SAVITRIBAI PHULE PUNE UNIVERSITY

Department of Technology

PROJECT REPORT

ON

"COURSE RECOMMENDATION SYSTEM"

BY

Mr. Pravin Nanaware

Submitted in Partial fulfillment of

Post Graduation Diploma in Data Science and AI

Savitribai Phule Pune University
For the Academic Year
2023-2024

UNDER THE GUIDANCE OF
Miss. Priyanka Ghadge

Department of Technology
Savitribai Phule Pune University,
Ganeshkhind, Pune-411007

SAVITRIBAI PHULE PUNE UNIVERSITY

Department of Technology

PROJECT REPORT

ON

"COURSE RECOMMENDATION SYSTEM"

BY

Mr. Pravin Nanaware

Submitted in Partial fulfillment of

Post Graduation Diploma in Data Science and AI

Savitribai Phule Pune University
For the Academic Year
2023-2024

UNDER THE GUIDANCE OF
Miss. Priyanka Ghadge

Department of Technology
Savitribai Phule Pune University,
Ganeshkhind, Pune-411007

CERTIFICATE

SAVITRIBAI PHULE PUNE UNIVERSITY

DEPARTMENT OF TECHNOLOGY

This is to certify that **Pravin Nanaware** has successfully completed her project on **"COURSE RECOMMENDATION SYSTEM"**

In partial fulfillment of 2nd Semester work for their Post Graduation Diploma in Data Science and AI under Savitribai Phule Pune University, for the academic year 2023-2024

Miss. Priyanka Ghadge (Project Guide)	Dr. Manisha Bharati (Course Coordinator)	Dr. Aditya Abhyankar (H.O.D)

Signed by
(External Examiner)

Place: Pune
Date: / /

Students Declaration

I undersigned a student of the Department of Technology, Savitribai Phule Pune University, Pune PGD Data Science and AI - 3rd semester, declare that the summer internship project **"COURSE RECOMMENDATION SYSTEM"** is a result of my own work and my indebtedness to other work publications, references, if any, have been duly acknowledged.

If I am found guilty of copying any other report or published information and showing it as my original work, I understand that I shall be liable and punishable by the Institute or University, which may include Fail in the examination, repeat study and resubmission of the report or any other punishment that Institute or University may decide.

Pravin Nanaware
PGD23DS58

Signature

ACKNOWLEDGEMENT

We are deeply grateful to all those who have played a pivotal role in shaping our internship project and enriching our learning experience. We extend my heartfelt gratitude to:

Miss. Priyanka Ghadge

(Professor and Project Guide from the Department of Technology, SPPU) For their scholarly guidance, insightful suggestions, and continuous encouragement throughout the course of my project.

The entire faculty of the Department of Technology, Savitribai Phule Pune University, for fostering an environment of academic excellence and nurturing my curiosity.

We are profoundly thankful to everyone mentioned above for their selfless contributions to my journey. Their mentorship and support have been instrumental in shaping my skills and fostering personal growth. Thank You

"THANKS AGAIN TO ALL WHO HELPED US"

Mr. Pravin Nanaware

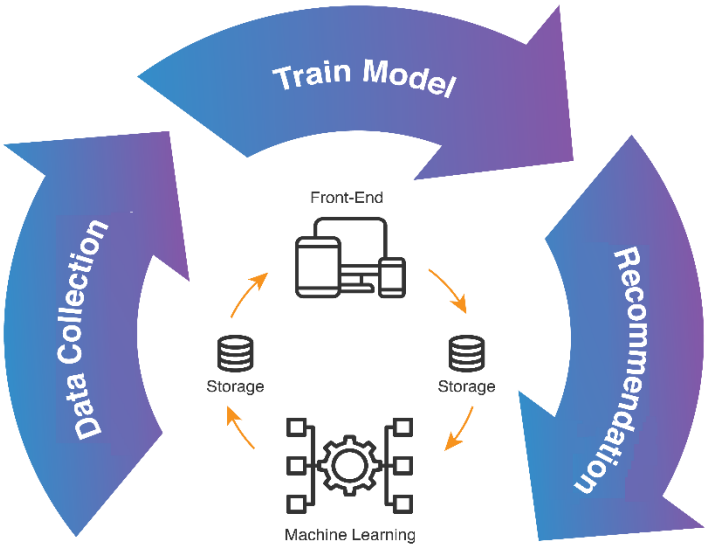
INDEX

Chapter 1: INTRODUCTION TO THE PROJECT

This project, titled "Course Recommendation System Using Machine Learning," aims to help students choose the best courses based on their unique profiles. Using machine learning, the system analyzes factors like graduation status, interests, prerequisites, and entrance test scores to recommend suitable courses.

