AI600 Assignment 1

Student Name: Danial Maqbool
Student ID: 25280029
Date: February 17, 2026

Question No 1

Part A

1. Dataset Structure

The dataset consists of 41348 samples and 7 features. The data types include only 2 categorical variables (neighbourhood_group, room_type) while rest are numerical variables (minimum_nights, amenity_score, etc.). The target variable is 'price_class'.

2. Missing Values Analysis & Strategy

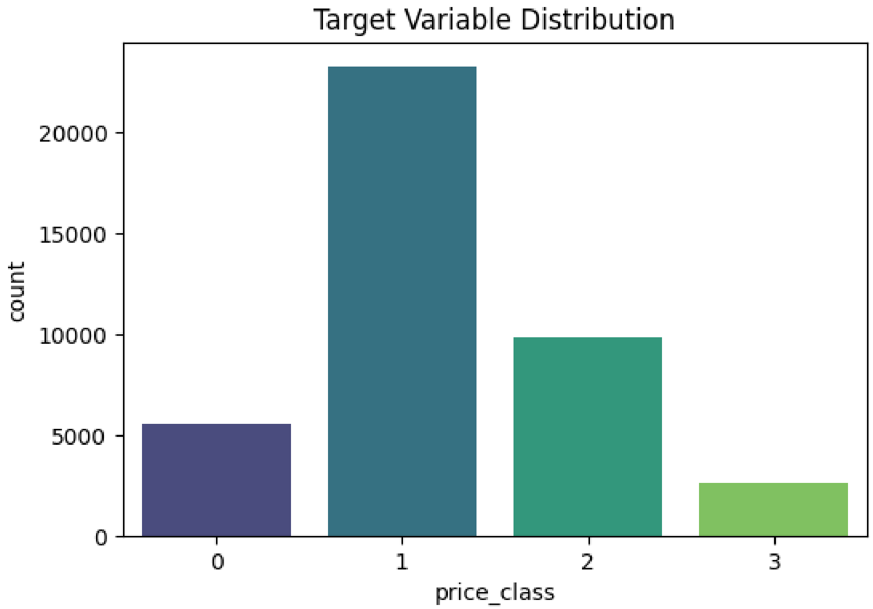
The following features contained missing values:

Feature	Missing Count
neighbourhood_group	839
room_type	611
minimum_nights	1322
amenity_score	916
number_of_reviews	1123
availability_365	595

Strategy Used: Imputation.

1. **Categorical Features:** Filled with the Mode (most frequent value). Justification: This preserves the probability distribution of categories.
2. **Numerical Features:** Filled with the Median. Justification: the mean can be effected greatly by the outlier or extreme values while median stays, more or less, the middle value of the dataset and strong against the outliers.

3. Class Distribution Analysis



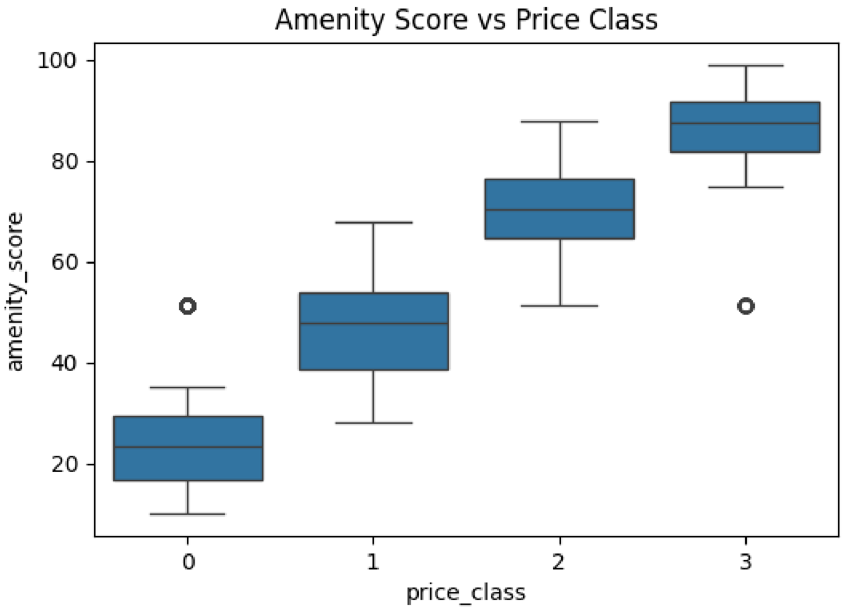
Observation: The dataset is highly imbalanced. Class 1 (Moderate) accounts for 56.3% of the data, while Class 3 (Luxury) accounts for only 6.4%. This suggests the model may be biased towards the majority class if not handled correctly.

4. Encoding & Normalization

Encoding Strategy: One-Hot Encoding was used for 'neighbourhood_group' and 'room_type', because these are nominal categories with no inherent order. Label encoding (0,1,2) would give them false priorities like 2 is better than 1 but that’s not true for our case.

