DEEP LEARNING ASSIGNMENT 1

Roll Number 25280097
Name: Namra Nadeem

Link to Github repository:

https://github.com/namranadeem-wq/Deep-Learning-Assignment-1/tree/main/AI%20600%20Assignment

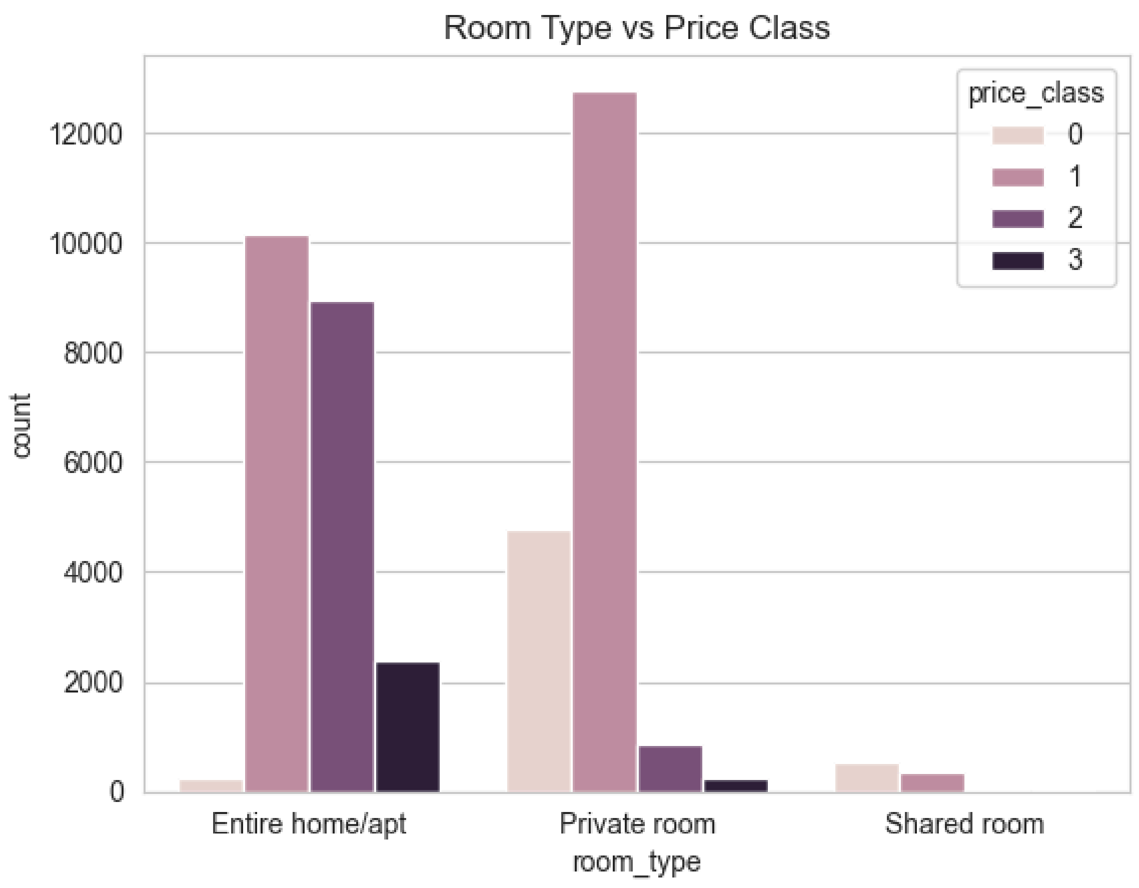
PART A

Exploratory Data Analysis: Feature Relationships and Insights

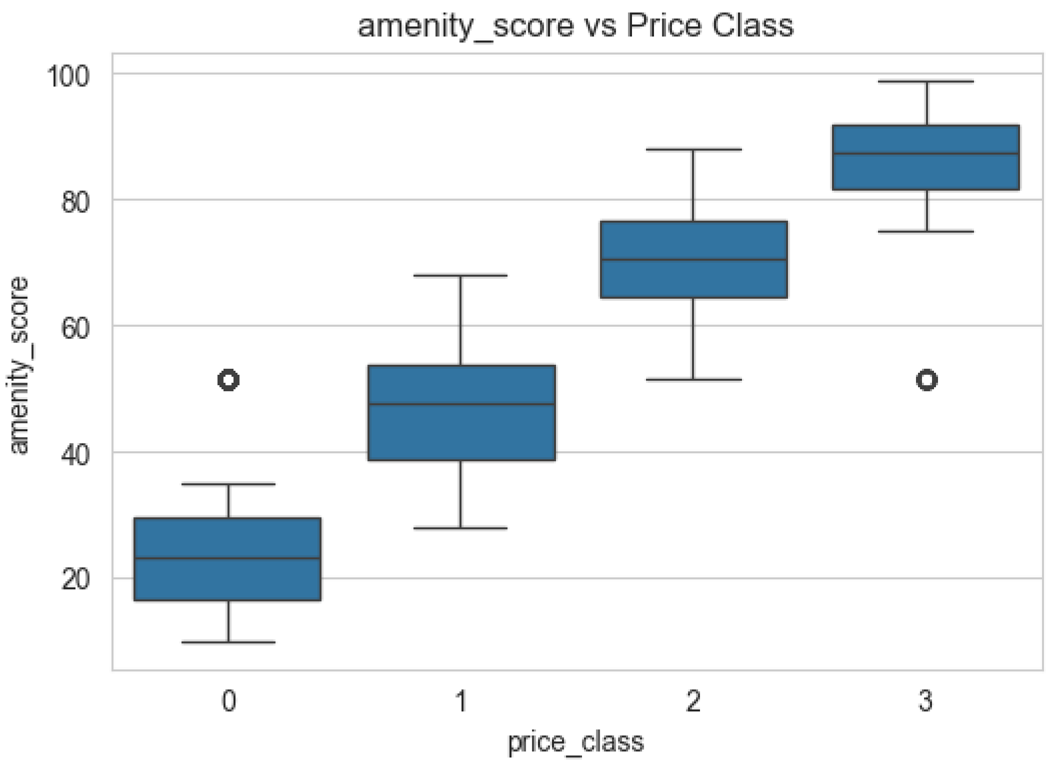
Several visualizations were used to analyze the relationships between input features and the target variable (price_class), along with the correlations among numerical features.

The class distribution analysis showed that the dataset is imbalanced. Moderate listings constitute approximately 56% of the dataset, Premium listings about 24%, Budget listings around 13%, and Luxury listings about 6%. This imbalance may influence model training, as the neural network may bias predictions toward majority classes.

