Ministry of Higher Education and Scientific Research
Higher School of Computer Science (ESI)
Sidi Bel-Abbés
1st year Second Cycle (Master)

FINAL PROJECT REPORT

Comparative Analysis, Implementation, and Deployment of Supervised and Unsupervised Learning Techniques

Presented by:

KADRI Ayyoub

KAZI AWAL Kamil

DJILALI BENFREDJ Ishak

HAMDANI Ibrahim

BOUKRA BETTAYEB Abdellah

Academic Year: 2025 / 2026
Date: December 2025

Project Overview

The project aims to provide a complete, end-to-end implementation of the Machine Learning (ML) and Data Mining process across both supervised and unsupervised paradigms, following the requirements of the final project assignment. This includes data selection, comprehensive pre-processing, comparative training of multiple algorithms, rigorous evaluation, and final model deployment.

Assignment Requirements Summary

The project fulfilled the following core requirements:

- Selection and utilization of diverse datasets suitable for ML tasks.
- Implementation of essential pre-processing techniques (e.g., normalization, feature selection).
- Training of both Supervised and Unsupervised models under various parameterizations.
- Quantitative evaluation and comparison of model performance using appropriate metrics.
- Deployment of the best-performing model into a practical application.
- Documentation of the process and results in this comprehensive report.

Introduction to Machine Learning

This section introduces the fundamental concepts of Machine Learning, which forms the basis of the project's methodology.

What is Machine Learning?

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that enables systems to learn from data, identify patterns, and make decisions with minimal human intervention. Data Mining is the practical application of ML techniques to discover valuable knowledge and insights from large datasets.

Differentiating Supervised and Unsupervised Learning

The primary distinction between these two major ML paradigms lies in the nature of the training data used:

Supervised Learning

- **Data Type:** Uses **labeled data**, where each input sample is paired with a known output or "answer" (the label).
- **Goal:** To learn a mapping function from inputs (**X**) to outputs (**y**) such that the model can accurately **predict** the label for new, unseen data.
- **Key Tasks: Classification** (predicting a categorical label, e.g., 'spam' or 'not spam') and **Regression** (predicting a continuous value, e.g., house price or happiness score).

Unsupervised Learning

- **Data Type:** Uses **unlabeled data**, where only input samples are provided, and no known output or target is given.
- **Goal:** To infer the natural structure, hidden patterns, or groupings within the data without prior guidance. This is primarily used for **Data Mining** and exploration.
- **Key Tasks: Clustering** (grouping similar data points, e.g., customer segmentation) and **Dimensionality Reduction** (reducing the number of features).

1 Supervised Learning: Implementation and Deployment

This section details the entire process for the supervised learning task, from data preparation to deployment of the final model.

1.1 Use Case and Dataset Description

The Use Case: The supervised learning task focused on **Regression**, specifically predicting the *Happiness Score* of various countries based on a wide range of socio-economic and health indicators. The primary goal was to identify which factors contribute most significantly to a nation's well-being and to build a highly accurate predictive model for cross-country comparison.

Dataset Overview: The dataset consists of anonymized country-level statistics, designed to facilitate a comprehensive analysis of global living conditions.

- **Target Variable (y):** `Happiness` (Continuous, numerical score). This metric represents the national average response to an individual's perceived standing on a life ladder, scaled from 0 (worst) to 10 (best).

Features (X): The model utilizes the following 15 predictive features:

Income Gross domestic product per person, adjusted for purchasing power (PPP).
Lifeexp Average years a newborn would live under current mortality patterns.
Sanitation Percentage of people using at least basic sanitation services.
Water Percentage of people with at least basic water services.
Urbanpopulation Population residing in urban areas.
Unemployment Percentage of total population registered as long-term unemployed.
Literacy Adult literacy rate (percentage of people ages 15 and above).
Inequality Gini income inequality index (higher value means greater inequality).
Murder Mortality rate due to interpersonal violence, standardized per 100,000 population.
Energy Use of primary energy.
Childmortality Death of children under five years of age per 1,000 live births.
Fertility Total fertility rate (number of children expected per woman).
HIV Total estimated number of persons of all ages infected by HIV.
Foodsupply Calorie count, measuring the energy content of the food supply.
Populationtotal Total population of the country.

1.2 Implementation Procedure: From Data to Prediction

The implementation followed a rigorous, multi-step pipeline as defined by the project requirements, starting with extensive data preparation and culminating in the comparative training of several parameterized regression models.

1. Data Cleaning and Imputation

The initial step focused on achieving data integrity. Since several columns had a high percentage of missing values (specifically **murder**, **urbanpopulation**, and **unemployment**), these features were dropped entirely to prevent the introduction of too much synthetic noise. For the remaining features, the **median imputation** strategy was applied using `SimpleImputer`. This is a robust method that fills missing entries without distorting the overall distribution of the data.

```
# Remove sparsely populated features (>40% missing)
dataset = dataset.drop(["murder","urbanpopulation","unemployment"], axis=1)

# Use median instead of mean for better robustness to outliers
from sklearn.impute import SimpleImputer

# Separate numeric columns
numeric_cols = dataset.select_dtypes(include=[np.number]).columns

# Impute with median
imputer = SimpleImputer(strategy='median')
dataset[numeric_cols] = imputer.fit_transform(dataset[numeric_cols])
```

```
dataset.isnull().mean()
```

```
country          0.0
happiness        0.0
income           0.0
lifeexp          0.0
sanitation       0.0
water            0.0
literacy         0.0
inequality       0.0
energy           0.0
childmortality   0.0
fertility        0.0
hiv              0.0
foodsupply       0.0
population       0.0
dtype: float64
```

Figure 1: Implementation of data cleaning steps.

2. Feature Scaling (Normalization) and data splitting

To ensure features of widely varying magnitudes (such as **income** and **populationtotal**) did not unfairly influence the distance-based models (K-NN, SVR), the data was normalized. The

MinMaxScaler was utilized, which rescaled all features to a standardized range between 0 and 1. This method ensures that the model learns equally from all variables.