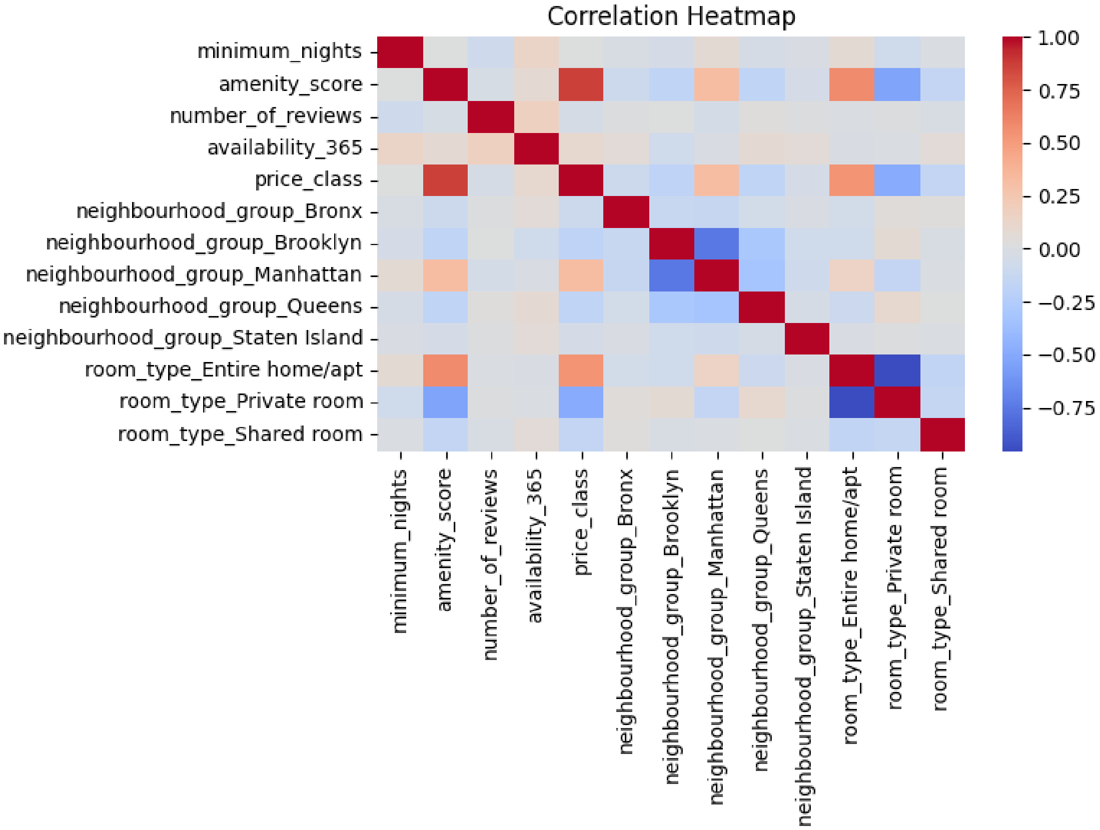
Normalization Strategy: Standard Scaling (Z-score) is used because features like 'availability_365' and 'minimum_nights' have very different ranges. Scaling makes sure that during backpropagation the weights are distributed much more smoothly and more quickly making the overall process much more efficient.

5. Feature Relationships



Analysis: The boxplot above shows a strong positive relationship between “Amenity Score” and “Price Class”. This means that as we move towards higher price class the amenity score is also moving towards higher values.

6. Correlation Analysis



Findings: The feature 'amenity_score' has the highest correlation with the target variable of ‘price_class’ (0.87). This feature is most dominant in sense that it has the greatest effect on the price_class, which is our target.

7. Conclusion

Based on the EDA, 'amenity_score' is the most influential feature for predicting our price_class target values. The 'Entire home/apt' room type is also a strong indicator of higher prices. The dataset imbalance, where class 1 (moderate) has very high frequency as compared to other classes in our dataset, is a critical factor that must be addressed during model training evaluation.

Part B

(a)