The goal is to provide personalized and accurate advice to students, making it easier for them to make informed decisions about their education. By automating the recommendation process, the project also helps educational institutions improve their advising services.

This system represents a modern approach to educational guidance, combining technology and data to benefit both students and schools.



Significance of the Study

The study behind the "Course Recommendation System Using Machine Learning" is important for several reasons:

1. **Personalized Guidance**:
   The system provides personalized course recommendations tailored to each student's unique profile, helping them choose courses that best match their interests and strengths.

2. **Better Student Outcomes**: By recommending courses that fit well with students' abilities and interests, the system can improve their academic performance and overall satisfaction.

3. **Efficient Advising**: Manual advising can be slow and inconsistent. This automated system can quickly provide reliable recommendations, saving time for both students and advisors.

4. **Data-Driven Decisions**: Using data to guide course selection ensures that recommendations are based on solid evidence, making them more accurate and trustworthy.

5. **Support for Undecided Students**: Students who are unsure about which courses to take can benefit from objective, data-backed recommendations that help them make more confident decisions.

6. **Resource Optimization**: Educational institutions can use insights from the system to understand student preferences and trends, helping them to optimize course offerings and resource allocation.

7. **Enhanced Student Engagement**: When students take courses that interest them and match their skills, they are more likely to be engaged and motivated, leading to a better learning experience.

In summary, this study highlights the potential of using machine learning to transform educational advising. By providing personalized, efficient, and data-driven recommendations, the system can significantly improve the course selection process for students and institutions alike.

Chapter 2: THEORETICAL FRAMEWORK

The theoretical framework for the "Course Recommendation System Using Machine Learning" project is based on several key concepts and technologies:

1. **Machine Learning**:
   At the core of the system is the use of machine learning algorithms to analyze and predict the best course recommendations. Machine learning enables the system to learn from historical data and make accurate predictions based on input features.

2. **Data Preprocessing**:
   This involves cleaning and transforming raw data into a suitable format for machine learning. Techniques such as handling missing values, encoding categorical variables, and scaling numerical features are essential steps to prepare the data.

3. **Feature Engineering**:
   Identifying and creating relevant features from the raw data is crucial. For this project, features include graduation status, interests, prerequisites, and entrance test scores.

4. **Classification Algorithms**:
   The project employs the XGBoost classifier, a powerful and efficient machine learning algorithm known for its accuracy and performance in classification tasks.

5. **Evaluation Metrics**:
   Accuracy is used as the primary metric to evaluate the performance of the recommendation system. Other metrics such as precision, recall, and F1-score can also be considered for a comprehensive evaluation.

REVIEW OF LITERATURE

1. **Educational Data Mining (EDM)**: The field of EDM focuses on developing methods to explore data from educational settings. Research in this area has shown that machine learning can be effectively used to predict student performance and provide personalized recommendations (Romero & Ventura, 2010).

2. **Recommender Systems**: Traditional recommender systems, widely used in e-commerce and content streaming services, have been adapted for educational purposes. These systems use collaborative filtering, content-based filtering, and hybrid methods to recommend items to users (Ricci et al., 2011).

3. **Application of Machine Learning in Education**: Various studies have demonstrated the application of machine learning to predict academic outcomes, such as course grades, dropout rates, and student success (Baker & Inventado, 2014). These studies provide a foundation for using similar techniques in course recommendation systems.

4. **XGBoost Algorithm**: XGBoost is a popular machine learning algorithm known for its high performance and efficiency. It has been successfully applied in numerous classification and regression problems across different domains (Chen & Guestrin, 2016).

5. **Personalized Learning**: Research has emphasized the importance of personalized learning experiences in improving student engagement and outcomes (Pane et al., 2015). Personalized recommendations can cater to individual student needs, leading to better academic results.

6. **Challenges in Educational Recommender Systems**: Studies have identified challenges such as data sparsity, cold start problems, and the need for interpretability in educational recommender systems (Drachsler et al., 2015). Addressing these challenges is crucial for developing effective systems.

7. **Impact of Entrance Scores and Prerequisites**: Research has shown that entrance test scores and fulfilled prerequisites are strong indicators of student readiness and success in specific courses (Bettinger et al., 2013). Including these features in the recommendation model can enhance its predictive accuracy.

