Part 1: Short Answer Questions (30 points)

1. Problem Definition (6 points)

- **Hypothetical AI Problem:** Predicting customer churn for a telecommunications company.
- **Objectives:**
    1. Minimize customer attrition rate.
    2. Identify high-risk customers for targeted retention campaigns.
    3. Improve customer lifetime value (CLTV).
- **Stakeholders:**
    1. Marketing Department: Interested in identifying customers for targeted campaigns.
    2. Customer Service Department: Needs to understand common churn reasons and improve service.
- **Key Performance Indicator (KPI):** Reduction in monthly churn rate by X%. (e.g., Reduce monthly churn from 2% to 1.5% within 6 months).

2. Data Collection & Preprocessing (8 points)

- **2 Data Sources:**
    1. **Customer Relationship Management (CRM) System:** Contains customer demographics, service plans, contract details, and historical interactions.
    2. **Billing and Usage Data:** Records call duration, data usage, SMS activity, billing cycles, and payment history.
- **1 Potential Bias:**
    - **Selection Bias:** If the historical data primarily reflects churn among a specific demographic or usage pattern (e.g., only data from urban areas, or only data from customers on specific older plans), the model might not generalize well to other customer segments, leading to inaccurate predictions for those groups.
- **3 Preprocessing Steps:**
    1. **Handling Missing Data:** Impute missing values (e.g., using mean/median for numerical features, mode for categorical features, or more sophisticated

methods like K-Nearest Neighbors imputation) or remove rows/columns with excessive missingness.

2. **Normalization/Standardization:** Scale numerical features (e.g., call duration, data usage) to a common range or distribution (e.g., min-max scaling to [0,1] or standardization to mean 0 and standard deviation 1). This is crucial for distance-based algorithms and can help gradient-based optimization.
3. **Encoding Categorical Features:** Convert categorical variables (e.g., service plan type, payment method) into a numerical format that machine learning models can understand (e.g., One-Hot Encoding for nominal categories or Label Encoding for ordinal categories).

3. Model Development (8 points)

- **Chosen Model & Justification:**
  - **Model:** Gradient Boosting Machine (GBM) / XGBoost.
  - **Justification:** GBMs are powerful ensemble methods known for their high accuracy and ability to handle various data types. They are particularly effective for structured/tabular data like customer churn datasets. They can capture complex non-linear relationships and are less prone to overfitting than some other complex models (like deep neural networks) on moderately sized tabular datasets, given proper tuning. They also provide feature importance scores, which can be valuable for understanding factors contributing to churn.
- **Data Splitting:**
  - **Training Set (e.g., 70%):** Used to train the model. The model learns the underlying patterns and relationships from this data.
  - **Validation Set (e.g., 15%):** Used during the model development phase for hyperparameter tuning and early stopping. It helps prevent overfitting to the training data and provides an unbiased estimate of model performance during training.
  - **Test Set (e.g., 15%):** A completely unseen dataset used only once at the very end to evaluate the final model's generalization performance. It provides an unbiased estimate of how the model will perform on new, real-world data.
- **2 Hyperparameters to Tune & Why:**

1. **n_estimators (Number of boosting stages/trees):** This controls the number of weak learners (decision trees) in the ensemble.
    - **Why tune:** Too few trees might lead to underfitting, while too many can lead to overfitting and increased computational cost. Tuning helps find the optimal number that balances bias and variance.
2. **learning_rate (Shrinkage factor):** This determines the step size at each boosting iteration.
    - **Why tune:** A smaller learning rate requires more trees but makes the model more robust to overfitting. A larger learning rate converges faster but can overshoot the optimal solution. Tuning helps find a balance for optimal convergence and performance.

4. Evaluation & Deployment (8 points)

- **2 Evaluation Metrics & Relevance:**
    1. **F1-Score:**
        - **Relevance:** F1-score is the harmonic mean of precision and recall. In customer churn prediction, the dataset is often imbalanced (far fewer churners than non-churners). F1-score is particularly useful in such cases because it balances the concern for correctly identifying churners (recall) with the concern for not incorrectly identifying non-churners as churners (precision), which could lead to wasted retention efforts.
    2. **Area Under the Receiver Operating Characteristic Curve (AUC-ROC):**
        - **Relevance:** AUC-ROC measures the model's ability to distinguish between the positive and negative classes across various classification thresholds. A higher AUC indicates that the model is better at separating churners from non-churners. It's robust to imbalanced datasets and gives a comprehensive view of the model's performance without having to choose a specific threshold.
- **Concept Drift & Monitoring Post-Deployment:**
    - **Concept Drift:** Occurs when the relationship between the input variables and the target variable changes over time. In our customer churn example, this could mean that the factors influencing churn (e.g., competitor offerings, economic conditions, new service features) evolve, making the old model's learned patterns less relevant.

