```
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 sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score

# Initialize the model
gbr = GradientBoostingRegressor(random_state=42)

# Train
gbr.fit(X_train, y_train)

# Predict
y_train_pred = gbr.predict(X_train)
y_test_pred = gbr.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(gbr, X, y, cv=5, scoring='r2')
print("Cross-validation R² scores:", cv_scores)
print("Mean CV R² score:", cv_scores.mean())
```

```
Train R²: 0.8429696423933132
Test R²: 0.8204219829835239
Train MSE: 79187561769.56638
Test MSE: 81647625796.03629
Cross-validation R² scores: [0.72932606 0.82269386 0.84522755 0.83495686 0.44881906]
Mean CV R² score: 0.7362046767981718
```

```
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]
}

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

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

# Best model
best_gbr = grid.best_estimator_

# Predict
y_train_pred = best_gbr.predict(X_train)
y_test_pred = best_gbr.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 (on full data using best model)
cv_scores = cross_val_score(best_gbr, 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 8 candidates, totalling 40 fits
🔧 Best Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}
Train R²: 0.928253599098017
Test R²: 0.890889140305139
Train MSE: 36180408933.41198
Test MSE: 49608759416.42939
Cross-validation R² scores: [0.8112893  0.87419963 0.89334909 0.8888044  0.64816818]
Mean CV R² score: 0.8231621196862029
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.