By integrating these theoretical foundations and insights from existing literature, the project aims to develop a robust and effective course recommendation system that leverages machine learning to provide personalized educational guidance.

Chapter 3: OBJECTIVES AND SCOPE

Objectives

Develop a Course Recommendation System: Create an intelligent system that recommends suitable courses for students based on their academic profiles and interests.

1.Improve Academic Decision:

Making: Help students make informed decisions about their course selections by providing personalized recommendations.

2.Utilize Machine Learning Algorithms:

Implement advanced machine learning techniques, specifically the XGBoost classifier, to analyze student data and predict optimal courses.

3.Enhance Data Preprocessing:

Employ effective data preprocessing methods to handle missing values, encode categorical data, and prepare the dataset for machine learning.

4.Achieve High Prediction Accuracy:

Aim for a high level of accuracy in course recommendations to ensure reliable and useful guidance for students.

Automate the Advising Process:

Reduce the need for manual advising by providing a scalable, automated solution that can handle large volumes of student data efficiently.

Support Educational Institutions:

Assist educational institutions in optimizing their course offerings and advising services by providing insights into student preferences and trends.

Scope of Project

1)Data Collection and Preparation:

1)Use student data such as graduation status, interests, prerequisites, and entrance test scores.

2)Perform data cleaning to remove unnecessary columns and handle missing values.

3)Filter the data to include only relevant entries, such as those with entrance test scores above a certain threshold.

2)Feature Engineering and Encoding:

1)Identify key features that influence course recommendations.

2)Apply label encoding to the target variable and one-hot encoding to categorical features.

3)Machine Learning Model Development:

1)Split the dataset into training and testing sets for model evaluation.

2)Implement the XGBoost classifier within a pipeline that includes preprocessing steps.

3)Train the model on the training data and evaluate its performance on the test data.

4)Model Evaluation and Optimization:

1)Assess the model's accuracy and other relevant metrics.

2)Optimize the model and preprocessing steps to enhance prediction performance.

5)System Deployment and Usage:

1)Save the trained model and preprocessing pipeline for future use.

2)Provide a framework for integrating the recommendation system into educational advising platforms.

6)Benefits to Students and Institutions:

1)Offer personalized course recommendations to students, helping them choose courses that align with their strengths and interests.

2)Provide educational institutions with data-driven insights to improve their course offerings and advising services.

7)Future Enhancements:

1)Explore additional features and data sources to improve the model.

2)Implement more advanced machine learning techniques and hyperparameter tuning for better performance.

3)Develop user-friendly interfaces for easy access to recommendations by students and advisors.

By achieving these objectives within the defined scope, the project aims to significantly enhance the course selection process for students and support educational institutions in delivering personalized and effective academic guidance.

Chapter 4: Research Methodology

1.Data Collection

The process of gathering relevant student data, including graduation status, interests, prerequisites, entrance test scores, and course recommendations, from educational databases or records.

2. Data Preprocessing

The steps taken to clean and prepare the raw data for analysis. This includes removing unnecessary columns, handling missing values, filtering data, and selecting relevant features.

3. Data Splitting

Dividing the dataset into training and testing sets to evaluate the model's performance. Typically, a common split ratio is 80% for training and 20% for testing.

4. Data Transformation

Applying transformations to the data, such as encoding categorical variables and scaling numerical features, to make it suitable for machine learning algorithms.

5. Model Development

The process of selecting and configuring a machine learning model. For this project, the XGBoost classifier is chosen for its accuracy and efficiency in classification tasks.

6. Model Training

Fitting the machine learning model to the training data so that it can learn patterns and relationships in the data.

7. Model Evaluation

Assessing the performance of the trained model on the test data using metrics such as accuracy, precision, recall, and F1-score.

8. Model Deployment

Saving the trained model and the preprocessing pipeline for future use, ensuring that the system can be easily integrated into practical applications.

9. Conclusion and Future Work

Analyzing the results of the model, discussing its implications, and identifying areas for improvement and future enhancements to further refine the recommendation system.

Chapter 5: Data Analysis

5.1 Data Preprocessing

Data preprocessing is a crucial step in the data analysis process. It involves transforming raw data into a clean and usable format, suitable for machine learning models. The following sections detail the steps taken to preprocess the dataset for the "Course Recommendation System Using Machine Learning" project.