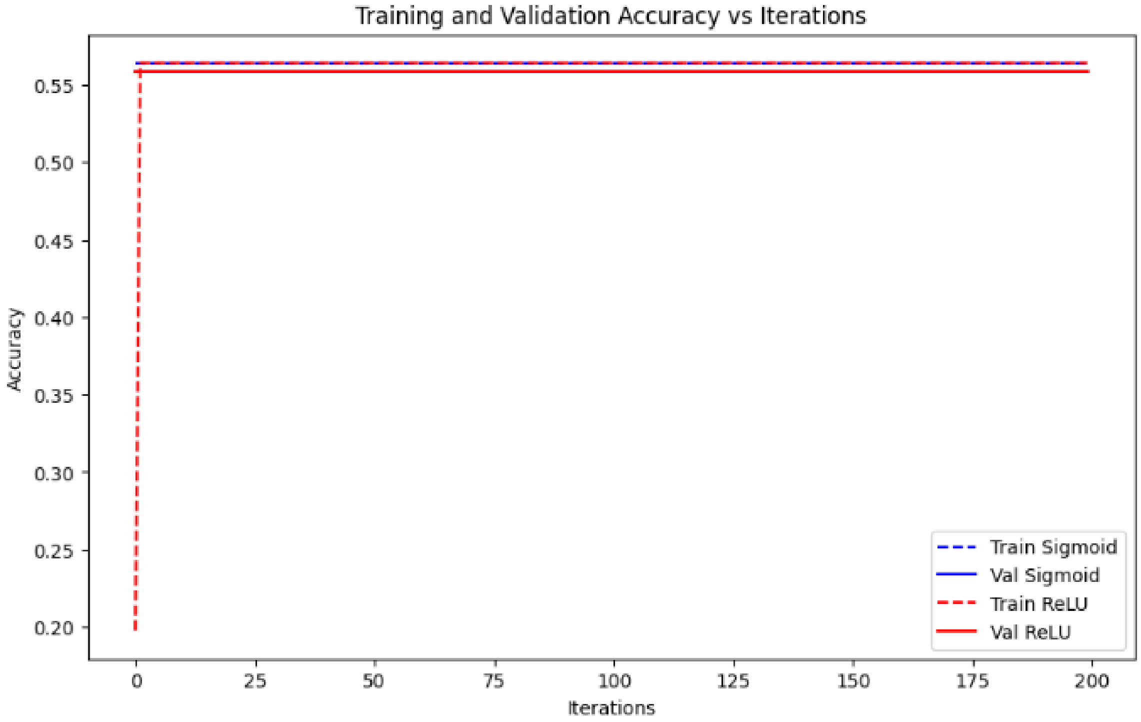
Results:

Sigmoid Train: 0.5642723260172925

Sigmoid Val: 0.5588875453446192

ReLU Train: 0.5642723260172925

ReLU Val: 0.5588875453446192



Interpretation:

The multiplicative decay of gradients (vanishing gradients) of the Sigmoid function makes deep network optimization more difficult. By maintaining gradient magnitude, ReLU improves optimization and allows the network to learn better representations of features in lower layers. The accuracies are approximately 56% which is almost same as the proportion of class 1 (moderate). This shows that the model is converging to a local minima and there it is just predicting the majority class for all input. This is a common issue of batch/vanilla gradient descent method when applied on imbalanced class.

Part B

(b)

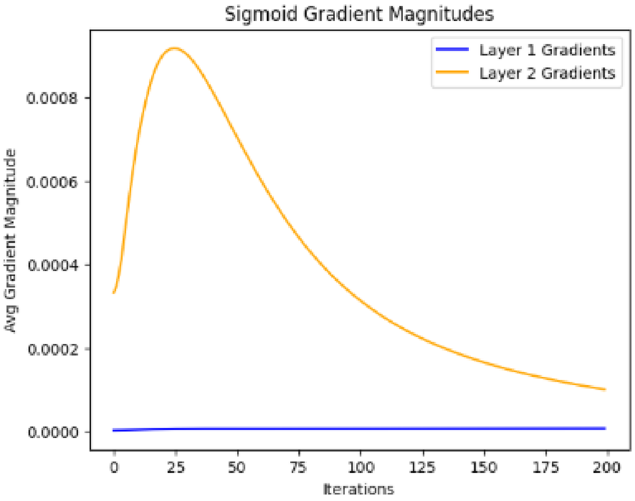
Results:

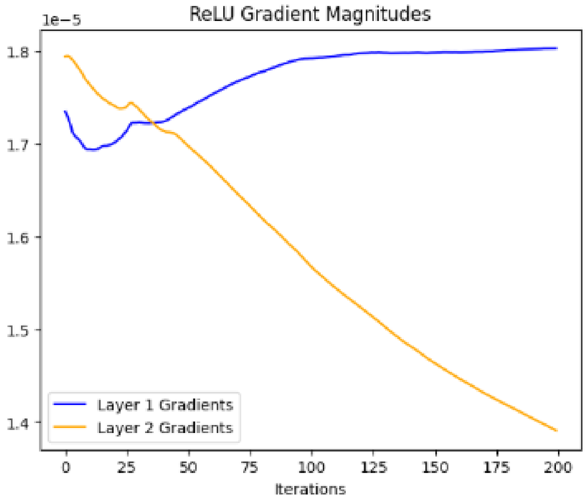
Sigmoid Layer 1 mean grad: 6.138980048976196e-06

