```
#svm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/cats_vs_dogs.csv')
df.head()
```

	Unnamed: 0	state	n_households	percent_pet_households	n_pet_households	percent_dog_owners	n_dog_households	avg_dogs_per_househo
0	1	Alabama	1828	59.5	1088	44.1	807	
1	2	Arizona	2515	59.5	1497	40.1	1008	
2	3	Arkansas	1148	62.4	716	47.9	550	
3	4	California	12974	52.9	6865	32.8	4260	
4	5	Colorado	1986	61.3	1217	42.5	845	

```
df.columns
```

```
Index(['Unnamed: 0', 'state', 'n_households', 'percent_pet_households',
       'n_pet_households', 'percent_dog_owners', 'n_dog_households',
       'avg_dogs_per_household', 'dog_population', 'percent_cat_owners',
       'n_cat_households', 'avg_cats_per_household', 'cat_population'],
      dtype='object')
```

```
# Define a threshold for high vs. low pet ownership
threshold = 55
df['high_pet_ownership'] = (df['percent_pet_households'] > threshold).astype(int)
```

```
df.head()
```

	Unnamed: 0	state	n_households	percent_pet_households	n_pet_households	percent_dog_owners	n_dog_households	avg_dogs_per_househo
0	1	Alabama	1828	59.5	1088	44.1	807	
1	2	Arizona	2515	59.5	1497	40.1	1008	
2	3	Arkansas	1148	62.4	716	47.9	550	
3	4	California	12974	52.9	6865	32.8	4260	
4	5	Colorado	1986	61.3	1217	42.5	845	

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report
```

```
# Select features and target
features = df.drop(columns=['Unnamed: 0', 'state', 'high_pet_ownership'])
target = df['high_pet_ownership']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

```
# Normalize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Train the SVM classifier
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train_scaled, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
svm_classifier.score(X_train,y_train) #train score
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:458: UserWarning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
0.4358974358974359
```

```
svm_classifier.score(X_test,y_test) # test score
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:458: UserWarning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
0.5
```

```
# Make predictions
y_pred = svm_classifier.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:458: UserWarning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
```

```
# Evaluate the classifier
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.44      1.00      0.62         4
           1       1.00      0.17      0.29         6

    accuracy                           0.50        10
   macro avg       0.72      0.58      0.45        10
weighted avg       0.78      0.50      0.42        10
```

Start coding or generate with AI.