**Technical Documentation for Predicting Student Dropout**

In this documentation, I’ll Walk you through the entire pipeline I used to predict student dropout. From data preparation all the way to deploying the model using Streamlit, I'll share the steps I took and the considerations I made to ensure a robust solution for this prediction task.

**1. Data Preprocessing**

To kick things off, I needed to clean and prepare the dataset so that it was ready for model training. This involved several tasks:

- Handling Missing Values: I filled in missing values where necessary, using strategies like mean or median imputation, or dropped rows/columns when it made sense to do so.

- Data Type Conversion: I converted categorical features to numerical values using encoding techniques such as label encoding.

- Exploratory Data Analysis: I did Univariate, Bivariate and Multivariate Analysis

- Class Imbalance Note: I noticed that the target classes were imbalanced, which would need to be dealt with later.

**2. Data Validation**

Before moving forward, I made sure to validate the data's quality:

- Schema Checks: I checked if all data conformed to the expected structure, with the correct data types.

- Range Checks: I made sure numerical values were within reasonable ranges (e.g., age at enrollment, grades).

- Unique Values: I reviewed categorical columns for any unexpected or erroneous values.

**3. Ensuring Proper Encoding**

I verified that all categorical columns were properly converted to numerical values:

- Encoding Review: I double-checked that all categorical features had been encoded, whether by one-hot encoding or label encoding.

- Error Checks: I ensured that no mistakes occurred during the encoding process (e.g., unintended label values).

**4. Feature Importance Analysis**

To get a sense of which features were most influential in predicting dropout, I conducted a feature importance analysis:

- Using Tree-based Models: I employed tree-based models (like XGBoost and Random Forest) to rank features based on their importance.

- Correlation Analysis: I also examined correlations to see how different features related to the target variable.

**5. Selecting the Most Important Features**

I decided to narrow down the number of features for model training:

- Top 10 Features Selection: I picked the top 10 most important features based on the previous analysis, ensuring the selected features were meaningful and added value.

- Target Variable Included: I made sure the target variable, 'Dropout,' was part of the selection.

**6. Defining Features (X) and Target (y)**

With the most important features selected, I proceeded to separate them from the target variable:

- Feature Matrix (X): I extracted the selected features for training.

- Target Vector (y): I used the 'Dropout' column as the target variable for prediction.

**7. Splitting the Dataset**

To avoid overfitting and ensure the model’s generalizability, I split the data:

- Training Set (60-70%): This was used to train the model.

- Validation Set (15-20%): I used this set for hyperparameter tuning and model selection.

- Test Set (15-20%): The final evaluation was done here to assess how the model would perform on unseen data.

**8. Addressing Class Imbalance with SMOTE**

Since the dataset was imbalanced, I applied SMOTE to ensure the model wouldn’t be biased towards the majority class:

- Synthetic Over-sampling: I used SMOTE to create synthetic samples for the minority class, balancing the dataset.

- Verifying Balance: I confirmed that the distribution of classes was even after applying SMOTE.

**9. Scaling the Features**

To standardize the feature ranges, I scaled the data using the Min-Max Scaler:

- Min-Max Scaling: This transformation brought all features into a range between 0 and 1, making them more comparable.

- Data Leakage Prevention: I applied the same scaling to both training and test sets to avoid any leakage.

**10. Training the Model and Evaluating Performance**

Now, it was time to train the model and see how well it performed:

- Model Training: I trained an XGBoost model using the processed training data.

- Evaluation Metrics: I evaluated the model using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

- Confusion Matrix Analysis: I used the confusion matrix to understand where the model was getting it right and where it was making mistakes (true positives, false positives, etc.).

**11. Hyperparameter Tuning**

To optimize the model, I performed hyperparameter tuning:

- GridSearchCV: I used GridSearchCV to find the best combination of parameters that yielded the highest performance on the validation set.

- Reviewing Results: I compared the performance with different parameter settings to ensure I was choosing the best configuration.

**12. Saving the Best Model**

Once I had the best-performing model, I saved it for future use:

- Pickle Serialization: I saved the model using Pickle to ensure it could be easily loaded and used later.

- Version Control: I kept track of the model version so I could revert to previous versions if needed.

**13. Deployment Using Streamlit**

Finally, I deployed the model to a web app for user-friendly access:

- Building the Streamlit App: I created a Streamlit application that allows users to interact with the model and make predictions.

- Model Integration: I loaded the saved model into the app and added functionality for real-time predictions.

- User Testing: I tested the app extensively with different inputs to make sure it was working correctly and delivering accurate predictions.

**Additional Notes**

- Logs and Documentation: I documented each step and maintained logs for the training, tuning, and deployment phases to make future updates easier.

This process provided a comprehensive approach to building, evaluating, and deploying a machine learning model for predicting student dropout, with careful attention given to data quality, model performance, and usability.

**User Guide for the Streamlit App: Predicting Student Dropout**

This guide will help you navigate and use the Streamlit app for predicting student dropout. The app allows users to input relevant student information and receive a prediction on whether a student is likely to drop out or not. Below are the steps for using the app effectively.

**1. Launching the App**

- Access the App: Open the Streamlit app using the provided URL or by running the app locally on your machine. If you are running it locally, use the command `streamlit run app.py` in your terminal.

- Homepage Overview: Once the app is open, you’ll see a clean interface with a title, description of the app, and input fields for entering student data.

**2. Inputting Student Data**

The app requires specific information about the student to make an accurate prediction. Each field is important and should be filled in as accurately as possible:

**- Personal Details:**

- Age at Enrollment: Enter the student's age when they enrolled in the program.

**- Academic Information:**

- Admission Grade: Provide the grade the student had upon admission.