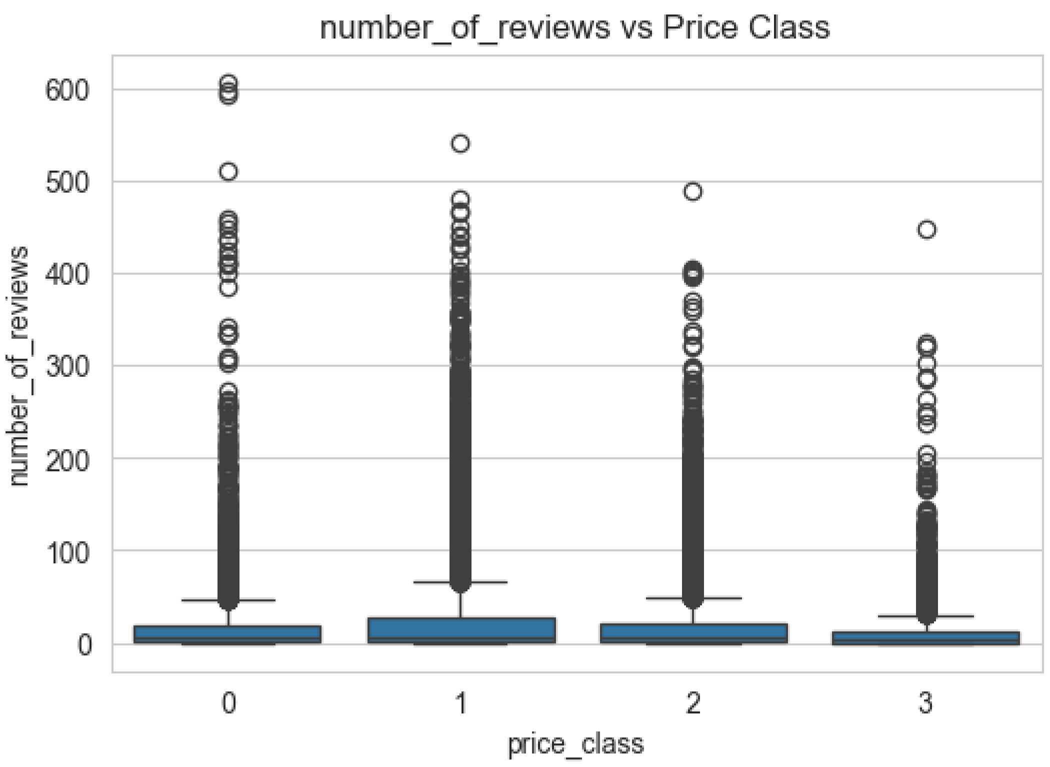
Class Number	Price Category	Count	Percentage
0	Budget	5567	~13.5%
1	Moderate	23287	~56.3%
2	Premium	9844	~23.8%
3	Luxury	2650	~6.4%


Room Type vs Price Class
count
12000
10000
8000
6000
4000
2000
0
Entire home/apt
Private room
Shared room
room_type
price_class
0
1
2
3

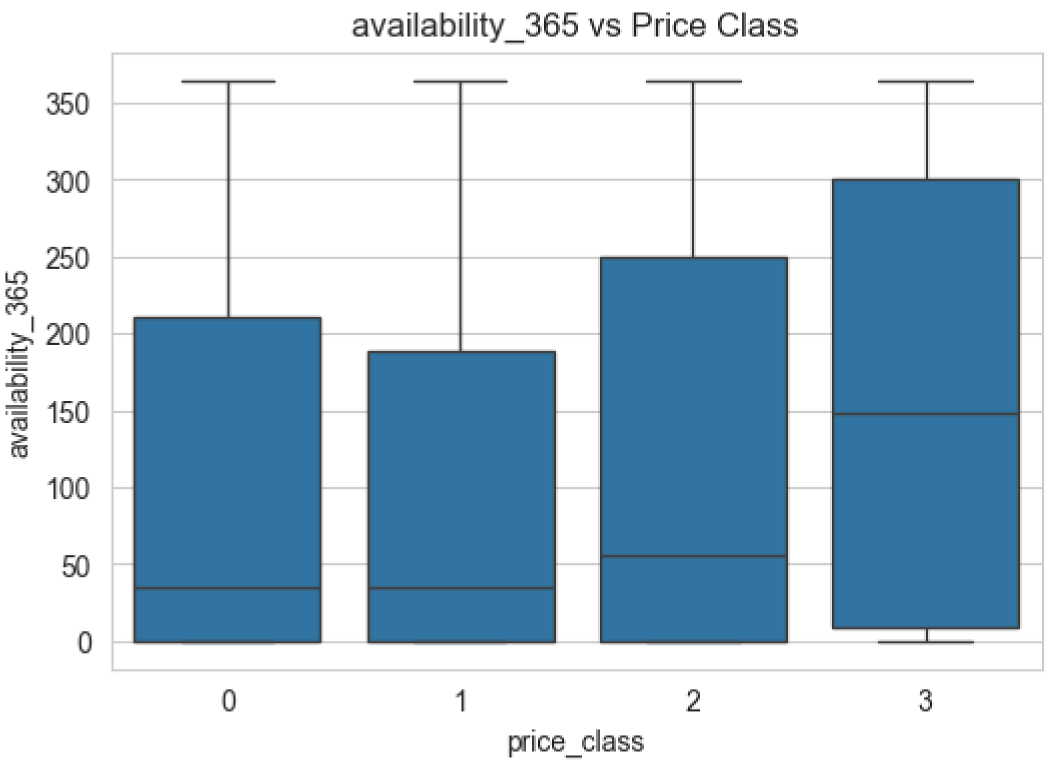
The boxplot of amenity_score versus price_class revealed a clear increasing trend in median values as the price category increases. Budget listings generally have low amenity scores, while Luxury listings have the highest scores. Although some overlap exists, the separation between distributions indicates that amenity_score is a strong predictor of price category and is expected to play a significant role in model predictions.



