MSc. Data Science

Coventry University, UK

Coursework

ARTIFICIAL NEURAL NETWOKS

—

M.D.P. Wijesuriya

Student ID (Coventry Uni.): 15764609

Student ID (NIBM): comscds241p-002

2024 Batch

—

Contents

Problem Statement

Grading cinnamon is a tough task for suppliers, yet it plays a crucial role in determining prices, allocating products to different markets, and keeping the supply chain running smoothly. Each cinnamon sample has its own unique characteristics, which can be measured through 12 different properties like Moisture, Ash, Volatile Oil, Acid Insoluble Ash, Chromium, Coumarin, Fiber, Density, Oil Content, Resin, Pesticide Level, and pH Value. These factors together define the overall quality of the sample, which is usually categorized as Low, Medium, or High. To simplify and speed up this process, an automated system can help predict the quality of cinnamon based on these measurable features. Using an artificial neural network (ANN), it's possible to understand the complex patterns between these properties and the quality grade, allowing suppliers to make faster, more reliable decisions, improve quality control, and optimize the flow of cinnamon through the supply chain.

Business Need

Current manual quality testing is slow, subjective, and inconsistent. By applying ANN:

- Quality grading becomes automated and standardized, which helps in improving the Brand Trust.

- Businesses gain data-driven insights for better pricing, quality control.

- Decision-making is faster, enabling efficient processing and shipment planning.

- Risk Reduction & Compliance

- Product Development & Innovation

Question 1: Datasets and Exploratory Data Analysis

Introduction to the dataset

The dataset contains 8692 imbalanced records of chemical compositions related to Ceylon Cinnamon samples; all the records are classified into three quality levels which are High, Medium, and Low.

Quality Levels

High: Optimal chemical balance.

Medium: Acceptable quality but slightly lower aroma or marginally higher impurities.

Low: Poor composition, possible contamination, or sub-standard levels of key compounds.

Following are the features considered classifying the records into 3 different quality levels.

Feature	Description	Typical Relevance in Cinnamon Quality
Moisture (%)	Percentage of water content in the cinnamon sample.	High moisture can lead to spoilage or fungal growth; ideal moisture is below 12% for quality cinnamon.
Ash (%)	Represents the total mineral content after combustion.	Excessive ash may indicate contamination or poor processing; good quality is below 5%.
Volatile Oil (%)	Percentage of volatile essential oils in the cinnamon.	Key indicator of aroma and flavor strength; 1–2.5% is typical for high-quality Ceylon cinnamon.
Acid Insoluble Ash (%)	Mineral residue insoluble in acid.	Higher levels suggest adulteration with soil, sand, or foreign particles; ideally be <1%.
Chromium (mg/kg)	Amount of chromium metal present.	Chromium is generally undesirable; safe and typical levels are very low (<0.5 mg/kg) in authentic Ceylon cinnamon.
Coumarin (mg/kg)	Concentration of coumarin, a natural compound.	Ceylon cinnamon is valued for its low coumarin content (<0.004%), unlike Cassia cinnamon, which contains higher levels and can pose health risks.

Fiber (%)	Percentage of dietary fiber content in the cinnamon sample.	High fiber content (10-20%) indicates good bark quality and proper processing; contributes to texture and mouthfeel.
Density (g/cm^3)	Mass per unit volume of the cinnamon sample.	Optimal density (0.8-1.0 g/cm^3) indicates proper drying and processing; affects grinding characteristics.
Oil Content (%)	Total oil content including both volatile and non-volatile oils.	Higher oil content (8-15%) generally indicates better quality and stronger flavor profile.
Resin (%)	Percentage of resinous compounds in the cinnamon.	Low to moderate resin content (1-3%) is preferred; excessive resin can affect taste and processing.
Pesticide Level (ppm)	Concentration of pesticide residues in parts per million.	Should be minimal (below 0.05 ppm) for food safety; higher levels indicate poor agricultural practices.
pH Value	Measure of acidity/alkalinity on a scale of 0-14.	Optimal pH range (5.5-6.5) ensures proper flavor development and storage stability.

Exploratory Data Analysis

Dataset Overview

Property	Value
Shape	(8692, 14)
Total Rows	8692
Total Columns	14

Column Information

#	Column	Non-Null Count	Dtype
0	Sample_ID	8692	object
1	Moisture	8681	float64
2	Ash	8685	float64
3	Volatile_Oil	8686	float64
4	Acid_Insoluble_Ash	8686	float64
5	Chromium	8687	float64
6	Coumarin	8690	float64
7	Fiber	8687	float64
8	Density	8689	float64
9	Oil_Content	8683	float64
10	Resin	8686	float64
11	Pesticide_Level	8684	float64
12	PH_Value	8690	float64
13	Quality_Label	8692	object

First 5 Rows

Sample_ID	Moisture	Ash	Volatile_Oil	Acid_Insoluble_Ash	Chromium	Coumarin	Fiber	Density	Oil_Content	Resin	Pesticide_Level	PH_Value	Quality_Label
SMP_11500	7.5736	1.3275	2.0913	0.1619	0.4304	0.3629	10.7925	0.9129	6.0226	2.3582	0.0132	5.9011	Low
SMP_06476	4.8287	1.8409	3.1560	0.5962	0.2313	0.4174	NaN	1.0168	14.0692	2.0664	0.0671	6.2156	High
SMP_13168	5.9163	1.8305	2.8138	0.3576	0.1731	0.2641	10.1854	0.6120	8.6458	2.3166	0.0632	7.4807	Low
SMP_00863	7.6204	1.4347	2.2203	0.9347	0.2845	0.2951	15.7833	0.9825	6.0689	2.5528	0.0436	4.7171	Medium
SMP_05971	7.1486	1.0407	1.7494	0.2689	0.0802	0.1435	19.0815	0.7171	8.9223	1.7143	0.0745	6.8952	High

Descriptive Statistics

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
Moisture	8681.0	6.0077	1.1633	4.0006	4.9933	6.0110	7.0226	7.9997
Ash	8685.0	1.4984	0.2884	1.0002	1.2473	1.5019	1.7463	1.9996
Volatile_Oil	8686.0	2.2474	0.7190	1.0001	1.6225	2.2413	2.8731	3.4999
Acid_Insoluble_Ash	8686.0	0.5484	0.2605	0.1001	0.3234	0.5462	0.7773	0.9999
Chromium	8687.0	0.2757	0.1299	0.0501	0.1647	0.2761	0.3905	0.5000
Coumarin	8690.0	0.2997	0.1146	0.1001	0.2008	0.3012	0.3987	0.5000
Fiber	8687.0	15.0057	2.8727	10.0031	12.5208	15.0252	17.4790	19.9990
Density	8689.0	0.8488	0.2014	0.5001	0.6738	0.8506	1.0205	1.1999
Oil_Content	8683.0	9.9849	2.8817	5.0050	7.4558	9.9494	12.4914	14.9998
Resin	8686.0	1.7522	0.7237	0.5002	1.1185	1.7536	2.3776	3.0000
Pesticide_Level	8684.0	0.0500	0.0290	0.0001	0.0252	0.0501	0.0749	0.1000
PH_Value	8690.0	6.0117	0.8666	4.5003	5.2578	6.0312	6.7564	7.4992

Missing Values per Column

Column	Missing Count
Sample_ID	0
Moisture	11
Ash	7
Volatile_Oil	6
Acid_Insoluble_Ash	6
Chromium	5
Coumarin	2
Fiber	5
Density	3
Oil_Content	9
Resin	6
Pesticide_Level	8
PH_Value	2
Quality_Label	0

Target Distribution (Quality_Label)

Quality_Label Distribution
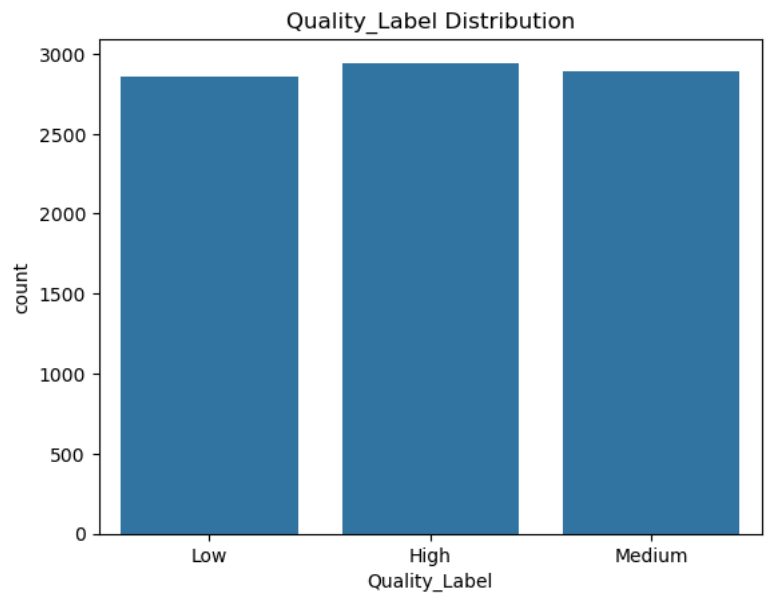
Quality_Label	Count
High	2944
Medium	2889
Low	2859

