Final Exam

Machine Learning for Finance (2023-2024)

Examination Details and Instructions

Instructions:

- **Duration**: This exam is 2 hours long.
- **No Devices**: Use of computers, smartphones, or any internet-enabled devices is prohibited.
- **No Notes**: Use of personal notes is forbidden.
- **Evaluation of Research Traces**: Evidence of research will be taken into account in the grading.
- **Neural Network Architecture**: For the architectures requested in Questions 8, and 12 you may illustrate each with a diagram.

Description: The exam is graded on a 100 point scale and is divided into three independent parts. Below is the mark distribution for each question:

Problem	Question	Number of Marks
Problem A	Question 1	5
	Question 2	15
	Question 3	15
	Question 4	5
Problem B	Question 5	15
	Question 6	5
Problem C	Question 7	4
	Question 8	10
	Question 9	3
	Question 10	5
	Question 11	6
	Question 12	12

Please read the questions carefully and do your best. Good luck!

1 Problem A: Fama-French Three-Factor Model (40 marks)

This exercise will guide you through the application of the Fama-French three-factor model to Apple Inc. (AAPL) stock returns. For each time step $t \in [0, T]$, the return r_t of the AAPL stock can be expressed in the Fama French Three Factor model as:

$$r_t - r_{ft} = \alpha_{AAPL} + \beta_M(r_{Mt} - r_{ft}) + \beta_{SMB}SMB_t + \beta_{HML}HML_t + \epsilon_t \tag{1}$$

where r_{ft} is the risk-free rate at time t, r_{Mt} is the market return, SMB_t is the small minus big size premium, and HML_t is the high minus low value premium and ϵ_t an idiosyncratic risk of AAPL at time t that follow a $\mathcal{N}(0, \sigma^2)$ independent of time t.

Question 1 — 5 Marks

Give the name of the machine learning model described in Equation 1.

Solution: The model described in Equation 1 is a Linear Regression.

Question 2 — 15 Marks

Reformulate the Fama-French Model in matrix notation for the period from $t = 0$ to T ($T+1$ steps) as indicated in the following Equation 2.

$$\hat{R}_T = X_T\hat{\beta} + \tilde{\epsilon}_T \tag{2}$$

Clarify how these matrices interact within the model, ensuring the dimensions align correctly for matrix operations from the beginning to the end of the specified time interval.

Please include the following details in your response:

- Describe the matrix X_T and explain what each column represents.
- Define the shapes of $\hat{R}_T$, $\hat{\beta}$, and $\tilde{\epsilon}_T$ in the context of this model.
- Explain how these matrices relate to the original variables in the Fama-French model.

Solution: We can rewrite the Equation (1) as:

$$\begin{bmatrix} r_0 - r_{f0} \\ \vdots \\ r_T - r_{fT} \end{bmatrix} = \begin{bmatrix} r_{M0} - r_{f0} & SMB_0 & HML_0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ r_{MT} - r_{fT} & SMB_T & HML_T & 1 \end{bmatrix} \begin{bmatrix} \beta_M \\ \beta_{SMB} \\ \beta_{HML} \\ \alpha_{AAPL} \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \vdots \\ \epsilon_T \end{bmatrix} \tag{3}$$

We deduce the following matrices and their shapes:

The vector $\hat{R}_T = \begin{bmatrix} r_0 - r_{f0} \\ \vdots \\ r_T - r_{fT} \end{bmatrix}$ represents the adjusted returns and has the shape $(T+1, 1)$. The beta coefficients vector $\hat{\beta} = \begin{bmatrix} \beta_M \\ \beta_{SMB} \\ \beta_{HML} \\ \alpha_{AAPL} \end{bmatrix}$ is organized as a column and has the shape $(4, 1)$. The matrix $X_T = \begin{bmatrix} r_{M0} - r_{f0} & SMB_0 & HML_0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ r_{MT} - r_{fT} & SMB_T & HML_T & 1 \end{bmatrix}$ contains the factor values and a constant, with

the shape $(T+1, 4)$. Finally, the vector of residuals $\tilde{\epsilon}_T = \begin{bmatrix} \epsilon_0 \\ \vdots \\ \epsilon_T \end{bmatrix}$ has the shape $(T+1, 1)$.