- **Monitoring Post-Deployment:**
    - **Regular Retraining/Recalibration:** Periodically retrain the model with fresh data to adapt to new patterns.
    - **Monitoring Prediction Drift:** Compare the distribution of the model's predictions (e.g., churn probability scores) over time with the distribution observed during training. Significant shifts could indicate concept drift.
    - **Monitoring Feature Drift:** Track changes in the distribution of input features over time. Changes in customer usage patterns, demographics, or billing cycles could signal underlying concept drift.
    - **Performance Monitoring on Latest Data:** Continuously evaluate the model's performance (e.g., F1-score, AUC) on recently collected, labeled data (if available or through periodic manual labeling). A sustained decline in performance is a strong indicator of concept drift.

- **1 Technical Challenge during Deployment:**
    - **Scalability and Latency:** The deployed model needs to handle a high volume of prediction requests (e.g., real-time churn risk assessment for millions of customers) with low latency. Ensuring the infrastructure (e.g., servers, APIs) can scale horizontally and vertically to meet demand without compromising response times is a significant challenge. This involves choosing appropriate serving frameworks (like TensorFlow Serving, BentoML), designing efficient APIs, and potentially using containerization and orchestration tools (like Docker, Kubernetes).

Part 2: Case Study Application (40 points)

Scenario: A hospital wants an AI system to predict patient readmission risk within 30 days of discharge.

Problem Scope (5 points):

- **Problem:** To accurately predict which patients are at high risk of readmission to the hospital within 30 days of discharge.
- **Objectives:**
    1. Reduce the 30-day patient readmission rate.
    2. Enable proactive intervention for high-risk patients (e.g., enhanced post-discharge care, follow-up calls).

3. Improve patient outcomes and satisfaction.
4. Optimize hospital resource allocation.

- **Stakeholders:**
    1. **Hospital Administration:** Interested in reducing readmission penalties, improving efficiency, and enhancing patient care quality.
    2. **Clinicians (Doctors, Nurses):** Need actionable insights to identify at-risk patients and tailor care plans.
    3. **Patients and their Families:** Directly benefit from reduced readmissions and better health outcomes.
    4. **Health Insurance Providers:** Interested in reducing healthcare costs associated with avoidable readmissions.

Data Strategy (10 points):

- **Proposed Data Sources:**
    1. **Electronic Health Records (EHRs):** Rich source of patient demographics (age, gender), medical history (diagnoses, comorbidities, previous hospitalizations), medications, lab results, vital signs, discharge summaries, and clinical notes.
    2. **Billing and Claims Data:** Provides information on procedures, treatments received, and potentially insights into socioeconomic status or insurance coverage.
    3. **Socio-Demographic Data:** Could be external data sources linked by anonymized patient IDs, providing insights into factors like income level, education, living conditions, and access to transportation (social determinants of health).
    4. **Patient Self-Reported Data/Surveys:** Information on lifestyle factors, support systems at home, or adherence to medication, if collected.
- **2 Ethical Concerns:**
    1. **Patient Privacy and Data Security (HIPAA Compliance):** Handling sensitive patient health information (PHI) requires strict adherence to privacy regulations like HIPAA (Health Insurance Portability and Accountability Act). Breaches could lead to severe legal and reputational consequences.
    2. **Algorithmic Bias and Fairness:** The model might inadvertently learn biases present in historical data. For example, if certain demographic groups have

historically received different levels of care or have less complete data in EHRs, the model might disproportionately misclassify their readmission risk, leading to unequal access to intervention programs. This could exacerbate existing health disparities.

- **Preprocessing Pipeline (including Feature Engineering steps):**
    1. **Data Cleaning:**
        - **Handling Missing Values:** Impute missing lab results (e.g., median imputation), discharge dates (if missing, might flag an issue), or demographic information. For sparse clinical notes, consider strategies like marking as 'unknown' or using advanced NLP.
        - **Outlier Detection and Treatment:** Identify and handle extreme values in numerical data (e.g., unusually high or low vital signs, lab results) which might be data entry errors or genuine but rare cases needing careful consideration.
        - **Standardizing Data Formats:** Ensure consistent date formats, units of measurement (e.g., mg vs. g for medications), and coding systems (e.g., ICD codes for diagnoses).
    2. **Feature Engineering:**
        - **Comorbidity Index:** Create a score (e.g., Elixhauser Comorbidity Index or Charlson Comorbidity Index) based on the presence of multiple chronic conditions from diagnosis codes.
        - **Medication Adherence Features:** Extract features related to the number of prescribed medications at discharge, potential drug-drug interactions, or history of non-adherence.
        - **Length of Stay:** Calculate the duration of the current hospitalization. Longer stays might indicate higher severity or more complex conditions.
        - **Number of Previous Admissions:** Count the number of past hospitalizations within a specific timeframe (e.g., last 12 months) as a strong predictor of future readmission.
        - **Time Since Last Admission:** Calculate the days since the previous discharge.

