```
import pandas as pd


df = pd.read_excel('/content/car.xlsx')
df
```

	brand	model	transmission	age	fuel	price	mileage	power	seats
0	18	244	1	4	1	1231000.0	19.01	4.496471	5
1	10	263	1	6	4	786000.0	19.01	4.496471	5
2	31	123	1	2	1	1489000.0	19.01	4.496471	5
3	9	55	0	1	4	1227000.0	19.01	4.496471	5
4	8	82	1	3	1	887000.0	19.01	4.496471	5
...	...	...	...	...	...	...	...	...	...
32009	5	199	1	6	4	292000.0	19.01	4.496471	5
32010	32	295	1	6	4	534000.0	19.01	4.496471	5
32011	33	25	1	8	4	424000.0	19.01	4.496471	5
32012	10	120	0	5	4	685000.0	19.01	4.496471	5
32013	31	247	1	2	4	392000.0	19.01	4.496471	5

32014 rows × 9 columns

```
X = df.drop('price',axis=1)
y = df['price']


from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=True)


from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score

# Initialize model
xgb = XGBRegressor(random_state=42)

# Train
xgb.fit(X_train, y_train)

# Predict
y_train_pred = xgb.predict(X_train)
y_test_pred = xgb.predict(X_test)

# Evaluate
print("Train R²:", r2_score(y_train, y_train_pred))
print("Test R²:", r2_score(y_test, y_test_pred))
print("Train MSE:", mean_squared_error(y_train, y_train_pred))
print("Test MSE:", mean_squared_error(y_test, y_test_pred))

# Cross-validation
cv_scores = cross_val_score(xgb, X, y, cv=5, scoring='r2')
print("Cross-validation R² scores:", cv_scores)
print("Mean CV R² score:", cv_scores.mean())
```

```
Train R²: 0.9463881557940258
Test R²: 0.894281932296258
Train MSE: 27035480841.700462
Test MSE: 48066179675.89659
Cross-validation R² scores: [0.82394955 0.87848573 0.89472928 0.89632106 0.70016122]
Mean CV R² score: 0.838729366097031
```

```
from sklearn.model_selection import GridSearchCV

# Define hyperparameter grid
param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.05, 0.1],
    'max_depth': [3, 5],
    'subsample': [0.8, 1]
}

# Grid Search
grid = GridSearchCV(XGBRegressor(random_state=42), param_grid, cv=5, scoring='r2', verbose=1, n_jobs=-1)
```

```
# Train
grid.fit(X_train, y_train)

# Best model
best_xgb = grid.best_estimator_

# Predict
y_train_pred = best_xgb.predict(X_train)
y_test_pred = best_xgb.predict(X_test)

# Evaluate
print("🔧 Best Parameters:", grid.best_params_)
print("Train R²:", r2_score(y_train, y_train_pred))
print("Test R²:", r2_score(y_test, y_test_pred))
print("Train MSE:", mean_squared_error(y_train, y_train_pred))
print("Test MSE:", mean_squared_error(y_test, y_test_pred))

# Cross-validation
cv_scores = cross_val_score(best_xgb, X, y, cv=5, scoring='r2')
print("Cross-validation R² scores:", cv_scores)
print("Mean CV R² score:", cv_scores.mean())
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
🔧 Best Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200, 'subsample': 1}
Train R²: 0.9213729737768
Test R²: 0.8859961782122705
Train MSE: 39650183510.38786
Test MSE: 51833412214.30537
Cross-validation R² scores: [0.81538452 0.8738343  0.89714999 0.88858739 0.64147353]
Mean CV R² score: 0.8232859473828155
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.