PPT by:
Sripooja Mallam

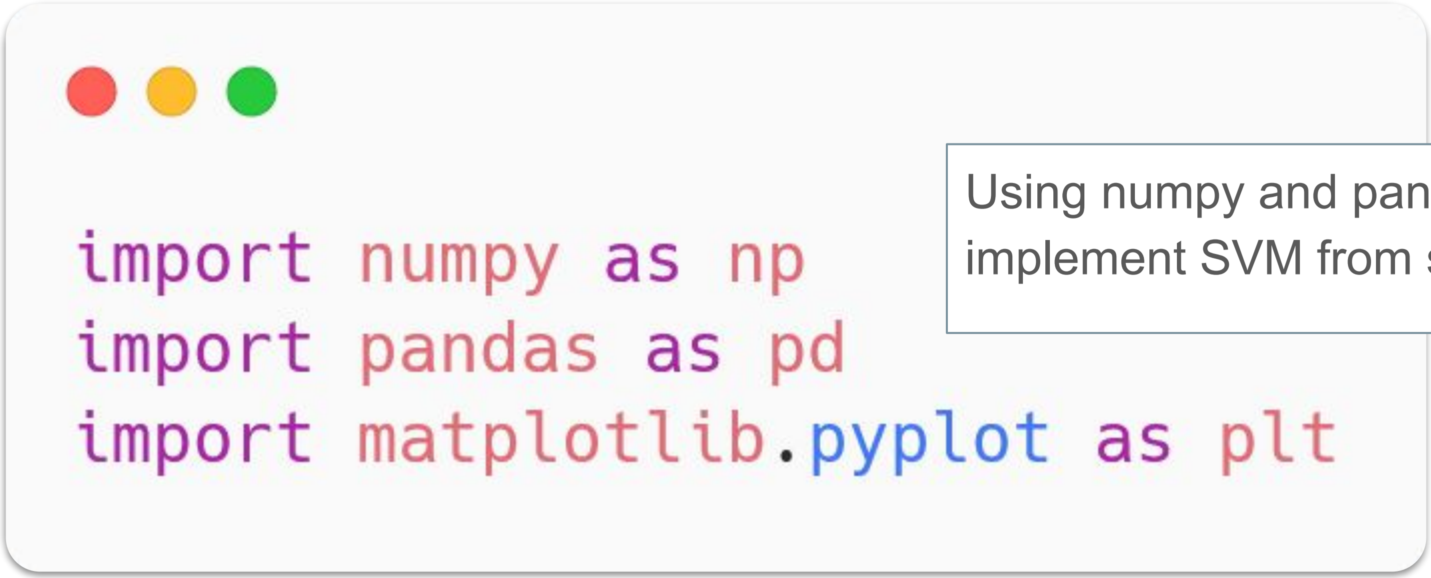
Support Vector Machine (SVM)