**- Curricular Units (1st Semester):** Fill in the following:

- Number of curricular units credited.

- Number of curricular units enrolled.

- Number of curricular units evaluated.

- Number of curricular units passed.

- Average grade for the first semester.

**- Curricular Units (2nd Semester):** Similarly, fill in:

- Number of curricular units credited.

- Number of curricular units enrolled.

- Number of curricular units evaluated.

- Number of curricular units passed.

- Average grade for the second semester.

**- Additional Factors:**

- Parent’s Qualification: Select the level of education for both the mother and father.

- Student's Nationality: Choose the student's nationality from the dropdown list.

- Tuition Fee Status: Indicate whether the student's tuition fees are up-to-date.

**3. Making a Prediction**

- Submit the Form: After filling in all the required fields, click the "Predict" button.

- View the Prediction: The app will process the information and provide an output:

- Prediction Outcome: It will display whether the student is "Likely to Dropout" or "Likely No Dropout."

- Prediction Probability (Optional): If enabled, it may also show the probability score, giving and indication of the model's confidence in the prediction.

**4. Interpreting the Results**

- Likely to Dropout: If the prediction indicates that the student is likely to drop out, it may be worth investigating further or providing additional support to the student.

- Likely to Not Dropout: If the prediction is that the student is likely to continue, this suggests the student is on a stable path.

**5. Updating the App**

For users who need to update the app:

- Model Updates: If a newer model is available, replace the existing model file (saved as a `. pkl` file) with the updated version.

- Adjust Input Fields: If you need to add more features or modify existing ones, update the code in the `app.py` file accordingly.

- Restart the App: Make sure to restart the app after making any changes by using the command `streamlit run app.py`.

**6. Troubleshooting**

- App Not Loading: Make sure all required libraries (e.g., `streamlit`, `pandas`, `numpy`) are installed. Use `pip install -r requirements.txt` to install dependencies.

- Incorrect Predictions: If you suspect the predictions are off, ensure the model is up-to-date and check if the data preprocessing steps were correctly followed.

- Error Messages: If you encounter any error messages, follow the stack trace for debugging or refer to the Streamlit documentation for troubleshooting tips.

**Model Card for Student Dropout Prediction Model**

This model card presents the details of the student dropout prediction model, including its performance metrics, limitations, and ethical considerations. The model aims to predict whether a student is likely to drop out based on a variety of academic and personal factors.

**1. Model Overview**

The student dropout prediction model is built using the XGBoost algorithm, a gradient boosting technique known for its ability to handle complex tabular data. The model uses features related to students' demographics, academic records, and socioeconomic factors to predict the likelihood of dropping out (binary classification: dropout = 1, not dropout = 0).

**2. Model Performance**

The model's performance was assessed using a test dataset, with the following metrics:

- Model Used: Loaded XGBoost Model

- Accuracy: 0.875 (87.46%)

- Precision:0.853 (85.31%)

- Recall: 0.736 (73.59%)

- F1 Score: 0.790 (79.02%)

- Confusion Matrix:

- True Negatives (correctly predicted non-dropouts): 565

- False Positives (incorrectly predicted as dropouts): 36

- False Negatives (missed dropouts): 75

- True Positives (correctly predicted dropouts): 209

These metrics indicate that the model performs well in predicting student dropout, with a high accuracy and good precision. The recall value shows reasonable sensitivity in identifying students likely to drop out, while the F1 score balances precision and recall.

**3. Limitations**

Despite good overall performance, the model has certain limitations:

- Data Quality and Representativeness: The model's reliability depends on the quality and diversity of the training data. If the data is biased or unrepresentative of certain groups, the model's predictions may be skewed. For example, if there is a limited number of records for students from certain socioeconomic backgrounds, the model might not generalize well for those groups.

- Context Dependency: Since the model was trained on a specific dataset, it may not generalize effectively to other educational contexts or institutions with different curricula, grading systems, or demographics without retraining or fine-tuning.

- False Negative Predictions: The model's recall rate indicates that some students at risk of dropping out may not be correctly identified (75 false negatives in the confusion matrix). This could result in some at-risk students not receiving the necessary interventions.

- Complexity of XGBoost: Although XGBoost is powerful, it may lack interpretability for non-experts compared to simpler models. Understanding how specific features influence the predictions may require additional tools like SHAP or LIME for interpretability.

**4. Ethical Considerations**

Ethical considerations are essential when using the model to make decisions about students:

Fairness and Bias: The model’s predictions might carry over some biases from the data it was trained on. It's important to make sure that decisions based on the model's predictions are fair and don’t negatively affect any group. Regular checks should be done to see if the model’s accuracy is different for various groups of students.

- Data Privacy: The dataset used for training contains sensitive information about students. Ensuring compliance with data privacy regulations, such as anonymizing data and obtaining consent for data use, is vital.

- Transparency and Accountability: Users should be aware of the model's limitations and understand that its predictions are a tool to assist decision-making rather than an absolute verdict. Transparent reporting of how predictions are made and involving stakeholders in the evaluation of outcomes can improve the model's responsible use.

- Supporting, Not Replacing Human Judgment: The model should not replace human judgment but complement it. Predictions should be used alongside other indicators and expert input to determine the best course of action for supporting students.

**Recommendations for Users**

- Use Predictions as a Support Tool: The model's output should help identify students who may need additional support, but the final decisions should also consider human judgment and qualitative insights.

- Monitor Performance Regularly: Periodically retrain the model with updated data to maintain accuracy and fairness.

- Conduct Bias Checks: Routinely assess whether the model's performance varies across different groups and take corrective actions if disparities are detected.

The model provides a useful tool for predicting student dropout, but its limitations and ethical implications should be considered when using it for decision-making.