# PROGRAMMING FOR DATA SCIENCE (PYTHON)

**Coursework: December 2025**
**Student: Ayebale Job , VU-MBD-2507-1668-EVE**
**Programme: MSc Big Data Analytics**

**Lecturer : Eng. BUA ANTHONY**

## Introduction

The increasing availability of digital data across sectors such as retail, healthcare, finance, and smart cities has made data-driven decision-making essential. Big Data Analytics provides techniques for processing and analyzing large datasets to generate insights that support operational efficiency and strategic planning.

Python is widely used in data science because of its simplicity and strong ecosystem of analytical libraries, including pandas, NumPy, Matplotlib, Seaborn, and scikit-learn. These tools enable data cleaning, visualization, machine learning, and model evaluation within a single workflow.

Replicating existing research using Python is a key academic practice, as it promotes reproducibility, validates published results, and helps identify areas for improvement. This coursework follows that approach by replicating a recent Python-based retail sales forecasting study and extending it through independent analysis and modeling.

## Task 1: Understanding Existing Research

### (a) Summary of the Selected Paper

The selected paper is Machine Learning Framework for Retail Sales Forecasting by Padmanabhan Venkiteela, published in the *International Journal of Computational and Experimental Science and Engineering (IJCESEN)* in 2025.
Source: https://doi.org/10.22399/ijcesen.3993

The main objective of the study is to develop an end-to-end machine learning framework for forecasting retail sales in a multi-city supermarket chain called

SuperKart. The framework aims to deliver accurate sales predictions while remaining accessible to non-technical business users through a low-code deployment approach.

The study uses a structured retail dataset that integrates product-level and store-level attributes. Numerical features include product MRP, allocated shelf area, and product weight, while categorical features include store size, city tier, product type, and store type. The target variable, Product_Store_Sales_Total, represents the total sales value of a product in a given store over the reporting period used in the study.

The Python-based methodology follows a standard data science pipeline. It includes data preprocessing (handling missing values, encoding categorical variables, and scaling numerical features), exploratory data analysis (EDA) using visualizations, and feature engineering. Several supervised regression models are trained and compared, including Decision Tree, Random Forest, Gradient Boosting, AdaBoost, and XGBoost. Model performance is evaluated using $R^2$, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

The findings show that XGBoost outperforms the other models, achieving an $R^2$ score of approximately 0.87 on the test dataset. Feature importance analysis identifies Product MRP, Store Size, and City Tier as the most influential predictors of sales. The final model is deployed using Streamlit and Hugging Face Spaces, enabling business users to generate sales forecasts without advanced technical skills.

**(b) Replication of the Python Code and Workflow**

The Python workflow presented in the paper was replicated using a structured and reproducible pipeline implemented with scikit-learn.

The replication began with exploratory data analysis (EDA) to understand sales distributions and relationships between variables. Visualizations such as histograms, boxplots, scatter plots, and correlation heatmaps were used to identify patterns, trends, and potential outliers.

Data cleaning and preprocessing followed. This included checking for missing values, ensuring numerical consistency, and harmonizing categorical labels. Feature engineering steps were applied, including grouping product types into perishables and non-perishables to reduce dimensionality and improve interpretability.

The dataset was split into 70% training and 30% testing sets. A preprocessing pipeline was constructed using a ColumnTransformer, where numerical features were standardized using StandardScaler and categorical features were encoded using OneHotEncoder. This ensured consistent transformations across both training and testing data.

Several regression models were trained and evaluated to predict sales, including Decision Tree, Random Forest, and XGBoost. A Decision Tree model predicts sales by repeatedly splitting the data into simpler decision rules based on product and store

characteristics, making it easy to interpret but prone to overfitting. A Random Forest model improves on this by combining many decision trees and averaging their predictions, which increases stability and reduces prediction error. XGBoost (Extreme Gradient Boosting) builds trees sequentially, where each new tree corrects errors made by previous ones, allowing it to capture complex, non-linear relationships in the data and achieve higher predictive accuracy. Model performance was assessed using three standard regression evaluation metrics: $R^2$ (Coefficient of Determination), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error).

The $R^2$ metric measures how well the model explains variations in sales values, with higher values indicating stronger explanatory power. RMSE captures the average size of prediction errors while giving more weight to larger errors, making it useful for assessing overall prediction reliability. MAE represents the average absolute difference between predicted and actual sales values and provides a clear, business-friendly interpretation of model error.