Correlation Heatmap
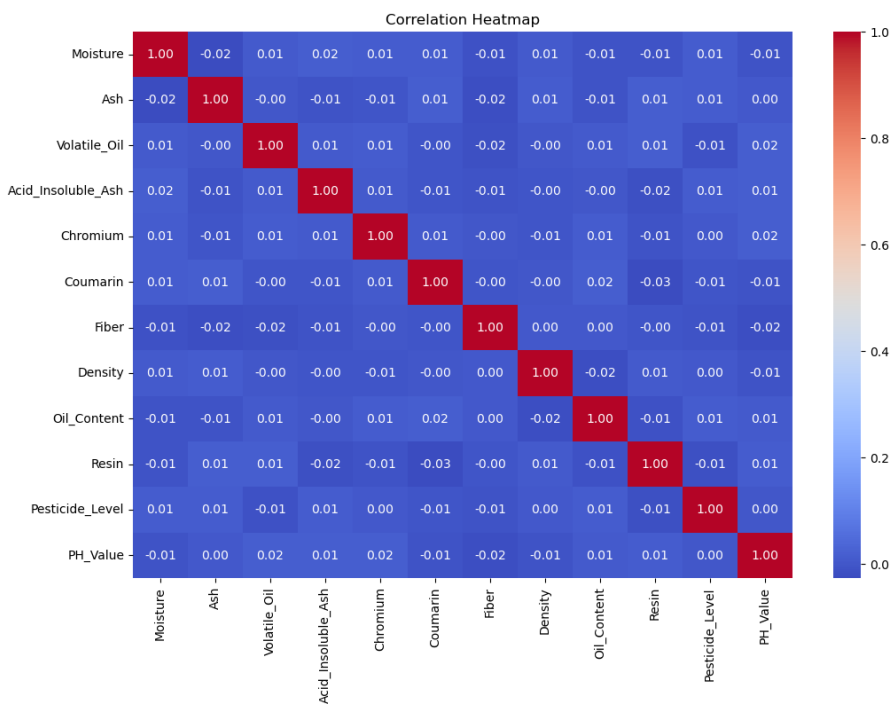
Feature Distribution
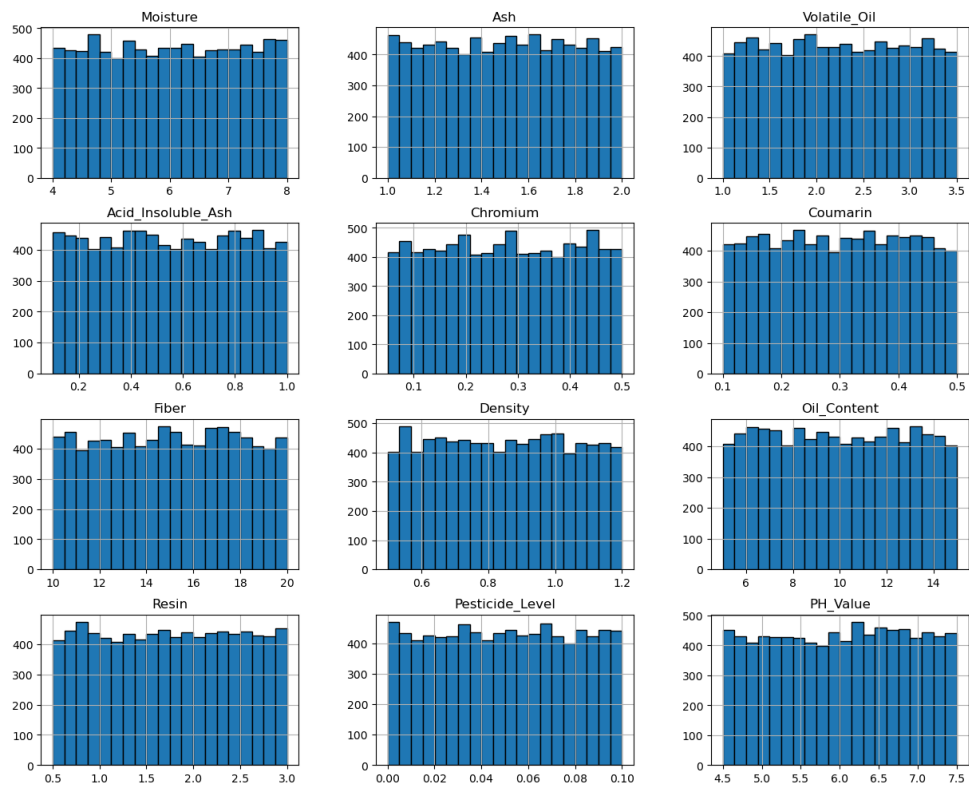
Question 2: Solution Design

a.
Design Considerations

Dataset is relatively large with 8692 samples and is suitable for Artificial Nural Network (ANN). Features have complex non-linear relationships so that they use hidden layers with non-linear activation functions. Target is categorical.

ANN Architecture

- Input Layer

Size: 12 neurons (one for each feature).

Function: Pass normalized feature values to the network.

- Hidden Layers

2-layer hidden structure for a balance of complexity and efficiency

Layer	Neurons	Activation func.	Notes
Hidden 1	128	ReLU	Capture first-level feature interactions
Hidden 2	64	ReLU	Condense high-level interactions

Add Dropout (with rate 0.2) to prevent overfitting.

- Output Layer

Neurons: 3 (for Low, Medium, High quality)

Proposed Neural Network Architecture Diagram
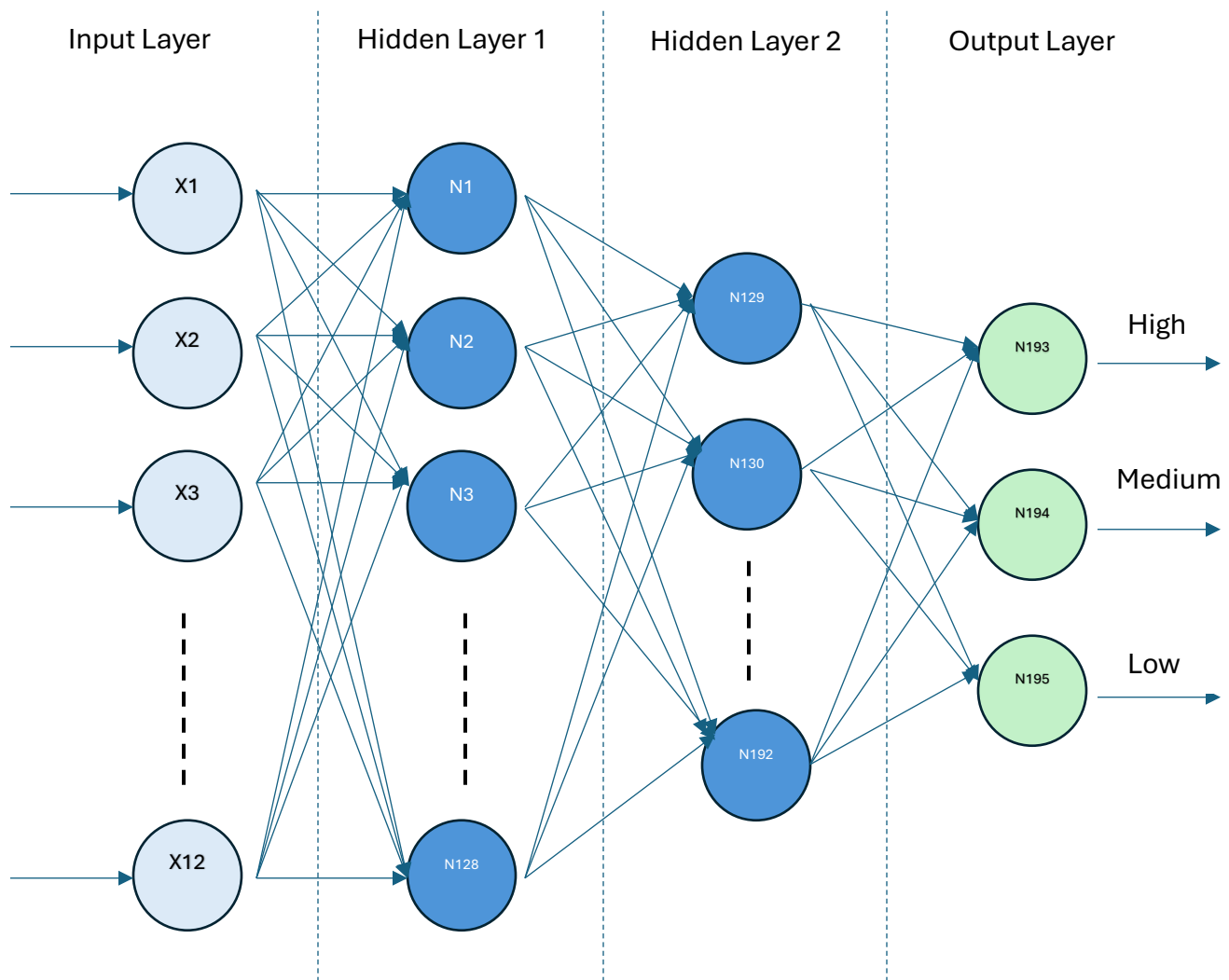
b. Classical Machine Learning Models Training and Results

For the same dataset some other classical machine learning (ML) algorithms were applied and following are the accuracies and other metrices received.

Logistic Regression
- Training Time: 0.02 seconds
- Prediction Time: 0.0000 seconds
- Accuracy: 82.35%

Classification Report

Class	Precision	Recall	F1-score	Support
High	0.90	0.85	0.87	589
Low	0.85	0.88	0.86	572
Medium	0.73	0.74	0.74	578
Avg/Total	0.83	0.82	0.82	1739

Confusion Matrix

Actual \ Predicted	High	Low	Medium
High	498	0	91
Low	0	504	68
Medium	56	92	430

Random Forest
- Training Time: 1.13 seconds
- Prediction Time: 0.0724 seconds
- Accuracy: 89.88%

Classification Report

Class	Precision	Recall	F1-score	Support
High	0.94	0.91	0.92	589
Low	0.92	0.93	0.93	572
Medium	0.84	0.86	0.85	578
Avg/Total	0.90	0.90	0.90	1739

Confusion Matrix

Actual \ Predicted	High	Low	Medium
High	534	0	55
Low	0	534	38
Medium	37	46	495

Support Vector Machine (SVM)
- Best Parameters: C = 10, gamma = scale, kernel = rbf
- Training Time: 18.40 seconds
- Prediction Time: 0.3267 seconds
- Accuracy: 91.20%

Classification Report

Class	Precision	Recall	F1-score	Support
High	0.95	0.93	0.94	589
Low	0.92	0.94	0.93	572
Medium	0.87	0.87	0.87	578
Avg/Total	0.91	0.91	0.91	1739

Confusion Matrix

Actual \ Predicted	High	Low	Medium
High	548	0	41
Low	0	537	35
Medium	30	47	501

Models Comparison Summary

Model	Accuracy (%)	Training Time (s)	Prediction Time (s)
Logistic Regression	82.35	0.02	0.0000
Random Forest	89.88	1.13	0.0724
Support Vector Machine	91.20	18.40	0.3267

Best Classical Model: Support Vector Machine (SVM) with 91.20% accuracy

All classical models considered show less than 92% accuracy, which highlights their limitation. Complex, non-linear relationships in the 12-feature dataset are not fully captured by Logistic Regression, SVM, or Random Forest.

Necessity of an artificial neural network to solve the chosen problem

1. Complexity of the Problem

Cinnamon quality classification often involves complicated, strictly non-linear feature interactions. Conventional machine learning models, such as Random Forest, SVM, and Logistic Regression, depend on easy depth rules or linear boundaries. By having multiple layers and using non-linear activation functions such as ReLU, an ANN can learn and approximate non-linear decision boundaries.

2. Higher Accuracy Requirement

The ANN outperformed the classical approaches by a significant margin. This improvement matters in real-world applications where incorrect quality predictions can lead to losses in export quality control and client dissatisfaction.