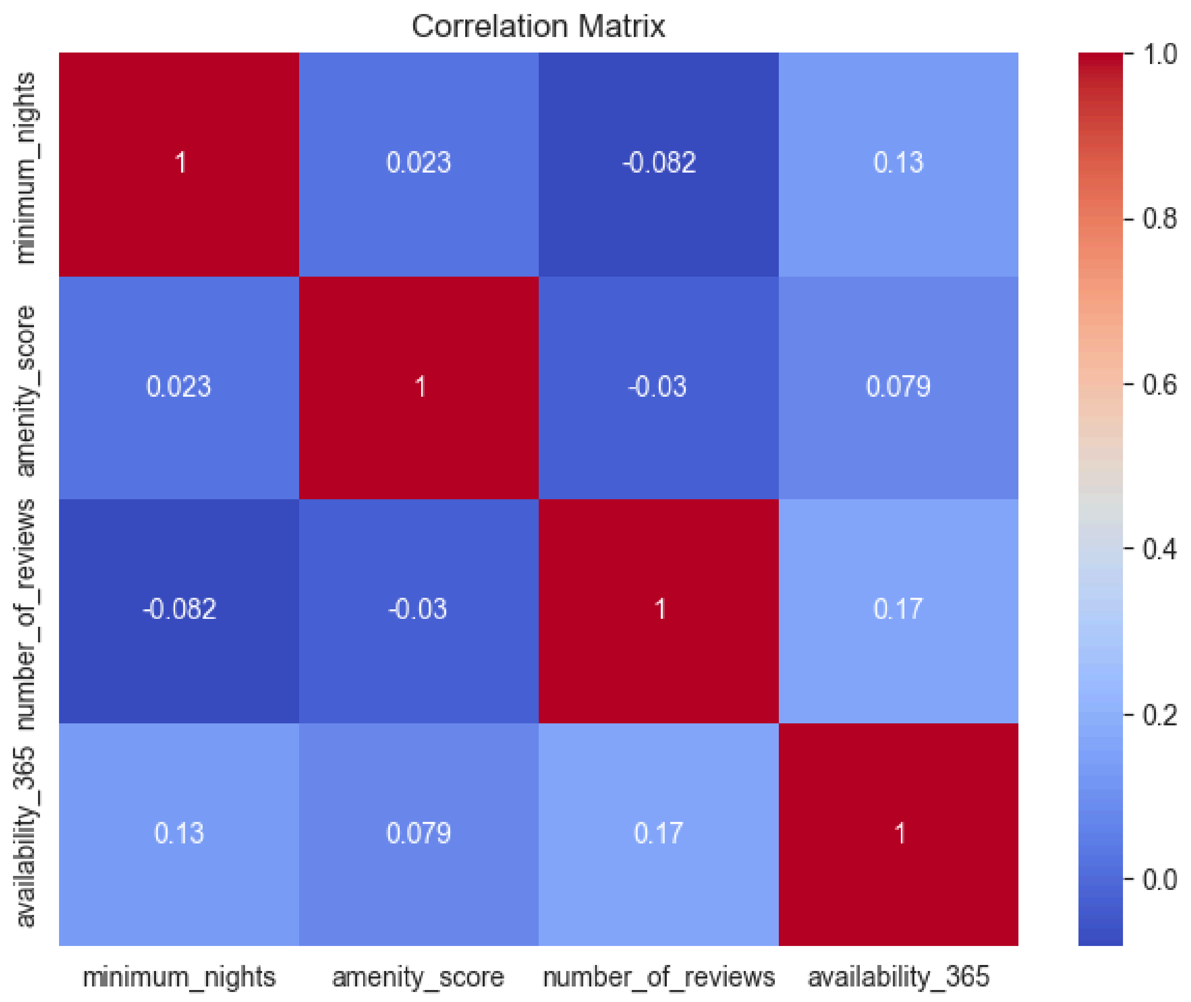
In contrast, the number_of_reviews versus price_class showed substantial overlap among all classes. Most listings in every category have relatively low numbers of reviews, with a small number of listings having very high counts, resulting in many outliers. The absence of a clear trend suggests that number_of_reviews is a weaker predictor and likely provides only supplementary information to the model.



