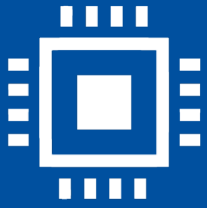
NTNU Norges teknisk-naturvitenskapelige universitet

Prosjektinnlevering - Fagnært prosjekt

Eksamen - del 2

INGT1001



Innholdsfortegnelse

Introduksjon

Eksamen i INGT1001 består av to deler:

1. En individuell prøve som gjennomføres på Inspera 11/12. Den omhandler tematikken som ikke er Python eller Fagnært prosjekt. Se nærmere info under folderen Eksamensinfo på hovedsiden til INGT1001 i Blackboard.
2. **Prosjektinnlevering - Fagnært prosjekt. Denne gjelder kun Elektro, og er den innleveringen som dette dokumentet handler om. Frist 11/12 kl. 12.00.**
    a. **Gruppeinnlevering av prosjektrapporten i Inspera (en av gruppens deltakere leverer på vegne av hele gruppa)**
    b. **Videopresentasjon av prosjektet (Frist vil oppgis på BB)**

Prosjektinnleveringen er altså både en del av eksamen og avslutningen av prosjektdelen i INGT1001 høsten 2020. I den skal dere vise hva dere har tilegnet dere av kunnskap på følgende temaområder, samtidig som det forutsettes et grunnlag i Python fra programmerings-modulen i emnet:

- Raspberry Pi (RPi) og Sense Hat (SH).
    - Grunnleggende kommandoer i Linux terminal for å manøvrere rundt i RPi'en og administrere filer og brukere.
    - Programmering av RPi og SH med Python for innsamling og behandling av sensordata fra SH, styring av RGB-matrisa på SH, etc.
- Dataanalyse med Python
    - Gangen i et dataanalyseprosjekt: Stille opp sentrale spørsmål som det søkes svar på, innsamling av relevante data, vasking av data, uttrekk av spesifikke elementer, analyse for å finne svar, og visualisering av resultater.
    - Verktøy i dataanalysen, slik som numpy, matplotlib, Pandas.
    - Besvare selvvalgte spørsmål rundt egne data innsamlet med RPi/SH.
    - Analysere større datasett for å besvare sammensatte spørsmål.
- Programutviklingsverktøy
    - Anaconda-distribusjonen med sine pakker

- Spyder: utviklingsomgivelse (IDE - Integrated Development Envirornment)
- Jupyter Notebook: en interaktiv en programomgivelse som gjør det mulig å skrive dokumenter som inneholder direkte kjørbar Python-kode, interaktive widgets, plott, tekst, likninger, bilder og videoklipp. Vi har innledningsvis benyttet Jupyter Notebook for å skrive og kommentere/plotte mindre kodesnutter, men dens primære bruksområde er formidling og rapportering, og det skal dere benytte den til her. Mer om rapporten i senere kapittel.

Vi kan betrakte prosjektet som bestående av to elementer: innsamling av data vha. RPi/SH, og et dataanalyse-element. Vi tar for oss disse to elementene i det etterfølgende:

Element 1 – Datainnsamling

Det skal programmeres en Raspberry Pi tilknyttet sensorkortet Sense Hat for innsamling av data. Hver gruppe definerer selv spørsmålene de ønsker svar på. Det skal så samles inn data som behandles og analyseres for å finne svar på spørsmålene. Det kan også være at nye spørsmål dukker opp i denne prosessen og erstatter tidligere – det er helt greit. Gruppene skal dessuten demonstrere enkel bruk av «services» i systemd i Linux, som skal brukes til å sørge for at datainnsamlingen kjører i bakgrunnen av operativsystemet. Dermed vil det ikke være nødvendig å holde en ssh-tilkobling åpen for at Python-programmet som sørger for datainnsamlingen holdes gående.

Element 2 – En større dataanalyse

En komplett dataanalyse skal kjøres med utgangspunkt i gitte, store datasett og forhåndsdefinerte spørsmål. Dette er med andre ord en større og mer kompleks versjon av dataanalysen som skulle utføres på de egeninnsamlede data i Element 1. Det blir gitt en oppgaveliste som skal lede dere gjennom hele dette elementet.