Question 3 **15 Marks**

Prove the equivalence between maximum likelihood estimation (MLE) and ordinary least squares (OLS) for estimating the parameter $\hat{\beta}$ in a linear regression model.

Consider the OLS problem defined as:

$$\min_{\hat{\beta}} \| R_T - X_T\hat{\beta}\|^2 \tag{4}$$

Follow these steps to establish the equivalence:

- Assume that the residuals $\epsilon_T = R_T - X_T\hat{\beta}$ are normally distributed with mean zero and variance $\sigma^2 I$, where I is the identity matrix. Write the likelihood function for this setup.
- Derive the log-likelihood function from the likelihood function. Simplify the expression.
- Perform the differentiation of the log-likelihood function with respect to β and set the derivative to zero to find the maximum likelihood estimate of β.
- Conclude that this derivation leads to the same solution as the minimization problem in Equation 4.

Solution: *Some students didn't provide the optimal beta or missed the optimal beta. We can accept the answers if the equivalence is well proven.*

To prove the equivalence between maximum likelihood estimation (MLE) and ordinary least squares (OLS) for estimating the parameter β in a linear regression model, follow these steps:

1. **Assume normally distributed residuals**: Given the assumptions that the residuals $\epsilon_T = R_T - X_T\hat{\beta}_T$ are normally distributed with mean zero and variance $\sigma^2 I$, the likelihood function of the observed data R_T, given the parameters β and σ^2, can be written as:

$$L(\beta, \sigma^2; R_T) = \prod_{i=0}^{T} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(R_T^{(i)} - X_T^{(i)}\beta)^2}{2\sigma^2}\right)$$

   Here, $R_T^{(i)}$ and $X_T^{(i)}$ are the i-th components of R_T and rows of X_T respectively.

2. **Derive the log-likelihood function**: The log-likelihood function is the logarithm of the likelihood function:

$$\ell(\beta, \sigma^2; R_T) = \sum_{i=0}^{T} \left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(R_T^{(i)} - X_T^{(i)}\beta)^2}{2\sigma^2}\right)$$

Simplifying this, we get:

$$\ell(\beta, \sigma^2; R_T) = -\frac{T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=0}^{T}(R_T^{(i)} - X_T^{(i)}\beta)^2$$

3. **Differentiation and setting the derivative to zero**: To find the maximum likelihood estimate of β, differentiate the log-likelihood function with respect to β and set this derivative to zero:

$$\frac{\partial \ell}{\partial \beta} = \frac{1}{\sigma^2}\sum_{i=0}^{T} X_T^{(i)}(R_T^{(i)} - X_T^{(i)}\beta) = 0$$

This leads to:

$$\sum_{i=0}^{T} X_T^{(i)} R_T^{(i)} = \sum_{i=0}^{T} X_T^{(i)} X_T^{(i)}\beta$$

Solving for β, we find:

$$\hat{\beta}_{MLE} = \left(X_T^T X_T\right)^{-1} X_T^T R_T$$

4. **Equivalence with OLS**: The minimization problem in the ordinary least squares (OLS) formulation is:

$$\min_{\beta} \|R_T - X_T\beta\|^2$$

Differentiating the above expression with respect to β and setting it to zero also leads to:

$$X_T^T X_T \beta = X_T^T R_T$$

Solving for β in this context gives the same result:

$$\hat{\beta}_{OLS} = (X_T^T X_T)^{-1} X_T^T R_T$$

Therefore, the maximum likelihood estimate (MLE) under the assumption of normally distributed errors is equivalent to the ordinary least squares (OLS) estimate. Both methods yield the same estimator $\hat{\beta}$, confirming their equivalence in the context of linear regression with normally distributed errors.

