Hermanni Taimisto 50210361

Harjoitustyön kuvaus

Johdanto datatieteeseen

SISÄLLYSLUETTELO

[1. Kehitysympäristö 1](#_Toc101948454)

[1.1 Hyödyllisiä lähteitä 2](#_Toc101948455)

[2. datan Kerääminen 3](#_Toc101948456)

[2.1 Hyödyllisiä lähteitä 4](#_Toc101948457)

[3. Jalostaminen 5](#_Toc101948458)

[3.1 Hyödyllisiä lähteitä 6](#_Toc101948459)

[4. kuvaileminen 7](#_Toc101948460)

[4.1 Hyödyllisiä lähteitä 9](#_Toc101948461)

[5. Koneoppiminen 10](#_Toc101948462)

[5.1 Hyödyllisiä lähteitä 12](#_Toc101948463)

[6. Toimeenpano 13](#_Toc101948464)

[6.1 Hyödyllisiä lähteitä 15](#_Toc101948465)

.

# 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.

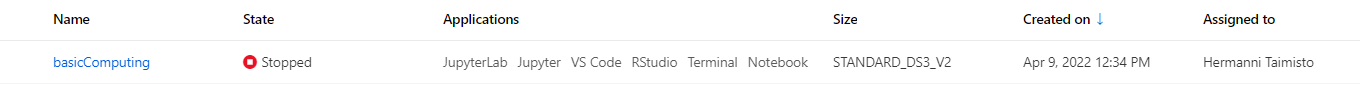
Lueskelin jo palautettuja toteutuksien kuvauksia, ja oikean lisenssin valintaa lukuun ottamatta Machine Learning -ympäristön luominen sujui ongelmitta. Rekisteröimällä Azure tilin TUNI-sähköpostilla sai opiskelijaedut käyttöönsä, joka tarkoitti 100USD edestä resursseja sekä mahdollisuus käyttää laskentatehoa. 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

Tavoitepisteet : 2

Data raavittiin autotalli.com-sivustolta. Itse data oli jo raavittuna suhteellisen siistiä, mutta esimerkiksi raapimisen aikana tuli haettua jonkin verran samoja ilmoituksia, mutta nämä poimittiin pois jo ennen datasetin tallentamista csv-muotoon. Duplikaatit johtuivat luultavasti joka sivulla pyörivistä ”ohituskaista” -ilmoituksista.

#Write vector to pd dataframe

dfRaw = pd.DataFrame(raw\_data)

dfRaw.drop\_duplicates(inplace=True)

Datasettiin ilmaantunut turha ensimmäinen kolumni poistettiin heti alkuunsa

Table

Description automatically generated

data.drop(columns = ['Unnamed: 0'], inplace = True)

Seuraavaksi päädyin tarkastelemaan uniikkeja arvoja eri luokissa, ja kilometrilukemien sekä myyntihintojen perusteella poistin sellaiset rivit, jotka viittasivat leasing-sopimuksiin tai ei-tieliikennekelpoisiin autoihin.

#Drop all rows where car has been driven under 5000 km

for index, row in data.iterrows():

    if row['Odometer'] < 5000:

        data.drop(index, inplace = True)

#Drop all rows where Sales price in under 1000 euros

for index, row in data.iterrows():

    if row['Sales price'] < 1000:

        data.drop(index, inplace = True)

Jalostamisen jälkeen datasetissä on 45 037 riviä.

Table

Description automatically generated

Haastavaa jalostamisessa oli aineiston koko, sillä yksittäisien poikkeavien arvojen löytäminen manuaalisti olisi ollut turhan työlästä. Tähän soveltui näppärästi unique() -komento. Mietityttämään jäi myös tämän vaiheen helppous, sillä en oikein keksinyt enempää jalostamisen aiheita. Varmasti visualisoinnin aikana tulee lisää asioita mieleen.

