AI Development Workflow Assignment: Theoretical Answers

Part 1:

1. Problem Definition

I chose to work on a hypothetical problem: predicting which students might drop out of a university. This is a big deal because losing students hurts both the university and the students themselves, who miss out on opportunities. By catching at risk students early we can step in and help them stay on track.

Objectives:

- Catch students at risk of dropping out early so we can offer support like tutoring or counseling.
- Boost the university's retention rate by at least 10% over two years.
- Make sure resources like academic advisors are used efficiently by focusing on the students who need help most.

Stakeholders:

- The university administration—they are the ones who will use this system to plan interventions and keep enrollment steady.
- The students themselves—they benefit from getting the support they need to succeed.

Key Performance Indicator (KPI):
I would measure success by the **F1-score** of the predictions meaning what percentage of students we correctly flag as at risk. This gives us a clear way to know if the model is doing its job.

---

2. Data Collection and Preprocessing

To build this dropout prediction system we need solid data to work with.

Data Sources:

- University records like grades attendance and course enrollment details. These show how students are performing and engaging.
- Student surveys that ask about things like their financial situation or mental health. This helps us understand personal factors that might lead to dropping out.

Potential Bias:
One issue could be that surveys might miss students who are already checked out. If disengaged students do not fill out the survey our data could be skewed making it seem like fewer students are at risk than actually are.

Preprocessing Steps:

- Handle missing data: If some grades are missing I would fill them in with the median grade for that course to avoid messing up the model.
- Normalization: I would scale things like GPA to a 0-to-1 range so the model does not get confused by different scales.
- Encoding: For categories like a student's major I would turn them into one hot encodings (like 0s and 1s) so the model can process them.

---

3. Model Development

For this problem I would go with a Random Forest model.

Why Random Forest? It is great because it can handle both numbers (like grades) and categories (like major) without much hassle. Plus it is less likely to overfit compared to other models and we can see which factors like low attendance matter most for dropouts.

Data Split:
I would split the data into three parts: 70% for training (to teach the model) 15% for validation (to tweak it) and 15% for testing (to check how it performs on new data). I would make sure the split keeps the same balance of at risk and not at risk students so the model does not get biased.

Hyperparameters to Tune:

- Number of trees: More trees can make the model better but too many slow it down so I would test different amounts.
- Maximum depth: This limits how deep each tree grows to prevent overfitting and I would try different depths to find the sweet spot.

---

4. Evaluation and Deployment

Once the model is built we need to check if it is good and figure out how to use it in the real world.

Evaluation Metrics:

- F1-Score: Since dropout data might be imbalanced (fewer dropouts than non-dropouts) F1 balances precision (how many flagged students actually drop out) and recall (how many at risk students we catch).
- AUC-ROC: This shows how well the model separates at risk students from others which is great for understanding overall performance.

Concept Drift:
Concept drift happens when the model starts failing because the data it's seeing changes over time like if new academic policies change how students behave. To monitor it I would track the F1-Score monthly. If it drops by 5% or more I would retrain the model with fresh data.

Technical Challenge:
One issue during deployment could be scalability. If the university has thousands of students running predictions for everyone might be slow. To fix this I would process data in batches to keep things running smoothly.

Part 2: Case Study Application (Theoretical Part)

Problem Scope

The case study is about building an AI system for a hospital to predict which patients are likely to be readmitted within 30 days of leaving. This matters because readmissions are costly and can mean patients aren't getting the care they need.

Problem: Create a model that flags patients at high risk of readmission so the hospital can step in with extra care.

Objectives:

- Cut down readmission rates by focusing on high risk patients.
- Help patients stay healthier with tailored follow-up plans.
- Save the hospital money by reducing unnecessary readmissions.

Stakeholders:

- Hospital staff like doctors and nurses who will use the predictions to plan care.
- Patients who get better support to avoid coming back to the hospital.

Data Strategy

To make this work we need good data but we also have to be careful about ethics.

Data Sources:

- Electronic Health Records (EHRs): These include diagnoses, medications and lab results which show a patient's health history.
- Demographics: Things like age gender and income level which can affect readmission risks.

Ethical Concerns:

- Patient privacy: EHRs have sensitive info and a data breach could be a disaster. We need to keep everything secure.
- Bias: If historical data has fewer records from minority groups the model might not work as well for them which is not fair.

Preprocessing Pipeline:
Here is how I would get the data ready:

- Feature Engineering:
    - Create a yes/no feature for whether a patient has a chronic illness, like diabetes.
    - Calculate how many days since their last hospital stay.
- Steps:
    - Remove duplicate patient records to avoid confusion.
    - Fill in missing lab results with the average for that test.
    - Normalize numbers like blood pressure to a 0-to-1 scale.
    - Turn categories like diagnosis codes into one-hot encodings for the model.

Model Development (Theoretical Part)

I would use Logistic Regression for this.

Why Logistic Regression? It's simple, easy to explain to doctors, and works well for yes/no predictions like readmission risk. In healthcare being able to explain why a patient's flagged is super important and this model makes that straightforward.

Deployment

Getting the model into the hospital's system is a big step and we have to follow rules like HIPAA.

Integration Steps:

- Build an API to plug the model into the hospital's EHR system.
- Show risk scores on a dashboard where doctors can see them during patient reviews.
- Set up daily batch jobs to run predictions for all recently discharged patients.

HIPAA Compliance:

- Encrypt all data whether it is stored or being sent around.
- Limit access so only authorized staff can see the predictions.
- Strip out personal details like names during model training to keep things anonymous.

Optimization

To make sure the model doesn't overfit (where it is too tuned to the training data and flops on new patients) I would use L2 regularization. This adds a penalty for big model weights which helps the model generalize better to new data.

Part 3: Critical Thinking

Ethics and Bias

Biased data can mess things up big time in the hospital case study.

Impact of Biased Data:
If the training data has fewer records from minority groups, the model might miss readmission risks for those patients. This could mean they do not get the extra care they need leading to worse health outcomes and unfair treatment.

Mitigation Strategy:
To fix this I would use stratified sampling when preparing the data and consider reweighting underrepresented groups during training to enforce demographic parity. This means making sure the training set has a balanced mix of patients from different demographic groups so the model learns to predict well for everyone.

Trade offs

Building AI for healthcare isn't just about getting the best accuracy it is about balancing other factors too.

Interpretability vs. Accuracy:
Fancy models like deep neural networks might give slightly better predictions but they are like black boxes hard to explain. In healthcare doctors need to understand why a patient is flagged as high risk so a simpler model like logistic regression is often better even if it is a bit less accurate. Plus regulations like HIPAA favor models you can explain.

Limited Computational Resources:
If the hospital is computers aren't super powerful we might have to skip complex models like gradient boosting and stick with something lighter like logistic regression. This could mean slightly worse predictions but it is better than a model that crashes the system or takes forever to run.

Part 4: Reflection and Workflow Diagram

Reflection

This assignment made me think hard about the whole AI process and honestly the toughest part was data preprocessing. Dealing with missing values and biases while keeping the data usable felt like walking a tightrope. You do not want to throw out too much data, but you also cannot let bad data ruin your model.

If I had more time or resources, I would dig deeper into feature selection tools to pick the best predictors automatically. I would also try to get bigger more diverse datasets to make the model even stronger.