```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load the data
try:
    df = pd.read_csv('/content/cats_vs_dogs.csv')
except FileNotFoundError:
    print("Error: The file 'cats_vs_dogs.csv' was not found. Please ensure it is in the same directory as the script.")
    exit()

# 2. Select features and create the target variable
# We will classify states based on which pet is more popular in terms of household ownership.
# Create a new binary target variable 'is_more_dogs'
df['is_more_dogs'] = (df['n_dog_households'] > df['n_cat_households']).astype(int)

# Define features (X) and target (y)
X = df[['n_dog_households', 'n_cat_households']]
y = df['is_more_dogs']

# 3. Data Scaling
# It's crucial to scale the features for SVM as it is sensitive to the scale of the data.
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. Model Training
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Initialize and train the SVM classifier
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# 5. Model Evaluation
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['More Cats', 'More Dogs'], yticklabels=['More Cats', 'M
plt.title('Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()

# Visualize the decision boundary (optional but helpful for understanding)
def plot_svm_decision_boundary(X_train, y_train, model, title):
    plt.figure(figsize=(10, 8))
    ax = plt.gca()

    # Get the min and max of the features
    x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
    y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1

    # Create a mesh grid
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))

    # Predict the class for each point in the mesh grid
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot the decision boundary
    ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.3)
```

```
    # Plot the training data points
    scatter = ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.coolwarm, s=50, edgecolors='k')

    # Plot the support vectors
    ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
               s=200, facecolors='none', edgecolors='k', label='Support Vectors')

    plt.title(title)
    plt.xlabel('Scaled Number of Dog Households')
    plt.ylabel('Scaled Number of Cat Households')
    plt.legend()
    plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.60      1.00      0.75         6

    accuracy                           0.60        10
   macro avg       0.30      0.50      0.38        10
weighted avg       0.36      0.60      0.45        10


Confusion Matrix:
[[0 4]
 [0 6]]
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```