Sigmoid Layer 2 mean grad: 0.0004027172747484585

ReLU Layer 1 mean grad: 1.770561040697503e-05

ReLU Layer 2 mean grad: 1.578877056375359e-05





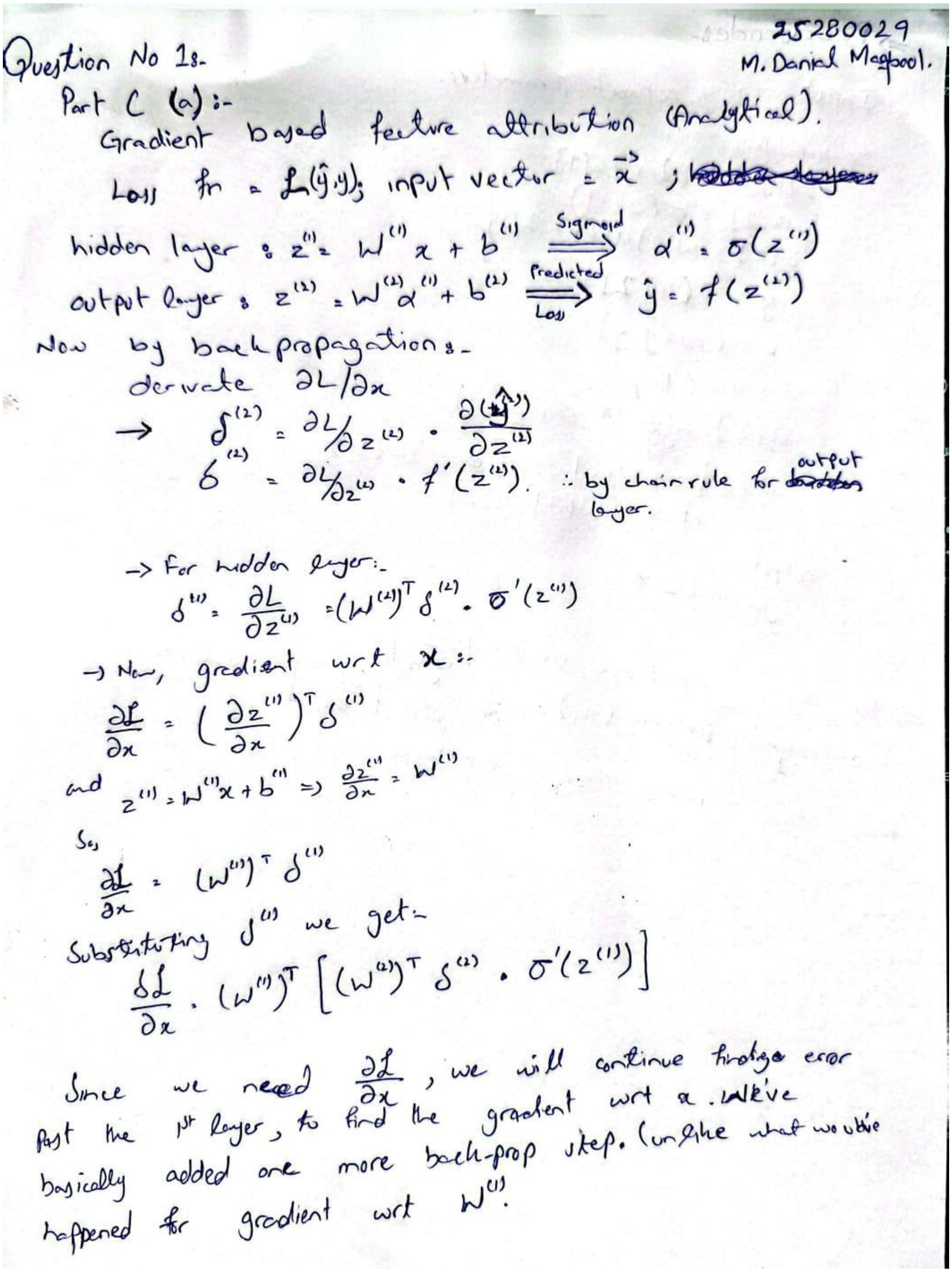
Analysis:

Sigmoid (Vanishing Gradient): During backprop, as we move from layer 2 to layer 1, we can see that there is a huge drop in the gradient’s magnitude. This shows the presence of the famous Vanishing Gradient Problem. This happens because the derivatives of sigmoid are very small (< 1) so multiplying these small derivatives across multiple layers causes the gradient to decrease/drop very quickly, making it very difficult for early layers to learn.

ReLu: In comparison to the Sigmoid, the ReLu gives similar order magnitude for each layer. The Relu preserves the gradient since its derivative either gives 1 or 0 as output preserving the decay unlike Sigmoid Function. Hence, ReLu gives more or less similar accuracies but since ReLu can provide a better gradient flow, making it comparatively superior for training deep neural networks.

Part C

(a)



25280029
M. Daniel Maqbool.

Question No 1:-

Part C (a):-

Gradient based feature attribution (Analytical).

