ML project 1: Predicting house prices: application of regression techniques

Daniele Prevedello

Abstract

This projects focus on using various machine learning methods, especially linear regression and decision trees, to predict the house prices in Ames, Iowa. The best model is XGBoost, and we discuss all the models we tried and evaluate them.

1 Notebook for the code

https://colab.research.google.com/drive/1euCWl6FxlGudbNgncRvD8qepspScR3cc?usp=sharing

2 Introduction

In this project we try to predict the final sales prices for houses in Ames, Iowa. There are 79 explanatory variables describing different aspects of residential homes, we try to use different machine learning models to make prediction, and evaluate the performance of these models. The metric we use to evaluate the prediction will be Root Mean Squared Error (RMSE) for the log of the predicted and actual prices. By taking log, errors in predicting expensive houses and cheap houses will be treated equally.

3 Data examination

There are 1460 entries in the training set and 1459 on the test set. Among the 79 features, 36 are numerical and 43 are categorical. Some columns has missing values, especially categorical ones, where NA usually indicates the lack of the feature.

4 Tree-based models

We first try decision-tree-based models. One advantage of them is that we do not have to clean the data beforehand, decision trees can handle categorical, numerical, and missing values on themselves. We consider three tree-based models, CART, random forest, and gradient boosting. We use 5-fold cross-validation to find the best two among these three: gradient boosting and random forest. Then we test these on the test set. RMSE of gradient boosting is 0.1334, and for random forest, it is 0.1467.

4.1 Improvement

In the above model, we simply impute the numerical columns with median and impute categorical columns with mode. This is not accurate, since a lot of the missing data in the categorical columns means the lack of the feature, therefore they should be treated as a separate category. Upon fixing this, the RMSE of gradient boosting and random forests improves to 0.1316 and 0.1462 respectively.

4.2 Important features

For all these tree-models, the most important features by far is Overall quality, followed by Total living area above ground. Other important features include garage size, basement area, first and second floor areas. We can see from here that a lot of these features are actually highly correlated, and we need more careful feature engineering and data cleaning.

5 XGBoost

To go further, next we try XGBoost, the leading model for working with standard tabular data. XGBoost improves upon gradient boosting. We train the data with hyperparameters (n estimators=1200, learning rate=0.05, max depth=4, subsample=0.8, colsample bytree=0.8), and the resulting RMSE is 0.1297.

5.1 Improvements

There are two improvements we made at this step. First instead of training with sales prices, we train with log of the sales prices, since the RMSE will be evaluated on the log scale. Also we use grid search on the training set to find the best hyperparameter ('colsample bytree': 0.8, 'learning rate': 0.03, 'max depth': 3, 'n estimators': 1200, 'subsample': 0.8), on test set the RMSE is 0.1277. However if we further tune the parameters on a smaller grid, the effect of overfitting will occur. On the training set the larger n estimators we have the better, but if we take n estimators=1500, the RMSE on the test set will increase to 0.1321.

6 Linear regression

Next we will try basic linear regression techniques, including OLS, Ridge and Lasso. On training set, the RMSE of OLS is 0.1569, for Ridge it is 0.1525, and for LASSO it is 0.1501.

6.1 Improvement

We use grid search and 5-fold Cross-Validation to find the best alpha for LASSO and Ridge. After tuning we run on test set and have RMSE 0.1320 for Ridge and 0.1336 for LASSO.

7 Other models

We also try K-th nearest neighbor and sequential feature selector for only keep the best features. After 5-fold cross validation the best KNN=9, and on test set the RMSE=0.1693.

Alternatively, we can consider best features selector. We use forward feature selection. For parameter 5, the RMSE on test data is 0.1696. With parameter 10 it improves to 0.1534. Note that testing SFS is significantly slower than other models.

8 Discussion

8.1 Model comparisons

The models from best to worst are: XGBoost, gradient boosting, ridge, LASSO, random forest, OLS, Sequential selector, KNN.

The best model on the test set is XGBoost. It improves upon gradient boosting, which is better than other tree-based models. For linear regression with L1 and L2 penalty, if we use grid search to find the best alpha, it improves the model significantly.

KNN is one of the worse models, unsurprisingly. With 79 features we are in a very high dimensional space, a geometric properties of high dimensional unit balls shows that, by simple integration, one can realize that the mass of the unit ball lies almost entirely on the sphere. Therefore just finding the K-th nearest neighbor will not be a good idea when we have too many features.