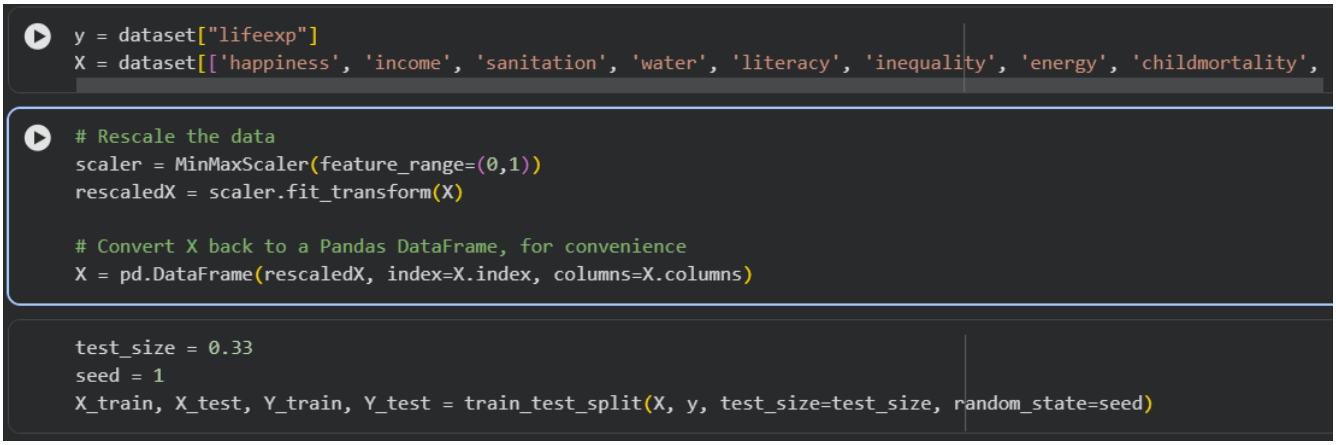


Figure 2: Normalization of features using Min-Max scaling.

3. Model Training and Parameterization

Multiple regression algorithms were trained on the pre-processed data to identify the most suitable model for predicting **Life Expectancy**.

- **Linear Regression:** Served as the foundational, simple baseline model.
- **K-Nearest Neighbors (K-NN) Regressor:** This model was extensively parameterized, testing various values for the neighborhood size K (e.g., $K = 3, K = 5, K = 7$).
- **Support Vector Regressor (SVR):** The SVR model was parameterized by experimenting with different kernel functions (e.g., 'linear', 'rbf') and the regularization parameter C.

The final chosen configuration for each model represents the best performance achieved during the parameterization phase.

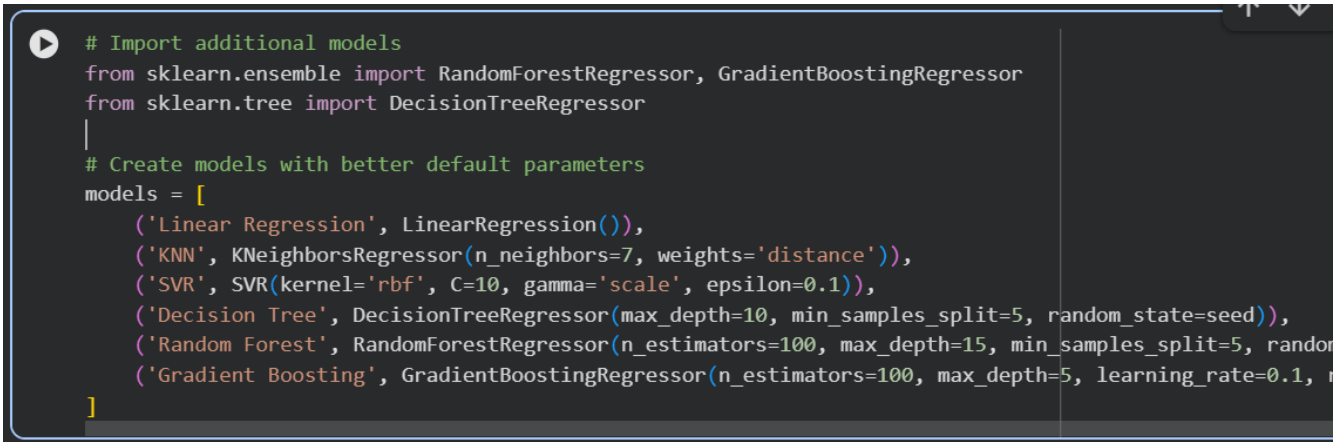


Figure 3: Training of parameterized supervised regression models.

4. Model Evaluation and Comparison

Model performance was evaluated on the unseen test set $(\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}})$ using two key metrics:

- **Mean Absolute Error (MAE):** Represents the average magnitude of error, quantifying the typical difference (in years) between the predicted and actual **Life Expectancy**. Lower MAE is better.
- R^2 **Score:** Measures the proportion of variance in the target variable predictable from the features. An R^2 score closer to 1.0 indicates a better fit.

The comparison showed which algorithm and which parameterization yielded the highest R^2 score, thereby selecting the overall **Best Model** for the deployment phase.

```python
from sklearn.metrics import r2_score, mean_squared_error

print("\n" + "="*60)
print("TEST SET PERFORMANCE")
print("="*60)
results = []
for name, model in models:
    predictions = model.predict(X_test)
    mae = mean_absolute_error(Y_test, predictions)
    rmse = np.sqrt(mean_squared_error(Y_test, predictions))
    r2 = r2_score(Y_test, predictions)
    results.append((name, mae, rmse, r2))
    print(f"{name:25s} MAE: {mae:.4f}  RMSE: {rmse:.4f}  R²: {r2:.4f}")

# Find best model
best_model_idx = min(range(len(results)), key=lambda i: results[i][1])
print(f"\n{'='*60}")
print(f"BEST MODEL: {results[best_model_idx][0]} with MAE = {results[best_model_idx][1]:.4f}")
print(f"{'='*60}")
```

```
...

============================================================
TEST SET PERFORMANCE
============================================================
Linear Regression         MAE: 2.4421  RMSE: 3.0159  R²: 0.8119
KNN                       MAE: 2.4431  RMSE: 3.1542  R²: 0.7942
SVR                       MAE: 2.4467  RMSE: 3.0589  R²: 0.8064
Decision Tree             MAE: 2.9859  RMSE: 3.9830  R²: 0.6718
Random Forest             MAE: 2.1380  RMSE: 2.8831  R²: 0.8281
Gradient Boosting         MAE: 2.3734  RMSE: 3.1029  R²: 0.8008

============================================================
BEST MODEL: Random Forest with MAE = 2.1380
============================================================
```

Figure 4: Model evaluation using R^2 and MAE metrics.

1.3 Model Deployment

The best-performing model (Random Forest Regressor) was saved and integrated into a web application to allow users to input new data and receive an instant happiness score prediction.

