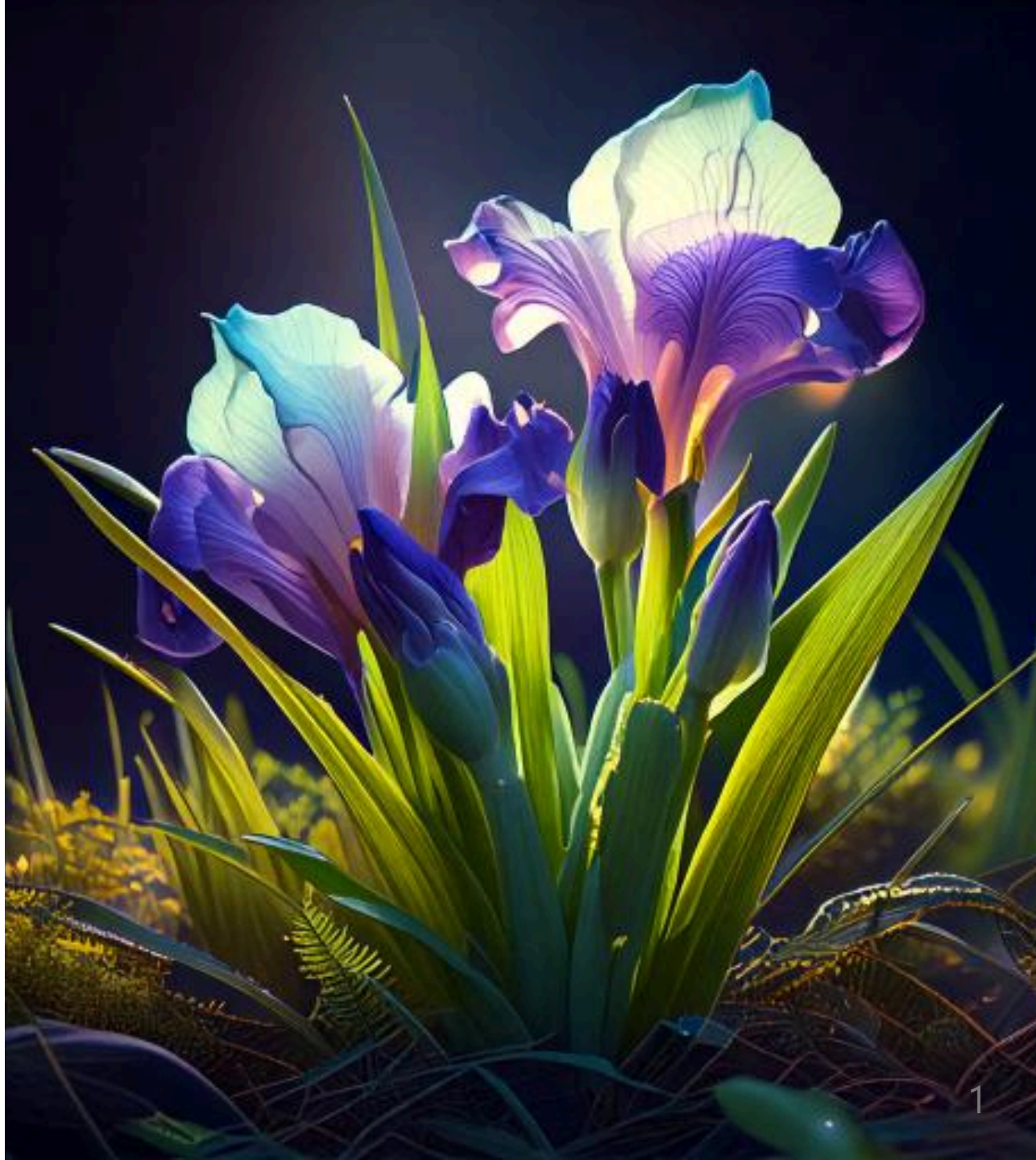
Trends in AI

Hoofdstuk 5

Lab deel 2: leren uit data

Hassan Haddouchi

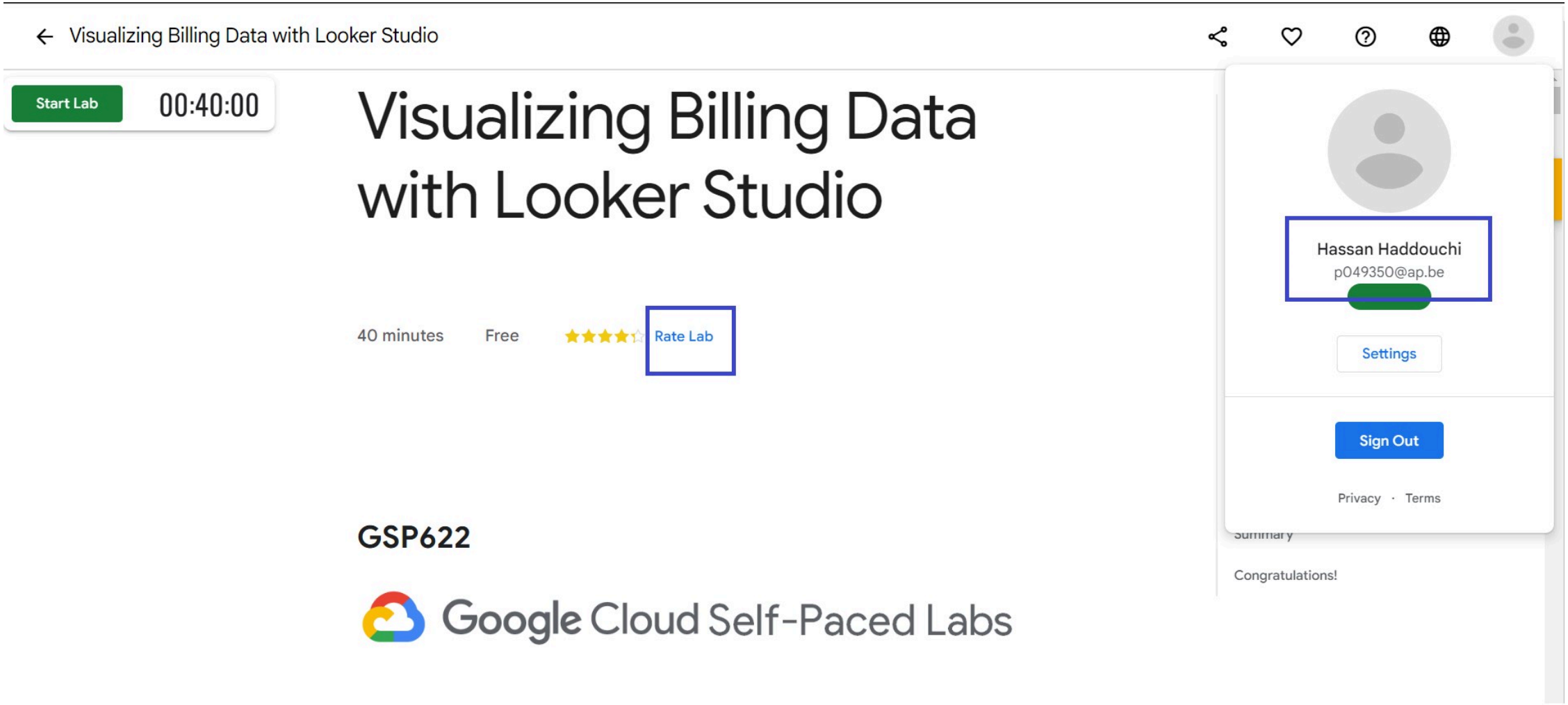


Dit tweede deel bestaat zelf uit 2 delen:

1. Een opwarmer met BigQuery ML
2. Een hands-on oefening waar je zelf code schrijft om door middel van een ML model uit data te leren.

Deel 1

Voer de stappen uit in dit BigQuery ML Lab en neem een screenshot zoals op de volgende slide, dat je straks zal indienen. Begin daarna aan het tweede deel.


Visualizing Billing Data with Looker Studio
Start Lab
00:40:00
Visualizing Billing Data with Looker Studio
40 minutes
Free
Rate Lab
GSP622
Google Cloud Self-Paced Labs
Hassan Haddouchi
p049350@ap.be
Settings
Sign Out
Privacy · Terms
Summary
Congratulations!

Deel 2

Inleiding

Stel: je werkt aan een project voor een online winkel en wilt voorspellen of een klant tevreden zal zijn met een aankoop. Op basis van historische gegevens over klanten kun je leren welke factoren (zoals bezorgtijd, prijsklasse, en producttype) bijdragen aan tevredenheid.

Ik zal hiervoor de dataset *Online Retail II* gebruiken.

