```
import os
import numpy as np, pandas as pd
import matplotlib.pyplot as plt, seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

sns.set()
```

```
fname = "teleCust.csv"
if not os.path.exists(fname):
    try:
        from google.colab import files
        print("Upload 'teleCust.csv' now.")
        uploaded = files.upload()
        if uploaded:
            uploaded_name = list(uploaded.keys())[0]
            if uploaded_name != fname:
                os.rename(uploaded_name, fname)
    except Exception:
        raise FileNotFoundError(f"'{fname}' not found. Upload it to Colab Files.")

df = pd.read_csv(fname)
print("Loaded:", df.shape)
display(df.head())
```

```
Loaded: (1000, 12)
```

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

```
candidates = [c for c in df.columns if c.lower() in ('custcat','category','class','target','label')]
target = candidates[0] if candidates else df.columns[-1]   # fall back to last column
print("Using target column:", target)
```

```
Using target column: custcat
```

```
X = df.drop(columns=[target])
y = df[target]
# if target is text, map to integers
if y.dtype == object or y.dtype.name == 'category':
    y = pd.factorize(y)[0]
```

```
X = pd.get_dummies(X, drop_first=True)
X = X.fillna(X.median())
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
scaler = StandardScaler().fit(X_train)
X_train_s = scaler.transform(X_train)
X_test_s  = scaler.transform(X_test)
```

```
param_grid = {'n_neighbors': list(range(1,16,2))}
knn = KNeighborsClassifier()
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
gs = GridSearchCV(knn, param_grid, cv=cv, scoring='accuracy', n_jobs=-1, verbose=0)
gs.fit(X_train_s, y_train)
best_k = gs.best_params_['n_neighbors']
print("Best k:", best_k)
```

```
Best k: 15
```

```
best_knn = gs.best_estimator_
y_pred = best_knn.predict(X_test_s)
acc = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {acc:.4f}\n")
print("Classification report:\n", classification_report(y_test, y_pred, digits=4))
```

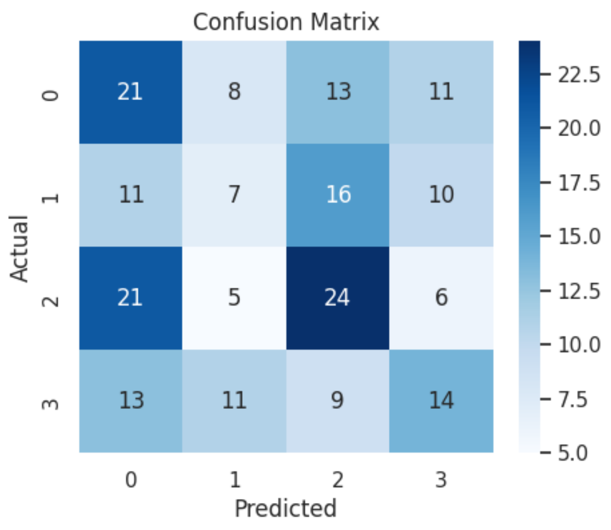
```
Test Accuracy: 0.3300

Classification report:
              precision    recall  f1-score   support

           1     0.3182    0.3962    0.3529        53
           2     0.2258    0.1591    0.1867        44
           3     0.3871    0.4286    0.4068        56
           4     0.3415    0.2979    0.3182        47

    accuracy                         0.3300       200
   macro avg     0.3181    0.3204    0.3161       200
weighted avg     0.3226    0.3300    0.3233       200
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4)); sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix'); plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.show()
```



```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4)); sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix'); plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.show()
```