Project Overview

Objective:
Develop a web application that predicts credit scores (Good/Standard/Poor) using machine learning and Django. Key Features:
✓ User authentication (login/registration)
✓ Financial data input forms
✓ Real-time credit score prediction
✓ Admin dashboard for model management Technology
Stack:

- Frontend: HTML/CSS, JavaScript, Bootstrap
- Backend: Django (Python)
- ML Model: Random Forest (72.2% accuracy)
- Database: MySql

1. Dataset Description

Source: https://www.kaggle.com/datasets/parisrohan/credit-score-classification
Purpose: Predict creditworthiness (categorized as ***Poor, Standard,*** *or* ***Good***) based on financial behavior and demographic data.

Key Features

Category	Feature	Description	Type
Personal Info	Age	Customer's age in years	Numerical
	Occupation	Employment sector (e.g., "Scientist", "Teacher")	Categorical
Financials	Annual_Income	Gross yearly earnings (USD)	Numerical
	Monthly_Inhand_Salary	Net monthly take-home pay	Numerical
	Num_Bank_Accounts	Number of active bank accounts	Numerical
Credit History	Num_Credit_Card	Number of active credit cards	Numerical
	Interest_Rate	Average APR (%) on credit products	Numerical
	Num_of_Loan	Total active loans	Numerical
	Type_of_Loan	Categories of loans (e.g., "Auto", "Mortgage")	Categorical
Payment Behavior	Delay_from_due_date	Average days overdue per payment	Numerical
	Num_of_Delayed_Payment	Count of late payments (last 12 months)	Numerical
	Changed_Credit_Limit	Recent adjustments (±%) to credit limit	Numerical
Credit Profile	Credit_Mix	Diversity of credit types (e.g., "Standard", "Good")	Categorical
	Outstanding_Debt	Total current debt owed (USD)	Numerical
	Credit_Utilization_Ratio	Percentage of total credit limit used	Numerical
	Credit_History_Age	Years since first credit account	Numerical

Category	Feature	Description	Type
Behavioral Metrics	Payment_Behaviour	Spending/repayment habits (e.g., "High spend, low payments")	Categorical
	Monthly_Balance	Average end-of-month bank balance	Numerical
Target Variable	Credit_Score	Risk classification: **Poor**, **Standard**, **Good**	

2. ML Model Training Steps

a. Data Processing

1. Removal of unnecessary columns (ID, Customer_ID, Month, Name, SSN)

2. Handling anomalies:
   - Fixed unreasonable values in Num_Credit_Card column (values >10 replaced with median)
   - Normalized credit utilization ratios to 0-100 range

3. Feature engineering:
   - Created debt_income_ratio
   - Created salary_emi_ratio
   - Generated high_income_flag

b. Model Training

1. Data Preparation
   - Split dataset into training (80%) and testing (20%) sets
   - Applied standard scaling to numerical features
   - Encoded categorical variables using one-hot encoding

2. Model Selection
   - Tested multiple algorithms:
   - Random Forest
   - AdaBoost
   - Logistic Regression
   - SVM
   - Gradient Boosting

   Selected Random Forest based on performance metrics

c. Hyperparameter Tuning

- Used GridSearchCV for optimization
- Tuned parameters:
- n_estimators
- max_depth
- min_samples_split

d. Model Evaluation

- Accuracy
- Precision
- Recall
- F1-Score
- Cross-validation scores

3. Authentication

Initial Set Up

1. Added 'django.contrib.auth' to INSTALLED_APPS in settings.py
2. Enabled built-in authentication system
3. Used Django's default authentication backend.
    - Configured in forms.py
    - Handles database authentication
4. Used default User model (django.contrib.auth.models). Includes built-in fields:
    - username
    - password
    - email
    - first_name/last_name
    - Permissions system
5. Added built-in auth URLs in forms.py:
    - /accounts/login/
    - /accounts/logout/
    - Password change/reset endpoints

Authentication Views

1. Registration View:
    - Handles new account creation.
    - Validates and saves user data (username, email, password).
    - Auto-logs in users upon successful registration.
2. Login View:
    - Authenticates existing users via username/email and password.
    - Redirects to the prediction page on success; displays errors otherwise.
3. Logout View:
    - Terminates user sessions securely.

Form Implementation

- Registration Form:
    - Fields: username, email, password (with confirmation).
    - Validation:
        - Password strength requirements (min length, special chars).
        - Unique email constraint.
- Login Form:
    - Fields: username/email, password.
    - Validates credentials against the database.

Security Measures

- Password Hashing: Uses Django's PBKDF2 for secure storage.
- Session Management: Tracks active logins via request.session.
- CSRF Protection: Enabled for all form submissions.
- Form Validation: Prevents SQL injection/XSS via Django's built-in sanitization.

Access Control

- @login_required Decorator: Restricts sensitive views (e.g., prediction form) to authenticated users.
- Redirects: Unauthenticated users are redirected to the login page.

---

4. Steps for Integration

Project Structure Setup

1. Created Django project structure
2. Set up main app (predictor)
3. Organized files:

    - Templates (HTML)
4. Created directories for:
    - ML model storage
    - Data files

Database Integration

1. Configured mySQL database
2. Created models for:
    - User data
    - Prediction history
    - Model metadata
3. Executed migrations to create tables

ML Model Integration

1. Saved trained model using joblib
2. Created model loading utility
3. Prediction pipeline:
    - Data preprocessing
    - Feature scaling
    - Model prediction
    - Result formatting

Backend Integration

1. Created Django views for:
    - Model prediction
    - Data validation
    - Result handling
2. Configured URL routing
3. Implemented:
    - Form handling
    - Error handling

Frontend Integration

1. Designed base template
2. Implemented:
    - Prediction form
    - Result display section
3. Added:
    - Bootstrap styling
    - JavaScript for:
        - Form validation
        - AJAX submission
        - Dynamic updates

API Integration

1. Created endpoints for:
    - Prediction requests
    - User authentication
    - Data validation
2. Implemented:
    - Error response formatting

Authentication Integration

1. Integrated Django auth system
2. Added forms for:
    - Login
    - Registration
3. Configured:
    - User sessions

- Access control

Form Validation Integration

1. Implemented:
    - Client-side validation
    - Server-side validation
2. Created custom validators
3. Added error message handling

5. Problems Encountered (but all fixed)

1. In data cleaning, issues are found including duplication of data columns, anomalies everywhere, and low accuracy target.
2. URLs and paths configuration
3. Database connection mysql is not set at first since it's always error.
4. Forms are not connected to the model itself wherein the inputs are not same with the model in terms of number.
5. Modals are not showing, especially in prediction.
6. Analytics are not load correctly,
7. User authentication in Django not working.
8. Repository problems: Can't push and can't find files.