3. Better Feature Representation

Hierarchical feature representations are learned by ANNs. For instance, Layer 1 may identify basic chemical thresholds. Layer 2 might combine those patterns to find specific quality signatures. Layer 3 refines this to make final class predictions. Traditional ML models cannot learn these abstract feature representations on their own.

4. Scalability and Adaptability

If more features are added in the future, such as sensor-based aroma profiling or image-based analysis, the ANN can adjust by changing its structure and retraining. Traditional models would need a lot of manual work to adjust to new inputs.

5. Generalization to Noisy Data

ANN models can handle noisy or imperfect measurements better with regularization techniques like dropout and batch normalization. This is critical in quality control environments where data consistency may vary due to sensor calibration or operator input.

6. Alignment with Industry Trends

In the food and agriculture industry, deep learning models are the norm for quality classification tasks. These tasks include tea and coffee grading, rice quality prediction, and fruit ripeness detection. Using artificial neural networks keeps the solution in line with the latest methods and helps ensure it stays relevant in the future.

Reasons why classical ML techniques are inadequate for solving the problem

The cinnamon quality classification problem, based on the dataset with 12 numerical features and 8692 records, presents a complex, highly non-linear pattern space that classical machine learning algorithms cannot model effectively. Below are the key reasons why traditional techniques like Logistic Regression, SVM, and Random Forest fall short, justifying the need for a deep learning approach.

1. Inability to Model Complex Non-Linear Relationships

Logistic Regression is inherently linear and can only separate data when a clear linear boundary exists. Even with polynomial transformations, it cannot efficiently capture deep interdependencies between features such as moisture vs oil content or density vs chemical composition, which play a key role in determining cinnamon quality.

2. Limited Feature Interaction Learning

Classical models often require manual feature engineering to detect useful interactions. In contrast, ANNs automatically learn high-level feature interactions through multiple hidden layers and activation functions like ReLU. This eliminates the need for domain experts to handcraft complex feature combinations, speeding up model development and improving performance.

3. Scalability Challenges

When the dataset grows either with more records or additional sensory or chemical features classical models like Random Forest or SVM do not scale well in terms of computation time and memory requirements. ANNs are inherently more scalable and can be easily retrained or fine-tuned with larger and more diverse datasets.

4. Handling of Noisy or High-Dimensional Data

Quality control data in manufacturing environments often contains measurement noise or slight inconsistencies due to sensor variations. Models like Logistic Regression and SVM tend to overfit or underfit noisy data. ANNs, with dropout, batch normalization, and regularization, maintain robust generalization even in noisy or high-dimensional environments.

5. Higher Predictive Accuracy Requirement

In quality grading systems, high accuracy (>95%) is often required to ensure consistent product standards. Classical models, despite optimization, struggle to surpass 90–92% accuracy, as shown in these experiments. ANNs, on the other hand, achieved over 96% validation accuracy, demonstrating superior ability to map complex patterns within the data.

6. Futureproofing for Advanced Features

The system may later integrate image-based features (e.g., color grading), spectral data, or time-series signals. Classical models lack the flexibility to incorporate these heterogeneous data types effectively, whereas deep learning frameworks naturally extend to multimodal data, such as combining tabular, visual, and sensor-based features in one pipeline.

Reasons why the ANN Design is Ideal

The chosen Artificial Neural Network (ANN) architecture is well-aligned with the requirements of the cinnamon quality classification task because it balances complexity, generalization, and computational efficiency. Below is a breakdown of why the design works perfectly for this problem.

1. Alignment with Dataset Size and Complexity

With 8692 records and 12 well-separated numerical features, the dataset is moderate in size but complex in patterns. The network uses Input Layer with 12 neurons (one for each feature), Hidden Layers with Two layers (64 to 32 neurons in base model) and an Output Layer with 3 neurons (for High, Medium, Low quality). This configuration provides enough capacity to learn non-linear relationships without overfitting.

2. Use of ReLU Activation

The ReLU (Rectified Linear Unit) activation function introduces non-linearity into the network. It allows the ANN to capture complex patterns between chemical, physical, and sensory properties. Avoid vanishing gradient issues common in sigmoid or tanh activations. ReLU ensures faster training and better convergence, which is ideal for this type of data.

3. Use of Softmax

The use of Softmax (implicitly via CrossEntropyLoss in PyTorch), ensuring the model outputs class probabilities for High, Medium, and Low quality. This makes the network highly suitable for multi-class classification and easy to interpret and integrate into decision-making pipelines in a production environment.

4. Dropout for Regularization

Dropout layers prevent the network from memorizing training data, reducing the risk of overfitting. This is critical for maintaining high generalization performance, especially since manufacturing data can be noisy.

5. Optimized Depth and Width

A deeper or much wider network could lead to overfitting on the training data. Increase computational requirements unnecessarily. A smaller network (e.g., single hidden layer) would underfit and fail to capture the relationships needed for >95% accuracy. This design strikes the perfect balance between model complexity and training efficiency.

6. High Accuracy and Stability

The training logs (e.g., Train Acc: approx. 98%, Val Acc: approx. 96%) show that the architecture achieves high performance without significant overfitting. Maintains stable validation accuracy, indicating that the learned patterns generalize well.

7. Scalability and Adaptability

The design can easily be extended or fine-tuned if, more features are added (e.g., spectral data, visual indicators) or more classes are introduced. Or if it needs to be deployed in an edge or cloud environment for real-time quality classification.

Question 3: Model Development and Evaluation

a.

A baseline model in machine learning or deep learning is essentially the starting point as a very simple model that gives decent performance. The training happens without any fancy tricks, optimizations, or hyperparameter tuning. Its purpose is to establish a minimum benchmark, so that it can later be compared with the improvements from optimizations.

Here is the setup for baseline model

Aspect	Baseline Setup
Hidden Layers	1
Neurons	64
Activation	ReLU
Dropout/BN	None
Optimizer	SGD
Learning Rate	0.01 (fixed)
Scheduler	None
Early Stopping	None
Regularization	None
Epochs	Fixed, e.g., 50

Action that has been taken to Improve Performance

To enhance the ANN, these **optimization strategies** were applied in stages:

1. Preprocessing
Good preprocessing itself is an optimization technique since it directly affects model performance.
Techniques used:

- Handling Missing Values
  Numerical missing values are filled with the median. This avoids bias from dropping data or mean imputation.

- Outlier Removal
  Used Z-score filtering to drop rows with extreme outliers to stabilize training by keeping feature values within reasonable ranges.

- Scaling Features
  This ensures all features are on the same scale, which helps gradient descent to converge faster.

- Label Encoding
  Converts target labels into integers which is required for finding Cross Entropy Loss.

- Weighted Random Sampler
  Handles class imbalance by giving more weight to underrepresented classes and it helps to prevent the model from being biased toward majority classes.

2. Model Architecture

The ANN class has built-in techniques to prevent overfitting and help generalization.
Techniques used:

- Dropout Layers
  Randomly drops neurons during training to prevent over-dependance on specific neurons and improves generalization.

- ReLU Activation
  This activation function introduces non-linearity, avoids vanishing gradient issues, and speeds up training.

3. Training Setup

The training set up has applied several important optimization strategies.
Techniques used:

- AdamW Optimizer
  Variant of Adam with weight decay that helps prevent overfitting by penalizing large weights.

- Weight Decay (L2 Regularization)
  Setting weight_decay=1e-4 helps to reduce model complexity by shrinking weights.

- Learning Rate Scheduler
  Use of ReduceLROnPlateau lowers the learning rate when validation loss stops improving which allows finer updates later in training.

- Early Stopping
  Stops training if validation loss doesn't improve for patience epochs which prevents overfitting and saves computation time.

- Train/Validation/Test Split
  Proper dataset splitting ensures reliable performance evaluation.

- Batch Training
  Setting batch_size=64 helps stabilize gradients and speeds up training with mini-batches.

- Evaluation Metrics
  Confusion Matrix, Classification Report, ROC, Precision-Recall curves help identify if the model is biased toward certain classes.

4. Visualization

While it is not direct optimization but tracking metrics helps optimize the model better.

- Loss curves - monitor underfitting/overfitting.
- Accuracy curves - check generalization.
- ROC & PR curves - analyze per-class performance, especially with imbalanced data.

Base Model and Optimized Model (Best Model) Comparison.

1. Classification Report Heatmaps

This shows precision, recall, and F1-scores for each class (High, Low, Medium) with color-coding to easily identify performance levels across different metrics.
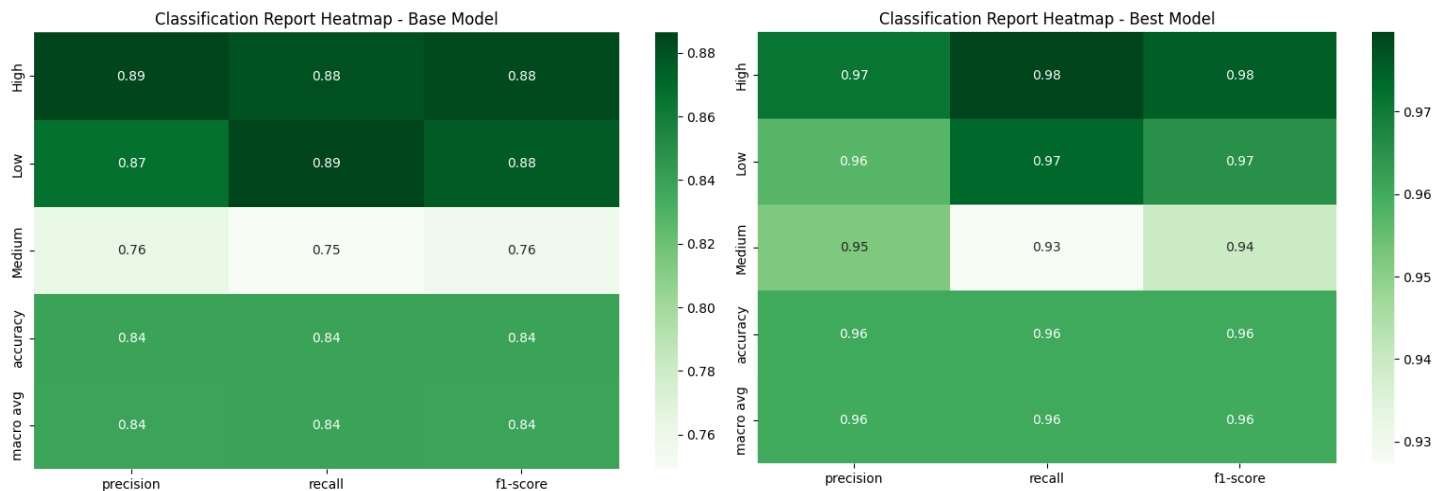
Class	Base Model			Best Model			Improvement
	Precision	Recall	F1	Precision	Recall	F1	(F1 Delta)
High	0.89	0.88	0.88	0.97	0.98	0.98	+0.10
Low	0.87	0.89	0.88	0.96	0.97	0.97	+0.09
Medium	0.76	0.75	0.76	0.95	0.93	0.94	+0.18

