Experiment 1: Devise a program to import, load, and view a dataset.

RA2311003010359

M.SUNAYANA

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

# Read uploaded file data
= []

with open("dataset1.txt", "r") as f:
    for line in f:
        parts = line.strip().split()
if len(parts) >= 25:
            try:
                weight = float(parts[22])  # 23rd column
height = float(parts[23])  # 24th column
gender = int(parts[24])    # 25th column
data.append((height, weight, gender))            except
ValueError:
                continue

df = pd.DataFrame(data, columns=["Height", "Weight", "Gender"])
```

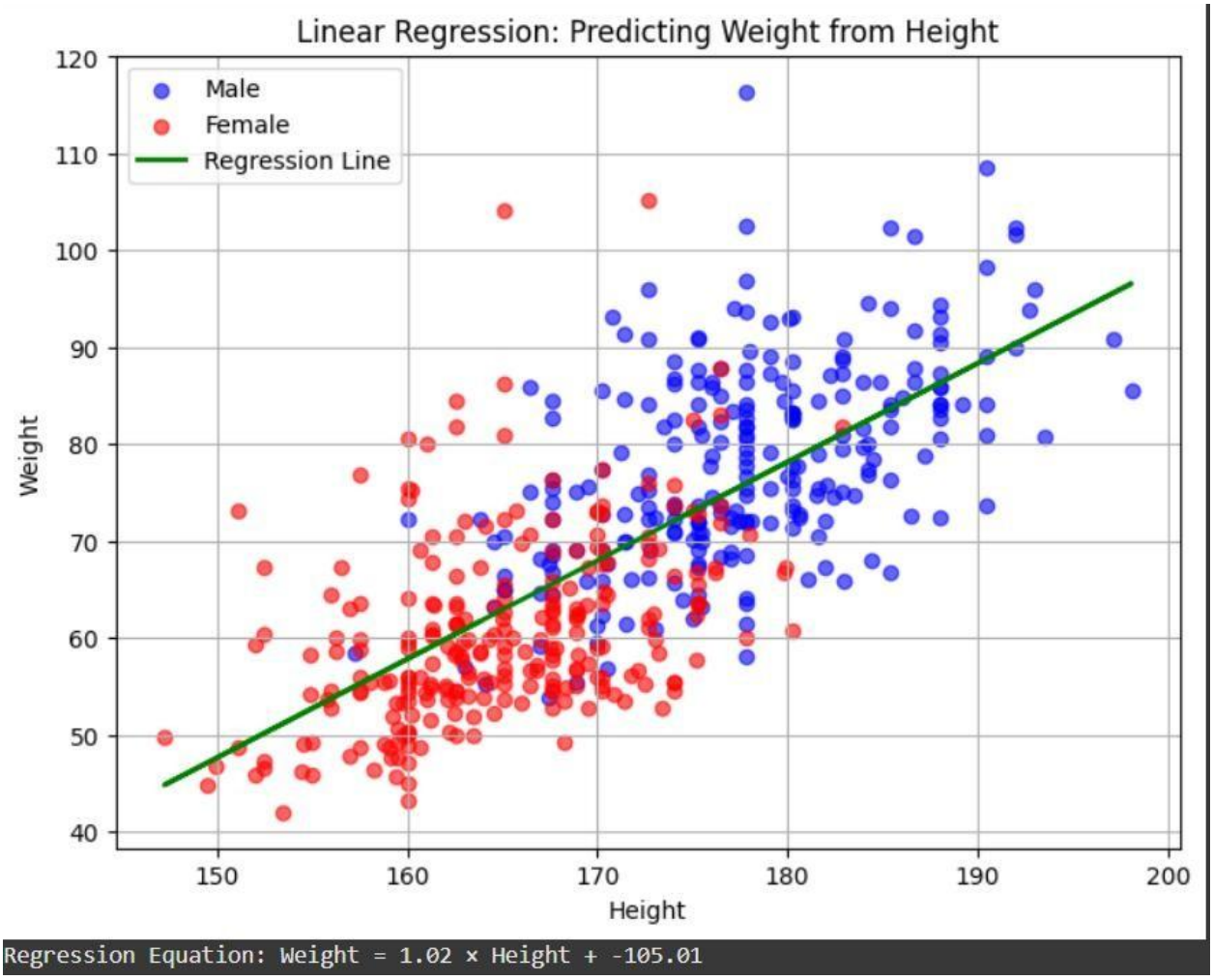
```
# Scatter plot by gender
plt.figure(figsize=(8, 6))
