AIM learning

Koneoppi pythonilla

Oppijan taso odotus: Keskitaso

Tämä materiaali on tarkoitettu henkilölle, joka osaa ehkä hieman Pythonia, mutta koneoppi tulee uutena. Ohje on suunnattu Windowsille, missä käytetään Pythonin virtuaaliympäristöjä ja Jupyter Notebookkia.

Mitä osaamispaketti sisältää?

Harjoituksen aikana tehdään neljä asiaa:

1. Luodaan Python-projektille oma kansio
2. Luodaan ja aktivoidaan virtuaaliympäristö
3. Asennetaan tarvittavat kirjastot
4. Ajetaan kolme hyvin yksinkertaista koneoppimisesimerkkiä Jupyter Notebookissa

Käytettävät algoritmit ovat:

Algoritmi	Mitä sillä tehdään?
Lineaarinen regressio	Ennustetaan numeroarvoa
KNN-luokittelija	Luokitellaan havainto lähimpien esimerkkien perusteella
Päätöspuu	Tehdään luokittelu yksinkertaisten sääntöjen avulla

---

Ennen aloittamista

Varmista, että koneella on Python asennettuna. Avaa komentorivi (cmd) ja tarkista Pythonin asennus:

```
python –version
```

Jos tämä näyttää esimerkiksi version `Python 3.11.x` tai `Python 3.12.x`, kaikki on hyvin. Muuten, Python ei todennäköisesti ole oikein asennettu tai se ei ole mukana Windowsin PATH-asetuksissa.

Projektikansion luominen

Luo uusi kansio koneoppimisharjoitusta varten. Kirjoita komentorivillä:

```
mkdir ml-aloitus
cd ml-aloitus
```

Komento `mkdir` tekee uuden kansion ja komento `cd` siirtyy kyseiseen kansioon. Tällä tavoin kaikki harjoituksen tiedostot tulevat kansioon `ml-aloitus`.

Euroopan unionin osarahoittama

Elinvoimakeskus

centria University of Applied Sciences

Kokkola Karleby

kpedu

AIM learning

Virtuaaliympäristön luominen

Mikä virtuaaliympäristö on?

Pythonin virtuaaliympäristö venv on projekteja varten tarkoitettu työkalu, mihin voi asentaa kirjastoja, jotka ovat asennettu vain ympäristöön eikä koko tietokoneelle. Tämä on hyvä käytäntö, kun eri projektit tarvitsevat omat kirjastot tai eri versiot niistä.

Virtuaaliympäristön luonti

Kirjoita projektikansiossa:

```
python -m venv ml_venv
```

Tämä luo projektikansioon uuden kansion nimeltä `ml_venv`, joka tekee projektille oman Python-ympäristön.

Virtuaaliympäristön aktivointi

Virtuaaliympäristö pitää aktivoida aina, kun aloitat työskentelyn tässä projektissa uudessa komentorivi-ikkunassa. Avaa Command Prompt ja aja:

```
ml_venv\Scripts\activate
```

Tämä aktivoi virtuaaliympäristön ja täällä kaikki ajetut käskyt tapahtuvat ympäristön sisässä.

Kirjastojen asentaminen

Kun virtuaaliympäristö on aktiivinen, asenna tarvittavat kirjastot (asennus kestää hetken):

```
pip install numpy pandas matplotlib scikit-learn jupyter
```

Kirjastot tarkoittavat käytännössä valmiita työkaluja:

Kirjasto	Mihin sitä käytetään?
numpy	Numeroiden ja taulukoiden laskentaan
pandas	Taulukkomuotoisen datan käsittelyyn
matplotlib	Kuvaajien piirtämiseen
scikit-learn	Koneoppimismallien tekemiseen
jupyter	Notebookien ajamiseen selaimessa

Voit tarkistaa, että asennus onnistui:

```
pip list
```

Listassa pitäisi näkyä ainakin `numpy`, `pandas`, `matplotlib`, `scikit-learn` ja `jupyter`.

