AI Project Report: Predicting Student Dropout Rates in Kenyan Universities

1. Problem Definition

Problem:

Predicting student dropout rates in Kenyan universities using academic, financial, and behavioral data.

Objectives:

1. Identify students at risk of dropping out early.
2. Improve student retention rates by informing interventions.
3. Support policy-making in higher education.

Stakeholders:

- University administration
- Ministry of Education

Key Performance Indicator (KPI):

Dropout Prediction Accuracy - Measuring the model's ability to correctly classify students who are likely to drop out, targeting accuracy above 85%.

2. Data Collection & Preprocessing

Data Sources:

1. University student records (grades, attendance, course load).
2. Financial aid and fee payment data (e.g., HELB loan status).

Potential Bias:

Socioeconomic Bias - The dataset may overrepresent students from urban areas who have consistent internet access and better educational backgrounds, potentially underrepresenting students from rural or marginalized areas.

Preprocessing Steps:

1. Handling Missing Data - Impute missing grades or income data using averages or machine learning methods.
2. Normalization - Standardize continuous features such as GPA and age for better model performance.
3. Encoding Categorical Variables - Convert non-numerical data like program or gender into numerical format using label or one-hot encoding.

3. Model Development

Model Choice:

Random Forest - A powerful ensemble model ideal for mixed data types and capable of handling nonlinear relationships. It reduces overfitting by averaging multiple decision trees.

Data Splitting:

- 70% Training Set - Used to train the model.
- 15% Validation Set - Used for hyperparameter tuning and performance validation.
- 15% Test Set - Used for final evaluation.

Hyperparameters to Tune:

1. n_estimators - Number of trees in the forest, which affects performance and overfitting.
2. max_depth - The maximum depth of each tree to control model complexity and prevent overfitting.

4. Evaluation & Deployment

Evaluation Metrics:

1. Precision - Ensures students falsely flagged as at risk are minimized.
2. Recall - Captures as many actual dropouts as possible, helping with early interventions.

Concept Drift:

Concept drift refers to changes in the data patterns over time, such as new academic policies or curriculum updates. This may reduce model accuracy if unaddressed.

Monitoring Strategy:

- Regular retraining with updated data.
- Performance tracking through dashboard metrics every academic term.

Technical Deployment Challenge:

Scalability - Deploying the model across various institutions with large datasets may strain computational resources. Solutions include cloud infrastructure and batch processing.

<u>References</u>

1. Breiman, L. (2001). Random Forests. Machine Learning.
2. University of Nairobi Student Records (Hypothetical source).
3. Kenya Higher Education Loan Board (HELB) - Financial Aid Data.
4. Gama, J. et al. (2014). A survey on concept drift adaptation.
5. Provost, F., & Fawcett, T. (2013). Data Science for Business.

CASE STUDY STEP 4

Model Deployment Plan

1. Integration into Hospital System

To deploy the Random Forest model into the hospital's system:

- **Model Serialization**:

  Save the trained model using `joblib` or `pickle` for later use.

  ```python
  import joblib
  joblib.dump(model, 'readmission_model.pkl')
  ```

- Develop a REST API:

Use a lightweight framework like Flask or FastAPI to expose the model via an endpoint.

```python
from flask import Flask, request, jsonify
import joblib
app = Flask(__name__)
model = joblib.load('readmission_model.pkl')
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    prediction = model.predict([data['features']])
    return jsonify({'prediction': prediction.tolist()})
```

- Integrate with Hospital Management System (HMS):

Collaborate with the IT department to connect the API to the EHR/HMS so doctors and staff can access predictions directly.

- Automate Preprocessing and Data Input:

Build preprocessing pipelines that clean and format data before feeding it to the model in real-time.

2. Ensuring Compliance with Healthcare Regulations (HIPAA or local laws)

- To maintain ethical and legal standards:

- Data Anonymization:

Strip out or mask sensitive patient identifiers before processing data.

- Access Controls:

Limit access to the model/API to authorized personnel only.

- Audit Logging:

Log every access to the prediction system to ensure traceability.

- Secure Infrastructure:

Deploy the model over HTTPS with strong authentication (e.g., API keys, OAuth2).

- Compliant Storage:

Store data on secure, encrypted, and regulation-compliant cloud or local servers.

- Bias and Fairness Testing:

Regularly audit the model for bias across demographic groups.

- Transparent Use:

Inform patients if AI tools are part of their care process.

CASE STUDY PART 5

Method: Cross-Validation

Description:
Instead of training and testing your model on a single train/test split, you divide your training data into multiple folds (typically 5 or 10). The model is trained and validated multiple times—each time on a different fold. The performance is then averaged.

Why it helps:

- Ensures the model generalizes better to unseen data.
- Reduces the risk of fitting too closely to a specific train/test split.

- Gives a more robust estimate of model performance.

Part 3: Critical Thinking

Ethics & Bias

Q1: How might biased training data affect patient outcomes in the case study?

Biased training data may reflect historical inequalities or systemic discrimination in healthcare. If certain groups (e.g., based on age, race, gender, or socioeconomic status) are underrepresented or previously received inconsistent care, the model may unfairly predict higher or lower readmission risks for them. This could lead to unequal treatment plans, misallocation of hospital resources, and worsen health disparities among vulnerable populations.

Q2: Suggest 1 strategy to mitigate this bias.

One effective strategy is **rebalancing the dataset** using techniques such as oversampling underrepresented groups, undersampling dominant classes, or using **fairness-aware algorithms**. Additionally, continuous monitoring of model performance across demographic groups ensures fairness in deployment.

Trade-offs (10 points)

Q3: Discuss the trade-off between model interpretability and accuracy in healthcare.

In healthcare, **interpretability is often prioritized** because doctors and stakeholders need to understand the reasoning behind predictions to make informed decisions. However, highly interpretable models (e.g., logistic regression) might have lower predictive power compared to complex models (e.g., random forests, neural networks), which are often more accurate but harder to explain. Striking a balance between accuracy and explainability is key, especially in life-impacting scenarios.

Q4: If the hospital has limited computational resources, how might this impact model choice?

Hospitals with limited computing power may prefer **lightweight models** like decision trees or logistic regression. These models train faster, require less memory, and can be deployed on basic infrastructure. More complex models like ensemble methods or deep learning may be

avoided due to their high processing demands, unless optimized deployment tools or cloud services are available.

PART 4 :Reflection (5 points)

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

The most challenging part of the workflow was data preprocessing, particularly handling missing values and inconsistent data entries (e.g., '?') across a large and complex dataset. This required careful inspection, decisions on imputation or exclusion, and memory optimization when transforming data—especially during feature scaling. Ensuring the data was clean, meaningful, and usable without compromising its integrity took significant time and consideration.

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

With more time and resources, I would:

- Implement automated data validation and cleaning pipelines using tools like Great Expectations.
- Explore distributed computing tools (e.g., Dask or Spark) to handle memory-intensive operations.
- Perform hyperparameter tuning using GridSearchCV or Optuna to further optimize model performance.
- Incorporate more interpretable models or SHAP/LIME for explainability.
- Deploy the model in a test environment using cloud services (e.g., AWS SageMaker or Azure ML).