Code walkthrough

Neil Gogte
KMIT

Basic SVM code

PPT by:
Sripooja Mallam



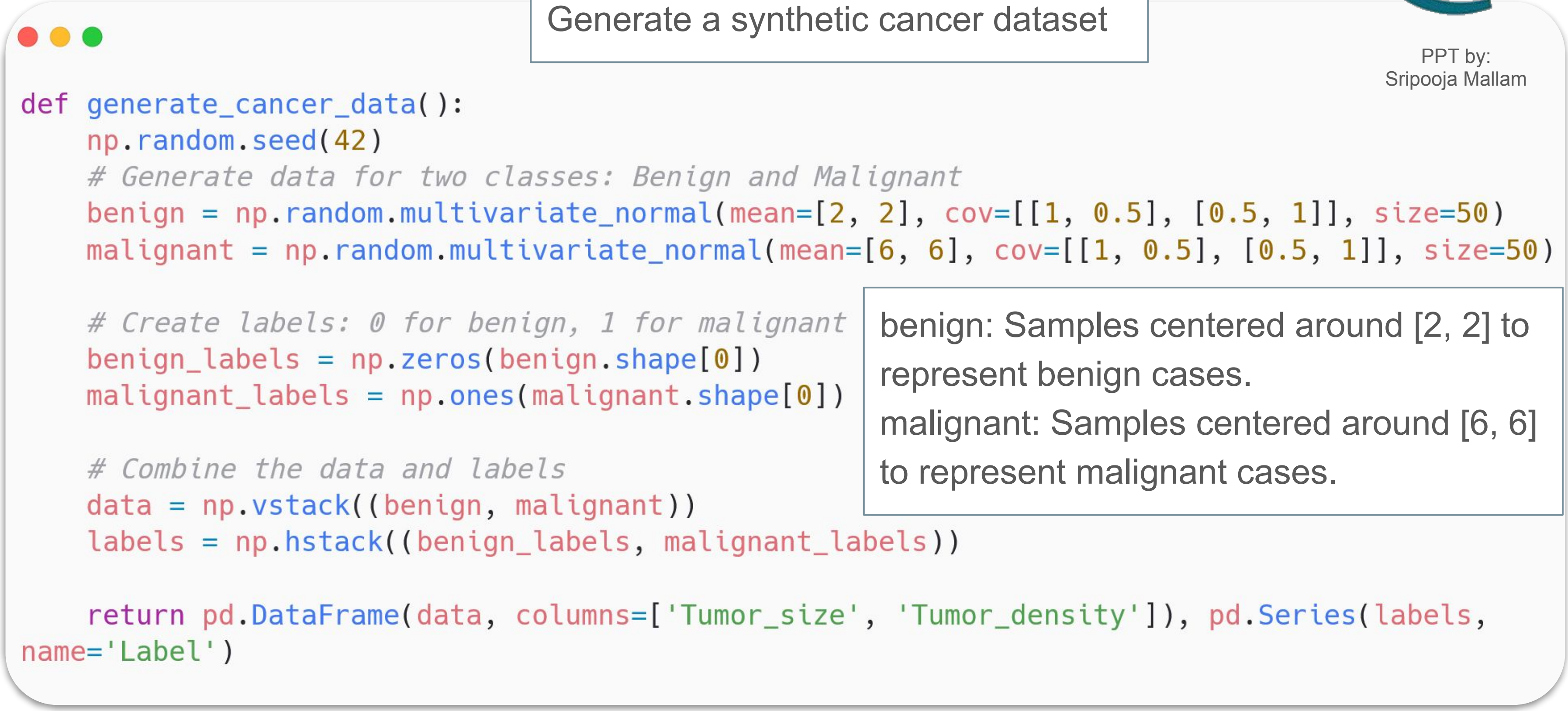
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Using numpy and pandas to implement SVM from scratch