plt.scatter(df[df.Gender == 1].Height,
df[df.Gender == 1].Weight, color='blue',
label='Male', alpha=0.6)
plt.scatter(df[df.Gender == 0].Height, df[df.Gender == 0].Weight, color='red', label='Female',
alpha=0.6)

# Fit least-squares regression line X
= df["Height"].values.reshape(-1, 1)
y = df["Weight"].values

model =
LinearRegression()
model.fit(X, y) y_pred =
model.predict(X)

plt.plot(X, y_pred, color='green', label='Regression Line',
linewidth=2) plt.xlabel("Height") plt.ylabel("Weight")
plt.title("Linear Regression: Predicting Weight from Height")
plt.legend() plt.grid(True) plt.show()

# Print regression equation slope = model.coef_[0] intercept =
model.intercept_ print(f"Regression Equation: Weight = {slope:.2f} ×
Height + {intercept:.2f}")
```




```
Regression Equation: Weight = 1.02 × Height + -105.01
```

Experiment 2: Create a program to display the summary and statistics of the dataset.

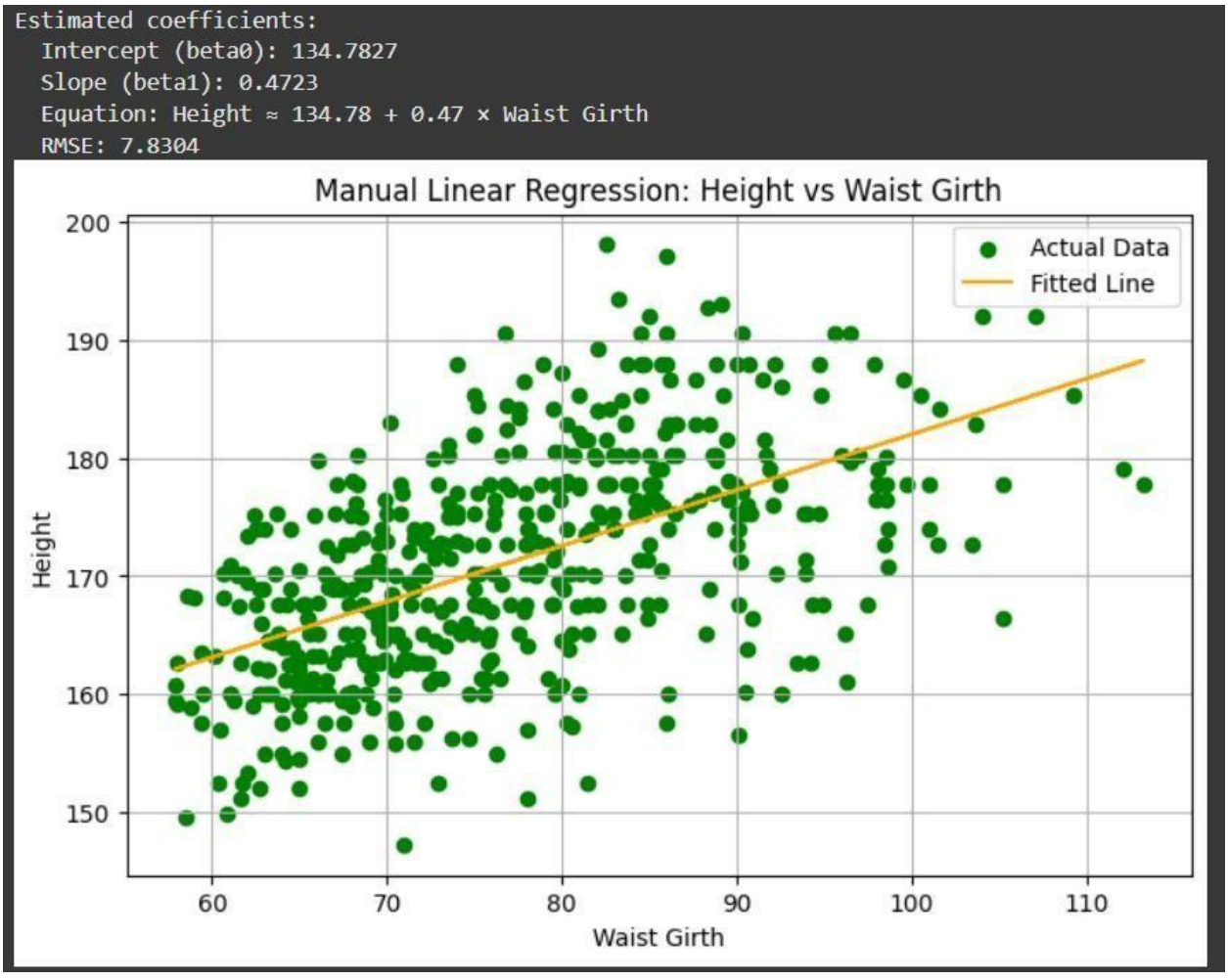
RA2311003010359

M.SUNAYANA

numerator = np.sum((x - x_mean) * (y - y_mean)) denominator
= np.sum((x - x_mean)**2)

beta_1 = numerator / denominator
beta_0 = y_mean - beta_1 * x_mean

```
Estimated coefficients:
  Intercept (beta0): 134.7827
  Slope (beta1): 0.4723
  Equation: Height ≈ 134.78 + 0.47 x Waist Girth
  RMSE: 7.8304
```



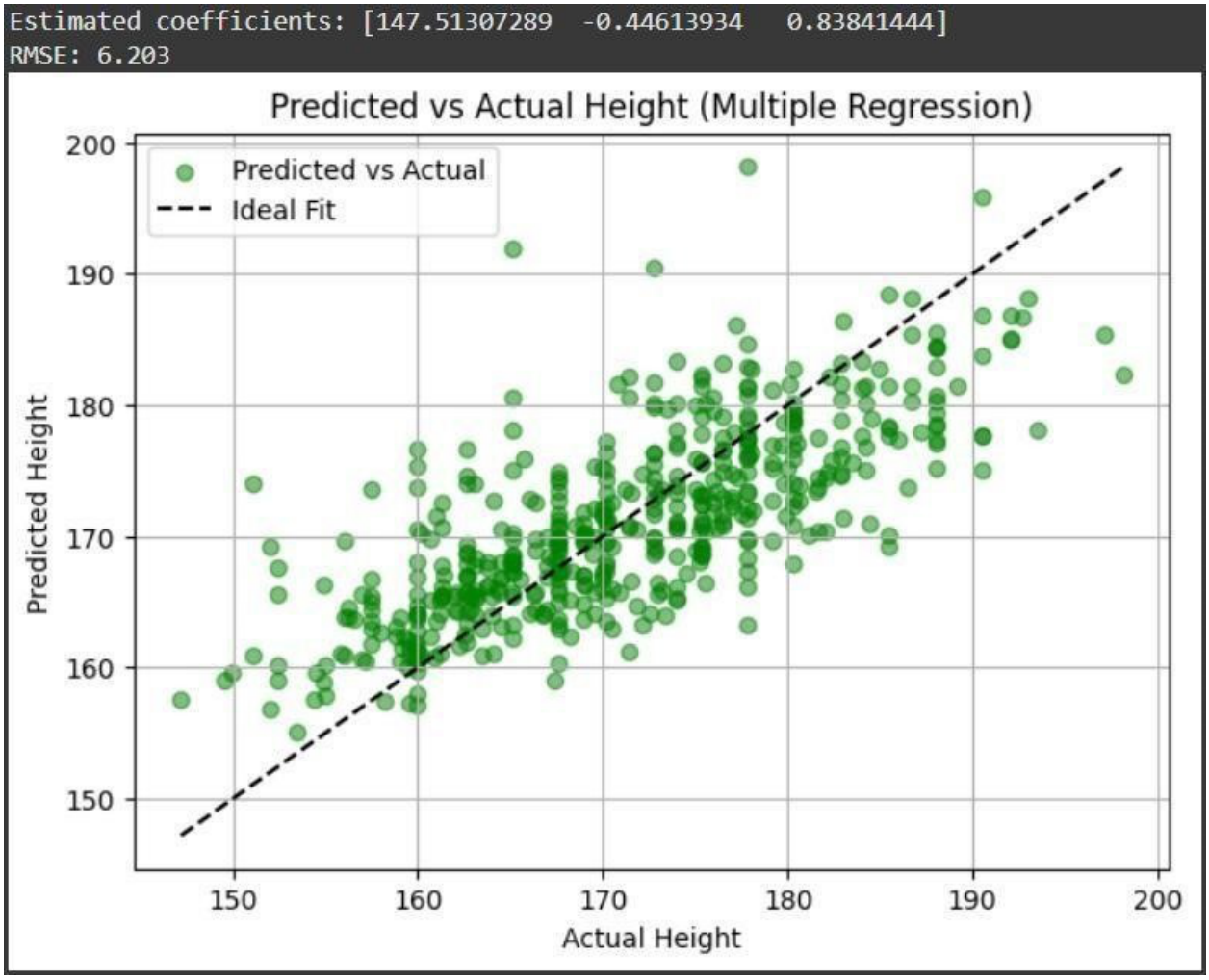
Experiment 3: Implement linear regression to perform prediction.

