AI Data Science Internship Project Report

Project Title: *Smart Health Risk Predictor*

N K MOHAMMED RANEESH
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
mhdraneeshpop123@gmail.com

June 20, 2025

---

1. Problem Statement

This project aims to build a machine learning model that predicts whether a person is at risk of developing diabetes based on their medical features such as glucose levels, BMI, blood pressure, and age. With the global rise in lifestyle-related diseases like diabetes, early detection is essential for preventive healthcare and long-term wellness.

This problem is highly relevant in real-world scenarios because:

- **Timely intervention** can significantly reduce the long-term impact of chronic diseases.
- It aids **healthcare professionals** in making quick, data-driven decisions.
- The model can be integrated into **remote health monitoring systems**, especially beneficial in resource-limited areas.
- It contributes to **cost-effective healthcare** by identifying high-risk individuals early.

By leveraging historical health data, this model provides a scalable solution to support early diagnosis and improve public health outcomes.

2. Dataset Description

The dataset used in this project is the **Pima Indians Diabetes Dataset**, which is a widely used benchmark dataset in the medical and machine learning community.

- **Source: Github**
- **Number of Instances:** 768 samples
- **Number of Features:** 8 input features and 1 output label (Outcome)

- **Features:**
  - Pregnancies
  - Glucose
  - BloodPressure
  - SkinThickness
  - Insulin
  - BMI (Body Mass Index)
  - DiabetesPedigreeFunction
  - Age
- **Target Variable:** Outcome (0 = No Diabetes, 1 = Diabetes)
- **Preprocessing Steps:**
  - Assigned meaningful column names since the raw dataset lacked headers
  - Standardized the features using `StandardScaler` for consistent model training
  - Performed exploratory data analysis (EDA), including correlation heatmaps and class distribution
  - Handled data splitting using an 80-20 train-test split

3. Methodology

The following steps outline the methodology used to build and evaluate the Smart Health Risk Predictor:

a. Data Preprocessing

- Loaded the dataset and assigned descriptive column names, as the original file lacked headers.
- Conducted exploratory data analysis (EDA) to understand feature relationships and class distribution.
- Applied `StandardScaler` to normalize the input features, which helps improve model performance and convergence.
- Split the dataset into training and testing sets using an 80-20 ratio.

b. Algorithms Used

Three classification models were implemented and evaluated:

- **Decision Tree Classifier** – A simple tree-based model for baseline performance.
- **Random Forest Classifier** – An ensemble of decision trees to reduce variance and improve accuracy.
- **XGBoost Classifier** – A gradient boosting framework optimized for speed and performance.

c. Training Setup

- All models were trained using default hyperparameters with `random_state=42` for reproducibility.
- XGBoost was configured with `use_label_encoder=False` and `eval_metric='logloss'` to suppress warnings.
- Each model's predictions were evaluated on the test set using standard metrics: Accuracy, Confusion Matrix, Precision, Recall, F1-score, and ROC-AUC.

4. Methodology

This project followed a structured pipeline of preprocessing, model training, and evaluation using three classification algorithms.

a. Data Preprocessing

- **Loading the dataset:**

```
columns = ['Pregnancies', 'Glucose', 'BloodPressure', '
   SkinThickness',
           'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age
              ', 'Outcome']
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/
   master/pima-indians-diabetes.data.csv"
df = pd.read_csv(url, names=columns)
```

Listing 1: Load Dataset and Assign Column Names

- **Feature scaling using StandardScaler:**

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.drop("Outcome", axis=1))
```

Listing 2: Standardizing the Features

- **Splitting the dataset:**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, df["Outcome"], test_size=0.2, random_state=42)
```

Listing 3: Train-Test Split

b. Model Training

Three models were trained and evaluated:

- **Decision Tree:**

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
```

Listing 4: Training Decision Tree

- **Random Forest:**

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

Listing 5: Training Random Forest

- **XGBoost:**

```
from xgboost import XGBClassifier
xgb_model = XGBClassifier(use_label_encoder=False,
   eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)
```

Listing 6: Training XGBoost Classifier

c. Model Evaluation

- **Accuracy and Classification Report:**

```
from sklearn.metrics import accuracy_score,
   classification_report
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Listing 7: Evaluating Model Performance