The boxplot of availability_365 versus price_class demonstrated considerable variability across all classes. Luxury listings showed a slightly higher median availability, but large overlaps were observed among categories. This indicates that availability_365 may contribute moderately to prediction but is unlikely to be a dominant feature on its own. This feature primarily reflects hosting patterns and booking frequency rather than price directly.



The correlation matrix of numerical features indicated generally weak relationships among variables, with all correlation coefficients below 0.2 in magnitude. The highest correlation was observed between number_of_reviews and availability_365, but this relationship was still weak. The low correlations suggest minimal multicollinearity, meaning each feature provides distinct information useful for prediction.



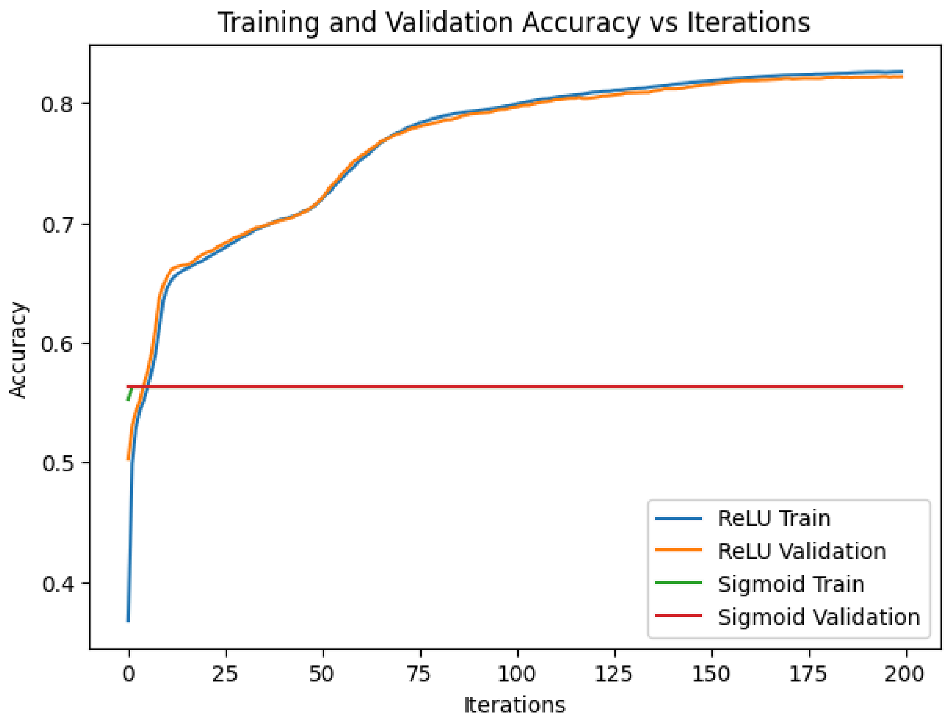
Overall, the exploratory analysis suggests that amenity_score, room_type, and neighbourhood_group are likely to be the most influential predictors of price_class. Availability_365 and number_of_reviews provide additional contextual information, while minimum_nights may have less consistent predictive influence due to its high variability and presence of extreme outliers. These observations indicate that the prediction task requires combining multiple features, making it well suited for a feedforward neural network.