Forberedende arbeid – systemd

Hva kan systemd gjøre for oss?

Ved bruk av systemd vil man kunne kjøre kode på Raspberry Pi-en uten at det må kjøres i et shell. I dette tilfelle vil «et shell» tilsvare PuTTY eller annen ssh-tilkobling. Systemd tillater bruker altså å kjøre kode som en bakgrunnsprosess. I tillegg til dette kan systemd brukes til å kjøre bakgrunnsprosesser automatisk ved oppstart. Det betyr at dersom Pi-en skrur seg av og på under et målingsintervall, vil koden starte opp på nytt igjen uten ytre påvirking. Dette er en ekstra praktisk løsning dersom det skal gjøres målinger på en plass uten WiFi; da slipper en å bruke PC for å starte Pi-en på nytt.

Hvordan bruke systemd?

Systemd trenger en Unit-fil for hver bakgrunnsprosess, som beskriver hva som skal skje. Denne må en selv lage og legge i /etc/systemd/system. Den er av filtypen «.service», eksempelvis: «vaermaaling.service». Under viser vi hvordan innholdet i .service-filen skal se ut for formålet beskrevet videre i element 1. Det vil si å kunne kjøre en Python-fil uten shell.

```
[Unit]
After=multi-user.target

[Service]
ExecStart=sudo python3 FILPLASSERING HER!

[Install]
WantedBy=multi-user.target
```

Bruk av .service-filen

Før man kan kjøre .service-filen er man nødt til å skrive *«sudo systemctl daemon-reload».* Dette er for å oppdatere operativsystemets oversikt over filene og hvordan de er avhengige av hverandre. Deretter, for å kjøre en service skriver man *«sudo systemctl start filnavn».* Merk at filplasseringen ikke skal skrives, kommandoen vet at filen ligger i /etc/systemd/system. Benytt *«sudo systemctl start filnavn»* for å stanse service-en. Service-en vil også stoppe dersom Pi-en skrus av og på.

Eksempel på hvordan starte en service: «sudo systemctl start maalinger.service»

For å få service-en til å kjøre ved oppstart skriver man *«sudo systemctl enable filnavn»*. Om man skal skru av automatisk kjøring ved oppstart brukes *«sudo systemctl disable filnavn».*

En viktig ting å tenke på

En ting man er nødt til å være forsiktig med ved bruk av *systemd* slik tidligere beskrevet, er at Python-filen nå kjøres fra root. Det vil si at en relativ path i koden vil **starte fra root**, **ikke i mappen Python-filen ligger i**. Om man da skriver koden under vil man lage en fil «datafil.txt» i root, noe som helst bør unngås, da det blir svært rotete dersom flere filer skal logges.

```
with open('datafil.txt', 'a') as f:
    f.write('test')
```

Å lage en ny fil per måling – med navn basert på dato og klokkeslett anbefales.

Hva er egentlig systemd?

Systemd er en samling programmer som til sammen har prøvd å fylle flere roller i systemet til Linux. Blant annet systemlogging, nettverktilkobling, device management, innlogging, og mye mer. Den delen av systemd vi tar i bruk er delen vi kan kalle «service manager», som er omtalt som den viktigste delen av systemd. Om man har vært borte i Windows' «task manager» så har man noe å sammenlikne med, da systemene kan likne en del.

Navnet «systemd» er et ordspill på uttrykket «System D» fra fransk «Système D», som betyr å kunne håndtere enhver situasjon. Det er også navngitt etter standarden for navngivning av «daemons», ved at siste bokstaven i navnet er «d».

En daemon er en prosess som kjører i bakgrunnen på et operativsystem. Navnet kommer fra den gresk-mytologiske skikkelsen ved samme navn; daemonene var skapninger uten noen formening om godt eller ondt. De jobbet i bakgrunnen av verden med forskjellige «automatiske» prosesser, og kan også ha vært de dødeliges «skytsengler».

