```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from xgboost import XGBClassifier
```

```
raw_df = pd.read_excel("default of credit card clients.xls", header=1)
```

```
raw_df.rename(columns={'default payment next month': 'default'}, inplace=True)
```

```
df = raw_df.drop(columns=['ID'])
```

```
X = df.drop(columns='default')
y = df['default']
```

```
print(f"Shape of X: {X.shape}, Shape of y: {y.shape}")
```

```
Shape of X: (30000, 23), Shape of y: (30000,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
```

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
```

```
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)
```

```
grid_search.fit(X_train_scaled, y_train)
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [17:27:17] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
▸ GridSearchCV
▸ best_estimator_:
  XGBClassifier
    ▸ XGBClassifier
```

```
best_model = grid_search.best_estimator_
```

```
y_pred = best_model.predict(X_test_scaled)
```

```
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
```

```
print("Best Parameters:", grid_search.best_params_)
print(f"Accuracy: {acc:.4f}")
print("Confusion Matrix:\n", cm)
print("Classification Report:\n", cr)
```

```
Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100, 'subsample': 1.0}
Accuracy: 0.8180
Confusion Matrix:
 [[4435  238]
 [ 854  473]]
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.95      0.89      4673
           1       0.67      0.36      0.46      1327

    accuracy                           0.82      6000
   macro avg       0.75      0.65      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

Start coding or generate with AI.