RA2311003010379

Gopika Dasari

X_transpose = X.T beta = np.linalg.inv(X_transpose @ X)

@ X_transpose @ y

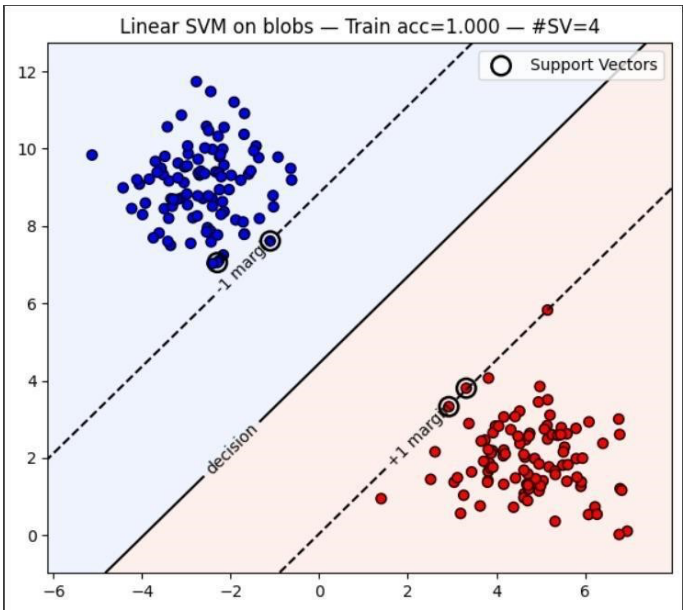


GOPIKA RATNAM DASARI

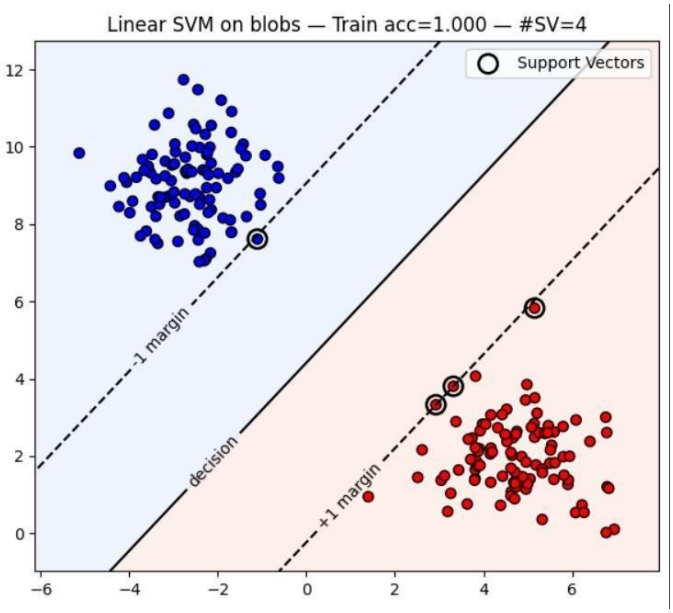
RA2311003010379

EXPERIMENT – 4

Exercise 1: Change regularization parameter C



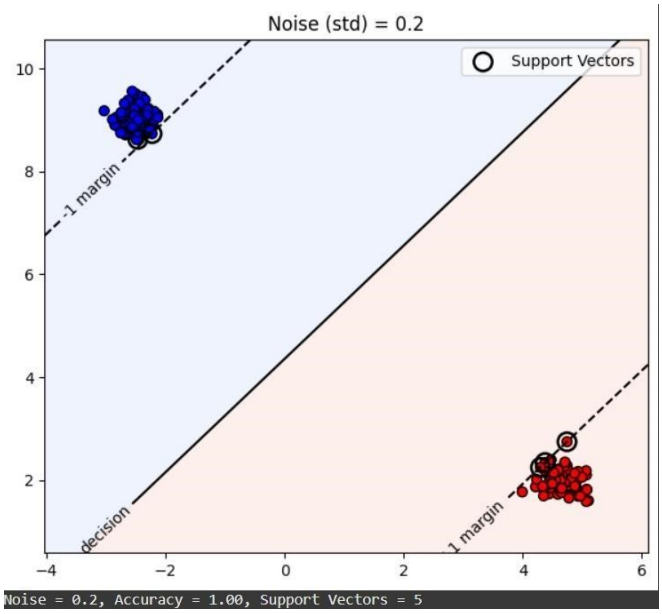
- When C = 0.1:
- The margin width becomes wider.
