APPLIED AI MODULE

COMP534

ASSIGNMENT 2

Student's name	Huy Pham	Saeth Wannasuphoprasit
ID	201534475	201585689

Note that (‘number followed by topic name’) is the section in the given python file. For example, (‘2. Data preparation’) is the section number 2 in the given python file.

I. Introduction

1. The libraries used

Library’s name	Used Purpose
pandas, Series, DataFrame	load and read data
matplotlib.pyplot, seaborn	visualize data
math, numpy	calculate with array
warnings	ignore warnings
RandomizedSearchCV	find the best hyperparameters
keras, tensorflow, Sequential, Dense, layers, KerasRegressor	Train and test ANN model
Preprocessing from sklearn	Data normalization
train_test_split from sklearn	Split data into train and test
make_scorer from sklearn	Calculate accuracy score

2. The detail of the development process

In the data preparation process (‘2.2.1 deal with data type’), there were no null values in the dataset. So, we didn’t need to deal with missing values. The ‘date’ feature was transformed from the string of dates into an integer in order to be calculatable in the regression task. According to the columns’ description, in the ‘bathroom’ feature, records that were not ended with 0 or 0.5 were eliminated, and the ‘floor’ feature was rounded from the float into an integer. Next (‘2.2.2 correlation and features selection’), correlation scores of each feature were calculated to find how relevant each feature was for the target output feature (price). The features with an absolute value of correlation score lower than 0.2 were dropped because they were not that relevant to the output feature. This also reduced the complexity of the models since there were fewer input features. After that (‘2.2.3 normalize data’), all data were normalized into a 0-1 scale because features had different ranges, and this may affect the performance of the regression task (reference). For example, the ‘view’ feature ranged from 0 to 4, whereas the ‘grade’ feature ranged from 1 to 13. Then, in (‘2.2.4 detect and remove outliers’), the outliers (z-scores were lower than 3) were dropped because those can be misinformative and were not make sense. For example, ‘sqft_basement’ had the maximum value of 4130 which is very far away from the average value (around 230). Finally, in (‘2.2.5 test and train split’), the whole dataset was split into the training dataset (80%) and testing dataset (20%) to prepare the data to train and test the upcoming models.

The next step after data preparation was to find the initial hyperparameters used as default values in the hyperparameters tuning process. To do this, in (‘3.1 function to create models’), the pyramid structure network was selected because of the decrease of complexity in the

hidden layers (fewer nodes in hidden layers resulting in less running time to train the models). In the output layer, there was only one node (linear activation) because the output was a single float number (price), and linear activation is suitable for regression tasks. Next, in (‘3.2 random search cv’), the initial hyperparameters from the top best 5 models with the highest MAPE scores (mentioned below) were selected using RandomsearchCV (because it saved running time by sampling some hyperparameters’ configurations not all combinations like GridsearchCV) with cross-validation fold = 5 from the training dataset, and the scoring method was Mean absolute percentage error (MAPE) (suitable for regression task).

The final step (‘4. hyperparameters tuning’) was hyperparameters tuning to find the best possible model. For each hyperparameter, we varied its value and fixed the rest hyperparameters to the default values mentioned above to see the impact of the selected hyperparameter on the MAPE accuracy. Then, in each round, we selected values of hyperparameters that tended to improve the performance (high accuracy) of the models and defined the best hyperparameters using RandomsearchCV (the same process with ‘3.2 random search cv’ mentioned above). Then, in the next round and so on, the remaining hyperparameters, not being selected to vary yet, were optimized using the same process.

3. The values of the hyperparameters tested in the project.

Optimizer: ['SGD', 'RMSprop', 'Adam', 'Adamax', "Nadam"]

first_layer_nodes: [2,3,5,10,15,20,25,30,40,50,60,70,80]

last_layer_nodes: [1]

n_layers: [2,3,5,7,9,12,15,20]

activation: ['softmax', 'relu', 'tanh', 'sigmoid']

loss: ["mean_squared_error", "mean_absolute_error", "mean_squared_logarithmic_error"]

batch_size: [2,4,8, 16, 32, 64, 128, 256, 512, 1024, 2048]