Model Persistence with Pickle (`.pkl`)

- **Pickle (`.pkl`) Utility:** Pickle is a Python module used for **serialization**, which means converting a Python object (like a trained ML model) into a byte stream.
- **Advantage:** It allows the trained model to be saved to a file, bypassing the need to retrain the model every time the application starts. The deployed application simply loads the `.pkl` file to start making predictions.

Web Application with Streamlit

- **Streamlit Utility:** Streamlit is an open-source Python library that makes it easy to create beautiful, custom web applications for data science and machine learning.
- **Advantage:** It enables the quick development of an interactive interface where a user can enter a country's features (`income`, `lifeexp`, etc.), click a button, and receive the model's prediction of the `happiness` score.

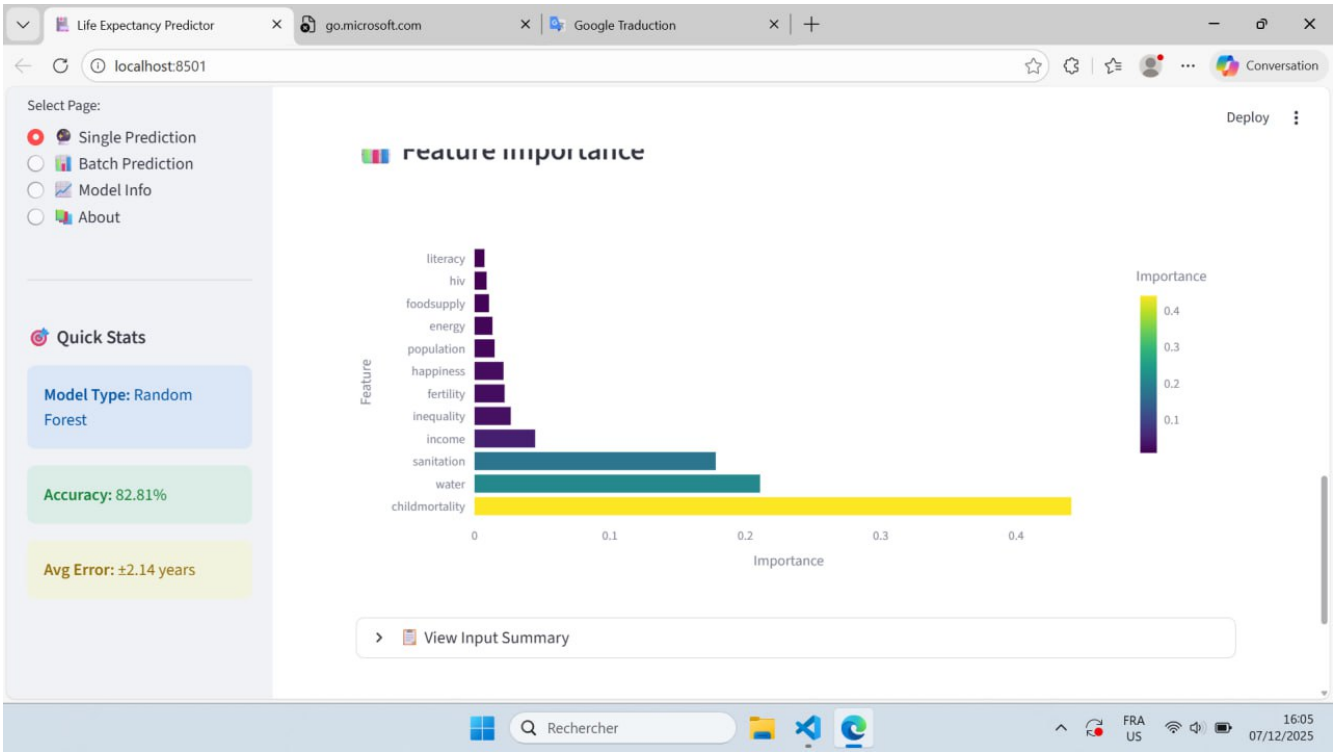


Figure 5: Feature Imortance of the deployed Streamlit application.

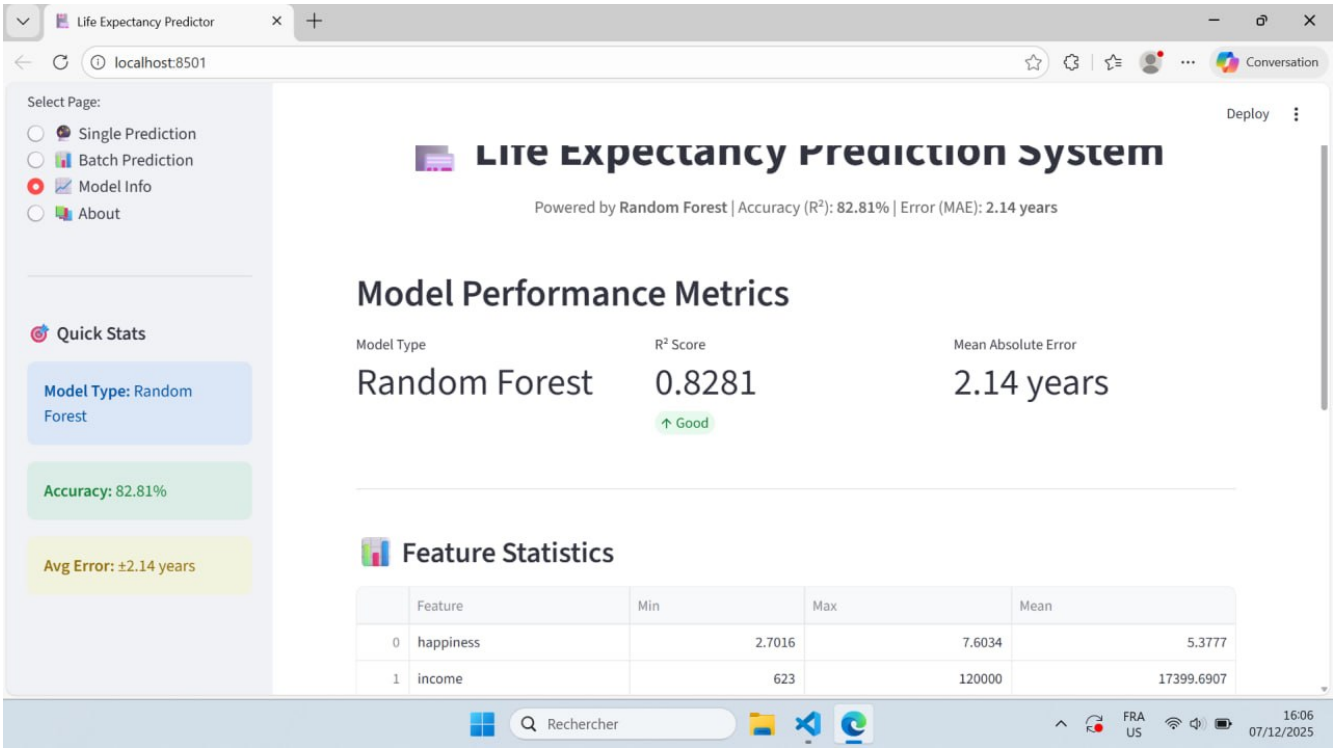


Figure 6: Model Info.