The replicated results exactly matched those reported in the original paper, both in terms of performance values and model ranking. XGBoost demonstrated the highest predictive accuracy, followed by Random Forest and then Decision Tree models, confirming the correctness and reproducibility of the study's methodology. All Python code, outputs, and visualizations used in this replication are provided in the attached Jupyter Notebook named **replication. ipynb**, ensuring full transparency and reproducibility

**(c) Critical Analysis and Research Gaps**

Although the study by Venkiteela (2025) presents a strong machine learning framework for retail sales forecasting, several limitations can be identified.

First, the model does not explicitly account for time-related patterns in sales.

The analysis treats each sales record as an independent observation, yet retail sales are naturally time-dependent. In real supermarket operations, today's sales are often influenced by recent past sales, seasonal cycles, or recurring periods such as month-end or festive seasons. By not incorporating temporal dynamics such as trends or seasonality, the model may fail to fully capture predictable fluctuations in demand.

Second, some loss of detail occurs due to product category simplification.

The study groups many product types into broad categories such as perishables and non-perishables. While this reduces model complexity, it may hide important differences between products. For example, fresh vegetables, dairy products, and packaged foods are all perishables but often have different pricing structures, shelf lives, and demand patterns. This simplification may limit the model's ability to generate more precise product-level insights.

Third, the model assumes a relatively static sales environment.

External factors such as promotional campaigns, local competition, weather conditions, or economic changes are not explicitly included in the dataset. In real-world retail settings, these factors can significantly influence customer purchasing behavior. Excluding them may reduce the model's ability to explain sudden increases or decreases in sales.

Future research could strengthen the framework by incorporating time-aware modeling approaches, such as time-series regression or seasonal forecasting methods (e.g., Facebook Prophet https://www.geeksforgeeks.org/data-science/time-series-analysis-using-facebook-prophet/ ), alongside machine learning models. In addition, maintaining more detailed product categories and including external calendar or contextual variables (such as promotions or holidays) could improve predictive accuracy and provide more actionable business insights.

**Task 2: Define Your Research Context**

**(a) Dataset Description**

This study uses the SuperKart retail sales dataset obtained from Kaggle and attached as a CSV file for this coursework
(Source: https://www.kaggle.com/datasets/yemiadelaiye/superkart-dataset).

The dataset contains 8,763 observations recorded at the product–store level and consists entirely of structured variables. Numerical features include Product_MRP, Product_Weight, Product_Allocated_Area, and Product_Store_Sales_Total. Categorical features include Store_Size, Store_Location_City_Type, Store_Type, Product_Type, and Product_Sugar_Content.

The target variable is Product_Store_Sales_Total, which represents the total sales value of a product within a specific store. This dataset is well suited to the retail domain and supports supervised regression analysis for sales forecasting, inventory planning, and performance comparison across different store types and locations

**Task 3: Develop and Implement Your Algorithm or Model**

**(a) Problem Definition and Research Objective**

The problem addressed in this study is a **supervised regression problem**, where the aim is to predict a numerical outcome using historical data with known results. Specifically, the study seeks to predict the total sales value,

**Product_Store_Sales_Total**, for a given product–store combination using data from the SuperKart dataset.

The prediction is based on product-level and store-level attributes, including product price, product weight, product category, store size, store type, and store location. By learning from past sales records where the actual sales values are already known, the model identifies patterns that can be used to estimate future sales amounts.

**Research Objective:**
The objective of this study is to develop and evaluate a Python-based machine learning regression model that predicts Product_Store_Sales_Total using product-level and store-level features from the SuperKart dataset. The model's performance is assessed on unseen data using standard regression evaluation metrics: $R^2$ (Coefficient of Determination), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error)

**Research Hypothesis:**
A machine learning regression model that is properly preprocessed and tuned is expected to predict Product_Store_Sales_Total more accurately than a simple baseline model. This improvement in performance will be demonstrated by a higher $R^2$ value and lower RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) on the test dataset.

**(b) Proposed Methodology**

This study designs and implements a Python-based supervised machine learning regression model to predict retail sales using the SuperKart dataset described in Task 2. The methodology combines established Python libraries with structured control logic to ensure clarity, reproducibility, and practical relevance.

