```
# 🧠 SVM Cancer Prediction (Linear, Poly, RBF, Sigmoid) with Metrics + ROC Graphs
import io, pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns
from google.colab import files
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import (accuracy_score, recall_score, precision_score, f1_score,
                             jaccard_score, confusion_matrix, roc_curve, auc)

# ---------- Load CSV ----------
print("📁 Upload your CSV file (e.g. samples_cancer (1).csv)")
uploaded = files.upload()
fname = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[fname]))

# ---------- Data Prep ----------
# Detect target column (diagnosis/label/class/target or last)
for cand in ("diagnosis","target","class","label"):
    if cand in df.columns:
        target_col = cand
        break
else:
    target_col = df.columns[-1]

X = df.drop(columns=[target_col])
y_raw = df[target_col]

# Encode target and features
le = LabelEncoder()
y = le.fit_transform(y_raw.astype(str))
for c in X.select_dtypes(include=['object','category']).columns:
    X[c] = LabelEncoder().fit_transform(X[c].astype(str))
X = X.fillna(0)
X = StandardScaler().fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)

# ---------- Train SVM Models ----------
kernels = ['linear','poly','rbf','sigmoid']
results, models = [], {}

for k in kernels:
    clf = SVC(kernel=k, probability=True, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    try:
        y_score = clf.predict_proba(X_test)[:,1]
    except Exception:
        y_score = clf.decision_function(X_test)
        y_score = (y_score - y_score.min()) / (y_score.max() - y_score.min() + 1e-12)

    acc = accuracy_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, zero_division=0)
    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)

    models[k] = {'fpr': fpr, 'tpr': tpr, 'auc': roc_auc}
    results.append({'Kernel': k, 'Accuracy': acc, 'Recall': rec, 'Precision': prec,
                    'F1-Score': f1, 'Jaccard': jacc, 'Error Rate': err, 'ConfusionMatrix': cm})

# ---------- Results Table ----------
res_df = pd.DataFrame(results).set_index('Kernel')
print("\n=== Metrics Table ===")
display(res_df[['Accuracy','Recall','Precision','F1-Score','Jaccard','Error Rate']])

# ---------- Confusion Matrices ----------
fig, axes = plt.subplots(1, 4, figsize=(16, 4))
for ax, k in zip(axes, kernels):
    sns.heatmap(results[kernels.index(k)]['ConfusionMatrix'], annot=True, fmt='d', cmap='Blues', ax=ax)
    ax.set_title(k)
    ax.set_xlabel("Predicted")
    ax.set_ylabel("Actual")
plt.suptitle("Confusion Matrices")
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

```
plt.show()

# ---------- ROC Curve ----------
plt.figure(figsize=(8,6))
for k in kernels:
    plt.plot(models[k]['fpr'], models[k]['tpr'], label=f"{k} (AUC={models[k]['auc']:.3f})")
plt.plot([0,1],[0,1],'--',color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison - SVM Kernels')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

Upload your CSV file (e.g. samples_cancer (1).csv)
Choose Files samples_cancer (1).csv
**samples_cancer (1).csv**(text/csv) - 20675 bytes, last modified: 11/8/2025 - 100% done
Saving samples_cancer (1).csv to samples_cancer (1).csv

=== Metrics Table ===

| Kernel | Accuracy | Recall | Precision | F1-Score | Jaccard | Error Rate |
|---|---|---|---|---|---|---|
| **linear** | 0.965714 | 0.950000 | 0.950000 | 0.950000 | 0.904762 | 0.034286 |
| **poly** | 0.914286 | 0.750000 | 1.000000 | 0.857143 | 0.750000 | 0.085714 |
| **rbf** | 0.954286 | 0.966667 | 0.906250 | 0.935484 | 0.878788 | 0.045714 |
| **sigmoid** | 0.965714 | 0.966667 | 0.935484 | 0.950820 | 0.906250 | 0.034286 |