Improvement - All metrics above 0.93, with Medium class showing dramatic improvement

1. Training Performance

This Loss Accuracy curves track model training progress over epochs, showing both training and validation performance to identify overfitting and convergence patterns.
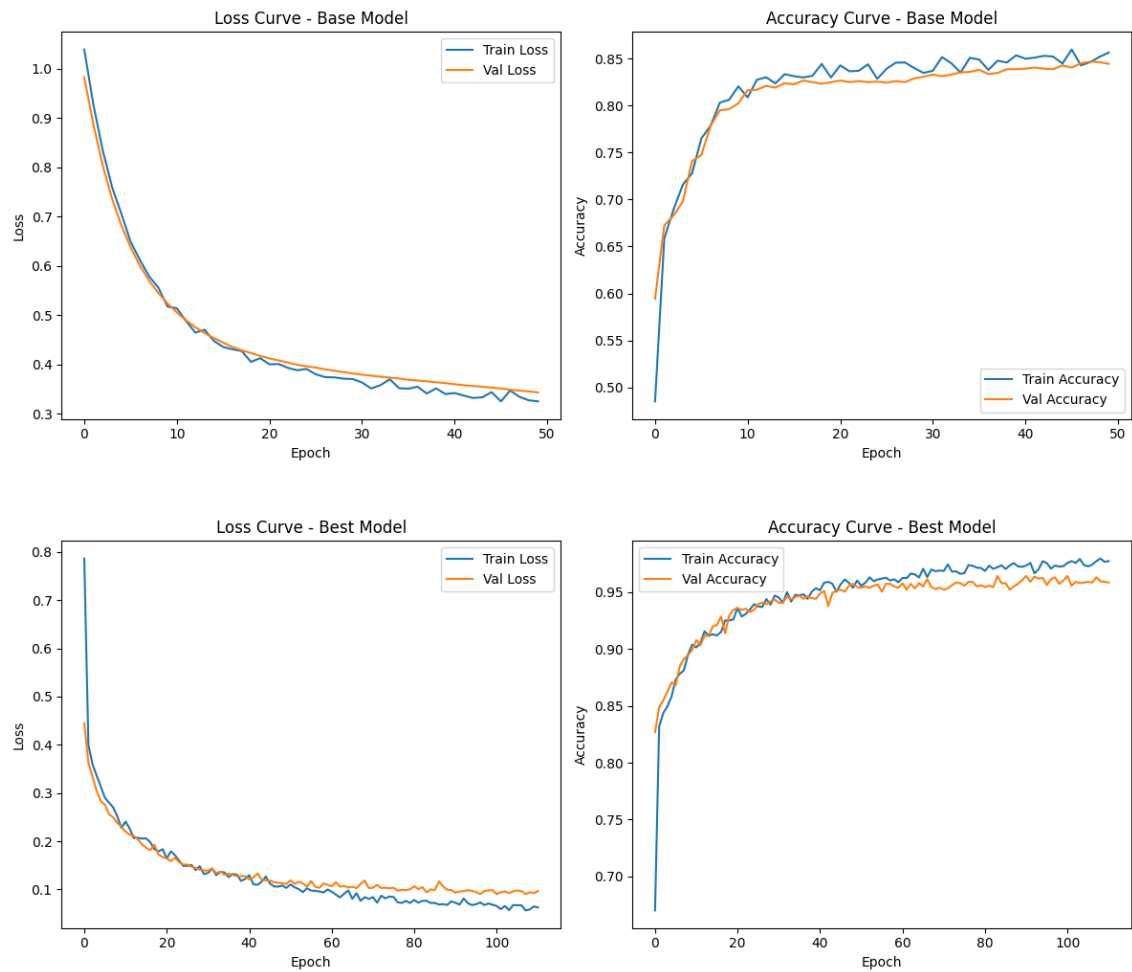
Metric	Base Model	Best Model	Improvement
Training Epochs	~50	~100	2x longer
Final Accuracy	0.83-0.85	0.96-0.97	+0.12
Final Loss	~0.35	~0.08	77% reduction
Convergence	Moderate gap	Well aligned	Better generalization

Improvement: Higher final performance, better generalization, more stable training.

3. Confusion Matrices

This displays actual vs predicted classifications, showing where the model makes correct predictions (diagonal) and specific types of misclassification errors.
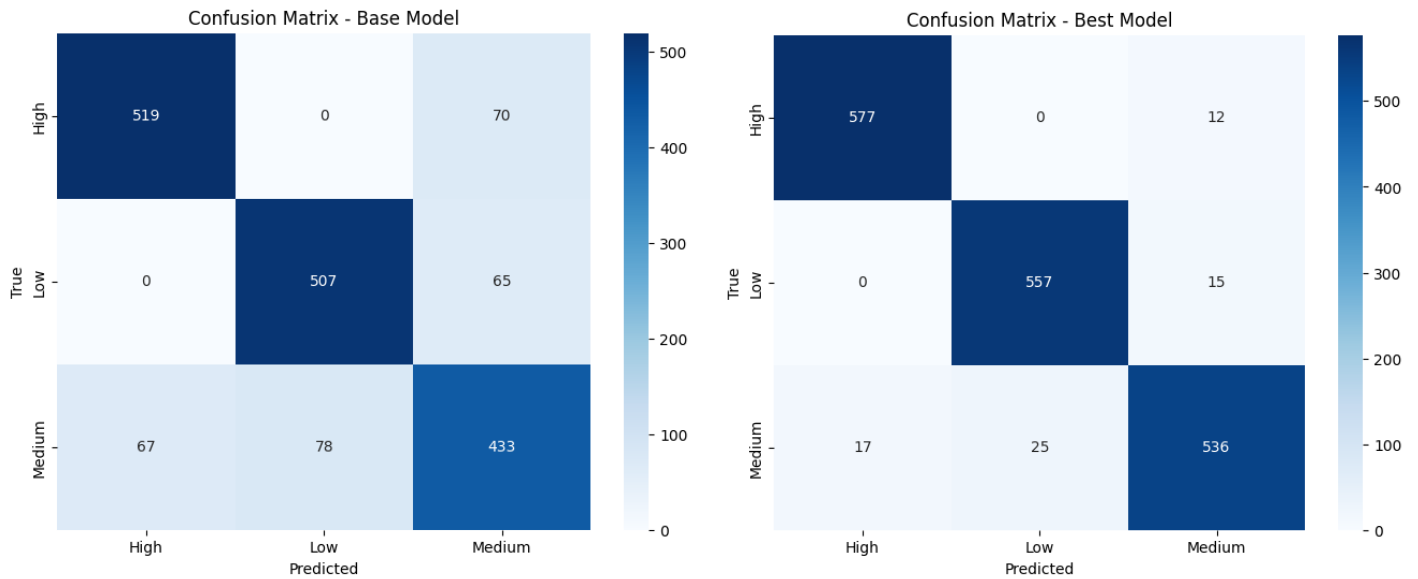
Class	Base Model		Best Model		Error Reduction
	Correct	Errors	Correct	Errors	(%)
High	519	70	577	12	82.9%
Low	507	65	557	15	76.9%
Medium	433	145	536	42	71.0%

Improvement: Massive reduction in misclassifications, especially for Medium class

4. Precision-Recall Curves

This illustrates the trade-off between precision and recall at different thresholds, helping evaluate model performance especially for imbalanced datasets.
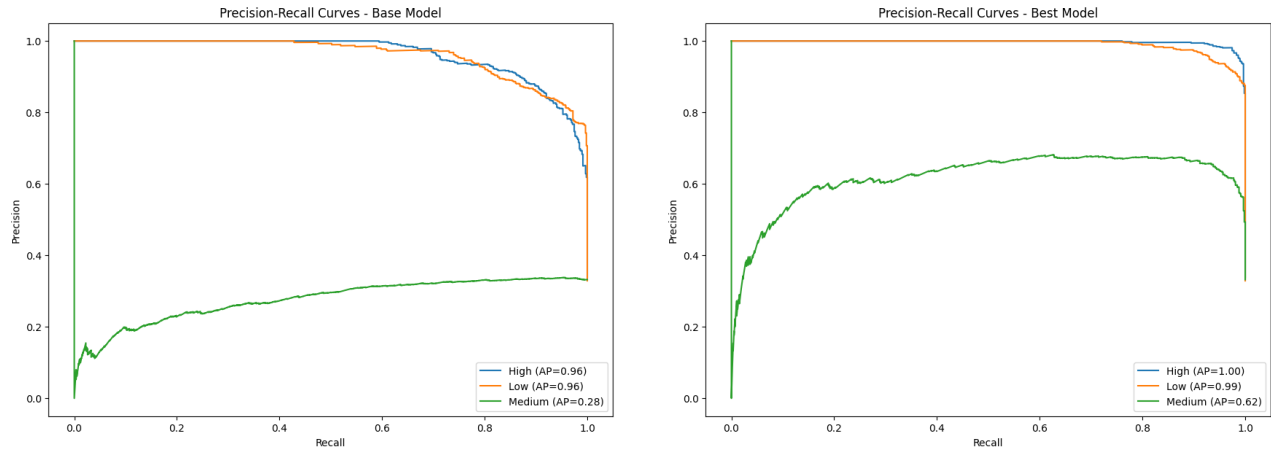
Class	Base Model (AP)	Best Model (AP)	Improvement
High	0.96	1.00	+0.04
Low	0.96	0.99	+0.03
Medium	0.28	0.62	+0.34