        - **Demographic Ratios/Interactions:** Create features like "age_gender_interaction" to capture complex relationships.
        - **Admission Type and Source:** One-hot encode features like "Emergency," "Elective," "Transfer from another hospital," etc.
        - **Clinical Note Features (if using NLP):** Extract keywords, sentiment, or topic models from discharge summaries to capture nuanced clinical information not explicitly coded.
    3. **Feature Scaling/Normalization:** Apply standard scaling to continuous numerical features (e.g., age, lab values, number of medications) to ensure they contribute equally to distance-based models and to improve the convergence of gradient-based optimizers.
    4. **Encoding Categorical Features:** One-hot encode nominal categorical features like "diagnosis codes" (if fewer distinct codes), "admission type," "discharge disposition," or "insurance type." For features with many categories (like specific ICD codes), consider dimensionality reduction techniques or target encoding.

Model Development (10 points):

- **Selected Model & Justification:**
    - **Model:** LightGBM (Light Gradient Boosting Machine).
    - **Justification:**
        - **High Performance:** LightGBM is known for its speed and accuracy, often outperforming other boosting algorithms on large datasets.
        - **Handles Tabular Data Well:** Excellent for the structured nature of EHR and administrative data.
        - **Feature Importance:** Provides feature importance scores, which can be crucial in healthcare for understanding what factors contribute to readmission risk, aiding clinical interpretability.
        - **Handles Missing Values (to some extent):** Can handle missing values internally, reducing the need for extensive imputation (though prior imputation is still recommended).
        - **Scalability:** Efficient with large datasets, which is common in healthcare.

        - **Interpretability (relative to deep learning):** While complex, it's generally more interpretable than deep neural networks, allowing clinicians to gain some insight into the drivers of risk.

- **Confusion Matrix and Precision/Recall (Hypothetical Data):**

Let's assume, for a hypothetical test set, the following confusion matrix:

	Predicted: Readmitted (Positive)	**Predicted: Not Readmitted (Negative)**
Actual: Readmitted (Positive)	150 (True Positives, TP)	50 (False Negatives, FN)
Actual: Not Readmitted (Negative)	20 (False Positives, FP)	780 (True Negatives, TN)

Export to Sheets

    - **Total Actual Readmitted:** TP + FN = 150 + 50 = 200
    - **Total Actual Not Readmitted:** FP + TN = 20 + 780 = 800
    - **Total Predictions:** 150 + 50 + 20 + 780 = 1000
    - **Precision (of 'Readmitted' class):**
        - Precision = TP / (TP + FP)
        - Precision = 150 / (150 + 20) = 150 / 170 approx **0.882** (88.2%)
        - *Interpretation:* When the model predicts a patient will be readmitted, it is correct about 88.2% of the time.
    - **Recall (of 'Readmitted' class):**
        - Recall = TP / (TP + FN)
        - Recall = 150 / (150 + 50) = 150 / 200 = **0.75** (75%)
        - *Interpretation:* The model correctly identifies 75% of all actual readmitted patients.

Deployment (10 points):

- **Steps to Integrate the Model into the Hospital's System:**

1. **API Development:** Create a RESTful API endpoint for the trained model. This API will expose a prediction service where clinical systems can send patient data (e.g., patient ID, recent lab results, diagnoses) and receive a readmission risk score.
2. **Containerization:** Package the model, its dependencies, and the API code into a Docker container. This ensures consistency across different environments and simplifies deployment.
3. **Deployment Platform Selection:** Choose a robust and secure deployment platform suitable for healthcare data. Options include on-premise servers (for strict data control), cloud-based ML platforms (AWS SageMaker, Azure ML, Google Cloud Vertex AI) with proper security configurations, or Kubernetes for orchestration.
4. **Integration with EHR/Clinical Workflow:**
    - **Real-time Prediction:** Integrate the API directly into the EHR system or a clinical decision support system. When a patient is nearing discharge, relevant data is automatically sent to the API, and the risk score is displayed to the clinician (e.g., as part of the discharge summary or in a dedicated dashboard).
    - **Alerting System:** Develop a system to trigger alerts or notifications for high-risk patients to relevant care teams (e.g., care coordinators, social workers) so they can initiate targeted interventions.
5. **Logging and Monitoring:** Implement robust logging of all prediction requests and responses. Set up real-time monitoring dashboards to track model performance, latency, throughput, and data drift.
6. **Security Measures:** Implement strong authentication, authorization, and encryption (in transit and at rest) for all data exchanged with the model. Regular security audits are crucial.