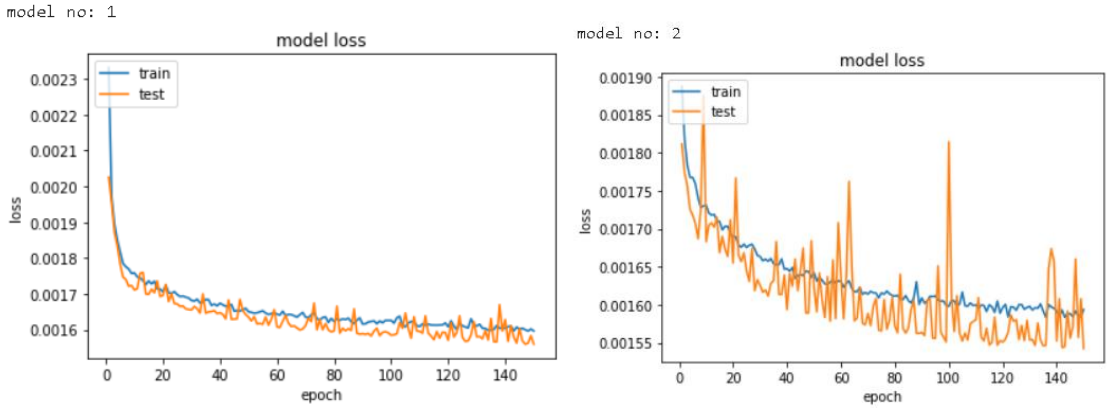
epochs: [5, 10, 50, 75, 100, 150, 200, 350, 500, 750, 1000, 1500, 2000]

learning_rate: [0.0001, 0.001, 0.01, 0.1, 1, 10]

II. Results description

1. Explanation of results and the impact of each hyperparameter.

After (‘3.2 random search cv’), the top 5 models with the highest MAPE score were plotted to visualize training and testing loss (mean_absolute_error loss) in **(‘3.3 model testing to find the best initial model’)**. Model no 1 was chosen as an initial configuration because it had the highest MAPE accuracy of the validation score (72.13) compared to the rest, didn’t overfit (testing loss was slightly less than training loss along the process), and the testing loss didn’t fluctuate, and surpass the training loss along the process like model no 2, resulting in the inconsistency of the prediction (testing loss) of the model as shown in picture 1 right.



Picture 1: model no 1 and 2 loss plot in '3.3 model testing to find the best initial model'

Then ('4.1 all hyperparameters vs accuracy'), each hyperparameter was varied with the rest being fixed with the initial configuration mentioned above (model no 1) to see the impact of each of them on the MAPE accuracy of the models. From the results, it was obvious that 'loss' must be mean_absolute_error since it had the highest accuracy (more suitable for regression task compared to the rest loss function, reference). So, we fixed the loss to be this value. Note that the graph results of this step will be used in the section below.

After this, ('4.2 phase 1: random search for the optimizer, activation, and learning rate'), the optimizer, activation function, and learning rate were varied using RandomsearchCV (the same process as '3.2 random search cv'). Regarding values selected in this phase, 0.0001 and 0.001 were the possible best learning rates because they showed high accuracy compared to the rest values (the more the learning rate than these, the more models learn faster and miss to capture important features resulting in underfitting, but the less learning rate than these would result in overfitting since models perform well in training dataset by capturing too much information but don't generalize well in the unknown dataset). Relu and Tanh were the possible best activation functions since their boundary decision to map input into output were suitable for this specific regression task. Adam, RMSprop, and Adamax were the possible best optimizer because their functions to adjust weights were suitable for this specific regression task. Model no 5 was chosen as the new initial hyperparameters' configuration to be used in the next section below because of the same reason mentioned in picture 1 **('3.3 model testing to find the best initial model').**

The same process of hyperparameters plotting with the models' accuracy continued in ('4.3 phase 1: hyperparameters plotting for the selected model') for choosing the best possible hyperparameters which hadn't been optimized. In this second phase (4.4 phase 2: random search for first_layer_nodes, and n_layers ''), first_layer_node, and n_layers were optimized. The possible best values of them were [15, 20, 30, 40, 50, 60, 70, 80] (first_layer_node below 15 didn't have enough representation power to generalize the model resulting in underfitting), and [2,3,5,7,9] (n_layers above 9 were overfitting since the model were too complex with many hidden layers to capture too much information). The result of this optimization was model no 5 being selected as the new initial hyperparameters' configuration to be used in the next section below (same reason mentioned in picture 1 **'3.3 model testing to find the best initial model'**).

