**Book Impact Prediction**

**Objective:**

1. The goal of the project is to predict book impact based on text (such as book descriptions, titles, and other metadata).
2. The task involves building a feature-rich representation of books that can help in predicting “Impact” based on both textual and numerical information, leveraging Spark’s distributed processing framework.

**Approach**:

1. The project follows a typical machine learning pipeline that includes data preprocessing, feature engineering, model training, evaluation, and deployment using MLFlow for tracking and logging experiments.
2. The model's performance is evaluated using metrics such as RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error).
3. Different models like Linear Regression are trained to predict the "Impact" based on the input features.

**Key Steps Taken:**

1. **Data Loading**:
   * The dataset (books\_task.csv) is loaded into a Spark DataFrame using spark.read.csv.
   * Necessary preprocessing steps are carried out to clean and transform the data.
2. **Data Preprocessing**:
   * Missing values in text columns such as "description," "Title," "categories," and "publisher" are filled with default values ("Unknown").
   * Text data from "Title" and "description" columns are merged to create a combined text feature, merged\_text.
   * Text data is preprocessed by:
     + Converting to lowercase for uniformity.
     + Removing special characters using regular expressions.
     + Tokenizing the text and filtering out stop words.
     + Transforming the tokens into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency).
   * Categorical features like "categories," "authors," and "publisher" are transformed into numerical features using Feature Hashing.

Note:

1. We could have also used Text Embedding based model to convert Description and Titles into embeddings and used it for data context. But for simplicity we have used TF-IDF based numerical representations for training of the model.
2. Different variants of Categorical encoding methods could have been used like Frequency based categorical encoder. Since, categorical features contains huge number of distinct values in it, it would be difficult to use onehot encoder as it will increase the dimension of the model hence, we used hashing technique to encode categorical features.
3. **Feature Engineering**:
   * A VectorAssembler is used to combine the features (TF-IDF features and hashed features) into a single feature vector (features column).
   * Optionally, PCA (Principal Component Analysis) could be applied to reduce dimensionality (though not included in the final pipeline).
4. **Model Selection**:
   * Models considered for training:
     + **Linear Regression**: A baseline model for regression tasks.
     + **Random Forest** and **Gradient Boosted Trees (GBT)** were also considered but not included in the final run.
5. **Model Training and Evaluation**:
   * The dataset is split into training and testing sets using a random split (80% for training, 20% for testing).
   * MLflow is used for experiment tracking, including logging parameters, metrics, and models.
   * A cross-validation strategy with 2 folds is used to train the model, and RMSE is used as the evaluation metric.
   * The best model is selected based on the cross-validation performance.
   * Metrics such as RMSE and custom MAPE are calculated to evaluate model performance.
6. **Hyperparameter Tuning**:
   * The project includes a ParamGridBuilder for hyperparameter tuning (though no specific grid is set in the final run).
   * Cross-validation is performed with the best model selected based on RMSE.
7. **MLFlow Integration**:
   * MLflow is set up for experiment tracking and model logging.
   * Each experiment is logged with key metrics (RMSE, MAPE) and the best model is saved in the MLflow tracking server.
   * Visualizations of the actual vs. predicted values are generated for model evaluation.
8. **Training with Different Worker Configurations**:
   * The experiment is run with different Spark worker configurations (1, 2, and 4 workers) to evaluate the impact of parallelism on training time.
   * A separate Spark session is created for each worker configuration, and the total training time is measured for comparison.
9. **Performance Results**:

* The final results include metrics such as RMSE and MAPE, which help assess the accuracy and performance of the model.
* A comparison of model training times for different worker configurations is also provided.