Question 4 — 5 Marks

In our model the expected prediction error at a point x can be decomposed into bias, variance, and noise according to the following equation:

$$\text{MSE} = \text{Bias}^2(\hat{r}) + \text{Var}(\hat{r}) + \sigma^2 \tag{5}$$

where $\hat{r}$ is the prediction made by the model at x, $\text{Bias}(\hat{r})$ is the difference between the expected prediction and the true value, $\text{Var}(\hat{r})$ is the variance of the prediction at x, and σ^2 represents irreducible error or noise.

- Explain the concepts of bias and variance in this setting and discuss their trade-off.

- Suggest strategies to minimize bias and strategies to reduce variance in model predictions.

Solution:

- **Bias:** Refers to the error due to overly simplistic assumptions in the model. High bias can cause the model to miss relevant relations between features and target outputs (underfitting).
- **Variance:** Indicates how much the model's predictions would change if it were estimated from a different training data set. High variance can cause the model to model the random noise in the training data, rather than the intended outputs (overfitting).

Trade-off:

- Increasing model complexity decreases bias but increases variance. Simplifying the model increases bias and decreases variance. The goal is to achieve a balance that minimizes overall error.

Strategies to Minimize Bias:

- Use more complex models or add more features to capture more data trends.
- Decrease regularization strength to allow the model to fit more closely to the data.

Strategies to Reduce Variance:

- Increase the size of the training data set to improve model generalization.
- Use regularization techniques (e.g., Ridge, Lasso) to simplify the model.
- Implement ensemble methods such as bagging and random forests to average multiple predictions.

2 Problem B: Decision Tree (20 marks)

Imagine you are working with a dataset intended for a binary classification task. The dataset contains the following features: Age, Salary, and Education Level. Your target variable is *Purchased*, which is binary (1 for yes, 0 for no).

The training data snapshot is as follows:

Age	Salary	Education Level	Purchased
25	\$50,000	High School	0
40	\$65,000	Bachelor's	1
30	\$80,000	Bachelor's	0
35	\$95,000	Master's	1
45	\$110,000	Doctorate	1

Question 5 **15 Marks**

Divide the dataset into two groups based on the Salary being either less than or equal to \$90,000, and more than \$90,000. Calculate the Gini index for each group and the overall Gini index for the split.

Hint: To calculate the Gini index for a group:

$$\text{Gini} = 1 - (p_1^2 + p_0^2)$$

where p_1 and p_0 are the proportions of class 1 and class 0 in the group, respectively.

To compute the overall Gini index for the split:

$$\text{Gini}_{\text{split}} = \left(\frac{n_A}{n}\right) \text{Gini}_A + \left(\frac{n_B}{n}\right) \text{Gini}_B$$

where n_A and n_B are the number of samples in each group, and n is the total number of samples.

Solution:

Step 1: Divide the dataset based on the split point:

- Group 1: Salary $\leq \$90,000$
- Group 2: Salary $> \$90,000$

Group 1 Data:

- Age 25, Salary \$50,000, Purchased 0
- Age 40, Salary \$65,000, Purchased 1
- Age 30, Salary \$80,000, Purchased 0

Group 2 Data:

- Age 35, Salary \$95,000, Purchased 1
- Age 45, Salary \$110,000, Purchased 1

Step 2: Calculate proportions and Gini index for each group:

Group 1:

- Purchased = 0: 2 out of 3
- Purchased = 1: 1 out of 3

$$\text{Gini}_{\text{Group 1}} = 1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right) = 1 - \left(\frac{4}{9} + \frac{1}{9}\right) = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

Group 2:

- Purchased = 1: 2 out of 2 (homogeneous group)

$$\text{Gini}_{\text{Group 2}} = 1 - \left(1^2 + 0^2\right) = 0$$

Step 3: Calculate the overall Gini index for the split:

- Total samples = 5, Group 1 = 3, Group 2 = 2

$$\text{Gini}_{\text{overall}} = \left(\frac{3}{5}\right) \times 0.444 + \left(\frac{2}{5}\right) \times 0 = 0.2664$$

Question 6 **5 Marks**

Consider how different educational levels might affect the likelihood of purchasing, and propose a split. Discuss why this split might make sense from an information gain perspective, focusing on how it could help segregate the data into more homogeneous subsets without using mathematical terms.

Solution: Proposal for Split:

- Group 1: High School and Bachelor's
- Group 2: Master's and Doctorate

Justification:

1. **Purchasing Pattern in the Data:**
   - From the snapshot, we see that no one with less than a Master's degree has purchased except for one Bachelor's degree holder. This suggests that higher educational qualifications might correlate with a higher probability of purchasing.
2. **Homogeneity of Groups:**
   - Splitting this way creates a group (Group 2) that is entirely purchasers (homogeneous with respect to the target variable). This increases the purity of this node, which is desirable in decision tree models as it helps in reducing the entropy or increasing the information gain.
3. **Information Gain:**
   - By splitting into these groups, the subsets become more homogeneous (especially Group 2), likely leading to higher information gain. Since the goal in decision tree models is to maximize information gain at each split, this criterion should theoretically yield a beneficial split.

3 Problem C: Sentiment Analysis (40 marks)

Consider a dataset for sentiment analysis related to stock price prediction. The dataset comprises N sentences of **varying lengths**, each annotated with one of three sentiments: *Good*, *Neutral* or *Bad*. These sentences are extracted from sources such as news articles, social media posts, or financial reports.

The dataset used for this exercise is as follows:

Index	Sentence	Sentiment
1	Company profits soar.	*Good*
2	Market stability remains uncertain.	*Neutral*
⋮	⋮	⋮
N	Legal issues impact stock prices.	*Bad*

3.1 FeedForward Neural Network

To effectively utilize a Deep Neural Network with 3 dense layers, including the last one, for our classification task, we need to preprocess the data appropriately.

Question 7 **4 Marks**

Describe four preprocessing steps necessary to transform our sentences into fixed-length vectors (of length V).

Solution: Good Answers (4 Marks): To transform sentences into fixed-length vectors of length V using one-hot encoding, you can follow these four preprocessing steps:

1. **Tokenization:** Tokenize each sentence into individual words. Tokenization involves splitting the sentences into a list of words (or tokens).
2. **Counting Word Frequencies:** Count the frequency of each word in the corpus. This helps to identify the most common words which will be included in the fixed vocabulary.
3. **Building the Vocabulary:** Build a vocabulary of size V by selecting the top V most frequent words. Any words not in the top V are considered out of vocabulary (OOV).
4. **Create Presence Vector:** For each sentence, create a vector of zeros of dimension V. For each word in the vocabulary, if the word is present in the sentence, change the corresponding 0 to 1.

Some students provide this answer, which is good for working with RNNs. We accept it (4 marks): To prepare the data for a deep learning model, especially for natural language processing tasks like sentiment analysis, it is essential to convert the textual data into a

fixed-length numerical format. Here are four critical preprocessing steps necessary to transform sentences into fixed-length vectors:

1. **Tokenization:** Convert each sentence into a sequence of words or tokens. This process involves splitting the text into words, punctuation, or other meaningful elements called tokens.
2. **Vectorization:** Transform the tokens into numerical vectors that can be processed by the neural network.
3. **Padding and Truncation:** To ensure that all input vectors have the same length (denoted as V), apply padding or truncation. Padding involves adding a special 'pad' token to shorter sequences to reach the desired length, while truncation removes elements from sequences that exceed the length V.

