In [3]:

```python
# ============================================================
# Notebook : boston_regression.ipynb
# Objectif : Modèle de régression + suivi MLflow + sauvegarde best_pipe
# ============================================================

# ----------------------------
# 1. Imports et configuration
# ----------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from pathlib import Path

from sklearn.model_selection import train_test_split, KFold, cross_validate
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from lightgbm import LGBMRegressor

import mlflow
import mlflow.sklearn
import joblib

pd.set_option("display.max_columns", None)
plt.style.use("seaborn-v0_8")

# ----------------------------
# 2. Chargement des données
# ----------------------------
# Le notebook est dans ./notebook, le CSV dans ../data
data_path = Path("../data/bostonhousing-693729dedb019653836667.csv")

df = pd.read_csv(data_path)

print("Aperçu des données :")
display(df.head())

print("\nInfo sur le DataFrame :")
df.info()

print("\nStatistiques descriptives :")
display(df.describe().T)

# ----------------------------
# 3. Visualisation rapide
# ----------------------------
df.hist(figsize=(12, 8), bins=20)
plt.tight_layout()
```

```python
plt.show()

# ----------------------------
# 4. Définition cible / features
# ----------------------------
# ⚠️ Vérifie que la cible s'appelle bien "medv" dans df.columns
# Sinon, adapte cette ligne (ex : target = "MEDV")
target = "medv"

if target not in df.columns:
    raise ValueError(
        f"La colonne cible '{target}' n'existe pas dans le DataFrame. "
        f"Colonnes dispo : {df.columns.tolist()}"
    )

X = df.drop(columns=[target])
y = df[target]

print("\nColonnes d'entrée (features) :")
print(X.columns.tolist())
print("\nColonne cible :", y.name)

# ----------------------------
# 5. Séparation train / test
# ----------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

print("\nShapes train / test :")
print("X_train :", X_train.shape)
print("X_test  :", X_test.shape)

# ----------------------------
# 6. Préprocessing (scaling)
# ----------------------------
numeric_features = X.columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features)
    ],
    remainder="drop"
)

# ----------------------------
# 7. Définition des modèles
# ----------------------------
models = {
    "LinearRegression": LinearRegression(),
    "RandomForest": RandomForestRegressor(
        n_estimators=300,
        random_state=42,
        n_jobs=-1
```

```python
    ),
    "LightGBM": LGBMRegressor(
        n_estimators=500,
        learning_rate=0.05,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=42,
        n_jobs=-1
    )
}

print("\nModèles définis :", list(models.keys()))

# ----------------------------
# 8. Configuration MLflow
# ----------------------------
mlflow.set_experiment("boston_housing_regression")

# Active l'autolog pour scikit-learn
mlflow.sklearn.autolog()

# ----------------------------
# 9. Validation croisée + logging MLflow
# ----------------------------
cv = KFold(n_splits=5, shuffle=True, random_state=42)

scoring = {
    "rmse": "neg_root_mean_squared_error",
    "mae": "neg_mean_absolute_error",
    "r2": "r2"
}

results = []

for name, model in models.items():
    print(f"\n============================")
    print(f"=== Entraînement modèle : {name}")
    print(f"============================")

    with mlflow.start_run(run_name=name):
        pipe = Pipeline(steps=[
            ("preprocess", preprocess),
            ("model", model)
        ])

        cv_res = cross_validate(
            pipe,
            X_train,
            y_train,
            cv=cv,
            scoring=scoring,
            return_train_score=False
        )

        rmse_mean = -cv_res["test_rmse"].mean()
        mae_mean = -cv_res["test_mae"].mean()
```

