COMP 433 - Assignemnt 1

Adib Akkari - 40216815

Question 1

(a) Why are some columns encrypted and their impact on machine learning?

Sensitive fields such as name, full address, and date of birth are encrypted to ensure compliance with data protection regulations (e.g., GDPR, PCI DSS) and to protect customer privacy. These fields are personally identifiable information (PII), and direct exposure could lead to privacy breaches or misuse.
For machine learning, encryption has two main impacts:

- Loss of Meaning: Once encrypted, the fields become random strings and cannot contribute useful patterns to a model.
- Feature Limitations: Tasks such as age-based profiling or geographic analysis are no longer possible using encrypted raw values.

However, the bank can still support machine learning by providing anonymized or aggregated features instead of raw PII. For example:

- Instead of date of birth, provide age or age groups (e.g., 20–25, 26–30).
- Instead of full address, provide city or region (e.g., Montreal, Toronto).
- Names may be omitted entirely, as they rarely add predictive value.

This approach balances privacy protection with the need to retain useful patterns in the data pipeline.

(b) AI Task Using Traditional Programming (Expert System)

Task: Loan Approval / Credit Risk Assessment
Description: An expert system can automatically decide whether a loan application should be approved, rejected, or flagged for further review. The system uses a knowledge base containing predefined rules based on banking policies and regulations. For example:

- IF income $<$ \$30,000 AND debt $>$ \$10,000 $\Rightarrow$ reject loan
- IF credit score $>$ 700 AND debt $<$ \$5,000 $\Rightarrow$ approve loan
- IF income between \$30,000–\$50,000 AND debt between \$5,000–\$10,000 $\Rightarrow$ flag for review

The inference engine applies these rules to each applicant's data and produces a decision.
Why this does not require machine learning:

- The task relies on well-defined rules and expert knowledge, not on discovering patterns from data.
- Outcomes are deterministic: given the same input, the system will always give the same result.
- There is no need for model training since all decision logic is encoded manually.
- Transparency and explainability are critical in banking; expert systems make decisions that are easy to audit.

(c) Beneficial Supervised Regression Task

Task: Predicting the monthly spending or account balance of clients.
Description: Using features such as income, age group, account type, and past transaction history, a regression model (e.g., linear regression or random forest regression) can estimate future spending or balances. This can help the bank plan offers, manage liquidity, and provide personalized financial advice.

(d) Beneficial Supervised Classification Task

Task: Predicting customer churn.
Description: A classification model (e.g., logistic regression, decision tree, or gradient boosting) can predict whether a client is likely to leave the bank based on features such as transaction frequency, account activity, service usage, and age group. This helps the bank take proactive retention measures.

(e) Beneficial Unsupervised Learning Task

Task: Customer segmentation.
Description: An unsupervised clustering algorithm (e.g., K-means or hierarchical clustering) can group clients based on features such as spending patterns, account type, age group, and transaction frequency. This helps the bank tailor marketing strategies, product offerings, and personalized services.

(f) Typical Data Split for Supervised Tasks

A common split is:

- Training set: 70% of the data, used to train the model.
- Validation set: 15% of the data, used to tune hyperparameters and prevent overfitting.
- Test set: 15% of the data, used to evaluate the final performance of the model on unseen data.

Purpose:

- Training set: Learn the model parameters.
- Validation set: Optimize model settings and select the best model.
- Test set: Assess generalization performance and ensure the model works on new, unseen data.

(g) AI Task Suitable for Deep Learning

Task: Predicting financial transaction fraud using sequential transaction data.
Description: Deep learning models such as recurrent neural networks (RNNs) or transformers can analyze sequences of transactions to detect complex patterns that indicate fraud. These patterns may involve temporal dependencies and subtle correlations that traditional machine learning models cannot easily capture.
Why deep learning is necessary:

- Captures complex, high-dimensional patterns in sequential data.
- Learns features automatically, reducing the need for manual feature engineering.
- Performs better than traditional machine learning when the dataset is large and unstructured.

Question 2

The explanation of activation functions is adapted from [1]

(a) Why do we use activation functions in a neural network?