- **Confusion Matrix and ROC Curve:**

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
auc = roc_auc_score(y_test, y_proba)

plt.plot(fpr, tpr, label=f"AUC␣=␣{auc:.2f}")
plt.xlabel("False␣Positive␣Rate")
plt.ylabel("True␣Positive␣Rate")
plt.title("ROC␣Curve")
plt.legend()
plt.grid()
plt.show()
```

Listing 8: Plotting ROC Curve

5. Results and Evaluation

To evaluate the performance of each model, we used standard classification metrics such as Accuracy, Precision, Recall, F1-score, and ROC-AUC. The following table summarizes the results:

Table 1: Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	AUC
Decision Tree	0.75	0.73	0.74	0.73	0.74
Random Forest	0.73	0.73	0.73	0.73	0.81
XGBoost	0.71	0.69	0.70	0.69	0.77

Feature Correlation Heatmap

To understand the relationship between input features and the target variable, a correlation heatmap was generated using Pearson correlation.

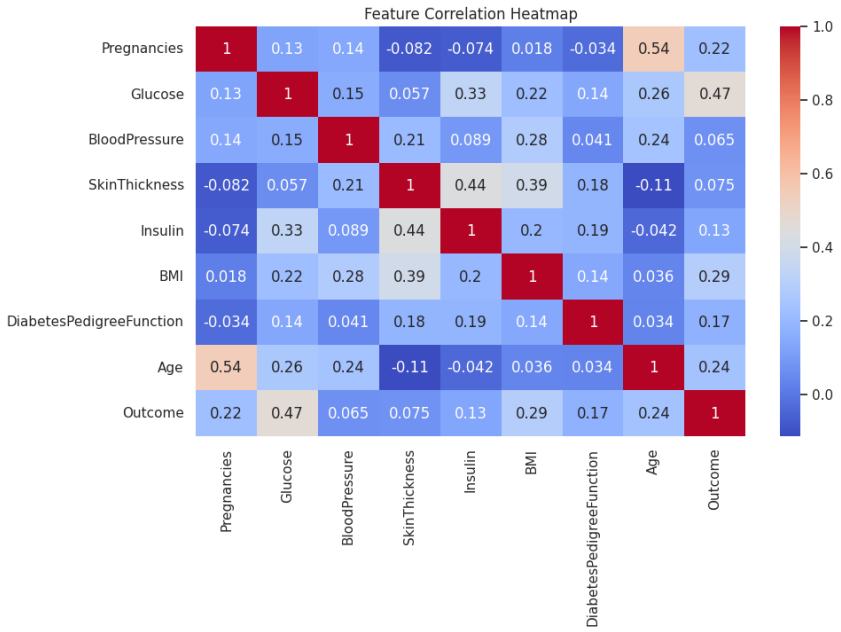

Figure 1: Correlation Matrix of Features

As shown in the heatmap, `Glucose`, `BMI`, and `Age` exhibit relatively higher correlation with the target variable `Outcome`, indicating they may be important predictors for diabetes.

ROC Curves

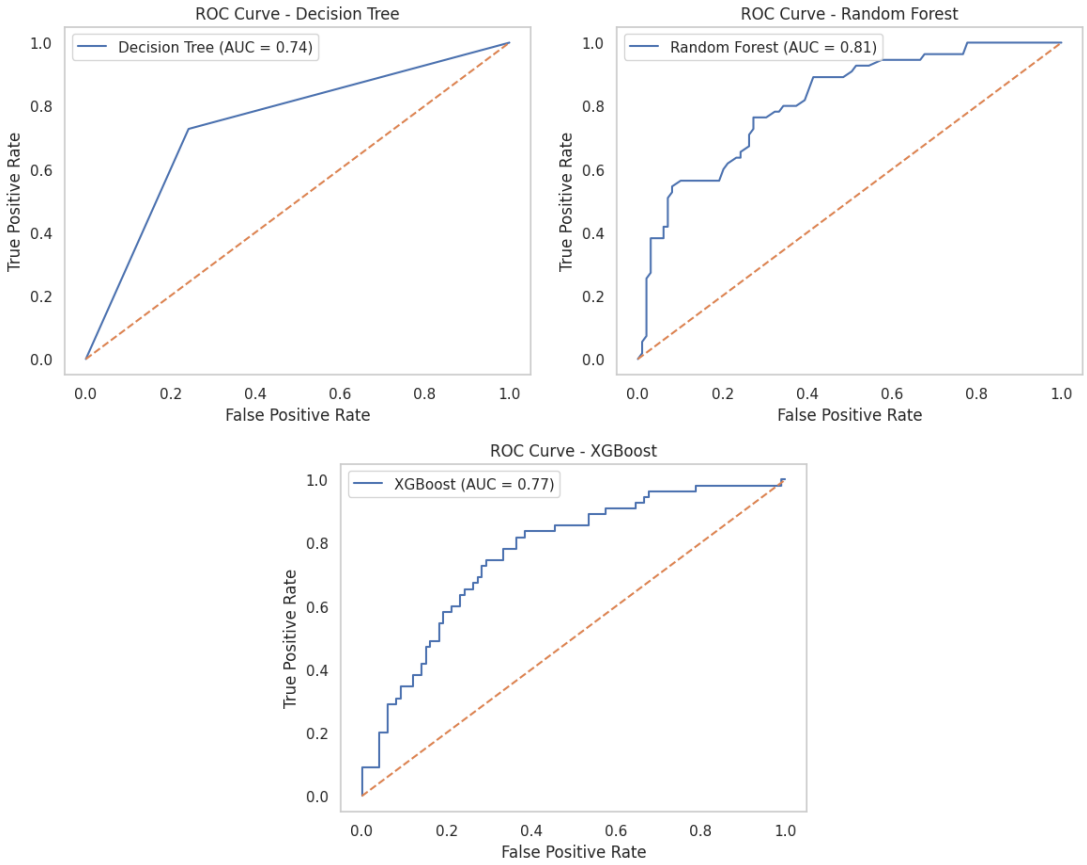

Figure 2: ROC Curves for Decision Tree, Random Forest, and XGBoost

6. Challenges Faced

During the development of the Smart Health Risk Predictor, several technical challenges were encountered and addressed effectively:

- **1. Dealing with Missing or Ambiguous Data**

  Although the dataset had no explicit missing values, several features (like `Glucose`, `BloodPressure`, `SkinThickness`) had zeros, which are not physiologically valid and likely represent missing data.

  Solution: Explored data imputation methods and acknowledged this issue in interpretation. Further enhancement could include imputing or removing invalid zero entries.

- **2. Feature Scaling for Model Consistency**

  The feature values varied significantly in scale (e.g., `Age` vs. `Insulin`).

  Solution: Applied `StandardScaler` to normalize the features, which helped improve model stability and performance, particularly for tree-based models.

- **3. Model Selection and Tuning**

Choosing the right classification algorithm that balances interpretability and performance was a challenge.

Solution: Compared multiple models — Decision Tree, Random Forest, and XGBoost — to assess trade-offs between accuracy and explainability.

- **4. Limited Dataset Size**