- The model allows some misclassifications.
- The number of support vectors increases.



When C = 10:

- The margin width becomes narrower.
- The model tries to classify all training points correctly.
- The number of support vectors decreases.

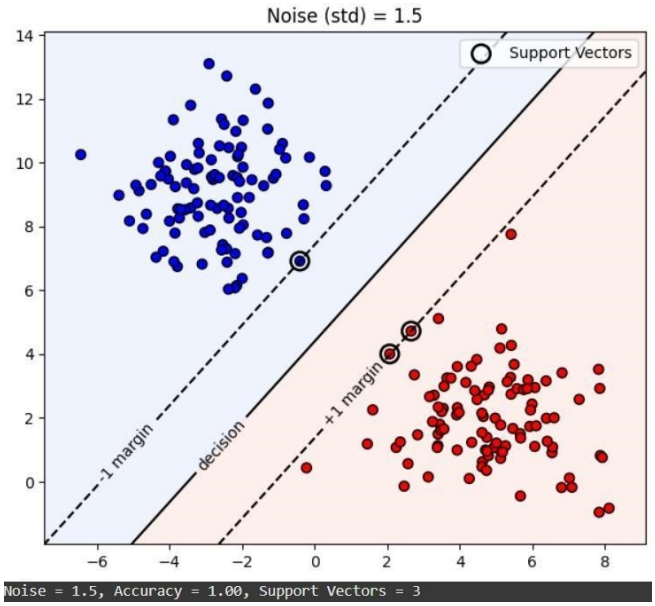
Exercise 2: Change dataset noise



cluster_std = 0.2 (almost perfectly separable) o

The classes are very cleanly separated. o The decision boundary is very clear and stable. o Accuracy is ~100%.

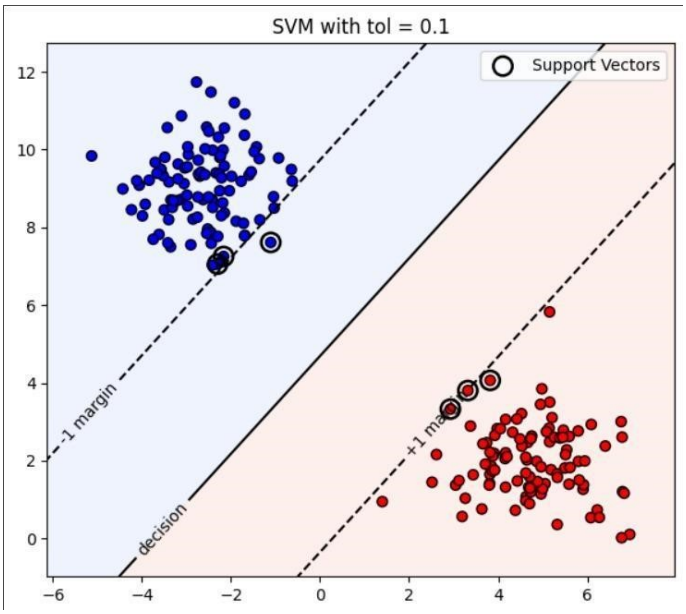
o Few support vectors are needed (only the closest points to the margin).



