**Project 1: Healthcare Data Analysis using Python**

**Objective**

To analyze a healthcare dataset in order to uncover trends and anomalies in patient demographics, admission rates, treatment effectiveness, and discharge patterns. The goal is to derive insights that can enhance operational efficiency and improve patient care.

**Tools and Libraries Used**

* **Pandas**: Data manipulation and analysis.
* **NumPy**: Numerical operations.
* **Matplotlib & Seaborn**: Data visualization.
* **Scikit-learn**: Machine learning models and preprocessing.
* **XGBoost**: Advanced gradient boosting model.

**Step-by-Step Process**

**Step 1: Import Libraries**

Essential libraries for data handling, visualization, and modeling are imported.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.svm import SVC

from xgboost import XGBClassifier

**Step 2: Load the Dataset**

The dataset is read from a CSV file into a Pandas DataFrame.

df = pd.read\_csv('PROJECT-1 (DATA SET).csv')

**Step 3: Data Cleaning**

* **Remove rows with more than 20% missing values**.
* **Forward-fill remaining missing values**.

df.dropna(thresh=0.8 \* len(df.columns), inplace=True)

df.fillna(method='ffill', inplace=True)

**Step 4: Encode Categorical Data**

All object-type columns are encoded into numeric form using LabelEncoder.

for col in df.select\_dtypes(include='object').columns:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col].astype(str))

**Step 5: Exploratory Data Analysis (EDA)**

Basic statistics and a heatmap to understand feature correlations.

print(df.describe())

sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

**Step 6: Feature Selection**

Separating features (X) and target variable (y).

X = df.drop('Readmission', axis=1)

y = df['Readmission']

**Step 7: Train/Test Split**

Splitting data into training and testing sets with 80-20 ratio.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step 8: Feature Scaling**

Standardizing the features to have zero mean and unit variance.

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**Step 9: Train and Evaluate Models**

Multiple models are trained and evaluated using accuracy and classification reports.

models = {

'Logistic Regression': LogisticRegression(max\_iter=1000),

'Random Forest': RandomForestClassifier(n\_estimators=100),

'SVM': SVC(),

'Gradient Boosting': GradientBoostingClassifier(),

'XGBoost': XGBClassifier(use\_label\_encoder=False, eval\_metric='mlogloss')

}

results = {}

for name, model in models.items():

model.fit(X\_train\_scaled, y\_train)

y\_pred = model.predict(X\_test\_scaled)

acc = accuracy\_score(y\_test, y\_pred)

results[name] = acc

print(f"\n{name} Accuracy: {acc:.4f}")

print(classification\_report(y\_test, y\_pred))

**Step 10: Compare Model Performance**

Visualization to identify the best-performing model.

best\_model = max(results, key=results.get)

print("\n\U0001F3C6 Best Performing Model:", best\_model)

plt.figure(figsize=(8, 5))

sns.barplot(x=list(results.keys()), y=list(results.values()))

plt.ylabel("Accuracy Score")

plt.xticks(rotation=45)

plt.title("Model Comparison")

plt.tight\_layout()

plt.show()

**Conclusion**

This project followed a full pipeline from loading and cleaning healthcare data to training and evaluating multiple machine learning models. The approach provides actionable insights into patient readmissions and allows healthcare providers to identify areas of improvement.