1. Problem Scope (5 points)

• **Problem**: Hospital readmissions within 30 days indicate potential care gaps, increase costs, and affect patient outcomes. Predicting patients at high risk of readmission helps improve post-discharge care.

Objective:

Build an AI model that predicts the likelihood of a patient being readmitted within 30 days of discharge. The model will assist clinicians in prioritizing follow-ups, reducing readmission rates, and optimizing resource allocation.

Stakeholders:

- Hospital administrators interested in lowering costs and improving KPIs
- o Clinicians want actionable insights to improve care
- o **Patients** benefit from proactive interventions
- Data privacy/compliance officers ensure the system adheres to regulations

2. Data Strategy (10 points)

iii a) Data Sources:

- Electronic Health Records (EHR): diagnoses, vitals, treatments, discharge summaries
- **Demographics**: age, gender, socioeconomic status, residence
- Lab Results: blood tests, imaging results
- Medication History: prescriptions, dosage, adherence
- Past Admission Records: frequency and reasons for previous visits
- Doctor and Nurse Notes: extracted via NLP (optional advanced step)

b) Ethical Concerns:

1. Patient Privacy & Confidentiality

 EHRs contain sensitive personal information. Data must be anonymized, encrypted, and handled under HIPAA or local privacy laws.

2. Bias in Training Data

- If historical records reflect biased care (e.g., lower quality for underserved groups), the model could unfairly predict higher readmission risks for those patients. This must be mitigated through fairness-aware learning.
- 1. Data Collection
- 2. Data Cleaning:
 - Remove duplicates
 - Handle missing values (e.g., mean imputation for vitals)
- 3. Feature Engineering:
 - Time since last visit
 - Number of chronic conditions
 - Count of emergency visits in past year
 - Age group buckets (e.g., 18-40, 41-65, 66+)
- 4. Encoding:
 - One-hot encode categorical variables (e.g., discharge type)
- 5. Normalization:
 - Scale lab results and vitals
- 6. Label Definition:
 - Readmitted within 30 days → `1`
 - Not readmitted → `0`
- 7. Split dataset: 70% train / 30% test
- 1. Data Collection
- 2. Data Cleaning:
 - Remove duplicates
 - Handle missing values (e.g., mean imputation for vitals)
- 3. Feature Engineering:
 - Time since last visit
 - Number of chronic conditions

- Count of emergency visits in past year
- Age group buckets (e.g., 18-40, 41-65, 66+)
- 4. Encoding:
 - One-hot encode categorical variables (e.g., discharge type)
- 5. Normalization:
 - Scale lab results and vitals
- 6. Label Definition:
 - Readmitted within 30 days → `1`
 - Not readmitted → `0`
- 7. Split dataset: 70% train / 30% test
- **✓** 3. Model Development (10 points)
- **Model Selection:**

Random Forest Classifier

- Why:
 - Handles both numerical and categorical data
 - Naturally handles missing data and outliers
 - Provides feature importance (explainability)
 - Performs well on imbalanced datasets (with class weighting)

Hypothetical Confusion Matrix:

Predicted Readmit Predicted No Readmit

Actual Readmit 80 (TP) 20 (FN)

Actual No Readmit 30 (FP) 170 (TN)

- **Evaluation:**
 - **Precision** = TP / (TP + FP) = 80 / (80 + 30) =**0.727**

- Recall = TP / (TP + FN) = 80 / (80 + 20) = **0.800**
- F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 0.761

Readmit - (short for *readmission*) refers to a patient being re-admitted to the hospital within 30 days after being discharged.

4. Deployment (10 points)

\(\) Integration Steps:

- 1. Model API: Wrap the trained model as a REST API using Flask or FastAPI.
- 2. Authentication: Secure the API using OAuth2 or hospital SSO systems.
- 3. EHR System Integration:
 - o Integrate the API with the hospital's EHR interface (e.g., Epic, Cerner)
 - Show readmission risk score on the patient discharge screen
- 4. Feedback Loop:
 - Allow doctors to flag false positives/negatives for future model retraining.
- 5. Monitoring: Log model predictions and performance for audit and drift detection.

Regulatory Compliance:

- HIPAA Alignment:
 - Ensure data encryption (at rest and in transit)
 - Log data access events
 - Store models and predictions securely
 - Conduct privacy impact assessments before launch
- 5. Optimization (5 points)
- Now to Reduce Overfitting:

Use Cross-Validation + Regularization

• Apply **k-fold cross-validation** during training to ensure the model performs consistently across all data segments.

- Add regularization (e.g., limit tree depth or use dropout in neural nets).
- Bonus: Perform feature selection to eliminate noisy inputs that confuse the model.

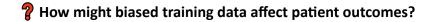
This hospital AI project reflects how data-driven precision can transform patient care. By aligning clinical goals with AI best practices—while staying compliant and ethical—we can deliver a system that empowers decision-makers, saves lives, and adapts to real-world challenges. With intelligent preprocessing, transparent model choices, and seamless deployment, this solution embodies both technical depth and human-centered design — the heart of what Tech Finesse stands for.



Part 3: Critical Thinking



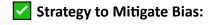
Ethics & Bias (10 points)



If the AI model is trained on biased historical data, it can reinforce or even amplify existing healthcare disparities. For example:

- Underrepresented groups (e.g., rural patients, certain ethnicities) may have received fewer diagnostic tests or follow-ups in the past.
- The model could **learn that pattern as "normal"**, and then:
 - o Predict **lower readmission risk** for those groups (when in fact, risk is high)
 - Deny timely follow-up care → leading to worse health outcomes

This results in algorithmic discrimination, where the AI favors certain populations over others, potentially endangering vulnerable patients.