## Hyödyllisiä lähteitä

Pandas-dokumentaatio

<https://pandas.pydata.org/pandas-docs/stable/>

Drop duplicates

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html>

# kuvaileminen

Tavoitepisteet: 3

Datan visualisoinnin *scatter matrixia* lukuun ottamatta toteutin Power BI:ssa. Alkuperäisenä ajatuksena oli yhdistää training-data ML datasettinä PowerBI:n, jotta muutokset datassa olisivat näkyvissä raportissa. Tämä ei kuitenkaan ollut suoraviivaista, sillä kehitysympäristö vakiotietorakennetta ei saanut yhdistettyä PowerBI:n, vaan tiedosto olisi pitänyt linkittää erilliseen Bulb-tietorakenteeseen. Päädyin lopulta vain lataamaan jalostetun training-datasetin PowerBI:n csv-muodossa, sillä lopulta se on kuitenkin täysin staattinen. Visualisointi löytyy osoitteesta: <https://app.powerbi.com/view?r=eyJrIjoiODk2M2IzZWYtOWJmZi00MmVmLWEzZGMtOWZkMzhhODVkYTZlIiwidCI6ImZhNjk0NGFmLWNjN2MtNGNkOC05MTU0LWMwMTEzMjc5ODkxMCIsImMiOjl9&pageName=ReportSection> . Alun perin raportti piti myös upottaa Notebookiin, mutta Azure ML ei tätä vielä tukenut toimivasti.

Päädyin luomaan interaktiiviset sekä staattiset sivut raportille. Staattisen puolen kuvaajia toteutin osittain PowerBI ja Python integraatiolla, mutta esimerkiksi alussa mainittu *scatter matrix* ei ollut mahdollista toteutttaa. Korrelaatioiden tarkastelemiseen löytyi mukava lisäosa ”*correlation plot”.* Kyseiset python-kaaviot eivät näy PowerBI:n verkkoon julkaistussa versiossa, mutta alla olevassa kuvassa ne ovat esillä.

Graphical user interface

Description automatically generated

Haasteita aiheutti Python-koodin ajaminen PowerBI desktopissa. PowerBI tuki ainoastaan versioon 3.7.7 asti olevia kääntäjiä, joten jouduin asentamaan uuden ympäristön. Lisäksi tarvittavat paketit (matplotlib, pandas) joutui asentamaan tietokoneen komentokehotteen kautta, ennen kuin ensimmäinenkään rivi koodista suostui ajamaan.

Graphical user interface, text, application

Description automatically generated

En ollut ennen ajanut Python koodia PowerBI:ssä, joten tämä osoittautui mukavaksi tehtäväksi. Koodia on mahdollista muokata esimerkiksi Visual Studiossa ”yhden napin painalluksella”, joten myös laajempi debuggaaminen on suhteellisen helppoa. PowerBI ei kuitenkaan tue vielä koodin tuomista VS Codesta takaisin päin, joten copy-paste -hommia siinäkin on tiedossa. Koin, että visualisointien luominen PowerBI:ssa osoittautui huomattavasti tehokkaammaksi kuin ainoastaan Python-koodin vääntäminen.

## Hyödyllisiä lähteitä

Pythonin ajaminen PowerBI:ssä

<https://docs.microsoft.com/en-us/power-bi/connect-data/desktop-python-scripts>

Visualisointien luominen PowerBI:ssä Pythonilla

<https://docs.microsoft.com/en-us/power-bi/connect-data/desktop-python-visuals>

# Koneoppiminen

Tavoitepisteet: 2

Koneoppimisen päätin toteuttaa täysin Azure ML ympäristössä, sillä tämä antaa parhaat valmiudet työelämää varten. Valmiiden työkaluiden käyttö mahdollistaa tehokkaat oppimismallit sekä käyttäjäystävälliset integraatiot. Tavoitteena on viimeisessä vaiheessa luoda notebookin käyttäjälle mahdollisuus hakea ennuste auton hinnalle parametrilistan perusteella. ML mallin luomisen jälkeen se on mahdollista ”deployata” web-ympäristöksi. ML ympäristöä hyödynnettäessä mielenkiintoista on myös se, että datasetissä tekstimuotoisia auton merkkiä ja mallia ei tarvinnut muuttaa numeromuotoon. Tämä olisi toki onnistunut suhteellisen lyhyellä koodilla.

