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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
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 accuracy_score, precision_score, recall_score, f1_score

# Load the HCV dataset from UCI repository or replace with your desired dataset
df = pd.read_csv('/content/drive/MyDrive/Data/hcvdat0.csv')

df.head(10)
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9
5	6	0=Blood Donor	32	m	41.6	43.3	18.5	19.7	12.3	9.92	6.05	111.0	91.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 615 entries, 0 to 614
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  615 non-null    int64  
 1   Category    615 non-null    object 
 2   Age         615 non-null    int64  
 3   Sex         615 non-null    object 
 4   ALB         614 non-null    float64
 5   ALP         597 non-null    float64
 6   ALT         614 non-null    float64
 7   AST         615 non-null    float64
 8   BIL         615 non-null    float64
 9   CHE         615 non-null    float64
 10  CHOL        605 non-null    float64
 11  CREA        615 non-null    float64
 12  GGT         615 non-null    float64
 13  PROT        614 non-null    float64
```

```
dtypes: float64(10), int64(2), object(2)
memory usage: 67.4+ KB
```

```
from sklearn.preprocessing import LabelEncoder

# Encode the 'Sex' column using label encoding
sex_encoder = LabelEncoder()
df['Sex'] = sex_encoder.fit_transform(df['Sex'])

# Extract the numeric values from the 'Category' column
df['Category'] = df['Category'].str.extract('(\d+)').astype(int)

# Encode the 'Category' column using label encoding
category_encoder = LabelEncoder()
df['Category'] = category_encoder.fit_transform(df['Category'])

df.head()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT
0	1	0	32	1	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1
1	2	0	32	1	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6
2	3	0	32	1	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2
3	4	0	32	1	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8

```
# Check for missing values
missing_values = df.isnull().sum()
print(missing_values)
```

```
Unnamed: 0     0
Category       0
Age            0
Sex            0
ALB            1
ALP           18
ALT            1
AST            0
BIL            0
CHE            0
CHOL          10
CREA           0
GGT            0
PROT           1
dtype: int64
```

```
# Drop rows with missing values
df.dropna(inplace=True)
```

```
# Check for missing values
missing_values = df.isnull().sum()
print(missing_values)
```

```
Unnamed: 0    0
Category      0
Age           0
Sex           0
ALB           0
ALP           0
ALT           0
AST           0
BIL           0
CHE           0
CHOL          0
CREA          0
GGT           0
PROT          0
dtype: int64
```

```
# Preprocess the data
X = df.drop('Category', axis=1)
y = df['Category']
print("Dataset shape:", X.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=36)
print("Training set shape:", X_train.shape)
print("Test set shape:", X_test.shape)
```

```
Dataset shape: (589, 13)
Training set shape: (471, 13)
Test set shape: (118, 13)
```

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

```
# Train SVM models with different kernels
svm_linear = SVC(kernel='linear')
svm_poly = SVC(kernel='poly')
svm_rbf = SVC(kernel='rbf')

svm_linear.fit(X_train, y_train)
svm_poly.fit(X_train, y_train)
svm_rbf.fit(X_train, y_train)

# Make predictions on the test set
y_pred_linear = svm_linear.predict(X_test)
y_pred_poly = svm_poly.predict(X_test)
y_pred_rbf = svm_rbf.predict(X_test)

# Evaluate model performance
accuracy_linear = accuracy_score(y_test, y_pred_linear)
precision_linear = precision_score(y_test, y_pred_linear, average='weighted')
recall_linear = recall_score(y_test, y_pred_linear, average='weighted')
f1_linear = f1_score(y_test, y_pred_linear, average='weighted')

accuracy_poly = accuracy_score(y_test, y_pred_poly)
```

```
precision_poly = precision_score(y_test, y_pred_poly, average='weighted')
recall_poly = recall_score(y_test, y_pred_poly, average='weighted')
f1_poly = f1_score(y_test, y_pred_poly, average='weighted')

accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
precision_rbf = precision_score(y_test, y_pred_rbf, average='weighted')
recall_rbf = recall_score(y_test, y_pred_rbf, average='weighted')
f1_rbf = f1_score(y_test, y_pred_rbf, average='weighted')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samp
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.model_selection import cross_val_score

# Perform 10-fold cross-validation
scores_linear = cross_val_score(svm_linear, X, y, cv=10)
scores_poly = cross_val_score(svm_poly, X, y, cv=10)
scores_rbf = cross_val_score(svm_rbf, X, y, cv=10)

# Perform 5-fold cross-validation
scores_linear_5fold = cross_val_score(svm_linear, X, y, cv=5)
scores_poly_5fold = cross_val_score(svm_poly, X, y, cv=5)
scores_rbf_5fold = cross_val_score(svm_rbf, X, y, cv=5)

# Interpret the cross-validation results
mean_score_linear = scores_linear.mean()
mean_score_poly = scores_poly.mean()
mean_score_rbf = scores_rbf.mean()

print("Accuracy - Linear Kernel:", accuracy_linear)
print("Accuracy - Polynomial Kernel:", accuracy_poly)
print("Accuracy - RBF Kernel:", accuracy_rbf)
```

```
Accuracy - Linear Kernel: 0.9830508474576272
Accuracy - Polynomial Kernel: 0.9661016949152542
Accuracy - RBF Kernel: 0.8898305084745762
```

✓ 0s completed at 10:31 AM