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

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

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

```
file_path='/content/drive/My Drive/machine
learning/ObesityDataSet_raw_and_data_sinthetic.csv'
df=pd.read_csv(file_path)
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2111,\n  \"fields\":
[\n    {\n      \"column\": \"Gender\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Male\",\n          \"Female\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 6.3459682737322405,\
n        \"min\": 14.0,\n        \"max\": 61.0,\n
\"num_unique_values\": 1402,\n        \"samples\": [\n
25.526746,\n          26.740655\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Height\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0.09330481986792,\
n        \"min\": 1.45,\n        \"max\": 1.98,\n
\"num_unique_values\": 1574,\n        \"samples\": [\n
1.760175,\n          1.688436\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Weight\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 26.191171745204688,\n        \"min\":
39.0,\n        \"max\": 173.0,\n        \"num_unique_values\": 1525,\n
\"samples\": [\n          120.702935,\n          64.4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"family_history_with_overweight\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"no\",\n
\"yes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"FAVC\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"yes\",\
n          \"no\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"FCVC\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
```

```
\"std\": 0.5339265785033023,\n        \"min\": 1.0,\n        \"max\":
3.0,\n        \"num_unique_values\": 810,\n        \"samples\": [\n
2.987148,\n          2.939727\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"NCP\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.7780386488418594,\n        \"min\":
1.0,\n        \"max\": 4.0,\n        \"num_unique_values\": 635,\n
\"samples\": [\n          1.468948,\n          2.9948\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CAEC\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Frequently\",\n          \"no\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SMOKE\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"yes\",\n          \"no\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"CH2O\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.6129534517968702,\n        \"min\":
1.0,\n        \"max\": 3.0,\n        \"num_unique_values\": 1268,\n
\"samples\": [\n          2.395387,\n          1.983973\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SCC\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"yes\",\n          \"no\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"FAF\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.8505924308367011,\n
\"min\": 0.0,\n        \"max\": 3.0,\n        \"num_unique_values\":
1190,\n        \"samples\": [\n          1.655488,\n
2.433918\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TUE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.6089272596763761,\n        \"min\": 0.0,\n        \"max\":
2.0,\n        \"num_unique_values\": 1129,\n        \"samples\": [\n
1.416353,\n          0.878258\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"CALC\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Sometimes\",\n          \"Always\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MTRANS\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
5,\n        \"samples\": [\n          \"Walking\",\n
\"Bike\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NObeyesdad\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 7,\n        \"samples\":
```

```
[\n          \"Normal_Weight\",\n          \"Overweight_Level_I\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

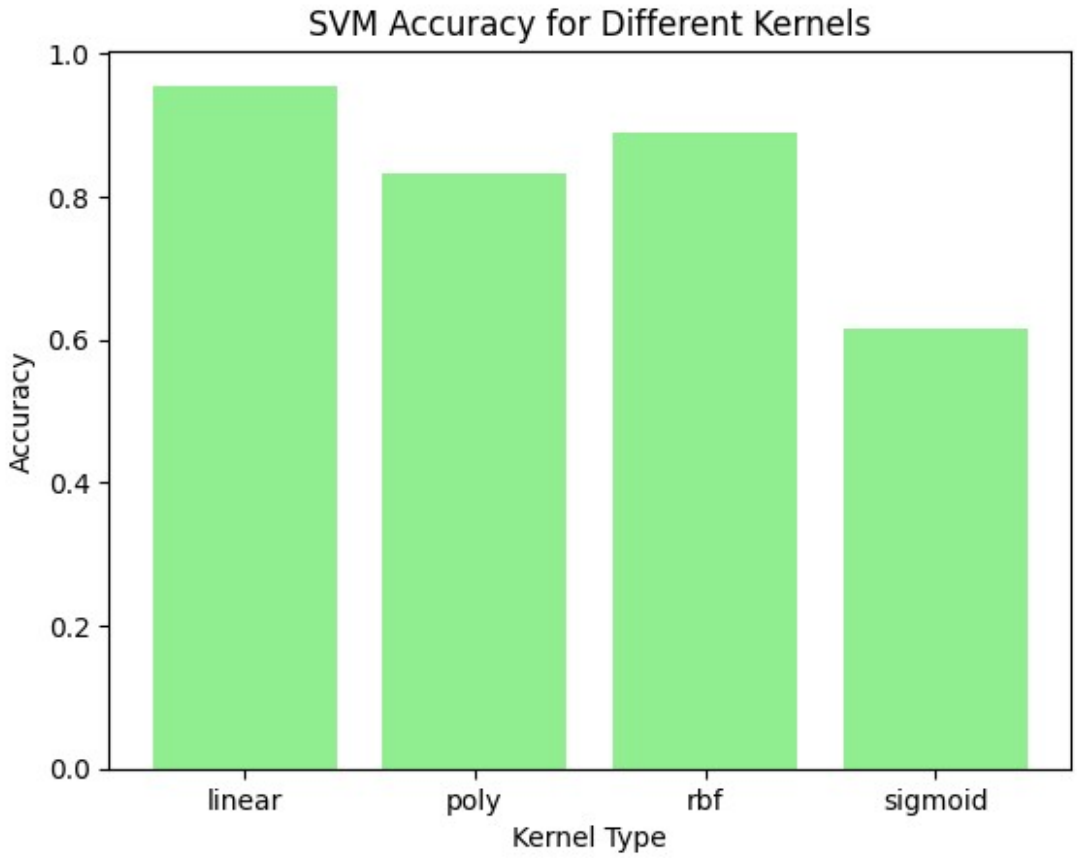
```
X = df.drop("NObeyesdad", axis=1)
y = df["NObeyesdad"]

