

ML - Experiment 8 - SVM

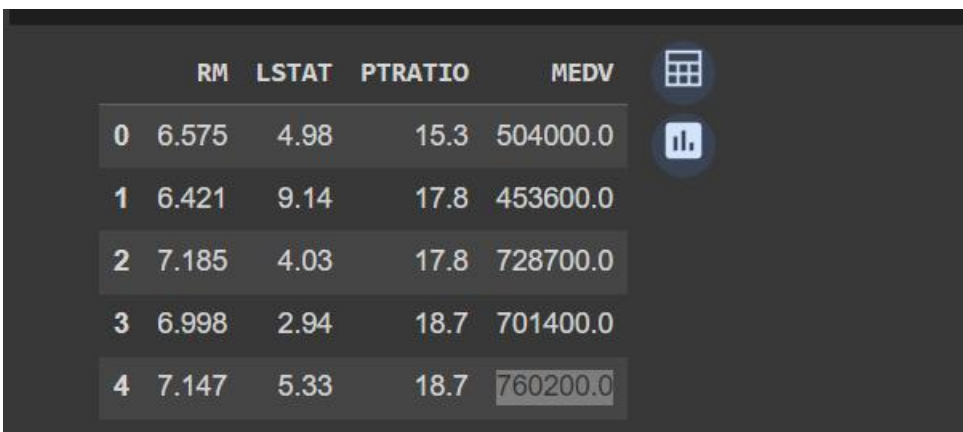
```
from google.colab import drive
drive.mount('/content/gdrive')

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
from sklearn.preprocessing import KBinsDiscretizer
```

LINEAR

DATASET

```
# Load the dataset
dataset_path = '/content/gdrive/MyDrive/ML/california.csv'
data = pd.read_csv(dataset_path)
data.head()
```



	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

```
X = data.drop(columns=['medv']) # Features
y = data['medv'] # Target variable
```

```
kbins = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='quantile')
y_discrete = kbins.fit_transform(y.values.reshape(-1, 1))
```

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

```
pca = PCA(n_components=2)  
X_pca = pca.fit_transform(X_scaled)
```

```
svm = SVC(kernel='linear')  
svm.fit(X_pca, y_discrete)
```

```
plt.figure(figsize=(8, 6))
```

```
h = .02
```

```
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
```

```
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

```
Z = svm.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
Z = Z.reshape(xx.shape)
```

```
plt.contourf(xx, yy, Z, cmap='coolwarm', alpha=0.8)
```

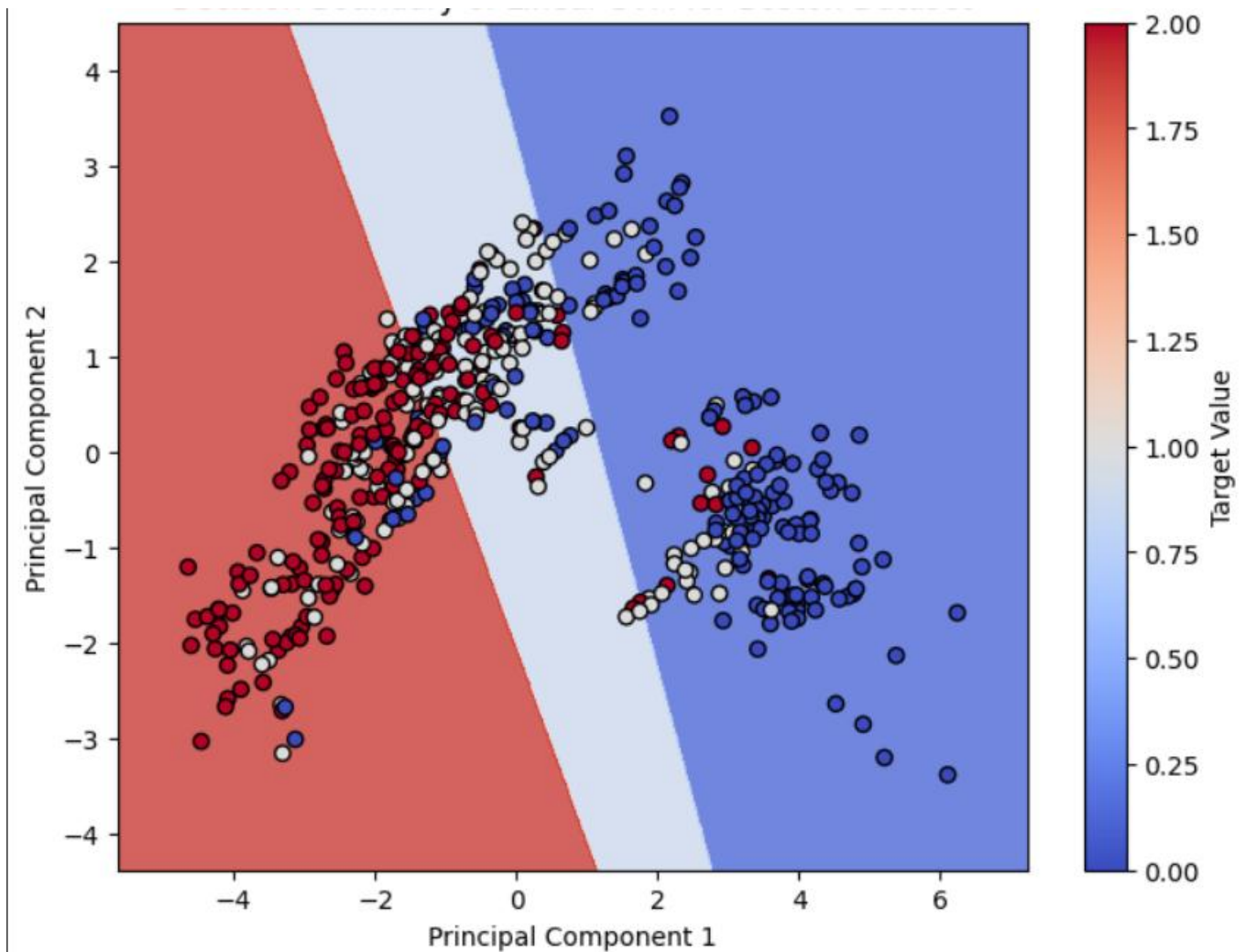
```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_discrete, cmap='coolwarm', marker='o',  
edgecolors='k')
```