- **Ensuring Compliance with Healthcare Regulations (e.g., HIPAA):**
    1. **Data De-identification/Anonymization:** Wherever possible, patient data used by the model (especially during training and in logs) should be de-identified or anonymized to remove personally identifiable information (PII) while retaining clinical utility.

2. **Access Control and Authorization:** Implement strict role-based access control (RBAC) to ensure only authorized personnel and systems can access the model API and the underlying data. All access should be logged and audited.
3. **Data Encryption:** Encrypt all patient data, both when it's at rest (e.g., in databases, storage) and in transit (e.g., via HTTPS for API calls).
4. **Secure Hosting Environment:** Deploy the model on infrastructure that meets HIPAA compliance standards (e.g., dedicated HIPAA-compliant cloud regions, on-premise secure servers).
5. **Audit Trails:** Maintain comprehensive audit trails for all data access, model inferences, and system changes to demonstrate compliance.
6. **Data Minimization:** Only collect and process the minimum amount of patient data necessary for the model to make predictions.
7. **Consent and Transparency:** Ensure patients are informed about how their data is being used for AI-driven predictions and obtain necessary consents where required. The logic of the model, to the extent possible, should be explainable.
8. **Regular Security Assessments:** Conduct periodic security audits, penetration testing, and vulnerability assessments to identify and address potential weaknesses.
9. **Data Retention Policies:** Implement clear policies for data retention and destruction, ensuring sensitive data is not kept longer than necessary.

Optimization (5 points):

- **1 Method to Address Overfitting:**
    - **Regularization (e.g., L1/L2 Regularization in LightGBM, or Dropout for Neural Networks):**
        - **Explanation:** Regularization techniques add a penalty term to the model's loss function during training, discouraging the model from assigning excessively large weights to specific features. This prevents the model from becoming too complex and fitting the noise in the training data.
        - **In the context of LightGBM:** LightGBM has parameters like lambda_l1 and lambda_l2 (for L1 and L2 regularization, respectively) and num_leaves (controlling tree complexity). By increasing the regularization parameters or reducing the maximum number of leaves per tree, we can

constrain the model's capacity and make it generalize better to unseen data, thereby mitigating overfitting.

- **Other relevant methods:** Early stopping (stopping training when validation performance degrades), cross-validation, and acquiring more diverse training data.

Part 3: Critical Thinking (20 points)

Ethics & Bias (10 points):

- **How might biased training data affect patient outcomes in the case study?** Biased training data can have severe and inequitable consequences for patient outcomes in the readmission prediction system:

    1. **Discriminatory Risk Assessment:** If the historical data contains systemic biases (e.g., certain racial or socioeconomic groups historically received less comprehensive follow-up care or were documented differently in EHRs), the model might inaccurately predict their readmission risk. It could overestimate risk for some groups, leading to over-intervention (and potentially unnecessary resource allocation), or underestimate risk for others, leading to under-intervention.
    2. **Exacerbation of Health Disparities:** If the model disproportionately misses high-risk patients from underserved communities due to biased data, these patients may not receive the necessary post-discharge support. This would worsen existing health disparities, as those already vulnerable are further disadvantaged by the AI system.
    3. **Misallocation of Resources:** An AI system that biases risk prediction could lead to misallocation of hospital resources. For example, if it falsely identifies low-risk patients as high-risk, resources might be diverted from genuinely high-risk patients. Conversely, if it misses high-risk patients, the hospital might not deploy interventions where they are most needed.
    4. **Erosion of Trust:** If patients or clinicians perceive the AI system to be unfair or discriminatory, it can erode trust in the technology, leading to low adoption rates and resistance to using a tool that could otherwise be beneficial.

5. **Legal and Reputational Risks:** Deploying a biased AI system can expose the hospital to legal challenges (e.g., discrimination lawsuits) and significant reputational damage.

