Compte Rendu : Amélioration du Modèle de Prédiction des Prix (Stage 2)

Camille DUPRE LA TOUR

Novembre 2024

Introduction

Dans le cadre de ce deuxième stage, l'objectif était de poursuivre le travail effectué dans le premier stage en améliorant les performances du modèle de prédiction des prix en utilisant plusieurs techniques de régression. Le modèle initial était basé sur des méthodes simples, telles que la régression linéaire et les méthodes de régression régulière, mais il s'avérait nécessaire d'explorer de nouvelles approches et d'optimiser le modèle pour obtenir de meilleures performances. L'approche a consisté à tester plusieurs algorithmes de régression, dont la régression Ridge, Lasso et Random Forest, en procédant à une optimisation des hyperparamètres pour chaque modèle. Ce rapport présente les étapes suivies, les résultats obtenus, ainsi que les conclusions tirées de cette analyse.

Plan d'attaque

Le plan d'attaque pour l'amélioration du modèle s'est structuré autour des étapes suivantes :

1. Choix des modèles

Afin de comparer plusieurs techniques de régression et d'identifier celle qui offrirait les meilleures performances, j'ai sélectionné cinq modèles de régression pertinents pour le problème de prédiction des prix :

- **Régression Ridge**
- **Régression Lasso**
- **Random Forest Regressor**
- **XGBoost** (initialement prévu mais non implémenté dans ce projet)

Chaque modèle a été évalué avec ses hyperparamètres par défaut pour effectuer une première analyse des performances et déterminer quels modèles pourraient être affinés.

2. Implémentation des modèles

L'implémentation de chaque modèle a été réalisée en utilisant la bibliothèque `sklearn`. La procédure a consisté à entraîner chaque modèle avec les hyperparamètres par défaut, puis à effectuer des prédictions sur l'ensemble de test. Les performances des modèles ont été évaluées en utilisant trois principales métriques : l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne (MAE) et le coefficient de détermination R^2.

3. Résultats initiaux

Pour chaque modèle, les performances ont été analysées. En particulier, j'ai observé que certains modèles, tels que **Random Forest Regressor**, semblaient surperformer les autres en termes de MSE et de R^2. En revanche, les modèles Ridge et Lasso se sont montrés moins performants, notamment en raison d'une mauvaise convergence lorsque les hyperparamètres étaient mal adaptés.

4. Optimisation des hyperparamètres

Le prochain objectif a été de réaliser un **fine-tuning** des modèles les plus performants, en particulier le Random Forest. Un **RandomizedSearchCV** a été utilisé pour tester une large gamme d'hyperparamètres, tels que le nombre d'arbres (`n_estimators`), la profondeur des arbres (`max_depth`), et d'autres paramètres influençant la performance du modèle. L'objectif était de trouver les meilleurs hyperparamètres pour maximiser les performances sur l'ensemble de validation tout en évitant le sur-apprentissage (overfitting).

5. Visualisation des résultats

Les performances de chaque modèle ont été visualisées à travers des courbes de validation et de comparaison entre les valeurs réelles et les valeurs prédites. Des graphiques ont été générés pour observer les comportements des modèles et vérifier la présence éventuelle de sous-apprentissage ou de sur-apprentissage.

Réalisations et Analyse du Code

Choix des modèles

Le choix des modèles a été fait en fonction de leur capacité à traiter les problèmes de régression complexes. La régression linéaire est un modèle de base qui a permis de poser une première référence. La régression Ridge et Lasso sont des versions régularisées de la régression linéaire, ce qui les rend intéressantes dans le contexte de données présentant des caractéristiques multicolinéaires. Le Random Forest Regressor est une méthode d'ensemble puissante capable de capturer des relations non linéaires dans les données.

Implémentation des modèles

Les modèles ont été implémentés de manière standard en utilisant la bibliothèque `sklearn`. Le code pour la régression Ridge et Lasso ressemble à ce qui suit :

```
from sklearn.linear_model import Ridge, Lasso
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)
ridge_pred = ridge_model.predict(X_test)
```

Le Random Forest Regressor a été implémenté en utilisant les hyperparamètres par défaut pour débuter les tests :

```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
```

Résultats initiaux

Les résultats des performances initiales ont révélé que le Random Forest Regressor était le modèle le plus performant en termes de R^2 et de MSE. Les résultats des autres modèles, en particulier les régressions Ridge et Lasso, ont montré que les performances étaient moins bonnes. Voici un résumé des premières performances observées :

- **Ridge et Lasso** : Moins performants avec des valeurs de MSE et R^2 relativement faibles.
- **Random Forest** : A montré une meilleure capacité à prédire les prix avec un R^2 élevé et un MSE plus faible.

Fine-tuning du modèle Random Forest

Le Random Forest Regressor a été choisi comme modèle à affiner davantage. J'ai utilisé `RandomizedSearchCV` pour tester une gamme d'hyperparamètres différents. Les hyperparamètres inclus dans la recherche étaient les suivants :

```
param_distributions = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2']
}
```

Les meilleures performances ont été obtenues avec les hyperparamètres suivants :

- `n_estimators` = 100
- `max_depth` = 20
- `min_samples_split` = 10

Visualisation des résultats

Après avoir affiné les hyperparamètres, j'ai généré des graphiques pour visualiser la performance du modèle sur les données de test. En particulier, j'ai utilisé un graphique de réalité vs prédiction pour observer la capacité du modèle à prédire les valeurs avec précision.

Voici le code pour la visualisation :

```
def plot_reality_vs_prediction(y_test, y_pred, model_name):
    plt.figure(figsize=(8, 8))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.6, color="blue", edgecolor="k")
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=
    plt.xlabel("Valeurs réelles")
    plt.ylabel("Valeurs prédites")
    plt.title(f"Réalité vs Prédiction ({model_name})")
    plt.grid()
    plt.show()
```

Cela a permis de visualiser l'adéquation entre les valeurs réelles et les valeurs prédites par le modèle affiné, ce qui a montré une bonne capacité du modèle à prédire les prix.

Conclusion

Les résultats obtenus au cours de ce stage montrent que le modèle Random Forest, après une optimisation de ses hyperparamètres, a considérablement amélioré les prédictions des prix. La régression Ridge et Lasso n'ont pas montré une amélioration significative, en particulier avec les valeurs par défaut de leurs hyperparamètres. L'optimisation des modèles à travers `RandomizedSearchCV` a permis de fine-tuner le modèle Random Forest pour obtenir les meilleures performances.

En termes de performances, le Random Forest Regressor a offert un $\mathbf{R^2}$ **de 0.98**, ce qui témoigne d'une bonne qualité de prédiction. Ce projet a permis d'identifier les meilleures pratiques pour l'optimisation des hyperparamètres, et montre que des techniques d'optimisation avancées peuvent conduire à des améliorations significatives dans les modèles de régression.

Ce travail a été une étape cruciale dans le processus d'amélioration de la prédiction des prix, et des recherches supplémentaires, notamment en explorant des techniques comme XGBoost, pourraient encore améliorer les résultats.