These preprocessing steps are fundamental for converting raw text data into a structured form that a feedforward neural network can process, leading to effective learning and prediction in sentiment analysis tasks.

Question 8 **10 Marks**

For this task, your goal is to design a neural network that classifies sentiments into three categories: *Good*, *Neutral* and *Bad*. Provide a detailed description of your design by addressing the following points:

Neural Network Architecture

- **Input Layer:**
  - Specify the expected dimension V of input vectors, explaining how V is determined based on preprocessing steps.
- **Dense Layers:**
  - Detail the number of units and the type of activation functions used in these layers.
  - Explain why you chose these specific configurations.
- **Output Layer:**
  - State the number of units and describe the activation function used. Explain how this configuration aligns with the task of classifying sentiments into three categories.

Activation Functions

- Clarify the activation functions used in different parts of the network.
- Discuss the rationale behind choosing each activation function, particularly focusing on the output layer.

Data Tensor Shape

- Describe the shape of the data tensor that feeds into the network.
- Explain the significance of each dimension in the tensor.

Solution: We propose the following architecture tailored to classify sentiments into three categories: "Good," "Neutral," and "Bad." The architecture, activation functions, and data tensor shape are described as follows:

Neural Network Architecture

1. **Input Layer:** The input layer should accommodate vectors of fixed length V, which is determined by the preprocessing step (Vocabulary Size).
2. **Dense Layers: (Proposed Architecture)**
   - **First Dense Layer:** This layer can have 128 units with a ReLU (Rectified Linear Unit) activation function, which is standard for hidden layers in neural networks due to its efficiency and effectiveness in non-linear transformations.
   - **Second Dense Layer:** A subsequent layer of 64 units, also with ReLU activation, to continue the pattern recognition process started in the first dense layer.
3. **Output Layer:** The last layer of the neural network, which is crucial for classification, should have 3 units corresponding to the three sentiment categories. The activation function for this layer should be the softmax function, which is ideal for multi-class classification tasks because it outputs a probability distribution across the classes.

Activation Functions and Output Dimension

- **Activation Functions:** The hidden layers use the ReLU activation function for introducing non-linearity, while the output layer uses the softmax function to provide a probabilistic interpretation of class memberships.
- **Output Dimension:** The final output layer has 3 units, corresponding to the softmax activation which outputs the probability distribution over the three classes.

Shape of the Data Tensor

- **Input Data Tensor:** The shape of the input data tensor to the network would typically be $[N, V]$, where N is the number of sentences in the batch, and V is the length of each vectorized sentence.

Question 9 **3 Marks**

Specify the suitable loss function necessary for our neural network model.

Solution: Loss Function

- **Categorical Cross-Entropy:** Since our task involves classifying input sentences into one of three categories ("Good," "Neutral," or "Bad"), the categorical cross-entropy loss function is most suitable. This loss function is ideal for multi-class classification problems where the output layer includes a softmax activation function. It measures the discrepancy between the predicted probabilities and the actual class in the form of a one-hot encoded vector. Mathematically, it is defined as:

$$L = -\sum_{i=1}^{C} y_i \log(p_i)$$

where y_i is the binary indicator (0 or 1) if class label i is the correct classification for the observation, and p_i is the predicted probability of the observation being of class i.

Question 10 **5 Marks**

After the first training, we obtain this graph:

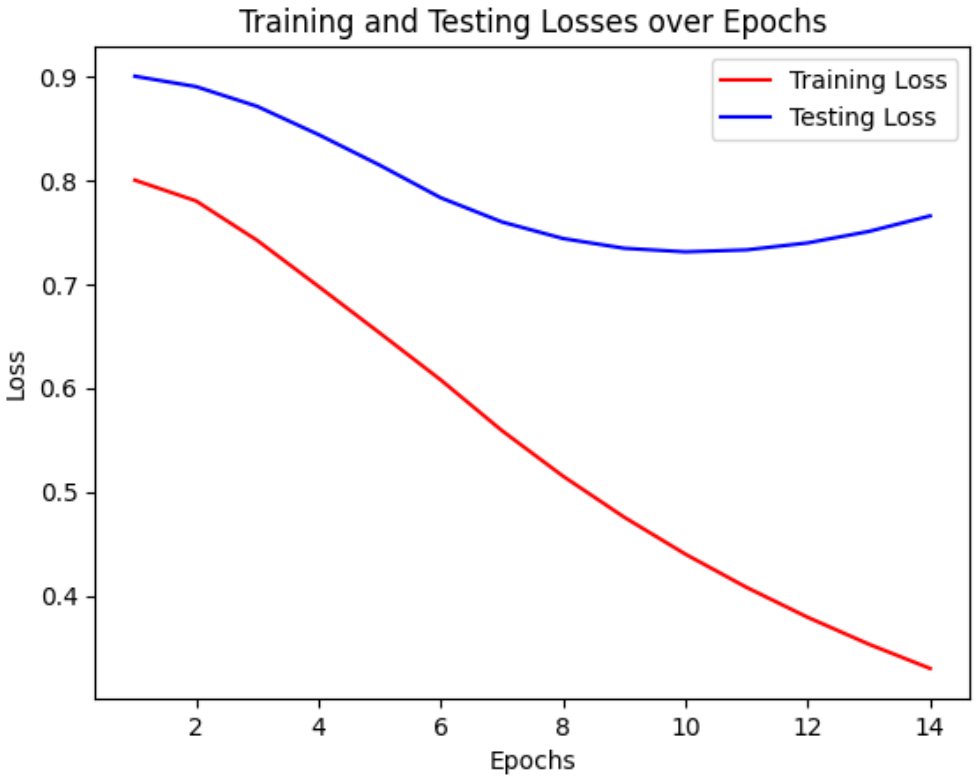

Figure 1: Training and Testing Losses over Epochs

Explain what happened and give two possible reasons for a such outcome.

Solution: The following methods can be employed to reduce the occurrence of overfitting in machine learning models:

- **Increase Training Data:** Enhance the generalization capability by increasing the volume of training data, or use data augmentation techniques to create variations of the training data.
- **Reduce Model Complexity:** Simplify the model by selecting a model with fewer parameters or by reducing the number of layers and nodes in neural networks.
- **Implement Regularization:** Apply L1 (Lasso), L2 (Ridge), or elastic net regularization to constrain the model's complexity.
- **Use Cross-Validation:** Employ techniques like k-fold cross- validation to ensure the model performs well across different subsets of the dataset.
- **Early Stopping:** Stop training when the validation performance starts to degrade, even if the training performance continues to improve.
- **Dropout:** Use dropout in neural networks, which randomly drops units (along with their connections) during training to prevent overfitting.

These strategies are effective in controlling overfitting and should be tailored based on the specific requirements and constraints of the machine learning task at hand.

3.2 Sequential Neural Network

We aim to introduce Long Short-Term Memory (LSTM) models instead of using FeedForward network for conducting sentiment analysis.

Question 11 **6 Marks**

Explain the functionality of the gates within an LSTM architecture and how they mitigate the vanishing gradient problem inherent in traditional recurrent neural networks.

Solution: Solution:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to address the vanishing gradient problem associated with standard recurrent networks. LSTMs incorporate several gates that help maintain a balance between long-term memory and short-term input relevance. These gates are:

- **Forget Gate:** The forget gate decides what information should be discarded from the cell state. It looks at the previous hidden state and the current input and passes this through a sigmoid function, which outputs numbers between 0 and 1. A 1 means "completely keep this" while a 0 means "completely get rid of this." This selective remembering helps prevent

the vanishing gradient problem by discarding irrelevant information and thus simplifying the learning process.

