

Uczenie Maszynowe

[Strona główna](#) / [Moje kursy](#) / [UM](#) / [28 lutego - 6 marca](#) / [Laboratorium: Wprowadzenie \(piątek – grupy: 2b, 3a, 3b\)](#)

Laboratorium: Wprowadzenie (piątek – grupy: 2b, 3a, 3b)

Wymagane do: piątek, 4 marca 2022, 23:59

Zbiór danych

Każdy proces uczenia maszynowego rozpoczyna się od pozyskania odpowiednich danych, np. w sposób przedstawiony poniżej. Pozyskany zbiór zawiera informacje o cenach domów w Kalifornii.

```
mkdir data
cd data
curl -O https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.tgz
tar xfz housing.tgz
gzip housing.csv
rm housing.tgz
```

A tak można podglądnąć dane:

```
gzcat data/housing.csv.gz | head -4
```

Środowisko pracy

W laboratorium pracujemy na kontach Użytkownika Gość (Guest User). Nie jest potrzebne hasło, ale pamiętaj, że przy wylogowaniu zawartość konta jest czyszczona.

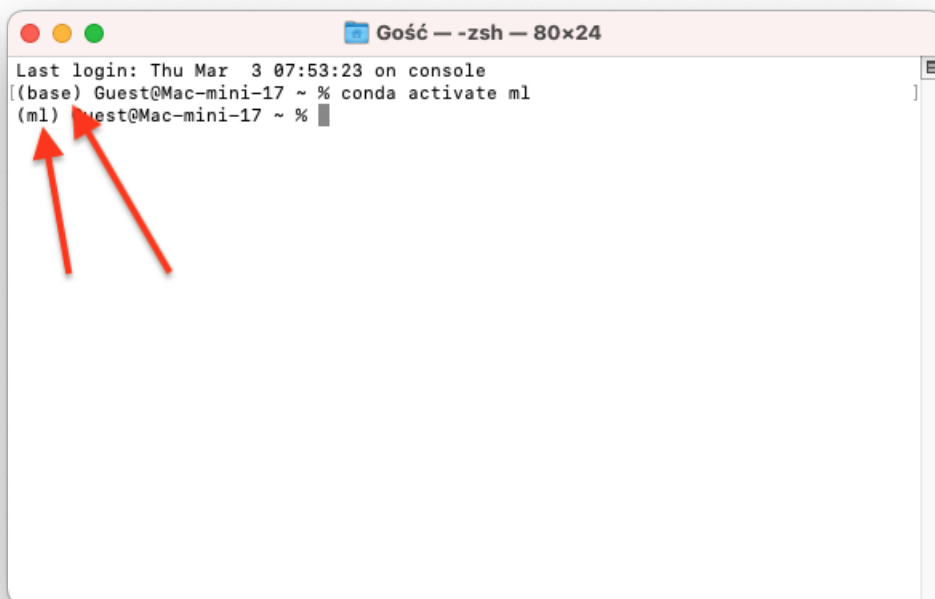
Na komputerach w laboratorium jest zainstalowane narzędzie [Miniforge](#) (bliski krewny [Miniconda](#)), i przy jego pomocy zarządzamy tam środowiskami m.in. Pythona.

Na potrzeby tego kursu utworzyliśmy środowisko o nazwie `ml`, które zawiera wszystkie niezbędne narzędzia. Otwórz aplikację *Terminal* i aktywuj środowisko przy pomocy polecenia:

```
conda activate ml
```

Środowisko aktywne w danej chwili wyświetlane jest zawsze w nawiasach przed znakiem zachęty, tak jak oznaczono strzałkami na poniższym obrazku:



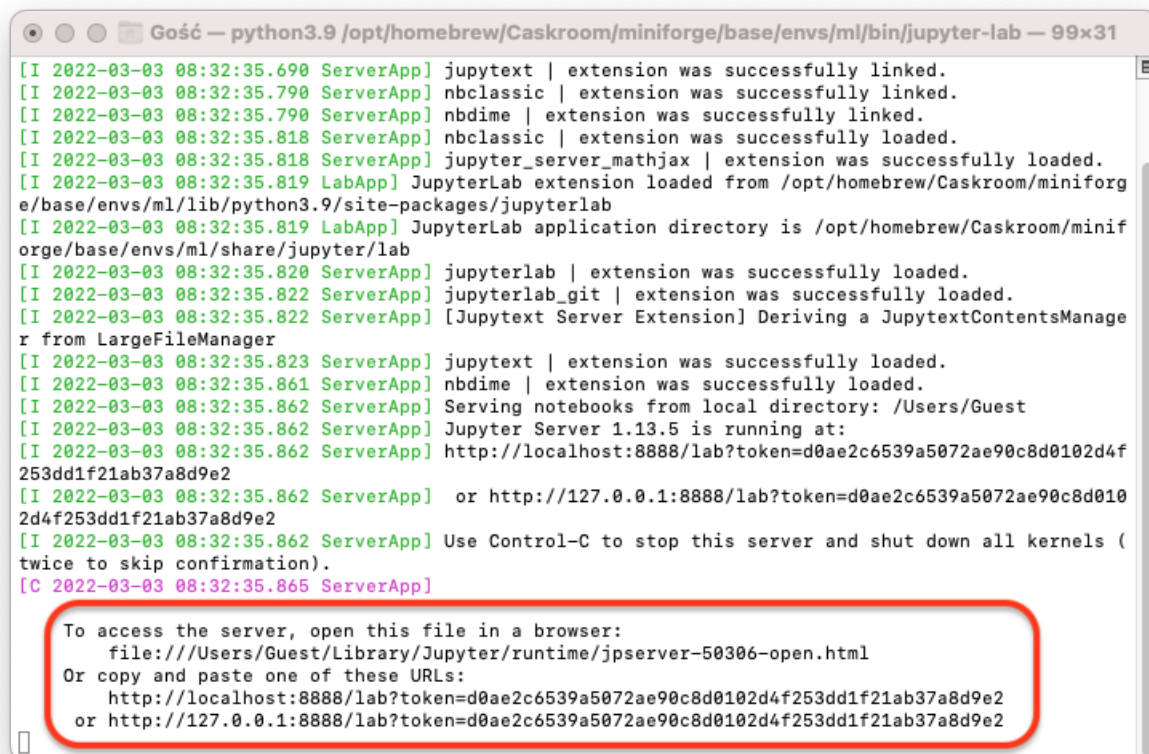


```
Gość — zsh — 80x24
Last login: Thu Mar 3 07:53:23 on console
(base) Guest@Mac-mini-17 ~ % conda activate ml
(ml) Guest@Mac-mini-17 ~ %
```

Uruchom JupyterLab w wybranym katalogu:

```
jupyter lab
```

Spowoduje to uruchomienie lokalnego serwera HTTP z interfejsem JupyterLab, najprawdopodobniej na porcie 8888. Jupyter powinien automatycznie otworzyć go w przeglądarce, ale jeżeli tak się nie stało, skopiuj adres URL wraz z tokenem do przeglądarki:



```
Gość — python3.9 /opt/homebrew/Caskroom/miniforge/base/envs/ml/bin/jupyter-lab — 99x31
[I 2022-03-03 08:32:35.690 ServerApp] jupyterlab | extension was successfully linked.
[I 2022-03-03 08:32:35.790 ServerApp] nbclassic | extension was successfully linked.
[I 2022-03-03 08:32:35.790 ServerApp] nbdlm | extension was successfully linked.
[I 2022-03-03 08:32:35.818 ServerApp] nbclassic | extension was successfully loaded.
[I 2022-03-03 08:32:35.818 ServerApp] jupyter_server_mathjax | extension was successfully loaded.
[I 2022-03-03 08:32:35.819 LabApp] JupyterLab extension loaded from /opt/homebrew/Caskroom/miniforge/base/envs/ml/lib/python3.9/site-packages/jupyterlab
[I 2022-03-03 08:32:35.819 LabApp] JupyterLab application directory is /opt/homebrew/Caskroom/miniforge/base/envs/ml/share/jupyter/lab
[I 2022-03-03 08:32:35.820 ServerApp] jupyterlab | extension was successfully loaded.
[I 2022-03-03 08:32:35.822 ServerApp] jupyterlab_git | extension was successfully loaded.
[I 2022-03-03 08:32:35.822 ServerApp] [Jupyter Server Extension] Deriving a JupyterTextContentsManager from LargeFileManager
[I 2022-03-03 08:32:35.823 ServerApp] jupyterlab | extension was successfully loaded.
[I 2022-03-03 08:32:35.861 ServerApp] nbdlm | extension was successfully loaded.
[I 2022-03-03 08:32:35.862 ServerApp] Serving notebooks from local directory: /Users/Guest
[I 2022-03-03 08:32:35.862 ServerApp] Jupyter Server 1.13.5 is running at:
[I 2022-03-03 08:32:35.862 ServerApp] http://localhost:8888/lab?token=d0ae2c6539a5072ae90c8d0102d4f253dd1f21ab37a8d9e2
[I 2022-03-03 08:32:35.862 ServerApp] or http://127.0.0.1:8888/lab?token=d0ae2c6539a5072ae90c8d0102d4f253dd1f21ab37a8d9e2
[I 2022-03-03 08:32:35.862 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2022-03-03 08:32:35.865 ServerApp]

To access the server, open this file in a browser:
file:///Users/Guest/Library/Jupyter/runtime/jpserver-50306-open.html
Or copy and paste one of these URLs:
http://localhost:8888/lab?token=d0ae2c6539a5072ae90c8d0102d4f253dd1f21ab37a8d9e2
or http://127.0.0.1:8888/lab?token=d0ae2c6539a5072ae90c8d0102d4f253dd1f21ab37a8d9e2
```