Ensimmäiseksi loin edellisessä vaiheessa analysoidusta training\_datasta datasetin ML ympäristöön tabulaarisessa muodossa. Tämän jälkeen ML mallin luominen tapahtuu valitsemalla ”Automated ML”, sekä täyttämällä siihen tarvittavat tiedot. Ennustettava parametri on auton myyntihinta. Laskentavaiheessa valitsin monimuuttujaregression normalisoidulla neliöllisellä keskiarvolla.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Seuraavassa vaiheessa valitsin testidataksi 15% alkuperäisestä datasetistä, jolloin mallin soveltuvuutta voidaan testata.

Graphical user interface, text, application, email

Description automatically generated

Malli antoi seuraavilla painotuksilla ennusteen myyntihinnan määräytymiselle. Merkki sekä vuosimalli vaikuttavat tässä mallissa hintaan eniten.

Chart, bar chart

Description automatically generated

Mallin testaus datasetin 15%:lla antoi alla olevassa kuvassa näkyvät tulokset. Selitysaste 88% sekä pieni keskivirhe viestivät suhteellisen onnistuneesta mallista.

Haasteita ML ympäristön käyttö aiheutti erityisesti laajoista ominaisuuksista johtuen. Ympäristössä on niin paljon mielenkiintoista testausvaiheessa olevia ominaisuuksia, että aivan perus monimuuttujaregression tekemiseen vaaditaan aika paljon paneutumista. Itse ympäristön hienouteen päästään paneutumaan (toivottavasti) seuraavassa vaiheessa.

## Hyödyllisiä lähteitä

Deploy ML model in Azure

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where?tabs=azcli>

Azure ML dataset

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-create-register-datasets>

# Toimeenpano

Tavoitepisteet: 2

Edellisessä kohdassa hyödynnetty Azure ML ympäristö antaa käyttäjäystävällisen ratkaisun luomiseen erinomaiset eväät. Harjoitustyön tavoitteena on ennustaa auton markkinakelpoista hintaa esimerkiksi myynti-ilmoitusta tai ostotarjousta laatiessa.

Koneoppimismallin luomisen jälkeen tämän mallin pystyi deployamaan Azuressa web-requesteja tukevaksi ympäristöksi:

Graphical user interface, text, application, email

Description automatically generated

Kyselyihin valitsin käytettäväksi kiinteän avainautentikaation, jolloin Azuren luoma ”Key” upotettiin notebookiin luotuun koodiin. Endpoint tarjosi koodipohjan muutamalle eri ohjelmointikielelle:

Graphical user interface, application, Word

Description automatically generated

Tätä koodia jalostamalla notebookiin luotiin funktio sekä muutaman rivin koodi, joka kysyy käyttäjältä parametrit, muodostaa kyselyn niiden perusteella, muotoilee returnina saatavan hinnan (biteistä intiksi) sekä tulostaa näytölle ennustetun hinnan:

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Lopputuloksena projektissa saatiin luotua käyttäjälle ”sovellus”, jossa pystyy tarkistamaan auton sen hetkisen markkinahinnan perustuen autotalli.com-sivustolta haettuihin tietoihin. Raportti on siltä osin staattinen, että varsinkin tässä maailmantilanteessa autojen (polttomoottori vs. sähkö) hinnat saattavat vaihdella rajustikin. Tästä johtuen toimivampi ratkaisu olisi hyödyntää esimerkiksi Nettiauton tarjoamaa rajapintaa sekä hakea sieltä joka pyynnön yhteydessä markkinahinta, sekä verrata sitä jo aiemmin opetettuun oppimismalliin. Pienellä testauksella todettuna malli kuitenkin ennustaa suhteellisen hyviä tuloksia autojen hinnoille. Malli antaa hinnan hieman yläkanttiin, varsinkin kun huomioidaan, että alkuperäinen data on myynti-ilmoitusten perusteella haettua dataa. Lopullista myyntihintaa ei tästä palvelusta ole mahdollista hakea ainakaan julkisesti. Lisää tarkkuutta mallille olisi tuonut ottamalla tarkasteluun mukaan varustetietoja kuten vaihteisto, lisävarusteet sekä moottorin tyyppi. Ehkä tällainen projekti tuleekin tehtyä, jos API-ohjelmoinnista karttuu lisää kokemusta :).

## Hyödyllisiä lähteitä

Deploy ML model

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where?tabs=azcli>

Python multiregression

<https://www.w3schools.com/python/python_ml_multiple_regression.asp>

JSON encoding and decoding

<https://docs.python.org/3/library/json.html>