```python
        r2_mean = cv_res["test_r2"].mean()

        results.append({
            "model": name,
            "RMSE_cv_mean": rmse_mean,
            "RMSE_cv_std": cv_res["test_rmse"].std(),
            "MAE_cv_mean": mae_mean,
            "R2_cv_mean": r2_mean
        })

        # Log des métriques moyennes dans MLflow
        mlflow.log_metric("RMSE_cv_mean", rmse_mean)
        mlflow.log_metric("MAE_cv_mean", mae_mean)
        mlflow.log_metric("R2_cv_mean", r2_mean)

# Tableau récapitulatif
results_df = pd.DataFrame(results).sort_values("RMSE_cv_mean")
print("\nRésultats validation croisée (ordonnés par RMSE croissant) :")
display(results_df)

# ----------------------------
# 10. Sélection du meilleur modèle
# ----------------------------
best_model_name = results_df.iloc[0]["model"]
print("\nMeilleur modèle (CV) :", best_model_name)

best_model = models[best_model_name]

best_pipe = Pipeline(steps=[
    ("preprocess", preprocess),
    ("model", best_model)
])

print("\nPipeline sélectionné :")
print(best_pipe)

# ----------------------------
# 11. Entraînement final et métriques test
# ----------------------------
best_pipe.fit(X_train, y_train)

y_pred = best_pipe.predict(X_test)

# ⚠️ Correction ici : on ne passe plus squared=False
# Ancienne version (qui plante chez toi) :
# rmse_test = mean_squared_error(y_test, y_pred, squared=False)

# Nouvelle version compatible : MSE puis racine carrée
mse_test = mean_squared_error(y_test, y_pred)
rmse_test = np.sqrt(mse_test)

mae_test = mean_absolute_error(y_test, y_pred)
r2_test = r2_score(y_test, y_pred)

print("\nPerformances sur le jeu de test :")
print(f"RMSE (test) : {rmse_test:.4f}")
```

```python
print(f"MAE  (test) : {mae_test:.4f}")
print(f"R²   (test) : {r2_test:.4f}")

# ----------------------------
# 12. Graphique Réel vs Prédit
# ----------------------------
plt.figure(figsize=(6, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot(
    [y_test.min(), y_test.max()],
    [y_test.min(), y_test.max()],
    "r--"
)
plt.xlabel("Prix réel (medv)")
plt.ylabel("Prix prédit")
plt.title(f"{best_model_name} - Réel vs prédit (test)")
plt.show()

# ----------------------------
# 13. Distribution des résidus
# ----------------------------
residuals = y_test - y_pred

plt.figure(figsize=(6, 4))
sns.histplot(residuals, bins=30, kde=True)
plt.xlabel("Résidu")
plt.title("Distribution des résidus")
plt.show()

# ----------------------------
# 14. Importance des variables (si modèle d'arbres)
# ----------------------------
if hasattr(best_model, "feature_importances_"):
    importances = best_model.feature_importances_
    feature_importances = pd.Series(importances, index=numeric_features).sort_value

    plt.figure(figsize=(8, 5))
    sns.barplot(x=feature_importances.values, y=feature_importances.index)
    plt.title(f"Importance des variables - {best_model_name}")
    plt.xlabel("Importance")
    plt.ylabel("Variable")
    plt.show()

    print("\nImportance des variables :")
    display(feature_importances)
else:
    print("\nLe modèle sélectionné ne fournit pas d'importance de features.")

# ----------------------------
# 15. Log du modèle final + sauvegarde pour FastAPI
# ----------------------------
# 1) Nouveau run MLflow pour le modèle final sur le test
with mlflow.start_run(run_name=f"{best_model_name}_final"):
    mlflow.sklearn.log_model(best_pipe, artifact_path="model")
    mlflow.log_metric("RMSE_test", rmse_test)
    mlflow.log_metric("MAE_test", mae_test)
```

```python
    mlflow.log_metric("R2_test", r2_test)

# 2) Sauvegarde locale pour l'API FastAPI
models_dir = Path("../models")
models_dir.mkdir(exist_ok=True)

model_path = models_dir / "best_model.pkl"
joblib.dump(best_pipe, model_path)

print(f"\n✅ Modèle final sauvegardé dans : {model_path.resolve()}")
print("   Ce fichier sera utilisé par l'API FastAPI (main.py).")
```