Basic SVM code

PPT by:
Sripooja Mallam

Generate a synthetic cancer dataset



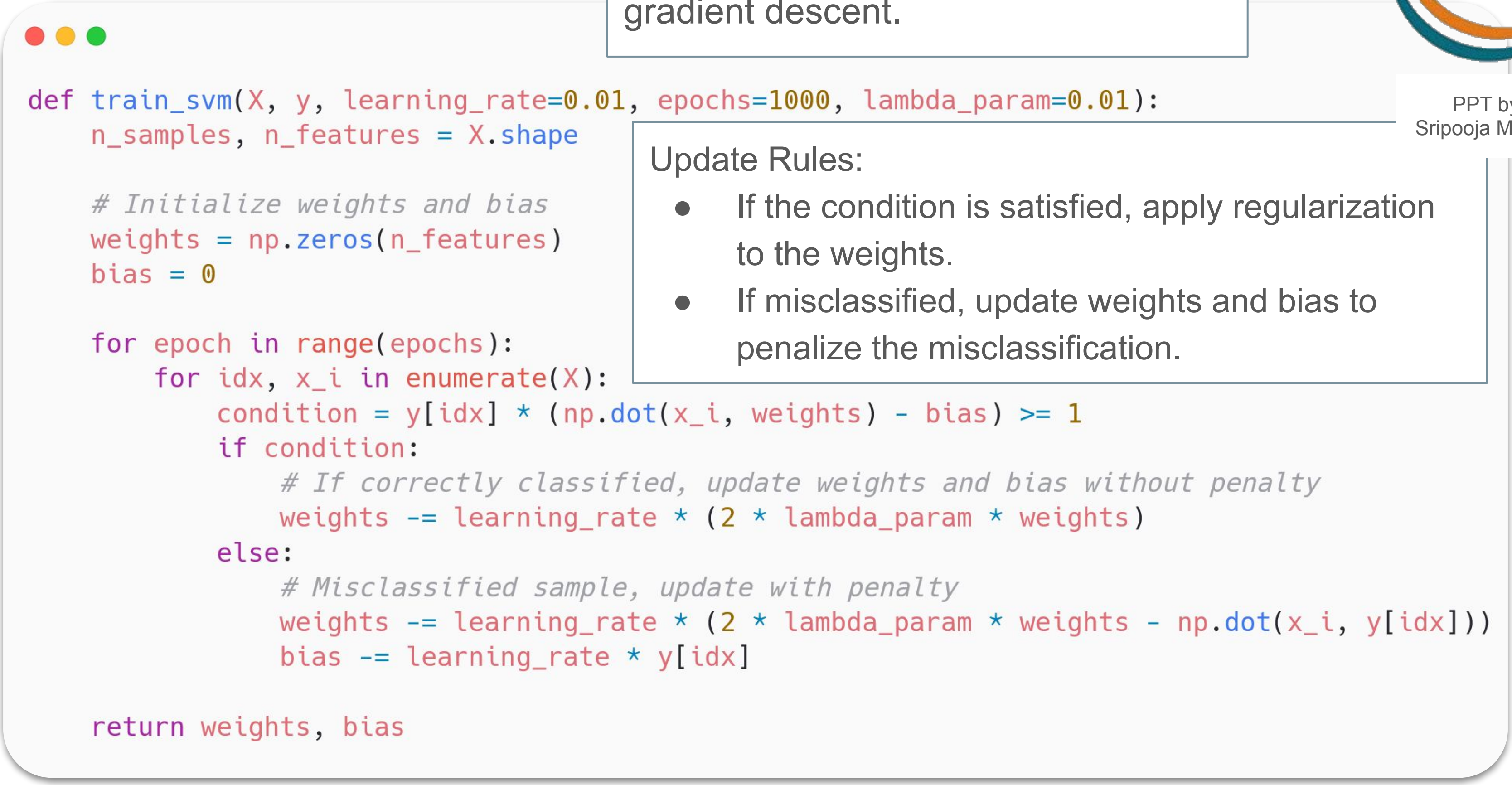
```python
def generate_cancer_data():
    np.random.seed(42)
    # Generate data for two classes: Benign and Malignant
    benign = np.random.multivariate_normal(mean=[2, 2], cov=[[1, 0.5], [0.5, 1]], size=50)
    malignant = np.random.multivariate_normal(mean=[6, 6], cov=[[1, 0.5], [0.5, 1]], size=50)

    # Create labels: 0 for benign, 1 for malignant
    benign_labels = np.zeros(benign.shape[0])
    malignant_labels = np.ones(malignant.shape[0])

    # Combine the data and labels
    data = np.vstack((benign, malignant))
    labels = np.hstack((benign_labels, malignant_labels))

    return pd.DataFrame(data, columns=['Tumor_size', 'Tumor_density']), pd.Series(labels,
name='Label')
```

benign: Samples centered around [2, 2] to represent benign cases.
malignant: Samples centered around [6, 6] to represent malignant cases.

Basic SVM code

Implement a basic linear SVM using gradient descent.

