**Summary of Training Scripts**

**1. Defect Prediction Model**

**Objective:**  
Predict the number of defects in a software project based on various factors like development effort, code complexity, and testing coverage.

**Features Used:**

* **Numerical Features:**
  + defect\_fix\_time\_minutes
  + size\_added
  + size\_deleted
  + size\_modified
  + effort\_hours
  + complexity\_score
  + testing\_coverage
* **Categorical Features:**
  + team\_key (encoded using **Target Encoding**)

**ML Techniques Used:**

* **Stacking Regressor** with base models:
  + **Random Forest Regressor**
  + **XGBoost Regressor**
  + **Gradient Boosting Regressor**
  + **LightGBM Regressor**
* **ElasticNet** as the meta-model.

**Why These Techniques?**

* **Stacking Regressor** combines the strengths of multiple models to improve accuracy.
* **Random Forest** and **XGBoost** handle non-linearity and complex interactions well.
* **Gradient Boosting** reduces variance, and **LightGBM** enhances efficiency.
* **ElasticNet** prevents overfitting by combining L1 (Lasso) and L2 (Ridge) regularization.

**Why Not Other Techniques?**

* **Linear Regression** assumes a linear relationship, which is unrealistic for defect prediction.
* **Deep Learning** is unnecessary due to the relatively small dataset size.
* **Single Decision Trees** lack generalization ability, leading to overfitting.

**2. Project Timeline Prediction Model**

**Objective:**  
Predict the duration (in days) required to complete a software project based on various project characteristics.

**Features Used:**

* Team\_Size
* Task\_Count
* Developer\_Experience
* Priority\_Level
* Task\_Complexity
* Project\_Size
* Effort\_Density
* Team\_Productivity
* LoC\_per\_Team\_Member

**ML Techniques Used:**

* **Stacking Regressor** with base models:
  + **Random Forest Regressor**
  + **XGBoost Regressor**
  + **Gradient Boosting Regressor**
* **ElasticNet** as the meta-model.

**Why These Techniques?**

* **Stacking improves accuracy** by leveraging multiple weak learners.
* **XGBoost and Gradient Boosting** excel at handling structured tabular data.
* **ElasticNet** balances bias-variance tradeoff to prevent overfitting.

**Why Not Other Techniques?**

* **Simple Regression Models** would not capture complex relationships between features.
* **Neural Networks** would be overkill given the dataset size and nature.
* **Time Series Models** were not used since the problem is not sequentially dependent.

**3. Task Allocation Model**

**Objective:**  
Predict the best team for a given task based on task complexity, specialization, and past team performance.

**Features Used:**

* **Categorical Features:**
  + task\_type
  + specialization
* **Numerical Features:**
  + task\_complexity
  + task\_priority
  + estimated\_effort\_hours
  + team\_experience\_level
  + past\_projects\_completed

**ML Techniques Used:**

* **Random Forest Classifier** (best model found via GridSearchCV)
* **SMOTE (Synthetic Minority Over-sampling Technique)** to handle class imbalance.
* **One-Hot Encoding & Standard Scaling** for preprocessing.

**Why These Techniques?**

* **Random Forest Classifier** is robust and interpretable.
* **SMOTE** balances the dataset to improve model fairness.
* **GridSearchCV** ensures the best hyperparameters are selected.

**Why Not Other Techniques?**

* **K-Nearest Neighbors (KNN)** is computationally expensive with high-dimensional data.
* **SVM (Support Vector Machine)** struggles with large datasets.
* **Neural Networks** require a large amount of data and fine-tuning, which was not practical.