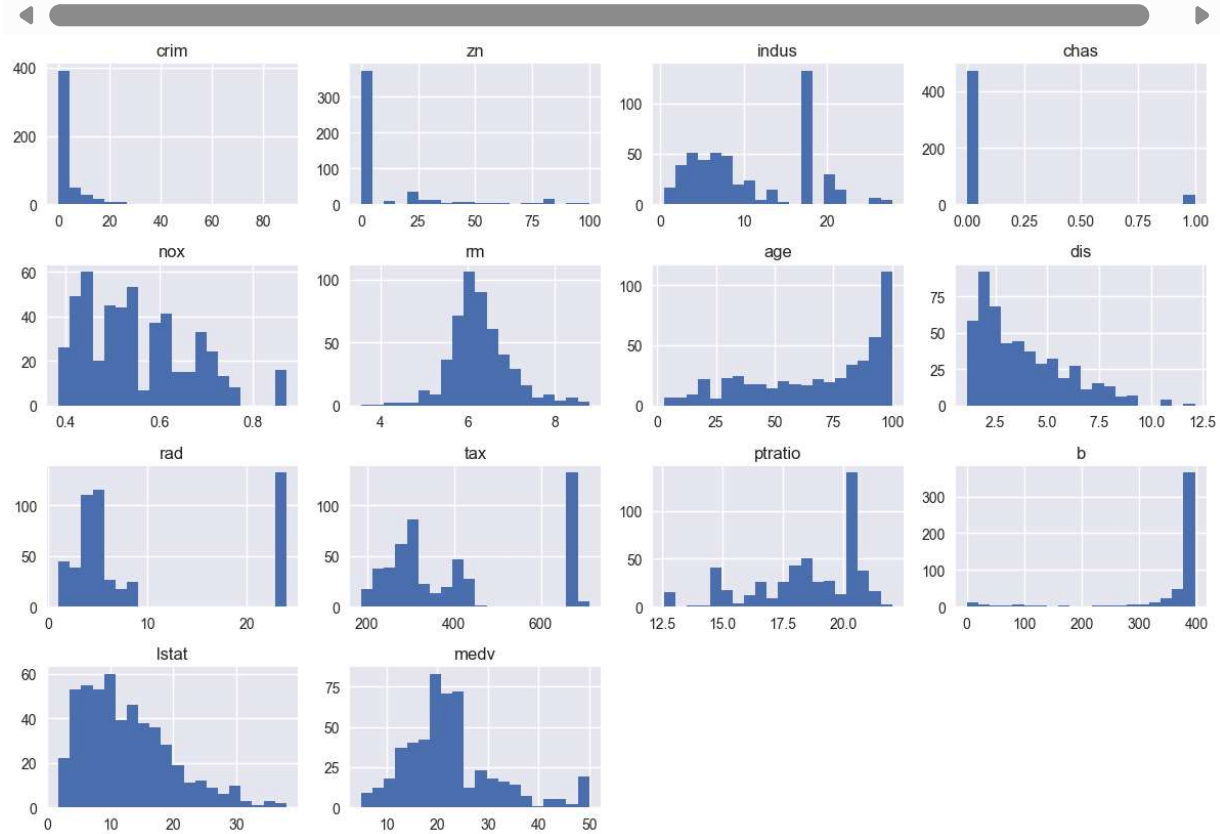
```
Aperçu des données :
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	me
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	2
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	2
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	3
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	3
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	3

