Hermanni Taimisto 50210361

Harjoitustyön kuvaus

Johdanto datatieteeseen

SISÄLLYSLUETTELO

[1. Kehitysympäristö 1](#_Toc100497135)

[1.1 Hyödyllisiä lähteitä 2](#_Toc100497136)

[2. Yhteenveto 3](#_Toc100497137)

[Lähteet 4](#_Toc100497138)

.

# Kehitysympäristö

Tavoitepisteet: 3.

Päädyin käyttämään Microsoftin Azure -pilviympäristöä harjoitustyön kehitysympäristönä. Oppimispäiväkirjaa tehtäessä olen tutustunut Jupyter Lab Notebook:n ajamiseen paikallisesti omalla koneella sekä Google Colab -ympäristössä. Olen kiinnostunut räpeltämisestä, joten oman pilviympäristön luominen tuntui luontevalta vaihtoehdolta. Harkitsin myös hyödyntäväni kotona pyörivän Raspberry Pi 3:n laskentatehoa, mutta onnettomat speksit sekä ajankäytölliset haasteet ohjasivat valmiin pilviympäristön hyödyntämiseen.

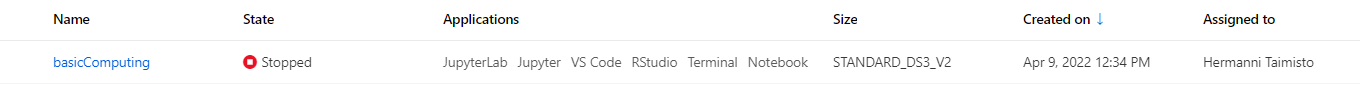
Lueskelin jo palautettuja toteutuksien kuvauksia, ja oikean lisenssin valintaa lukuun ottamatta Machine Learning -ympäristön luominen sujui ongelmitta. Rekisteröimällä Azure tilin TUNI-sähköpostilla sai opiskelijaedut käyttöönsä, joka tarkoitti 100USD edestä resursseja sekä mahdollisuus käyttää laskentatehoa. Alla olevassa kuvassa 1 näkyy ML ympäristön automaattisesti luomat resurssit:

A picture containing graphical user interface

Description automatically generated

**Kuva 1:** Machine Learning ympäristön resurssit

Ympäristön luomisen jälkeen täytyi vielä luoda laskentatehoa varten Computing-virtuaalikone. Valitsin ympäristön suositteleman perusversion, jossa hinta pyöri kymmenissä senteissä per tunti. Kuvassa 2 näkyvät speksit sekä saatavilla olevat applikaatiot valitulle versiolle.



**Kuva 2:** basicComputing laskentakone ML projektille

Näiden vaiheiden jälkeen ML ympäristössä on mahdollista ajaa Jupyter Lab -notebookeja, joiden avulla aion toteuttaa projektin. Mahdollisuuksien mukaan tutkin jatkossa myös VS Code integraatiota sujuvamman koodinkirjoittamisen avuksi. Yhdistin ML ympäristön jo aiemmin luotuun Git-repositoriooni, jolloin myös oppimispäiväkirjan koodin ajaminen pilviympäristössä on mahdollista.

## Hyödyllisiä lähteitä

Azure for Students -lisenssi:

<https://azure.microsoft.com/en-us/free/students/>

Jupyter Notebookin ajaminen Azuressa:

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-run-jupyter-notebooks>

Git-repositorion liittäminen Azuren ML computeen:

<https://docs.microsoft.com/en-us/azure/machine-learning/concept-train-model-git-integration>

# datan Kerääminen

Tavoitepisteet: 3

Auton hankinta oli itselleni ajankohtainen vuodenvaihteessa, ja käytettyjen autojen markkinoiden selaamiseen tuli käytettyä rutkasti aikaa. COVID-19-kriisistä johtuen uusien autojen saatavuus on heikkoa pitkistä toimitusajoista johtuen, ja käytettyjen autojen markkinat käyvät kuumana. Harjoitustyöideaa kehiteltäessä nousi esiin ajatus koneoppimismallista, joka ennustaa käytetyn auton hinnan eri parametrein. Tutkin mahdollisuutta raapia tietoa myynti-ilmoituksista nettiauto.com -palvelusta, mutta sivuston lataamaa javascript-koodi esti ruudunraavinnan. Päädyin hakemaan tiedot autotalli.com -palvelusta. Siellä ilmoituksia on karkeasti katsottuna puolet vähemmän kuin Nettiautossa, mutta dataa kertyi siltikin yli 40 000 riviä. Päädyin siis raapimaan kaikki myynti-ilmoitukset.

Muiden harjoitustyökuvauksia selaillessa silmään osui yksi aiempi raavinta kyseiseltä sivustolta, mutta päädyin käyttämään eri menetelmää tiedon keräämiseen. Olin aiemmin tutustunut BeautifulSoup-scraperiin, ja sen käyttäminen tehtävässä tuntui luonnolliselta. Olen jonkin verran paininut HTML:n sekä javascriptin kanssa, joten oikeiden elementtien tunnistaminen verkkosivulta sujui suhteellisen ongelmitta. Alkuun haasteita aiheutti Nettiauton raapijalleni tarjoama dummy-site, jonka tunnistamiseen kului ”turhaa aikaa”.

Päädyin raapimaan sivustolta seuraavat tiedot:

#Dataset for scraped data. This dataset is saved into .csv file as a backup, so no need to scrape multiple times if an error with later coding is occurred.

raw\_data = {

    'Make': [],

    'Model': [],

    'Year': [],

    'Odometer': [],

    'Sales price': []

}

Tähän olisi voinut lisätä vielä varustetietoja, mutta päätin tässä vaiheessa, että koneoppimismallista olisi tullut turhan haastava. Päädyin käymään nettisivun osoitteita läpi yksinkertaisella for-loopilla sekä muuttamalla nettiosoitetta seuraavalle riville suhteellisen yksinkertaisesti.

for note in salesNotes:

        details = note.find(class\_="carsListItemCarNameContainer").find(class\_="carsListItemNameLink").text

        #Take car make and model out of details

        details\_splitted = details.split()

        make = details\_splitted[0]

        model = details\_splitted[1]

        year = note.find(class\_="usedCarsListItemCarModelYear").find(class\_="carsListItemCarBottomContainerItem").text

        price = note.find(class\_="carsListItemCarPrice").find(class\_="carsListItemCarBottomContainerItem").text

        odo = note.find(class\_="usedCarsListItemCarMeterReading").find(class\_="carsListItemCarBottomContainerItem").text

        #Add data to vector

        raw\_data['Make'].append(make)

        raw\_data['Model'].append(model)

        raw\_data['Year'].append(year)

        raw\_data['Odometer'].append(odo)

        raw\_data['Sales price'].append(price)

Tähänkin olisi voinut toteuttaa hienomman viritelmän, kuten koodiklinikan Scrapy-esimerkissä tehtiin, mutta nettiosoitteen muuttaminen jokaisessa loopissa osoittautui myös toimivaksi.

## Hyödyllisiä lähteitä

BS4 lähteitä

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<https://beautiful-soup-4.readthedocs.io/en/latest/>

<https://realpython.com/beautiful-soup-web-scraper-python/>

# Jalostaminen