Element 1 – Datainnsamling – Plan

Dere kan selv velge hvor dere tar målinger. Her er et par forslag.

Forslag A: Hjemme

Målinger kan bli tatt hjemme hos en av medlemmene i gruppa. Å legge Pi-en utenfor vinduet er grei nok løsning for innsamling av utendørs data, ellers kan spørsmål for innendørs data stilles i stedet.

Forslag B: Ute på tur

Å ta målinger i sekk på veg rundt omkring kan også gjøres. Det vil åpne for veldig annerledes spørsmål i oppgaven. Eksempelvis kan Pi-en gjøres om til en skritteller.

Forslag C: På taket til Elektroblokkene

Et *begrenset antall* grupper kan få sette opp Pi-ene sine på taket. Maksimalt to personer per gruppe vil i så fall møte opp av gangen. Gruppene skal deretter få avtalt hentetidspunkt et par dager senere hvor de kan hente Pi-ene. Dette innebærer litt mer ansvar, tidsbruk, og planlegging for gruppene som velger det. Ta kontakt med Sebastian eller Even for avtale om innsamling på tak.

Element 1 – Datainnsamling – Kriterier

Datamengde

- Den innsamlede dataen må tilsvare minst 24 timer med målinger totalt.
- Datainnskriving skal skje **hvert minutt eller oftere**.
- Minst **tre forskjellige** sensorer på Sense Hat skal logge data.

Formål

- Gruppen må stille minst to enklere **eller** ett mer omfattende spørsmål de ønsker å svare på i innleveringen. Det er eksempler på slike spørsmål senere i dokumentet.

Å endre spørsmålene man vil ha svar på underveis i prosessen, eller å legge til spørsmål, er greit.

Tips – før målingene

- Test en gang i 5 minutter. Test så en gang i 30 minutter. Forsikre dere om at dataene deres gir mening før dere sier dere klare til å ta måling over flere dager.

Element 1 – Eksempler på spørsmål til egenmålte data

Bruk disse forslagene som veiledere og inspirasjon til å velge egne spørsmål.

Eksempler på spørsmål – Enklere

- Hva er den høyeste verdien, den laveste verdien, og den gjennomsnitlige verdien av **alle** målingene våre. Hvilke av disse verdiene høres ut som de gir mening?
- Varierer jordas magnetfelt ut i fra om det er dag eller natt? Ja/nei med begrunnelse i data.
- Kan vi ut i fra temperaturmålingene våre se nøyaktig når sola går ned? Kan vi se når Pi-en ikke lenger får direkte sollys? Og kan vi se når sola står opp, og når Pi-en får direkte sollys?

Eksempler på spørsmål – Omfattende

- Hvor mye avhenger luftfuktighet av temperatur? Lag en funksjon som tar inn temperatur, fuktighet, temperaturendring, og gir ut fuktighetsendring som er **basert på deres data**. Sammenlikn med eventuelle ekte formler.
- Med hjelp fra eksterne temperatursensorer sine data, kan vi lage en funksjon for å oversette våre (sannsynligvis) feilmålte temperaturer i nøyaktig disse omgivelsene til nesten å bli den faktiske lufttemperaturen? Hva var feilkildene som førte til at våre målinger ikke ble helt reelle?
- Med Pi-en i ryggsekken eller i lomma, kan vi få Pi-en til å fungere som en skritteller? Hvor mye bommer den i forhold til en kommersiell skritteller?

Element 2 – Prosjektrapport

Denne delen av prosjektinnleveringen fokuserer på

- Bruk av Pandas-metoder til å importere, utforske, trekke ut, fjerne, og organisere data.
- Bruk av matplotlib til å visualisere data.
- Bearbeiding av data med Pandas og NumPy for å svare på reelle problemstillinger.