In de volgende stappen zal ik tonen hoe ik dit doel kan bereiken. Uw taak is om hetzelde te doen, weliswaar in een andere context (zie slides verderop).

Op basis van historische verkoopgegevens en kenmerken zoals aantal aankopen, productprijs, en land van de klant, wil ik voorspellen of een klant tevreden zal zijn met een aankoop.

Ik zal hiervoor data moeten verkennen, verwerken en een eenvoudig model trainen dat voorspelt of een klant tevreden is (tevreden = 1) of niet (tevreden = 0).

Zoals eerder aangegeven, maakt mijn opdracht gebruik van de Online Retail II Dataset, die gegevens bevat over aankopen van een online winkel.

Kolommen die we gebruiken:

- CustomerID: unieke klant-ID.
- Quantity: aantal verkochte eenheden.
- UnitPrice: prijs per eenheid van het product.
- Country: land van de klant.
- tevreden: toegevoegde kolom (1 = tevreden, 0 = ontevreden), - gebaseerd op eenvoudige regels:
    - Als Quantity > 10 en UnitPrice < 50, dan tevreden (1).
    - Anders ontevreden (0).

Dit zijn de stappen die ik zal uitvoeren

Stap 1: dataset laden en eerste rijen of samenvatting verkennen

Stap 2: voorbewerking

- De relevante kolommen (CustomerID, Quantity, UnitPrice, Country) selecteren.
- Een nieuwe kolom tevreden toevoegen.
- Controleren op ontbrekende waarden en deze aanvullen of de rijen verwijderen.

Stap 3: data splitsen

- De dataset splitsen in een trainingsset (70%) en een testset (30%) met *train_test_split*.

Stap 4: model bouwen

- Een DecisionTreeClassifier trainen met de trainingsset.
- De testset gebruiken om de prestaties van het model te evalueren.

Stap 5: resultaten analyseren

- De nauwkeurigheid van het model berekenen.
- Een diagram tonen om het belang van kenmerken te visualiseren (feature importance).

1. Dataset laden en verkennen

```
import pandas as pd

# Dataset laden
df = pd.read_excel("Online Retail.xlsx")

# Relevante kolommen selecteren en missing values verwijderen
df = df[["CustomerID", "Quantity", "UnitPrice", "Country"]].dropna()

# Tevredenheidskolom toevoegen
df["tevreden"] = (df["Quantity"] > 10) & (df["UnitPrice"] < 50)
df["tevreden"] = df["tevreden"].astype(int)  # Converteer naar 0 of 1

# Bekijk de eerste rijen
print(df.head())
```

2. Data splitsen

```
from sklearn.model_selection import train_test_split

# Features en labels definiëren
X = df[["Quantity", "UnitPrice"]]
y = df["tevreden"]

# Data splitsen
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Controleer de splitsing
print("Trainingsset:\n", X_train.head())
print("\nTestset:\n", X_test.head())
```

3. Model bouwen

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Model trainen
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Voorspellingen maken
y_pred = model.predict(X_test)

# Nauwkeurigheid berekenen
accuracy = accuracy_score(y_test, y_pred)
print(f"Nauwkeurigheid: {accuracy:.2f}")
```

4. Analyseer de resultaten

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

# Kenmerkbelang visualiseren
feature_importances = model.feature_importances_
features = X.columns

plt.barh(features, feature_importances, color="skyblue")
plt.xlabel("Belang van kenmerken")
plt.title("Kenmerkbelang (Feature Importance)")
plt.show()
```

Nu is het aan jou!

Context:

Je werkt aan een botanische applicatie om bloemen te classificeren op basis van hun kelkblad- en kroonbladmaten. Het doel is om de soort bloem (setosa, versicolor, of virginica) te voorspellen op basis van vier meetwaarden.

- Dataset analyseren en transformeren.
- Een SVM-model trainen om bloemsoorten te classificeren.
- Het model evalueren en de prestaties evalueren.

Gebruik de **Iris Dataset** voor deze opdracht. Deze dataset is direct beschikbaar in *sklearn.datasets* via *load_iris*.

