BTP-2 (LoCoML)

Project Report

Team Members:

- Ayush Agrawal (2020101025)
- Siddharth Mavani (2020101122)
- Rohan Chowdary (2020101130)

Project Guide: Dr. Karthik Vaidhyanathan

Introduction

The advent of machine learning has opened up a plethora of opportunities across various domains. However, the process of training and deploying machine learning models often requires a high level of technical expertise and can be time-consuming. This project aims to streamline this process by introducing a low code machine learning platform that automates the entire process, making it accessible to individuals with limited technical abilities.

The platform is designed with a user-friendly interface that allows users to train and deploy models with just a few clicks. It caters not only to novices but also to those with a background in machine learning, providing them with the flexibility to customize models according to their needs.

The platform boasts a wide array of features aimed at enhancing the user experience and providing comprehensive insights into the models. These include model deployment, versioning, and a dashboard that provides statistics and visualizations of the model's performance. The versioning feature allows users to track and manage different versions of their models.

Furthermore, the platform integrates a robust analytics suite, enabling users to understand their model's performance in depth. This includes metrics such as accuracy, precision, recall, and F1 score, among others. Visualizations such as ROC curves, confusion matrices, and feature importance plots provide further insights into the model's behavior.

In addition to these, the platform also supports various machine learning algorithms and provides recommendations on the best algorithm to use based on the dataset. It also handles data preprocessing tasks such as handling missing values, encoding categorical variables, and scaling features, thereby reducing the data preprocessing burden on the user.

In conclusion, this platform serves as a comprehensive solution for individuals looking to leverage machine learning without the need for extensive technical knowledge. It simplifies the process of model training and deployment while still offering a high degree of flexibility and control to the users. Through this project, we aim to democratize access to machine learning and enable more people to harness its potential.

Design

The design of the platform is centered around the principles of simplicity, accessibility, and flexibility. Here are the key aspects of the design:

1. **User Interface (UI)***:* The platform features a clean and intuitive user interface, designed to be easily navigable even by individuals with limited technical abilities. The UI uses simple and clear language, with tooltips and guides to assist users in understanding various features and functionalities.
2. **User Experience (UX)***:* The platform is designed to provide a seamless user experience. Tasks such as data upload, model selection, training, and deployment are streamlined into a step-by-step process, reducing the complexity typically associated with these tasks.
3. **Modular Design***:* The platform follows a modular design approach. Each feature - data preprocessing, model training, evaluation, and deployment - is a separate module that can be used independently. This allows for easy updates and enhancements to individual modules without affecting the rest of the platform.
4. **Dashboard***:* The platform features a comprehensive dashboard that provides an overview of the user's projects, including the models trained, their performance metrics, and deployment status. This allows users to manage their projects effectively.

Features

The low code ML platform project has been designed with a wide array of features to make machine learning accessible to individuals with limited technical abilities. Here are the key features of the platform:

1. **Model Deployment**: This feature allows users to deploy their trained machine learning models on Docker, a platform that packages and runs applications in loosely isolated environments called containers. The deployed model runs on a Flask server inside the Docker container, which can be used to make predictions. This feature is implemented in the deployModel.py script.

2. **Exploratory Data Analysis (EDA)**: EDA is a crucial step in any data science project. It involves understanding the underlying structure of the data, identifying outliers and anomalies, discovering patterns, and testing underlying assumptions. The platform provides comprehensive EDA on datasets, including computing statistical measures for numerical columns and counting unique and missing values for all columns. This feature is implemented in the eda.py script.

3. **Dataset Management**: The platform allows users to store new datasets and fetch information about existing datasets. It also provides the functionality to retrieve the column names of the dataset on which a particular model was trained. This feature is implemented in the getDatasets.py and storeDataset.py scripts.

4. **Model Management**: The platform provides functionalities to fetch information about already trained models and retrieve the pickle file of a specific model version. This feature is implemented in the getTrainedModels.py script.

5. **Inference**: The platform provides endpoints for making predictions using trained models. It supports both single and batch inference modes. In single inference mode, a prediction is made for a single instance, while in batch inference mode, predictions are made for multiple instances. This feature is implemented in the inference.py script.

6. **Data Preprocessing**: The platform provides an endpoint for preprocessing a specified dataset, including tasks like dropping duplicate rows, interpolating missing values, and normalizing features. This feature is implemented in the preprocess.py script.