```
plt.xlabel('Principal Component 1')
```

```
plt.ylabel('Principal Component 2')
```

```
plt.colorbar(label='Target Value')
```

```
plt.show()
```



```
data = pd.read_csv('/content/gdrive/MyDrive/ML/california.csv')

bins = [0, 453600.0, 760200.0, float('inf')]
labels = ['low', 'medium', 'high']
data['medv_category'] = pd.cut(data['MEDV'], bins=bins, labels=labels)

X = data.drop(columns=['MEDV', 'medv_category'])
y = data['medv_category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

kernels = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernels:

    svm = SVC(kernel=kernel)
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
```

```

confusion_mat = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Kernel: {kernel}")
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion_mat)
print("Classification Report:\n", class_report)
print()

```

OUTPUT

```

Kernel: linear
Accuracy: 0.8367346938775511
Confusion Matrix:
[[ 2  0  0]
 [ 0 52 10]
 [ 0  6 28]]
Classification Report:

```

	precision	recall	f1-score	support
high	1.00	1.00	1.00	2
low	0.90	0.84	0.87	62
medium	0.74	0.82	0.78	34
accuracy			0.84	98
macro avg	0.88	0.89	0.88	98
weighted avg	0.84	0.84	0.84	98

Kernel: poly
Accuracy: 0.8163265306122449
Confusion Matrix:
[[2 0 0]
[0 51 11]
[0 7 27]]

Classification Report:

	precision	recall	f1-score	support
high	1.00	1.00	1.00	2
low	0.88	0.82	0.85	62
medium	0.71	0.79	0.75	34
accuracy			0.82	98
macro avg	0.86	0.87	0.87	98
weighted avg	0.82	0.82	0.82	98

Kernel: rbf
Accuracy: 0.8163265306122449
Confusion Matrix:
[[0 0 2]
[0 53 9]
[0 7 27]]

Classification Report:

	precision	recall	f1-score	support
high	0.00	0.00	0.00	2
low	0.88	0.85	0.87	62
medium	0.71	0.79	0.75	34
accuracy			0.82	98
macro avg	0.53	0.55	0.54	98
weighted avg	0.81	0.82	0.81	98

Kernel: sigmoid
Accuracy: 0.6326530612244898
Confusion Matrix:
[[0 2 0]
[0 62 0]
[0 34 0]]

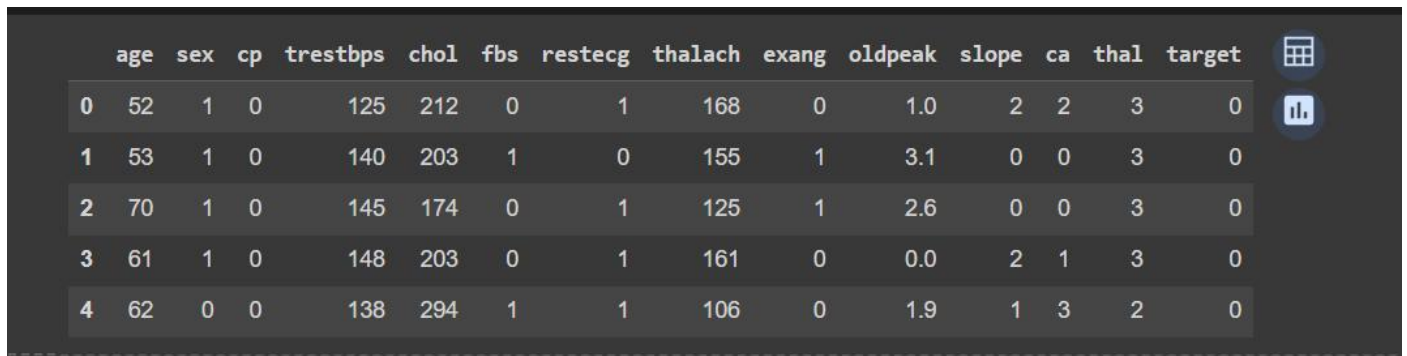
Classification Report:

	precision	recall	f1-score	support
high	0.00	0.00	0.00	2
low	0.63	1.00	0.78	62
medium	0.00	0.00	0.00	34
accuracy			0.63	98
macro avg	0.21	0.33	0.26	98
weighted avg	0.40	0.63	0.49	98

NON - LINEAR

DATASET