Ponieważ w ramach laboratorium chcemy zbudować oprogramowanie, które m.in. przeprowadzi proces pozyskiwania danych, to:

1. znajdź odpowiednie biblioteki, aby w/w kroki przeprowadzić w Jupyterze/Pythonie (pobranie danych za pomocą HTTP, rozpakowanie archiwum TAR, kompresja danych GZIP, manipulacja plikami; odpowiedź: `os`, `tarfile`, `urllib`, `gzip`) – jeżeli brakuje jakiegoś pakietu Python, zainstaluj go przy pomocy polecenia `pip install --user ...` i zgłoś to prowadzącemu,
2. napisz kod Python realizujący powyższy proces pozyskania danych, wraz z wyświetleniem początkowych rekordów z pliku,
3. sprawdź, czy w/w kod będzie równie dobrze działać, jeżeli uruchomisz go w Pythonie w sposób nieinteraktywny – możesz wygenerować skrypt `.py` wybierając z menu JupyterLab opcję `File > Save and Export Notebook As > Executable Script`

Informacje o zbiorze danych

Użyj [pandas](#) aby zobaczyć czym są pobrane dane.

```
import pandas as pd
df = pd.read_csv('data/housing.csv.gz')
```

Czym jest zmienna **df**?

Przetestuj następujące metody:

```
df.head()
df.info()
```

Sprawdź jakiego rodzaju dane znajdują się w kolumnie **ocean_proximity**. Przetestuj na niej metody:

```
value_counts()
describe()
```

Wizualizacja

Jupyter zawiera funkcje pozwalające na wizualizację danych, które domyślnie korzystają z biblioteki [matplotlib](#).

Sprawdź jak działają – spróbuj użyć poniższego kodu.

```
df.hist(bins=50, figsize=(20,15))
```

A teraz tego:

```
df.plot(kind="scatter", x="longitude", y="latitude",
        alpha=0.1, figsize=(7,4))
```

I może jeszcze tego:

```
import matplotlib.pyplot as plt    # potrzebne ze względu na argument cmap
```

```
df.plot(kind="scatter", x="longitude", y="latitude",
        alpha=0.4, figsize=(7,3), colorbar=True,
        s=df["population"]/100, label="population",
        c="median_house_value", cmap=plt.get_cmap("jet"))
```

1. Poczytaj w dokumentacji o w/w funkcjach/metodach/parametrach, dowiedz się jak działają.
2. Zapisz w/w trzy rysunki do trzech plików PNG o nazwach **obraz[1-3].png**. Ta umiejętność będzie przydatna gdybyś chciał(a) zmienić notebook Jupytera na kod w czystym Pythonie, który może być potrzebny do wdrożenia w środowisku produkcyjnym.