Loss fn = $\mathcal{L}(\hat{y}, y)$; input vector = $\vec{x}$; ~~hidden layers~~

hidden layer : $z^{(1)} = W^{(1)} x + b^{(1)} \xRightarrow{\text{Sigmoid}} a^{(1)} = \sigma(z^{(1)})$

output layer : $z^{(2)} = W^{(2)} a^{(1)} + b^{(2)} \xRightarrow[\text{Loss}]{\text{Predicted}} \hat{y} = f(z^{(2)})$

Now by backpropagation:-

derivate $\partial L / \partial x$

→ $\delta^{(2)} = \partial L / \partial z^{(2)} \cdot \dfrac{\partial (\hat{y})}{\partial z^{(2)}}$

$\delta^{(2)} = \partial L / \partial z^{(2)} \cdot f'(z^{(2)})$. ∴ by chain rule for ~~hidden~~ output layer.

-> For hidden layer:-

$$\delta^{(1)} = \frac{\partial L}{\partial z^{(1)}} = (W^{(2)})^T \delta^{(2)} \cdot \sigma'(z^{(1)})$$

-> Now, gradient wrt x:-

$$\frac{\partial \mathcal{L}}{\partial x} = \left(\frac{\partial z^{(1)}}{\partial x}\right)^T \delta^{(1)}$$

and $z^{(1)} = W^{(1)} x + b^{(1)} \Rightarrow \dfrac{\partial z^{(1)}}{\partial x} = W^{(1)}$

So,

$$\frac{\partial \mathcal{L}}{\partial x} = (W^{(1)})^T \delta^{(1)}$$

Substituting $\delta^{(1)}$ we get:-

$$\frac{\delta \mathcal{L}}{\partial x} = (W^{(1)})^T \left[(W^{(2)})^T \delta^{(2)} \cdot \sigma'(z^{(1)})\right]$$

Since we need $\dfrac{\partial \mathcal{L}}{\partial x}$, we will continue through error past the 1st layer, to find the gradient wrt x. We've basically added one more back-prop step. (unlike what would've happened for gradient wrt $W^{(1)}$.

⇒ Pseudocodes-

```
Input : $x, y$, network parameters.
forward pass :
    $z[1] = w[1] x + b[1]$
    $a[1] = \sigma(z[1])$
    $z[2] = W[2] a[2] + b[2]$
    $\hat{y} = f(z[2])$
    $L = Loss(\hat{y}, y)$
  Backward Prop:-
    $\delta[2] = \frac{\partial L}{\partial \hat{y}} * f'(z[2])$
    $\delta[1] = (W[2]^T \delta[2] \odot \sigma'(z[1]))$
    gradient_x $= W[1]^T . \delta[1]$
  output:
    gradient_x.
```

Now

⇒ For x_i : importance $(x_i) = \left| \frac{\partial L}{\partial x_i} \right|$

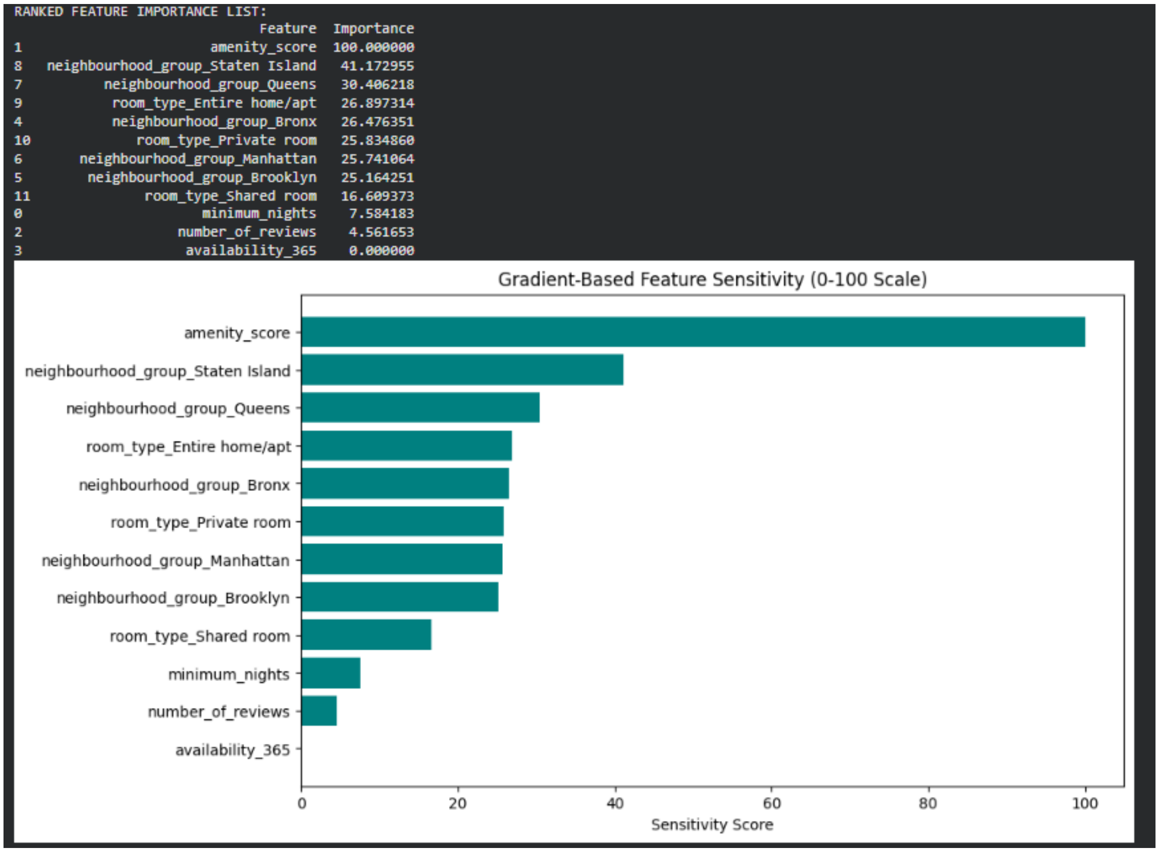
average importance $(x_i) = \frac{1}{N} \sum_{j=1}^{N} \left| \frac{\partial L^{(j)}}{\partial x_i} \right|$ ∴ N = number of Samples.