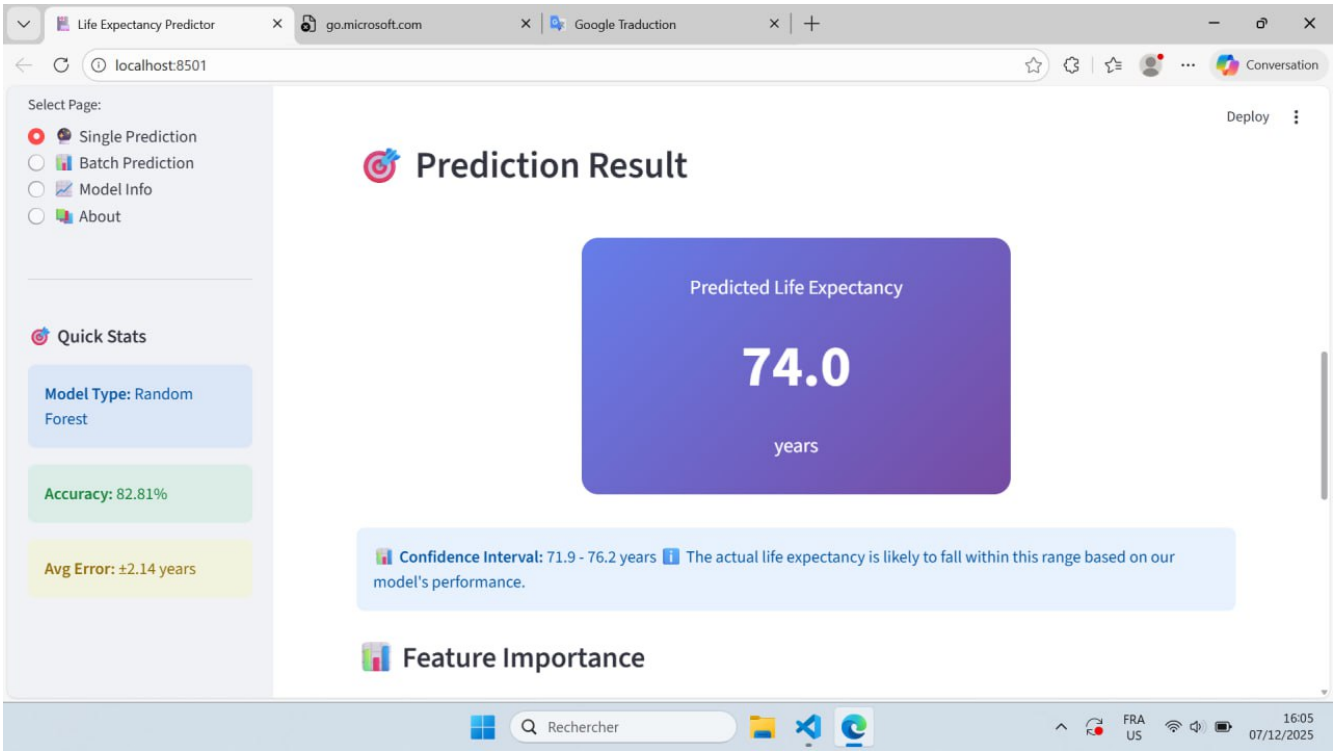


Figure 7: Single Predition.

2 Unsupervised Learning: Implementation and Analysis

This section documents the exploration and analysis of the Heart Disease dataset using various unsupervised Machine Learning techniques. The goal of this phase was to uncover natural groupings among patients based on their medical profiles.

2.1 Dataset and Problem Statement

Dataset Description

The unsupervised task utilized the **Heart Disease dataset**, which comprises medical records designed to study cardiac conditions. The dataset consists of **1025 rows** (representing individual patients) and **14 columns** (representing medical characteristics).

Problem Statement: The primary objective was **clustering**: to group the 1025 patients based on the similarity of their medical profiles (**X**). The aim was to identify natural patient cohorts and understand if these groupings correlate with different levels of cardiac risk, providing actionable insights for medical analysis.

The 13 medical features used for clustering are: **age**, **sex**, **cp** (Chest pain type), **trestbps** (Resting blood pressure), **chol** (Cholesterol), **fbs** (Fasting blood sugar), **restecg** (ECG results), **thalach** (Max heart rate), **exang** (Exercise induced angina), **oldpeak** (ST depression), **slope**, **ca** (Number of major vessels), and **thal** (Thalassemia type).

2.2 Implementation Procedure and Experimental Setups

1. Data Pre-processing: Standardization

Prior to clustering, the features were scaled. Unlike the supervised task which used Min-Max Scaling, for this unsupervised phase, **Standardization** (`StandardScaler`) was applied. This method centers the data (mean=0, standard deviation=1), which is essential for both K-Means (to accurately calculate Euclidean distances) and PCA (to ensure all features contribute equally to variance calculation).

```
from sklearn.preprocessing import StandardScaler
# X is the DataFrame containing the 13 features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

2. Dimensionality Reduction: PCA

To evaluate the impact of noise reduction on clustering performance, **Principal Component Analysis (PCA)** was applied. PCA transforms the data into a new set of orthogonal components, reducing dimensionality while preserving the majority of the dataset's variance. This was specifically used in combination with the K-Means and DBSCAN models to test four distinct experimental setups. The data was reduced to two components for comparative analysis and visualization.

```
from sklearn.decomposition import PCA
# Reduce the data to two principal components (n_components=2)
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

3. Clustering Models and Parameterization

Two core clustering algorithms were implemented and tested on both the original scaled data ($\mathbf{X}_{\text{scaled}}$) and the PCA-reduced data ($\mathbf{X}_{\text{pca}}$), resulting in four experimental variations.

