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

```
df=pd.read_csv('pima-indians-diabetes.data.csv')
df.head()
```

```
   6  148  72  35    0  33.6  0.627  50  1
0  1   85  66  29    0  26.6  0.351  31  0
1  8  183  64   0    0  23.3  0.672  32  1
2  1   89  66  23   94  28.1  0.167  21  0
3  0  137  40  35  168  43.1  2.288  33  1
4  5  116  74   0    0  25.6  0.201  30  0
```

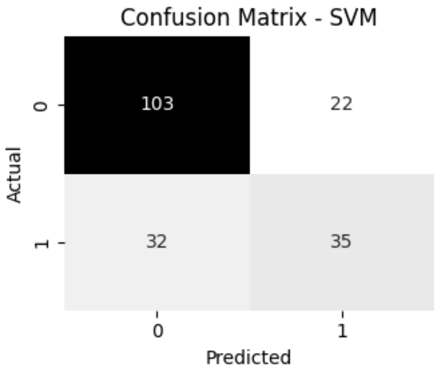
Next steps: Generate code with df New interactive sheet

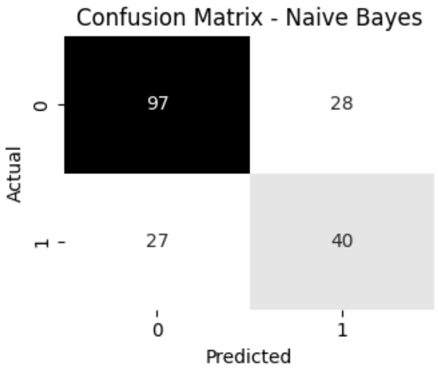
```
df.columns = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree','Age','Outcome']
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

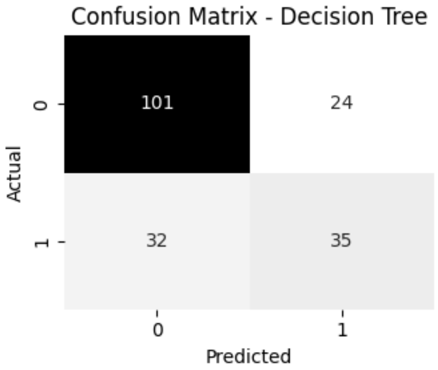
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

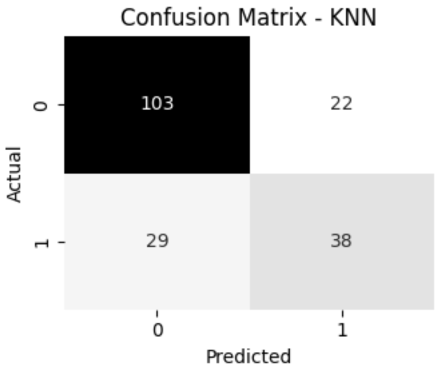
```
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, confusion_matrix, roc_curve, auc

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 = {}
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]
    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[name] = {
        'accuracy': acc,
        'recall': rec,
        'precision': prec,
        'f1': f1,
        'confusion_matrix': cm,
        'fpr': fpr,
        'tpr': tpr,
        'auc': roc_auc
    }
```

```
import seaborn as sns
import matplotlib.pyplot as plt

for name, r in results.items():
    plt.figure(figsize=(3.5,3))
    sns.heatmap(r['confusion_matrix'], annot=True, fmt='d', cmap='Greys', cbar=False)
    plt.title(f'Confusion Matrix - {name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.tight_layout()
    plt.show()
```





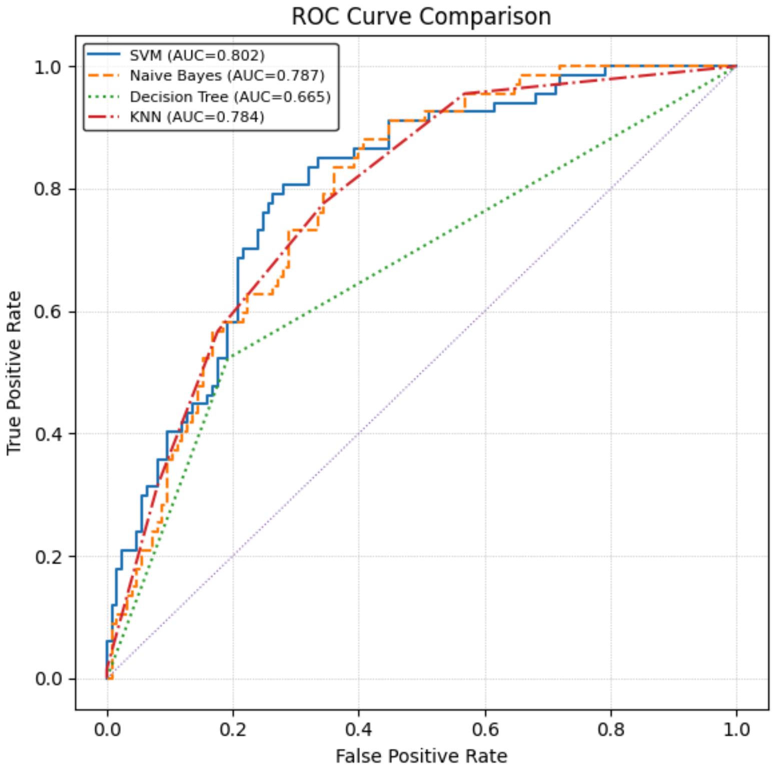




```
comparison = pd.DataFrame({
    'Model': list(results.keys()),
    'Accuracy': [results[m]['accuracy'] for m in results],
    'Recall': [results[m]['recall'] for m in results],
    'Precision': [results[m]['precision'] for m in results],
    'F1-Score': [results[m]['f1'] for m in results]
})
print(comparison)
```

```
           Model  Accuracy    Recall  Precision  F1-Score
0            SVM  0.718750  0.522388   0.614035  0.564516
1    Naive Bayes  0.713542  0.597015   0.588235  0.592593
2  Decision Tree  0.708333  0.522388   0.593220  0.555556
3            KNN  0.734375  0.567164   0.633333  0.598425
```

```
plt.figure(figsize=(6,6))
linestyles = ['-','--',':','-.']
for (name, r), ls in zip(results.items(), linestyles):
    plt.plot(r['fpr'], r['tpr'], linestyle=ls, label=f"{name} (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')
plt.legend(edgecolor='black', fontsize=8)
plt.grid(True, linestyle=':', linewidth=0.5)
plt.tight_layout()
plt.show()
```


ROC Curve Comparison
SVM (AUC=0.802)
Naive Bayes (AUC=0.787)
Decision Tree (AUC=0.665)
KNN (AUC=0.784)
True Positive Rate
False Positive Rate