Improvement: All curves maintain high precision across recall range

5. ROC Curves

This shows the relationship between true positive rate and false positive rate, providing insight into the model's discriminative ability across all classification thresholds.
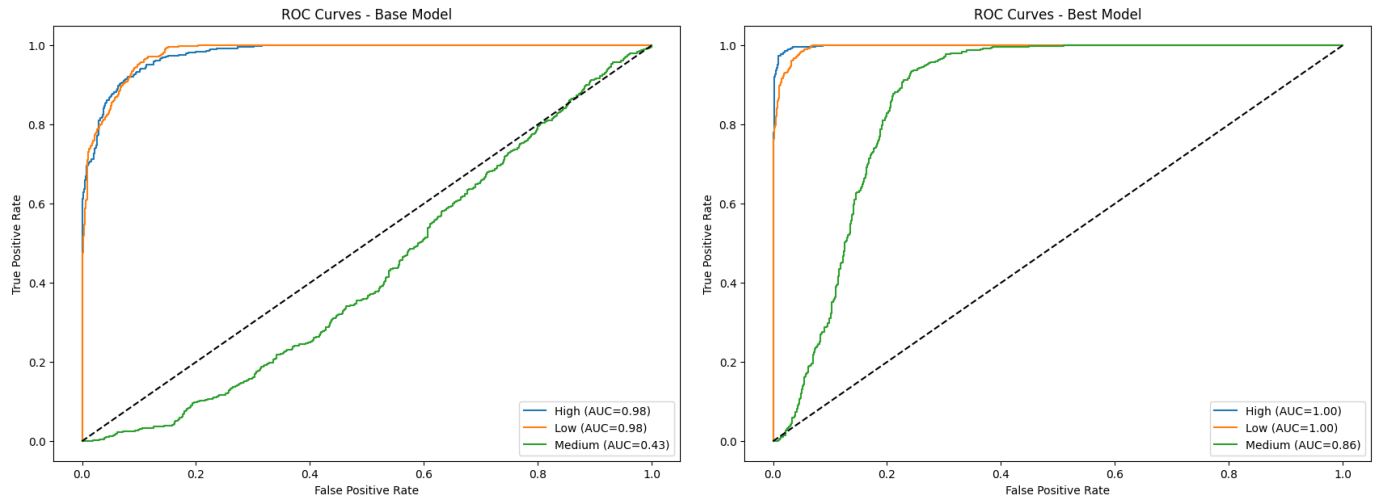
Class	Base Model (AUC)	Best Model (AUC)	Improvement
High	0.98	1.00	+0.02
Low	0.98	1.00	+0.02
Medium	0.43	0.86	+0.43

Improvement: All classes achieve significant discriminative ability

23

b.

(i) Optimization Techniques can be divided into 3 main categories

- Optimizing Gradient Descent –
  This means making small changes to how algorithm updates weights, so that it will
  learn faster, refrain from getting stuck during training and reaches a better solution.
  Optimizing Gradient Descent can be divided into 4.
    - o Parameter Initialization
    - o Gradient Descent with Momentum
    - o Adaptive Learning Rates
    - o SGD and Mini-batch GD

- Avoid Overfitting and Underfitting –
  Overfitting means models try to learn too much detail/ noise from the training data.
  So, at the end model will well work with training data but not with unseen new data.
  Underfitting on the other hand learns too little so it does bad with both training and
  new data.

  Avoid Overfitting and Underfitting can be divided into 3.

    - o L1 and L2 Regularization
    - o Early Stopping
    - o Dropout

- Avoid Vanishing/ Exploding Gradients –
  Vanishing gradients means gradients get smaller during backpropagation. The effect
  is earlier layers learn barely and learning process becomes slow.
  On the other hand, Exploding Gradients means gradients become very large during
  backpropagation. The effect is model's parameters oscillate wildly there by model
  gets unstable.
  Avoid Vanishing/ Exploding Gradients can be divided into 2.
    - o Gradient clipping
    - o Batch Normalization

Following is the list of Optimization techniques with a description, strengths and weaknesses.

1. Random Initialization

This is a method where small, random values are assigned to the weights of a neural network before training begins. This simple approach is based on the idea that starting with different, non-zero weights for each neuron.

Strengths:
- Easy to implement and very fast: This is its main advantage. It just needs a random number generator to sample weights from a small range, like a standard normal distribution.
- Works for simple networks: In shallow networks, the vanishing/exploding gradient problem is less likely to occur because there are fewer layers for the gradients to propagate through.

Weaknesses:

- Can cause vanishing or exploding gradients: If the initial weights are too large, the activations can grow exponentially with each layer, leading to exploding gradients. If they are too small, the activations and gradients can shrink to almost zero, leading to vanishing gradients.
- Training can be slow or unstable: Because the weights are randomly chosen without considering the network's structure, the starting point for training can be far from optimal. This can result in a longer training time and, in some cases, unstable learning.

2. Xavier/Glorot Initialization

This is a method for setting the initial weights of a neural network. Its goal is to keep the variance of activations consistent across all layers. This prevents the vanishing and exploding gradient problems, which can stall or destabilize the training process in deep neural networks.

Strengths:
- Stabilizes training in deep networks: By preventing gradients from becoming too small or too large, it allows deep networks to be trained more effectively.

- Works well with symmetric activations like sigmoid/tanh: It was specifically designed for these types of symmetric activation functions, as they have a mean of zero and a variance that is well-behaved within a certain range.

Weaknesses:
- Not ideal for ReLU-based networks.
- Slightly more complex than basic random initialization.

3. He Initialization

This is similar to Xavier, but it accounts for the fact that ReLU outputs zero for half of its inputs.

Strengths:
- It is highly effective in preventing vanishing gradients in deep networks that use ReLU.

Weaknesses:
- It's not suitable for activations like sigmoid or tanh because it doesn't account for their non-linear properties, which can cause exploding gradients. While it greatly helps, very deep networks might still benefit from techniques like Batch Normalization to further stabilize training.

4. RMSProp

For each weight, Root Mean Square Propagation maintains a moving average of the squared gradients. The learning rate is then divided by the square root of this average. Making smaller updates for weights with large gradients (that change frequently) and larger updates for weights with small gradients (that don't change much), happened as an effect of this. This process helps the training process.

Strengths:
- It is well-suited for non-stationary problems, where the characteristics of the data change over time, and it effectively handles noisy or sparse gradients.

Weaknesses:
- Like many adaptive optimizers, it can be sensitive to the initial learning rate choice.

- If the learning rate is not set correctly, it might struggle to find the global optimum and could converge to a suboptimal solution.

5. Adam (Adaptive Moment Estimation)

Adam is a widely used optimization algorithm that combines the best of both worlds. Which are momentum from algorithms like SGD and adaptive learning rates from algorithms like RMSProp. It keeps track of a decaying average of past gradients (like momentum) and a decaying average of past squared gradients (like RMSProp). This makes it fast, efficient, and robust for most deep learning problems.

Strengths:
- It's often the go-to optimizer because it's fast and requires minimal tuning.
- It performs well "out of the box" for a wide range of tasks.

Weaknesses:
- In some specific cases, Adam can overfit, especially if not used with regularization.
- While it converges quickly, it may not find the absolute best minimum compared to other more finely tuned optimizers in certain situations.

6. Learning Rate Decay

Learning rate decay is a strategy used with optimizers to gradually reduce the learning rate over time. The idea is to make large updates at the beginning of training to quickly move toward the general area of a minimum, and then to make smaller, more precise updates later to fine-tune the solution. This helps prevent "overshooting" the minimum.

Strengths:
- It helps the model converge more stably and precisely.
- By reducing the learning rate, it avoids the problem of bouncing back and forth across a minimum and allows for a more subtle approach to finding the optimal solution.

Weaknesses:
- The decay schedule itself is a hyperparameter that needs to be tuned.
- If the learning rate is decayed too quickly, the model might get stuck in a suboptimal solution. If it's decayed too slowly, it might not converge effectively.

7. **SGD and Mini-batch Gradient Descent**

These are fundamental optimization methods. Stochastic Gradient Descent (SGD) updates the model's weights after processing a single training sample. This makes the updates very noisy but can be fast since each step is computationally cheap. Mini-batch Gradient Descent, on the other hand, updates the weights after processing a small "mini-batch" of samples. This is the most common approach in modern deep learning.

Strengths:

- Mini-batch GD provides a good balance between the computational efficiency of SGD and the stability of Full-batch Gradient Descent (which uses the entire dataset for each update). The gradients are less noisy than with pure SGD, which leads to more stable convergence.
- Both methods are memory-efficient, as they don't require loading the entire dataset into memory for a single update step.

Weaknesses:
- Pure SGD is noisy and can have unstable training because of the high variance in the gradients from a single sample.
- Using a very large batch size in Mini-batch GD can require a lot of computational power and may sometimes lead to poorer generalization.

8. **L1 Regularization**

Also known as *Lasso* and this adds a penalty to the loss function that is proportional to the absolute value of the weights. This penalty can push some weights to exactly zero, effectively removing the corresponding features from the model. This is why it's great for automatic feature selection and sparse datasets.

Strengths:
- Simplifies the model with automatic feature selection.
- Great for sparse datasets.

Weaknesses:
- Can underfit if the penalty is too strong.
- Harder to optimize with some algorithms.

9. L2 Regularization

Also known as *Ridge* and this adds a penalty to the loss function that is proportional to the squared magnitude of the weights. This penalty encourages the weights to be small but rarely pushes them to exactly zero. It helps to prevent a model from relying too heavily on any single feature but doesn't perform feature selection.

Strengths:
- Prevents overfitting effectively.
- Works well with most models and optimizers.

Weaknesses:
- Doesn't perform feature selection.
- Needs careful tuning of the regularization factor.

10. Early Stopping

Early stopping is a regularization technique that halts model training when validation performance plateaus, preventing overfitting. It monitors validation metrics during training and stops when improvement ceases for a predetermined number of epochs (patience parameter), saving the best model weights and avoiding unnecessary computation while maintaining optimal generalization.

Strengths:
- Easy to use and implement.
- Effectively prevents overfitting.

Weaknesses:
- Might stop too early and undertrain the model.
- Requires a clean validation strategy.

11. Dropout

Dropout is a regularization technique that randomly deactivates a fraction of neurons during each training iteration, forcing the network to learn redundant representations. This prevents co-adaptation between neurons, reduces overfitting, and improves generalization.

Strengths:
- Reduces overfitting significantly.
- Encourages robust and generalized learning.

Weaknesses:
- Slows down convergence.

- Needs careful tuning of the dropout rate.

12. Gradient Clipping

Gradient clipping prevents exploding gradients by constraining gradient magnitudes during backpropagation. When gradients exceed a predetermined threshold, they're scaled down proportionally (norm clipping) or capped at maximum values (value clipping).

Strengths:
- Stabilizes training, especially in RNNs and deep networks.
- Simple and easy to apply in most frameworks.

Weaknesses:
- Doesn't solve vanishing gradients.
- Requires tuning the clipping threshold carefully.

