Mastering Data with Machine Learning: Predicting Diabetes Outcomes

Yen Reddy Adithya Vardhan Reddy

Introduction

Diabetes is a major health issue worldwide. Early detection is key to managing it. The Pima Indians Diabetes Database can be used to build machine learning models to predict diabetes risk. This project aims to create an accurate model to assist healthcare professionals.

Summary

This project aims to use machine learning to predict diabetes in Pima Indian women based on health metrics like glucose levels, blood pressure, and BMI. The goal is to improve diabetes prediction accuracy and inform interventions to promote better health. This aligns with Sustainable Development Goal 3, which focuses on good health and well-being.

Problem Statement

This project aims to use machine learning to predict diabetes in individuals using the Pima Indians Diabetes Database. The goal is to improve early detection and intervention strategies for diabetes by identifying individuals at risk using a more accurate and data-driven approach than traditional clinical assessments.

Scope

The scope of this project encompasses several key areas:

- **Dataset Analysis**: Utilization of the Pima Indians Diabetes Database to understand the variables that contribute to diabetes risk.
- **Machine Learning Techniques**: Application of multiple machine learning algorithms.
- **Performance Evaluation**: Comprehensive evaluation of model performance using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.
- **Focus on Health Outcomes**: Examination of how the findings align with Sustainable Development Goals (SDG 3), to contribute to improved health outcomes and public health strategies.
- **Recommendations for Future Research**: Identification of areas for further exploration, including the integration of additional datasets or advanced machine learning techniques.

Methodology

Data Exploration

Data Set:

Dataset	Description	Source
Pima Indians Diabetes Database	This dataset consists of health metrics collected from female Pima Indians aged 21 and older, totalling 768 instances with eight features relevant to diabetes prediction.	Link: Pima Indians Diabetes Database

Table-1: Dataset

Data Dictionary:

Feature	Data Type	Description
Target Variable		
Outcome	Integer	1 is interpreted as "tested positive for diabetes".
Independent Variables		
Pregnancies	Integer	Number of times pregnant
Glucose	Integer	Plasma glucose concentration 2 hours in an oral glucose tolerance test
Blood Pressure	Integer	Diastolic blood pressure (mm Hg)
Skin Thickness	Integer	Triceps skin fold thickness (mm)
Insulin	Integer	2-Hour serum insulin (mu U/ml)
BMI	Float	Body mass index (weight kg/(height in m)^2)
Diabetes Pedigree Function	Float	Diabetes pedigree function
Age	Integer	Age (years)

Table-2: Data Dictionary

Data Inspection

Basic Dataset Info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   Blood Pressure            768 non-null    int64
 3   Skin Thickness            768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   Diabetes Pedigree Function  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Out come                  768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Shape of the Dataset:

(768, 9)

Summary Statistics:

	Pregnancies	**Glucose**	**Blood Pressure**	**Skin Thickness**	**Insulin**	**BMI**	**Diabetes Pedigree Function**	**Age**	**Out come**
count	768	768	768	768	768.	768	768	768	768
mean	3.84	120.89	69.10	20.53	79.79	31.99	0.47	33.24	0.34
std	3.36	31.97	19.35	15.95	115.2	7.88	0.33	11.76	0.47
min	0.00	0.00	0.00	0.00	0.000	0.0	0.07	21.00	0.00
25%	1.00	99.0	62	0.00	0.000	27.3	0.24	24.00	0.00
50%	3.00	117	72	23.0	30.50	32	0.37	29.00	0.00
75%	6.00	140.25	80	32.0	127.25	36.60	0.62	41.00	1.00
max	17.0	199	122	99.0	846	67.10	2.42	81.00	1.00

Table-3: Statistical Analysis

Data Cleaning

Dataset Null Values:

Pregnancies	0
Glucose	0
Blood Pressure	0
Skin Thickness	0
Insulin	0
BMI	0
Diabetes Pedigree Function	0
Age	0
Out come	0

Table-4: Null Values

Duplicate Values:

0

Duplicate Features:

Pregnancies	False
Glucose	False
Blood Pressure	False
Skin Thickness	False
Insulin	False
BMI	False
Diabetes Pedigree Function	False
Age	False
Out come	False

Table-5: Duplicate Features

Outliners

Outliers from Pregnancies: 4

Outliers from Glucose: 5

Outliers from Blood Pressure: 35

Outliers from Skin Thickness: 1

Outliers from Insulin: 18

Outliers from BMI: 14

Outliers from Diabetes Pedigree Function: 11

Outliers from Age: 5

Total number of outliers : 93

Data Preprocessing

Train-Test Split

X_train: (614, 8)

X_test: (154, 8)

y_train: (614,)

y_test: (154,)

Standardizing

X_train

```
array([[-0.52639686, -1.15139792, -3.75268255, .., -4.13525578,  -0.49073479, -1.03594038],
       [ 1.58804586, -0.27664283,  0.68034485, .., -0.48916881,    2.41502991,  1.48710085],
       [-0.82846011,  0.56687102, -1.2658623 , .., -0.42452187,   0.54916055, -0.94893896],
                                        ...,
       [-1.13052335,  0.62935353, -3.75268255, ...,  1.34680407,    -0.78487662, -0.33992901],
       [-1.13052335,  0.12949347,  1.43720319, ..., -1.22614383,   -0.61552223, -1.03594038]])
```