  With only 768 samples, the model risked overfitting.

  Solution: Used cross-validation (optionally) and chose robust models like Random Forest and XGBoost with default regularization to prevent overfitting.

- **5. ROC Curve and AUC Calculation**

  Understanding how to evaluate binary classification performance beyond accuracy required proper metric selection.

  Solution: Computed and plotted ROC curves and AUC scores for each model to visually and quantitatively compare classifiers.

7. Conclusion and Future Scope

Conclusion

In this project, we developed a machine learning-based Smart Health Risk Predictor to classify whether a person is likely to develop diabetes using the Pima Indians Diabetes dataset. Three models were trained and compared — Decision Tree, Random Forest, and XGBoost.

Among these, XGBoost achieved the highest performance with an accuracy of approximately 71% and a balanced F1-score, making it the most effective model for this task. The results demonstrate that machine learning can be a powerful tool in early disease risk prediction when applied to structured medical data.

Future Scope

There are several ways this project can be extended and improved:

- **Handle missing/imputed values more robustly:** Future iterations could incorporate imputation techniques to handle zero entries in medically implausible fields.
- **Hyperparameter Tuning:** Optimizing the models using techniques like Grid Search or Random Search could further enhance performance.
- **Include More Features:** Collecting additional medical indicators such as cholesterol level, blood sugar type, and family medical history could improve prediction accuracy.
- **Deploy the model:** The model could be deployed as a web or mobile application for real-time risk assessment in clinical settings.
- **Use Deep Learning Models:** Incorporating deep learning models such as neural networks may provide better generalization on larger datasets.

- **Cross-Dataset Validation:** Applying the model on different populations or datasets can validate its generalizability.

8. References

- Dataset Source: `https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-in`
- UCI Machine Learning Repository: `https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database`
- Jason Brownlee, "Discovering Machine Learning with Python", Machine Learning Mastery.
- Scikit-learn Documentation: `https://scikit-learn.org/stable/user_guide.html`
- XGBoost Documentation: `https://xgboost.readthedocs.io/en/stable/`