AI og data mini projekt

Indhold:

Student: Jacob Selbo
Studie nummer: 20233533

GitHub: https://github.com/jacobselbok/ai-og-data-mini-projekt

Herinde er alt kode der er brugt til at løse opgaverne + eventuelle billeder og filer.

Lektion 2: Databaser

1. Skriv et Python script der opretter en SQLite database 'school.db' (se https://realpython.com/python-sql-libraries/)
2. Udvid scriptet til at oprette to tabeller 'Students' og 'Courses'.
    1. Students skal indeholde student_id, name, og major
    2. Courses skal indeholde course_id, course_name, og instructor
3. Indsæt mindst 5 records i hver tabel.
4. Opret en tredje tabel 'Enrollments' der opretter relationer mellem Students og Course.
    1. Enrollments skal indeholde enrollment_id, student_id, og course_id
5. Lav forespørgsler (queries) der
    1. Vælger alle kurser som en specific studerende er tilmeldt
    2. Vælger alle studerende der er tilmeldt et specifikt kursus.

1. Opgave 1:

For at oprette en SQLite database, impoter jeg sqlite3. Herefter opretter jeg en connection object til en ”path” der kaldes school.db:

```
connection = sqlite3.connect(path)
```

2. Opgave 2:

Jeg executer ”CREATE TABEL IF NOT EXISTS” på cursor objectet.

```
cursor = connection.cursor()
try:
    cursor.execute(query)
```

Querierne laver 2 tabeller i school.db databasen, kaldet henholdsvis ”Students” og ”Courses”. Begge tabeller får deres egne id for at de nemmere kan samles senere.

```
student_db = "CREATE TABLE IF NOT EXISTS Students(student_id INTEGER, name TEXT NOT NULL, major TEXT NOT NULL)"
execute_query(connection, student_db)
courses_db = "CREATE TABLE IF NOT EXISTS Courses(course_id INTEGER, course_name TEXT NOT NULL, instructor TEXT NOT NULL)"
execute_query(connection, courses_db)
```

3. Opgave 3:

Jeg executer INSERT INTO på tabellen og dens 3 columns og angiver 5 unikke studenter. Det samme bliver gjort for Courses.

```
student_records = """
INSERT INTO
  Students (student_id, name, major)
VALUES
  (2, 'James', 'AI'),
  (5, 'Leila', 'Software'),
  (7, 'Brigitte', 'Medicin'),
  (11, 'Mike', 'IT'),
  (12, 'Elizabeth', 'Hardware');
"""
```

4. Opgave 4:

Jeg laver en ny tabel til Enrollments på samme måde som ved Students og Courses i opgave 2. Herefter tilføjer jeg 10 rækker til Enrollments hvor student_id og course_id kan relateres til hinanden.

```
enrollments_records = """
INSERT INTO
  Enrollments (enrollments_id, student_id, course_id)
VALUES
  (41, 2, 22),
  (42, 2, 27),
  (43, 5, 22),
  (44, 5, 34),
  (45, 5, 35),
  (46, 7, 22),
  (47, 7, 37),
  (48, 11, 22),
  (49, 12, 27),
  (50, 12, 35);
"""
```

5. Opgrave 5:

Her bruges .fetchall() funktionen til at returnere en liste af tuples der indeholder sammenhængen mellem de valgte records.

```
cursor.execute(query)
result = cursor.fetchall()
```

Jeg SELECTer først at jeg vil have student navnene og course navnene. Herefter JOINer jeg Students tabellen til Enrollments tabellen via student_id. Så JOINer jeg det resulterende tabel sammen med courses tabellen via course_id. Så bruger jeg WHERE til at specificere at jeg, i den resulterende tabel, kun vil have information fra den student med student_id = 5. Til sidst printer jeg de tupler som .fetchall() returnerede.

```
query = """
SELECT
  Students.name,
  Courses.course_name
FROM
  Students
JOIN
  Enrollments ON Students.student_id = Enrollments.student_id
JOIN
  Courses ON Enrollments.course_id = Courses.course_id
WHERE
  Students.student_id = 5;
"""
results = execute_read_query(connection, query)
for result in results:
    print(result)
```

