The Boundary Boss

SVM

Prompts

- Train an SVM classifier with 'yammy' as the target
- Visualize the classified data and decision boundaries in 2D space

Strategy

- Prepare the data
- Train the SVM classifier with linear and RBF kernels
- Check the Accuracy
- Visualise decision boundaries

Code & Results

1. Prepare the data

This code checks the "star_rating" and puts a 1 in the new "yammy" column if it is greater than or equal to 4.5.

```
import pandas as pd

# Assuming you already have a DataFrame named df
data['yammy'] = (data['star_rating'] >= 4.5).astype(int)
data.head()
```

This code puts features between 0 and 1.

```
from sklearn.preprocessing import MinMaxScaler

# Select only numeric columns (excluding 'yammy' if it's a label)
features = data.drop(columns=['yammy'])  # Drop target column if needed
numeric_cols = features.select_dtypes(include=['float64', 'int64']).columns

scaler = MinMaxScaler()
data[numeric_cols] = scaler.fit_transform(data[numeric_cols])

data[numeric_cols].head()
```

This code first splits the teacher from the features and then splits the data into training and testing sets.

```
from sklearn.model_selection import train_test_split

# Separate features and target
X = data.drop(columns=['yammy','star_rating'])
y = data['yammy']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Train the Model

This code creates an SVM classifier with a linear kernel and trains it using the training set.

```
from sklearn.svm import SVC

# Create and fit the model
svm = SVC(kernel='linear')  # 'rbf' is the default; you can also try 'linear', 'poly', or 'sigmoid'
svm.fit(X_train, y_train)
```

3. Check the Accuracy & Confusion Matrix

This code evaluates model accuracy and calculates a confusion matrix.

```
y_pred = svm.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.84
Confusion Matrix:
 [[11  4]
 [ 0 10]]
```

✅ The model got 84% of the tins right.

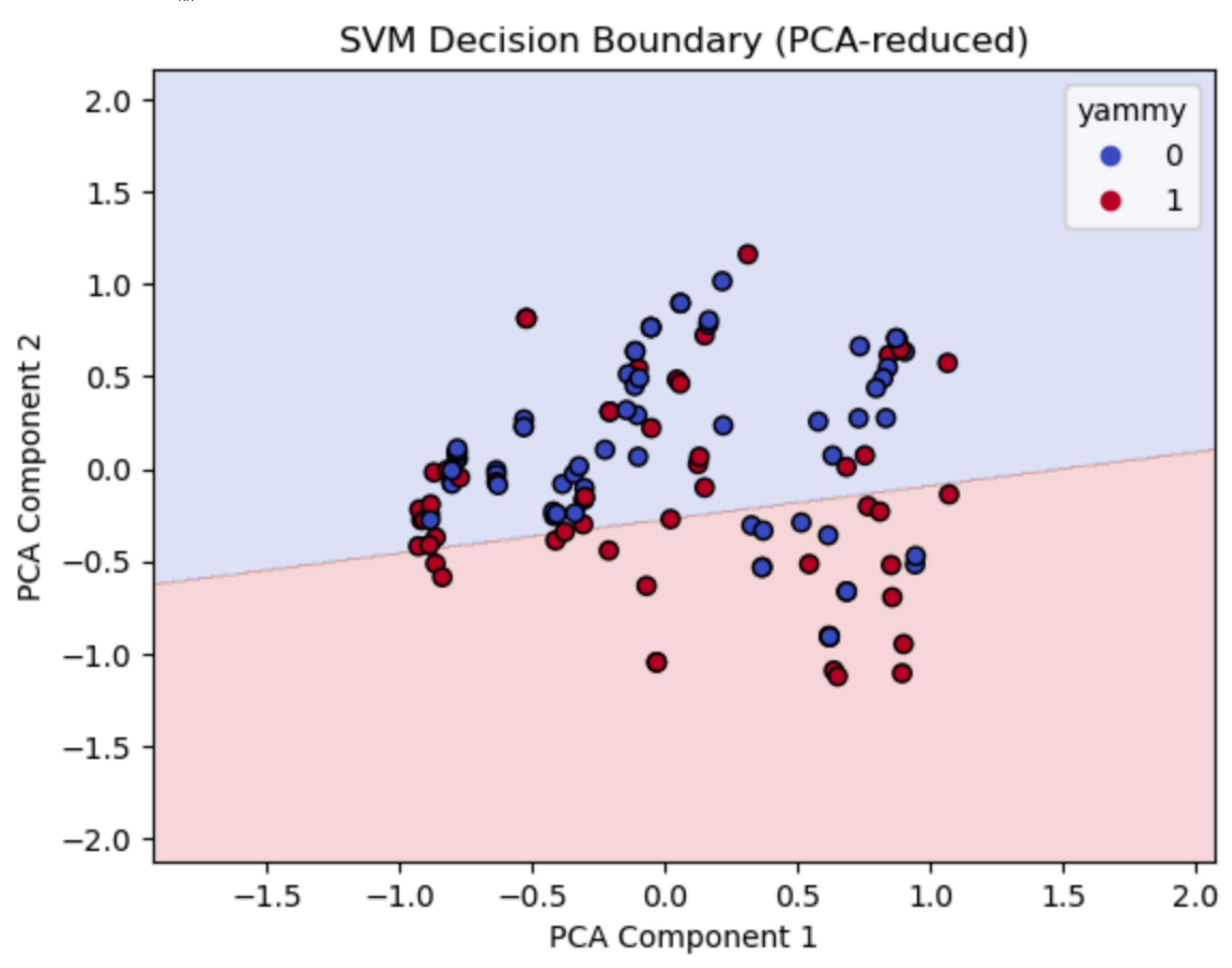
❌ It made some mistakes:

- It mislabeled 4 out of 15 meh tins as yammy

4. Visualize decision boundaries

This code visualizes the decision boundaries in 2D space.

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import numpy as np

# Reduce feature space to 2D
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)

# Train SVM on reduced 2D data
svm_2d = SVC(kernel='linear')
svm_2d.fit(X_reduced, y)

# Create meshgrid for plotting decision boundary
x_min, x_max = X_reduced[:, 0].min() - 1, X_reduced[:, 0].max() + 1
y_min, y_max = X_reduced[:, 1].min() - 1, X_reduced[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 500),
                     np.linspace(y_min, y_max, 500))

# Predict on grid
Z = svm_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot decision boundary and data
plt.contourf(xx, yy, Z, alpha=0.2, cmap=plt.cm.coolwarm)
scatter = plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('SVM Decision Boundary (PCA-reduced)')
plt.legend(*scatter.legend_elements(), title="yammy")
plt.show()
```



Made by: @katerynakononova
Website: https://katerynakononova.github.io/meowlearning/
Book: https://www.amazon.com/dp/B0CW9SFYXF