Innleveringsformat

- All kode skal utvikles i Spyder. Koden skal være i form av funksjoner som spiller på hverandre. Ikke start på nytt hver gang en funksjon skrives, prøv å bygge på tidligere funksjoner dersom det er mulig.
- Merk: Det trengs ikke å opprettes en funksjon for hver oppgave. Dersom det er eksempelvis tre oppgaver som kan slås sammen til en funksjon, så gjør gjerne det.
- **Importer funksjonene som en modul til en Jupyter Notebook**. Så, i forskjellige celler, gå gjennom oppgavene senere i dette dokumentet i Jupyter Notebook.
- Merk: Koden som skrives inn i Spyder, må være godt kommentert, og bør ikke svare på oppgavene. Jupyter Notebook bør derimot inneholde minst mulig kode, og heller bruke koden fra Spyder-modulen til å besvare oppgavene.

Informasjon om datafiler i det medfølgende «datasets.zip»

- Det inneholder:
    - En mappe kalt «weather» som inkluderer historiske temperaturdata mellom 2017 og 2019.
    - En mappe kalt «extra» som inneholder noen ekstra filer med temperaturdata.
    - En fil kalt «countryContinent.csv» som er et datasett med kolonner: land, landskoder, kontinentet, og annen ekstra informasjon.
- All temperaturdata er i Fahrenheit, med unntak av data i voll.csv, som er i Celsius
- Dataene ble samlet inn fra National Centers For Environmental Information og MET, Norge

Element 2 – Pandas

Oppgaver

1. Importer filen «2342202.csv» fra «datasets.zip».

2. Kjør head og info. Forklar hvordan dataen er organisert, og hvilke datatyper datasettet består av.

3. Dropp alle radene med manglende verdier og alle duplikerte rader (husk å fikse indeks). Kommenter antall gjenværende rader og gi begrunnelse for dette antallet. Tips: Prøv å se på den importerte filen i Microsoft Excel.

4. Del opp «NAME»-kolonnen i to kolonner: «Area» og «Country».

5. Del opp «Country»-kolonnen i to kolonner: «City» og «Country».

6. Arbeid med df-en fra trinn 4 og endre «Country»-kolonnen til landskoder på to bokstaver. Skriv så i rapporten to alternative framgangsmåter til hvordan dette kan gjøres, i tillegg til den benyttede metoden.

7. Importer en annen fil fra datasets.zip, og gjør trinn 3, 4, og 6.

8. Sett sammen de to df-ene du har fra trinn 6 og 7 vertikalt til én df (husk indeks). Tips: bruk pd.concat([list_of_dfs], axis =?), hvor «axis» avgjør om sammenslåingen skjer vertikalt eller horisontalt.

9. Kontroller at den resulterende df-en fra trinn 8 er riktig.

10. (FRIVILLIG) Skriv en funksjon som sjekker om elementene i landskolonnen har mellomrom, og del inn i land og by dersom de har det. Merk: funksjonen må testes på to filer, en som har byen inkludert i landkolonnen (som New York-data) og en som ikke har det.

11. Importer alle filene i «Weather»-mappen, gjør trinn 3, 4 og 6 på hver fil.

12. Importer trondheim.csv-filen separat fra «extra»-mappen og gjør trinn 3, 4 og 6.

13. Legg til en ny kolonne på df-en fra trinn 12 kalt «TAVG» som er gjennomsnittet av «TMAX» og «TMIN» kolonnene.

14. Fjern kolonnene «TMAX» og «TMIN» fra df fra trinn 13 og plasser de andre kolonnene i samme rekkefølge som i resten av værfilene.

15. Kombiner alle df-ene, inkludert trondheim sitt, til en stor df kalt «df_complete».
Tips: bruk sammenkoblingsmetoden fra trinn 8.

16. Importer «countryContinent.csv» til en ny df kalt «df_country_continent».

17. Skriv ut de ti første radene fra df-en i 16. Ta en titt på df-en. Kommenter hvordan den er strukturert og hvilke kolonner som er best egnet til å bruke fra «df_country_continent» og «df_complete» til å legge inn i en ny df: «df_complete_cont».
Tips: tenk på hvordan metoden df.merge() kan brukes.

18. Slå sammen «df_complete» og «df_country_continent» på passende kolonner som en ny df kalt «df_complete_cont».
Tips: standard «inner join» er god å bruke til å slå sammen.