Now we can rank feature importance by avg magnitude.

Part C

(b)

Ranked features importance list:



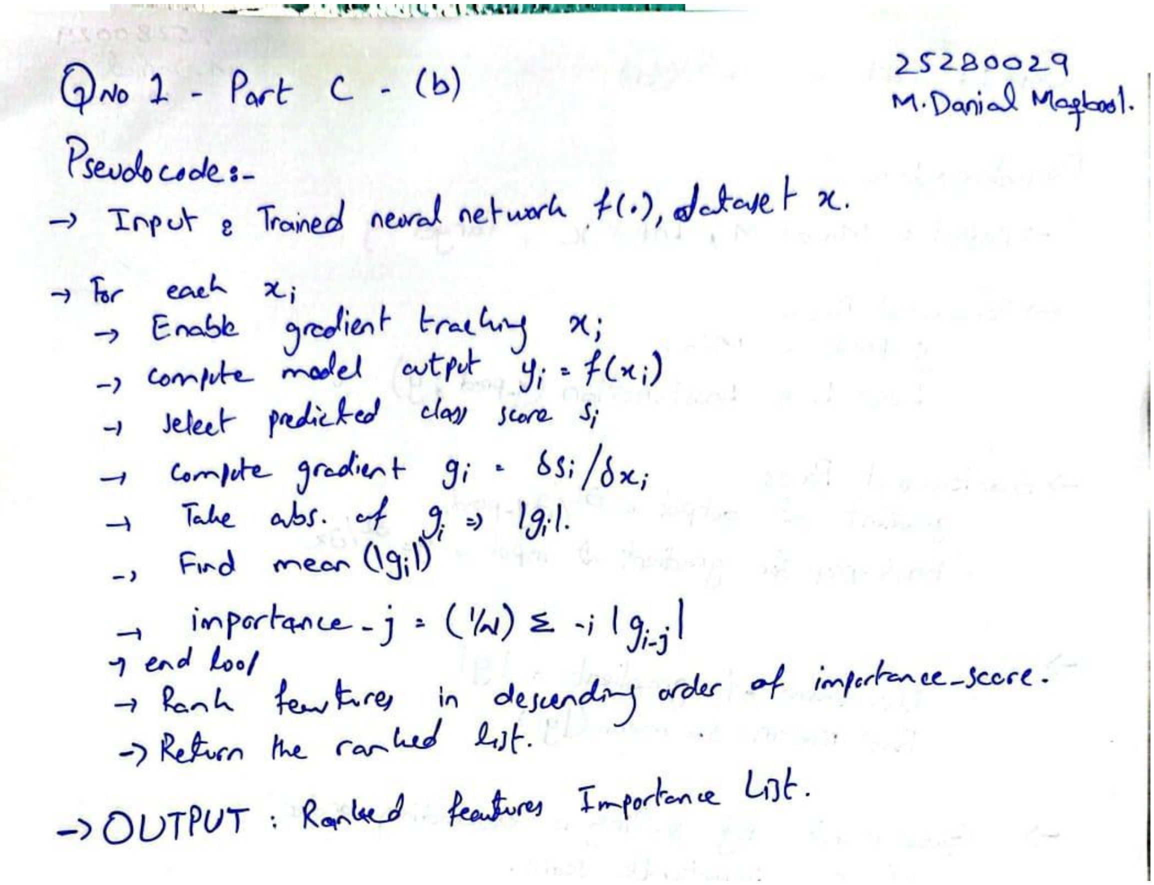
```
RANKED FEATURE IMPORTANCE LIST:
                          Feature  Importance
1                   amenity_score  100.000000
8   neighbourhood_group_Staten Island   41.172955
7         neighbourhood_group_Queens   30.406218
9           room_type_Entire home/apt   26.897314
4          neighbourhood_group_Bronx   26.476351
10            room_type_Private room   25.834860
6      neighbourhood_group_Manhattan   25.741064
5       neighbourhood_group_Brooklyn   25.164251
11             room_type_Shared room   16.609373
0                   minimum_nights    7.584183
2                number_of_reviews    4.561653
3                 availability_365    0.000000
```