Activation functions are used in neural networks to introduce non-linearity. Without an activation function, each layer would simply perform a linear transformation of its inputs using weights and biases. Stacking multiple linear layers is equivalent to a single linear transformation, which limits the network's ability to model complex patterns.
In other words, a neural network without non-linear activation functions behaves like a linear regression model. It cannot learn complex functional mappings from data. By using a non-linear activation function, the network can:

- Learn complex relationships between inputs and outputs.
- Approximate almost any function given sufficient layers and neurons.
- Improve prediction accuracy for real-world tasks.

Additionally, activation functions should be differentiable to allow the network to use gradient-based optimization methods (like backpropagation) to learn the weights effectively.

(b) Compare the ReLU activation function with the Leaky ReLU activation function

- **ReLU (Rectified Linear Unit):**
  - Defined as $f(x) = \max(0, x)$.
  - Outputs zero for negative inputs and passes positive inputs unchanged.
  - Advantages: Simple, computationally efficient, helps reduce the vanishing gradient problem for positive inputs.
  - Limitation: Can suffer from the "dying ReLU" problem, where neurons may get stuck outputting zero and never recover during training.
- **Leaky ReLU:**
  - Defined as $f(x) = x$ if $x > 0$, else $f(x) = \alpha x$ with a small slope α (e.g., 0.01) for negative inputs.
  - Addresses the dying ReLU problem by allowing a small, non-zero gradient for negative inputs.

  - Helps keep neurons active and allows the network to learn more robustly.

- **Comparison:** Leaky ReLU is a modification of ReLU that mitigates the problem of inactive neurons while keeping the advantages of ReLU, such as computational efficiency and reducing vanishing gradients.

Question 3: Linear Regression with Gradient Descent

We are given the linear regression model:

$$\hat{y} = w_1 x_1 + w_2 x_2 + w_0$$

with current weights:

$$w_1 = 2, \quad w_2 = -1, \quad w_0 = 3$$

and the dataset:

Sample	x_1	x_2	y
1	1	2	4
2	2	1	3
3	0	3	1

(a) Mean Squared Error (MSE)

Predictions for each sample:

$$\hat{y}_1 = 2(1) + (-1)(2) + 3 = 3$$
$$\hat{y}_2 = 2(2) + (-1)(1) + 3 = 6$$
$$\hat{y}_3 = 2(0) + (-1)(3) + 3 = 0$$

Squared errors:

$$(\hat{y}_i - y_i)^2 : \quad (3-4)^2 = 1, \quad (6-3)^2 = 9, \quad (0-1)^2 = 1$$

Mean Squared Error:

$$\text{MSE} = \frac{1}{3}(1 + 9 + 1) = \frac{11}{3} \approx 3.667$$

(b) Gradient Descent Updates (Learning rate $\eta = 0.1$)

The gradients for a single sample are:

$$\frac{\partial MSE}{\partial w_i} = (\hat{y} - y)x_i, \quad \frac{\partial MSE}{\partial w_0} = (\hat{y} - y)$$

Step 1: Sample 1 $(x_1 = 1, x_2 = 2, y = 4)$
Current weights: $w_1 = 2, w_2 = -1, w_0 = 3$
Prediction:

$$\hat{y}_1 = w_1 x_1 + w_2 x_2 + w_0 = 2(1) + (-1)(2) + 3 = 2 - 2 + 3 = 3$$

Compute errors:

$$\hat{y}_1 - y_1 = 3 - 4 = -1$$

Compute gradients:

$$\frac{\partial MSE}{\partial w_1} = (\hat{y}_1 - y_1)x_1 = (-1)(1) = -1$$

$$\frac{\partial MSE}{\partial w_2} = (\hat{y}_1 - y_1)x_2 = (-1)(2) = -2$$

$$\frac{\partial MSE}{\partial w_0} = (\hat{y}_1 - y_1) = -1$$

Update weights: (learning rate $\eta = 0.1$)

$$w_1 \leftarrow w_1 - \eta \frac{\partial MSE}{\partial w_1} = 2 - 0.1(-1) = 2 + 0.1 = 2.1$$

$$w_2 \leftarrow w_2 - \eta \frac{\partial MSE}{\partial w_2} = -1 - 0.1(-2) = -1 + 0.2 = -0.8$$

$$w_0 \leftarrow w_0 - \eta \frac{\partial MSE}{\partial w_0} = 3 - 0.1(-1) = 3 + 0.1 = 3.1$$

