```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
from sklearn.model_selection import train_test_split, cross_validate, KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import numpy as np
```

```
Mounted at /content/drive
```

```
df = pd.read_csv('/content/ObesityDataSet_raw_and_data_sinthetic.csv')
print("First five rows of the dataset:")
print(df.head())
```

```
First five rows of the dataset:
   Gender   Age  Height  Weight family_history_with_overweight FAVC  FCVC  \
0  Female  21.0    1.62    64.0                            yes   no   2.0
1  Female  21.0    1.52    56.0                            yes   no   3.0
2    Male  23.0    1.80    77.0                            yes   no   2.0
3    Male  27.0    1.80    87.0                             no   no   3.0
4    Male  22.0    1.78    89.8                             no   no   2.0

   NCP       CAEC SMOKE  CH2O  SCC  FAF  TUE        CALC  \
0  3.0  Sometimes    no   2.0   no  0.0  1.0          no
1  3.0  Sometimes   yes   3.0  yes  3.0  0.0   Sometimes
2  3.0  Sometimes    no   2.0   no  2.0  1.0  Frequently
3  3.0  Sometimes    no   2.0   no  2.0  0.0  Frequently
4  1.0  Sometimes    no   2.0   no  0.0  0.0   Sometimes

                 MTRANS           NObeyesdad
0  Public_Transportation        Normal_Weight
1  Public_Transportation        Normal_Weight
2  Public_Transportation        Normal_Weight
3               Walking   Overweight_Level_I
4  Public_Transportation  Overweight_Level_II
```

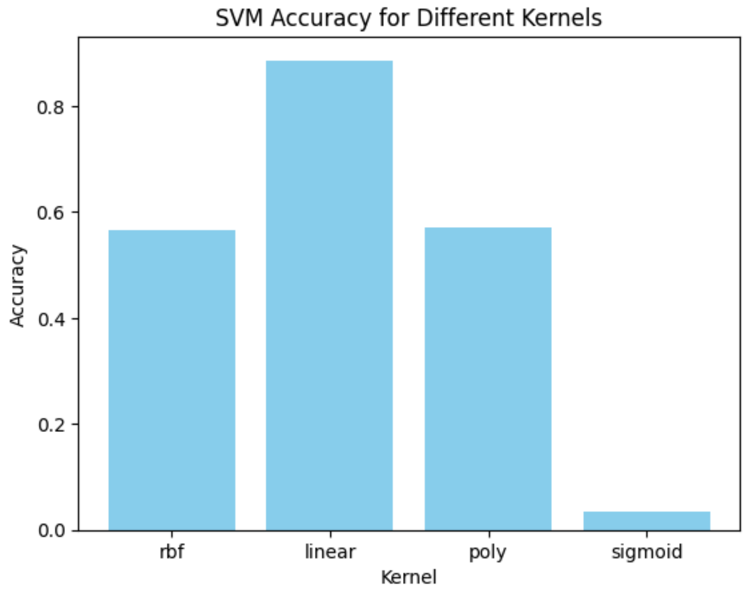
```
X = df.drop("NObeyesdad", axis=1)
y = df["NObeyesdad"]
le = LabelEncoder()
for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = le.fit_transform(X[col])
y = le.fit_transform(y)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
kernels = {
    'rbf': {'C': 1.0, 'gamma': 'scale'},
    'linear': {'C': 1.0},
    'poly': {'C': 1.0, 'degree': 3, 'gamma': 'scale'},
    'sigmoid': {'C': 1.0, 'gamma': 'scale'}
}

results = {}

for kernel, params in kernels.items():
    print(f"\n========== Kernel: {kernel} ==========")

    # Build the SVM model
    model = SVC(kernel=kernel, **params)

    # Train the model
    model.fit(X_train, y_train)

    # Testing and evaluation
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='macro')
    rec = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')
```

```
    print("Confusion Matrix:\n", cm)
    print(f"Accuracy:  {acc:.4f}")
    print(f"Precision: {prec:.4f}")
    print(f"Recall:    {rec:.4f}")
    print(f"F1-Score:  {f1:.4f}")

    results[kernel] = acc
```

```
========== Kernel: rbf ==========
Confusion Matrix:
 [[49  7  0  0  0  0  0]
 [18 21  0  0  0 20  3]
 [ 0  0 26  7 21  1 23]
 [ 0  0  5 24 29  0  0]
 [ 0  0  0  0 63  0  0]
 [ 2 14  0  0  0 27 13]
 [ 0  2  9  0  0 10 29]]
Accuracy:  0.5650
Precision: 0.5802
Recall:    0.5747
F1-Score:  0.5488

========== Kernel: linear ==========
Confusion Matrix:
 [[56  0  0  0  0  0  0]
 [10 40  0  0  0  8  4]
 [ 0  0 73  4  0  0  1]
 [ 0  0  0 58  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 0  3  0  0  0 49  4]
 [ 0  0  6  0  0  8 36]]
Accuracy:  0.8865
Precision: 0.8846
Recall:    0.8823
F1-Score:  0.8778

========== Kernel: poly ==========
Confusion Matrix:
 [[50  6  0  0  0  0  0]
 [21 22  0  0  0 16  3]
 [ 0  0 36  7  8  2 25]
 [ 0  0  6 22 30  0  0]
 [ 0  0  0  0 63  0  0]
 [ 3 15  0  0  0 27 11]
 [ 0  3  8  0  0 17 22]]
Accuracy:  0.5721
Precision: 0.5789
Recall:    0.5730
F1-Score:  0.5525

========== Kernel: sigmoid ==========
Confusion Matrix:
 [[ 5  0  0  0 51  0  0]
 [ 0  0  0  0 62  0  0]
 [ 0  7  0  0 67  4  0]
 [ 0 46  0  0 12  0  0]
 [ 0 47  0  0 10  6  0]
 [ 0  0  0  0 56  0  0]
 [ 0  0  0  0 50  0  0]]
Accuracy:  0.0355
Precision: 0.1475
Recall:    0.0354
F1-Score:  0.0311
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

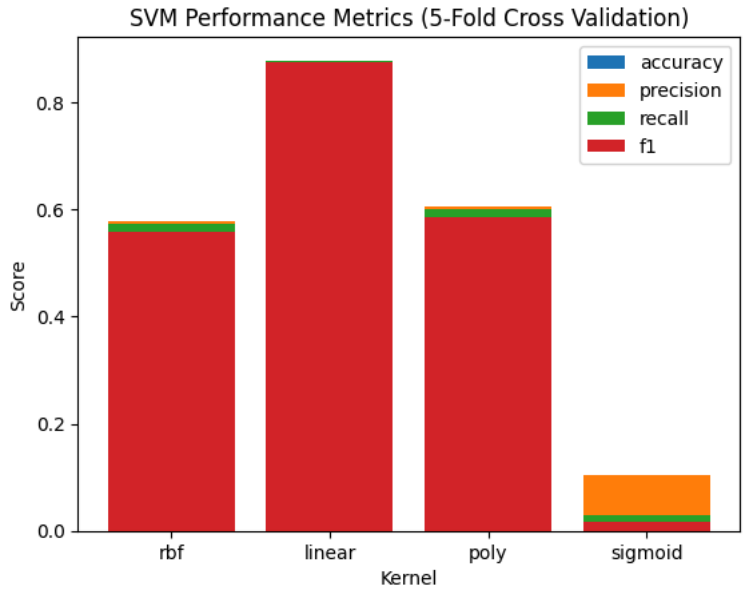
```
# Plot accuracy for all kernels
plt.bar(results.keys(), results.values(), color='skyblue')
plt.title("SVM Accuracy for Different Kernels")
plt.xlabel("Kernel")
plt.ylabel("Accuracy")
plt.show()
```




