie resztkowe, model wielorozgałęziony)

load\_model- Wczytuje model zapisany przez model.save(). **Ładuje model zapisany przez model. zapisz()** . Należy zauważyć, że po załadowaniu wagi modelu mogą mieć różne nazwy w zakresie. Nazwy w zakresie obejmują nazwy modelu/warstwy, takie jak „dense\_1/kernel:0” .

Matplotlib to **kompleksowa biblioteka do tworzenia statycznych, animowanych i interaktywnych wizualizacji w Pythonie** . Matplotlib czyni łatwe rzeczy łatwymi i trudnymi możliwymi.

Seaborn- to biblioteka wizualizacji danych Pythona zbudowana na bazie Matplotlib. Część kodu dotycząca importu seaborn mówi Pythonowi, aby wprowadził bibliotekę Seaborn do bieżącego środowiska.

Pandas- Pandas to pakiet Pythona typu open source, który jest **najczęściej używany do zadań związanych z nauką o danych/analizą danych i uczeniem maszynowym** . Jest zbudowany na innym pakiecie o nazwie Numpy, który zapewnia obsługę tablic wielowymiarowych.

NumPy- to biblioteka Pythona używana do pracy z tablicami. Posiada również funkcje do pracy w dziedzinie algebry liniowej, transformaty Fouriera i macierzy. NumPy został stworzony w 2005 roku przez Travisa Oliphanta. Jest to projekt open source i możesz z niego swobodnie korzystać. NumPy to skrót od **Numerical Python**.

#data.shape- wymiary tablicy z danymi, ilość wierszy i kolumn

**Różne sposoby na znalezienie kolumn, które zawierają Nan w DataFrame**

1. df.isna().any() -> Użyj isna() do znalezienia wszystkich kolumn z Nan
2. df.isnull().any()
3. df[df.columns[df.isna().any()]] -> Użyj isna() , aby wybrać wszystkie kolumny z wartościami NaN
4. df[df.columns[df.isnull().any()]]

data.columns[data.isna().any()] .tolist() -> tolist() -> używany do konwersji elementów danych tablicy na listę, w naszym przypadku plik wygląda przejrzyściej (w liście a nie w tabeli).

#len() jest jedną z wbudowanych funkcji Pythona. Zwraca długość obiektu. Na przykład może zwrócić liczbę pozycji na liście.

#isnull() Wykryj brakujące wartości dla obiektu podobnego do tablicy.

#sum() zwraca liczbę, sumę wszystkich elementów w iteracji.

30% u nas umownie. Jeżeli chcemy dokładniejsze usuwanie kolumn, to możemy użyć 10%. Myślę, że 30% pozwoli nam usunąć te najbardziej „szkodliwe” kolumny. Gdy kolumna ma powyżej 30% NaN to zepsuje nam wyniki.

#info() ; entries- wpisy

#drop() Usuń określone etykiety z wierszy lub kolumn. Usuń wiersze lub kolumny, określając nazwy etykiet i odpowiednią oś lub określając bezpośrednio nazwy indeksów lub kolumn.

data = data.drop('Wolumen\_ftse250\_d', axis=1)

**DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')**

Parameters: labels: single label or list-like

Index or column labels to drop. A tuple will be used as a single label and not treated as a list-like.

**-axis**{0 or ‘index’, 1 or ‘columns’}, default 0

Whether to drop labels from the index (0 or ‘index’) or columns (1 or ‘columns’).

#shift(-5) Aby łatwo przesunąć na przykład o 5 wartości; Przesuń indeks o żądaną liczbę okresów z opcjonalną częstotliwością czasu. SESJE OTWIERAJĄ SIĘ CODZIENNIE, WIĘC CODZIENNIE DANA AKCJA MA INNY KURS. W KOLUMNIE ZAKOŃCZENIA NA DANY DZIEŃ- CHCEMY JĄ PRZESUNĄĆ O 1.

#head() Zwraca pierwsze n wierszy. n domyślnie = 5