19. Dropp «code_2»-kolonnen, gi nytt navn til «Country»-kolonnen til «Country Code», og gi nytt navn til «continent»- og «country»-kolonnene så de har store forbokstaver.

20. (FRIVILLIG) Forklar reduksjonen i rader etter sammenslåing i trinn 15.
Tips: sjekk landskoder i hver dataramme og sammenlign med «countryContinent.csv».

21. Lag to variabler kalt «countries_in_df» og «continents_in_df» som har lister over alle unike land- og kontinentnavn. Sorter strengverdiene i alfabetisk rekkefølge. Diskuter i detalj hvordan en innebygd Pandas-funksjon kan brukes her, og hvordan man kunne gått fram hvis man skulle lage en egen funksjon for å utføre sorteringsoppgaven.

22. Konvertér verdiene i TAVG-kolonnen i «df_complete_cont» til Fahrenheit og rund av til to desimaler. Diskuter hvordan du ville brukt Pandas til å konvertere alle temperaturverdiene til Fahrenheit hvis de opprinnelige dataene inneholdt en blanding av verdier i både Celsius og Fahrenheit, men uten enheter, tenk utenfor boksen.
Tips: google formelen for konvertering fra Fahrenheit til Celsius.

Et eksempel på en endelig dataramme er gitt her:

	STATION	NAME	DATE	TAVG	Area	Country Code	Country	Continent
0	DEW00035032	RHIEN MAIN, DE	1/1/2017	-3.33	RHIEN MAIN	DE	Germany	Europe
1	DEW00035032	RHIEN MAIN, DE	1/2/2017	0.00	RHIEN MAIN	DE	Germany	Europe
2	DEW00035032	RHIEN MAIN, DE	1/3/2017	1.11	RHIEN MAIN	DE	Germany	Europe
3	DEW00035032	RHIEN MAIN, DE	1/4/2017	2.78	RHIEN MAIN	DE	Germany	Europe
4	DEW00035032	RHIEN MAIN, DE	1/5/2017	-0.56	RHIEN MAIN	DE	Germany	Europe
...	...	...	...	...	...	...	...	...
40345	KZ000035188	ASTANA, KZ	12/27/2019	-12.22	ASTANA	KZ	Kazakhstan	Asia
40346	KZ000035188	ASTANA, KZ	12/28/2019	-5.00	ASTANA	KZ	Kazakhstan	Asia
40347	KZ000035188	ASTANA, KZ	12/29/2019	-5.00	ASTANA	KZ	Kazakhstan	Asia
40348	KZ000035188	ASTANA, KZ	12/30/2019	-13.33	ASTANA	KZ	Kazakhstan	Asia
40349	KZ000035188	ASTANA, KZ	12/31/2019	-14.44	ASTANA	KZ	Kazakhstan	Asia

Problemer

Merk: bruk «df_complete_cont» og muligens egeninnsamlet data for å svare på følgende problemer, og skriv om alle funn i rapporten.

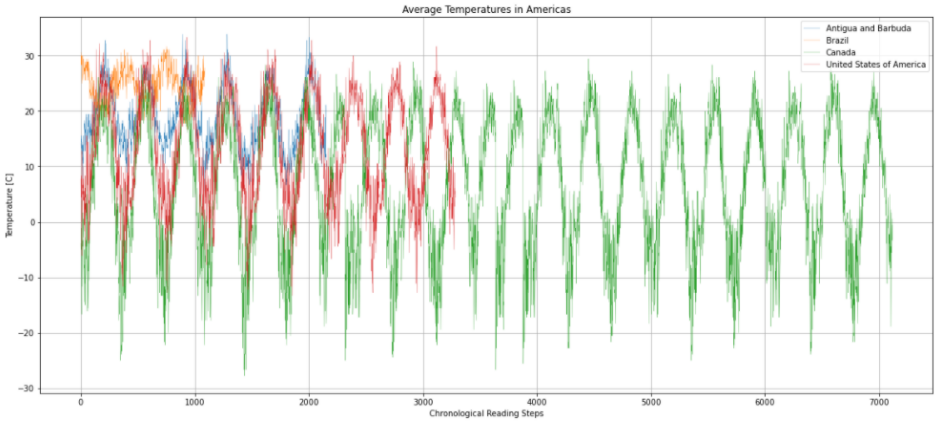
Problem 1 (FRIVILLIG): Finn den daglige gjennomsnittstemperaturen for dataene fra en ny utgave av værdataen fra Voll i Trondheim, og sammenlikn med egne målinger fra samme tidsintervall. Gjør det samme for trykkmålinger på begge plasser. Den nye utgaven vil bli gitt ut fredag 13. november 2020, så dere må i så fall ha gjort målinger før den tid.