- **Suggest 1 strategy to mitigate this bias.**
    - **Fairness-Aware Data Collection and Preprocessing:**
        - **Strategy:** Actively seek to understand and quantify existing biases in the historical data, particularly concerning protected attributes (e.g., race, gender, socioeconomic status). This involves **disaggregating data analysis** to examine the distribution and quality of data across different demographic groups.
        - **Mitigation Steps:**
            - **Bias Detection:** Use fairness metrics (e.g., disparate impact, equal opportunity difference) to identify if certain demographic groups are underrepresented or if their data quality differs significantly.
            - **Re-sampling or Re-weighting:** If specific groups are underrepresented or overrepresented in the training data, employ techniques like oversampling the minority class, undersampling the majority class, or re-weighting individual samples to balance the dataset's representation across sensitive attributes.
            - **Feature Engineering for Fairness:** Create or modify features to reduce reliance on proxies for sensitive attributes that might introduce bias. For instance, instead of relying solely on zip codes (which can correlate with socioeconomic status and race), consider more direct measures of social determinants of health if available and ethical.
            - **Data Augmentation:** Generate synthetic data for underrepresented groups, carefully ensuring it reflects realistic patterns without introducing new biases.

Trade-offs (10 points):

- **Discuss the trade-off between model interpretability and accuracy in healthcare.** In healthcare, the trade-off between model interpretability and accuracy is a critical ethical and practical consideration:

- **Accuracy:** A highly accurate model provides the best possible predictions, leading to potentially superior patient outcomes (e.g., correctly identifying more high-risk patients, leading to more targeted interventions and fewer readmissions). Complex models like deep neural networks or ensemble methods (e.g., XGBoost, LightGBM) often achieve the highest accuracy on complex tasks.
- **Interpretability:** An interpretable model allows clinicians to understand *why* a particular prediction was made. They can see which features or combinations of features contributed most to a patient's high or low readmission risk. This is vital because:
    - **Trust and Adoption:** Clinicians are more likely to trust and adopt a system if they understand its reasoning, rather than relying on a "black box."
    - **Accountability:** If an error occurs, interpretability helps in diagnosing the cause, identifying data issues, or understanding model limitations.
    - **Clinical Validation and Insight:** Interpretability can help clinicians validate the model's logic against their medical knowledge, and even gain new insights into disease progression or risk factors.
    - **Ethical Considerations:** Explaining *why* a patient is deemed high-risk is crucial for informed decision-making and for addressing potential biases.
- **The Trade-off:** Often, the most accurate models (e.g., deep learning) are the least interpretable ("black box" models), while highly interpretable models (e.g., linear regression, simple decision trees) may sacrifice some predictive accuracy.
- **Implications in Healthcare:**
    - **High-Stakes Decisions:** In healthcare, where decisions impact patient lives, interpretability is often highly valued, even if it means a slight reduction in absolute accuracy. A misdiagnosis or missed risk, even if rare, has severe consequences.
    - **Regulatory Scrutiny:** Regulatory bodies often prefer interpretable models for auditability and accountability.
    - **Hybrid Approaches:** The trend is towards finding a balance: using powerful models for accuracy but employing **post-hoc interpretability techniques** (e.g., SHAP, LIME, permutation importance) to explain their predictions. This allows leveraging the accuracy of complex models while still providing insights into their decision-making process.

- **If the hospital has limited computational resources, how might this impact model choice?** Limited computational resources would significantly constrain the choice of model, favoring less computationally intensive options:

    1. **Simpler Models:** The hospital would likely need to opt for simpler, less complex models.

        - **Preferred:** Logistic Regression, Decision Trees, Naive Bayes, Support Vector Machines (with linear kernels), or simpler ensemble methods like Random Forest (with fewer trees/depth) would be more feasible. These models require less memory and processing power for training and inference.

        - **Less Preferred/Avoided:** Deep Neural Networks (especially large ones), complex ensemble methods with many estimators, or models requiring extensive hyperparameter searches (like large-scale Bayesian optimization) would be computationally prohibitive. Training these can take days or weeks on powerful GPUs/TPUs, which would be unavailable.

    2. **Data Size and Feature Complexity:** The ability to handle very large datasets or highly engineered features (e.g., complex NLP embeddings) would be limited. Preprocessing might need to be less intensive or simpler feature engineering techniques would be chosen.

    3. **Training Time:** Training times would be a major factor. Models that can be trained quickly on standard CPUs would be prioritized, allowing for more frequent retraining if necessary.

    4. **Inference Speed (Prediction Time):** Even if a model could be trained offline, its ability to provide real-time predictions (low latency) would be crucial for clinical integration. Complex models can have higher inference times, impacting the user experience.

    5. **Deployment Footprint:** Models with smaller memory footprints are easier to deploy on resource-constrained servers, potentially reducing infrastructure costs.

In essence, resource limitations would push the hospital towards a trade-off where simplicity, speed, and efficiency take precedence over achieving peak accuracy with highly complex models.