PPT by:
Sripooja Mallam



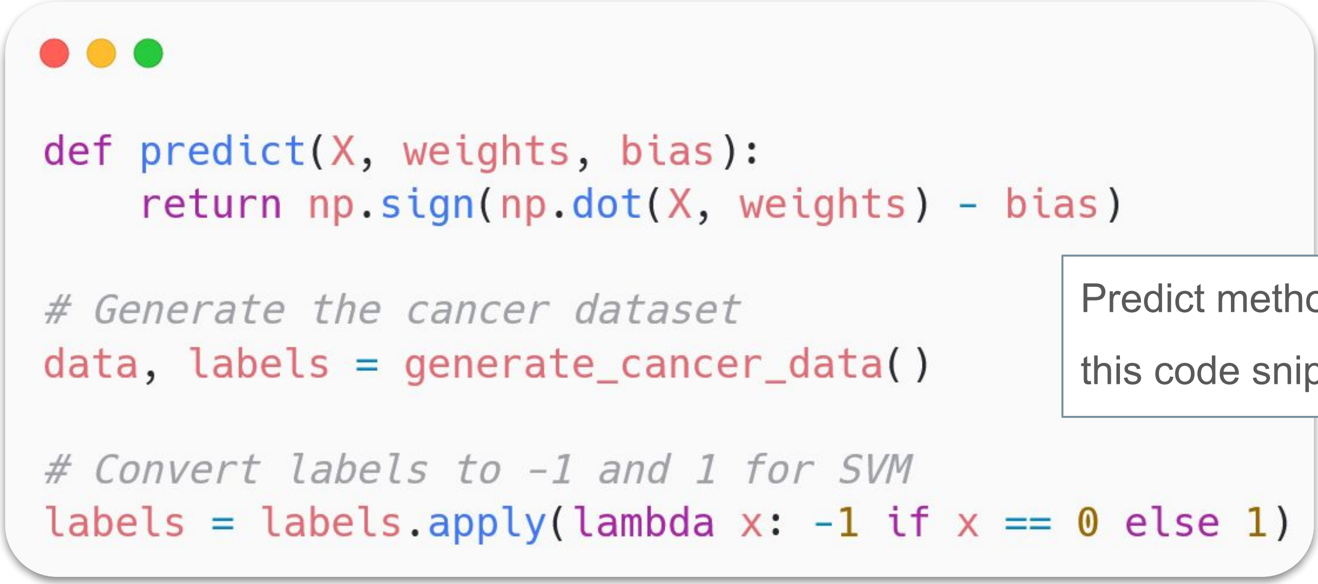
```python
def train_svm(X, y, learning_rate=0.01, epochs=1000, lambda_param=0.01):
    n_samples, n_features = X.shape

    # Initialize weights and bias
    weights = np.zeros(n_features)
    bias = 0

    for epoch in range(epochs):
        for idx, x_i in enumerate(X):
            condition = y[idx] * (np.dot(x_i, weights) - bias) >= 1
            if condition:
                # If correctly classified, update weights and bias without penalty
                weights -= learning_rate * (2 * lambda_param * weights)
            else:
                # Misclassified sample, update with penalty
                weights -= learning_rate * (2 * lambda_param * weights - np.dot(x_i, y[idx]))
                bias -= learning_rate * y[idx]

    return weights, bias
```

Update Rules:

- If the condition is satisfied, apply regularization to the weights.
- If misclassified, update weights and bias to penalize the misclassification.

Basic SVM code

PPT by:
Sripooja Mallam

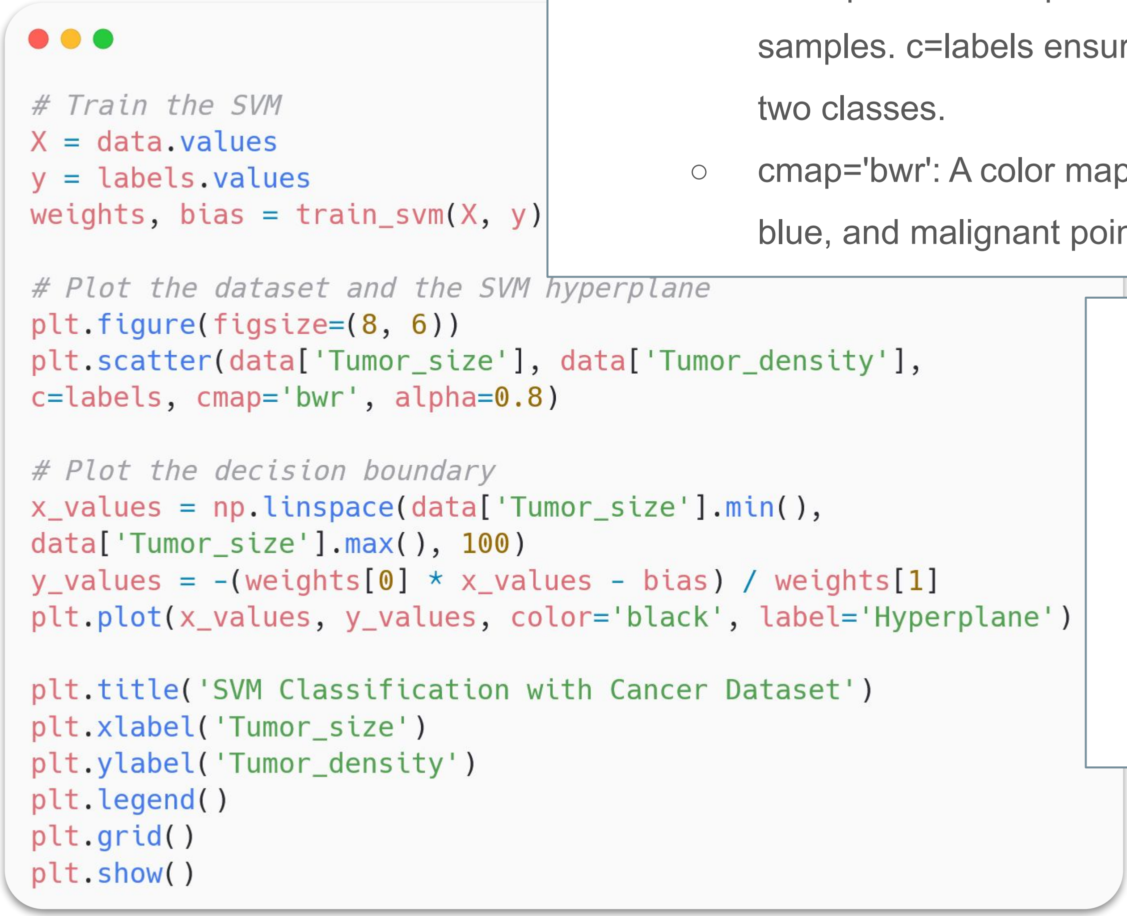


```
def predict(X, weights, bias):
    return np.sign(np.dot(X, weights) - bias)

# Generate the cancer dataset
data, labels = generate_cancer_data()

# Convert labels to -1 and 1 for SVM
labels = labels.apply(lambda x: -1 if x == 0 else 1)
```