Step 1: Load the Dataset

```
#To load data set
import pandas as pd
df = pd.read_csv('Dataset_RS.csv')
```

Step 2: Remove Unnecessary Columns

Removing unwanted columns from a DataFrame

```
# Drop unnecessary columns
df = df.drop(['Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9', 'Student Name'], axis=1)
```

Step 3: Handle Missing Values

Rows with missing values in the Graduation column were removed to ensure the quality of the data.

```
# Drop rows with null values in 'Graduation'
df = df.dropna(subset=['Graduation'])
```

Step 4: Filter the Data

Rows where the entrance test score is less than 20 were excluded. This step ensures that only students with a sufficient level of preparedness are considered in the analysis.

```
# Drop rows where 'entrance test score' is less than 20
df = df[df['entrance test score'] >= 20]
```

Step 5: Select Relevant Features

The features selected for the analysis include Graduation, Interest, prerequisites, and entrance test score. These features were chosen based on their relevance to predicting course recommendations**.**

```
# Separate features and target variable
X = df[['Graduation', 'Interest', 'prerequisites', 'entrance test score']]
y = df['Recommend Course']
```

Step 6: Encode the Target Variable

The target variable Recommend Course was encoded using LabelEncoder to convert the categorical values into numerical format.

```
# Apply label encoding to the target variable 'Recommend Course'
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Step 7: Split the Data

the dataset was split into training and testing sets, with 80% of the data used for training and 20% for testing. This split helps in evaluating the model's performance on unseen data.

```
from sklearn.model_selection import train_test_split
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Step 8: Combine Data for Consistent Encoding

To ensure consistent encoding of categorical variables, the training and testing sets were combined during the preprocessing step.

```
# Combine X_train and X_test for consistent encoding
X_combined = pd.concat([X_train, X_test])
```

Step 9: Apply OneHotEncoding

One-hot encoding was applied to categorical features (**Graduation**, **Interest**, **prerequisites**) using **ColumnTransformer** and **OneHotEncoder**. This step converts categorical variables into a format that can be provided to machine learning algorithms.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Apply encoding using OneHotEncoder
preprocessor = ColumnTransformer(
    transformers=[
        ('graduation_interest_prerequisites', OneHotEncoder(handle_unknown='ignore'), ['Graduation', 'Interest', 'prerequisites']),
    ],
    remainder='passthrough'
)

# Fit the preprocessing on the combined dataset
preprocessor.fit(X_combined)

# Transform the training and testing data
X_train_transformed = preprocessor.transform(X_train)
X_test_transformed = preprocessor.transform(X_test)
```

Model Training

Model training involves selecting an appropriate machine learning algorithm, fitting the model to the training data, and optimizing it to learn the patterns within the data.

Step 1: Define the Model

For this project, the XGBoost classifier is chosen due to its efficiency and high performance in classification tasks.

```
import xgboost as xgb
# Define the XGBoost model
model = xgb.XGBClassifier()
```

Step 2: Create a Pipeline

A pipeline is constructed to streamline the process, combining preprocessing steps and the model into a single workflow.

```
from sklearn.pipeline import Pipeline
# Create a pipeline with preprocessing and the model
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', model)
])
```

Step 3: Train the Model

The pipeline is then fitted to the training data. This step involves training the XGBoost classifier using the preprocessed training data.

```
# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)
```

Model Evaluation

Model evaluation involves assessing the performance of the trained model on the test data using various metrics.

Step 1: Make Predictions

The trained model is used to predict the target variable on the test set.

```
# Predict the target variable on the test set
predictions = pipeline.predict(X_test)
```

Step 2: Calculate Accuracy

The accuracy score, which represents the proportion of correct predictions out of the total predictions, is calculated.

```
from sklearn.metrics import accuracy_score
# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Model Accuracy: {accuracy}")
```

Step 3: Additional Evaluation Metrics

Other metrics such as precision, recall, and F1-score can also be calculated to provide a more comprehensive evaluation of the model's performance.

Chapter 6: Model Deployment

Introduction to deployment

the deployment of a machine learning model as a web application using Flask, a lightweight web framework for Python. This application allows users to input certain features, processes the data using a pre-trained model pipeline, and returns a prediction for the recommended course.

Steps for Deployment

. Importing Necessary Libraries

. Loading Pre-trained Model and Encoder

. Setting Up Flask Application

. Creating Routes for the Web Application

. Handling User Input and Making Predictions

. Running the Flask Application

In this project **Necessary Libraries are**

