Questions:

1. Feature Engineering for Time Series Data
   - What could happen when feature engineering is not done in time series analysis?
     - Model might struggle to learn from the raw data, especially if the data contains noise, irrelevant information or complex patterns that are not easy to derive for the model from the raw data format.
     - Might result in lower model performance, because they miss out on the enriched information that well-engineered features can provide

   - Analytically, what could be the impact of using frequency domain features rather than lag features?
     - Shift from time-based perspective to frequency-based perspective
     - Frequency domain features, are able to capture periodic and cycle patterns in the data.
     - If your time series data has strong periodic components (like seasonal effects in sales data), frequency domain features can more effectively capture these than lag features.
     - Lag features are more about capturing immediate past dependencies and trends, but may not be as effective in identifying underlying cyclical patterns that span longer time periods.

   - How is rolling window statistics capturing the trend or seasonality in the data?
     - A rolling mean or moving average smooths out short-term variations and highlights longer-term trends in the data.
       - e.g. rolling mean over a 7day window can show the weekly trend in step counts.
     - The choice of the window size is crucial. It should correspond to the length of the trend or seasonal cycle
     - A rolling mean over a longer period of time e.g. 30 days can capture monthly patterns(seasonality) e.g. that there exist for some month a higher activity as for others (summer/winter)
     - While rolling window statistics are useful, they have limitations. They assume that the trend and seasonality are relatively stable over time, which might not be the case in general.

2. Feature Selection using Correlation Matrix
   - What is the importance of correlation matrix in context of feature selection? How do you interpret the correlation coefficients?
     - A correlation matrix can quantify the linear relationship between variables.
     - It measures how changes in one variable are associated with changes in another.
     - Helps identify features with a strong linear relationship with the target variable, indicating their potential predictive power.
     - The value of a correlation coefficient ranges from -1 to 1
     - A values close to 1 indicates strong positive linear relationship, values close to -1 strong negative relationship(if one variable increases the other decreases)
     - A value around 0 suggest little to no linear relationship

   - Can you think of any other way to assess the relationship between features and the target variable? How is your method different from correlation matrix?
     - Using Random Forest for feature importance
       - can be used to rank the importance of features based on how much they decrease the impurity in the model.
     - Unlike the correlation matrix, which measures linear relationships, random forest feature importance can capture non-linear relationships between features and the target.
     - A simpler way is to compute Mean Squared Error between features. It's more robust in case we have small differences. Also for repeating signals we can use already mentioned Fourier transform for the whole time and compute MSE between them. Or, if one signal lags behind the other, match them using dynamic programming.

   - How could one solve the potential limitations of correlation matrix?
     - Limitation: Can just capture linear relationships between features and target
     - Solution: Use other metrics for non-linear relationships like Spearman's rank correlation or random forests, apply a statistical or NN model to transform the features independently and then apply correlation matrix between transformed features.

3. Feature Transformation
   - Why might feature transformation be necessary?
     - If the Algorithm assume that features follow a normal or Gaussian distribution
       - Can help normalising distributions
     - If the Algorithm is sensitive to feature scales
       - Different features often have different scales, some algorithms require features to be on the same scale for optimal performance
     - Handling Categorical data
       - Transformations like one-hot encoding or label encoding are essential for converting categorical data into a numerical format that can be used in most machine learning algorithms.

   - How can feature transformation affect model performance?
     - By transforming features to better align with the assumptions of the model, its able to learn more effectively from data, leading to potential higher accuracy.

4. Sequence Preparation for LSTMs
   - What is the significance of the sequence length in LSTM models?
     - Sequence length determine how far back in time the LSTM can learn dependencies

