Project Report

Grant Jones/Cy Dixon

11/1/2025

CS456G

Project Description

This project is a web-based machine learning platform developed using the Django framework. It allows users to upload datasets in various formats (.txt, .csv, and Excel) and automatically train, evaluate, and visualize machine learning models based on user input. Once a dataset is uploaded, the system preprocesses the data and applies a range of algorithms from the Scikit-learn library to generate results. The platform supports models such as Linear Regression, Logistic Regression, Decision Trees, Bagging and Boosting ensembles, Random Forests, Support Vector Machines, and user-defined Deep Neural Networks.

After training, the system evaluates model performance using key metrics such as accuracy, precision, recall, and ROC curves, along with other measures like feature importance. These results are displayed in an interactive dashboard through charts, tables, and metric cards designed for readability. Each user’s datasets and models are protected through Django’s authentication system, ensuring both privacy and personalized model management. Overall, the platform provides an accessible and organized way for users to explore machine learning models without manually writing code.

Functions Implemented

The application relies on a series of Python functions that perform data preprocessing, model training, evaluation, and visualization. Within the Scikit-learn library, the system provides methods such as train_test_split() for dividing data into training and testing subsets, SimpleImputer() for handling missing values, and StandardScaler() for feature normalization.

We organized machine learning logic in the modelutils.py file, which performs tasks such as train_model(), which fits the selected Scikit-learn methods to the provided data. It also implements evaluate_model(), which computes metrics like accuracy, precision, recall, F1-score, and AUC using functions from sklearn.metrics. Another important function, generate_visualizations(), uses Matplotlib to produce charts such as ROC curves, confusion matrices, and feature importance plots that are embedded in the dashboard.

Technical Details

The system is built primarily with Python as the backend language and Django as the core web framework. The project uses SQLite for database management, which integrates smoothly with Django's ORM to store user accounts, datasets, and model information. Key libraries include Scikit-learn for machine learning, Pandas for data manipulation, and Matplotlib for creating visualizations.

Development was done in Visual Studio Code, and the application runs locally on Django's development server. The frontend is created using Django templates with embedded HTML, CSS, and JavaScript, which manage dashboard interactivity and data display. JavaScript handles dynamic updates and enhances the responsiveness of the user interface.

Highlighted Features

A standout feature of this project is its clean and intuitive user interface. After logging in, users see a personalized dashboard showing their uploaded datasets and trained models. Each dataset view includes information such as file size, number of columns and rows, and the amount of missing data. Users can easily select their target and feature variables through clickable column headers before training models.

Model evaluation results are presented through clear tables and visualizations that compare and rank models by performance metrics. Charts allow users to quickly interpret accuracy, precision, recall, and ROC curve data. Each user's datasets and models are stored separately and securely using Django's authentication framework. With its simple navigation, minimal design, and interactive visuals, the dashboard makes it easy to understand and compare complex machine learning outputs.

Discussion

Throughout the project, Python proved to be the most effective language for implementing machine learning workflows because of its simplicity and wide library support. Scikit-learn provided reliable tools for traditional models such as regression, decision trees, and deep neural networks (using MLPClassifier and MLPRegressor). Combining these tools with Django made it possible to create a complete application that supports both model training and web deployment.

From a web development perspective, Django was an ideal framework due to its built-in user authentication, ORM, and structured project organization. Features like the @login_required decorator helped improve security, and Django's template system made it easy to integrate visual results directly into the web pages.

There are several alternative approaches that could be taken for a similar project. Frameworks such as Flask or FastAPI could be used for a lighter backend, while front-end frameworks like React or Vue.js could handle visualization more dynamically. A production-ready version might also use a more scalable database, such as PostgreSQL, and containerization tools like Docker for deployment.

Overall, the combination of Django, Scikit-learn, and PyTorch provided a balanced environment for development and experimentation. The project shows how machine learning and web technologies can work together to create a user-friendly platform for analyzing and visualizing data models.