```
df = pd.read_csv('/content/gdrive/MyDrive/ML/heart.csv')
df.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
dataset_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data'
```

```
names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',  
         'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']
```

```
data = pd.read_csv(dataset_url, names=names)
```

```
data.replace('?', np.nan, inplace=True)
```

```
data.dropna(inplace=True)
```

```
X = data.drop(columns=['target'])
```

```
y = data['target']
```

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

```
pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X_scaled)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2,  
                                                    random_state=42)
```

```
svm = SVC(kernel='rbf', gamma='scale')
```

```
svm.fit(X_train, y_train)
```

```
plt.figure(figsize=(8, 6))
```

```
h = .02 # step size in the mesh
```

```
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
```

```
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
```

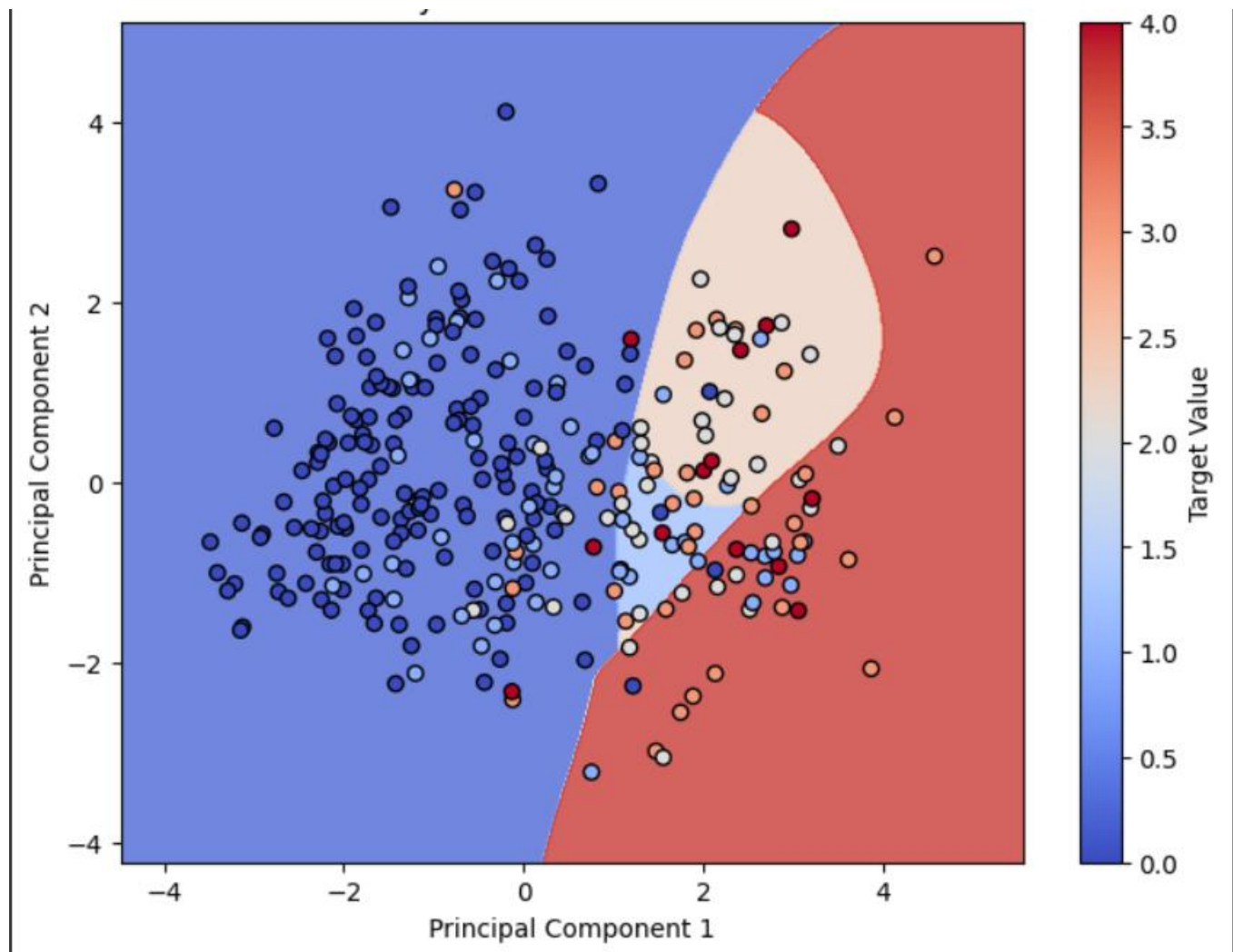
```
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

```

Z = svm.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap='coolwarm', alpha=0.8)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='coolwarm', marker='o', edgecolors='k')
plt.title('Decision Boundary of Non-linear SVM for Heart Disease')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Target Value')
plt.show()