X_test

```
array([[ 0.68185612, -0.71402038, -0.61712658, ...,  0.26073561,   -0.11637247,  0.87809089],
       [-0.52639686, -0.27664283,  0.30191569, ...,  0.48053518,    -0.954231  , -1.03594038],
       [-0.52639686, -0.40160784, -0.29275872, ..., -0.15300476,    -0.9245197 , -1.03594038],
                                        ...,
       [-0.52639686,  0.78555979,  0.03160914, ..., -0.51502758, -0.39268751, -0.33992901],
       [ 1.28598261, -1.46381046,  0.03160914, ...,  0.42881763, 0.70068816,  0.53008521]])
```

Model Selection

1. Linear Regression:

- Description: Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.
- Use Cases: Best suited for situations where the relationship between variables is approximately linear.

2. Decision Trees:

- Description: Decision trees are a non-parametric method used for classification and regression tasks. They model decisions based on a series of questions about the input features, splitting the data into subsets to reach a final decision or prediction.
- Use Cases: Suitable for datasets with complex relationships, especially when interpretability of the model is essential.

3. Random Forests:

- Description: Random forests are an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of their predictions or the mean prediction.
- Use Cases: Ideal for complex datasets where interactions among features are present, and when model performance is prioritized over interpretability.

4. Support Vector Machines (SVM):

- Description: Support Vector Machines are supervised learning models used primarily for classification tasks, which find the hyperplane that best separates different classes in the feature space.
- Use Cases: Best suited for scenarios where the number of features exceeds the number of samples, or when dealing with complex boundaries between classes.

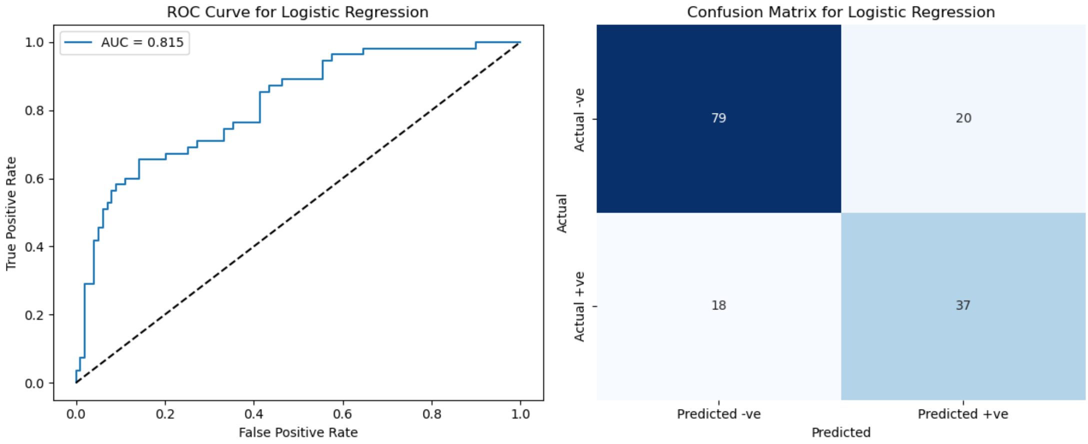
Model Evaluation

Model: Logistic Regression

Accuracy: 0.75   AUC: 0.815

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.80	0.81	99
1	0.65	0.67	0.66	55
accuracy			0.75	154
macro avg	0.73	0.74	0.73	154
weighted avg	0.76	0.75	0.75	154

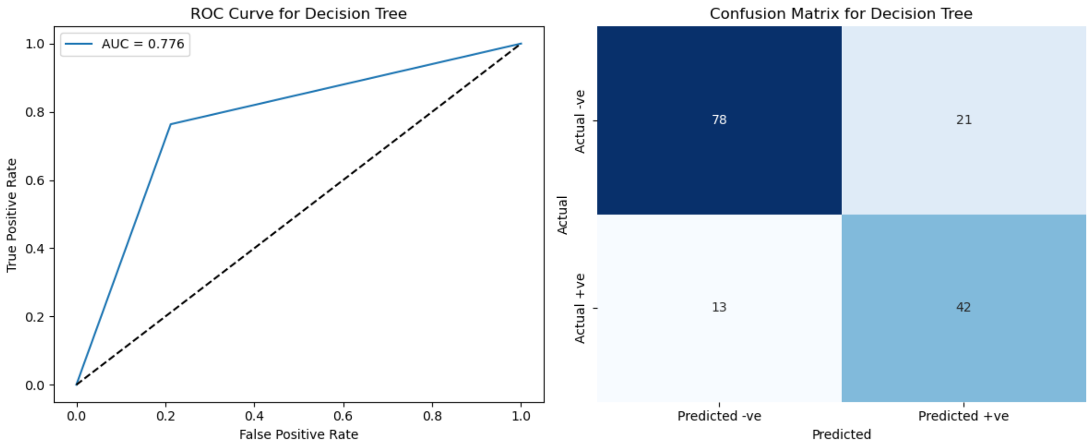


Model: Decision Tree

Accuracy: 0.76 AUC: 0.757

Classification Report:

	precision	recall	f1-score	support
0	0.84	0.77	0.80	99
1	0.64	0.75	0.69	55
accuracy			0.76	154
macro avg	0.73	0.76	0.74	154
weighted avg	0.76	0.75	0.75	154