Euroopan unionin osarahoittama

Elinvoimakeskus

centria University of Applied Sciences

Kokkola Karleby

kpedu

Jupyter Notebookin käynnistäminen

Jupyter on Python ohjelmointi ympäristö, jossa on helppo käsitellä kansioita ja sen notebookeissa on helppo ajaa koodi pienissä pätkissä. Jättäen kaikki ajot välimuistiin niin kuin kesken ohjelma-ajon. Tämä toimii erityisen hyvin koneoppi tehtäviin, missä välillä pitää tehdä pidempiä koulutuksia. Tällä tavoin ei välttämättä joka hetki tarvitse huomioida näiden tulosten tallentamista ja säilömistä.

Kun olet edelleen projektikansiossa ja virtuaaliympäristö on aktiivinen, aja komentorillä:

```
jupyter notebook
```

Tämä avaa selaimeen Jupyter Notebook -näkymän selaimeen.

Luo uusi notebook valitsemalla:

```
New -> Python 3
```

Anna notebookille esimerkiksi nimi:

```
ensimmainen-koneoppiminen.ipynb
```

Miten Notebookia käytetään?

Notebook koostuu soluista. Solu voi sisältää:

- Python-koodia
- Tekstiä
- Kuvia
- Taulukoita
- Tuloksia

Koodisolu ajetaan painamalla (tai play kolmiota):

```
Shift + Enter
```

Kokeile ensimmäiseen soluun:

```
print("Hei koneoppiminen!")
```

Aja solu painamalla `Shift + Enter`. Jos näet tulosteen, notebook toimii.

Seuraavaksi testataan kirjastot:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

print("Kirjastot toimivat!")
```

Euroopan unionin osarahoittama

Elinvoimakeskus
centria University of Applied Sciences
Kokkola Karleby
kpedu

AIM learning

Koneoppimisen perusidea

Koneoppimisessa tietokoneelle annetaan esimerkkidataa, josta mallin tehtävä on oppia datasta jonkinlainen yhteys. Esimerkiksi:

Syöte	Ennustettava asia
Asunnon koko	Asunnon hinta
Kukan mitat	Kukan laji
Sähköpostin sisältö	Onko se roskapostia?

Yksinkertainen työnkulku on yleensä tämä:

1. Otetaan data
2. Jaetaan data opetusdataan ja testidataan
3. Opetetaan malli opetusdatalla
4. Pyydetään mallia tekemään ennusteita testidatalla
5. Katsotaan, kuinka hyvin malli onnistui

Tärkeät termistö:

Sana	Selitys
Data	Esimerkit, joista malli oppii
Feature	Yksi selittävä tieto (sarake), esimerkiksi asunnon koko
Target / Label / Luokka	Se asia, jota yritetään ennustaa, esim. kukkalaji
Malli	Ohjelma, joka oppii datasta
Opetusdata	Data, jolla malli opetetaan
Testidata	Data, jolla tarkistetaan, osaako malli ennustaa uutta dataa

Esimerkki 1: Lineaarinen regressio

Regression kuuluu useampia ”versioita”, jotka pyrkivät ennustamaan numero arvoa, yleensä jatkuvaan trendimäiseen dataan. Lineaarinen regressio on tästä yksinkertainen tapaus, missä pyritään dataan sovittamaan sitä parhaiten kuvaava suora viiva.

Tässä esimerkissä luomme itse pienen datan. Ajatus on:

> *Mitä suurempi x-arvo, sitä suurempi y-arvo.*

Tämä voisi muistuttaa esimerkiksi tilannetta, jossa asunnon koko vaikuttaa hintaan.

Datan luominen

Aja notebookissa:

```
import numpy as np
import matplotlib.pyplot as plt

# Luodaan satunnaislukugeneraattori, jotta tulos on sama joka ajokerralla
rng = np.random.RandomState(42)

# X on syötemuuttuja
X = 10 * rng.rand(50, 1)
```

Euroopan unionin osarahoittama

Elinvoimakeskus
centria University of Applied Sciences
Kokkola Karleby
kpedu

```
# y on ennustettava arvo
# y riippuu X:stä, mutta mukana on hieman satunnaista vaihtelua
y = 2 * X.squeeze() + 3 + rng.randn(50) * 2

plt.scatter(X, y)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Esimerkkidata")
plt.show()
```

Mallin opettaminen

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Jaetaan data opetus- ja testidataan
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Luodaan malli
model = LinearRegression()

# Opetetaan malli
model.fit(X_train, y_train)
```

Tässä tapahtuu kolme asiaa:

1. Data jaetaan kahteen osaan (train ja test)
2. Luodaan lineaarinen regressiomalli
3. Malli opetetaan opetusdatalla

Ennusteiden tekeminen

```
y_pred = model.predict(X_test)

print(y_pred)
```

`y_pred` sisältää mallin tekemät ennusteet.