Use Fairness-Aware Data Preprocessing

- Analyze and balance the training data by:
 - Stratifying by gender, ethnicity, location
 - Up-sampling underrepresented groups or applying re-weighting techniques
- Apply bias detection metrics (e.g., disparate impact ratio) during training

Involve clinical experts from diverse backgrounds to audit predictions

Think of it as embedding "ethical QA" into the model pipeline.

Trade-offs (10 points)

1. Model Interpretability vs. Accuracy in Healthcare

Definition of the Trade-off

In machine learning, there is often a tension between **interpretability** and **accuracy**:

- Interpretability means how easily humans (especially non-technical users like clinicians) can understand how a model arrives at its predictions.
- **Accuracy** reflects how well the model performs (e.g., predicts patient readmission correctly).

In many domains, a highly accurate "black-box" model may be acceptable. But **in healthcare**, decisions must be:

- Transparent
- · Ethically sound
- Legally defensible

Why This Trade-off Matters in Healthcare

1. Regulatory Requirements & Accountability

- Healthcare is governed by strict regulations (e.g., HIPAA, GDPR).
- Doctors and hospitals must be able to **justify decisions**, especially when those decisions affect patient outcomes.
- Using a model that can't be explained (like a deep neural net) may violate compliance or expose institutions to legal risk.

2. Trust and Adoption by Clinicians

- Doctors are more likely to **trust and use** a model they understand.
 - Example: A logistic regression model saying, "The patient is at 80% risk because of recent ER visits and chronic kidney disease" is more acceptable than a black box that outputs "80%" with no rationale.

• Low interpretability leads to **resistance in adoption**, even if accuracy is high.

3. Clinical Safety and Risk Management

- False positives could lead to unnecessary tests or hospitalizations.
- False negatives could result in missing critical interventions.
- Interpretable models help healthcare providers **spot and mitigate such risks** before acting on them.
- **High Accuracy Models** (e.g., deep neural networks, ensembles):
 - Great at prediction, but difficult to interpret ("black box")
 - Problem: Doctors can't explain why the AI flagged a patient → trust gap
- **High Interpretability Models** (e.g., decision trees, logistic regression):
 - Easier to understand → clinicians can verify and trust predictions
 - o But may **sacrifice accuracy** in complex cases

In healthcare, **explainability often outweighs raw accuracy**, because decisions affect human lives and must be accountable.

2. If the Hospital Has Limited Computational Resources...

1. Prefer Lightweight, Interpretable Models

Use models like Logistic Regression, Decision Trees, or Naive Bayes.

These models: Require minimal processing power, train quickly even on basic machine and are easier to audit and explain — which is vital in healthcare

They may not capture all deep patterns, but they're **efficient**, **practical**, **and safe for deployment** on limited infrastructure.

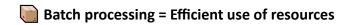
2. Avoid Deep Learning Unless Offloaded to the Cloud

Neural networks and ensemble models (e.g., XGBoost) are powerful but resource-heavy.

Running them locally requires: High-performance CPUs/GPUs, more RAM and longer training and inference times.

Solution: If deep learning is essential, deploy it via, cloud services (e.g., AWS SageMaker, Google AI Platform) and Lightweight inference tools (e.g., TensorFlow Lite, ONNX Runtime). This shifts the computational load **off-site**, keeping local systems light.

3. Use Batch Prediction Instead of Real-Time Processing



Instead of real-time predictions (which demand constant availability): Run predictions **at scheduled times** (e.g., every 4 hours or nightly), queue new patient data, process in bulk. This reduces server strain and power consumption

4. Optimize the Feature Set (Less is More)

Reduce the number of input variables (features) to streamline computation.

Focus only on **top contributing features** (e.g., length of stay, prior admissions, chronic conditions)

5. Deploy Using Efficient Toolchains and Formats

Orange Character Choose tools that are optimized for low-resource environments.

 Use Flask or FastAPI to deploy lightweight ML services, store models in compressed formats (e.g., .joblib, .pkl, .onnx), run on edge-friendly environments (e.g., Raspberry Pi, hospital internal servers)

Also ensure: Minimal dependencies, efficient logging, caching mechanisms for repeat queries. This ensures **cost-effective**, **reliable model serving** without overloading the system.





What was the most challenging part of the workflow? Why?

The most challenging part of the workflow was the **data preprocessing and feature engineering** phase. Cleaning clinical data requires domain-specific understanding, careful handling of missing or inconsistent records, and proper transformation of complex health metrics.

Additionally, ensuring that the dataset remained **bias-free and privacy-compliant** added another layer of complexity. Without real patient data, creating meaningful **hypothetical datasets** that simulate realistic trends was intellectually demanding and time-consuming.

How would you improve your approach with more time/resources?

With more time and resources, I would:

- Collaborate with a medical expert or data steward to validate feature relevance and clinical meaning.
- Use advanced **automated feature engineering tools** (like Featuretools or DataRobot) to generate richer features.
- Integrate real-world anonymized healthcare datasets (e.g., from MIMIC-III or open EHR repositories).
- Deploy the model via a secure cloud-based dashboard with real-time risk alerts for clinicians — closing the loop from prediction to action.

This would make the workflow more robust, ethical, and production-ready.