• cluster_std = 1.5 (more overlap) o The classes overlap a lot, so the decision boundary shifts to balance misclassifications. o Accuracy drops (not all points can be classified correctly).

o The number of support vectors increases, since more points lie close to or inside the margin.

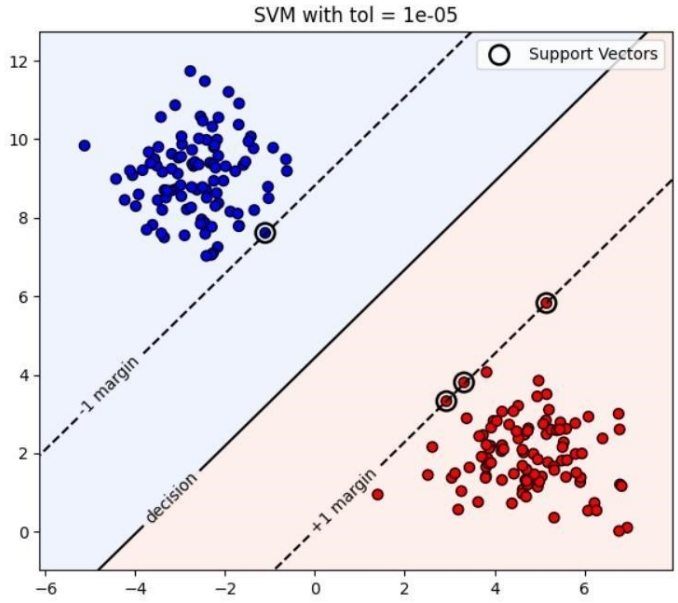
Exercise 3: Tolerance tol



tol = 1e-1 (looser stopping condition) o The optimizer stops earlier since

it accepts a larger error tolerance. o Training is faster (fewer iterations).

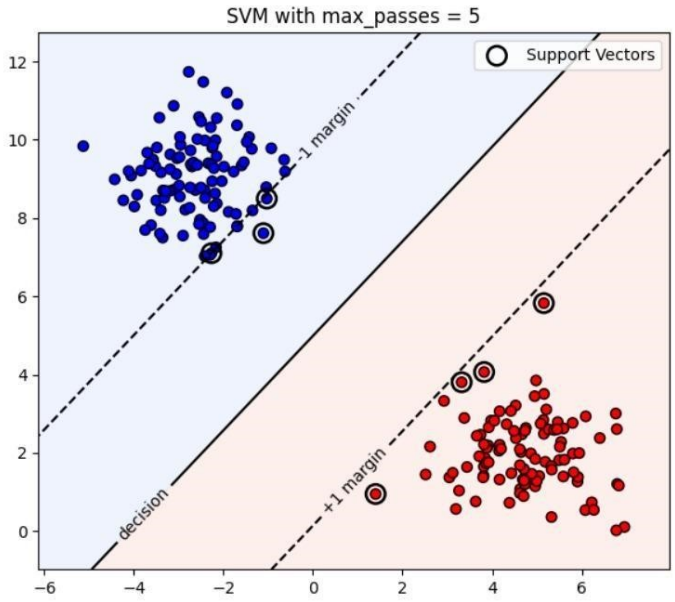
o The results change only slightly, accuracy is almost the same but the model may not be perfectly optimized.



tol = 1e-5 (tighter stopping condition)

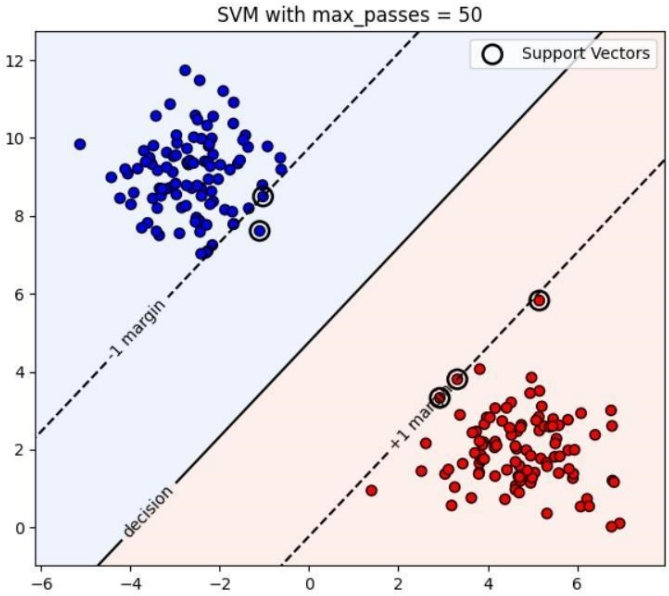
- The optimizer requires a very small error before stopping.