De Iris Dataset bevat meetwaarden van kelk- en kroonbladen voor drie soorten bloemen:

Soorten:

- setosa, versicolor en virginica

Kenmerken:

- Sepal length (kelkbladlengte)
- Sepal width (kelkbladbreedte)
- Petal length (kroonbladlengte)
- Petal width (kroonbladbreedte)

Stappen

Stap 1: dataset laden en verkennen (inclusief een correlatiematrix om de relaties te beschrijven)
Stap 2: Feature scaling (selecteer kenmerken op basis van hun correlatie met elkaar en met de doelvariabele)
Stap 3: Model trainen (SVC uit sklearn)
Stap 4: Resultaten evalueren
Stap 5: Visualiseren (scatterplot, SVM-beslissingsgrenzen)

Belangrijk:

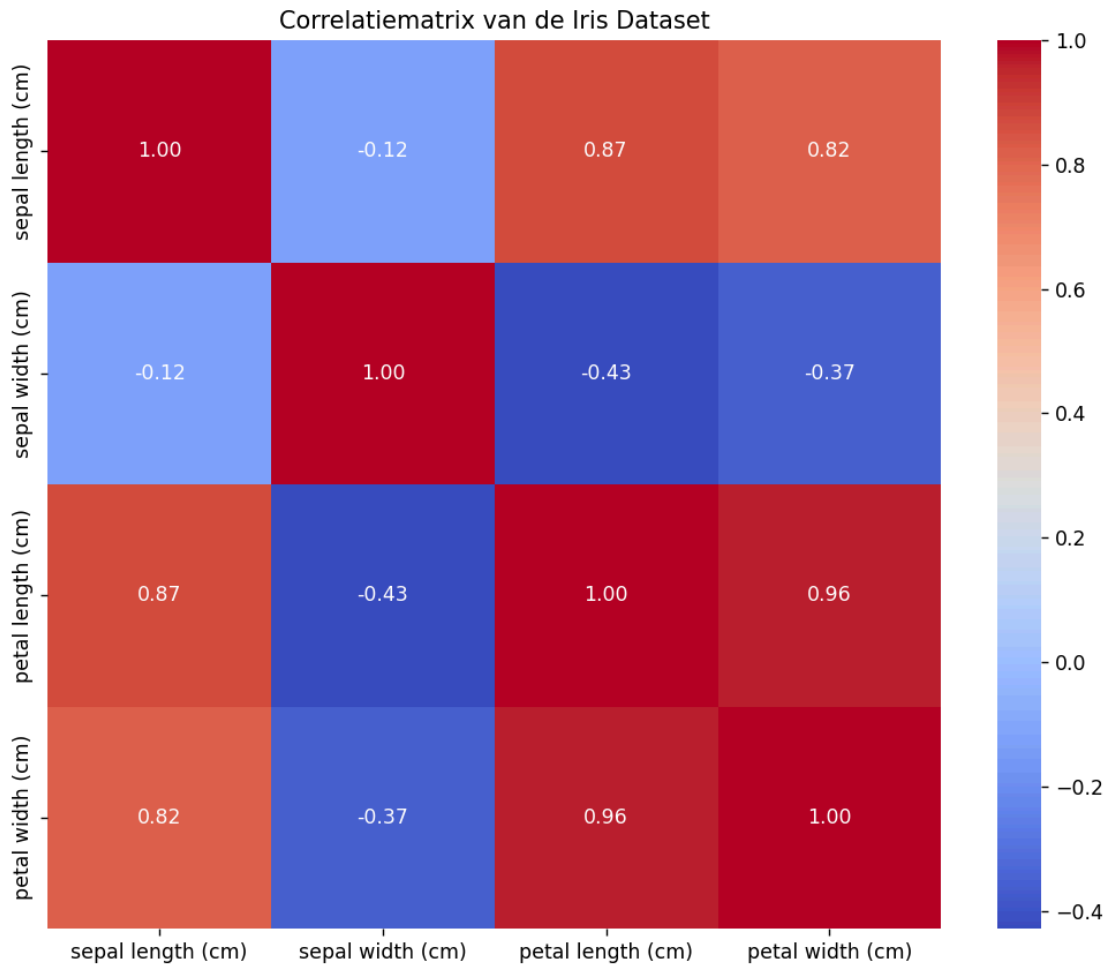
Zoals getoond en uitgelegd tijdens het labo is het van essentieel belang om bovenstaande stappen zorgvuldig uit te voeren. Ook verwacht ik dat uw code minstens een correlatiematrix, een scatterplot en een SVM plot toont met beslissingsgrenzen. Daarnaast moet je code ook een voorspelling produceren.

Om de modelprestatie te evalueren maak je gebruik van precision, recall, en F1-score. Druk deze waarden af in de console.