PART B (a)

Using batch gradient descent for 200 iterations, the ReLU-activated MLP achieved a final training accuracy of 0.826 and validation accuracy of 0.822, improving consistently as training progressed. In contrast, the sigmoid-activated MLP stagnated at approximately 0.563 training/validation accuracy throughout training, indicating failure to optimize effectively. The stagnant sigmoid accuracy is close to the majority-class proportion (~56.3%), suggesting the model collapsed to predicting the dominant class rather than learning discriminative decision boundaries. Sigmoid activations can suffer from vanishing gradients due to saturation, reducing gradient flow through earlier layers and slowing or preventing learning. ReLU maintains stronger gradients for positive activations, enabling more stable optimization and faster convergence.

PART B (b)

The average gradient magnitudes of the first and second hidden layer weights were monitored during training. For the ReLU network, gradients remained stable across both layers and the model showed consistent improvement in training and validation accuracy. In contrast, the sigmoid network exhibited gradient magnitudes of similar order, but the model failed to improve accuracy, indicating ineffective learning. This behavior can be explained by the saturation property of the sigmoid activation function, where gradients become less informative and weight updates produce minimal change in activations. Overall, the results demonstrate that ReLU provides more effective gradient flow and faster convergence in deep networks compared to sigmoid activation. These observations confirm that activation function choice significantly influences optimization dynamics and gradient propagation in multi-layer neural networks.



Date: Part C(a) Mon Tue Wed Thu Fri Sat

For a two hidden layer neural network, the forward pass can be written as

$$z_1 = Wx + b_1$$
$$a_1 = g(z_1)$$
$$z_2 = W_2 a_1 + b_2$$
$$a_2 = g(z_2)$$

$$z3 = W_3 2 + b_3$$
$$\hat{y} = \text{softmax}(z_3)$$
$$L = l(\hat{y}, y)$$

Using chain rule, gradients propagate backward through each layer

gradient at output layer: $\delta_3 = \frac{\partial L}{\partial z_3} = \hat{y} - y$

gradient at second hidden layer: ~~zd~~

~~$\delta_2 = \frac{\partial L}{\partial z_3}$~~

$$\delta_2 = (W_3^T \delta_3) \odot g'(z_2)$$

gradient at first hidden layer:

$$\delta_1 = (W_2^T \delta_2) \odot g'(z_1)$$

Date: gradient with respect to input: Mon Tue Wed Thu Fri Sat

$$\frac{\partial L}{\partial x} = W_1^T \delta_1$$

This final expression gives the senstivity of the loss to each input feature.

Pseudocode:

```
- perform a forward pass to compute predition ad loss
- compute gradients of loss using back propagation
- compute gradients with respect to input features
- compute the absolute magnitude of each gradient
- Average gradient magnitudes across all samples.
- Rank features based on average gradient magnitude.
```

The magnitude of $\left|\frac{\partial L}{\partial x_i}\right|$ indicates how much the loss changes when feature x_i changes slightly. A large gradient means that small changes in that feature strongly affect the model's predition, indicatig high importance. From an optimization perspective, gradients determine parameter updates during training. Features that strongly influence the loss will produce larger gradients, meaning the model is more sensitive to them.

PART C (b)

A comparable two-hidden-layer MLP was trained using PyTorch to enable gradient-based feature attribution. The model achieved a training accuracy of approximately 0.836 and a validation accuracy of approximately 0.826 after 20 epochs. The small gap between training and validation accuracy indicates stable learning and reasonable generalization, making the trained model suitable for feature attribution analysis.