```



```

dataset_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data'
names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
         'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']
data = pd.read_csv(dataset_url, names=names)
data.replace('?', np.nan, inplace=True)
data.dropna(inplace=True)
X = data.drop(columns=['target'])
y = data['target']

```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2,
random_state=42)
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
results = {}

for kernel in kernels:
    svm = SVC(kernel=kernel, gamma='scale')
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    results[kernel] = {'accuracy': accuracy, 'confusion_matrix': cm,
'classification_report': report}

for kernel, result in results.items():
    print(f"Kernel: {kernel}")
    print(f"Accuracy: {result['accuracy']}")
    print("Confusion Matrix:")
    print(result['confusion_matrix'])
    print("Classification Report:")
    print(result['classification_report'])
    print()
```

OUTPUT


```
Kernel: linear
Accuracy: 0.6666666666666666
Confusion Matrix:
[[35  0  1  0  0]
 [ 5  1  1  2  0]
 [ 1  1  1  2  0]
 [ 2  0  2  3  0]
 [ 2  0  0  1  0]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.97	0.86	36
1	0.50	0.11	0.18	9
2	0.20	0.20	0.20	5
3	0.38	0.43	0.40	7
4	0.00	0.00	0.00	3
accuracy			0.67	60
macro avg	0.37	0.34	0.33	60
weighted avg	0.60	0.67	0.61	60

```
Kernel: poly
Accuracy: 0.6333333333333333
Confusion Matrix:
[[35  0  1  0  0]
 [ 6  1  1  1  0]
 [ 2  2  0  1  0]
 [ 2  2  1  2  0]
 [ 2  0  0  1  0]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.97	0.84	36
1	0.20	0.11	0.14	9
2	0.00	0.00	0.00	5
3	0.40	0.29	0.33	7
4	0.00	0.00	0.00	3
accuracy			0.63	60
macro avg	0.27	0.27	0.26	60
weighted avg	0.52	0.63	0.57	60

```

Kernel: rbf
Accuracy: 0.6666666666666666
Confusion Matrix:
[[35  0  1  0  0]
 [ 5  1  1  2  0]
 [ 1  1  2  1  0]
 [ 2  1  2  2  0]
 [ 2  0  0  1  0]]
Classification Report:

```

	precision	recall	f1-score	support
0	0.78	0.97	0.86	36
1	0.33	0.11	0.17	9
2	0.33	0.40	0.36	5
3	0.33	0.29	0.31	7
4	0.00	0.00	0.00	3
accuracy			0.67	60
macro avg	0.36	0.35	0.34	60
weighted avg	0.58	0.67	0.61	60

```

Kernel: sigmoid
Accuracy: 0.6333333333333333
Confusion Matrix:
[[32  4  0  0  0]
 [ 4  3  2  0  0]
 [ 0  3  2  0  0]
 [ 3  1  2  1  0]
 [ 1  1  1  0  0]]
Classification Report:

```

	precision	recall	f1-score	support
0	0.80	0.89	0.84	36
1	0.25	0.33	0.29	9
2	0.29	0.40	0.33	5
3	1.00	0.14	0.25	7
4	0.00	0.00	0.00	3
accuracy			0.63	60
macro avg	0.47	0.35	0.34	60
weighted avg	0.66	0.63	0.61	60