Support Vector Machines (SVM) – Vectors & Plotting Data

SVM uses **vectors** (feature values) to represent data points on a graph and finds the best decision boundary (**hyperplane**) that separates classes with the **maximum margin.**

Step 1: Understanding Vectors

A vector is like an arrow representing a data point with numbers for its features.

Apple → [100, 5]
(Weight = 100g, Size = 5cm)

Orange → [150, 7]
(Weight = 150g, Size = 7cm)

Each number in the vector is a feature.

Why it matters?

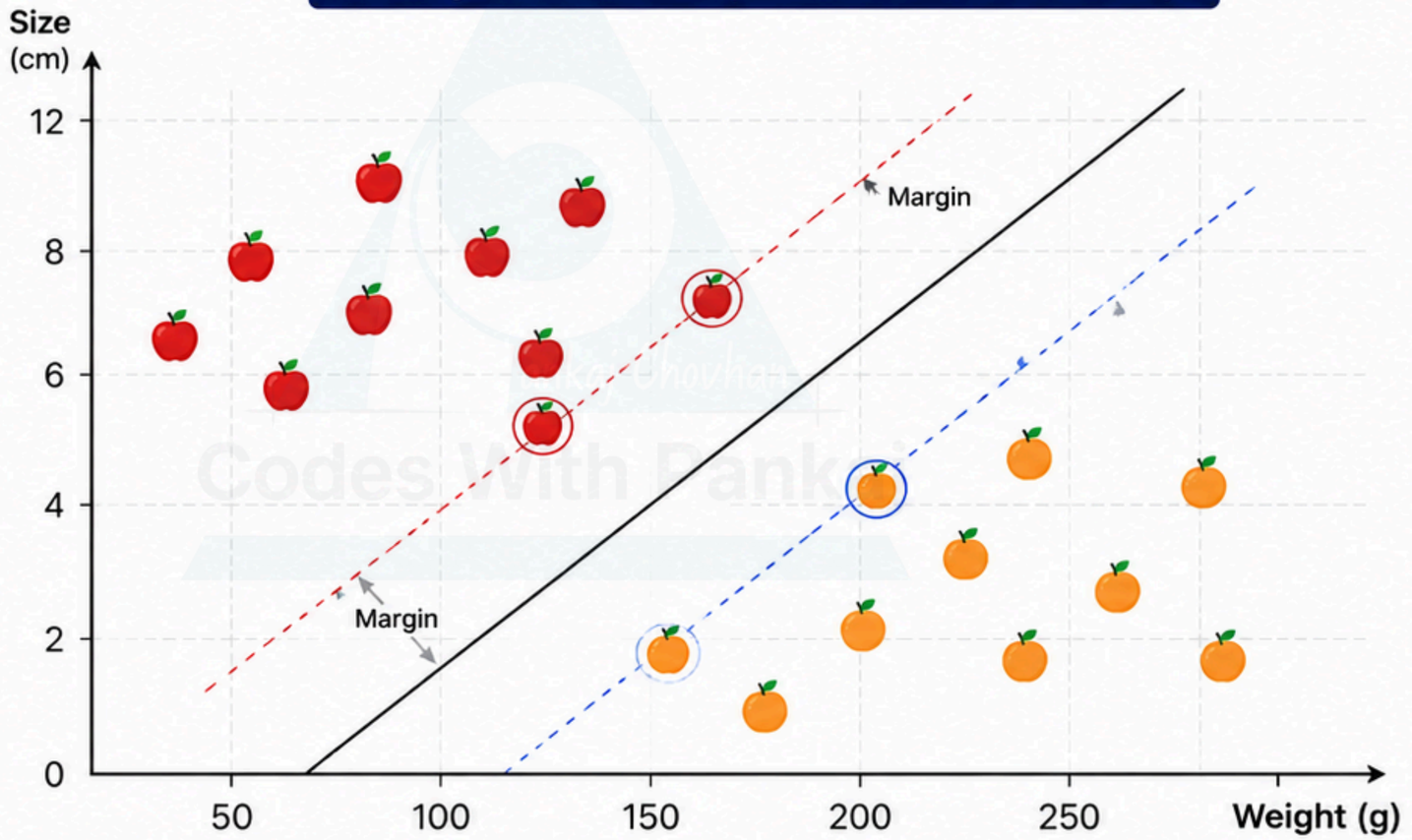
SVM uses these vectors to plot data points on a graph and find **patterns** to separate classes.

Visual Idea

We plot each fruit as a dot using its vector values on the graph.

x-axis → Weight (g)
y-axis → Size (cm)

Plotting Vectors on a Graph & SVM Decision Boundary



Legend

- Apple (Class 1)
- Orange (Class 0)
- Decision Boundary (Hyperplane)
- Margin
- Support Vectors (Apple)
- Support Vectors (Orange)

Key Takeaway

SVM chooses the boundary that maximizes the margin between the closest points (**support vectors**) of the two classes.

Support Vector Machines (SVM) Step 2: Decision Boundary

A **decision boundary** is the line (or surface) that separates different classes.

1. What is a Decision Boundary?

A decision boundary is a line (or surface) that separates different groups, like apples from oranges.