```
Info sur le DataFrame :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB

Statistiques descriptives :
```

	count	mean	std	min	25%	50%	75%	ma
crim	506.0	3.613524	8.601545	0.00632	0.082045	0.25651	3.677083	88.976
zn	506.0	11.363636	23.322453	0.00000	0.000000	0.00000	12.500000	100.000
indus	506.0	11.136779	6.860353	0.46000	5.190000	9.69000	18.100000	27.740
chas	506.0	0.069170	0.253994	0.00000	0.000000	0.00000	0.000000	1.000
nox	506.0	0.554695	0.115878	0.38500	0.449000	0.53800	0.624000	0.871
rm	506.0	6.284634	0.702617	3.56100	5.885500	6.20850	6.623500	8.780
age	506.0	68.574901	28.148861	2.90000	45.025000	77.50000	94.075000	100.000
dis	506.0	3.795043	2.105710	1.12960	2.100175	3.20745	5.188425	12.126
rad	506.0	9.549407	8.707259	1.00000	4.000000	5.00000	24.000000	24.000
tax	506.0	408.237154	168.537116	187.00000	279.000000	330.00000	666.000000	711.000
ptratio	506.0	18.455534	2.164946	12.60000	17.400000	19.05000	20.200000	22.000
b	506.0	356.674032	91.294864	0.32000	375.377500	391.44000	396.225000	396.900
lstat	506.0	12.653063	7.141062	1.73000	6.950000	11.36000	16.955000	37.970
medv	506.0	22.532806	9.197104	5.00000	17.025000	21.20000	25.000000	50.000



```
Colonnes d'entrée (features) :
['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio',
'b', 'lstat']

Colonne cible : medv

Shapes train / test :
X_train : (404, 13)
X_test  : (102, 13)

Modèles définis : ['LinearRegression', 'RandomForest', 'LightGBM']

==============================
=== Entraînement modèle : LinearRegression
==============================

==============================
=== Entraînement modèle : RandomForest
==============================

==============================
=== Entraînement modèle : LightGBM
==============================
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000419 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 812
[LightGBM] [Info] Number of data points in the train set: 323, number of used featur
es: 13
[LightGBM] [Info] Start training from score 22.786068
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000423 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 807
[LightGBM] [Info] Number of data points in the train set: 323, number of used featur
es: 13
[LightGBM] [Info] Start training from score 22.661610
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000453 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 817
[LightGBM] [Info] Number of data points in the train set: 323, number of used featur
es: 13
[LightGBM] [Info] Start training from score 22.900929
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.000308 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 813
[LightGBM] [Info] Number of data points in the train set: 323, number of used featur
es: 13
[LightGBM] [Info] Start training from score 22.551393
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000409 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 821
[LightGBM] [Info] Number of data points in the train set: 324, number of used featur
es: 13
[LightGBM] [Info] Start training from score 23.081790
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
2025/12/11 18:19:49 INFO mlflow.utils.autologging_utils: Created MLflow autologging
run with ID 'e3edefa0d3c842e2b155302f4df30566', which will track hyperparameters, pe
rformance metrics, model artifacts, and lineage information for the current sklearn
workflow
2025/12/11 18:19:49 WARNING mlflow.utils.autologging_utils: MLflow autologging encou
ntered a warning: "C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages
\mlflow\types\utils.py:452: UserWarning: Hint: Inferred schema contains integer colu
mn(s). Integer columns in Python cannot represent missing values. If your input data
contains missing values at inference time, it will be encoded as floats and will cau
se a schema enforcement error. The best way to avoid this problem is to infer the mo
del schema based on a realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as doubles (float64) whenever
these columns may have missing values. See `Handling Integers With Missing Values <h
ttps://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>
`_ for more details."
```

```
Meilleur modèle (CV) : LightGBM

Pipeline sélectionné :
Pipeline(steps=[('preprocess',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['crim', 'zn', 'indus',
                                                   'chas', 'nox', 'rm', 'age',
                                                   'dis', 'rad', 'tax',
                                                   'ptratio', 'b',
                                                   'lstat'])])),
                ('model',
                 LGBMRegressor(colsample_bytree=0.8, learning_rate=0.05,
                               n_estimators=500, n_jobs=-1, random_state=42,
                               subsample=0.8))])
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.000433 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 982
[LightGBM] [Info] Number of data points in the train set: 404, number of used featur
es: 13
[LightGBM] [Info] Start training from score 22.796535
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

