```
import pandas as pd

train = pd.read_csv("kidney_train.csv")
test = pd.read_csv("kidney_test.csv")

X_train = train.drop("classification", axis=1)
y_train = train["classification"].map({"ckd":1, "notckd":0})  # encode target
X_test = test.drop("classification", axis=1)
y_test = test["classification"].map({"ckd":1, "notckd":0})
```

```
X_train = pd.get_dummies(X_train)
X_test = pd.get_dummies(X_test)

# Align columns
X_train, X_test = X_train.align(X_test, join="left", axis=1, fill_value=0)
```

```
from xgboost import XGBClassifier

xgb_model = XGBClassifier(
    objective="binary:logistic",
    eval_metric="logloss",
    use_label_encoder=False,
    random_state=42
)

xgb_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:199: UserWarning: [19:25:27] WARNING: /workspace/src/learner.cc:
Parameters: { "use_label_encoder" } are not used.

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

```
XGBClassifier
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, ...)
```

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    "max_depth": [3, 5, 7],
    "learning_rate": [0.01, 0.1, 0.2],
    "n_estimators": [100, 200, 300],
    "subsample": [0.8, 1],
    "colsample_bytree": [0.8, 1]
}

grid = GridSearchCV(
    estimator=XGBClassifier(objective="binary:logistic", eval_metric="logloss"),
    param_grid=param_grid,
    scoring="f1",
    cv=5,
    verbose=1
)

grid.fit(X_train, y_train)
best_model = grid.best_estimator_
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

```
 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix

y_pred = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:,1]

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1:", f1_score(y_test, y_pred))
```

```
print("ROC-AUC:", roc_auc_score(y_test, y_proba))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.975
Precision: 0.9753086419753086
Recall: 0.9875
F1: 0.9813664596273292
ROC-AUC: 0.9984375
Confusion Matrix:
 [[38  2]
 [ 1 79]]
```

```
import matplotlib.pyplot as plt
import xgboost as xgb

xgb.plot_importance(best_model, importance_type="gain")
plt.show()
```