7. **Model Training**: The platform provides an endpoint to train a new model. It takes in all the necessary fields for training a model and trains a new model accordingly. After training, it stores the model info in the database and the model pickle file on the local disk system. This feature is implemented in the trainModel.py script.

8. **Model Updating**: The platform provides endpoints related to updating a model. Users can update a model in three ways: training on additional data, changing the hyperparameters, or changing the type of the estimator. The newly trained model is stored as a new version of the existing model. This feature is implemented in the updateModel.py script.

9. **Automated Machine Learning (AutoML)**: The platform supports both AutoML and custom training modes. In AutoML mode, it trains multiple classifiers or regressors and selects the best one based on a specified metric. In custom mode, it trains a specified model with given hyperparameters. This feature is implemented in the ClassificationUtility.py and RegressionUtility.py scripts in the functions folder.

Technologies used

1. **Flask**: Flask is a lightweight web framework for Python. It was used to build the backend of the platform, including all the API endpoints. Flask's simplicity and flexibility made it an ideal choice for this project.

2. **Redis**: Redis is an open-source, in-memory data structure store used as a database, cache, and message broker. In this project, Redis was used for creating a socket connection between the frontend and backend to update the user about the model training status in the progressbar.

3. **React**: React is a popular JavaScript library for building user interfaces, particularly single-page applications. It was used to build the frontend of the platform, providing a dynamic and interactive user interface.

4. **MongoDB**: MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. It was used to store information about datasets, models, and other metadata.

5. **Docker**: Docker is a platform that uses OS-level virtualization to deliver software in packages called containers. Docker was used to deploy the trained models, allowing them to be easily distributed and run in any environment.

Future Work

The future work for this project involves several exciting expansions and enhancements:

1. **Deep Neural Network Models**: The platform currently supports a variety of machine learning models. However, the power of deep learning, particularly deep neural networks, is yet to be harnessed. Future work will involve integrating deep learning capabilities into the platform, allowing users to train and deploy complex neural network models.
2. **Expansion into Vision, NLP, and Speech**: The platform aims to expand its capabilities to include more specialized fields of machine learning such as computer vision, natural language processing (NLP), and speech recognition. This would enable users to train models for tasks like image classification, sentiment analysis, language translation, speech-to-text, and more.
3. **Model Linking**: One of the most exciting future enhancements is the ability to create links between different parts of a model. For instance, a translation model developed in NLP can be connected to a text-to-speech (TTS) model in the speech domain to create a model that outputs in regional languages. This would allow users to build complex, multi-stage models that can perform a series of tasks.
4. **Customizable Model Pipelines**: Along with model linking, the platform aims to provide users with the ability to create customizable model pipelines. Users could design their own pipelines, choosing which models to link and in what order. This would offer users unprecedented control over their machine learning workflows.
5. **Advanced Analytics and Visualizations**: As the platform grows and supports more complex models and tasks, there will be a need for more advanced analytics and visualizations. Future work will involve developing new ways for users to understand and interpret their models' performance.

6. **Scalability and Performance Enhancements**: As more features are added and the platform handles more complex tasks, there will be a continuous need to improve its scalability and performance. Future work will involve optimizing the platform's architecture and implementing new technologies as needed.

These enhancements will significantly increase the platform's capabilities, making it an even more powerful tool for individuals looking to leverage machine learning without the need for extensive technical knowledge. The ultimate goal is to democratize access to machine learning and enable more people to harness its potential.

Conclusion

In conclusion, the low code ML platform project has made significant strides in democratizing access to machine learning. By streamlining the process of training and deploying models, the platform enables individuals with limited technical abilities to leverage the power of machine learning. The platform's wide array of features, including model deployment, exploratory data analysis, dataset management, model management, inference, data preprocessing, model training, model updating, and utilities, make it a comprehensive solution for machine learning tasks.

The future work for this project is equally exciting, with plans to expand into deep neural network models, vision, NLP, and speech. The ability to create links between different parts of a model opens up new possibilities for complex, multi-stage models. With these enhancements, the platform will continue to grow and evolve, furthering its mission to make machine learning accessible to all.

The project's success so far is a testament to the power of the technologies used, including Flask, Redis, React, MongoDB, and Docker. Each of these technologies has played a crucial role in the development of the platform, contributing to its functionality, performance, and usability.