Analiza

Policz [macierz korelacji](#) pomiędzy **median_house_value**, a pozostałym kolumnami.

```
df.corr()["median_house_value"].
    sort_values(ascending=False)
```

Przy pomocy *jednego polecenia* zapisz wyniki w pliku CSV o nazwie **korelacja.csv**, tak, aby kolumny miały nazwy: **atrybut** oraz **współczynnik_korelacji**. Podpowiedzi: [s.reset_index\(\)](#), [df.rename\(columns=...\)](#), [df.to_csv\(index=False\)](#).

Do wizualnej analizy związków pomiędzy zmiennymi często używamy tzw. *pair plot*. Wyświetl go dla zaimportowanego zbioru przy pomocy [odpowiedniej metody](#) biblioteki [seaborn](#):

```
import seaborn as sns
sns.pairplot(df)
```

Przygotowanie do uczenia

Używając [scikit-learn](#), podziel dane na [zbiór uczący i testujący](#).

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(df,
                                      test_size=0.2,
                                      random_state=42)

len(train_set), len(test_set)
```

1. Sprawdź w dokumentacji do czego służą proponowane parametry funkcji **train_test_split()**



2. Oglądnij zawartość zbioru uczącego i testującego.
3. Sprawdź macierze korelacji w obu powstałych zbiorach. Czy wyniki są podobne? Co to znaczy?
4. Zapisz zbiory do plików `pickle train_set.pkl` i `test_set.pkl`.

Zachowaj wyniki swojej pracy

1. Upewnij się jakie pliki i katalogi zostały przez Ciebie i Twoje oprogramowania utworzone podczas realizacji laboratorium.
2. Wykonaj kopie tych plików, tak aby móc z nich skorzystać poza laboratorium. Możesz skorzystać np. ze zdalnego repozytorium Git, [uczelnianej usługi NextCloud](#) lub innego dysku sieciowego.
3. Uwaga: pliki przechowywane na lokalnych maszynach są każdorazowo usuwane przy wylogowaniu.

Prześlij raport

Przy większości laboratoriów poprosimy Państwa o przesyłanie raportów w postaci działających skryptów Pythona (`.py`).

Będą one uruchamiane w środowisku analogicznym do laboratoryjnego (dlatego ważne jest zgłaszanie brakujących bibliotek) i oceniane pod względem:

- tego, czy skrypt działa poprawnie,
- czy wyniki jego działania (np. dokładność predykcji) są dopuszczalne.

Dobrym testem całości skryptu z poziomu JupyterLab jest restart kernela i wykonanie wszystkich komórek po kolei (*Run > Restart Kernel and Run All Cells...*).

Jeżeli cały notebook „przechodzi” test, możesz wyeksportować go jako skrypt Python tak jak na początku ćwiczeń, a następnie spróbować uruchomić go w terminalu (pamiętając o aktywacji odpowiedniego środowiska conda!).

Zmień nazwę pliku na `lab1.py` i prześlij go jako rozwiązanie niniejszego zadania w Moodle.

Dodatek: interaktywna wizualizacja

Zmień backend wizualizacyjny pandas na Plotly:

```
pd.options.plotting.backend = "plotly"
```

Spróbuj wykonać ponownie polecenia wcześniej używane do wyświetlania wykresów.

Potrzebne będą pewne drobne modyfikacje, ale teraz wyświetlane wykresy będą interaktywne – pozwolą m.in. na powiększanie i pomniejszanie, co jest bardzo przydatne np. przy przeglądaniu szeregów czasowych.

Status przesłanego zadania

Status przesłanego zadania	Nie próbowano
Stan oceniania	Nieocenione
Pozostały czas	Opóźnienie w przesłaniu: 34 dni 19 godzin
Ostatnio modyfikowane	-
Komentarz do przesłanego zadania	▶ Komentarze (0).



Platforma e-Learningowa obsługiwana jest przez:
Centrum e-Learningu AGH oraz Centrum Rozwiązań Informatycznych AGH

[Pobierz aplikację mobilną](#)
[Resetuj przewodnik użytkownika na tej stronie](#)