Gradient-based feature attribution was performed by computing the average magnitude of the gradient of the loss with respect to each input feature. The results indicate that amenity_score is the most influential feature, with a substantially larger gradient magnitude than all other variables. This finding is consistent with the exploratory data analysis, where amenity_score showed clear separation across price classes.

```
amenity_score 0.004634509
room_type_Shared room 0.0012950944
room_type_Private room 0.001006806
neighbourhood_group_Manhattan 0.0008542189
neighbourhood_group_Queens 0.00058581296
neighbourhood_group_Brooklyn 0.00051855086
minimum_nights 0.00044069
neighbourhood_group_Staten Island 0.0004104523
number_of_reviews 0.00026537434
availability_365 0.000225434
```

Categorical variables representing room type and neighbourhood group were also among the most influential predictors, which aligns with domain knowledge that accommodation type and location strongly affect pricing. Features such as minimum_nights, number_of_reviews, and availability_365 exhibited smaller gradient magnitudes, indicating comparatively lower influence on model predictions.

Overall, the feature attribution results closely match the patterns observed in Part A, confirming that the model relies primarily on amenity quality, accommodation type, and location when predicting price categories.

The agreement between gradient-based attribution and exploratory data analysis increases confidence that the model is learning meaningful relationships rather than relying on spurious correlations.

The gradient-based feature attribution analysis provides additional insight into how the neural network makes predictions. The results confirm that the model relies primarily on amenity quality, accommodation type, and neighbourhood when determining price categories, which is consistent with the patterns observed during exploratory data analysis in Part A. This consistency increases confidence that the model is capturing meaningful relationships in the data rather than overfitting to noise or irrelevant features.

Having identified the most influential features and verified that the model behavior aligns with domain expectations, the next step is to evaluate the trained model on the held-out test dataset and analyze its generalization performance. This evaluation will help determine how well the learned patterns transfer to unseen data and whether any performance gap exists between training, validation, and test results.

Date: Mon Tue Wed Thu Fri Sat

Part C(b) Algorithm Gradient Based Feature Attribution

Pseudocode:

```
1. Train neural network model.
2. Initialize feature importance scores to zero.
3. For each batch of validation data
   • Enable gradient tracking on inputs.
   • perform forward pass and compute loss.
   • perform backward pass to compute gradients
   • Accumulate absolute gradient values for
     each feature.
4. Average gradients across all samples.
5. Rank features in descending order of
   importance.
```

Part D: Test Evaluation and Generalization Analysis

The trained neural network achieved a training accuracy of approximately 0.836, a validation accuracy of approximately 0.826, and a test accuracy of 0.351. While the training and validation performances were similar, the significant drop in test accuracy indicates poor generalization to unseen data.

Exploratory data analysis in Part A showed class imbalance and overlapping feature distributions, which make classification more challenging and can reduce robustness on new data. Gradient-based feature attribution in Part C further revealed that the model relies heavily on amenity_score and a small set of categorical features such as room type and neighbourhood group. This strong reliance on a limited number of features may reduce the model's ability to generalize when feature distributions vary in the test dataset.

Overall, the performance gap suggests that the model learned patterns specific to the training distribution rather than fully generalizable relationships. Techniques such as regularization or class-balanced training could help mitigate this issue and improve performance on unseen data.

The model performs well during training because it learns strong correlations present in the training data, particularly those involving amenity quality and location. However, when evaluated on unseen data, slight differences in feature distributions and class proportions reduce prediction accuracy. This indicates that the model may have learned patterns that are specific to the training distribution rather than general patterns that hold across datasets.

One strategy to improve generalization would be to apply regularization techniques, such as dropout or weight decay, to reduce overfitting. Another effective approach would be to use class-weighted loss functions to address class imbalance, encouraging the model to learn more balanced decision boundaries across all price categories.

These results highlight the importance of analyzing model generalization in addition to training performance when evaluating neural networks on real-world tabular datasets.