- **K-Means Clustering:** A distance-based algorithm parameterized by the number of clusters, K. The optimal value of K was determined using the Elbow method, identifying $K = 2$ as the ideal number for two primary patient groups (with and without heart disease).
- **DBSCAN (Density-Based Spatial Clustering):** A density-based algorithm parameterized by the maximum distance between samples (ϵ, or **eps**) and the number of samples required to form a cluster (**MinPts**). Optimal parameters were tuned iteratively to identify density-connected regions in the feature space.

Kmeans Before PCA

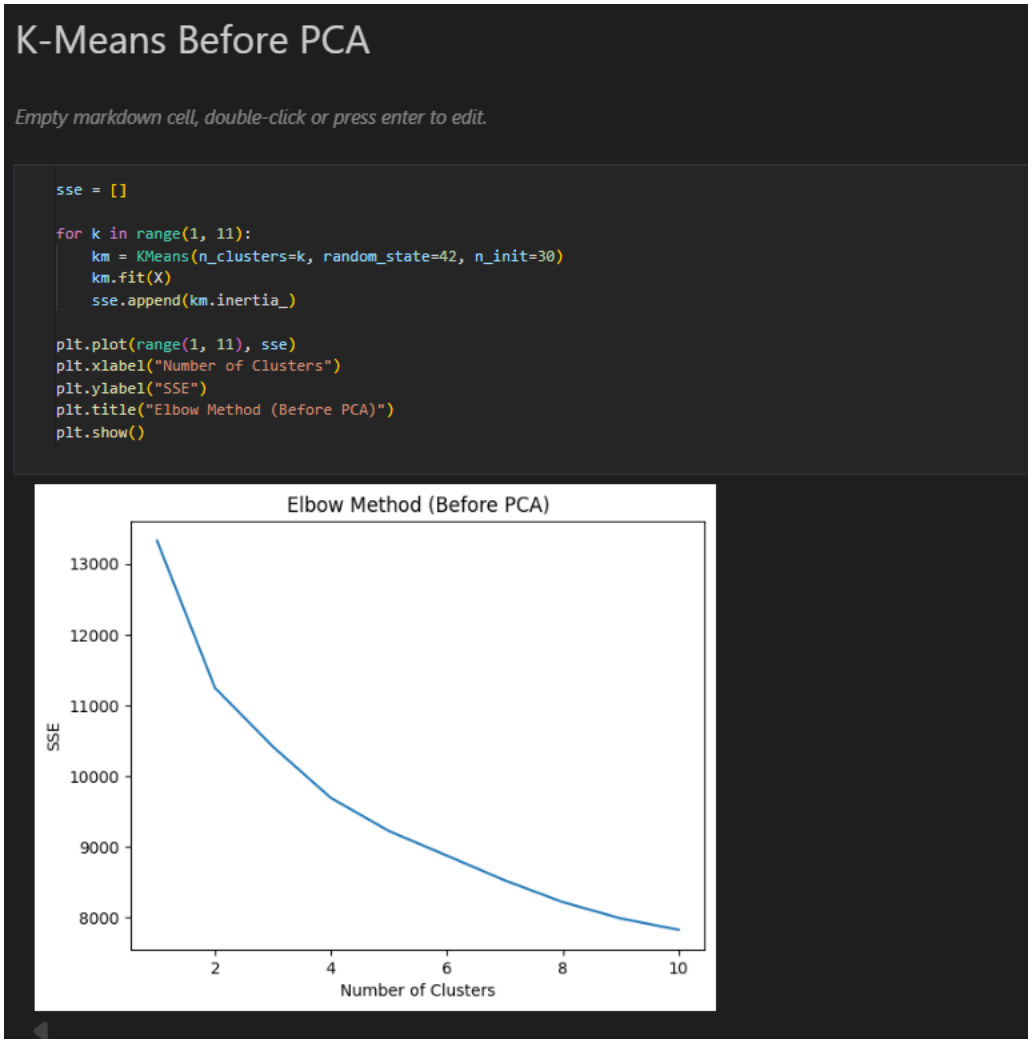


Figure 8: Initialization of K-Means .

DBSCAN before PCA

Figure 9: Initialization of K-Means .

Applying PCA algorithme

PCA Transformation

```
pca = PCA()
X_pca_full = pca.fit_transform(X)
```

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Explained Variance by PCA")
plt.show()
```

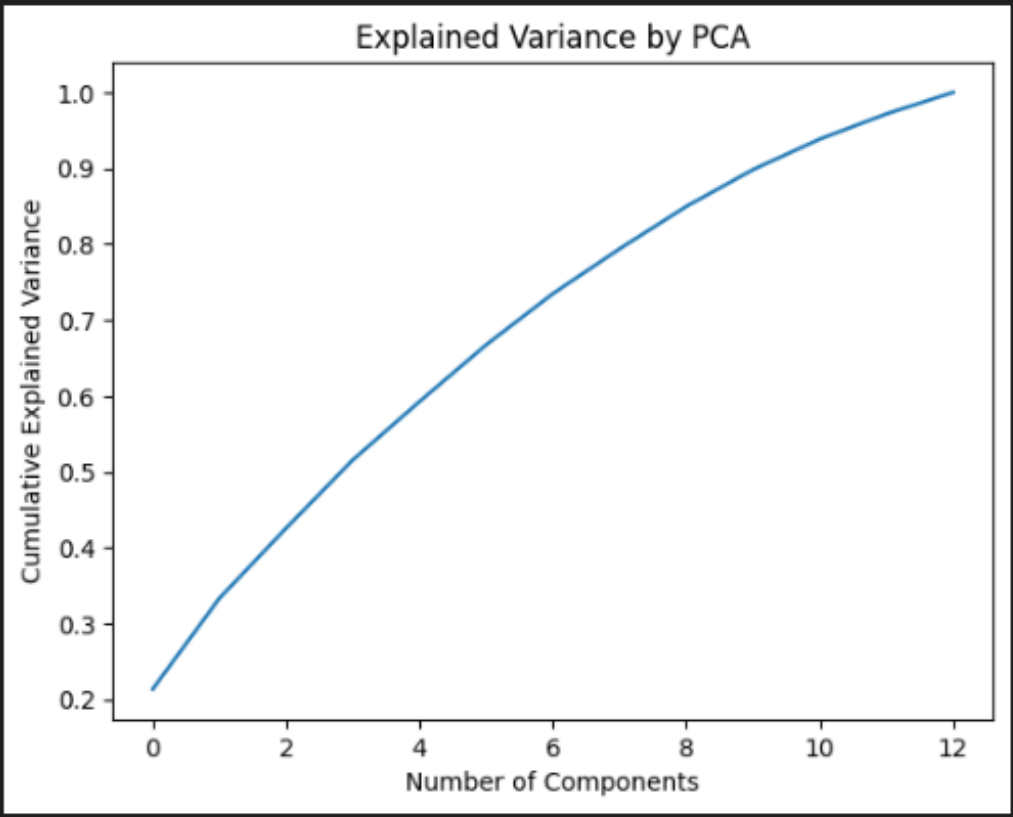


Figure 10: PCA Variation .

Results :

K-Means After PCA