8.2 Improvements

The performance can be further improved by careful data cleaning and feature engineering. For categorical data we did create a new category and use one-hot encoding for all the missing values, instead of simply imputing them. We can also look closely at the anomaly in the data, for example where NA in category does not mean lack of the feature, but instead means not available. For example there are three entries with Pool Area nonzero but Pool quality NA, in this case they do have pools and we should treat the NA differently comparing with others.

Another improvement comes from the log transform. The metric we will be evaluated on is the RMSE of the log price, so instead of training on the price, we transform it into log, and exponentiate the prediction back on the test set. This improves the model accuracy, since it reduces the outlier effect on the expensive houses.

Most importantly, some model performances can be enhanced by feature engineering. For example, the features "full bathroom", "half bathroom", "full bsmt bathroom", "half bsmt bathroom" can be linearly combined into a new feature. A lot of the 79 features can be grouped together this way and simplified using feature engineering.

9 Graphs

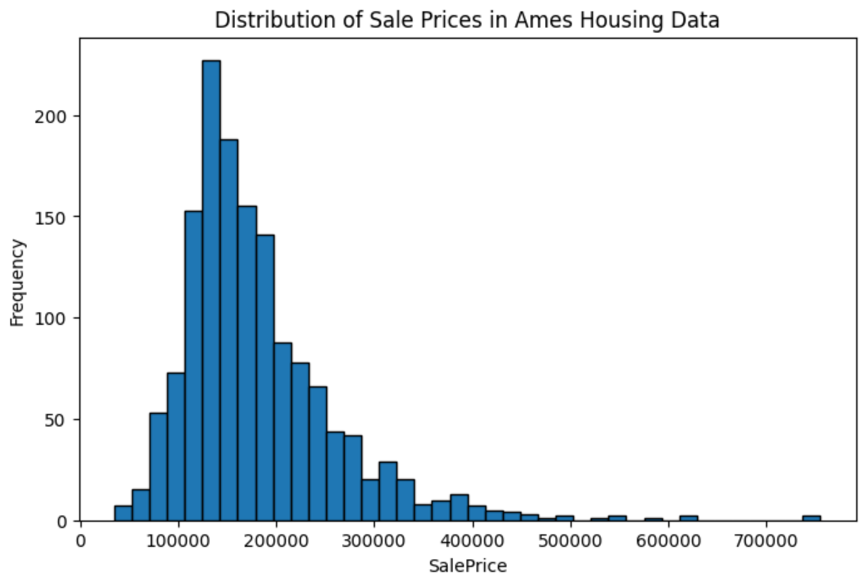


Figure 1: Sales Price Distribution

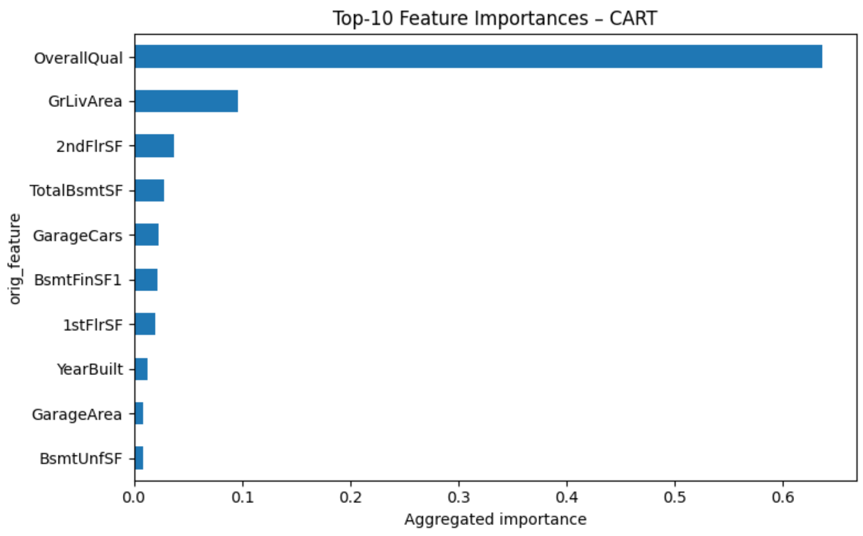


Figure 2: Top features CART

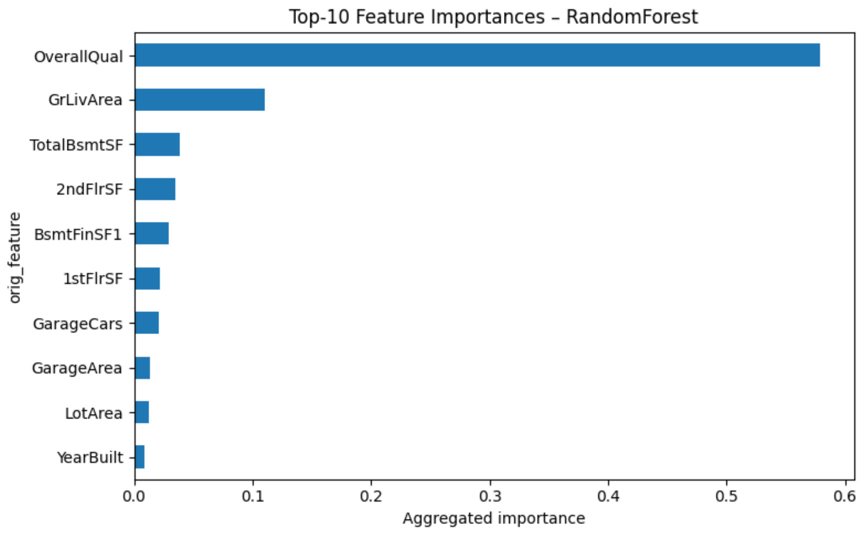


Figure 3: Top features Random Forest

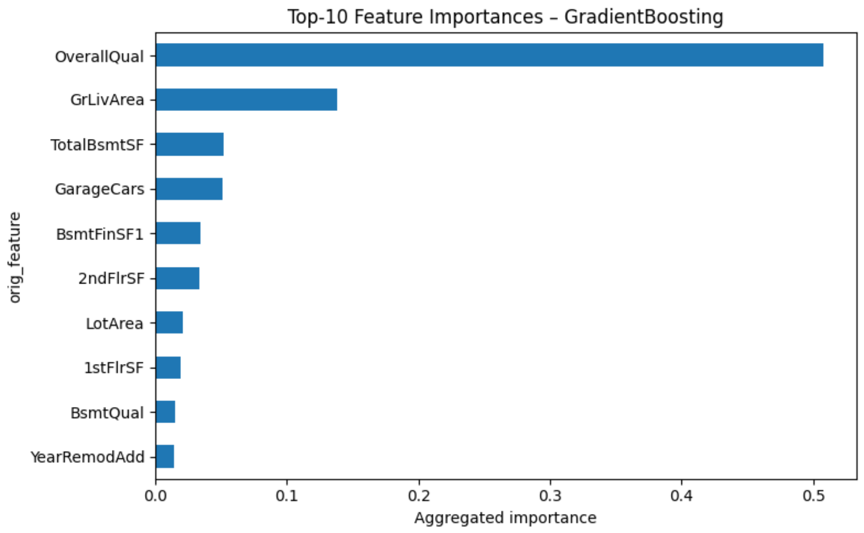


Figure 4: Top features Gradient Boosting

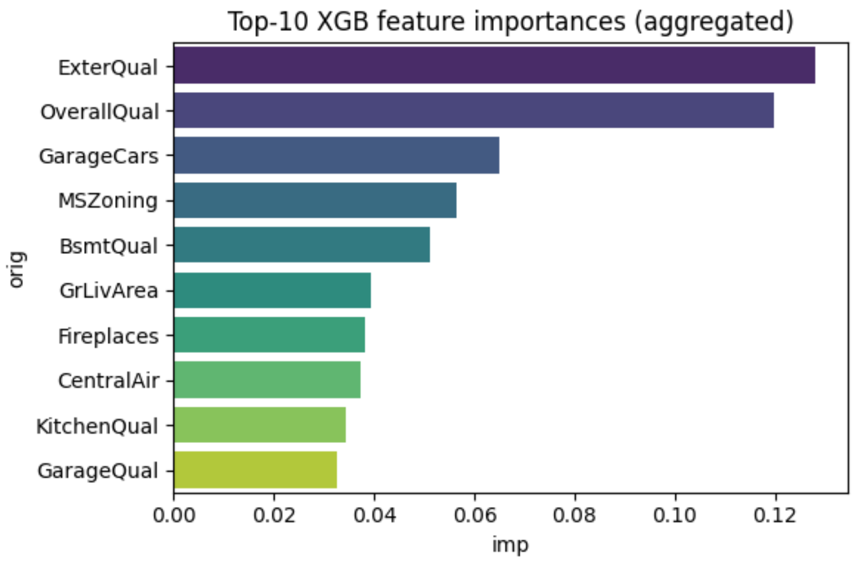


Figure 5: Top features XGB

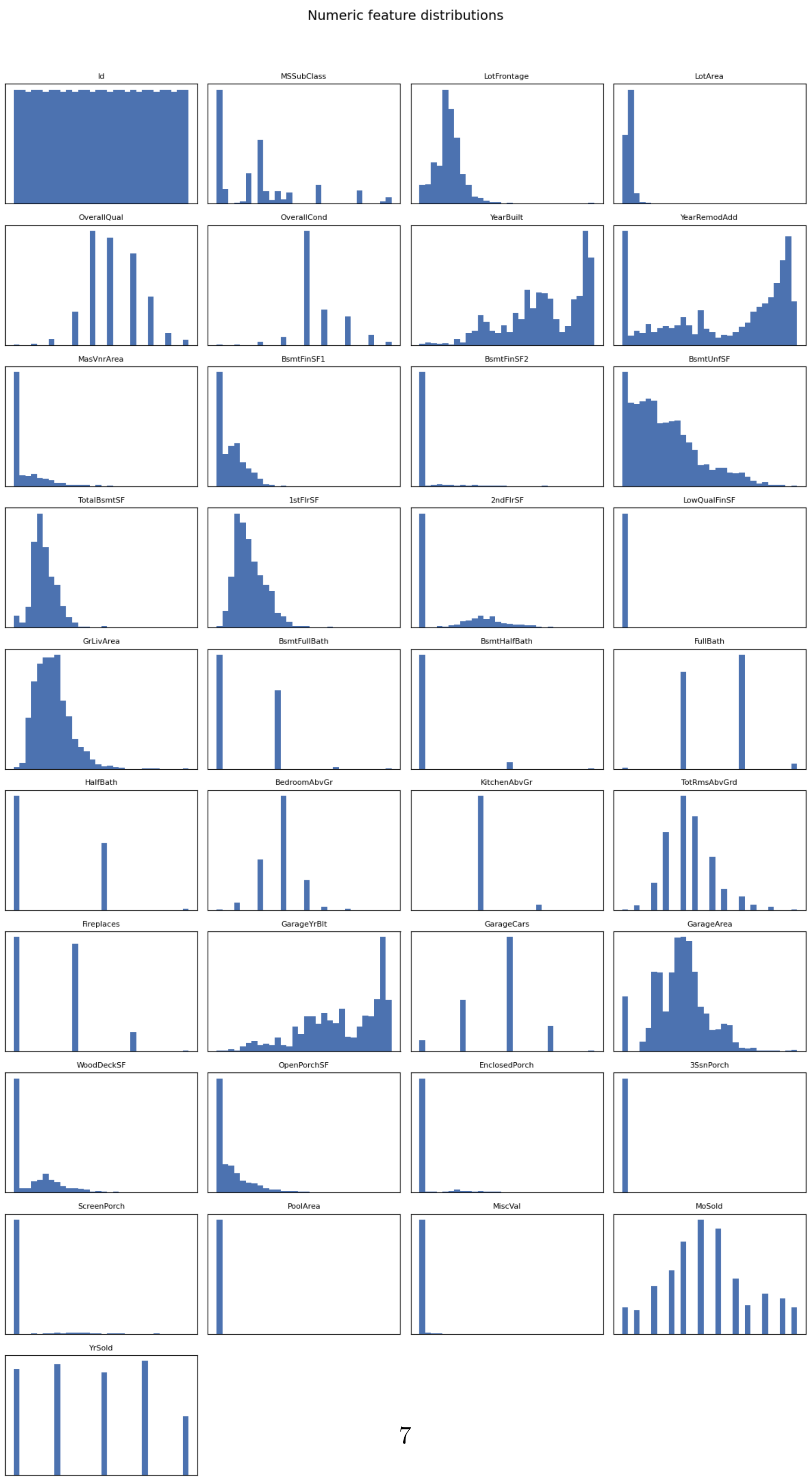
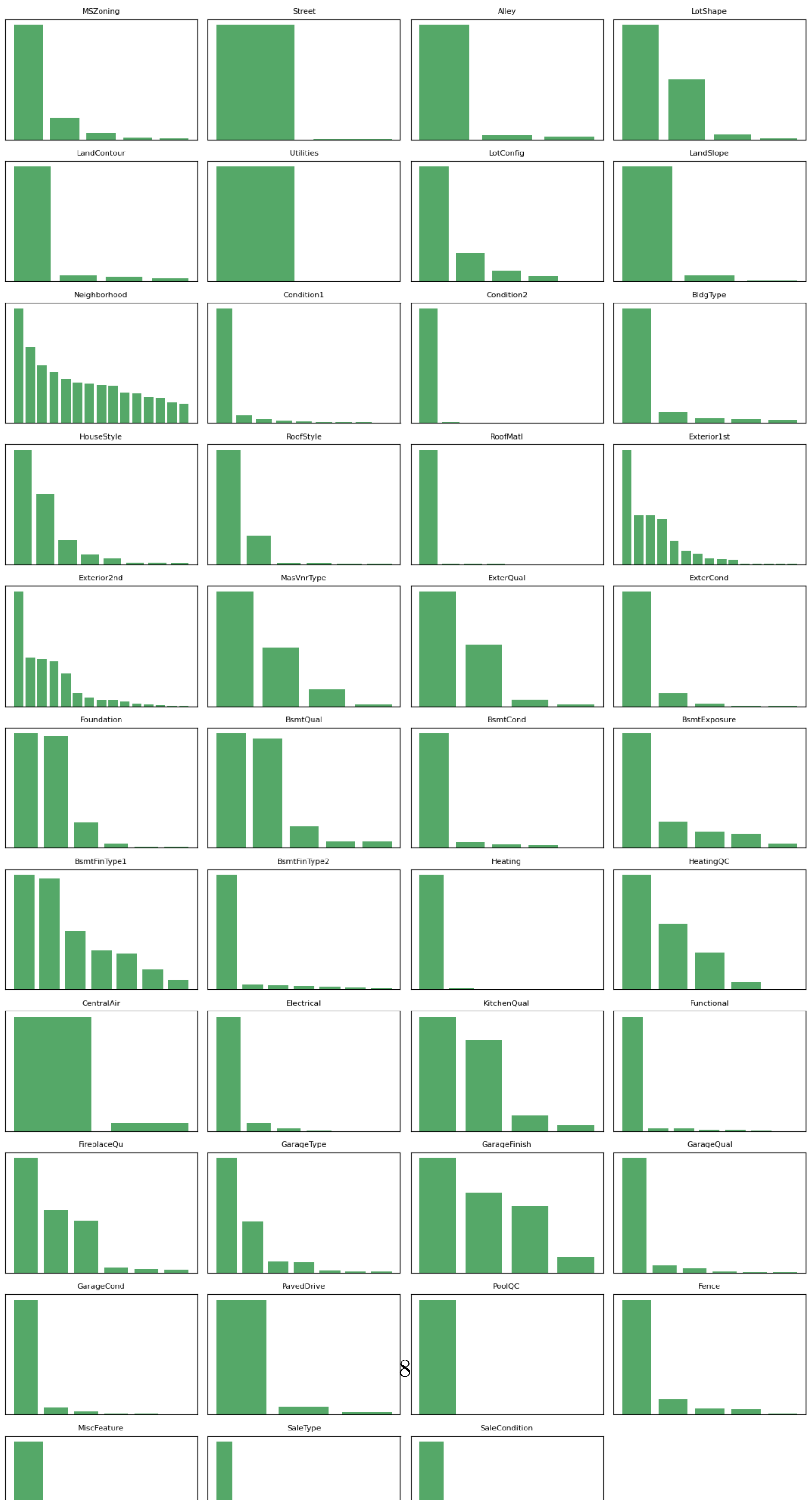


Figure 6: Numerical features

Categorical feature counts (top-15 per feature)