Den samme fremgangsmåde blev brugt til at finde alle studenter der er i en specifik course. Herunder er de returnerede tupler:

```
('Leila', 'Data extraction')
('Leila', 'Data noise')
('Leila', 'Data mishaps')
('James', 'Data extraction')
('Leila', 'Data extraction')
('Brigitte', 'Data extraction')
('Mike', 'Data extraction')
```

Lektion 3: Dataoprensning og præprocessering

Opgave1:

- Skriv et Python script der tager et rent gråskala billede som input, og tilføjer salt-pepper støj.
- Efter tilføjelse af støj skal:
    - X % af pixels skal være støj (heraf halvdelen hvide pixels og den anden halvdel sorte pixels).
- Implementer middelværdisfiltrering med en kernel på 3x3, og anvend det på det støjfulde billede.
- Implementer median filtrering og anvend det på det støjfulde billede.

Opgave 2:

- Skriv et Python script der tager et rent gråskala billede som input, og tilføjer Gaussisk støj.
- Scriptet kan I antage at:
    - \mu = 0
    - \sigma er en brugerparameter.
- Implementer middelværdisfiltrering med en kernel på 3x3, og anvend det på det støjfulde billede.
- Implementer median filtrering og anvend det på det støjfulde billede.

Opgave 3:

- Følg følgende guide (delen der vedrører lineær PCA), og forstå hvordan PCA kan anvendes til støjreduktion.
- https://scikit-learn.org/stable/auto_examples/applications/plot_digits_denoising.html
- Eksperimentér med forskellige antal af komponenter I rekonstruktionen.

Opgave 1:

Jeg starter med at indlæse et billede med cv2 og ændre det til gray scale:

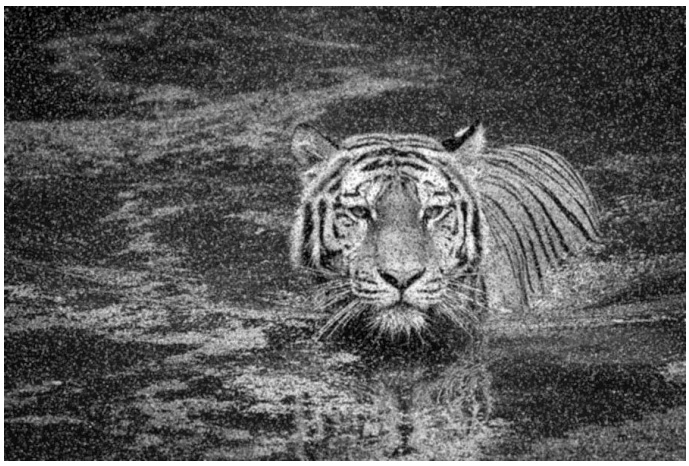
```
img_bgr = cv.imread(r"Lektion 3\image.png")
img_gray = cv.cvtColor(img_bgr, cv.COLOR_BGR2GRAY)
```

Herefter laver jeg en funktion der først finder det totale mænde af pixel. Så sætter den 10% tilfældigt udvalgte pixels, af alle pixels, og sætter dem til hvid og gør det samme med 10% til sort.

```
for i in range(number_of_pixels):
    y_coord=random.randint(0, row - 1)
    x_coord=random.randint(0, col - 1)
    img[y_coord][x_coord] = 255
```

Så tilføjer jeg middelværdisfiltrering med .blur funktion og median filtrering med .medianblue funktionen:

```
kernel_size = (3, 3)
img_filtered = cv.blur(img_salt_pepper, kernel_size)
img_median_filtered = cv.medianBlur(img_salt_pepper, 3)
```



Opgave 2:

Indlæsning af billedet skete på samme måde som opgave 1.