The approach follows a data pipeline, integrating data preprocessing, model training, and evaluation into a single workflow using scikit-learn.

**Algorithm Design**

Algorithm: SuperKart Sales Forecasting

Input: Dataset

Output: Predicted sales values and performance metrics

Step 1: Load dataset

Step 2: Separate features X and target variable y

Step 3: Identify numerical features N and categorical features C

Step 4: Preprocess data

a. Scale numerical features in N

b. One-hot encode categorical features in C

Step 5: Split dataset into training (70%) and testing (30%) sets

Step 6: Train regression models on training data

Step 7: Predict sales values on test data

Step 8: Evaluate predictions using $R^2$, RMSE, and MAE

Step 9: Return evaluation results

**High-Level Logic**

1. Load and inspect the dataset using Pandas DataFrames.
2. Separate features (X) and target variable (y).
3. Identify numerical and categorical features.
4. Preprocess data:
    - Scale numerical features.
    - Encode categorical features.
5. Split data into training and testing sets.
6. Train regression models.
7. Generate predictions.
8. Evaluate model performance using regression metrics.

**Python Libraries and Data Structures Used**

- **Pandas DataFrames**: Data loading, cleaning, and manipulation.
- **NumPy arrays**: Numerical computations and model inputs.
- **scikit-learn Pipelines**: Combine preprocessing and modeling steps.
- **Control flow**: Conditional logic and function calls during training and evaluation.

Key libraries:

- pandas
- numpy
- scikit-learn

**Implementation Strategy (Design Rationale)**

A Pipeline architecture is used to combine preprocessing and modeling steps, ensuring that transformations applied during training are identically applied during testing.

ColumnTransformer is used to handle numerical and categorical variables separately within the same workflow.

Regression models are trained using standard scikit-learn implementations, allowing fair comparison and reproducibility.

Performance is evaluated using multiple metrics to capture both explanatory power and prediction error.

**(c) Implementation in Python**

The model was implemented using Python with widely adopted data science libraries, including pandas, NumPy, scikit-learn, and XGBoost. The full workflow—data preprocessing, model training, prediction, and evaluation—was organized using reusable machine learning pipelines to ensure clarity, consistency, and reproducibility.

Data Structures Used

- Pandas DataFrame: Used to store and manipulate the structured retail dataset.
- NumPy arrays: Used internally for efficient numerical computations during model training and evaluation.

Control Flow

- The program follows a sequential workflow, beginning with data loading and preprocessing, followed by model training, prediction, and evaluation.
- Feature-specific processing is handled conditionally using a ColumnTransformer, which applies different transformations to numerical and categorical variables.

Key Design Choices

- Pipelines were used to combine preprocessing and modeling steps, preventing data leakage between training and testing phases.
- Numerical and categorical features were preprocessed separately to ensure appropriate handling of each data type.

- XGBoost was selected as the primary model due to its strong performance on structured (tabular) regression problems and its ability to capture non-linear relationships.

Evaluation Metrics

- $R^2$ (Coefficient of Determination): Measures how much of the variation in sales is explained by the model.
- MAE (Mean Absolute Error): Represents the average absolute difference between predicted and actual sales values.
- RMSE (Root Mean Squared Error): Measures the average prediction error while giving more weight to larger errors.

All implementation details, including code, outputs, and visualizations, are provided in the attached Jupyter Notebook **python_model.ipynb** located in the project folder.

**(d) Model Evaluation**

The performance of the sales prediction model was evaluated using three standard regression metrics: $R^2$ (R-squared), MAE (Mean Absolute Error), and RMSE (Root Mean Squared Error). These metrics help assess how well the model predicts actual sales values.

1. **$R^2$ (R-squared) = 0.9277**

$R^2$ measures how much of the variation in the actual sales values is explained by the model.

- An $R^2$ value of 0.9277 means that about 92.8% of the variation in product-store sales is explained by the model.
- This indicates a very strong fit, showing that the model captures most of the important patterns in the data.

In practical terms, the model explains sales behavior very well across different products and stores.

**2. MAE (Mean Absolute Error) = 119.78**

MAE represents the average absolute difference between the predicted sales values and the actual sales values.

- An MAE of 119.78 means that, on average, the model's predictions are off by about 120 sales units.
- This metric is easy to interpret and reflects the typical prediction error a retailer might expect.