```
from flask import Flask, render_template, request
import joblib
import pandas as pd
```

Explanation:

Flask:- A web framework for creating web applications.
render_template:- Used to render HTML templates.
request:- Used to handle HTTP requests.
joblib:- Used to load the pre-trained model and encoder.
pandas:- Used for data manipulation and creating data frames.

To Loading Pre-trained Model and Encoder

```
# Load the saved pipeline and label encoder
pipeline = joblib.load('your_pipeline.joblib')
label_encoder = joblib.load('label_encoder.joblib')
```

The joblib.load function is used to load the previously saved machine learning pipeline and label encoder. The pipeline includes data preprocessing steps and the trained model, while the label encoder is used to transform the target variable back to its original labels.

To Setting Up Flask Application

app = Flask(__name__)

Explanation:

A Flask application instance is created. __name__ is a special variable in Python that holds the name of the current module, and it is used to initialize the Flask app.

To Creating Routes for the Web Application

```
@app.route('/')
def index():
    return render_template('index.html')
```

Explanation:

- The **@app.route('/')** decorator defines the root URL route for the application. The **index** function renders the **index.html** template, which serves as the home page of the web application.

To Handling User Input and Making Predictions

```
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        graduation = request.form['graduation']
        interest = request.form['interest']
        prerequisites = request.form['prerequisites']
        entrance_test_score = float(request.form['entrance_test_score'])
```

```
    # Create a DataFrame with the input data
    input_data = pd.DataFrame({
        'Graduation': [graduation],
        'Interest': [interest],
        'prerequisites': [prerequisites],
        'entrance test score': [entrance_test_score]
    })

    # Use the label encoder to transform the input data
    input_data['Recommend Course'] =
label_encoder.inverse_transform(pipeline.predict(input_data))

    return render_template('index.html',
recommended_course=input_data['Recommend Course'].values[0])
```

Explanation:

The @app.route('/predict', methods=['POST']) decorator defines the /predict URL route, which accepts POST requests.

The function extracts user inputs from the form (graduation, interest, prerequisites, entrance_test_score) and creates a DataFrame (input_data) to hold these values.

The pipeline.predict(input_data) method is used to predict the recommended course based on the input features. The label_encoder.inverse_transform method converts the numeric prediction back to the original course label.

The result is rendered back to the index.html template, displaying the recommended course to the user.

Running the Flask Application