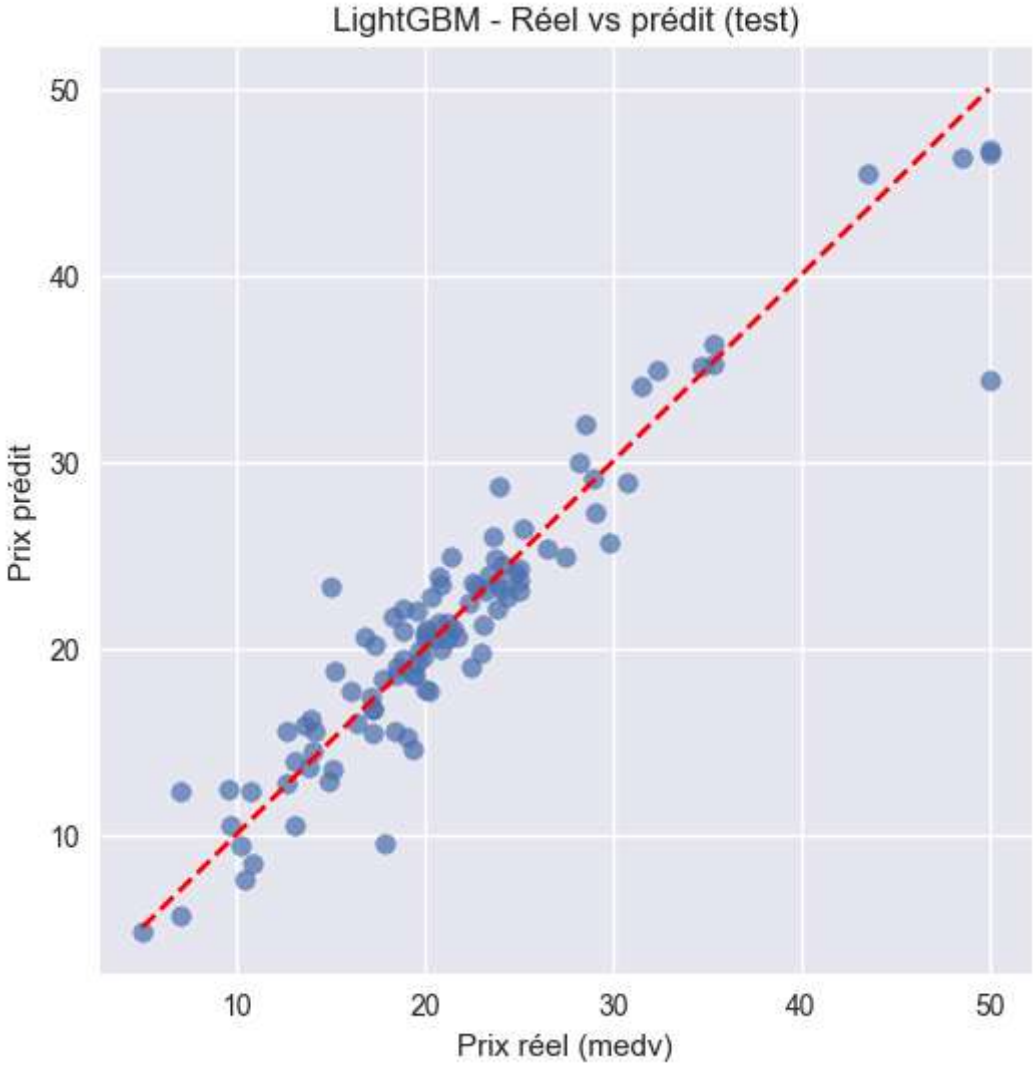
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