- **Input Gate:** The input gate determines what new information is stored in the cell state. It uses a sigmoid function to decide which values to update and a tanh function to create a vector of new candidate values that could be added to the state.
- **Output Gate:** The output gate decides what the next hidden state should be. It looks at the current input and the cell state, applying a sigmoid function to determine which parts of the cell state should be output and then uses a tanh function to scale the cell state to be between -1 and 1, which is then multiplied by the output of the sigmoid.

By carefully regulating the flow of information throughout the network, these gates allow LSTMs to mitigate the vanishing gradient problem. Specifically, they enable the network to selectively remember patterns over long time intervals, making it feasible to train on and remember information from earlier in the dataset, thereby preventing gradients from vanishing during backpropagation. This selective memory feature is key to the LSTM's ability to capture dependencies in sequence data significantly better than standard RNNs.

Question 12 **12 Marks**

Assume we want to develop a multi-layer LSTM architecture where each of the d LSTM layers outputs a vector of dimension d_p. Provide a comprehensive description of the proposed classification model architecture by addressing the following points:

- **Input Data Shape**:
  - Describe the initial shape of the input data. What does each dimension represent in the context of the problem?
- **Embedding Layer**:
  - Explain how the input shape is transformed by the embedding layer. What is the new shape and what does each dimension represent?
  - Detail the role of the embedding matrix (e.g., Word2vec) and its configuration (fixed or trainable).
- **LSTM Layers**:
  - Describe the structure and function of the LSTM layers in the network.
  - For each LSTM layer, explain how it transforms the shape of the data. What are the dimensions after each layer, and why is this transformation important for the model?
- **Final LSTM Layer**:
  - Discuss the output of the final LSTM layer. How does it differ from the previous LSTM layers in terms of data transformation and output shape?

- **Dense Output Layer**:
  - Specify the configuration of the output layer (number of units and activation function).
  - Explain how this layer transforms the output of the final LSTM layer into the final classification output. What is the final shape of the output tensor?
- **Summary of Shape Transformations**:
  - Provide a concise summary of the transformations from input to output. Include a brief description of the shape at each stage and the significance of these transformations for achieving the model's objective.

Solution:

Some students replace T by V. We are fine with that only if the rest of the answers is correct.

The proposed model is structured as a sequence of layers, processing input and transforming its shape at each stage as follows:

1. **Input Data:** The input tensor is of shape (N, T), where each entry is an integer representing a word in a sequence.
2. **Embedding Layer:**
   - Transforms the input tensor from shape (N, T) to (N, T, D), with each integer encoded into a D-dimensional vector.
   - This transformation utilizes a pre-trained embedding matrix like Word2vec. The embedding weights are not updated during training, ensuring the use of stable, pre-learned word representations.
3. **LSTM Layers:**
   - A stack of p LSTM layers processes the embedded sequence. Each layer in the stack returns a sequence of outputs, hence maintaining the second dimension T.
   - The first LSTM layer transforms the shape from (N, T, D) to (N, T, d_1), and subsequent LSTM layers transform the tensor further into (N, T, d_2), and so on, until the final LSTM layer outputs a tensor of shape (N, T, d_p).
4. **Final LSTM Layer:**
   - This layer returns only the last output of the sequence, reducing the tensor from (N, T, d_p) to (N, d), where d corresponds to the dimensionality of the last LSTM output.
5. **Dense Output Layer:**
   - The output from the last LSTM layer is fed into a dense layer consisting of 3 neurons (since $K = 3$ categories for the classification).

- A softmax activation function is applied to convert the logits to probabilities, resulting in the final output tensor of shape (N, K), where K represents the number of classes.

Summary of Shape Transformations:

- Input shape: (N, T)
- After embedding: (N, T, D)
- After p LSTM layers: (N, T, d_p)
- After the final LSTM layer: (N, d)
- Output shape: (N, K)

This architecture effectively captures the dependencies and features in sequence data, crucial for tasks involving natural language understanding and classification.