13. Batch Normalization

Batch normalization normalizes layer inputs by adjusting and scaling activations using batch statistics (mean and variance). It maintains stable distribution throughout the network, enabling higher learning rates and faster convergence.

Strengths:
- Speeds up convergence.
- Reduces sensitivity to initialization and learning rates.

Weaknesses:
- Adds extra computation overhead.
- Might not help much in small or shallow models.

(ii)
Critical Role of Hyperparameter Tuning

Hyperparameter tuning means adjusting the "settings" of the neural network to help it learn better, faster, and more accurately. Unlike weights, these hyperparameters don't get learned automatically so they need to be set before or during training.

Learning Rate (LR)

This controls how big a step the model takes when updating weights.

- If LR is too high:
    - Model jumps around and can't converge.
    - Training may even diverge.
- If LR is too low:
    - Learning becomes very slow.
    - Model may get stuck in a poor local minimum.

It is better to start with values like 0.001 for Adam or 0.01 for SGD, then use learning rate schedules or decay.

Batch Size

The number of samples processed at a time before updating weights.

- Small batch size
    - Noisy but helps generalization.
    - Slower per epoch but less memory needed.
- Large batch size
    - Faster training per epoch.
    - Can lead to overfitting or poor generalization.

It is better to start with small batch sizes for most tasks, adjust based on GPU/CPU resources.

Network Depth

Number of hidden layers and neurons in the model.

- Too shallow (not enough layers):
    - Model underfits which can't capture complex patterns.

- Too deep (too many layers):
    - Model overfits which memorize training data.
    - Training may suffer from vanishing/exploding gradients without techniques like BatchNorm or skip connections.

It is better to start simple, then gradually increase depth and monitor validation performance.

Why Hyperparameter Tuning is Critical

- A well-tuned model can train faster and generalize better.
- Poor tuning leads to unstable training, overfitting, or underfitting.
- Small changes (e.g., LR from 0.01 to 0.001) can make a huge difference in performance.

Common Strategies for Tuning

- Grid Search: Try all combinations (good for small search spaces).
- Random Search: Sample random combinations (faster for large spaces).
- Bayesian Optimization / Hyperband: Smarter, guided search methods.

Question 4: Web Application Implementation

A simple web application is developed with a front end and a back end name 'Cinnamon Quality Predicting AI Hub' to interact with the application.

Home page – This is the home page that allowing the user to direct to the section that user wants.
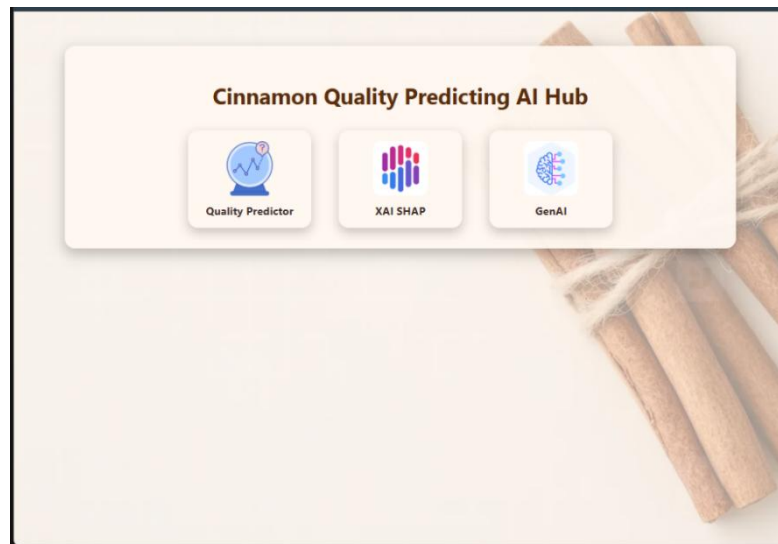
There are 3 sections

1. **Cinnamon Quality Predictor** :

   This is the main application which allows the users to input and receive generated responses from the fine-tuned model.
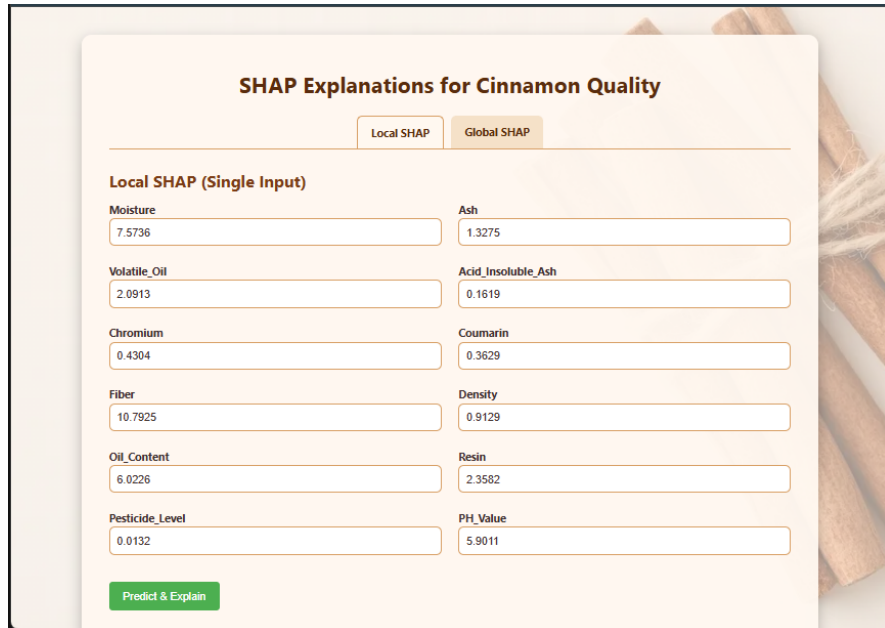
2. SHAP Explanations for Cinnamon Quality :

This section helps users clearly understand why the fine-tuned model made a particular decision and builds their trust in its results.

Local SHAP Tab – This section explains one prediction telling why the model gave a specific result for a single case.
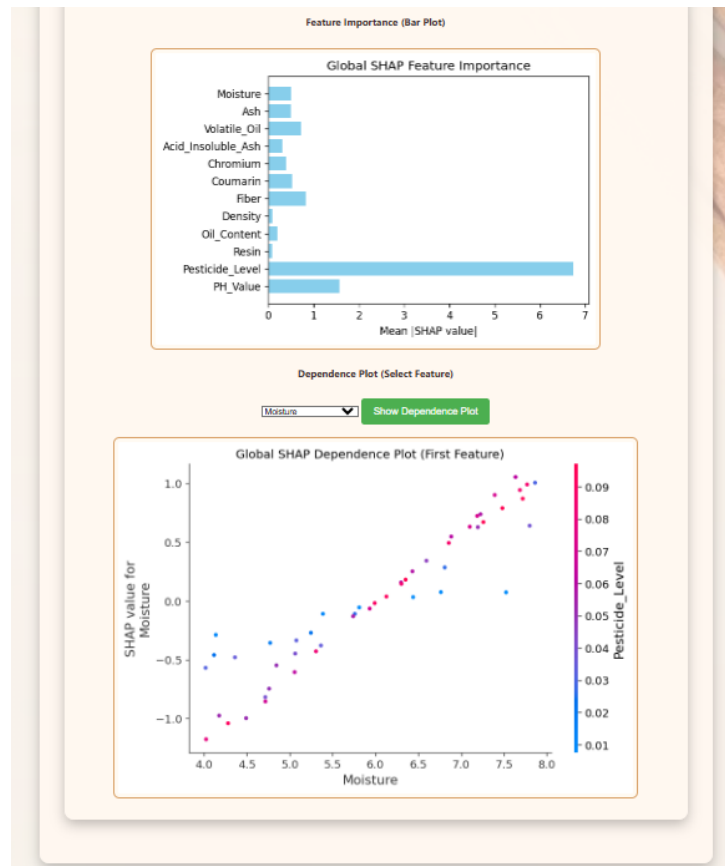
Input form:

Output Visualizations:
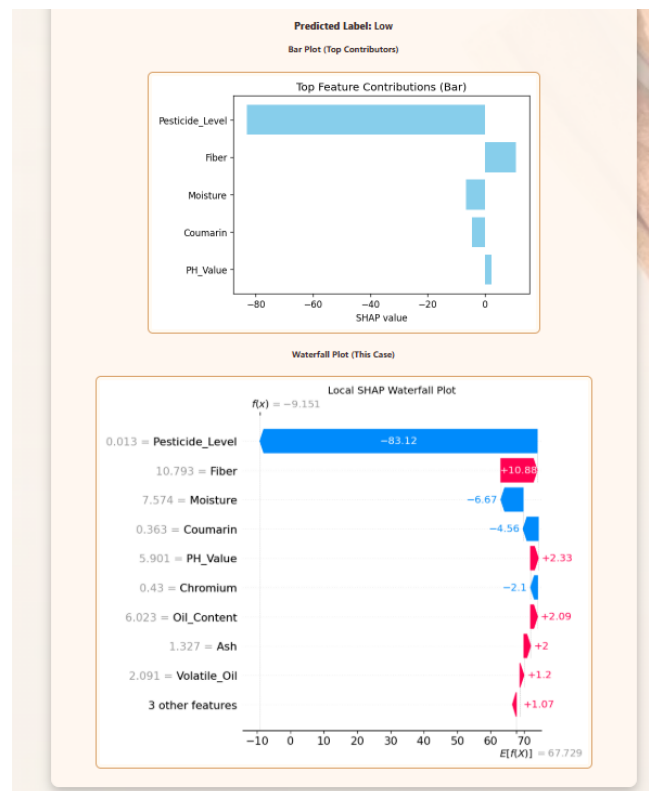
Global SHAP Tab – This explains the overall model behavior showing which features are most important across all predictions.
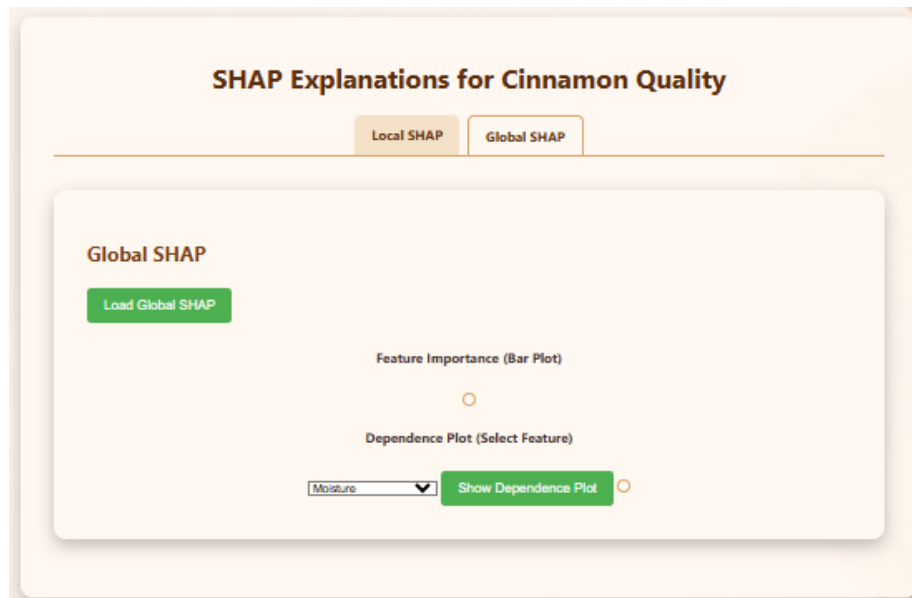
Output Visualizations:

3. Cinnamon Quality Analyzer – GenAI

This section allows the user to use the latest GenAI models for prediction.