Problem 2: Den høyeste registrerte temperaturen på jorda noensinne er 56,7 C, sjekk om dette samsvarer med all data i den nedlastede mappen.

Oppgave 3: Hvilke av de representerte kontinentene har de høyeste gjennomsnittstemperaturene og hvilke har de laveste?

Oppgave 4: Generer en figur med like mange subplots som antall kontinenter. Hver subplot må ha like mange linjer som antall land innenfor subplot-ets kontinent. På x-aksen skal det være indeks for avlesning, og på y-aksen skal du ha gjennomsnittstemperaturen for et bestemt land. Prøv å gjøre plottene så lesbare som mulig ved å endre størrelsen på plottet og bredden på linjene. Diskuter kort svingning i data, hurtig endring og andre ting å legge merke til.

Eksempel på plotting:



Problem 5: Importer «old_oslo.csv»-filen som en finner i «extra»-mappen. Utfør trinn 3, 4 og 6 på datasettet. Sammenlign året som er til stede i den filen med målingene fra oslo i «df_complete_cont» for året 2019. Det kan sammenlignes gjennom en plott. Er det noen endring i temperatur?

Problem 6: Kom med et spørsmål som gruppen finner interessante som dere kan få svar på ut ifra dataene. Spørsmålet burde ha nogen lunde grei vanskelighetsgrad, og/eller være veldig interessant.

Oppgave 7: Alle oppgitte temperaturdata er gjennomsnittlige daglige temperaturer. Anta at kapasiteten til å registrere data hadde vært med mye høyere oppløsning, hvordan ville det påvirket resultatene våre? Diskuter også hvilke faktorer som bidrar til en jevn dataanalyseprosess, og hvilke vanlige problemer man bør se etter i store datasett.

Rapporten

Rapporten skal leveres i to formater: Jupyter Notebook- og html-fil. Det må altså skrives en rapport i Jupyter Notebook, koden må kjøres, for så å eksporterere til html, slik at kodens resultater vises i html-filen.

I rapporten skal det bl.a. formidles:

- Overordnet:
    - Rapporten skal være så profesjonell som mulig. Det vil være eksempler på rapportoppsett tilgjengelig på Blackboard.
- Fra Element 1:
    - Kort om spørsmålene dere ville ha svar på
    - Kort om hvordan koden deres fungerer
        - Legg også ved koden (den skal være godt kommentert)
    - Kort om hvordan dere satte opp *systemd*
    - Hvordan dere analyserte dataen deres, **med celler som kan kjøres for å se resultatene deres**
        - Hvordan dataen deres besvarer spørsmålene dere stilte
- Fra Element 2:
    - Korte beskrivelser av hvordan oppgavene ble løst. Inkludér kjørbare kodesnutter tilhørende hver oppgave.
    - Svar på problemene
        - Vedlegg av modulen dere importerte til bruk på oppgavene
- Utfyllende om eventuelle problemer dere møtte på, hvordan dere håndterte dem, og hvordan dere ville ungått/løst dem videre.

Begrepet "kort" i denne sammenheng defineres som å inkludere kun det som er mest relevant.

- Videopresentasjonen som skal leveres er nærmere beskrevet i dokumentet "Videopresentasjonen - Krav" i folderen Prosjektet på BB-siden for Elektro (dvs. i organisasjonen INGT1001 Elektro 2020 Høst)