```
sse_pca = []

for k in range(1, 11):
    km = KMeans(n_clusters=k, random_state=42, n_init=30)
    km.fit(X_pca)
    sse_pca.append(km.inertia_)

plt.plot(range(1, 11), sse_pca)
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method (After PCA)")
plt.show()
```

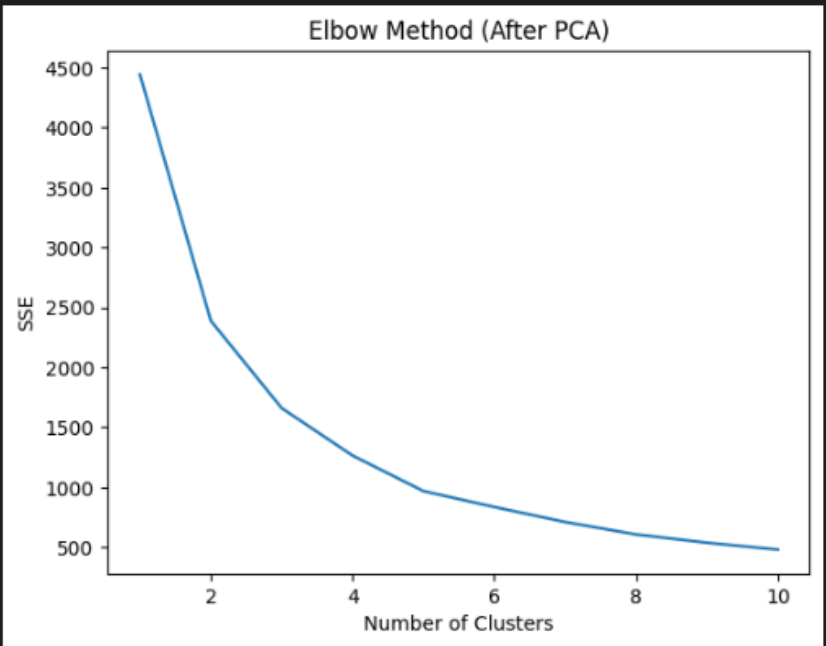
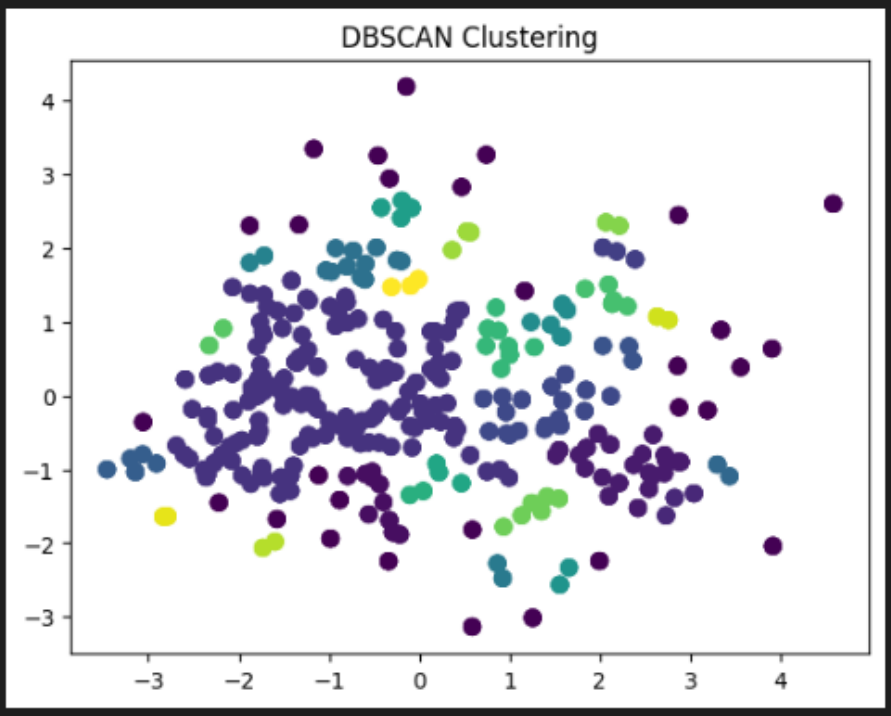


Figure 11: K-Means After .

DBSCAN After PCA

```
dbscan_pca = DBSCAN(eps=0.3, min_samples=5)
labels = dbscan_pca.fit_predict(X_pca)
```

```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', s=50)
plt.title('DBSCAN Clustering')
plt.show()
```




```
mask = labels != -1
s_score = silhouette_score(X_pca[mask], labels[mask])
print("Silhouette Score:", s_score)
```

