```
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.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score

# Initialize model
knn = KNeighborsRegressor()

# Train the model
knn.fit(X_train, y_train)

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

```
Train R2: 0.8959997726812934
Test R2: 0.8392815334132199
Train MSE: 52445428707.97314
Test MSE: 73072870702.13182
Cross-validation R2 scores: [0.68830242 0.82024626 0.83714765 0.82468885 0.69242185]
Mean CV R2 score: 0.7725614062124903
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor

# Define model
knn = KNeighborsRegressor()

# Hyperparameter grid to search
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
```

```
# GridSearch with 5-fold cross-validation
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='r2', n_jobs=-1)

# Fit on training data
grid_search.fit(X_train, y_train)

# Best model after tuning
best_knn = grid_search.best_estimator_
print("Best hyperparameters:", grid_search.best_params_)

# Predict and evaluate
y_train_pred = best_knn.predict(X_train)
y_test_pred = best_knn.predict(X_test)

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

```
Best hyperparameters: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}
Train R2: 0.9621322209006553
Test R2: 0.8631164284201553
Train MSE: 19096034309.597443
Test MSE: 62236006475.95242
Cross-validation R2 scores: [0.78612416 0.84699689 0.83975849 0.8644433  0.71163591]
Mean CV R2 score: 0.8097917487900348
```

Start coding or generate with AI.