**Coursework 2: Artificial Neural Networks**

**Group 12 Report**

**Model Structure**

1. Data Pre-processing
2. Given neural network requires inputs of numeric type, we start with processing categorical features into numeric features; for this data sample specifically, the only categorical feature is *ocean\_proximity*, which is a nominal categorical variable, therefore this feature is passed through label binarizer and gets converted into 5 columns of dummy variable.
3. We noted that *total\_bedroom* is the only feature with missing values for 1.02% of the total sample. Given the proportion of the missing value is not significant, the empty fields were filled with sample mean.
4. We also normalise the features with a minmax scaler. This normalization method is chosen as the features have bounded intervals and the distribution of features do not follow normal distribution, as observed from the chart below. Note that they are plotted with log scale on y-axis.

Chart, bar chart, histogram

Description automatically generated

1. Model Training
   1. input layer + 3 hidden layers + output layer

To obtain the optimal decision boundary, which is non-linear, we decided to use hidden layers. Also, multiple layers are better at generalizing because they could learn the intermediate features between the raw data and the final regression output.

* 1. activate functions

The activate function chosen for the output layer is ReLU, as the model is to predict for the median value of the houses of the block group, which is expected to be a non-negative continuous variable.

For activate functions in hidden layers, to model the non-linear relationships between features and target output variable, we chose among non-linear activate functions. The chart below illustrates how the MSE will converge as the number of epochs increases to 100. It shows that ReLU as the activation function allows the model to stabilise faster than Tanh or Sigmoid function. Therefore, we decided to use ReLU as activation functions for hidden layers.

Chart

Description automatically generated Chart, histogram

Description automatically generatedChart

Description automatically generated

* 1. dropouts

To mitigate possible overfitting, we also introduced dropouts in hidden layers.

**Evaluation Setup**

Given the final output “median house value” is a continuous variable, we use mean squared error (MSE) as the evaluation metrics.

**Results**

The entire dataset has been split into train : validate : test as 7 : 2 : 1; the validation set is to be used in the hyperparameter search.

We trained the model with the training set and default hyperparameters and noted the below performance:

* Chart, histogram

  Description automatically generatedscore on training set is 0.015376
* score on test set is 0.018059

To better contextualise the score, we inverse the prediction with the minmax normaliser, and visualise the difference between predicted median value and actual median value of the test set.

We noted the model performance suffers at both ends. This could be due to the fact that in the train, validate, test split used, the relative proportion of sample for cheaper houses is lower in training set than test set.

**Hyperparameter Search**

The hyperparameter search includes a grid search of 7 parameters:

* *nb\_epoch*: number of epochs used to train a model
* *batch\_size*: number of training samples to work through before the model’s internal parameters are updated
* *layer1/2/3*: number of neurons used in the 1st / 2nd / 3rd hidden layer; as the number of neurons increases, the layer has more flexibility to capture non-linear relationship.
* *prob*: probability of dropout; while a higher probability is more effective in mitigating overfit, it increases the possibility of underfit.
* *learning\_rate*: learning rate used in optimising for weights in the model
* In the hyperparameter search, the model with different settings of hyperparameters are fitted with train data set, and gets ranked based on their score, i.e. MSE from validation set. See below for an excerpt of the models with top 20 performance(as measured by mse from validation dataset.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **nb\_epoch** | **batch\_size** | **layer1** | **layer2** | **layer3** | **prob** | **lr** | **mse** |
| 500 | 1000 | 36 | 24 | 12 | 0.1 | 0.005 | 0.013201302 |
| 500 | 1000 | 36 | 24 | 12 | 0.1 | 0.001 | 0.013439043 |
| 500 | 500 | 36 | 24 | 12 | 0.1 | 0.005 | 0.01354664 |
| 500 | 500 | 24 | 24 | 6 | 0.1 | 0.001 | 0.013720992 |
| 500 | 500 | 36 | 12 | 6 | 0.1 | 0.001 | 0.013823677 |
| 500 | 1000 | 24 | 24 | 12 | 0.1 | 0.005 | 0.013827285 |
| 500 | 500 | 36 | 12 | 12 | 0.1 | 0.005 | 0.013832372 |
| 500 | 500 | 36 | 12 | 12 | 0.1 | 0.001 | 0.013927569 |
| 500 | 2000 | 36 | 24 | 6 | 0.1 | 0.01 | 0.013969643 |
| 500 | 2000 | 36 | 24 | 3 | 0.1 | 0.01 | 0.013981701 |
| 500 | 500 | 24 | 24 | 12 | 0.1 | 0.001 | 0.014007616 |
| 500 | 500 | 36 | 24 | 12 | 0.1 | 0.01 | 0.014009827 |
| 500 | 500 | 36 | 24 | 3 | 0.1 | 0.001 | 0.014044184 |
| 250 | 1000 | 24 | 12 | 6 | 0.1 | 0.005 | 0.014045978 |
| 500 | 1000 | 24 | 24 | 12 | 0.1 | 0.001 | 0.014051472 |
| 250 | 2000 | 36 | 12 | 6 | 0.1 | 0.01 | 0.014060537 |
| 500 | 1000 | 36 | 12 | 6 | 0.1 | 0.001 | 0.014134277 |
| 500 | 500 | 24 | 24 | 3 | 0.1 | 0.001 | 0.014142629 |
| 500 | 1000 | 24 | 12 | 6 | 0.1 | 0.005 | 0.014249532 |
| 500 | 1000 | 36 | 12 | 12 | 0.1 | 0.01 | 0.014309312 |

**Final Evaluation**

Chart, histogram

Description automatically generatedWe choose the optimal regressor from above and test its performance on the test set, which evaluated with MSE as 0.0136244. To better contextualise the score, we inverse the prediction with the minmax normaliser, and visualise the difference between predicted median value and actual median value of the test set.

This improvement could be attributed largely to the increase in number of neurons in the hidden layer – as the relationship embodied in input features were decomposed into more neurons, allowing neurons to learn the relationships at more dimensions.

Also, as the number of epochs increases, the neural network can learn more from the training data, resulting in improved accuracy and lower MSE. However, it's important to note that increasing the number of epochs beyond a certain point may lead to overfitting