Gradient-Based Feature Sensitivity (0-100 Scale)
amenity_score
neighbourhood_group_Staten Island
neighbourhood_group_Queens
room_type_Entire home/apt
neighbourhood_group_Bronx
room_type_Private room
neighbourhood_group_Manhattan
neighbourhood_group_Brooklyn
room_type_Shared room
minimum_nights
number_of_reviews
availability_365
0
20
40
60
80
100
Sensitivity Score

Comparison to Part A:

In part A during EDA, it can be seen that amenity score had highest correlation score with the price_class feature. Same thing is projected in the importance list that amenity score has a perfect (relative) score. Also, the availability_365 and number_of_reviews are almost 0 on importance scale which is against consistent with the correlation seen in part A.

Pseudocode:



Q No 1 - Part C - (b)

25280029
M.Danial Maqbool.

Pseudocode:-

```
-> Input : Trained neural network $f(\cdot)$, dataset $x$.

-> For each $x_i$
  -> Enable gradient tracking $x_i$
  -> compute model output $y_i = f(x_i)$
  -> select predicted class score $s_i$
  -> compute gradient $g_i = \delta s_i / \delta x_i$
  -> Take abs. of $g_i \Rightarrow |g_i|$.
  -> Find mean $(|g_i|)$
  -> importance_j = $(1/N) \Sigma_i |g_{i-j}|$
 -> end loop
 -> Rank features in descending order of importance_score.
 -> Return the ranked list.

-> OUTPUT : Ranked features Importance List.
```

Part D

Test Accuracies:

```
--- RESULTS ---

Final Train Accuracy: 84.33%

Final Val Accuracy:   82.32%

Test Dataset Accuracy: 34.14%


--- DIAGNOSTICS ---

Train Amenity Mean: 51.97422366257134

Test Amenity Mean:  54.15185692750446

Test Acc if 'amenity_score' is removed: 56.31%
```

Analysis of performance gap:

During training and validation, the results showed high accuracies and also the accuracies in training and validation sets were also very close to each other showing that there is no over-fitting of the trained model, but as soon as the model is applied to test dataset we can see a sharp drop in the accuracy, showing that there is some **spurious correlation** present in the training data but absent or different in our test set.

Root Cause Identification:

Evidence from Exploratory Data Analysis: in part A, I noticed that amenity_score had suspiciously high correlation with price_class so it was already kept under observation because it was potentially indicating that the score might be derived from the price itself.

Evidence from Gradient-Based Attribution: In part C, It became clear due to perfect 100 importance score given to amenity_score, while all other features were below 30 that it is a spurious correlation.