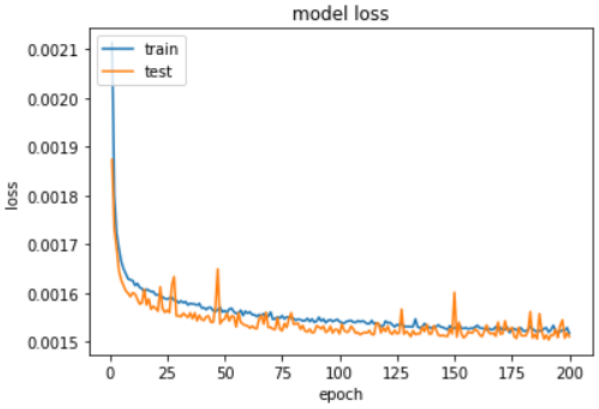
In the final phase of hyperparameters tuning (‘4.5 phase 2: hyperparameters plotting for the selected model’, and ‘4.6 phase 3: random search for epochs, and batch size’), epochs and batch size were optimized. The possible best values of them were [50, 100, 150, 200, 250, 300, 350, 400, 450, 500] (epochs below 50 were underfitting because models didn’t iterate enough through the dataset to capture important features. Whereas epochs above 500 iterated too much through the training dataset resulting in high performance in training but didn’t generalize well in the testing dataset), and [8, 16, 32, 64, 128, 256, 512] (batch sizes above 512 tended to lead to training instabilities and the model may not generalize well. In contrast, batch sizes below 8 showed an accuracy drop. This may be because those batch sizes didn’t perform well with the specified learning rate which was 0.0001 in this case). Finally, model no 1 was chosen to be the best one (same reason mentioned in picture 1 **‘3.3 model testing to find the best initial model’**).

2. Accuracy result of the final model

This was model no 1 in the final phase of hyperparameters tuning mentioned above. The result was in (‘5. the best model’) with the hyperparameters, MAPE accuracy, and loss plot of both training and testing process as shown in picture 2 below.

```
optimizer = 'Adamax'
first_layer_nodes  = 70
last_layer_nodes  = 1
n_layers = 3
activation = 'relu'
loss = "mean_absolute_error"
batch_size = 64
epochs = 200
learning_rate = 0.0001
```

```
train_acc:  74.37874356745812
test_acc:  74.17043407233253
```




```
final train loss:  0.0015173912979662418
final test loss 0.0015107550425454974
```

Picture 2: Hyperparameters (left) and, training and testing result of the final model (right)

III. Final Conclusion

1. Challenges of the project

There were four big challenges in the project. First, the dataset contained many invalid values. For instance, the number of bathrooms and floors had float values. This didn’t make any sense regarding features’ definition. So, we decided to drop and change the float values of some specific features to an integer. The data extraction process would be more accurate if we had more information about the data. Second, picking the right features as inputs for the model is one of the most important steps to take, even more important than the model itself. Eliminating irrelevant features reduces the model’s complexity and running time. Thus, we used the correlation score to select features. The higher the absolute value of the score, the more closely this feature and the target are linked. However, there is no clear threshold of this

score to filter out features (in this case was filtering out the features with absolute scores lower than 0.2). Third, choosing the appropriate structure of the neuron network for specific tasks is difficult. In this case, to find the best one for this regression task, many configurations had been tested such as numbers of network layers, and numbers of starting nodes (which didn't guarantee to be the best one). Fourth, regarding hyperparameters tuning, the search space of each hyperparameter might be infinitive and it is hard to define the possible best range of each hyperparameter. For instance, the range of the number of nodes or hidden layers could be from 1 to infinity (excluding hardware limits) resulting in high consumption of CPU/GPU power and time.

For future work, the total run time could be decreased significantly if our computers had higher computational power. The model could obtain more accuracy if we had more information to select the key features. Different tuning tools produced different accuracy. Therefore, in the future, we should try different tuning tools to find the optimal hyperparameters tuning library for this type of model.

2. Task allocation

Task	Person in charge	Deadline for each task
Data Cleaning	Saeth and Huy	22/03 – 26/03
Feature Selecting	Saeth	22/03 – 26/03
Data training and testing	Saeth and Huy	27/03 – 30/03
Hyperparameter tuning	Saeth	01/04 – 04/04
Plot the results	Saeth	05/04 – 06/04
Report writing	Saeth and Huy	06/04 – 11/04