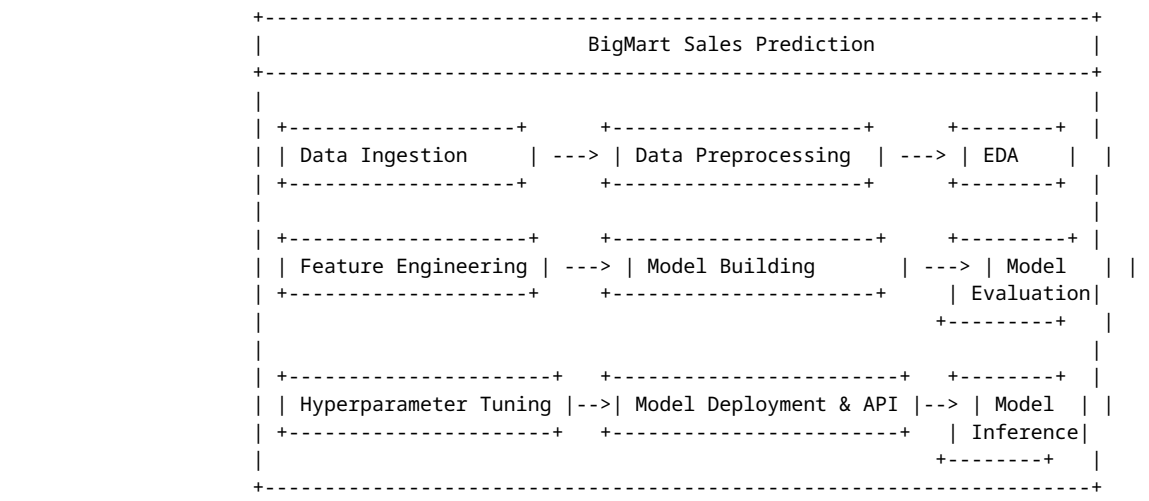
Architecture Design for BigMart Sales Prediction Project

The architecture of the BigMart Sales Prediction Project will follow a layered approach, ensuring clear separation of tasks such as data ingestion, preprocessing, model training, evaluation, and deployment. Below is the detailed architecture:

---

1. Architecture Diagram



---

2. Components Description

2.1 Data Ingestion Layer

- **Purpose:** Load the sales dataset from the CSV file into the system.
- **Tools:** `pandas`
- **Responsibilities:**
  - Load raw data into memory.
  - Inspect initial data structure (columns, types, missing values).
  - Validate data format and ensure no corrupted records.

2.2 Data Preprocessing Layer

- **Purpose:** Clean and preprocess the data to ensure it is ready for machine learning models.
- **Tools:** `pandas`, `numpy`
- **Responsibilities:**
  - Handle missing values (imputation).
  - Deal with outliers (capping or removing).
  - Normalize or scale features as needed (e.g., `Item_Visibility`, `Item_MRP`).
  - Convert categorical variables into numeric formats (label encoding or one-hot encoding).

2.3 Exploratory Data Analysis (EDA) Layer

- **Purpose:** Analyze the data to identify key trends and relationships between variables.
- **Tools:** `matplotlib`, `seaborn`
- **Responsibilities:**
  - Perform univariate analysis (e.g., distribution of sales).

- Conduct bivariate analysis to check relationships (e.g., between `Item_MRP` and `Item_Outlet_Sales`).
- Create visualizations like scatter plots, box plots, and histograms.

2.4 Feature Engineering Layer

- **Purpose:** Derive new meaningful features from the existing data to improve model performance.
- **Tools:** `pandas`, `numpy`
- **Responsibilities:**
  - Create new features such as `Outlet_Age`.
  - Group product categories into broader categories (e.g., Food, Non-Food).
  - Create interaction terms to capture product-store relationships.
  - Scale/normalize features if needed.

2.5 Model Building Layer

- **Purpose:** Train various machine learning models on the preprocessed dataset to predict sales.
- **Tools:** `sklearn`, `xgboost`
- **Responsibilities:**
  - Split the dataset into training and test sets.
  - Train multiple models (Linear Regression, Ridge/Lasso, RandomForest, XGBoost).
  - Evaluate models based on metrics like MAE and R^2.
  - Perform cross-validation to ensure model stability.