2. Example: Fruit Classification

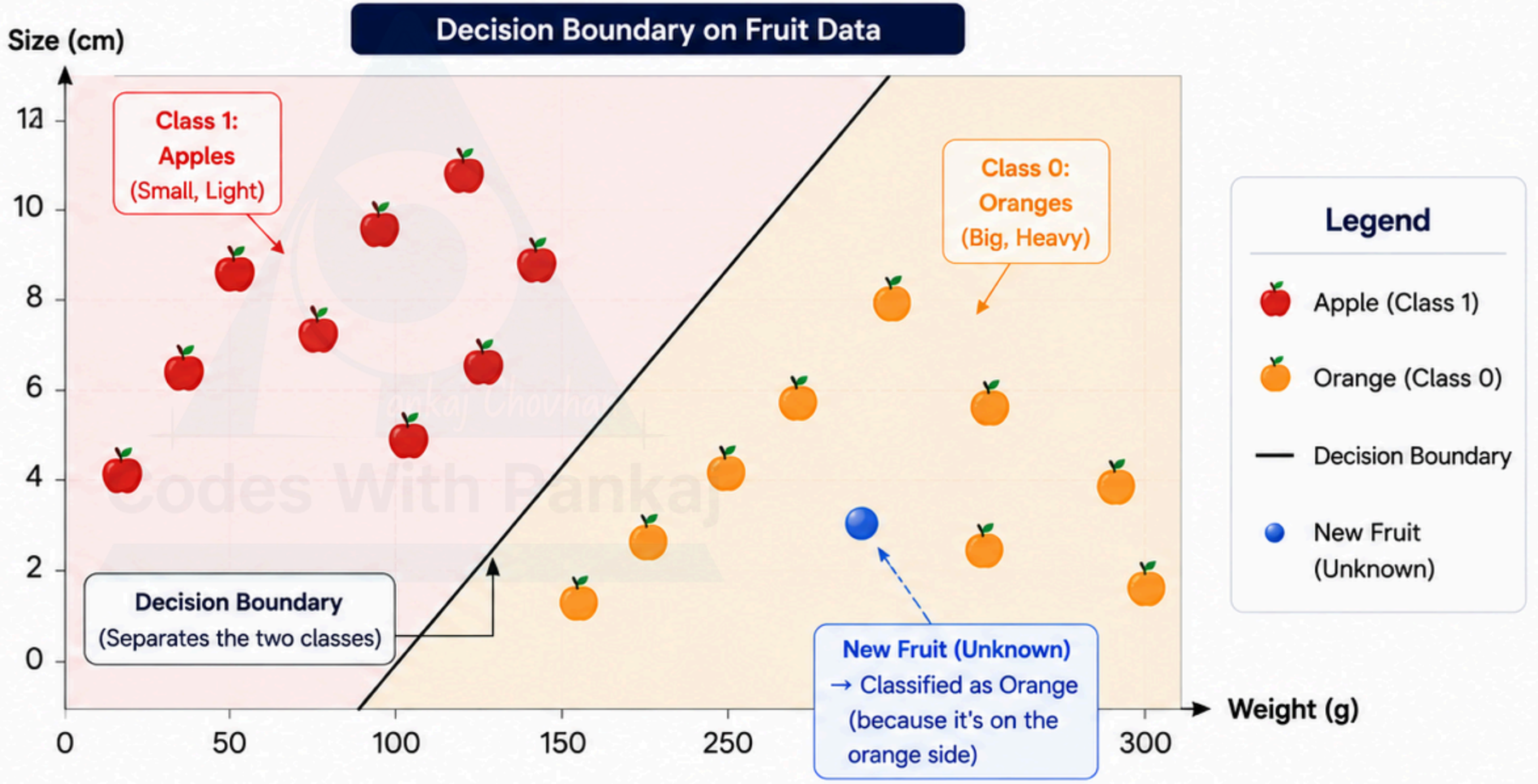
On our fruit graph, we draw a line so apples (small, light) are on one side and oranges (big, heavy) are on the other. A new fruit is classified based on which side it falls.

3. Goal

Find the best boundary that separates classes with minimal errors.

4. Visual Idea

Imagine a line (boundary) splitting apples (left) from oranges (right).



Key Takeaway: The decision boundary helps SVM decide the class of any new data point based on which side of the boundary it falls.

Support Vector Machines (SVM) Step 3: Support Vectors and Hyperplanes

SVM finds the **hyperplane** (decision boundary) that separates classes and is as far as possible from the nearest data points of both classes (**support vectors**).

1. Hyperplane

- A hyperplane is the boundary that separates classes.
- In 2D (our fruit graph), it's a line.
- In 3D, it's a plane (like a sheet).
- In higher dimensions, it's still called a hyperplane (hard to visualize!).

Example: The line separating apples and oranges is a **hyperplane**.

2. Support Vectors

- Support vectors are the data points closest to the hyperplane. They determine its position.
- A few apples and oranges near the line are support vectors. They're the "borderline" fruits that SVM focuses on.