Predict method not invoked in this code snippet

- Computes the dot product between input features X and learned weights weights, subtracting the bias.
- Uses the np.sign function to classify:
  - Returns **1 for malignant** and **-1 for benign**.

Basic SVM code



```
# Train the SVM
X = data.values
y = labels.values
weights, bias = train_svm(X, y)

# Plot the dataset and the SVM hyperplane
plt.figure(figsize=(8, 6))
plt.scatter(data['Tumor_size'], data['Tumor_density'],
c=labels, cmap='bwr', alpha=0.8)

# Plot the decision boundary
x_values = np.linspace(data['Tumor_size'].min(),
data['Tumor_size'].max(), 100)
y_values = -(weights[0] * x_values - bias) / weights[1]
plt.plot(x_values, y_values, color='black', label='Hyperplane')

plt.title('SVM Classification with Cancer Dataset')
plt.xlabel('Tumor_size')
plt.ylabel('Tumor_density')
plt.legend()
plt.grid()
plt.show()
```

- Data Visualization:
  - Uses plt.scatter to plot benign and malignant samples. c=labels ensures different colors for the two classes.
  - cmap='bwr': A color map where benign points are blue, and malignant points are red.

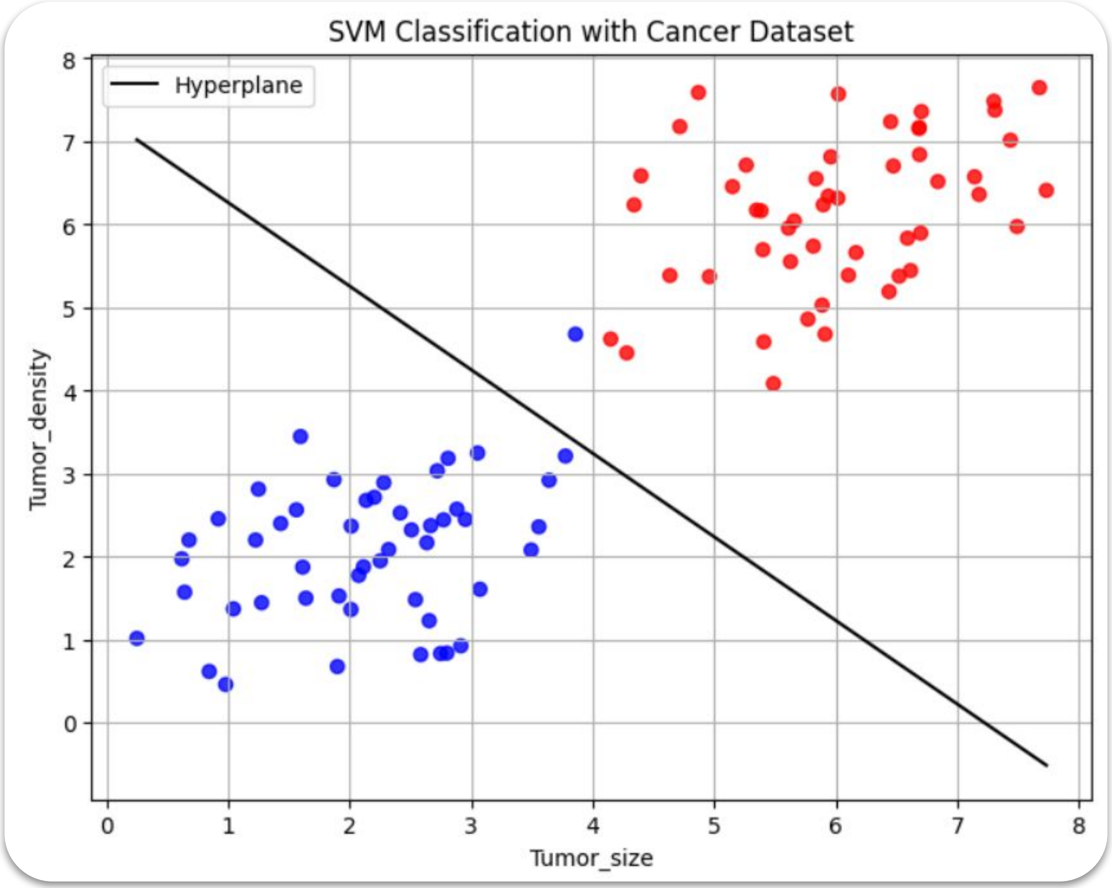
- Hyperplane Visualization:
  - y_values = -(weights[0] * x_values - bias) / weights[1] calculates the decision boundary (hyperplane equation for SVM).
  - Plotted using plt.plot

