Project Summary: Board Game Rating Predictor Web Application

This project involved developing a web application to predict board game ratings using machine learning, integrated with a user authentication system built with Django.

- **Dataset Description:** The application uses a Project.csv dataset containing features for various board games like publication year, player counts, play time, user ratings, BGG rank, complexity, ownership numbers, and categorical data for mechanics and domains. The target variable for prediction is the "Rating Average."
- **ML Model Training Steps:** The ML pipeline, developed in a Jupyter Notebook, included:
    1. **Preprocessing:** Loading data, dropping irrelevant columns, and converting categorical features (Mechanics, Domains) into numerical counts.
    2. **Model Selection:** Explored Linear Regression (baseline) and Random Forest Regressor.
    3. **Optimization:** The Random Forest model was tuned using both GridSearchCV and RandomizedSearchCV, with RandomizedSearchCV yielding the best performance (R^2 score ~0.93).
    4. **Persistence:** The two best-performing optimized Random Forest models were pickled for use in the web application.
- **Authentication Implementation:** A standard Django authentication system was added, featuring:
    1. **User Actions:** Login, logout, and new user registration.
    2. **Technology:** Utilized Django's built-in LoginView, LogoutView, and a custom view with UserCreationForm for registration.
    3. **Security & UX:** Logout uses POST requests for security. django.contrib.messages provides user feedback, and views are protected with @login_required where necessary. Forms are styled using django-crispy-forms with Bootstrap 4.

- **Integration Steps:**
    1. **Core App Setup:** A Django project (titanic) with an app (base) was structured.
    2. **Model Loading:** Pickled ML models were integrated into base/views.py to be loaded at application startup.
    3. **Page Structure:**
        - A public landing page (/) was created, which doubles as a dashboard for logged-in users.
        - A separate prediction tool page (/predict-tool/) for authenticated users allows input of game features to get a rating prediction.
    4. **Dashboard Content:** For authenticated users, the landing page/dashboard displays:
        - Dataset summary statistics (e.g., record count, feature summaries from Project.csv).
        - Performance metrics (MAE, RMSE, R^2) for both saved ML models.
        - Visualizations (target variable distribution, correlation heatmap, feature importances for both models) generated using Matplotlib/Seaborn and embedded as images.
    5. **UI:** Bootstrap 4 provides the base styling, enhanced with custom CSS and django-crispy-forms for cleaner form rendering. Template inheritance (base_generic.html) ensures consistent layout.
- **Challenges Encountered:**
    1. **File Paths:** Ensuring correct paths for loading Project.csv and pickled models within the Django environment.
    2. **Template Logic:** Correcting Django template syntax for conditional rendering.
    3. **Logout Mechanism:** Resolving an HTTP 405 error by changing the logout link to a POST request form.
    4. **Library Warnings:** Addressing FutureWarnings from Seaborn by updating plotting function parameters.
    5. **Feature Consistency:** Aligning feature sets between the Jupyter Notebook training environment and the Django view for accurate feature importance display.