```
import numpy as np
import pandas as pd
```

```
df=pd.read_csv('samples_cancer .csv')
df.head()
```

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

Next steps: Generate code with df | New interactive sheet

```
df['BareNuc'] = pd.to_numeric(df['BareNuc'], errors='coerce')
df = df.dropna().reset_index(drop=True)
X = df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize',
        'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']].values
y = df['Class'].map({2:0, 4:1}).values
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

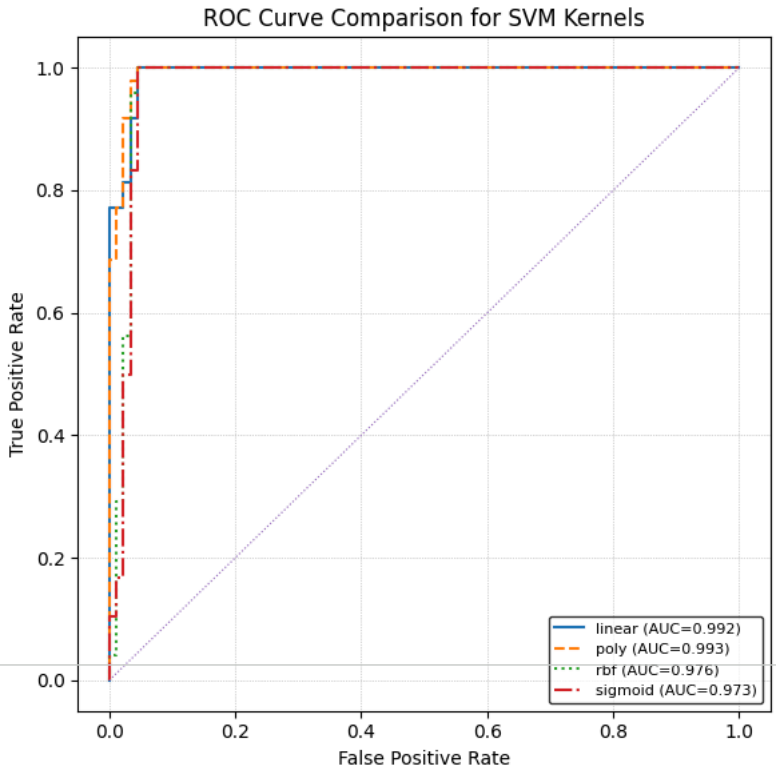
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, jaccard_score, confusion_matrix, roc_c

kernels = ['linear','poly','rbf','sigmoid']
results = {}
for kern in kernels:
    model = SVC(kernel=kern, probability=True, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_score = model.predict_proba(X_test)[:,1]
    acc = accuracy_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    jacc = jaccard_score(y_test, y_pred)
    err = 1 - acc
    cm = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)
    results[kern] = {
        'model': model,
        'accuracy': acc,
        'recall': rec,
        'precision': prec,
        'f1': f1,
        'jaccard': jacc,
        'error_rate': err,
        'confusion_matrix': cm,
        'fpr': fpr,
        'tpr': tpr,
        'auc': roc_auc
    }
```

```
for kern in kernels:
    r = results[kern]
    print(f"SVM kernel: {kern}")
    print(f"Accuracy : {r['accuracy']:.4f}")
    print(f"Recall   : {r['recall']:.4f}")
    print(f"Precision: {r['precision']:.4f}")
```

```
    print(f"F1-Score : {r['f1']:.4f}")
    print(f"Jaccard  : {r['jaccard']:.4f}")
    print(f"Error Rt : {r['error_rate']:.4f}")
    tn, fp, fn, tp = r['confusion_matrix'].ravel()
    print("CONFUSION MATRIX (rows=Actual, cols=Predicted)")
    print(f"       Pred:0   Pred:1")
    print(f"Act:0   {tn:4d}     {fp:4d}")
    print(f"Act:1   {fn:4d}     {tp:4d}")
    print(f"AUC      : {r['auc']:.4f}")
    print("-"*40)
```

```
SVM kernel: linear
Accuracy : 0.9562
Recall   : 0.9583
Precision: 0.9200
F1-Score : 0.9388
Jaccard  : 0.8846
Error Rt : 0.0438
CONFUSION MATRIX (rows=Actual, cols=Predicted)
       Pred:0   Pred:1
Act:0     85        4
Act:1      2       46
AUC      : 0.9918
----------------------------------------
SVM kernel: poly
Accuracy : 0.9562
Recall   : 0.9167
Precision: 0.9565
F1-Score : 0.9362
Jaccard  : 0.8800
Error Rt : 0.0438
CONFUSION MATRIX (rows=Actual, cols=Predicted)
       Pred:0   Pred:1
Act:0     87        2
Act:1      4       44
AUC      : 0.9927
----------------------------------------
SVM kernel: rbf
Accuracy : 0.9635
Recall   : 0.9792
Precision: 0.9216
F1-Score : 0.9495
Jaccard  : 0.9038
Error Rt : 0.0365
CONFUSION MATRIX (rows=Actual, cols=Predicted)
       Pred:0   Pred:1
Act:0     85        4
Act:1      1       47
AUC      : 0.9759
----------------------------------------
SVM kernel: sigmoid
Accuracy : 0.9416
Recall   : 0.9167
Precision: 0.9167
F1-Score : 0.9167
Jaccard  : 0.8462
Error Rt : 0.0584
CONFUSION MATRIX (rows=Actual, cols=Predicted)
       Pred:0   Pred:1
Act:0     85        4
Act:1      4       44
AUC      : 0.9731
----------------------------------------
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6))
linestyles = {'linear':'-','poly':'--','rbf':':','sigmoid':'-.'}
for kern in kernels:
    r = results[kern]
    plt.plot(r['fpr'], r['tpr'], linestyle=linestyles[kern], linewidth=1.5, label=f"{kern} (AUC={r['auc']:.3f})")
plt.plot([0,1],[0,1], linestyle=':', linewidth=0.8)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison for SVM Kernels')
plt.legend(edgecolor='black', fontsize=8)
plt.grid(True, linestyle=':', linewidth=0.5)
plt.tight_layout()
plt.show()
```


ROC Curve Comparison for SVM Kernels
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
linear (AUC=0.992)
poly (AUC=0.993)
rbf (AUC=0.976)
sigmoid (AUC=0.973)