Final Project

Multi-task Learning for Predicting House Prices and House Category

Seunghyun Park

260686853

Data Preprocessing

For the analysis, the 'test' and 'train' datasets were combined, resulting in a dataset that includes information on 2,919 houses, each characterized by 81 features.

Data Type Conversion

The 'MSSubClass' was converted from numerical to string format to recognize it as a categorical feature.

Missing Data

The initial review of the dataset revealed a variety of columns affected by missing values. A detailed count helped quantify the extent of missing data across different features. For each feature, the proportion of missing values was calculated, leading to the removal of columns where more than 30% of the data was missing.

For columns with less than 30% missing data, different imputation strategies were applied:

- Numerical values such as 'MasVnrArea' were filled with 0.
- Categorical features related to basement and garage conditions (e.g., 'BsmtQual', 'GarageType') were imputed with 'NA', representing the absence of these features in the property.

For the 'LotFrontage' feature, which represents the property's street connected frontage, KNN Imputer was used to fill the missing values. Additionally, columns that either lacked significant data or were deemed not to impact the model's performance significantly, such as 'ID', 'Electrical', and 'GarageYrBlt', were removed from the dataset.

Feature Engineering and Data Transformation

The dataset was refined by simplifying the 'HouseStyle' and 'BldgType' attributes. For instance, 'HouseStyle' entries such as '1Story', '1.5Fin', and '1.5Unf' were grouped under the label 'One story', while '2Story', '2.5Fin', and '2.5Unf' were categorized as 'Two story'. Split-level homes were labeled as 'Split'. Similarly, 'BldgType' categories 'TwnhsE' and 'Twnhs' were categorized as 'Townhouse'. Any styles or types not fitting these descriptions were classified as 'Other'.

Additionally, a new feature called 'YearAfterRemod' was introduced. This was calculated by subtracting the year of the last remodel ('YearRemodAdd') from the year the house was sold

('YrSold'), providing insight into the recency of updates prior to the sale. Following this, 'YearAfterRemod' was then standardized using a StandardScaler. Additionally, the simplified categorical features ('SimplifiedStyle' and 'SimplifiedBuildingType') were encoded using a OneHotEncoder.

Furthermore, the DBSCAN clustering was employed to create a new 'HouseCategory' feature, effectively categorizing houses into distinct groups based on their style, type, and time elapsed since the last remodel.

Cluster ID	**Style**	**Building Type**	**Average Years After Remodel**
0	Two-story	Other	15.82
1	One-story	Other	28.51
2	One-story	Townhouse	6.09
3	Split-level	Other	27.39
4	Two-story	Townhouse	5.46
5	Two-story	Townhouse	33.91
6	Split-level	Townhouse	24.84

Table 1: House Category

After assigning categories using clustering, 'HouseStyle', 'BldgType', 'YearBuilt', 'YearRemodAdd', columns were dropped from the dataset to prevent data leakage.

Data Preparation for Multi-Task Learning Model

The dataset was initially divided into predictor variables and target variables, which include 'SalePrice' and 'HouseCategory'. The features were separated into numerical and categorical groups, and categorical variables were encoded using one-hot encoding, while numerical features were scaled using MinMaxScaler.

Furthermore, to enhance computational efficiency and manage dimensionality, Principal Component Analysis (PCA) was employed to condense the dataset into 25 principal components. This step was crucial for improving model performance by minimizing the risk of overfitting.

The processed data was then converted into tensors, and split into training, validation, and testing sets.

Multi-task Learning Model for Predicting House Prices and House Category Using PyTorch Lightning

The multi-task learning model, crafted using PyTorch Lightning, was designed to simultaneously predict house prices (regression task) and house categories (classification task).

The model architecture featured shared layers that included LeakyReLU activations and dropout for regularization. Specifically, the structure consisted of three layers:

1. **First Layer**:
    - Linear transformation from 25 input features to 1024 neurons
    - LeakyReLU activation
    - Dropout at a 0.1 rate
2. **Second Layer**:
    - Linear transformation from 1024 to 512 neurons
    - LeakyReLU activation
    - Dropout at a 0.1 rate
3. **Third Layer**:
    - Linear transformation from 512 to 256 neurons
    - LeakyReLU activation
    - Dropout at a 0.1 rate

From the shared layers, the model architecture diverged into two distinct pathways to accommodate the dual objectives of the multi-task learning model:

1. **Price Prediction Pathway**:
    - This pathway started with a linear transformation from 256 to 512 neurons using GELU activation function, followed by batch normalization to standardize inputs.
    - Dropout rate of 0.1
    - ReLU activations function at different stages (from 512 to 256, then from 256 to 128, and finally from 128 to 64 neurons)
    - From 64 to a single output value, LeakyReLU activation function was used before the final output, which represents the predicted house price.
2. **Category Prediction Pathway**:

- This pathway started with a linear transformation from 256 to 512 neurons using GELU activation function, followed by batch normalization to standardize inputs.
- Dropout rate of 0.1
- ReLU activations function at different stages (from 512 to 256, then from 256 to 128, and finally from 128 to 64 neurons)
- From 64 to a to 7 output value, LeakyReLU activation function was used, which represents the predicted house category.

For the loss functions, CrossEntropyLoss and MSE were employed for house category classification and house price prediction respectively. In terms of optimization, An Adam optimizer was used.

During training, the model took in data to predict house prices and categories, calculated the losses for each, and used backpropagation to fine-tune the weights.

To improve the training process, the setup included TensorBoard logging for real-time performance monitoring, ModelCheckpoint to save the best-performing models based on validation loss, and EarlyStopping to halt training when there were no improvements in validation loss over several epochs.

Additionally, hyperparameter optimization was conducted using Optuna, which automated the exploration of various combinations of learning rates and batch sizes.

Result

The results from different configurations of the multi-task learning model demonstrated substantial improvements in both regression and classification tasks through systematic enhancements and optimizations. Initially, a model setup using ReLU activation and a single hidden layer yielded poor results, with a root mean square error (RMSE) of 185,650 for house price prediction and a mere 17% accuracy for house category classification.

In response to these initial results, the model architecture was refined to incorporate multiple hidden layers and advanced activation functions. This new model configuration significantly enhanced performance, achieving an RMSE of 53,980.14 and a test accuracy of approximately 56.29% for the classification task. Further detailed testing confirmed these results with a slight improvement in RMSE to 49,398.25, maintaining the same level of accuracy.

Additionally, employing Optuna for hyperparameter optimization enhanced the model's performance. The best trial from Optuna optimization yielded a validation loss of 52,975.99, identifying the optimal hyperparameters as a learning rate of approximately 0.00196 and a batch size of 16. Using the

configuration from Optuna Optimization, the model achieved an RMSE of 52,793 and a test accuracy of 59.73%. This reflects significant enhancements in the model's ability to predict house prices and classify house types accurately.