Herefter blev der lavet en ny funktion hvor der først blev udregnet gaussian støj for alle pixels og så blev det lagt til gray scale billedet:

```
gaussian_noise = np.random.normal(mean, sigma, (row, col)).reshape(row, col)
noisy_img = img + gaussian_noise
```

Herefter blev np.clip brugt til at sørge for at alle tal er imellem 0 og 255:

```
noisy_img = np.clip(noisy_img, 0, 255)
```

Til sidst blev dataframe tretuneret med .astype(np.uint8), hvilket afrunder eventuelle floats og ændre tallene til integers:

```
return noisy_img.astype(np.uint8)
```

Her kan sigma ændres som brugerparameter:

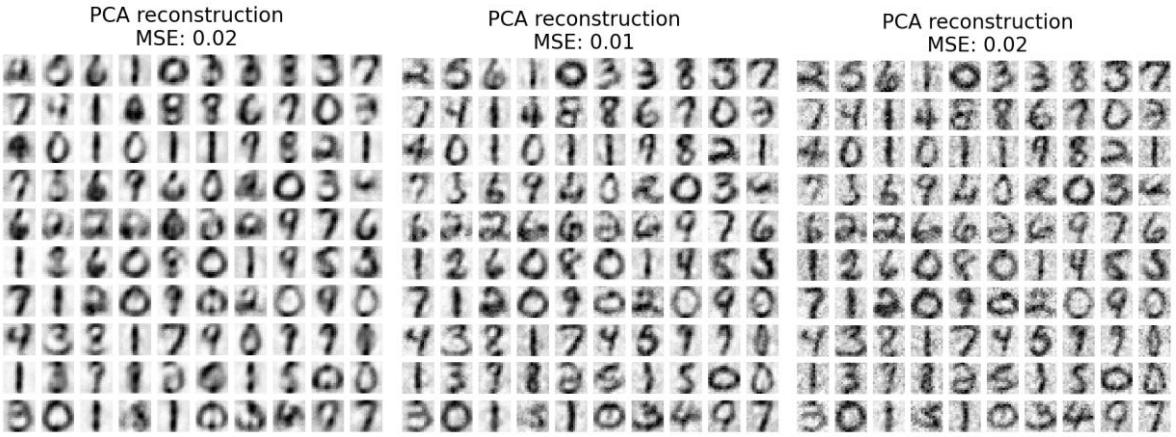
```
sigma_value = 25
```

Middelværdisfiltrering og median filtrering blev implementeret på samme måde som opgave 1. Billederne kan ses i github.

Opgave 3:

For de nedenstående billeder har n_components stået på henholdsvis 10, 32 og 80:

```
pca = PCA(n_components=80, random_state=42)
pca.fit(X_train_noisy)
```


PCA reconstruction
MSE: 0.02
PCA reconstruction
MSE: 0.01
PCA reconstruction
MSE: 0.02

Lektion 5: Intro til Pandas og Datamanipulering

1) Følg tutorial (under 'Forberedelse') grundigt på egen hånd. Opret en Google Colab notebook (https://colab.research.google.com/), og implementer og afprøv alle eksempler på egen hånd.
2) Anvend metoderne på datasættet fra kursusgang 4:
   - indlæs datasættet i en DataFrame
   - fjern manglende værdier
   - plot numeriske variabler i datasættet (https://python.pages.doc.ic.ac.uk/lessons/pandas/06-methods/05-plot.html)

1. Opgave 1:

Jeg lavede en fil kaldet ”pandas_intro.py” i lektion 5 mappen hvor jeg implementerede alle funktionerne jeg testede fra Pandas tutorialen.

2. Opgave 2:

Jeg indlæser datasættet i Pandas med følgende funktion:

```
df = pd.read_csv(r"Lektion 5\recipeData.csv")
```

Jeg printer .shape på dataframe før og efter jeg bruger .dropna funktionen:

```
print(df.shape)
df.dropna(inplace=True)
print(df.shape)
```

```
(73861, 23)
(757, 23)
```

Jeg bruger .info til at finde hvilke columns der har numeriske værdier:

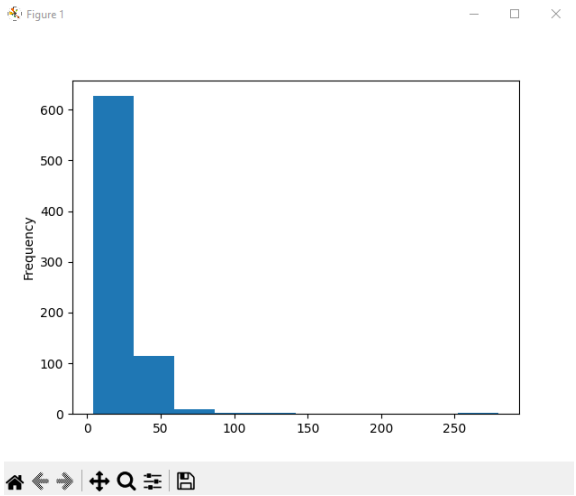
```
print(df.info())
```

Herefter plotter jeg for ”Size(L)” og ”Color”:

```
print(df.info())

df["Size(L)"].plot.hist()
plt.show()

df["Color"].plot.hist()
plt.show()
```



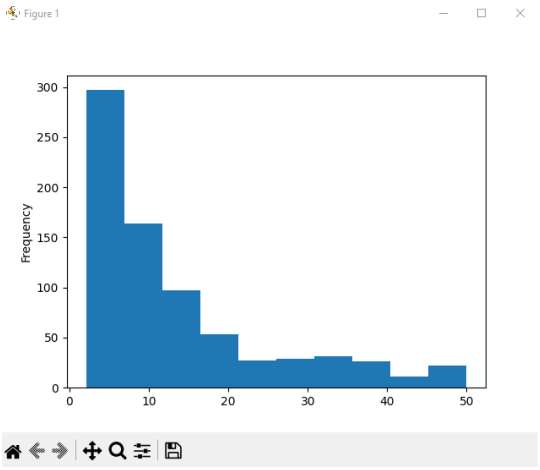


Figure 1
Frequency
300
250
200
150
100
50
0
0
10
20
30
40
50

Lektion 7: Interaktiv datavisualisering

- Følg tutorial (https://youtu.be/yVxnQQpBg-Q?si=mAvevB6T-yvkd9X0)* og lav jeres eget scatter plot (kode)
- Følg tutorials og lave jeres eget interaktive scatter plot
  i) tutorial 1 (https://youtu.be/j4VI47GlmGA?si=C9vZ3vwjEVear8ap)* (kode)
  ii) tutorial 2 (https://youtu.be/kyWZf3sNhtg?si=3ocH-ApWBpnVYZZ5)* (kode)
- Opdatér jeres kode til at bruge et valgfrit datasæt med features (f.eks. fra jeres projekt, eller AutoMPG)

Opgave 1:

Der startes med at indlæse csv med pandas:

```
df = pd.read_csv(r"Lektion 7\iris.csv")
```

Herefter bruges .info() til at få informationer om csv filen:

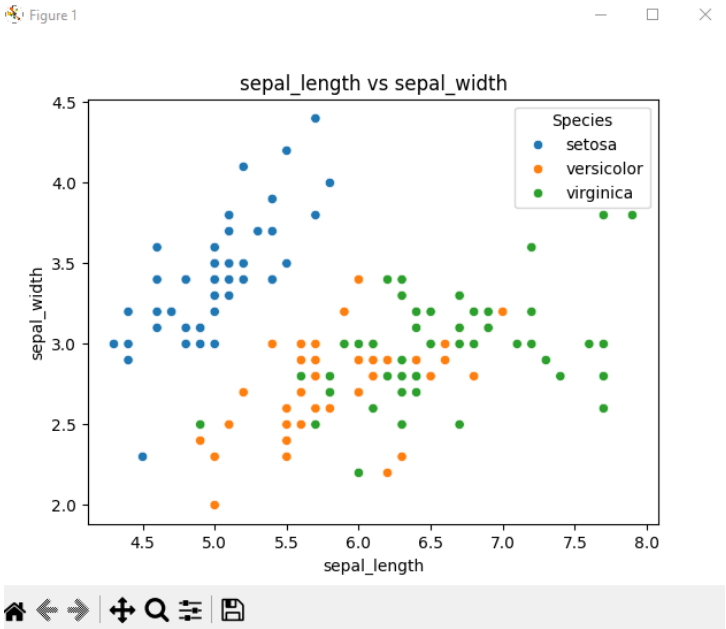
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Der laves et scatterplot udfra ”sepal_length” og ”sepal_width” med target på ”species” ved hjælp af seaborn og matplotlib.pyplot:

```
scatterplot(df, "sepal_length", "sepal_width", "species")
```

```
def scatterplot(df, x_plot, y_plot, target):
    sns.scatterplot(data=df, x=x_plot, y=y_plot, hue=target)
    plt.xlabel(x_plot)
    plt.ylabel(y_plot)
    plt.title(f"{x_plot} vs {y_plot}")
    plt.legend(title="Species")
    plt.show()
```

