Student Dropout Prediction ML App – Documentation

Overview

The Student Dropout Prediction ML App is a machine learning-based web application developed using Streamlit. Its main function is to predict the likelihood of a student dropping out of their academic program based on various input factors, including academic performance, tuition fee status, and age at enrollment. The app uses a pre-trained Random Forest model to provide real-time predictions, making it a valuable tool for educational institutions to assess students' risks and take proactive measures.

Key Features

- User-Friendly Interface: An easy-to-use interface for inputting key student-related information.
- Real-Time Prediction: The app instantly predicts if a student is at risk of dropping out based on the user inputs.
- Optimized Performance: Cached model loading and prediction functions for improved speed and efficiency.
- Scalable: Can be deployed on local systems or in cloud environments such as Streamlit Cloud, Heroku, or AWS.

Application Workflow

First, the User's Data are collected when the User provide input data such as grades, number of approved curricular units, tuition status, and other factors. Next is the Model Prediction, After receiving the input, the app processes the data and runs it through the pre-trained Random Forest model. Lastly, the Output, The app then outputs either "Dropout" or "Not Dropout" as the predicted outcome, indicating whether the student is at risk of dropping out.

Input Variables

The following input fields are used to collect data for making predictions:

- Curricular Units 2nd Semester (Approved): Number of courses passed in the second semester.
- Curricular Units 2nd Semester (Grade): Average grade in the second semester.
- Curricular Units 1st Semester (Approved): Number of courses passed in the first semester.
- Curricular Units 1st Semester (Grade): Average grade in the first semester.
- Tuition Fees Up to Date: Whether the student's tuition fees are up to date (Options: "Up to date", "Not up to date").
- Age at Enrollment: The age group of the student when they first enrolled in the program (Options: "18-24", "25-34", "35-44", "45+").
- Admission Grade: The grade obtained at the time of admission.
- Previous Qualification Grade: Grade obtained in the student's previous qualification.
- Curricular Units 2nd Semester (Evaluations): Number of evaluations (exams, tests) taken in the second semester.

Application Components

1. **Frontend – User Interface (UI)**

The user interface is built using Streamlit and consists of several input fields where users can provide relevant data. Key UI components:
- Number Input Fields: For collecting numerical inputs like grades and the number of approved curricular units.
- Selectbox Fields: For selecting categorical values such as tuition fee status and age group.
- Predict Button: A button that triggers the prediction when clicked.
- Result Display: The prediction result ("Dropout" or "Not Dropout") is displayed once the "Predict" button is clicked.

2. **Backend – Prediction Logic**

The backend of the app consists of:

- Model Loading: The Random Forest Classifier model is loaded from a pickle file and cached to avoid reloading it multiple times.
- Data Preprocessing: User inputs are converted into a numerical format using predefined mappings for categorical variables such as age and tuition fees.
- Prediction: The processed input data is passed into the classifier for prediction. The app returns either "Dropout" or "Not Dropout" based on the input.

Code Breakdown
1. **Model Loading.**
The model is stored in a `.pkl` file, which is loaded using the `pickle` library. This function is cached using `@st.cache_resource` to avoid loading the model multiple times during app usage, improving performance.
```python
@st.cache_resource()
def load_model():
    model_path = r"C:\Users\hp\Documents\3Signet Internship\Task 6\best_rf_model.pkl"
    with open(model_path, 'rb') as file:
        classifier = pickle.load(file)
    return classifier
classifier = load_model()
```
2. **Input Mappings**.
Categorical variables such as **Age at Enrollment** and **Tuition Fees Status** are mapped to numerical values that the model can interpret.
```python
age_enrollment_mapping = {
    "18-24": 0,
    "25-34": 1,
    "35-44": 2,
    "45+": 3
}
tuition_fees_mapping = {
    "Up to date": 0,
    "Not up to date": 1
}
```
3. **Prediction Function**
This function takes in user inputs, converts categorical values into numeric ones using mappings, and returns a prediction (either "Dropout" or "Not Dropout").
```python
@st.cache_data()
def prediction(curricular_units_2nd_approved, curricular_units_2nd_grade, curricular_units_1st_approved,
               curricular_units_1st_grade, tuition_fees_status, age_enrollment, admission_grade,
               previous_qualification_grade, curricular_units_2nd_evaluations):

    # Map 'Age at enrollment' and 'Tuition fees up to date' to numerical values
    age_enrollment_numeric = age_enrollment_mapping[age_enrollment]
    tuition_fees_numeric = tuition_fees_mapping[tuition_fees_status]

    # Making Predictions
    prediction = classifier.predict([[curricular_units_2nd_approved, curricular_units_2nd_grade,
                                      curricular_units_1st_approved, curricular_units_1st_grade,
                          tuition_fees_numeric, age_enrollment_numeric, admission_grade,
                          previous_qualification_grade, curricular_units_2nd_evaluations]])

    if prediction == 0:
        return "Not Dropout"
    else:
        return "Dropout"
```
4. **Streamlit Interface**
The user interface is developed using Streamlit, which allows users to input values and get predictions in real-time.
```python
def main():
    # Frontend elements of the web page
    html_temp = '''
    <div style='background-color: blue; padding:13px'>
    <h1 style='color: black; text-align: center;'>Student Dropout Prediction ML App</h1>
    </div>
    '''
    st.markdown(html_temp, unsafe_allow_html=True)

    # Input fields for user data
    curricular_units_2nd_approved = st.number_input("Curricular units 2nd sem (approved)")
    curricular_units_2nd_grade = st.number_input("Curricular units 2nd sem (grade)")
    curricular_units_1st_approved = st.number_input("Curricular units 1st sem (approved)")
    curricular_units_1st_grade = st.number_input("Curricular units 1st sem (grade)")
    tuition_fees_status = st.selectbox('Tuition fees up to date', tuple(tuition_fees_mapping.keys()))
    age_enrollment = st.selectbox('Age at enrollment', tuple(age_enrollment_mapping.keys()))
    admission_grade = st.number_input("Admission grade")
    previous_qualification_grade = st.number_input("Previous qualification (grade)")
    curricular_units_2nd_evaluations = st.number_input('Curricular units 2nd sem (evaluations)')

    # When 'Predict' is clicked, make prediction and display the result
    if st.button("Predict"):
        result = prediction(curricular_units_2nd_approved, curricular_units_2nd_grade,
                            curricular_units_1st_approved, curricular_units_1st_grade,
                            tuition_fees_status, age_enrollment, admission_grade,
                            previous_qualification_grade, curricular_units_2nd_evaluations)
        st.success(f"Prediction: {result}")
```
Performance Optimization
- Caching: The `@st.cache_resource()` and `@st.cache_data()` decorators are used to cache the model and prediction functions, preventing repeated loading or recalculation, thus improving the app's speed.
- Efficient Input Handling: Inputs are processed efficiently to ensure minimal latency when predictions are made.

System Requirements
- Python 3.x
- Streamlit library
- Scikit-learn for the machine learning model
- Pickle for loading the pre-trained model

How to Run the App Locally

1. Install the necessary dependencies:
2. Run the app using Streamlit:
3. The app will be hosted on your local machine, and you can interact with it through your web browser.

Deployment Options
- Local Deployment: The app can be run locally on any machine with Python and the necessary libraries installed.

Conclusion

The Student Dropout Prediction ML App offers an easy-to-use interface for predicting student dropout risk based on various academic and personal factors. It is a helpful tool for educational institutions to identify at-risk studentsoptimized for performance and can be deployed quickly for educational institutions or other use cases.