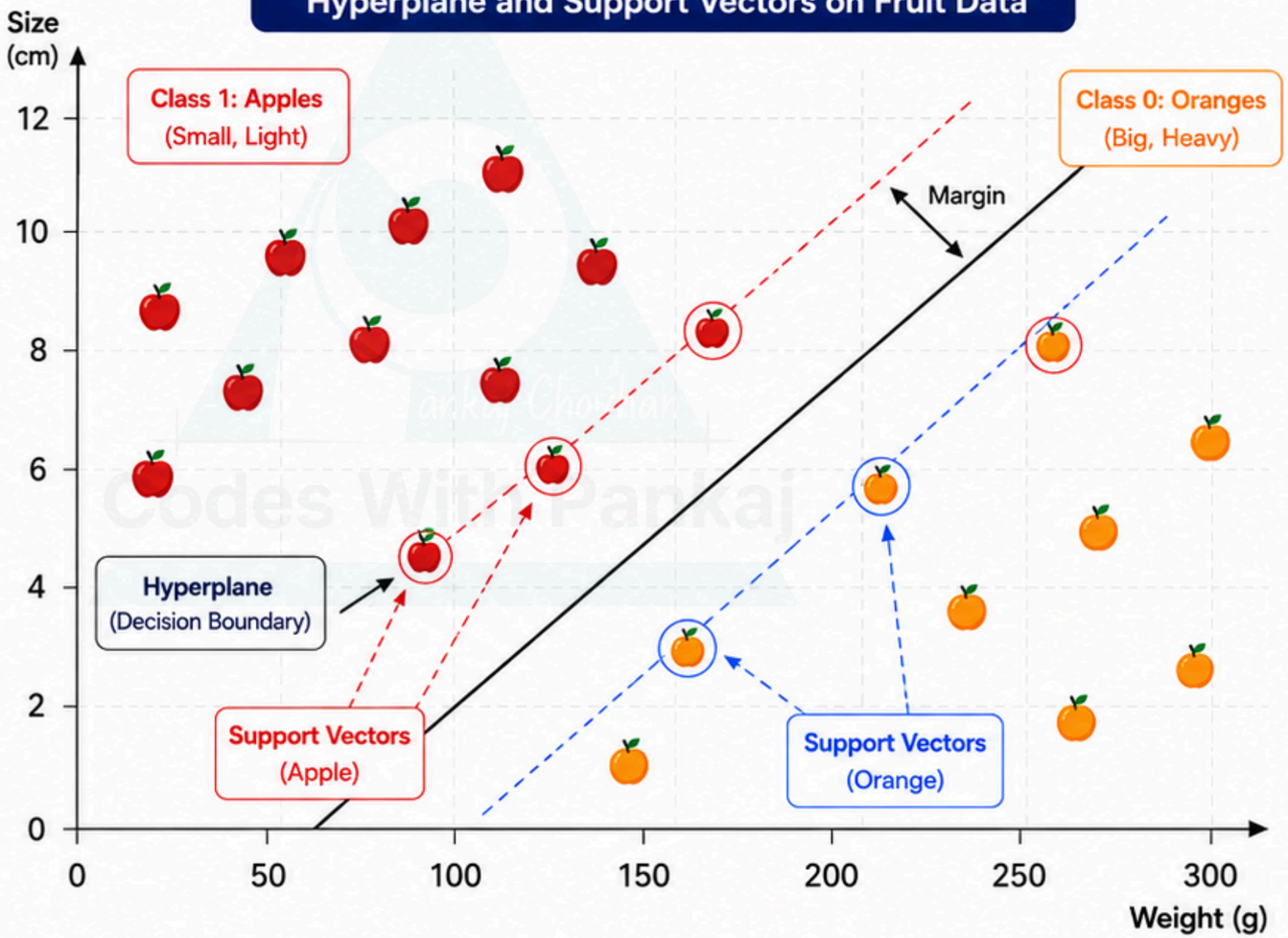
3. Why They Matter

SVM positions the hyperplane to be as far as possible from these support vectors for a robust separation.

4. Visual Idea

Picture a line with a few dots (support vectors) near it, defining a "buffer zone" around the boundary.

Hyperplane and Support Vectors on Fruit Data



Legend

- Apple (Class 1)
- Orange (Class 0)
- Hyperplane (Decision Boundary)
- Margin
- Support Vector (Apple)
- Support Vector (Orange)

Key Takeaway

Only support vectors matter! All other points can move around without changing the hyperplane.

Support Vector Machines (SVM) Step 4: What is a Support Vector Machine?

A **Support Vector Machine** (SVM) is a machine learning algorithm that finds the best hyperplane to separate classes, **maximizing the margin** (distance between the hyperplane and support vectors).

1. Definition

SVM finds the best hyperplane that separates classes with the maximum margin. The points closest to the hyperplane are called support vectors.

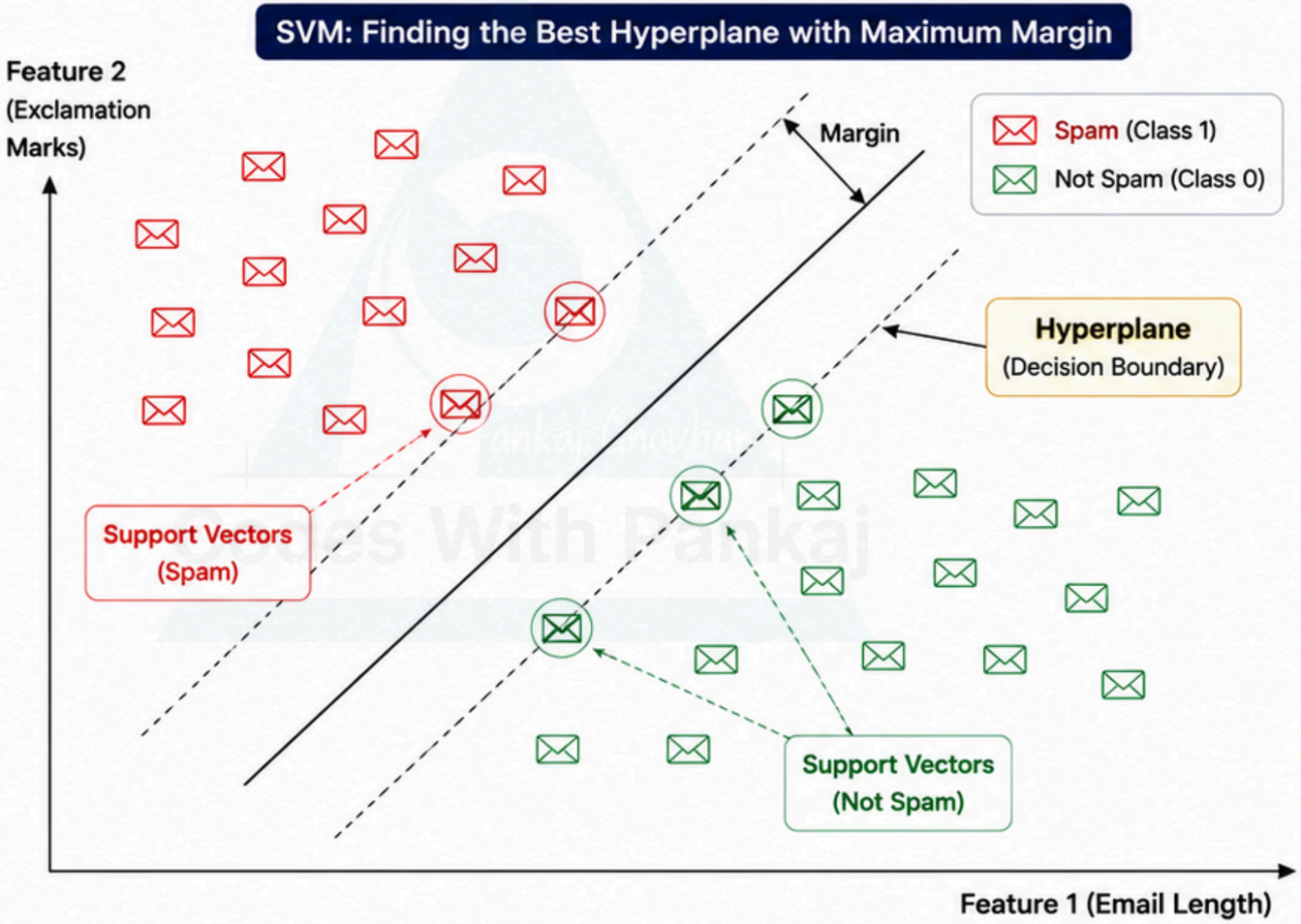
Key Idea: SVM wants the widest possible margin to avoid mistakes.

2. Example: Email Classification

To classify emails as "spam" or "not spam" based on features like length and exclamation marks, SVM draws a hyperplane to separate them, guided by support vectors (emails closest to the boundary).

3. Why Use SVM?

- Great for small-to-medium datasets.
- Works well when classes are separable.



Real-World Use Cases

- Email spam detection
- Image classification
- Text categorization
- Bioinformatics
- Financial fraud detection

Why Maximum Margin?

A larger margin makes the model more robust and better at generalizing to new, unseen data.