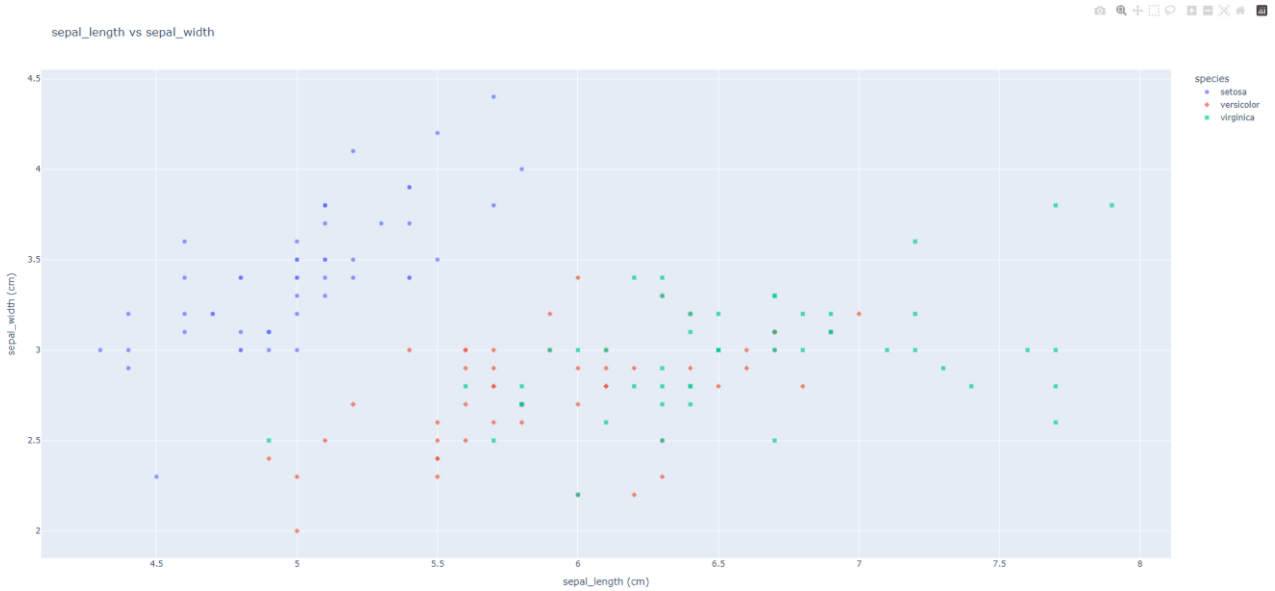


Opgave 2:

Der laves et interaktiv scatterplot, med hjælp af plotly.express:

```
int_scatterplot(df, "sepal_length", "sepal_width", "(cm)", "(cm)", "species")
```

```
def int_scatterplot(df, x_plot, y_plot, x_value, y_value, target):
    fig = px.scatter(df, x=x_plot, y=y_plot, color=target,
                     title=f"{x_plot} vs {y_plot}",
                     labels={x_plot: f"{x_plot} {x_value}", y_plot: f"{y_plot} {y_value}"},
                     hover_data={target: True},
                     opacity=0.7,
                     symbol=target)
    fig.show()
```



På dette scatterplot kan der zoomes ind og ud, man kan ”pan”, plot kan downloades som png osv, hvilket gør det til et interaktivt scatterplot.