Date: Question 2 Mon Tue Wed Thu Fri Sat

Shared Bias Gradient

Forward Pass:

$$h_1 = g(Wx + b)$$
$$\hat{y} = g_1(Uh_1 + b)$$

$$L = l(\hat{y}, y)$$

$$a_1 = Wx + b \qquad h_1 = g(a_1)$$
$$a_2 = Uh_1 + b \qquad \hat{y} = g_1(a_2)$$

step 1

$$\frac{\partial L}{\partial a_2} = \frac{\partial L}{\partial \hat{y}} \odot g_1'(a_2)$$

$$\delta_2 \triangleq \frac{\partial L}{\partial a_2} = \left(\frac{\delta l(\hat{y}, y)}{\delta \hat{y}} \right) \odot g_1'(a_2)$$

step 2 $a_2 = Uh_1 + b$

$$\frac{\partial L}{\partial h_1} = U^T \delta_2$$

step 3 since $h_1 = g(a_1)$

$$\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial h_1} \odot g'(a_1)$$

substitute step 2

$$\delta_1 \triangleq \frac{\partial L}{\partial a_1} = (U^T \delta_2) \odot g'(a_1)$$

Date: Bias b appears in two equations

$a_1 = Wx + b$

$a_2 = Uh_1 + b$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial b} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial b}$$

$$\frac{\partial a_1}{\partial b} = I \qquad \frac{\partial a_2}{\partial b} = I$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a_1} + \frac{\partial L}{\partial a_2}$$

shared bias : $$\boxed{\frac{\partial L}{\partial b} = \delta_1 + \delta_2}$$

Independent bias

$$\frac{\partial L}{\partial b_1} = \delta_1$$

$$\frac{\partial L}{\partial b_2} = \delta_2$$

Yes, bias sharing affects training because

- 1 bias b is used in both layers → parameter coupling
- Gradient update becomes $\delta_1 + \delta_2$ (two path sum) - changes update magnitude and conditioning
- This can slow convergence compared to separate biases b_1, b_2 when each layer needs different bias shifts.
- May require smaller learning rate for stability.

Date: Modified Backdrop: Mon Tue Wed Thu Fri Sat

1. Forward definitions

$$a_1 = Wx + b \quad (\in R^m)$$

$$h_1 = g(a_1)$$

$$a_2 = Uh_1 + b \quad (\in R^m \text{ since } m = k)$$

$$\hat{y} = g_1(a_2)$$

$$L = l(\hat{y}, y)$$

Path A : $b \to a_1 \to h_1 \to a_2 \to \hat{y} \to L$

Path B : $b \to a_2 \to \hat{y} \to L$

total gradient is the sum of both paths

Backdrop step by step (chain rule)

Step 1: Output preactivation gradient

$$\frac{\partial L}{\partial a_2} = \frac{\partial L}{\partial \hat{y}} \odot g_1'(a_2)$$

$$\delta_2 ~~=~~ \triangleq ~~\frac{\partial L}{\partial a_2}~~ \frac{\partial L}{\partial a_2}$$

since $a_2 = Uh_1 + b$ and $h_1 = g(a_1)$

$$\frac{\partial L}{\partial h_1} = U^T \delta_2$$

$$\delta_1 \triangleq \frac{\partial L}{\partial a_1} = U^T \delta_2 \odot g'(a_1)$$

Date: Mon Tue Wed Thu Fri Sat

From chain rule

$$\frac{\delta L}{\delta b} = \frac{\delta L}{\delta a_1}\frac{\delta a_1}{\delta b} + \frac{\delta L}{\delta a_2}\frac{\delta a_2}{\delta b}$$

$a_1 = Wx + b$

$$\frac{\delta a_1}{\delta b} = I$$

$a_2 = Uh_1 + b$

$$\frac{\delta a_2}{\delta b} = I$$

so

$$\frac{\delta L}{\delta b} = \frac{\delta L}{\delta a_1} + \frac{\delta L}{\delta a_2}$$

$$\boxed{\frac{\delta L}{\delta b} = \delta_1 + \delta_2}$$

Independent biases $\nabla_{b_1} L = \delta_1$

$\nabla_{b_2} L = \delta_2$

shared biases

compute δ_1 and δ_2

$$\nabla_b l = \delta_1 + \delta_2$$