2.6 Model Evaluation Layer

- **Purpose:** Evaluate model performance and select the best model for deployment.
- **Tools:** `sklearn.metrics`
- **Responsibilities:**
  - Compare model performance using MAE (Mean Absolute Error) and R^2 (R-squared).
  - Ensure that the model generalizes well to unseen data by testing it on the test set.

2.7 Hyperparameter Tuning Layer

- **Purpose:** Optimize the hyperparameters of the chosen models to achieve the best performance.
- **Tools:** `GridSearchCV`, `RandomizedSearchCV` (from `sklearn`)
- **Responsibilities:**
  - Perform grid search or randomized search to fine-tune hyperparameters (e.g., `n_estimators`, `max_depth` for RandomForest).
  - Track performance improvement after tuning.

2.8 Model Deployment & API Layer

- **Purpose:** Deploy the trained model for use in production and make predictions for new data.
- **Tools:** `Flask` or `FastAPI` (for API deployment), `joblib` or `pickle` (for model saving)
- **Responsibilities:**
  - Save the trained model using `joblib` or `pickle`.
  - Deploy the model as a REST API using `Flask` or `FastAPI`.
  - Expose API endpoints for prediction requests (input: product and store details; output: predicted sales).

2.9 Model Inference Layer

- **Purpose:** Use the deployed model to make real-time predictions on new or unseen data.
- **Tools:** `Flask`/`FastAPI` (API calls), `joblib` (load model)
- **Responsibilities:**
  - Accept new data as input via API calls.
  - Use the saved model to predict sales.
  - Return predictions as a JSON response.

---

3. Data Flow

1. **Data Ingestion:**
   - Load data from the CSV file into the system.
   - Ensure data consistency and format validation.
2. **Data Preprocessing:**
   - Clean the dataset by handling missing values and outliers.
   - Transform categorical variables into numerical format (label encoding, one-hot encoding).
   - Create new features like `Outlet_Age`.
3. **Exploratory Data Analysis (EDA):**
   - Analyze the distributions of numerical and categorical variables.
   - Visualize relationships between features and target variable.
4. **Feature Engineering:**
   - Add new features or interaction terms that might enhance model performance.
   - Group categories and apply transformations where necessary.
5. **Model Building:**
   - Train several machine learning models (Linear Regression, Ridge/Lasso, RandomForest, XGBoost).
   - Evaluate models using training and test data.
6. **Model Evaluation:**
   - Calculate metrics such as MAE and R^2 on the test set.
   - Select the best-performing model based on evaluation results.
7. **Hyperparameter Tuning:**
   - Optimize hyperparameters for RandomForest or XGBoost using GridSearchCV or RandomizedSearchCV.
8. **Model Deployment:**
   - Save the final trained model as a `.pkl` file.
   - Deploy the model using a REST API, exposing prediction endpoints.
9. **Model Inference:**
   - Make predictions for new products or outlets using the deployed API.

---

4. Technologies and Tools

Component	**Technology/Tool**
Data Ingestion	`pandas`
Data Preprocessing	`pandas`, `numpy`
Exploratory Data Analysis (EDA)	`matplotlib`, `seaborn`

Feature Engineering	`pandas`, `numpy`
Model Building	`sklearn`, `xgboost`
Model Evaluation	`sklearn.metrics`
Hyperparameter Tuning	`GridSearchCV`, `RandomizedSearchCV`
Model Saving	`joblib`, `pickle`
API Deployment	`Flask`, `FastAPI`
Model Inference	`Flask`, `joblib`

---

5. Summary

The architecture for the BigMart Sales Prediction project consists of multiple layers, from data ingestion and preprocessing to feature engineering, model training, and deployment. It is designed to efficiently handle data, build predictive models, and serve predictions in real time via a REST API. By following this architecture, BigMart can leverage machine learning to gain actionable insights and forecast future sales, ultimately boosting profitability.