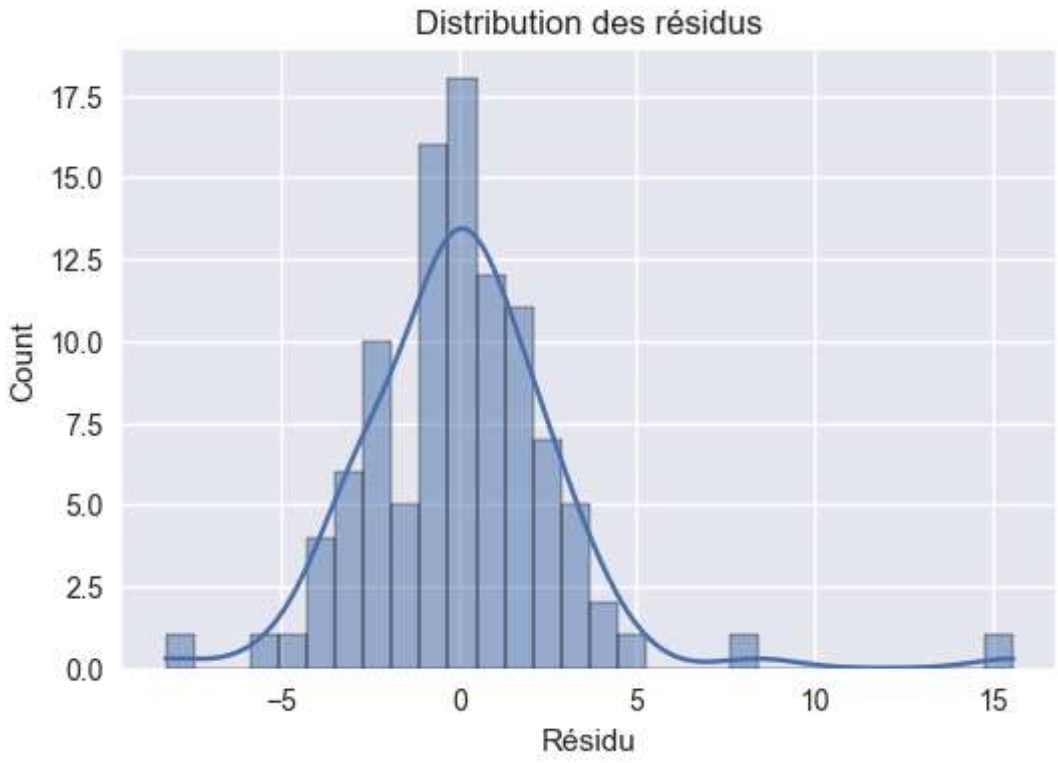
```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
2025/12/11 18:19:50 WARNING mlflow.utils.autologging_utils: MLflow autologging encou
ntered a warning: "C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages
\mlflow\types\utils.py:452: UserWarning: Hint: Inferred schema contains integer colu
mn(s). Integer columns in Python cannot represent missing values. If your input data
contains missing values at inference time, it will be encoded as floats and will cau
se a schema enforcement error. The best way to avoid this problem is to infer the mo
del schema based on a realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as doubles (float64) whenever
these columns may have missing values. See `Handling Integers With Missing Values <h
ttps://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>
`_ for more details."
```