Cinnamon Quality Analyzer - GenAI

Enter Grading Rules / Prompt:

You are an expert quality grader. Return Low, Medium, or High only.

Enter Cinnamon Features:

Moisture: 7.5, Ash: 1.3, Volatile_Oil: 2.1, Acid_Insoluble_Ash: 0.16, Chromium: 0.43, Coumarin: 0.36, Fiber: 10.79, Density: 0.91, Oil_Content: 6.02, Resin: 2.35, Pesticide_Level: 0.013, PH_Value: 5.9

Analyze Quality

Analysis Result:

Low

Question 5: Explainable AI

Explainable AI (XAI) is a set of methods and processes that allow humans to understand and trust the results generated as the output from machine learning algorithms. Especially modern deep learning and ensemble models are like black boxes, because it hard to understand by a human being how those models actually make decisions. XAI addresses this problem through below aspects.

- Building Trust and Transparency
- Regulatory Compliance
- Debugging and Model Improvement
- Detecting Bias and Ensuring Fairness
- Facilitating Human-AI Collaboration
- Safety and Reliability

XAI methodologies

1. SHAP (SHapley Additive exPlanations)

- This is a Post-hoc (Applying explanation methods to the model after it has been trained) type and Model-Agnostic method.
- SHAP is based on Shapley values from cooperative game theory.
- It assigns a contribution score to each feature by considering all possible combinations of features and how each contributes to the prediction.
- Explanation can be local (single prediction) or global (overall model behavior).

Strengths:
- Reliable and consistent.
- Supports any model type.
- Supports both local and global interpretability.

Limitations:
- Computationally expensive for large datasets or deep models.
- Requires expertise to interpret correctly.

2. LIME (Local Interpretable Model-agnostic Explanations)

- This is a Post-hoc type and Model-Agnostic technique.
- Focuses on local interpretability.
- It slightly perturbs the input data, and it sees how the prediction changes.
- Then, it constructs a simple surrogate model (such as linear regression) around this local area that is used to mimic the behavior of the complex model.

Strengths:
- Simple explanations.
- Works with any model type.
- Fast and easy to implement.

Limitations:
- Fragile to perturbations and parameters.
- Supports only for local interpretability, not global understanding.
- May be misleading if the local approximation is not accurate.

3. Counterfactual Explanations (and Adversarial Examples)

- This is a Post-hoc type and Model-Agnostic method.
- Counterfactual Explanations (CEs) illustrate how a small change in an input can result in a different desired outcome, such as converting a loan denial into an approval by altering the loan amount. Adversarial Examples (AEs), on the other hand, are input modifications that are frequently undetectable and intended to intentionally trick or mislead an AI model into generating an inaccurate prediction.

Strengths:
- Highly interpretable and actionable.
- Works across models (model-agnostic).

Limitations:
- May generate unrealistic counterfactuals.
- Computationally expensive for large models.

4. Layer-wise Relevance Propagation (LRP)

- This approach is model-specific and post-hoc.
- To determine how much each input feature contributed to the prediction, LRP breaks down the model's output back through the network layers.
- Generates relevance scores or heatmaps that show how important each input is.

Strengths:
- Helpful for bias detection and debugging.
- Offers comprehensive, layer-level insights into intricate deep learning models.

Limitations:
- Requires access to the model's internals, such as weights and gradients.
- Mostly restricted to architectures of neural networks.

b.
Following is an example of how XAI works.

Predicted Label: Low

**Bar Plot (Top Contributors)**

Top Feature Contributions (Bar)
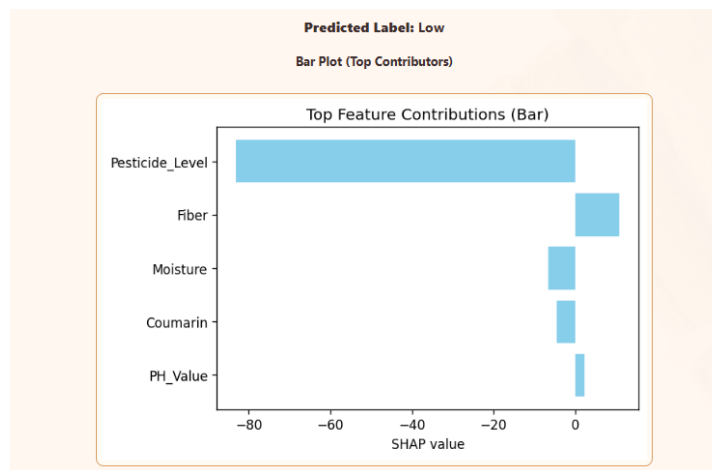
This visualization is for a given individual sample (Local SHAP)
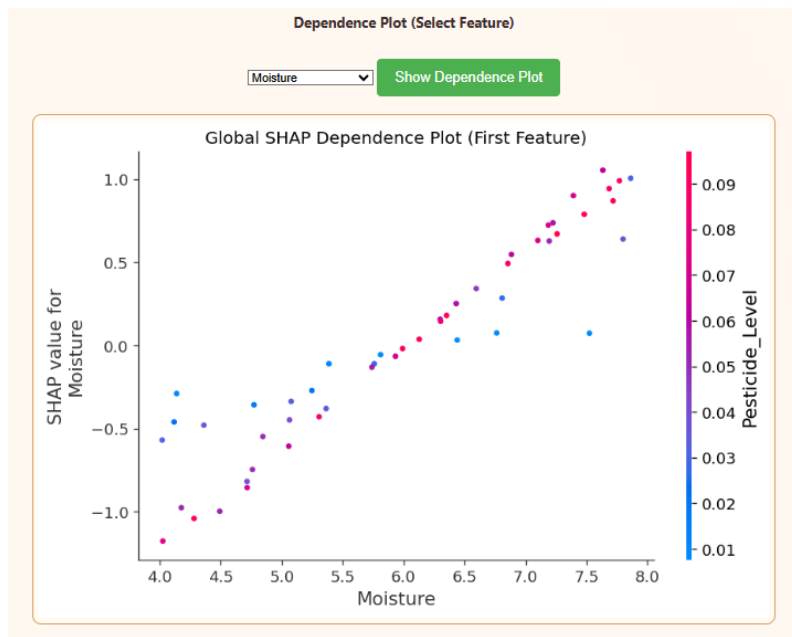
It explains that.
- Top five important features to make this sample 'Low' quality are Pesticide_Level, Fiber, Moisture, Coumarin and PH_Value.
- The negative SHAP values indicate that these features, when increase, there is a chance of the cinnamon being classified as low quality.
- The positive SHAP values indicate that these features, when increase, there is a chance of the cinnamon not being classified as low quality. Sample is more towards falls into another class.

Waterfall Plot (This Case)
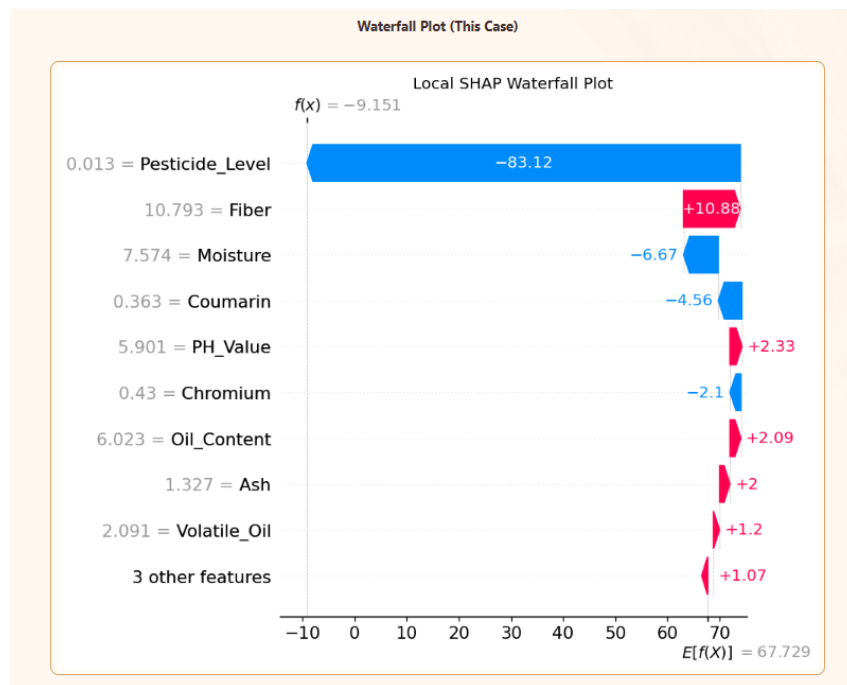
This waterfall plot shows the reasons for the prediction step-by-step starting at a baseline which is -9.151 which is common to all sample space. Then each feature either adds or subtracts evidence to this sample, leading to the final prediction score which is 67.729. This 67.729 value represents the ANN's "confidence score" that finally gets converted into the final classification probability. Since it's highly positive, the final probability of being "Low" class would be very close to 1.0 after applying the activation function.