Tuloksen katsominen kuvana

```
plt.scatter(X_test, y_test, label="Todelliset arvot")
plt.scatter(X_test, y_pred, label="Mallin ennusteet")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.title("Todelliset arvot ja ennusteet")
plt.show()
```

Yksinkertainen arviointi

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)
```

```
print("Keskimääräinen virhe:", mae)
```

Keskimääräinen virhe kertoo suunnilleen, kuinka paljon mallin ennuste poikkeaa oikeasta arvosta, mutta tämän luvun merkitys vaihtelee paljon operointi skaalasta.

Pohdi:

- Ovatko ennusteet lähellä oikeita pisteitä?
- Mitä tapahtuisi, jos datassa olisi enemmän satunnaista vaihtelua?

Esimerkki 2: KNN-luokittelu

KNN (k-nearest neighbors) on koneoppimisalgoritmi, jossa tarkastellaan datapisteen ympäristön ns. naapureita. Valitaan luku k, joka on huomioitavien naapureiden määrä. Näiden naapureiden luokkien perusteella datapiste liitetään enemmistön luokkaan. Eli ei enää ennusteta numeroa vaan joukkoa.

Tässä käytetään valmista Iris-dataa, joka on usein esimerkki datana koneoppimisalgoritmeille. Iris-data sisältää tietoja kukista, missä jokaisesta kukasta on mitattu esimerkiksi terälehden pituus ja leveys. Mallin tehtävä on päätellä mittojen perusteella, mikä kukkalaji on kyseessä.

Datan lataaminen

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()

X = iris.data
y = iris.target

print(iris.feature_names)
print(iris.target_names)
```

Tässä datassa:

- X sisältää kukan mitat
- y sisältää oikean kukkalajin

Datan katsominen taulukkona

```
df = pd.DataFrame(X, columns=iris.feature_names)
df["target"] = y

df.head()
```

`df.head()` näyttää taulukon ensimmäiset rivit.

Datan jako

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

KNN-mallin opetus

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Tässä `n_neighbors=3` tarkoittaa, että malli katsoo kolmea lähintä naapuria päätöksen tekoon.

Ennustaminen

```
y_pred = knn.predict(X_test)

print(y_pred)
print(y_test)
```

Ensimmäinen rivi näyttää mallin ennusteet.

Toinen rivi näyttää oikeat vastaukset.

Onnistumisen mittaaminen

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Oikein menneiden osuus:", accuracy)
```

Jos tulos on esimerkiksi `0.966`, se tarkoittaa, että noin 96,6 % testihavainnoista meni oikein.

Pohdi:

- Kuinka moni meni oikein?
- Mitä tapahtuu, jos muutat `n_neighbors=3` arvoksi `1` tai `10`?

Esimerkki 3: Päätöspuu

Yksinkertaisuudessaan päätöspuu pyrkii tekemään tällaisia sääntöjä:

```
Jos terälehden pituus on pieni, kukka on todennäköisesti setosa.
Muuten kysytään seuraava kysymys.
```

Päätöspuu on aloittelijalle hyvä malli, koska sen toimintaa on helppo tarkastella.

Euroopan unionin osarahoittama

Elinvoimakeskus
centria University of Applied Sciences
Kokkola Karleby
kpedu

Mallin opetus

Käytetään samaa Iris-dataa kuin edellisessä esimerkissä. Mikäli teit uuden pohjan niin ota siis KNN-malli tehtävästä kaikki datan käsittelyyn liittyvät vaiheet osaksi uutta pohjaa.

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)
```

`max_depth=3` viittaa päätöspuun syvyyteen, eli maksimissaan tehdään kolme peräkkäistä jakoa. Päätöspuun piirtämisosiosta (eteenpäin koodi esimerkissä) on helppo hahmottaa mitä tämä meinaa. Kolmen syvyinen päätöspuu on hyvin yksinkertainen tapaus hahmottaa.