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)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

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

for kernel in kernels:
    print("\n==============================")
    print("Using Kernel:", kernel)
    print("==============================")
    model = SVC(kernel=kernel, C=1.0, gamma='scale')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)
    print("Accuracy:", acc)
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test,
y_pred))
```

```
==============================
Using Kernel: linear
==============================
Accuracy: 0.9550827423167849
Confusion Matrix:
 [[56  0  0  0  0  0  0]
 [ 5 53  0  0  0  4  0]
 [ 0  0 75  2  0  0  1]
 [ 0  0  1 57  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 0  2  0  0  0 52  2]
 [ 0  0  0  0  0  2 48]]
Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        56
           1       0.96      0.85      0.91        62
           2       0.99      0.96      0.97        78
           3       0.97      0.98      0.97        58
           4       1.00      1.00      1.00        63
           5       0.90      0.93      0.91        56
           6       0.94      0.96      0.95        50

    accuracy                           0.96       423
   macro avg       0.95      0.96      0.95       423
weighted avg       0.96      0.96      0.95       423


==============================
Using Kernel: poly
==============================
Accuracy: 0.8321513002364066
Confusion Matrix:
 [[54  2  0  0  0  0  0]
 [ 5 34  4  1  0  9  9]
 [ 0  0 73  2  0  1  2]
 [ 0  0  2 56  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 1  6  4  1  0 41  3]
 [ 0  3 11  0  0  5 31]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.96      0.93        56
           1       0.76      0.55      0.64        62
           2       0.78      0.94      0.85        78
           3       0.93      0.97      0.95        58
           4       1.00      1.00      1.00        63
           5       0.73      0.73      0.73        56
           6       0.69      0.62      0.65        50

    accuracy                           0.83       423
   macro avg       0.83      0.82      0.82       423
weighted avg       0.83      0.83      0.83       423


==============================
Using Kernel: rbf
==============================
Accuracy: 0.8888888888888888
Confusion Matrix:
 [[52  4  0  0  0  0  0]
 [ 3 52  1  0  0  4  2]
```

```
 [ 0  0 70  4  0  2  2]
 [ 0  0  1 57  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 1 12  0  0  0 41  2]
 [ 0  2  2  0  0  5 41]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        56
           1       0.74      0.84      0.79        62
           2       0.95      0.90      0.92        78
           3       0.93      0.98      0.96        58
           4       1.00      1.00      1.00        63
           5       0.79      0.73      0.76        56
           6       0.87      0.82      0.85        50

    accuracy                           0.89       423
   macro avg       0.89      0.89      0.89       423
weighted avg       0.89      0.89      0.89       423


==============================
Using Kernel: sigmoid
==============================
Accuracy: 0.6146572104018913
Confusion Matrix:
 [[32 20  0  0  0  4  0]
 [28 19  0  0  0  8  7]
 [ 0  1 46 13  0  2 16]
 [ 0  0 12 46  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 9  7  2  0  0 29  9]
 [ 3  5  8  2  0  7 25]]
Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.57      0.50        56
           1       0.37      0.31      0.33        62
           2       0.68      0.59      0.63        78
           3       0.75      0.79      0.77        58
           4       1.00      1.00      1.00        63
           5       0.58      0.52      0.55        56
           6       0.44      0.50      0.47        50

    accuracy                           0.61       423
   macro avg       0.61      0.61      0.61       423
weighted avg       0.62      0.61      0.61       423
```

```
plt.bar(kernels, accuracies, color='lightgreen')
plt.xlabel('Kernel Type')
plt.ylabel('Accuracy')
plt.title('SVM Accuracy for Different Kernels')
plt.show()
```




```
print("\n===== 5-Fold Cross Validation Results =====")
for kernel in kernels:
    model = SVC(kernel=kernel, C=1.0, gamma='scale')
    scores = cross_val_score(model, X, y, cv=5)
    print("\nKernel:", kernel)
    print("Fold Accuracies:", scores)
    print("Mean Accuracy:", scores.mean())
```

```
===== 5-Fold Cross Validation Results =====

Kernel: linear
Fold Accuracies: [0.74704492 0.8957346  0.89336493 0.91469194
0.92417062]
Mean Accuracy: 0.8750014005131479

Kernel: poly
```

```
Fold Accuracies: [0.59338061 0.59952607 0.62796209 0.54265403
0.61848341]
Mean Accuracy: 0.5964012414148544

Kernel: rbf
Fold Accuracies: [0.61465721 0.62322275 0.56635071 0.50947867
0.57345972]
Mean Accuracy: 0.5774338117486246

Kernel: sigmoid
Fold Accuracies: [0.04728132 0.02843602 0.05450237 0.05450237
0.01184834]
Mean Accuracy: 0.039314084680626984
```