```
import io, pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (accuracy_score, recall_score, precision_score, f1_score,
                             confusion_matrix, roc_curve, auc, classification_report)
```

```
print("📁 Upload your CSV file (e.g. pima-indians-diabetes.data.csv)")
uploaded = files.upload()
fname = list(uploaded.keys())[0]
```

```
📁 Upload your CSV file (e.g. pima-indians-diabetes.data.csv)
Choose Files pima-indian...tes.data.csv
pima-indians-diabetes.data.csv(text/csv) - 23279 bytes, last modified: 11/8/2025 - 100% done
Saving pima-indians-diabetes.data.csv to pima-indians-diabetes.data.csv
```

```
try:
    df = pd.read_csv(io.BytesIO(uploaded[fname]))
except Exception:
    df = pd.read_csv(io.BytesIO(uploaded[fname]), header=None)
```

```
if len(df.columns) == 9:
    df.columns = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin',
                  'BMI','DiabetesPedigreeFunction','Age','Outcome']
elif len(df.columns) > 1 and not any(c.lower() in ('outcome','target','class') for c in df.columns):
    df.columns = [f'col{i}' for i in range(df.shape[1]-1)] + ['Outcome']

# ---- Split features/target ----
target_col = [c for c in df.columns if c.lower() in ('outcome','target','class')][-1]
X = df.drop(columns=[target_col])
y = df[target_col]
if y.dtype == 'object':
    y = LabelEncoder().fit_transform(y)
```

```
X = X.fillna(0)
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)
```

```
models = {
    'SVM': SVC(kernel='rbf', probability=True, random_state=42),
    'Naive Bayes': GaussianNB(),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'KNN': KNeighborsClassifier(n_neighbors=5)
}

results, roc_data = [], {}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_score = model.predict_proba(X_test)[:,1] if hasattr(model,"predict_proba") else y_pred

    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)
    cm = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)

    results.append({'Model': name, 'Accuracy': acc, 'Recall': rec, 'Precision': prec, 'F1-Score': f1})
    roc_data[name] = {'fpr': fpr, 'tpr': tpr, 'auc': roc_auc, 'cm': cm}
```

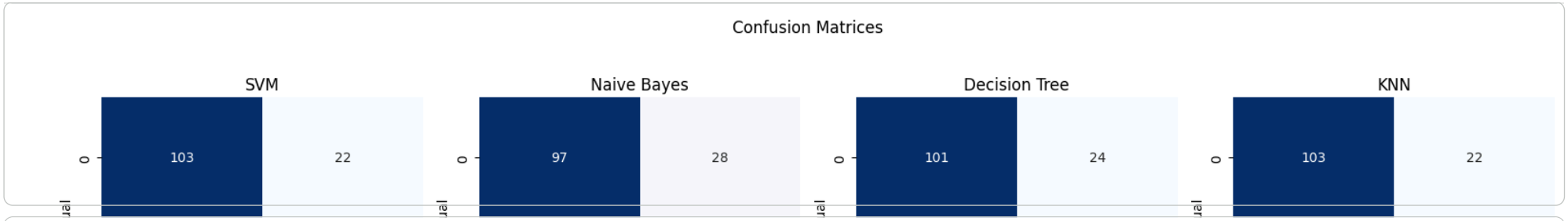
```
results_df = pd.DataFrame(results).set_index('Model')
print("\n=== Performance Metrics ===")
display(results_df)
```

```
=== Performance Metrics ===
```

Model	Accuracy	Recall	Precision	F1-Score
SVM	0.713542	0.507463	0.607143	0.552846
Naive Bayes	0.713542	0.597015	0.588235	0.592593
Decision Tree	0.708333	0.522388	0.593220	0.555556
KNN	0.739583	0.582090	0.639344	0.609375

Next steps: Generate code with `results_df` New interactive sheet

```
fig, axes = plt.subplots(1, 4, figsize=(16,4))
for ax, (name, data) in zip(axes, roc_data.items()):
    sns.heatmap(data['cm'], annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False)
    ax.set_title(name)
    ax.set_xlabel('Predicted'); ax.set_ylabel('Actual')
plt.suptitle('Confusion Matrices')
plt.tight_layout(rect=[0,0,1,0.95])
plt.show()
```




```
plt.figure(figsize=(7,6))
for name, data in roc_data.items():
    plt.plot(data['fpr'], data['tpr'], label=f"{name} (AUC={data['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')
plt.legend(); plt.grid(alpha=0.3)
plt.show()
```