Step 2: Sample 2 $(x_1 = 2, x_2 = 1, y = 3)$

$$\hat{y}_2 = 6.5, \quad \text{gradients: } \frac{\partial MSE}{\partial w_1} = 7, \quad \frac{\partial MSE}{\partial w_2} = 3.5, \quad \frac{\partial MSE}{\partial w_0} = 3.5$$

Updated weights:

$$w_1 \leftarrow 2.1 - 0.1(7) = 1.4$$
$$w_2 \leftarrow -0.8 - 0.1(3.5) = -1.15$$
$$w_0 \leftarrow 3.1 - 0.1(3.5) = 2.75$$

Step 3: Sample 3 $(x_1 = 0, x_2 = 3, y = 1)$

$$\hat{y}_3 = -0.7, \quad \text{gradients: } \frac{\partial MSE}{\partial w_1} = 0, \quad \frac{\partial MSE}{\partial w_2} = -5.1, \quad \frac{\partial MSE}{\partial w_0} = -1.7$$

Updated weights:

$$w_1 \leftarrow 1.4$$
$$w_2 \leftarrow -1.15 - 0.1(-5.1) = -0.64$$
$$w_0 \leftarrow 2.75 - 0.1(-1.7) = 2.92$$

Summary of Weight Updates:

Sample	w_1	w_2	w_0
Start	2.00	-1.00	3.00
1	2.10	-0.80	3.10
2	1.40	-1.15	2.75
3	1.40	-0.64	2.92

Question 4: Sentiment Analysis Model

Consider a sentiment analysis model that classifies sentences into three categories: Positive, Negative, or Neutral. For a dataset with five sentences, the neural network outputs the following probability distributions over the three classes:

Sample	Positive	Negative	Neutral	True Label
1	0.70	0.20	0.10	Positive
2	0.10	0.80	0.10	Negative
3	0.30	0.40	0.30	Neutral
4	0.60	0.25	0.15	Positive
5	0.20	0.30	0.50	Neutral

(a) Calculate Average Categorical Cross-Entropy Loss

The categorical cross-entropy loss formula is:

$$\text{Loss} = -\sum_{i} y_{\text{true},i} \log(y_{\text{pred},i})$$

where y_{true} is the one-hot encoded true label and y_{pred} is the predicted probability.
Step-by-step calculations:
Sample 1: True label = Positive [1, 0, 0]

$$\text{Loss}_1 = -(1 \times \log(0.70) + 0 \times \log(0.20) + 0 \times \log(0.10)) = -\log(0.70) = 0.357$$

Sample 2: True label = Negative [0, 1, 0]

$$\text{Loss}_2 = -(0 \times \log(0.10) + 1 \times \log(0.80) + 0 \times \log(0.10)) = -\log(0.80) = 0.223$$

Sample 3: True label = Neutral [0, 0, 1]

$$\text{Loss}_3 = -(0 \times \log(0.30) + 0 \times \log(0.40) + 1 \times \log(0.30)) = -\log(0.30) = 1.204$$

Sample 4: True label = Positive [1, 0, 0]

$$\text{Loss}_4 = -(1 \times \log(0.60) + 0 \times \log(0.25) + 0 \times \log(0.15)) = -\log(0.60) = 0.511$$

Sample 5: True label = Neutral [0, 0, 1]

$$\text{Loss}_5 = -(0 \times \log(0.20) + 0 \times \log(0.30) + 1 \times \log(0.50)) = -\log(0.50) = 0.693$$

Average Loss:

$$\text{Average Loss} = \frac{1}{5}(0.357 + 0.223 + 1.204 + 0.511 + 0.693) = \frac{2.988}{5} = 0.598$$

