**Machine Learning Techniques for Accurate Flight Delay Forecasting**

**Software Related to the Project**

* **Python (3.8.2)**
* **Django**
* **SQLite3**

### ****Python (3.8.2):****

### 

Python 3.8.2 served as the foundational programming language for this project due to its simplicity, readability, and extensive support for machine learning applications. It provided a powerful yet beginner-friendly platform to implement backend logic, perform data analysis, and train multiple machine learning models.

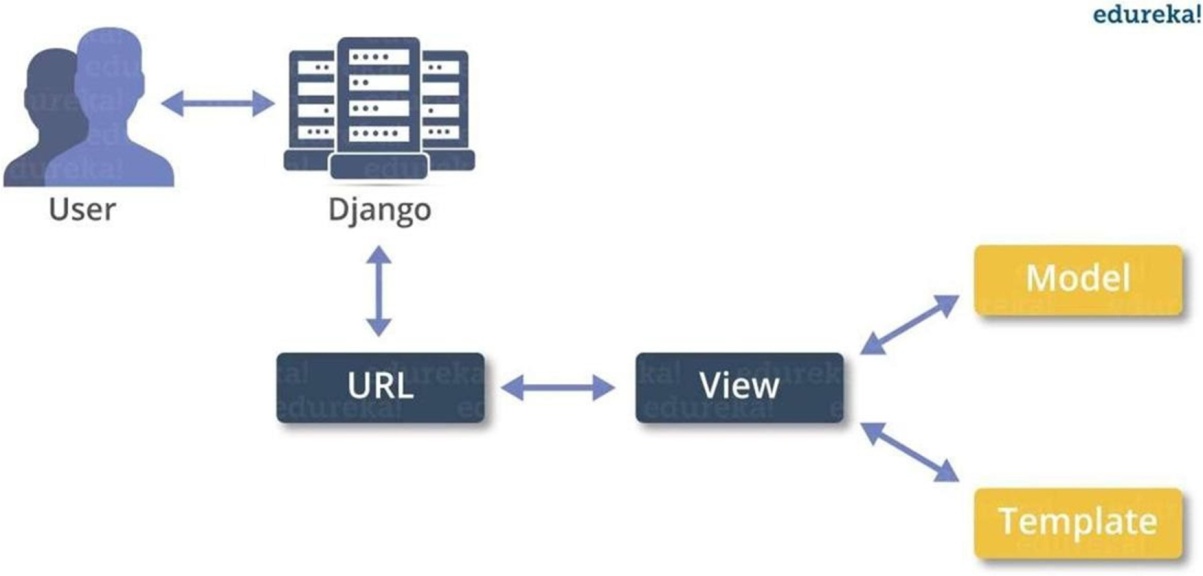
Python’s rich ecosystem enabled seamless integration of key libraries and tools used in this project, including:

* **pandas**: for importing, cleaning, and preprocessing historical flight data such as flight schedules, departure/arrival times, and delay durations.
* **numpy**: for numerical computations, array handling, and feature transformation.
* **scikit-learn**: for applying and evaluating machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting.
* **matplotlib**: for generating visual graphs of accuracy trends, model performance, confusion matrices, and delay patterns.

#### Key Features Implemented Using Python:

* **Data Preprocessing**: Cleaned the flight dataset by removing missing values, handling outliers, encoding categorical variables like airline and day of the week, and scaling features for better model accuracy.
* **Model Training**: Applied and trained several machine learning models on historical flight data to classify whether a flight is likely to be delayed or on-time.
* **Model Evaluation**: Evaluated all models using standard classification metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score**.
* **Prediction API Integration**: Integrated the trained machine learning model into Django views to provide real-time predictions based on flight input provided by the user.

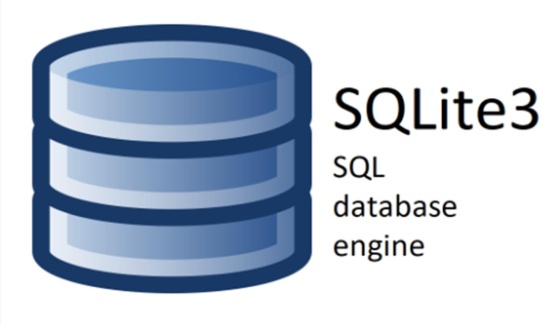
### ****Django Framework:****

Django is a high-level, open-source Python web framework that was used to build the web interface and handle server-side logic for the application. It follows the **Model-View-Template (MVT)** architecture, which made it easier to manage the flow of data and rendering of results.

#### Highlights of Django in this Project:

* **MVT Architecture**:
  + **Models** were used to store user data, prediction logs, and flight records.
  + **Views** processed user inputs, called the ML prediction function, and returned results.
  + **Templates** rendered clean, user-friendly pages with results and prediction forms.
* **Admin Panel**:
  + Provided administrators with a panel to manage user accounts, monitor system usage, and view prediction history.
  + Allowed for easy dataset management, including adding or deleting records.
* **User Authentication**:
  + Secure login and registration for users.
  + Ensured that only authenticated users could access prediction functionality.
* **ML Model Integration**:
  + The trained model, saved as a .pkl file, was loaded into Django views.
  + On form submission, user data was passed to the model and prediction results were returned instantly.
* **Security Features**:
  + Implemented CSRF protection, session cookies, and HTTPS support to safeguard user data.

### ****SQLite3 Database:****

SQLite3 was used as the backend database for storing user accounts, prediction input/output logs, and admin activity records. It is a lightweight, file-based database that integrates easily with Django’s ORM system.

#### Usage in the Project:

* **Data Storage**:
  + Stored user profiles, submitted flight details, prediction outcomes, and timestamps.
  + Maintained logs for system monitoring and access control.
* **Django ORM Integration**:
  + Models were created using Django ORM and automatically mapped to SQLite3 tables.
  + Simplified database migrations, queries, and maintenance without writing SQL code manually.
* **Performance**:
  + Provided fast read/write operations for this single-user development setup.
  + Its portability made it easy to transfer or back up the complete dataset as a fi