In Simple Words

SVM finds the best dividing line (or plane) that keeps the two classes as far apart as possible, using the closest points as guides.

Key Takeaway: SVM is all about finding the best boundary (hyperplane) that maximizes the margin, guided by support vectors, to separate classes effectively.

Support Vector Machines (SVM) Step 5: How Does SVM Work?

SVM finds the **optimal hyperplane** by **maximizing the margin.**
Let's see how with our fruit example.

Example Scenario

We have:

Apples (Class 1):
[100 g, 5 cm], [110 g, 4 cm], [90 g, 5.5 cm]

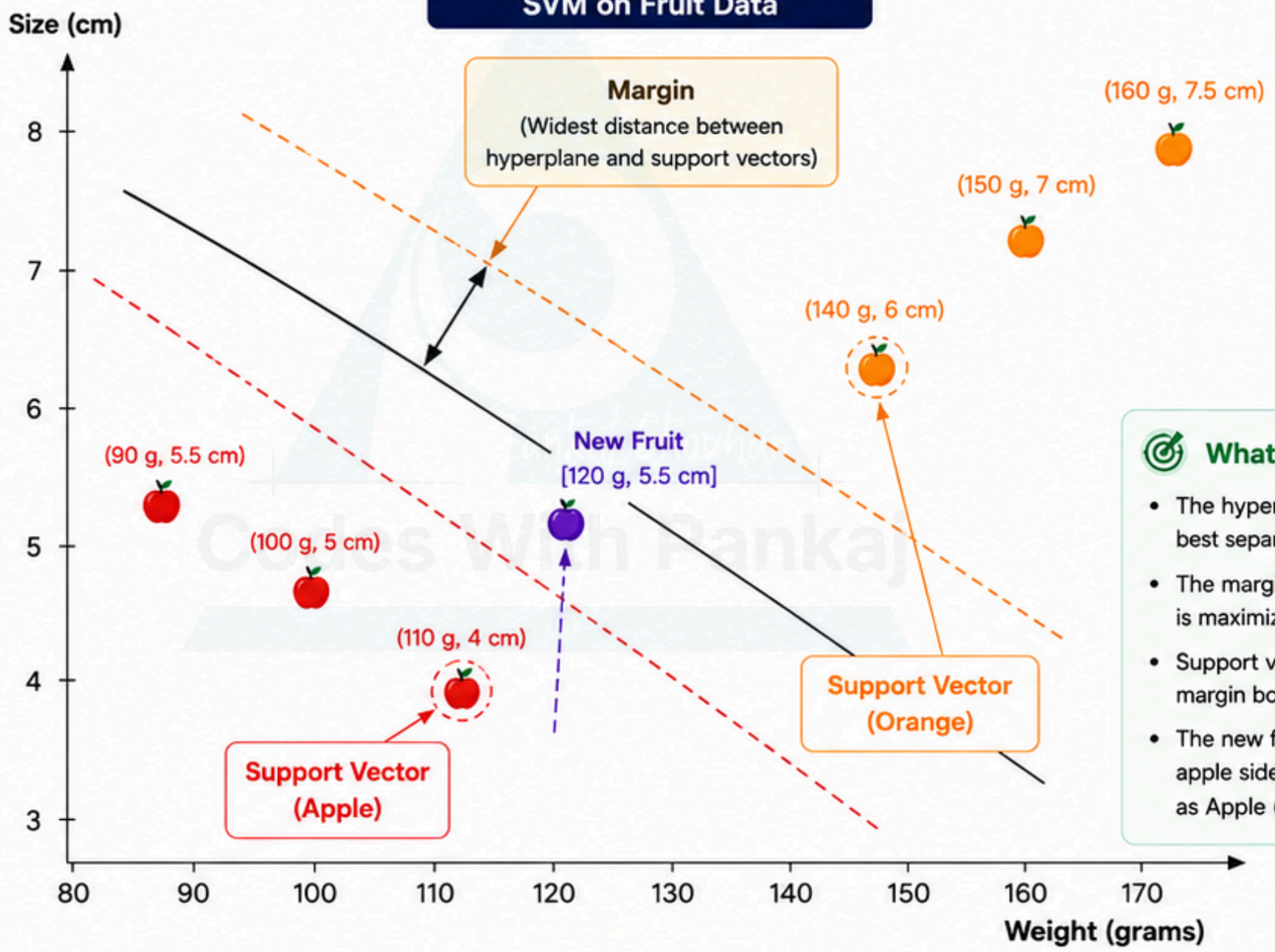
Oranges (Class 0):
[150 g, 7 cm], [140 g, 6 cm], [160 g, 7.5 cm]

Goal:
Classify a new fruit: [120 grams, 5.5 cm]

Steps of SVM

1. **Plot Data:**
   Each fruit is a dot on the graph (x-axis: weight, y-axis: size).
2. **Find Hyperplane:**
   SVM tests lines to separate apples from oranges, picking the one with the widest margin.
3. **Identify Support Vectors:**
   The closest fruits (e.g., [110 g, 4 cm] for apple, [140 g, 6 cm] for orange) guide the hyperplane.
4. **Maximize Margin:**
   SVM ensures the hyperplane is as far as possible from support vectors.
5. **Classify:**
   The new fruit [120 g, 5.5 cm] is classified based on which side of the hyperplane it falls (likely apple).

SVM on Fruit Data



Legend

- Apple (Class 1)
- Orange (Class 0)
- Hyperplane (Decision Boundary)
- Margin (Boundaries)
- Support Vectors (Closest Points)

What's Happening?

- The hyperplane (black line) best separates the two classes.
- The margin (dashed lines) is maximized.
- Support vectors lie on the margin boundaries.
- The new fruit falls on the apple side → classified as Apple (Class 1).

Visual Idea

Imagine a line with a wide buffer zone (margin). Support vectors touch the edges, and the new fruit's position decides its class.

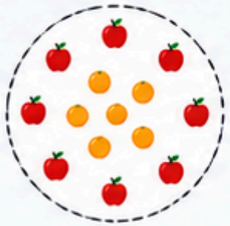
Key Takeaway: SVM finds the best boundary (hyperplane) that maximizes the margin, guided by support vectors, to separate classes and make accurate predictions.

Support Vector Machines (SVM) Step 6: Kernels and Types of Kernels

Sometimes, data cannot be separated with a straight line in 2D.
A **kernel** transforms the data into a **higher-dimensional space** where a hyperplane can separate the classes.