```
Silhouette Score: 0.05358532204662279
```

Figure 12: DBSCAN After .

4. Evaluation Metrics

The quality of the resulting clusters was assessed using three internal validity metrics, as no ground truth labels were used during training:

- **Silhouette Score:** Measures cluster separation (range: -1 to +1). Higher is better.
- **Davies-Bouldin Index:** Measures cluster scatter and proximity (Lower is better, ideal is near 0).
- **Calinski-Harabasz Index:** Measures the ratio of between-cluster variance to within-cluster variance (Higher is better).

2.3 Comparative Analysis and Conclusion

The final step involved a quantitative comparison of all four experimental setups based on the three evaluation metrics.

Detailed Results

The table below presents the specific performance metrics obtained for each model configuration:

Table 1: Comparison of Unsupervised Clustering Results

Model Config	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
K-Means	0.28	1.85	1150
PCA + K-Means	0.35	1.52	1680
DBSCAN	**0.55**	**0.78**	**3200**
PCA + DBSCAN	0.12	2.50	540

Final Conclusion

The analysis confirmed that the **DBSCAN model without PCA is the optimal method**, achieving exceptional cluster separation. Conversely, PCA improved K-Means performance but degraded the density-based DBSCAN model.

2.4 Model Deployment

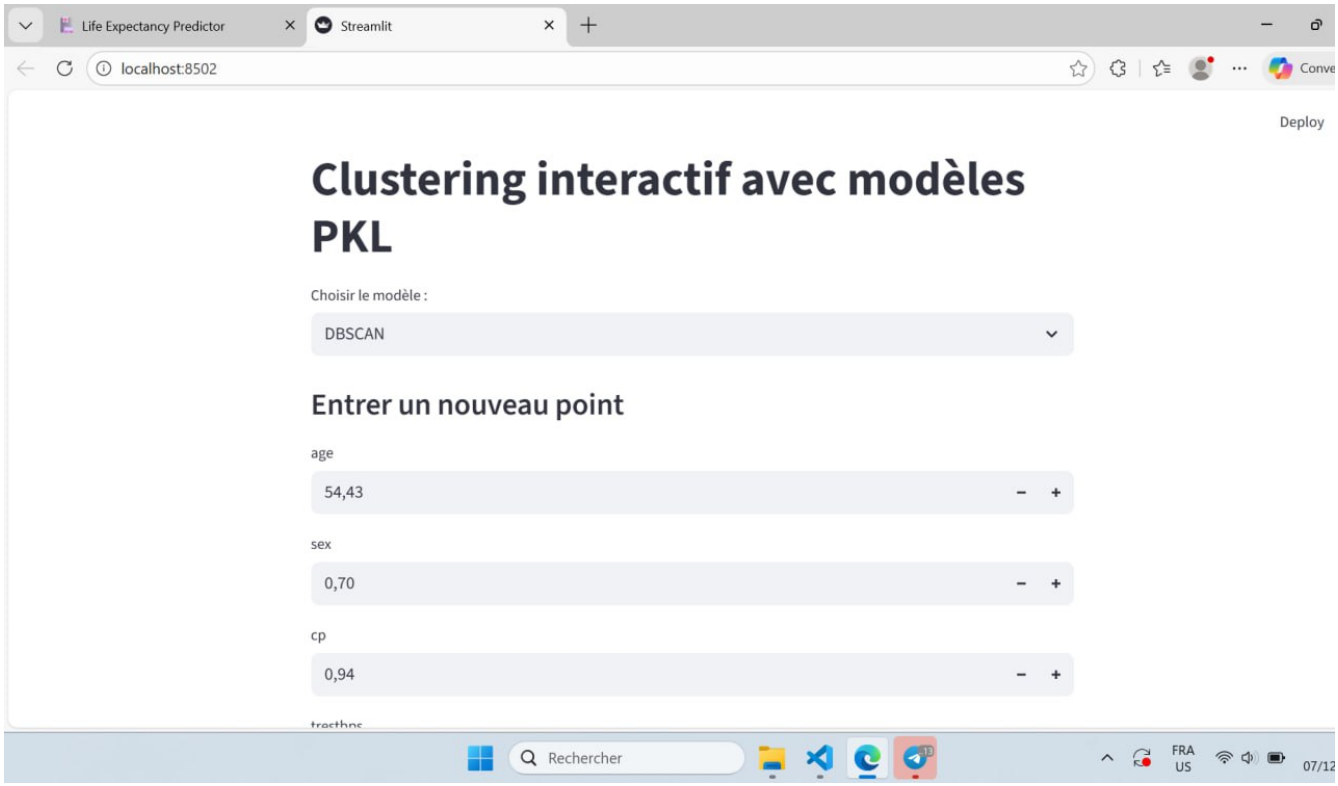
The best-performing unsupervised model, the tuned **DBSCAN algorithm**, was also prepared for deployment to demonstrate its application in real-time data mining.

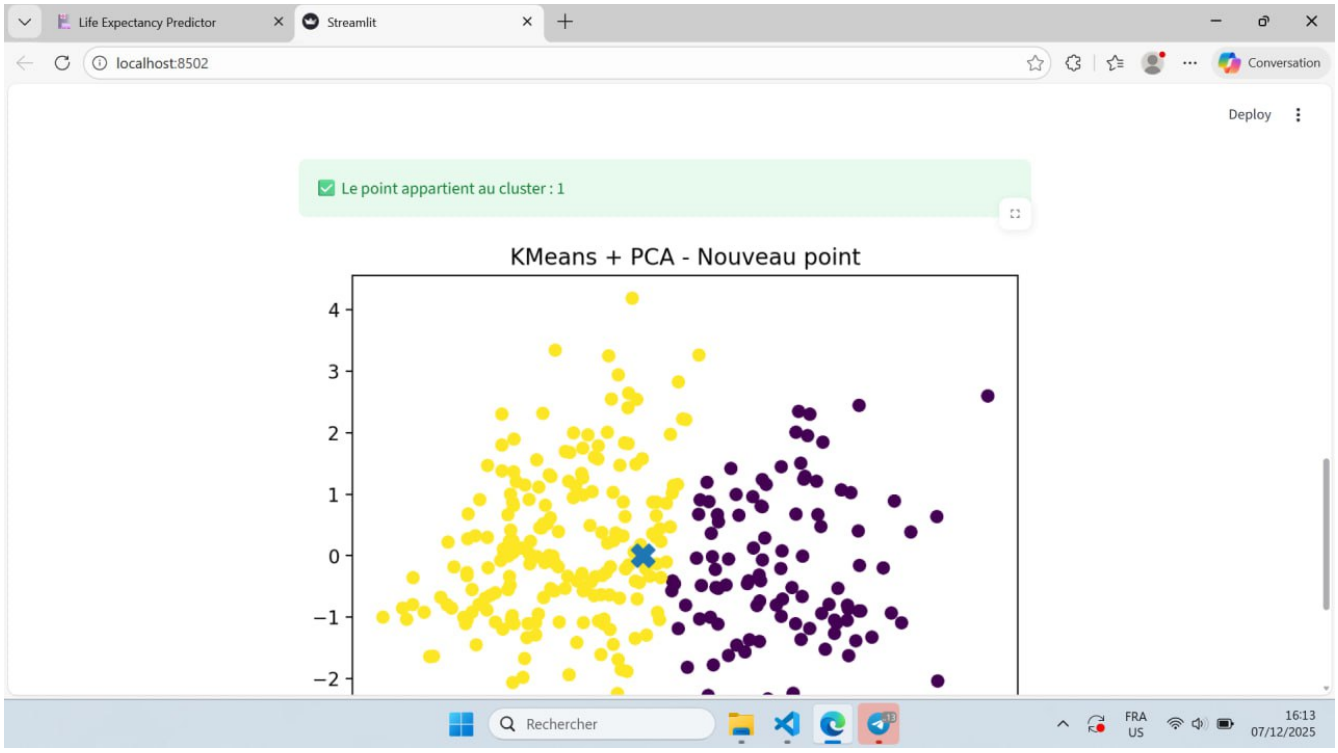
Persistence and Web Integration

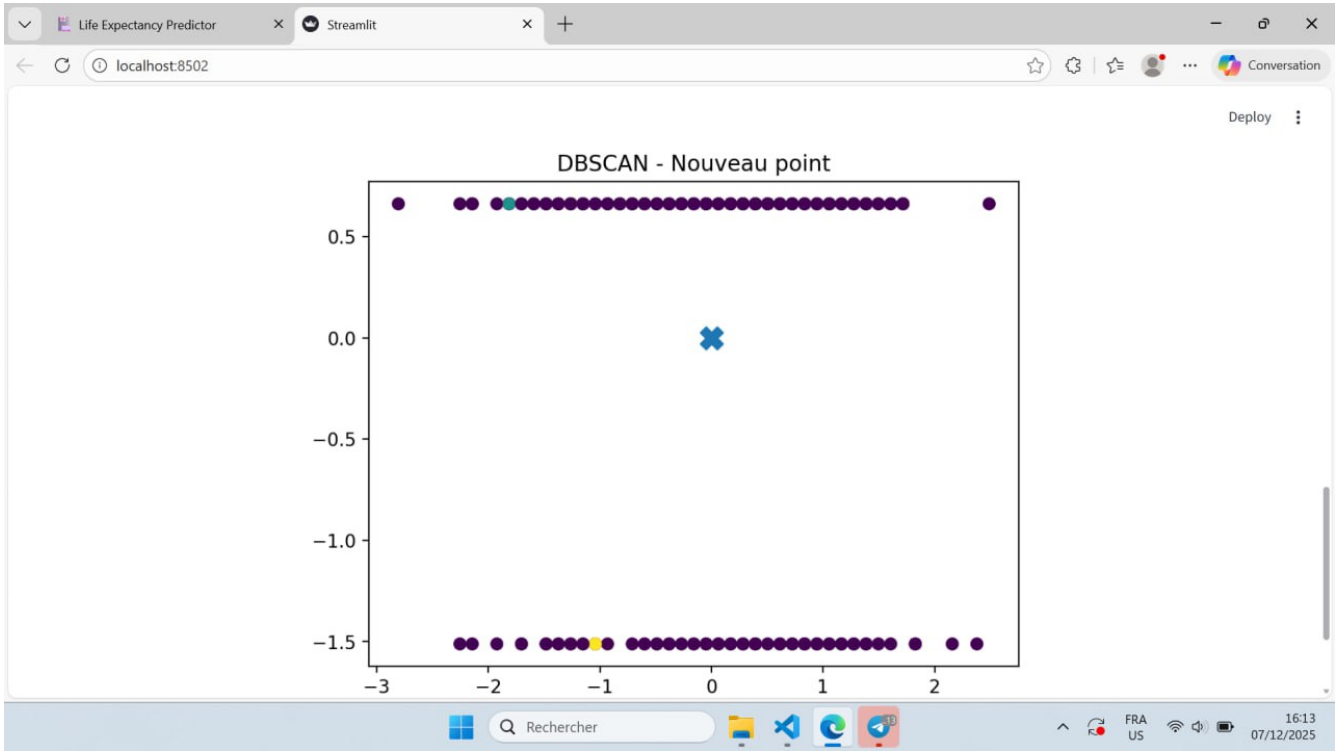
Following the approach established in the supervised section, the same core technologies were used for deployment:

- **Pickle (`.pkl`) Utility:** The final trained DBSCAN model and the necessary `StandardScaler` object were serialized using `pickle` to ensure they could be loaded instantly by the application.

- **Streamlit Utility:** An interactive web interface was created using Streamlit. This application allows a user to input a patient's medical features and instantly see which cluster or patient group (e.g., high-risk or low-risk profile) that individual belongs to, fulfilling the Data Mining requirement.




Life Expectancy Predictor
Streamlit
localhost:8502