```
# Perform 5-Fold Cross Validation
cv = KFold(n_splits=5, shuffle=True, random_state=42)
cv_results = {}

for kernel, params in kernels.items():
    model = SVC(kernel=kernel, **params)
    scores = cross_validate(
        model, X, y, cv=cv,
        scoring=['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']
    )
    cv_results[kernel] = {
        'accuracy': np.mean(scores['test_accuracy']),
        'precision': np.mean(scores['test_precision_macro']),
        'recall': np.mean(scores['test_recall_macro']),
        'f1': np.mean(scores['test_f1_macro'])
    }

# Print CV results
print("\n===== 5-Fold Cross Validation Results =====")
for k, v in cv_results.items():
    print(f"{k.upper()}: Accuracy={v['accuracy']:.4f}, Precision={v['precision']:.4f}, "
          f"Recall={v['recall']:.4f}, F1={v['f1']:.4f}")
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-de
  _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 ill-de
  _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 ill-de
  _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 ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

===== 5-Fold Cross Validation Results =====
RBF: Accuracy=0.5609, Precision=0.5788, Recall=0.5740, F1=0.5574
LINEAR: Accuracy=0.8787, Precision=0.8786, Recall=0.8774, F1=0.8746
POLY: Accuracy=0.5974, Precision=0.6057, Recall=0.6005, F1=0.5858
SIGMOID: Accuracy=0.0303, Precision=0.1051, Recall=0.0291, F1=0.0162
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Plot CV metrics
metrics = ['accuracy', 'precision', 'recall', 'f1']
for m in metrics:
    plt.bar(cv_results.keys(), [cv_results[k][m] for k in kernels], label=m)
plt.title("SVM Performance Metrics (5-Fold Cross Validation)")
plt.xlabel("Kernel")
plt.ylabel("Score")
plt.legend()
plt.show()
```


SVM Performance Metrics (5-Fold Cross Validation)
accuracy
precision
recall
f1
Score
0.0
0.2
0.4
0.6
0.8
rbf
linear
poly
sigmoid
Kernel