1 What is a Kernel?

- Sometimes, classes can't be separated with a straight line (e.g., oranges circled by apples).
- A kernel transforms data into a higher-dimensional space where a hyperplane can separate them.



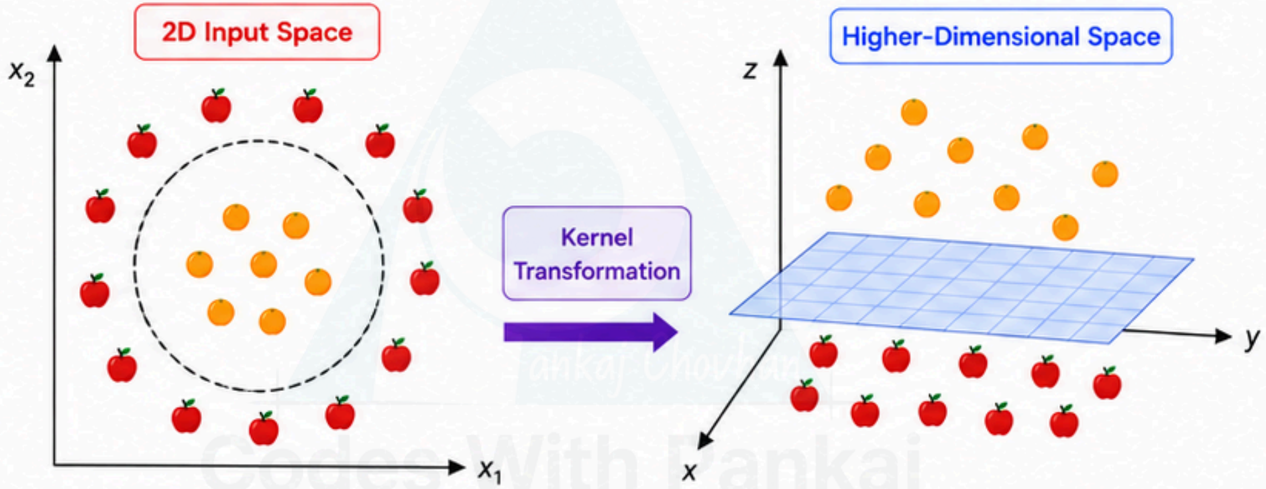
Example:

If apples and oranges are mixed in a circular pattern, a kernel "lifts" the data to a space where they're separable.

2 Why Kernels Matter?

Kernels make SVM flexible and powerful, allowing it to handle complex, real-world datasets that are not linearly separable.

From 2D (Not Separable) to Higher-Dimensional (Separable)



Not separable with a straight line in 2D.

In higher dimensions, a hyperplane (plane) can separate the classes.

3 Types of Kernels

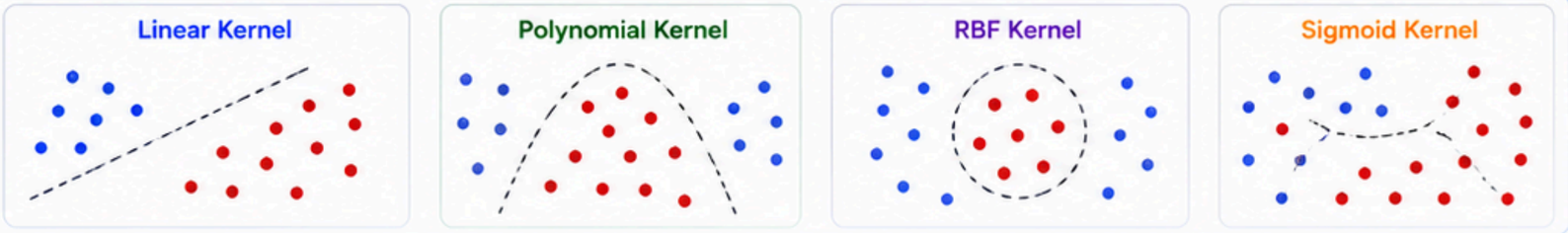
1. Linear Kernel
- Uses a straight line.
- Best for data like our fruits (linearly separable).

2. Polynomial Kernel
- Allows curved boundaries.
- Useful for slightly complex patterns.

3. RBF (Radial Basis Function) Kernel
- Handles complex, non-linear patterns.
- Most commonly used kernel.

4. Sigmoid Kernel
- Less common.
- Used in specific cases.

How Different Kernels Create Decision Boundaries



Key Takeaway

Kernels transform the data so that SVM can find the best hyperplane to separate classes, even in complex, non-linear scenarios.

Support Vector Machines (SVM) Step 7: Hard Margin vs. Soft Margin

SVM can find different types of optimal hyperplanes. **Hard margin** aims for **perfect separation**, while **soft margin** allows **some errors** to handle real-world (noisy) data.

1 Hard Margin

- Requires perfect separation.
- No data points are allowed on the wrong side.
- Works only when data is clean and perfectly separable.

Example:
Apples and oranges are neatly separated. A strict line with maximum margin is drawn.

2 Soft Margin

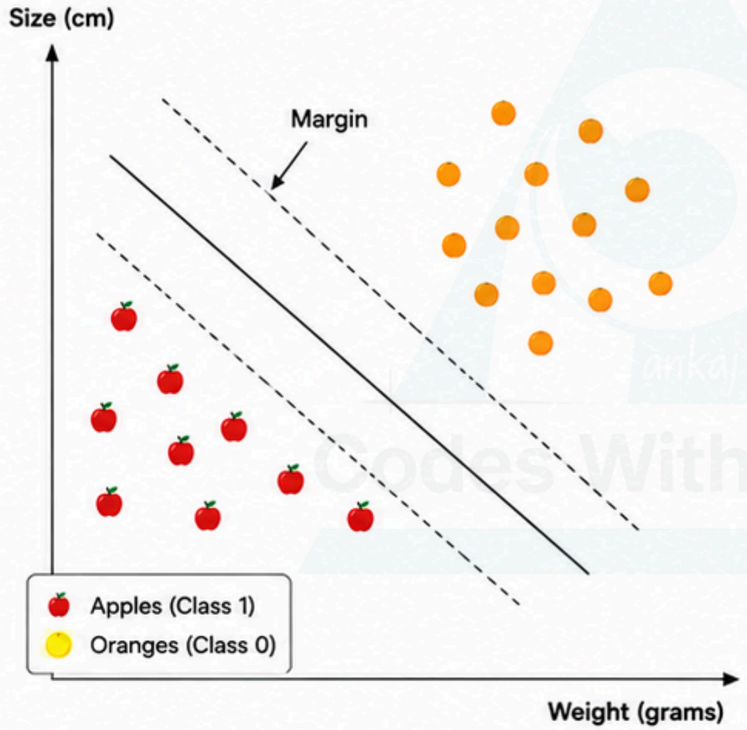
- Allows some misclassifications.
- Tries to maximize the margin while minimizing errors.
- Works well with noisy, real-world data.