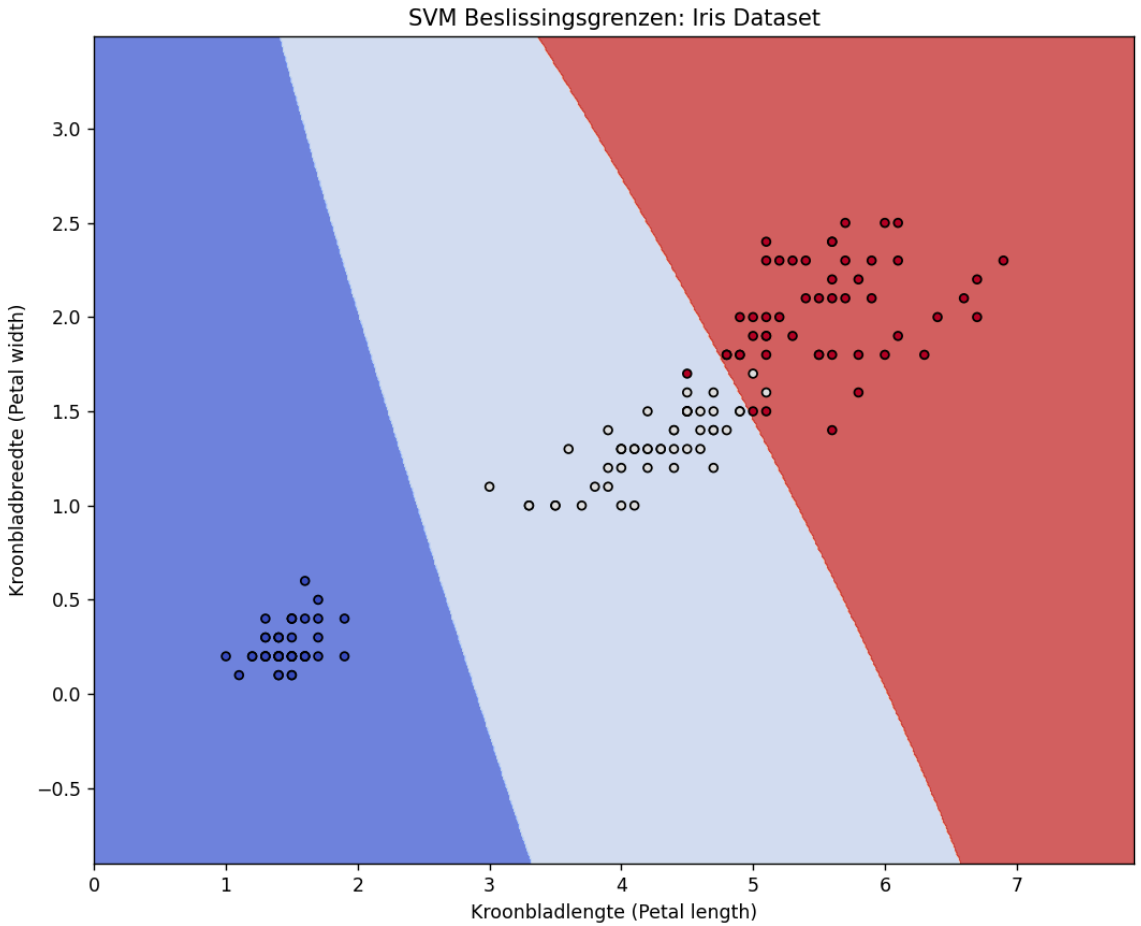
De macro average berekent de gemiddelde score per klasse en neemt daarna het rekenkundige gemiddelde van deze scores, zonder rekening te houden met het aantal voorbeelden per klasse.

De weighted average weegt de precision, recall, of F1-score van elke klasse op basis van het aantal voorbeelden in die klasse. Klassen met meer voorbeelden dragen meer bij aan de uiteindelijke score.

Mijn correlatiematrix



Mijn SVM Beslissingsgrenzen



Wat dien je in te leveren?

- Deel 1: een screenshot (voorbeeld hierboven)
- Deel 2: een werkend Python-script alsook je dataset, in dezelfde folder.

Steek alles in een .zip-folder.

Beantwoord de volgende vragen in je portfolio:

- Welke kenmerken zijn het meest informatief voor de classificatie van bloemen?
- Hoe kan de correlatiematrix helpen bij het vereenvoudigen of verbeteren van het model?