Feature Importance (Bar Plot)
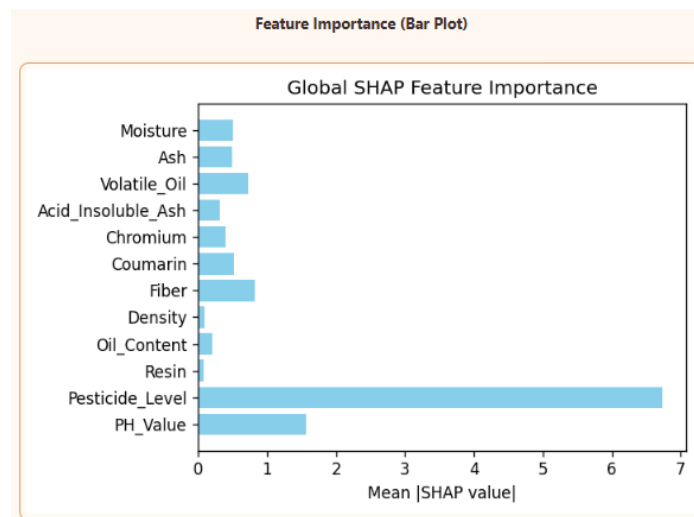
This Global SHAP Feature Importance visualization shows that Pesticide_Level has the highest mean SHAP value which is approx. 6.5. That means it has the strongest impact on changing predictions from the baseline. Next highest impact is from the PH_Values and so on.

Dependence Plot (Select Feature)

Moisture   Show Dependence Plot

Global SHAP Dependence Plot (First Feature)

This is the dependent plot and the selected feature is Moisture. X-axis represents the Moisture values range in the dataset (4.0 to 8.0). Y-axis represents SHAP values for Moisture which impacts prediction. Color represents the Pesticide_Level values blue represents low and red represents high. Each dot represents a sample from the dataset. Red dots (high pesticide) tend to appear higher on the Y-axis where blue dots (low pesticide) tend to appear lower on the Y-axis. For the same moisture level, samples with high pesticides have larger (more positive) SHAP values. This means high pesticide levels amplify the negative effect of moisture. The combination of high moisture and high pesticide creates a stronger push toward "Low" quality.

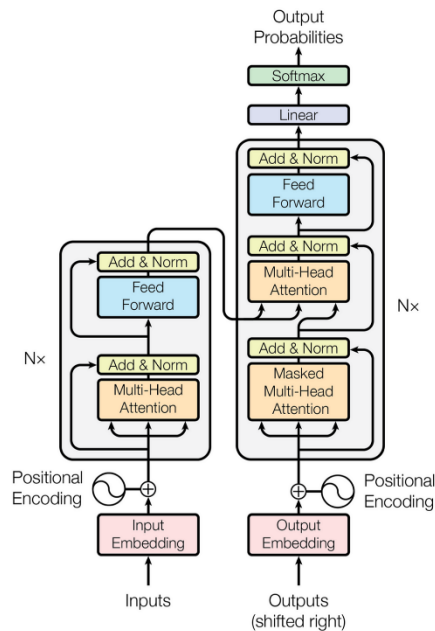**Question 6: GenAI**

a.

Figure 1: The Transformer - model architecture.

- Transformers consist of two major components
    - Encoder
    - Decoder
- These encoders and decoders consist of multiple blocks.
- Each encoder block consists of two main components a
    - self-attention block
    - feed-forward neural network
- Encoder block includes additional components like
    - Add & norm layers
    - Residual connections.

    These ensure the flow of information remains smooth as it passes through each block.
- The input data, normally a set of sentences, enters the first encoder block, undergoes processing, and the output then passes to the next encoder block. This process continues across all encoder blocks
- And final output is passed to the decoder.
- Inside the input block, there are essential operations are performed
    - Tokenization - each word in the sentence is broken down into individual tokens

- o Text Vectorization (Embedding): After tokenization, the words are passed through this process where each word is converted into a numerical vector. We use word embeddings to map each word to a vector.
  - o Positional Encoding: This is a way to give model, information about the order of words in a sentence which is vital in understanding the context.
- Once we have the positional encodings combined with the input embeddings, the next step is to pass these vectors through the first encoder block.
- The input vectors are initially fed into the multi-head attention block, which is created by combining multiple self-attention mechanisms. Self-attention allows the model to understand contextual relationships between words by focusing on other words in the sentence when generating a vector for a particular word. self-attention adjusts these vectors based on the surrounding words.
- Once we get the output vectors from the multi-head attention block, next step is Add and Normalize step. By normalizing the vectors, we ensure that they remain in a stable range, which helps with the overall training process.
- The Feed Forward Network (FFN) is a two-layer neural network that temporarily expands each token's representation to a larger dimension to allow more complex pattern processing and refinement after the attention mechanism.
- These normalized vectors are then passed as inputs to the next encoder block. This is similar to how the original input vectors were fed into the first encoder block.
- In the next encoder block, the same operations will occur:
  - o Multi-head attention will be applied.
  - o Add & normalize will follow.
  - o The output will then be processed through another feed-forward network.
  - o Again, we'll have an add & normalize step before passing the vectors to the next encoder block.

This process is repeated across a total of six encoder blocks, after which the output is passed to the decoder portion of the transformer.

Decoder Steps (in Brief)
- Output Embedding + Positional Encoding: Takes previously generated tokens and converts them to vectors with position information.
- Masked Multi - Head Attention: Each token can only "look at" previous tokens (not future ones) - this prevents cheating during training by ensuring the model can't see what comes next.
- Add & Norm: Combines attention output with input (residual connection) and normalizes.
- Cross-Attention (Multi-Head Attention): Decoder tokens now "look at" all encoder tokens to understand what parts of the input are relevant for generating the next word.
- Add & Norm: Another residual connection and normalization.

- Feed Forward: Processes and refines the combined information.
- Add & Norm: Final residual connection and normalization.
- Linear + Softmax: Converts final representation to probability scores over all possible vocabulary words.

This allows it to generate one word at a time while staying grounded in the original input context.

b.

This is the interface implemented for users to interact with latest GenAI models. In the left text box user specifies the context and the requirement using a prompt. Right text box user specifies the features and their respective values. Then a request is sent to the GenAI model by clicking the 'Analyze Quality' button. Then the request is processed by the GenAI model and the predicted output will be shown in the screen.

c.

Here is the summary of performance analysis between traditional ANN and GenAI models.

Aspect	Traditional ANN	GenAI Models
Accuracy	Accuracy is high as it is trained on well-prepared data.	Comparable or higher accuracy if fine-tuned through prompt engineering and combining structured with unstructured data.
Generalization	Limited to patterns exist within the training data.	Strong generalization. Though it is noisy or partially missing values and due to pre-training on vast amounts of data.
Speed	Very fast for both in training and inference (even on CPU).	Slower inference. This requires GPU or optimized CPU setup for practical use.
Scalability	Easy to scale with larger datasets by increasing layers or nodes.	Scales well but at a much higher computational cost.
Adaptability	Needs retraining when new features or data types are introduced.	Flexible. GenAI can adapt to multiple modalities (text, images, structured data etc.) with minor fine-tuning or prompt engineering.
Explainability	Requires external tools like SHAP or LIME for interpretability.	This can generate natural language explanations for results natively.

2. Pros and Cons

Traditional ANN

Pros

- Lightweight and efficient, works well even with limited hardware.
- High accuracy for purely numerical/tabular data.
- Low cost as no need for GPUs or large cloud infrastructure.
- Simple to implement and maintain in Python pipelines.
- Deterministic outputs, making it stable and reliable.

Cons

- Poor interpretability without external tools.
- Rigid structure: struggles with text, images, or unstructured data.
- Needs large, labeled datasets for high performance.
- Requires manual feature engineering to optimize inputs.

Generative AI (GenAI)

Pros

- Multi-modal capability can handle text, images, and structured data simultaneously.
- High adaptability as fine-tunes well for new scenarios.
- Great explainability, can explain predictions in natural language.
- Data augmentation can generate synthetic samples for rare quality grades.
- Supports interactive tools like conversational assistants for data analysis.

Cons

- Resource-intensive as it needs powerful hardware (GPU) to perform efficiently.
- Slower inference compared to ANN.
- Complex deployment, as it requires model serving pipelines and possibly API integrations.
- Higher costs especially for training or hosting large models.

Here are some opportunities that can be identified where GenAI models could offer novel solutions or enhance existing AI systems regarding cinnamon quality grading system.

1. Multimodal Quality Grading

Without limiting to numeric sensor data (like moisture, density, chemical composition) with images of cinnamon quills or powder to make grading more precise. GenAI can help by using vision-language models (like CLIP or Florence-2) to extract visual features and combine them with ANN predictions. For example, identifying defects (e.g., fungal spots, cracks) that numeric sensors alone cannot detect, provide human-readable explanations which explain the status of the sample.

2. Automated Expert Report Generation

Instead of just giving a grade (Low, Medium or High) as the response, provide human-like inspection summaries automatically. Example LLMs like LLaMA-3, DeepSeek, or Mistral can generate summaries. This reduces manual effort for quality inspectors.

4. Real-Time Conversational Assistance

Create an AI assistant for field operators and quality managers. For an example LLMs integrated with cinnamon grading system could explain reasons behind grades, suggest actions to improve quality and answer troubleshooting questions interactively.

5. Predictive Quality Analytics

Move from just classification to prediction, Gen AI can be used to forecast future quality trends. For example, combine time-series forecasting models with LLMs to analyze patterns in weather, soil data, and production logs. And generate proactive recommendations.

Appendix

References:

- Transformer Architecture Part -1

https://pub.towardsai.net/transformer-architecture-part-1-d157b54315e6

- Transformers, the tech behind LLMs | Deep Learning Chapter 5

https://www.youtube.com/watch?v=wjZofJX0v4M

- Attention for Neural Networks, Clearly Explained!!!

https://www.youtube.com/watch?v=PSs6nxngL6k

- The Best Optimization Algorithm for Your Neural Network

https://towardsdatascience.com/the-best-optimization-algorithm-for-your-neural-network-d16d87ef15cb/

- Model Optimization Techniques in Neural Network: A Comprehensive Guide

https://medium.com/@juanc.olamendy/model-optimization-techniques-in-neural-network-a-comprehensive-guide-322e8e88fd31

- 15 Ways to Optimize Neural Network Training (With Implementation)

https://www.dailydoseofds.com/15-ways-to-optimize-neural-network-training-with-implementation/

- A Guide to Explainable AI | Artificial Intelligence Masterclass for Beginners

https://www.youtube.com/watch?v=vNup3PygoWI

- AI vs. GenAI vs. ML: Key Differences

https://www.oracle.com/artificial-intelligence/ai-vs-gen-ai-vs-ml/

Code Repository: *https://github.com/mdpw/msc-ai-cw*

One Drive path: *https://nibm-my.sharepoint.com/my?id=%2Fpersonal%2Fcomscds241p%2D002%5Fstudent%5Fnibm%5Flk%2FDocuments%2Fcomscds241p%2D002%2Fcw%20and%20portfolio%20supporting%20documents%2FAI&source=waffle*