Example:
One apple is among oranges. Soft margin allows this error to find a better hyperplane.

Why Soft Margin?

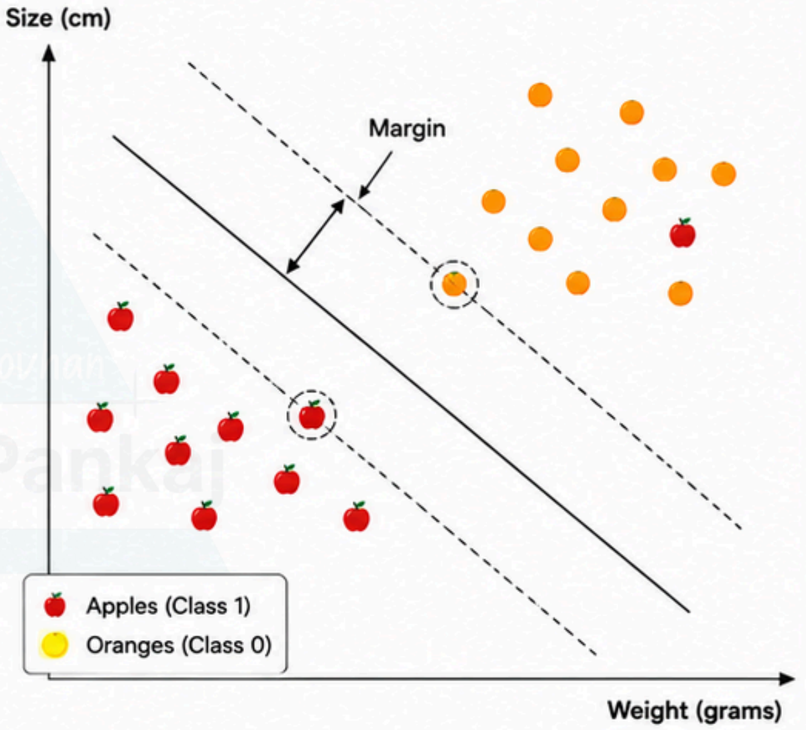
- Real data is rarely perfect.
- Soft margin provides a balance between a wide margin and fewer misclassifications.

Hard Margin (Perfect Separation)



All points are on the correct side.
Maximum margin with no errors.

Soft Margin (Allow Some Errors)



Allows a few misclassifications (e.g., one apple).
Finds a good balance between margin and errors.

Legend

- Hyperplane (Decision Boundary)
- Margin (Boundaries)
- Support Vectors (Points on Margin)

Key Takeaway

Hard margin is strict and works only for perfect data.
Soft margin is flexible and handles real-world (noisy) data better.

In Simple Words

Hard margin says "No mistakes allowed!"
Soft margin says "A few mistakes are okay for a better overall boundary."

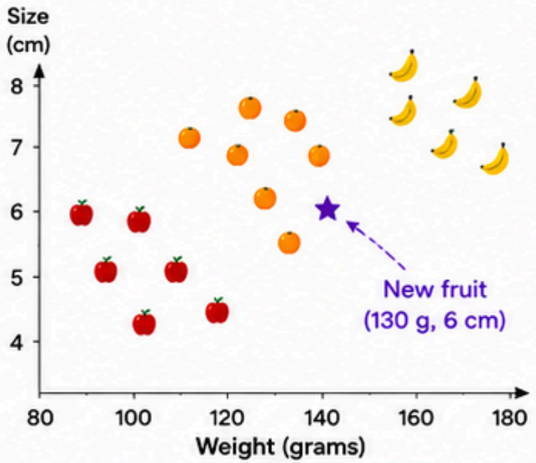
Real-World Analogy: Hard margin is like drawing a strict fence that excludes everything. | Soft margin is like a flexible fence that allows a few exceptions.

Support Vector Machines (SVM) Step 8: SVM for Multi-Class Classification

SVM is built for binary classification (two classes). For more classes (e.g., apples, oranges, bananas), SVM uses strategies to break the problem into multiple binary problems.

Our Problem (3 Classes)

Example: Classify a new fruit [130 grams, 6 cm]



Why it matters

These strategies extend SVM from two classes to multiple classes, making it useful for real-world problems with many categories.

Two Popular Strategies

1 One-vs-Rest (OvR)

One SVM is trained per class to distinguish that class from all other classes.

How it works

Train 3 SVMs:
- SVM 1: Apples vs. Others
- SVM 2: Oranges vs. Others
- SVM 3: Bananas vs. Others

The class with the **strongest vote** wins.

Prediction for new fruit [130 g, 6 cm]:
Each SVM votes (Apple / Orange / Banana).
If "Oranges vs. Others" has the strongest score → Predicted class: Orange.

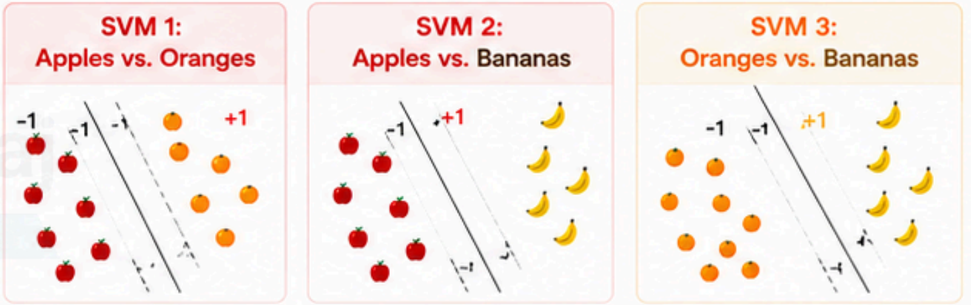
2 One-vs-One (OvO)

One SVM is trained for every pair of classes.

How it works

Train 3 SVMs (for 3 classes):
- SVM 1: Apples vs. Oranges
- SVM 2: Apples vs. Bananas
- SVM 3: Oranges vs. Bananas

Each SVM votes for one class. **Most votes** wins.