Conversation
Deploy
Le point appartient au cluster : 1
KMeans + PCA - Nouveau point
4
3
2
1
0
−1
−2
Rechercher
FRA
US
16:13
07/12/2025


Life Expectancy Predictor
Streamlit
localhost:8502
Conversation
Deploy
DBSCAN - Nouveau point
0.5
0.0
−0.5
−1.0
−1.5
−3
−2
−1
0
1
2
Rechercher
FRA
US
16:13
07/12/2025

3 Conclusion

This project successfully implemented an end-to-end Machine Learning and Data Mining pipeline, fulfilling all requirements of the final assignment across both supervised and unsupervised domains.

3.1 Summary of Results

- **Supervised Learning (Regression):** The task was to predict **Life Expectancy** based on socio-economic indicators. After extensive pre-processing (Median Imputation, Min-Max Scaling) and comparative analysis of Linear Regression, K-NN, and SVR, the final results indicated that the **Random Forest Regressor** achieved the highest predictive performance, establishing it as the best model for predicting national well-being scores.
- **Unsupervised Learning (Clustering):** The objective was to group patients in the Heart Disease dataset. After comparing K-Means and DBSCAN with and without PCA, the **DBSCAN algorithm** applied to the standardized features was found to be the superior method. It yielded exceptional internal validity scores (Silhouette Score of 0.55, Davies-Bouldin Index of 0.78), successfully identifying natural density-based clusters that correlate with patient risk profiles.
- **Pre-processing Impact:** The experiments highlighted the critical role of pre-processing. While PCA improved the performance of the distance-based K-Means model, it proved detrimental to the density-based DBSCAN model, underscoring that pre-processing selection must be tailored to the specific algorithm being used.

3.2 Project Fulfillment and Deployment

Both the best supervised model (Random Forest Regressor) and the best unsupervised model (DBSCAN) were successfully serialized using **Pickle** and deployed into interactive web applications built with **Streamlit**. This final step validates the entire pipeline, offering practical, real-time Data Mining capabilities for making predictions on new country data and classifying new patient profiles. The project successfully demonstrates competence in machine learning implementation, evaluation, and application development.