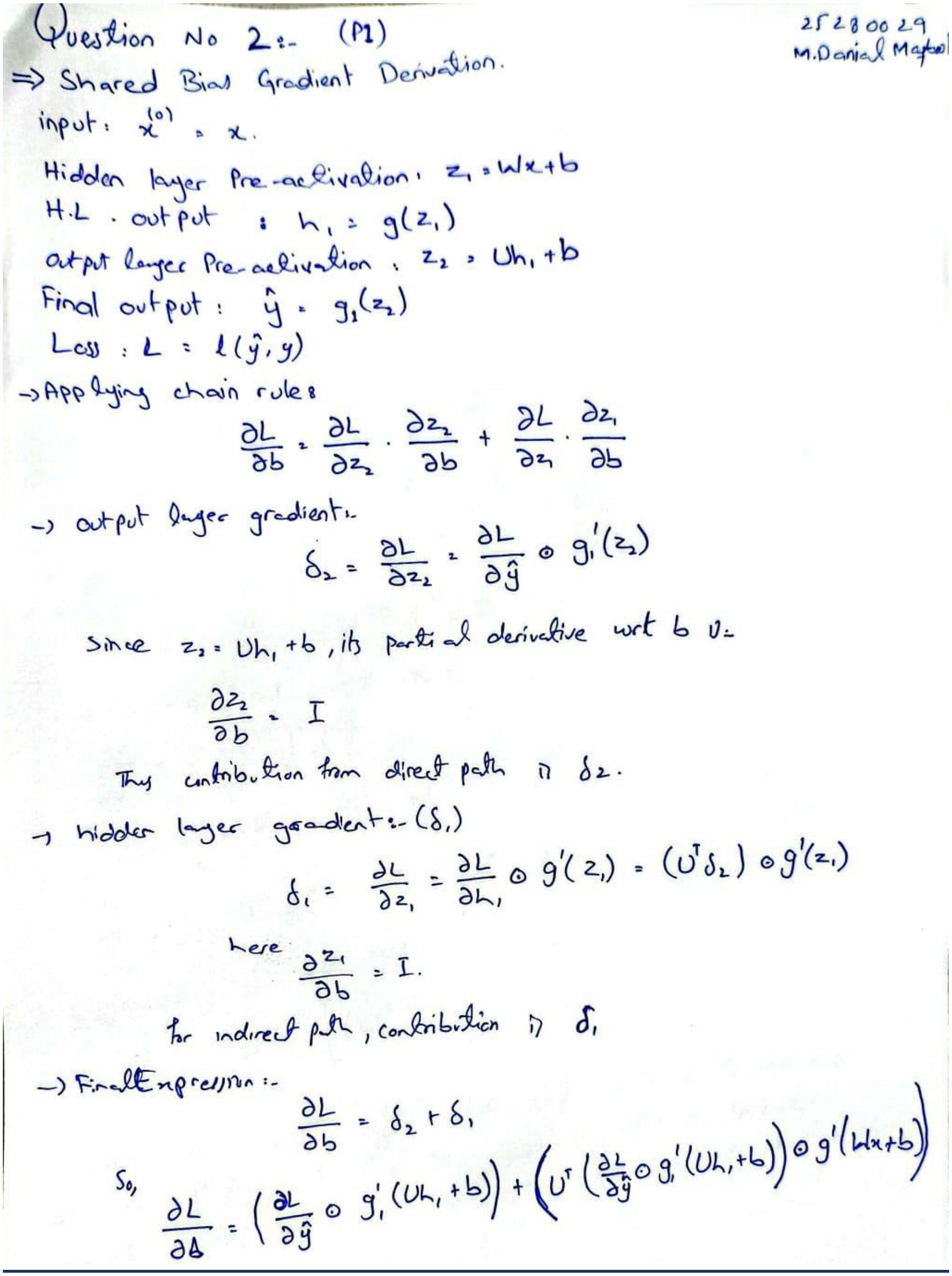
Explanation for failure: The model fell in a trap of Shortcut Learning. Rather than considering all the complicated stuff (such as how the location and the type of the room determine the price), it simply used the amenity_score. As that attribute essentially served as an ideal proxy of the label in the training data, all other good signals were overlooked by the model. However, when you apply it to data on which that correlation is likely to be non-existent or is distributed differently, the entire rule simply fails.

Mitigation Strategy:

Feature Selection / Pruning. remove the amenity_score feature in the training data.

As amenityscore is a leaky feature that does not allow the model to learn strong patterns, its removal will compel the optimization process to use other more generalizable features (e.g., neighbourhood_group and room_type). Although there may be a slight reduction in training accuracy (since the cheat code is now non-existent), an increase in test (generalization) accuracy is likely to be quite large.

Question No 2 – P1



Question No 2:- (P1)

25280029
M.Danial Maqbool

⇒ Shared Bias Gradient Derivation.

input: $x^{(0)} = x$.

Hidden layer Pre-activation: $z_1 = Wx + b$

H.L. output : $h_1 = g(z_1)$

output layer Pre-activation : $z_2 = Uh_1 + b$

Final output : $\hat{y} = g_1(z_2)$

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

→ Applying chain rule:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial b} + \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial b}$$

→ output layer gradient:-

$$\delta_2 = \frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial \hat{y}} \odot g_1'(z_2)$$

Since $z_2 = Uh_1 + b$, its partial derivative wrt b is:-

$$\frac{\partial z_2}{\partial b} = I$$

Thus contribution from direct path is δ_2.

→ hidden layer gradient:- (δ_1)

$$\delta_1 = \frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial h_1} \odot g'(z_1) = (U^T \delta_2) \odot g'(z_1)$$

here

$$\frac{\partial z_1}{\partial b} = I.$$

for indirect path, contribution is δ_1

→ Final Expression :-

$$\frac{\partial L}{\partial b} = \delta_2 + \delta_1$$

So,

$$\frac{\partial L}{\partial b} = \left(\frac{\partial L}{\partial \hat{y}} \odot g_1'(Uh_1 + b)\right) + \left(U^T \left(\frac{\partial L}{\partial \hat{y}} \odot g_1'(Uh_1 + b)\right) \odot g'(Wx + b)\right)$$

Question No 2 – P2

Yes, it is, and in effect it is likely to have a negative impact on stability and can decelerate the convergence. The reason is that in independent-bias model (b1, b2), the optimizer is able to modify b2 to bend the output distribution and b1 to bend the hidden representation separately. There is shared-bias model where b is coupled. The one-dimensional vector b is required to satisfy both optimality conditions of the input to hidden transformation Wx+b and the hidden to output transformation Uh1+b. The two layers may need varying scales or shifts in order to reduce loss in the most effective way possible. By compelling them to share the value of b, a constraint is imposed on them, which minimizes the effective degrees of freedom, and the optimization landscape becomes more complicated.

The common prejudice also imposes a strict bottleneck that presumably results in the loss landscape being smaller or harder to explore, resulting in instability or reduced convergence speed than in the independent situation where biases can find their own optimal values faster.