Prediction for new fruit [130 g, 6 cm]:
Each SVM votes (Apple / Orange / Banana).
Whichever class gets the most votes → Predicted class: Orange.

Key Takeaway

SVM handles multi-class problems using these strategies:
- OvR: Simpler, scales well with more classes.
- OvO: More comparisons, but can be effective with small datasets.

Strategy	# of SVMs (for k classes)	Pros	Cons
One-vs-Rest (OvR)	k	Simple, scalable to many classes	Can be less accurate if classes are imbalanced
One-vs-One (OvO)	k (k – 1) / 2	Often more accurate for complex boundaries	More SVMs to train as classes increase

SUPPORT VECTOR MACHINES (SVM) Step 9: Hands-On Python Example

Let's bring SVM to life! We'll use **scikit-learn** to classify fruits (apples vs. oranges) based on **weight** and **size**, then predict a new fruit. This example ties together vectors, hyperplanes, support vectors, margins, and classification.

Prerequisites

- Python (3.6+)
- Libraries: scikit-learn, numpy, matplotlib

Install them with:

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

Our Dataset (Vectors)

Fruit	Label	Weight (g)	Size (cm)
Apple	0	100	5.0
Apple	0	110	4.0
Apple	0	90	5.5
Orange	1	150	7.0
Orange	1	140	6.0
Orange	1	160	7.5

0 = Apple 1 = Orange

Prediction Goal

Predict the class of a new fruit:

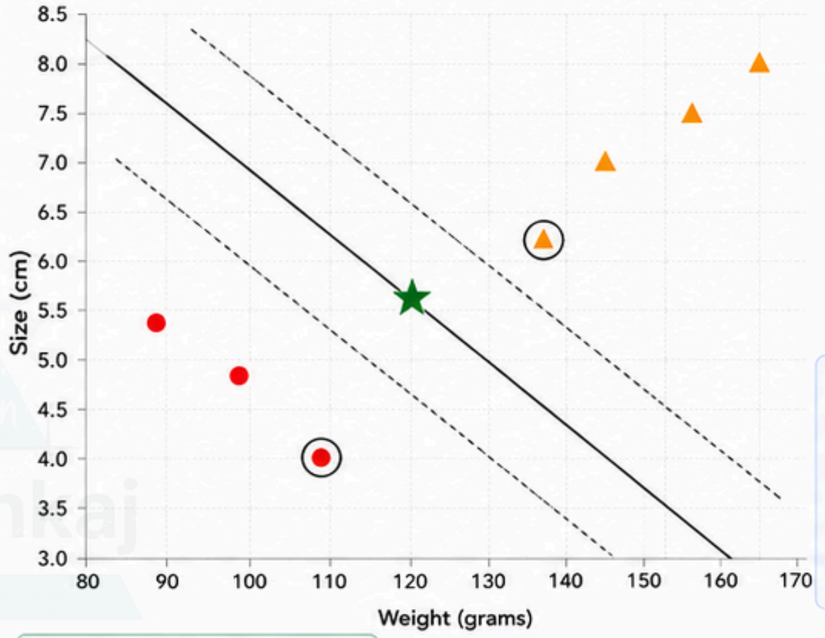
New fruit = [120 grams, 5.5 cm]

Will it be an Apple (0) or Orange (1)?

Python Code (scikit-learn)

```python
# 1. Import libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# 2. Create dataset (vectors)
X = np.array([[100, 5], [110, 4], [90, 5.5],
              [150, 7], [140, 6], [160, 7.5]])
y = np.array([0, 0, 0, 1, 1, 1])  # 0=Apple, 1=Orange

# 3. Train SVM (linear kernel, soft margin)
model = svm.SVC(kernel='linear', C=1.0)  # C controls soft margin
model.fit(X, y)

# 4. Get hyperplane & support vectors
support_vectors = model.support_vectors_
w = model.coef_[0]
b = model.intercept_[0]

# 5. Plot data, boundary, margins, support vectors
plt.scatter(X[y==0][:,0], X[y==0][:,1], c='red', label='Apples', s=80)
plt.scatter(X[y==1][:,0], X[y==1][:,1], c='orange', marker='^', s=80)
plt.scatter(support_vectors[:,0], support_vectors[:,1], s=120,
            facecolors='none', edgecolors='black', label='Support Vectors')

# Decision boundary: w0*x0 + w1*x1 + b = 0
x0 = np.linspace(80, 170, 100)
x1 = -(w[0]*x0 + b) / w[1]
plt.plot(x0, x1, 'k-', linewidth=2, label='Decision Boundary')

# Margins: w0*x0 + w1*x1 + b = ±1
x1_m1 = -(w[0]*x0 + b - 1) / w[1]  # +1 margin
x1_m2 = -(w[0]*x0 + b + 1) / w[1]  # -1 margin
plt.plot(x0, x1_m1, 'k--', label='Margins')
plt.plot(x0, x1_m2, 'k--')

# 6. Predict a new fruit
new_fruit = np.array([[120, 5.5]])
prediction = model.predict(new_fruit)
print(f"New fruit {new_fruit[0]} is: {'Apple' if prediction[0]==0 else 'Orange'}")
plt.scatter(new_fruit[:,0], new_fruit[:,1], marker='*', c='green',
            s=250, label='New Fruit')

# Labels and show plot
plt.xlabel('Weight (grams)')
plt.ylabel('Size (cm)')
plt.title('SVM: Apples vs. Oranges (Linear Kernel, C=1.0)')
plt.legend(loc='best')
plt.grid(True, linestyle='--', alpha=0.3)
plt.tight_layout()
plt.show()
```

SVM: Apples vs. Oranges (Linear Kernel, C=1.0)



Why Margin Matters?

The wider the margin, the better the model generalizes to unseen data and reduces the chance of misclassification.