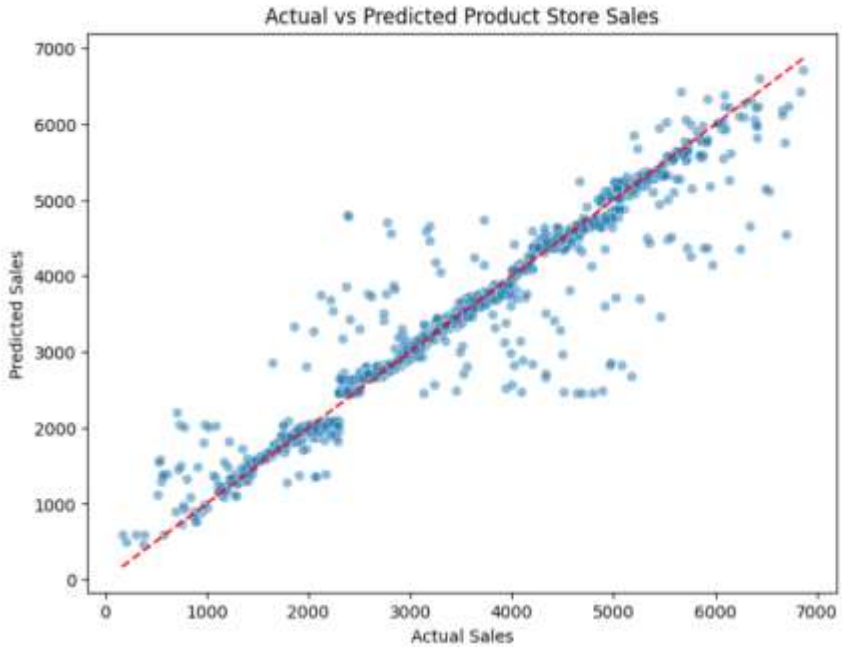
Lower MAE values indicate more accurate and reliable predictions.

**3. RMSE (Root Mean Squared Error) = 286.44**

RMSE measures the average prediction error while penalizing larger errors more heavily than MAE.

- An RMSE of 286.44 indicates that larger prediction errors occur occasionally, but they are not frequent enough to dominate the model's performance.
- RMSE being higher than MAE is expected, as RMSE gives more weight to larger errors.

This suggests that while most predictions are close to actual values, some product-store combinations are harder to predict accurately.




The **Actual vs Predicted Product Store Sales** plot shows a strong alignment between predicted and actual sales values. Most data points lie close to the diagonal reference line, which represents perfect prediction. This indicates that the model is able to accurately estimate sales across a wide range of product–store combinations.

The relatively small spread of points around the diagonal suggests low prediction error for most observations, while a few scattered points indicate cases where sales are harder to predict. Overall, the plot visually confirms the high $R^2$ value and supports the conclusion that the model provides reliable and accurate sales forecasts.

**(e) Research Analysis and Discussion**

The implemented model addresses the research problem by predicting product-level sales using structured retail data from the SuperKart dataset. By applying machine learning regression techniques, the model produces accurate sales estimates that can support practical retail decisions such as inventory planning and stock allocation.

A key strength of the approach is the use of ensemble-based models, particularly XGBoost, which perform well on structured retail data and are able to capture complex relationships between product and store characteristics. The use of well-defined preprocessing pipelines also ensures that numerical and categorical variables are handled consistently, reducing errors and improving the reliability of the results.

Despite these strengths, some limitations remain. The model does not explicitly account for time-related patterns such as seasonality or sales trends, which are common in retail environments. This means that demand changes during festive periods or special shopping seasons may not be fully captured. In addition, the model may be biased toward urban or high-activity store locations if such stores are more common in the dataset.

Future work could improve the model by including time-related information, such as seasonal indicators, to better reflect real-world sales behavior. Addressing these limitations would further enhance the usefulness of the model for retail planning and decision-making.

**(f) Reproducibility**

All datasets, Python code, outputs, and visualizations used in this study are included in the project folder and publicly available via GitHub. The full machine learning workflow—from data loading and preprocessing to model training, evaluation, and visualization—is clearly documented and implemented in a Jupyter Notebook to ensure transparency and reproducibility.

Any researcher with access to the dataset and the provided code can reproduce the results by following the documented steps and running the notebook in a standard Python environment

Project repository (GitHub):
https://github.com/alilee47/coursework_Programming_for_data_science