(b) Classification Accuracy

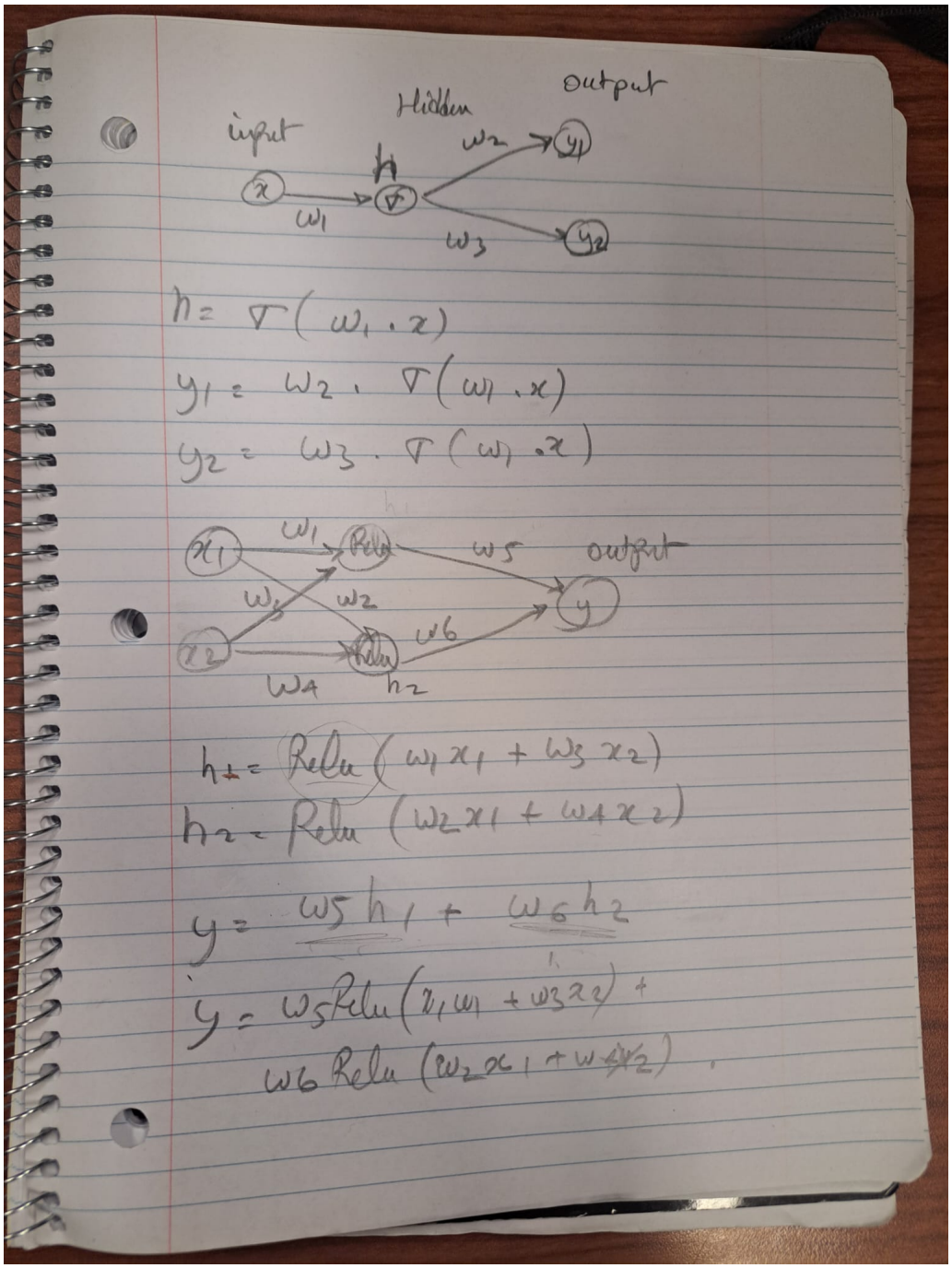
Assuming the model chooses the class with the highest probability as the output class, let's determine the predictions:

Sample	Predicted Class	True Label	Correct?
1	Positive (0.70)	Positive	Yes
2	Negative (0.80)	Negative	Yes
3	Negative (0.40)	Neutral	No
4	Positive (0.60)	Positive	Yes
5	Neutral (0.50)	Neutral	Yes

Number of correct predictions: 4 out of 5
Classification Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}} = \frac{4}{5} = 0.80 = 80\%$$

Question 5



Note: Output and Code are on Jupyter Notebook itself; no output is present in this PDF file. They are under the `titanic.ipynb` file.

References

[1] Gharat, S. (2019, April 14). What, Why and Which?? Activation Functions. Medium. Retrieved from `https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441`