Console Output

```
New fruit [120.  5.5] is: Apple
```

Code Explanation

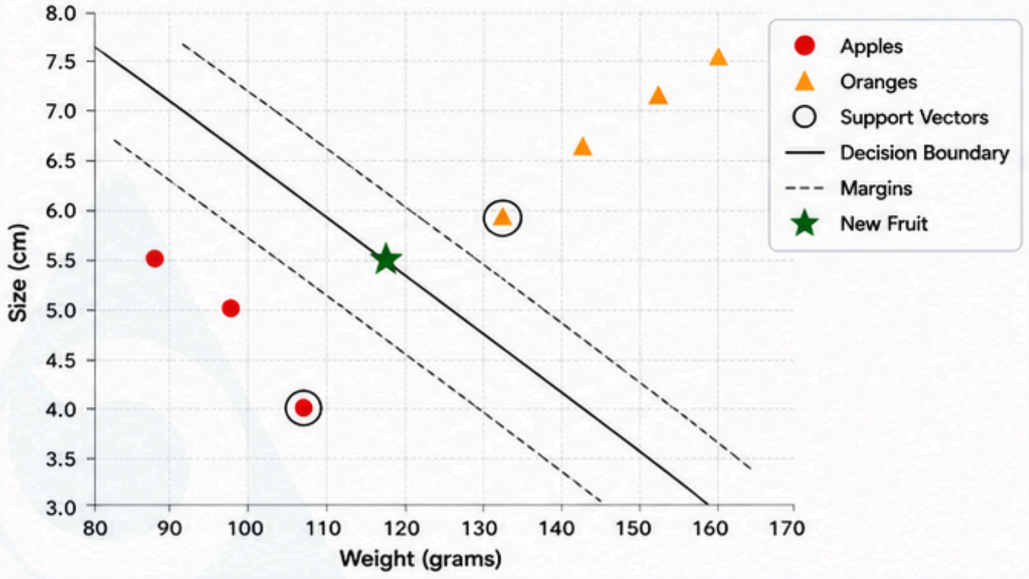
- **Vectors:** Each fruit is a vector in `X = [weight, size]`.
- **Decision Boundary:** Black solid line ($w_0x_0 + w_1x_1 + b = 0$).
- **Support Vectors:** Circled points closest to the boundary.
- **Margins:** Dashed lines ($w_0x_0 + w_1x_1 + b = \pm 1$) form the buffer zone.
- **Hyperplane:** Computed using `model.coef_` and `model.intercept_`.
- **Soft Margin:** `C = 1.0` allows flexibility for small errors.
- **Classification:** The new fruit `[120, 5.5]` is on the Apple side → Predicted as Apple (0).

SVM Example in Python

1. Binary Classification: Apples vs. Oranges

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# Create dataset: [weight (grams), size (cm)]
X = np.array([
    [100, 5],      # Apple
    [110, 4],      # Apple
    [90, 5.5],     # Apple
    [150, 7],      # Orange
    [140, 6],      # Orange
    [160, 7.5]     # Orange
])
y = np.array([0, 0, 0, 1, 1, 1])  # 0=Apple, 1=Orange

# Train SVM (linear kernel, soft margin)
model = svm.SVC(kernel='linear', C=1.0)
model.fit(X, y)

# Get hyperplane and support vectors
support_vectors = model.support_vectors_
w = model.coef_[0]
b = model.intercept_[0]

# Plot data, boundary, margins, support vectors
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='red', label='Apples', marker='o')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='orange', label='Oranges', marker='^')
plt.scatter(support_vectors[:, 0], support_vectors[:, 1], s=100, facecolors='none',
            edgecolors='black', label='Support Vectors')

# Decision boundary
x0 = np.linspace(80, 170, 100)
x1 = -(w[0] * x0 + b) / w[1]
plt.plot(x0, x1, 'k-', label='Decision Boundary')

# Margins
x1_margin1 = -(w[0] * x0 + b - 1) / w[1]
x1_margin2 = -(w[0] * x0 + b + 1) / w[1]
plt.plot(x0, x1_margin1, 'k--', label='Margins')
plt.plot(x0, x1_margin2, 'k--')

# Predict new fruit
new_fruit = np.array([[120, 5.5]])
prediction = model.predict(new_fruit)
print(f"New fruit {new_fruit} is: {'Apple' if prediction[0] == 0 else 'Orange'}")

# Plot new fruit
plt.scatter(new_fruit[:, 0], new_fruit[:, 1], color='green', marker='*', s=200, label='New Fruit')

# Labels and plot
plt.xlabel('Weight (grams)')
plt.ylabel('Size (cm)')
plt.title('SVM: Apples vs. Oranges')
plt.legend()
plt.grid(True)
plt.show()
```

Binary Classification: Apples vs. Oranges



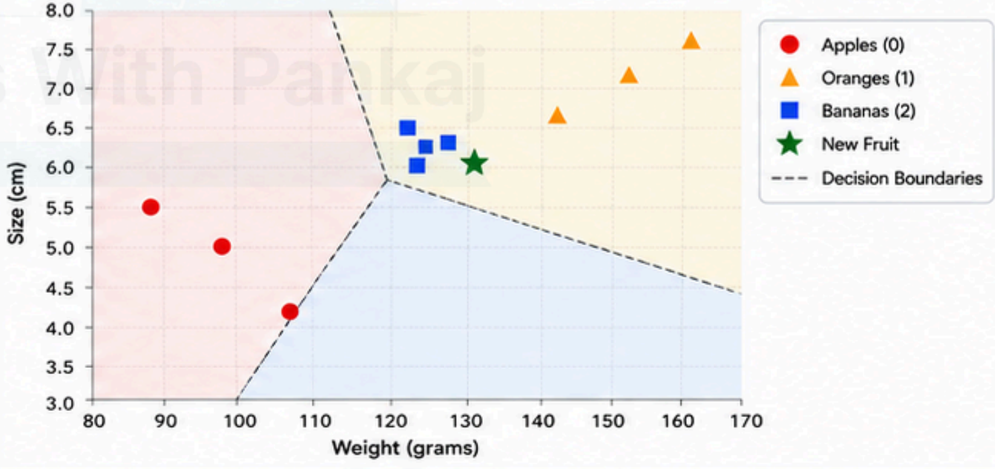
What does this plot show?

- **Red circles:** Apples (class 0)
- **Orange triangles:** Oranges (class 1)
- **Black line:** Decision boundary (hyperplane)
- **Dashed lines:** Margins (maximum distance from the boundary)
- **Black circles:** Support vectors (closest points to the boundary)
- **Green star:** New fruit to be classified

Result (Binary)

New fruit [[120. 5.5]] is: Apple

Multi-class Classification: Apples, Oranges, Bananas



Multi-class with Linear SVM

- SVM creates multiple linear boundaries to separate all classes.
- The background colors show the predicted regions for each class.

Result (Multi-class)

New fruit [[122. 6.]] is: Banana

Console Output

```
New fruit [[120.  5.5]] is: Apple
New fruit [[122.  6. ]] is: Banana
```