Ennustaminen

```
y_pred = tree.predict(X_test)

print(y_pred)
print(y_test)
```

Ensimmäinen rivi näyttää mallin ennusteet.

Toinen rivi näyttää oikeat vastaukset.

Arviointi

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Oikein menneiden osuus:", accuracy)
```

Päätöspuun piirtäminen

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plot_tree(
    tree,
    feature_names=iris.feature_names,
    class_names=iris.target_names,
    filled=True
)
plt.show()
```

Kuva näyttää päätökset mitä puu tekee.

Pohdi:

- Mitä ensimmäinen päätös kysyy?
- Onko puun idea helpompi ymmärtää kuin KNN:n?

AIM learning

Mitä kannattaa ymmärtää tästä harjoituksesta?

Tämän materiaalin jälkeen tärkeintä ei ole muistaa kaikkia koodia ulkoa vaan ymmärtää hieman mikä idea koneopin takana on.

Tähän liittyen oleellinen tieto ymmärtää on tämä peruskaava, joka toistuu hyvin monessa mallissa koneoppimiseen:

```
1. Data
2. Opetus- ja testijako
3. Mallin luonti
4. Mallin opetus
5. Ennusteet
6. Arviointi
```

Yhteenveto algoritmeista

Algoritmi	Käyttö	Selitys yhdellä lauseella
Lineaarinen regressio	Numeroarvon ennustaminen	Sovittaa datan läpi mahdollisimman hyvän suoran
KNN	Luokittelu	Päättelee luokan lähimpien esimerkkien perusteella
Päätöspuu	Luokittelu	Tekee päätöksiä opittujen sääntöjen avulla

Lopputehtävä

Tee notebookkien loppuun lyhyt yhteenveto omin sanoin.

Vastaa esimerkiksi näihin:

1. Mikä oli opetusdatan ja testidatan ero?
2. Mitä `fit()` tekee?
3. Mitä `predict()` tekee?
4. Mikä kolmesta mallista tuntui helpoimmalta ymmärtää?
5. Mikä vaihe tuntui vaikeimmalta?
6. Koititko muuttaa eri arvoja (parametrejä) algoritmeille? Miten nämä vaikuttivat tuloksiin?

Jupyter notebookista saa koodit helposti talteen File -> Save and Export Notebook As -> HTML.
Monet muut tallennusmuodot ovat myös hyödyllisiä, mutta vaatii muita lisäosia toimiakseen.

Valmista!

Nyt tiedät hieman perusteita siitä, miten koneoppimis- ja tekoälysovellukset toimivat käytännössä. Vaikka monimutkaisemmat rakenteet, kuten neuroverkot, ovat hankalampia ymmärtää, myös niistä on mahdollista rakentaa yksinkertaisia toteutuksia melko suoraviivaisesti.

Euroopan unionin osarahoittama

Elinvoimakeskus
Kokkola Karleby
centria University of Applied Sciences
kpedu

Tämän materiaalin tuottamisessa on hyödynnetty kielimalleja, mutta sisältö on vahvasti editoitu ja viimeistelty. Nämä työkalut ovat erinomaisia apuvälineitä oppimiseen, ja ne voivat auttaa sinua etenemään aiheessa.

Jos olet kiinnostunut oppimaan lisää, hyviä seuraavia askeleita ovat esimerkiksi datan siistimiseen ja puuttuvien arvojen käsittelyyn tutustuminen, yksinkertaisen monitasoisen perceptronin toteutus (esimerkiksi käsin kirjoitettujen numeroiden tunnistukseen) tai hyperparametrien optimointi. Koneoppimisen kenttä on laaja ja tarjoaa valtavasti tutkittavaa. Tervetuloa polulle!

Euroopan unionin osarahoittama

Elinvoimakeskus

Kokkola Karleby

centria University of Applied Sciences

kpedu