Kmit

PPT by:
Sripooja Mallam

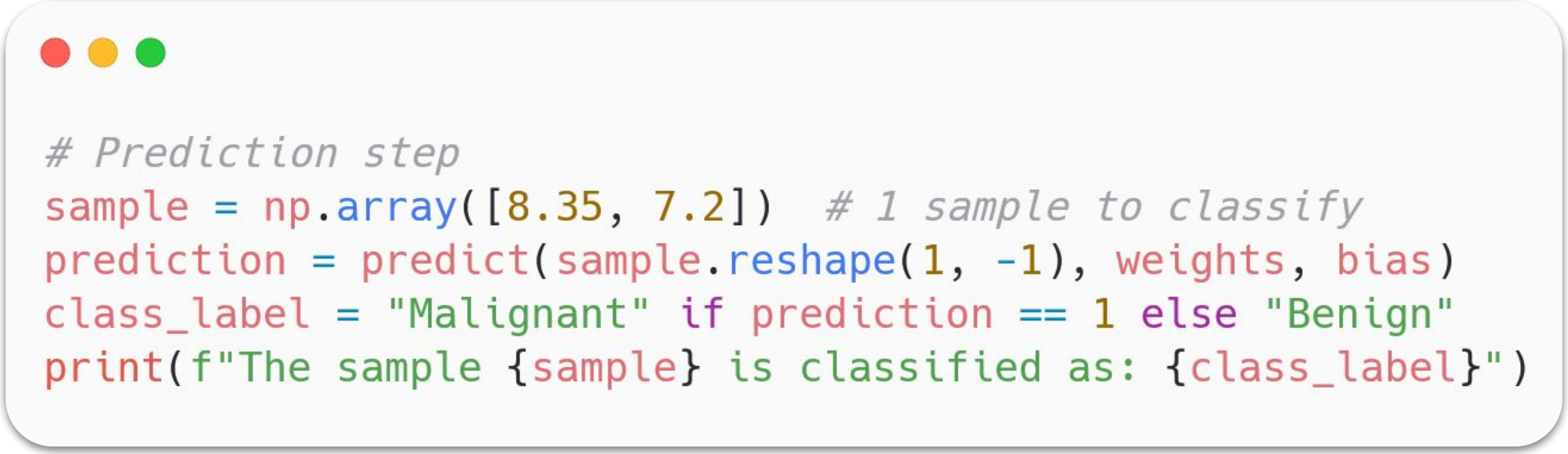
Basic SVM code

PPT by:
Sripooja Mallam



Basic SVM code

PPT by:
Sripooja Mallam



```python
# Prediction step
sample = np.array([8.35, 7.2])  # 1 sample to classify
prediction = predict(sample.reshape(1, -1), weights, bias)
class_label = "Malignant" if prediction == 1 else "Benign"
print(f"The sample {sample} is classified as: {class_label}")
```

Output is:

```
The sample [8.35 7.2 ] is classified as: Malignant
```