- Training takes longer (more iterations).
- Results (accuracy, boundary, support vectors) change only slightly compared to 1e-3.

Exercise 4: Maximum passes max_passes



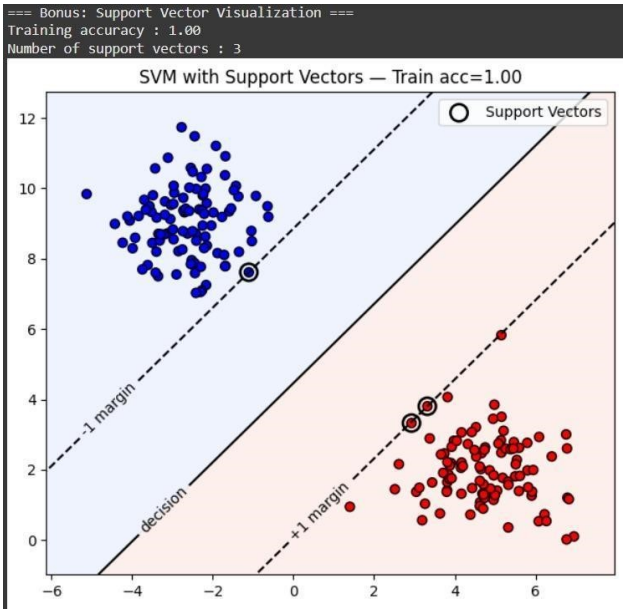
max_passes = 5 (fewer passes)

- Training may stop too early before the SMO fully converges.
- Accuracy can drop slightly because not all support vectors are properly optimized.
- Decision boundary may be less stable.



- max_passes = 50 (more passes) o Allows more iterations, ensuring convergence. o Accuracy may improve slightly if needed, but for simple linearly separable data, it usually remains the same. o Computational time increases slightly.

Exercise 5 (bonus): Visualize support vectorsx



Do support vectors always lie on the margin lines ($f(x) = \pm 1$)?

- Yes, ideally.

- Support vectors are the data points closest to the decision boundary, lying exactly on or very near the margin lines.
- Are they the critical points holding up the decision boundary?
- Yes.
- The SVM's decision boundary is determined only by the support vectors.
- Points farther away do not affect the position of the hyperplane.

3. Reflection Questions

1.Why does a smaller C allow more violations but produce a wider margin?

C is the regularization parameter in SVM.

Smaller C means we care less about misclassifying points and more about having a wide margin.

So the optimizer allows some points to violate the margin or even be misclassified to achieve a larger margin.

Larger C puts more emphasis on minimizing misclassification, leading to a narrower margin to correctly classify all points.

Intuition:

Small C → "I don't mind a few mistakes if the margin is wide."

Large C → "Every point must be classified correctly, even if it makes the margin tight."

2. Why do support vectors alone determine the boundary?

Support vectors are the points that lie on or inside the margin.

Points far from the margin have $\alpha = 0$ in the dual formulation, so they don't contribute to the decision function: $f(x)=i\sum\alpha iyi\langle xi,x\rangle+b$

Only the non-zero α's (the support vectors) affect w and b.

Intuition:

The boundary is "held up" by the points closest to it; the others are irrelevant.

3. If you increase dataset noise, why does accuracy drop but support vector count increase?

More noise → points overlap between classes.

The SVM cannot separate them perfectly, so some points are forced onto or inside the margin.

More points with $\alpha > 0$ → more support vectors.

Misclassified or borderline points → training accuracy drops.

Intuition:

Noisy data → decision boundary is "pulled" in different directions → more critical points (support vectors) and lower accuracy.