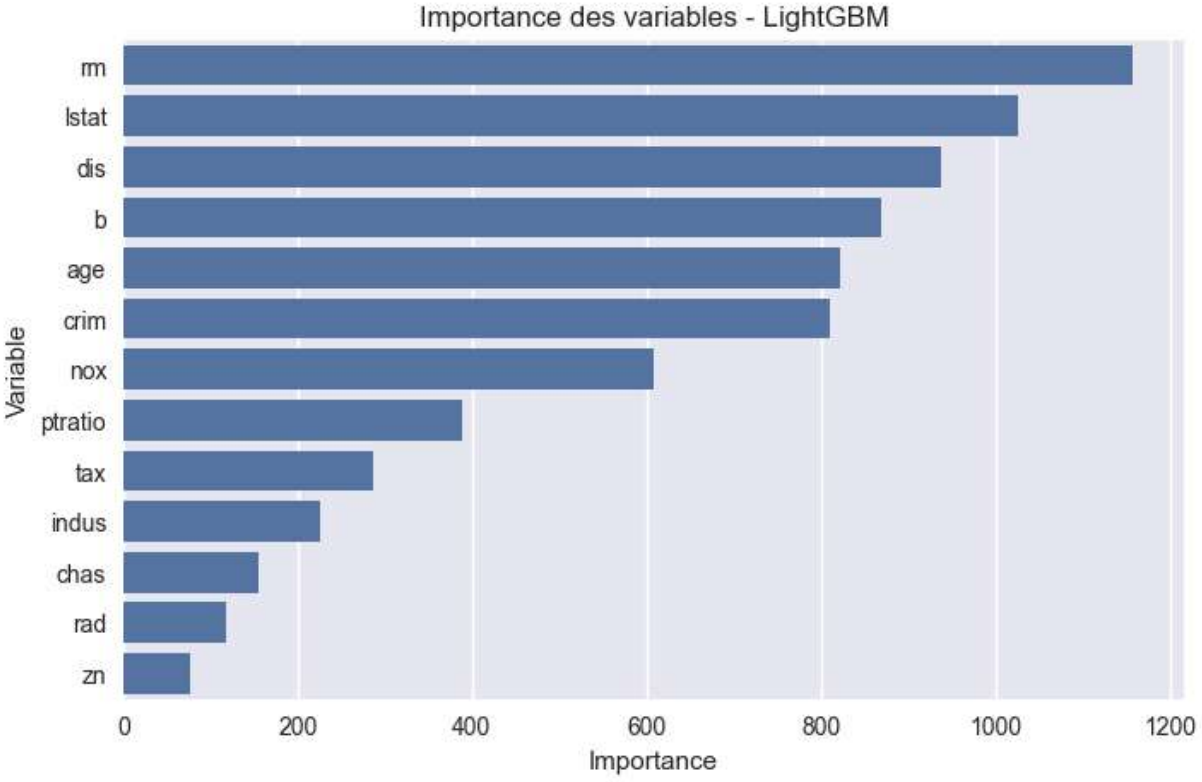
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages\sklearn\utils\valid
ation.py:2749: UserWarning: X does not have valid feature names, but LGBMRegressor w
as fitted with feature names
  warnings.warn(
2025/12/11 18:20:05 WARNING mlflow.utils.autologging_utils: MLflow autologging encou
ntered a warning: "C:\Users\pcino\source\repos\fastia-boston\venv\lib\site-packages
\mlflow\types\utils.py:452: UserWarning: Hint: Inferred schema contains integer colu
mn(s). Integer columns in Python cannot represent missing values. If your input data
contains missing values at inference time, it will be encoded as floats and will cau
se a schema enforcement error. The best way to avoid this problem is to infer the mo
del schema based on a realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as doubles (float64) whenever
these columns may have missing values. See `Handling Integers With Missing Values <h
ttps://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>
`_ for more details."
```

```
Performances sur le jeu de test :
RMSE (test) : 2.8186
MAE  (test) : 1.9215
R²   (test) : 0.8917
```







Importance des variables :

```
rm         1158
lstat      1025
dis         937
b           869
age         821
crim        811
nox         608
ptratio     389
tax         287
indus       226
chas        156
rad         118
zn           76
dtype: int32
```

```
2025/12/11 18:20:07 WARNING mlflow.models.model: `artifact_path` is deprecated. Plea
se use `name` instead.
```

```
✅ Modèle final sauvegardé dans : C:\Users\pcino\source\repos\fastia-boston\models
\best_model.pkl
   Ce fichier sera utilisé par l'API FastAPI (main.py).
```

In [ ]: