Name (netid): Jennifer Lim (jylim3)
CS 441 - Final Report

A Challenge I choose: House price regression:
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques.

Content
- Approach: Describe your final approach, ideally with enough detail for an ML expert to re-implement, but taking no more than 1 page.

I could divide it into four parts of my approach.

The first step is loading the dataset and importing the required libraries for this challenge. Since the datasets are given as CSV files, I called the train and test CSV files and read those using the read_csv function from pandas. In this process, I downloaded the data and stored it in my google drive, so I mounted it first and called data from there.

The second step is checking the information of each test and training data. With the info() function, I checked each dataset's name, type, and number of columns. I also used describe() function to check each mean, std, count, and other values for each column in each dataset. This is the set-up step to visualize data, especially finding out which columns or data types are needed to solve this challenge. After that, I used a basic library to visualize each data's relationship by drawing a histogram, bell curve, and heatmap. This helped me to visualize the relationship of the SalePrice column, which is the primary data type we need to check in this challenge. At this step, I used to display, correct, and heatmap functions from the Seaborn library.

The third step is preprocessing the data. Since we have a raw dataset containing null values and columns we do not want to include, we need to reorganize the data that fit our train. For that, I first remove rows with missing targets. Then, separate targets from the predictors by using drop and drop functions. After removing unnecessary rows from the raw train dataset, I used train_test_split from the sklearn library to divide the raw train dataset into train and validation data. This part is needed because we always need to check our training result with the validation dataset before the test dataset to check if we are on the right path in ML training. Moreover, I select columns with relatively low unique values in each column except four specific columns because it will be more helpful to get better training results without those 4 column values. Then, I also find columns with numeric types like int or float values. The sum of these two columns will be the columns I will use. I also impute my train, validation, and test datasets for better accuracy.

The fourth step is modeling the data with two different regressors: Random Forest and Gradient Boosting. I adapted each model from the sklearn library. After I make a model and fit to train data, I evaluate with MAE and RMSE. After getting a great accuracy value, I trained them again and predicted with the test dataset.

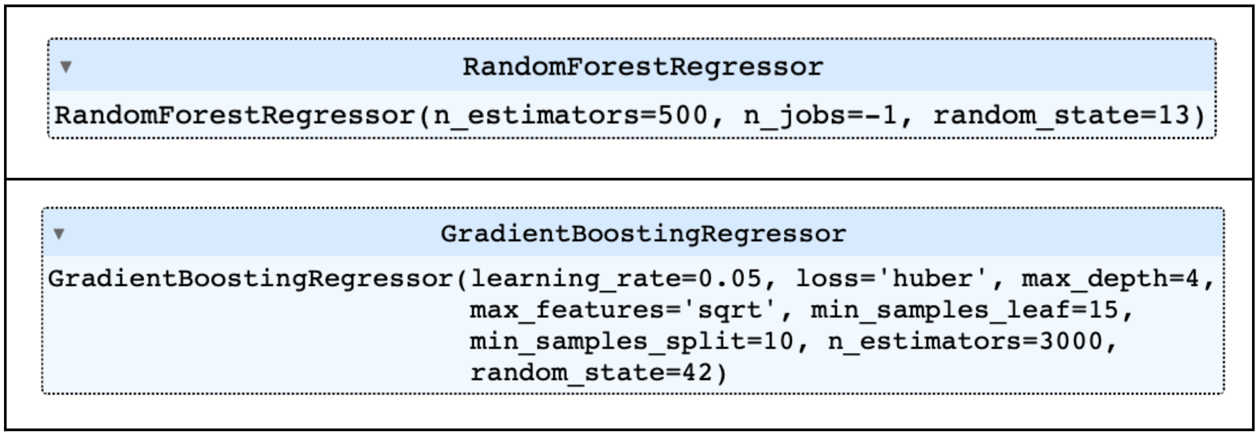
These are the brief steps I took to do this challenge.

- Implementation Details: List packages that you use (excluding basic ones like numpy) and external sources of code. Describe parameters/architecture.

Packages I used excluding basic

1. Sklearn
    a. model_selection
        i. train_test_split
    b. metrics
        i. mean_absolute_error
        ii. mean_squared_error
    c. ensemble
        i. RandomForestRegressor
        ii. GradientBoostingRegressor
    d. impute
        i. SimpleImputer

Parameters/architecture



```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=500, n_jobs=-1, random_state=13)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.05, loss='huber', max_depth=4,
                          max_features='sqrt', min_samples_leaf=15,
                          min_samples_split=10, n_estimators=3000,
                          random_state=42)
```

- Experiments: Include results of your approach and ablations indicating importance of design parameters.

With Random Forest Regressor, I resulted in 0.14945 public scores in the Kaggle challenge submission and scored 0.12765 with Gradient Boosting Regressor.

I ended up performing three ablations from this challenge.

First, I remove and modify features. In the preprocessing step, I remove some unnecessary columns (features) from the raw train and raw test datasets and modify those using the sklearn imputing function. As mentioned in the approach step, I remove features with low unique values and modify datasets with unique high values and numerical types features.

Second, I varying hyperparameters in my two models. Referring to parameters/architectures in the implementation section, I assigned n_estimators, n_jobs, and random_states to perform better in Random

Forest Regressor. In the Gradient Boosting Regressor model, I assigned many different parameters to increase performance: learning rate, loss, max_depth, and other parameter values. The lists of parameters I modified are also in the parameters/architectures in the implementation section.

Lastly, I adapted two different regression models to compare the performance. I could compare which regression models better fit this house price dataset, and we could compare them by the public scores I got in the Kaggle competition submission.

After finishing this challenge by adapting the following types of ablation studies, I could better understand the importance of different design parameters and potentially identify areas where I could improve the performance of my models.

- Discussion: What are your conclusions? What would you do differently next time? [¼ page]

Random Forest Regression and Gradient Boosting Regression are well-designed algorithms for the house price regression challenge. Getting around 0.13 public score, it is well-trained and predictable. However, as I finished my challenge, I believed there could be ways to improve. Here are the things I would like to do differently next time. I would like to preprocess the data in more detail. Around 30 and 80 features existed in the train and test dataset, and I believe if I preprocessed a more fitting way to this challenge than what I did, I would get better results. So, I need to visualize and set how each feature relates to others. This also applies to feature engineering. I will involve creating new features from the existing ones, selecting relevant features, and transforming them to make them more informative next time. I believe I performed well on hyperparameter tuning. However, sklearn regression models offer a variety of parameters; there could be better hyperparameters that perform better. I could do more research and find the best-fit hyperparameter next time. Finally, there are more regression models than Random Forest Regression and Gradient Boosting Regression like xgbregression. Next time, I could try different regression models and compare which model performs better.

- Citations/Acknowledgements: Cite sources of ideas and code, including websites/blogs/github pages and models. You do not need to cite standard libraries.

[1]“Comprehensive data exploration with Python,” *kaggle.com*.
https://www.kaggle.com/code/pmarcelino/comprehensive-data-exploration-with-python
: I got the basic ideas of data exploration and visualization from this notebook

- My code:

https://colab.research.google.com/drive/19qiqN9A-emKT2MezftLf_iSHKpr3BxXj?usp=sharing