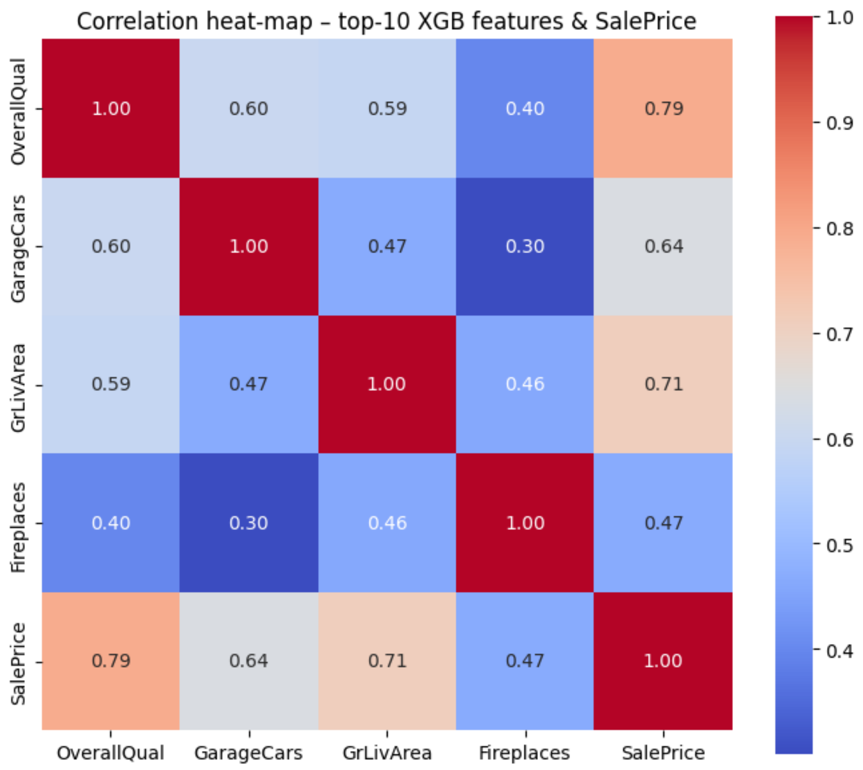


Figure 8: Top features correlation

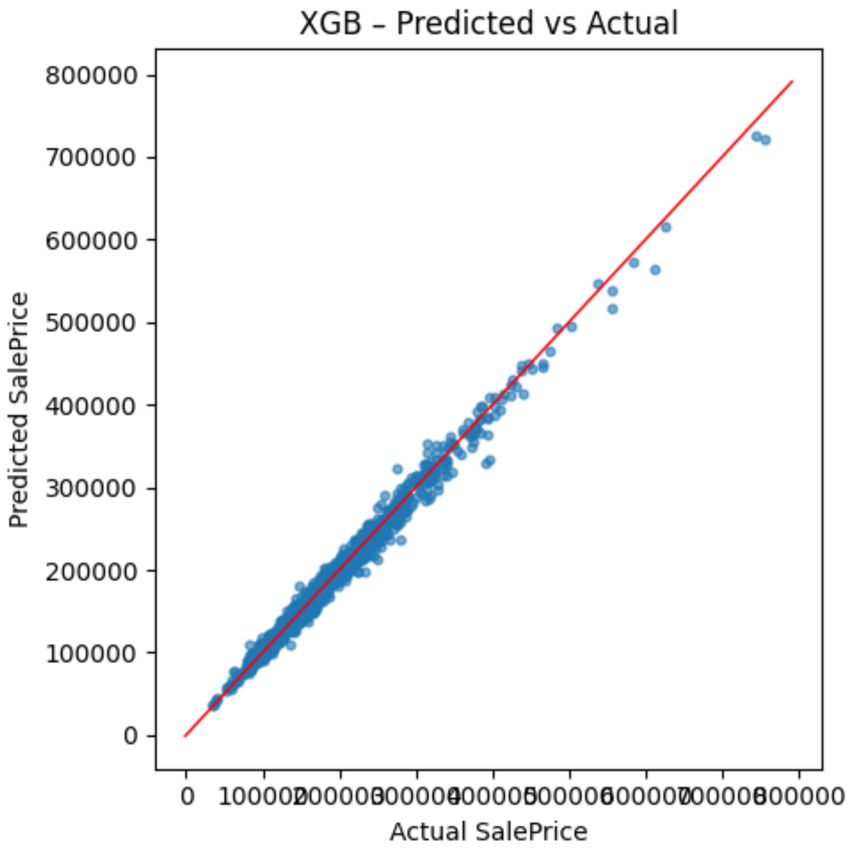


Figure 9: XGB prediction