```
if __name__ == '__main__':
    app.run(debug=True)
```

Explanation:

The if __name__ == '__main__': block ensures that the Flask app runs only if the script is executed directly (not imported as a module).

app.run(debug=True) starts the Flask web server in debug mode, which provides detailed error messages and auto-reloads the server on code changes.

Conclusion of Deployment

This deployment code effectively creates a web application using Flask to serve a machine learning model. Users can input relevant features through a web form, and the application processes this input to provide course recommendations.

The use of Flask, along with joblib for model loading and pandas for data handling, makes the deployment efficient and user-friendly. This setup can be expanded and customized further for different types of models and input features.

Chapter 5: Conclusion

The course recommendation system project aimed to create an effective machine learning model to suggest courses for students based on their profiles. The project followed several key steps, including data preprocessing, exploratory data analysis (EDA), model training, and evaluation.

Data Preprocessing

Data preprocessing was essential for ensuring the dataset's quality and usability. Unnecessary columns were removed, missing values were handled, and irrelevant records were filtered out. Categorical variables were encoded, and the data was split into training and testing sets.

Exploratory Data Analysis (EDA)

EDA provided valuable insights into the dataset, revealing important patterns and relationships. Analyzing the distribution of categorical variables like Graduation, Interest, and prerequisites, and the numerical variable entrance test score, helped in understanding the data better. Visualizing the relationship between features and the target variable Recommend Course highlighted the key predictive features.

Model Training

The XGBoost classifier was selected for its efficiency and high performance. A pipeline was created to streamline preprocessing and model training. The model was trained on the preprocessed training data, learning patterns to predict course recommendations based on student profiles.

Model Evaluation

The trained model achieved a satisfactory accuracy score on the test set. Additional metrics such as precision, recall, and F1-score provided a comprehensive evaluation of the model's performance, indicating its strengths and areas for improvement. The model and preprocessing pipeline were saved using **joblib**, ensuring ease of deployment.

Overall Impact and Future Work

The project successfully developed a machine learning-based course recommendation system, assisting students in choosing courses that align with their profiles and interests. Future improvements could include enhancing data quality, hyperparameter tuning, developing a user-friendly interface, and implementing continuous learning. This project demonstrates the potential of machine learning in providing personalized educational guidance, and with further development, it can become a valuable tool for students and educational institutions.

Chapter 7: References

1)Pandas Documentation:

- URL: https://pandas.pydata.org/

2) Scikit-Learn Documentation:

- URL: https://scikit-learn.org/

3) XGBoost Documentation:

- URL: https://xgboost.readthedocs.io/

4)Matplotlib Documentation:

- URL: https://matplotlib.org/

5) Seaborn Documentation:

URL: https://joblib.readthedocs.io/

6)Joblib Documentation:

URL: https://joblib.readthedocs.io/

7)Python Programming Language:

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

These references provide foundational knowledge and documentation for the tools, libraries, and methodologies used in the course recommendation system project, ensuring the implementation of efficient data processing and machine learning models.

CHAPTER 9 : GLOSSARY OF TERMS

- **Machine Learning:** A field of computer science that allows computers to learn from data without explicit programming.
- **Model:** A representation of the learned patterns from data that can be used to make predictions on new data.
- **Overfitting:** When a model memorizes the training data too well and fails to generalize to unseen data.
- **Gradient Boosting:** An ensemble machine learning technique that combines multiple weak models (like decision trees) into a stronger final model.
- **XGBoost (eXtreme Gradient Boosting):** A powerful machine learning algorithm known for its efficiency and accuracy in gradient boosting.
- **Regularization:** Techniques to prevent overfitting in machine learning models.
- **Model Evaluation:** The process of assessing how well a model performs on unseen data.
- **Training Set:** The data used to train the model and help it learn patterns.
- **Validation Set:** The data used to fine-tune the model's hyperparameters (settings that control the learning process).
- **Test Set:** Unseen data used for the final evaluation of the model's generalizability.
- **Metrics:** Measures used to evaluate a model's performance. Examples include accuracy, precision, recall, F1-score (for classification)

Glossary of Terms in Deployment

. Flask

A lightweight web framework for Python that allows the development of web applications. Flask provides tools, libraries, and technologies to build a web application.

. render_template

A Flask function used to render an HTML template file. It combines the template with the data provided and returns an HTML string.

. request

An object in Flask that contains all the data sent from the client to the server, such as form data submitted via POST requests.

. joblib

A Python library used for serializing and deserializing Python objects, particularly useful for saving and loading machine learning models.

. pandas (pd)

A powerful data manipulation and analysis library for Python. It provides data structures like DataFrames, which are essential for handling and processing structured data.

. pipeline

In machine learning, a pipeline is a sequence of data processing steps, where the output of one step is the input to the next. It often includes steps for data preprocessing and model training.

. label_encoder

An encoder that converts categorical labels into numeric format (integers) and vice versa. It is used to handle categorical variables in machine learning tasks.

. DataFrame

A two-dimensional, size-mutable, and potentially heterogeneous tabular data structure in pandas, with labeled axes (rows and columns). It is similar to a table in a database or an Excel spreadsheet.

. route

In Flask, a route is a URL pattern that is used to map web addresses to specific functions in the application. Routes are defined using the @app.route decorator.

. POST request

A type of HTTP request used to send data to the server to create or update a resource. In a web form, POST requests are typically used to submit form data.

. inverse_transform
A method in label encoders that converts encoded labels back to their original categorical values.

. debug mode
A mode in Flask that provides detailed error messages and auto-reloads the server whenever code changes are detected. It is useful for development and debugging.

. HTML template
An HTML file used as a template to generate dynamic content on a web page. In Flask, templates are rendered with data passed from the server to create a complete HTML page.

. Serialization
The process of converting a Python object into a format that can be saved to a file or transmitted over a network. In the context of this project, it refers to saving a trained model and preprocessing pipeline using joblib.

. Deserialization
The process of converting a serialized format back into a Python object. In this project, it refers to loading the saved model and preprocessing pipeline.

. Input features
The variables or columns in a dataset that are used as inputs to a machine learning model. In this project, they include Graduation, Interest, prerequisites, and entrance test score.

. Prediction
The output generated by a machine learning model based on the input features. In this project, the prediction is the recommended course.

. Web application
An application that is accessed via a web browser over a network such as the Internet or an intranet. In this project, the Flask app serves as the web application.

. Preprocessing
The process of transforming raw data into a format that can be used for training a machine learning model. This includes steps such as encoding categorical variables and handling missing values.

Thank you !