   - How do you decide the sequence length? What are the consequences of choosing a very long or very short sequence length?
     - It depends on the given problem that u want to solve
       - Empirical Testing
       - Domain Knowledge
       - Resource Constraints
     - Shorter Sequences for Immediate Past Influence
       - If u want to predict something that depends on the direct recent past, a shorter sequence length might be helpful because the model gets not distracted from past that has no information to the prediction value
       - If sequence length is too short, the model might miss historical context to make accurate predictions
       - If model is trained on too short sequences it might oversimplify the problem and fail to capture the underlying patterns in the data. This leads to poor performance
       - Higher sensitive to noise, because it have less context to distinguish between signal and noise
     - Longer Sequences for Long-Term Dependencies
       - If u want to predict something where the longer past plays a role in it, you need a longer sequence length
       - Overfitting if the sequence is to long given a problem, This happens because the model might pick up on irrelevant detail in the loner sequences
       - LSTMs can get affected by the Vanishing or Exploding Gradients Problem when dealing with very long sequences even if its unlikely. This would make training unstable and lead to poor model performance. Their are techniques like gradient clipping to mitigate these problems
       - Usage of memory and computational resources, without need if sequence length is too long

5. Building LSTM Model using PyTorch
   - How do you decide the number of hidden layers and hidden units in an LSTM?
     - Depends on the Complexity of the task
     - Depends how much Computational Resources are available
     - Balance Bias-Variance Tradeoff
     - Empirical Testing
     - Trade-Off between performance and Efficiency

   - What could be a consequence of not using optimizer and loss function?

- No training of the model is possible. Both are essential for the training process in machine learning
- Loss function
  - Measurement of how well the models predictions match the actual data
  - Feedback mechanism for the model
- Optimizer
  - Adjust the model in an iteratively manner with respect to the loss to minimise the loss

Questions to our analysis on heart beat data:

1. Which engineered features provided the most value in the LSTM model's predictions, based on feature selection results or model performance?
   - From the linear dependencies, 'Steps' and 'Intensity_1h_before' indicate the most value according to the correlation matrix.
   - Regarding the non-linear dependencies, 'Steps' have the highest feature importance among all others, with a significant discrepancy.

2. How did the sequence length and the decision to use (or not use) overlapping sequences impact the model's training and performance?
   - With a sequence length of 10, the test MSE error was 0.0199; with a length of 5, it was 0.0194; and with a length of 20, the test error increased to 0.0191.
     - Sequence Length 5 seems to work fine. Increasing the sequence length doesn't enable much improvement.
   - When using overlapping sequences, there were 4,375 datapoints for a sequence length of 5 in training. In contrast, for non-overlapping sequences, there were 874 datapoints for the same sequence length in training, and this approach performed slightly worse on the test set, yielding a mean squared error (MSE) of 0.0209 instead of 0.0194 for overlapping sequences.

3. How do the engineered time-based features, like hour of the day, potentially influence an LSTM's ability to predict patterns in time series data?
   - Even though the correlation value is not very high (0.19), it might provide meaningful context. For example, the intensity is often zero during the night. This feature could make it easier to predict patterns like the circadian cycle.

4. In the context of time series data, what are the challenges and benefits of using overlapping sequences?
   - Benefits:
     - Overlapping sequences result in more datapoints for training and evaluation.
     - Learning patterns that capture temporal dynamics, because the model is exposed to all possible windows of data.
   - Challenges:
     - Risk of Overfitting because the model sees the same data points not just once per epoch but also up to once per sequence length.

5. Given the sequences prepared for LSTM training, how might the sequence structure or order influence the LSTM's internal state and subsequent predictions?
     - An appropriate sequence length must be determined for a given problem, ensuring that the LSTM does not process data irrelevant to solving the problem. The sequence length affects how much information the LSTM's memory cell needs to store and process.
   - After creating the sequences the order of the Training Data depends on the problem itself.
     - E.g. if you want to predict stock prices given a specific stock, u don't want to shuffle the training data (given you have overlapping sequences) because if u do so u would disrupt the temporal order, making it impossible for the model to learn meaningful temporal patterns. Without shuffling the model sees data in the true chronological order, making it able to learn how patterns develop and change over time, which is crucial for making predictions about the next time period
     - For a dataset that consist of many patients that are follow one by one and have each their own time-range, not shuffling would make the model more biased potentially to patient

specific patterns that are not generalise well. In this case shuffling is crucial to learn a Generalisation across all the patients

- An issue with our dataset is that, when creating sequences, we end up with overlapping parts between patients that contain no meaningful information, introducing noise into our training data. Although these datapoints are just a small percentage of the overall data, they don't significantly impact the overall training. However, in a real-life clinical application scenario, we would ideally sort out these datapoints.