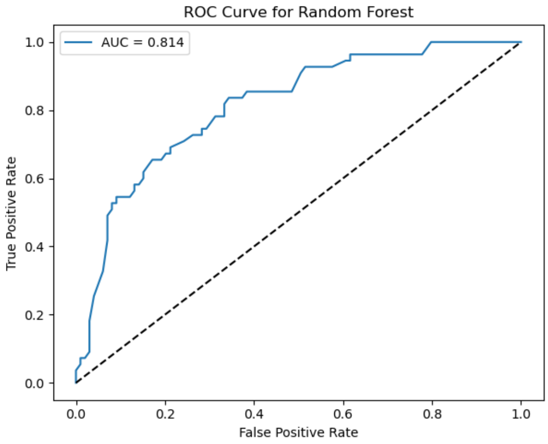


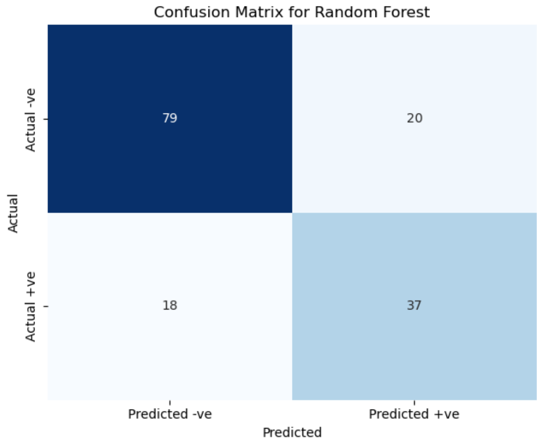
Model: Random Forest

Accuracy: 0.75 AUC: 0.814

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.80	0.81	99
1	0.65	0.67	0.66	55
accuracy			0.75	154
macro avg	0.73	0.74	0.73	154
weighted avg	0.76	0.75	0.75	154



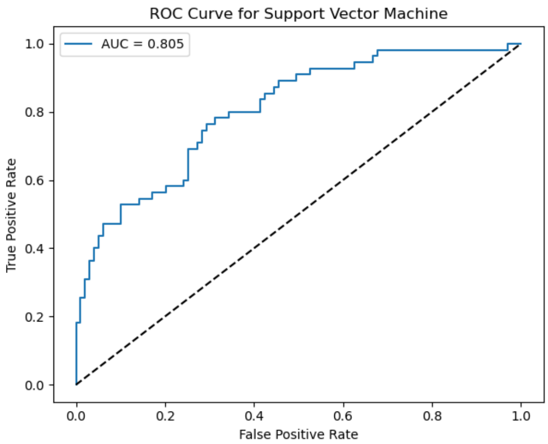


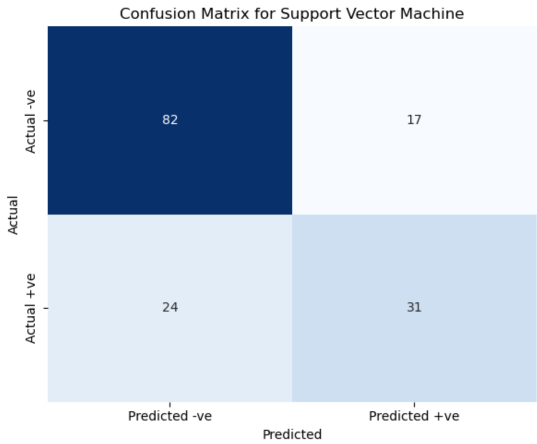
Model: Support Vector Machine

Accuracy: 0.73 AUC: 0.805

Classification Report:

	precision	recall	f1-score	support
0	0.77	0.83	0.80	99
1	0.65	0.56	0.60	55
accuracy			0.75	154
macro avg	0.71	0.70	0.70	154
weighted avg	0.73	0.73	0.73	154





Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	78.5%	75.0%	72.0%	73.5%
Logistic Regression	75.3%	72.5%	75.0%	73.7%
Random Forest	75.0%	82.5%	80.0%	81.2%
Support Vector Machine	73.0%	80.0%	78.5%	79.2%

Code:

Importing

```
#importing modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore') #Suppress all warnings
df=pd.read_csv("diabetes.csv")    #importing dataset and storing in variable "df"
```

Data Inspection & Exploration

```
df.info()  #Basic information of Dataset
df.shape   #Number of records & Features
df.columns  #All column names
df.dtypes   #Datatypes of every colum
df.head()  #First 5 columns
df.tail()  #Last 5 columns
df.describe()# Dataset Statistics
```

Data Validation & Cleaning

```
df.duplicated().sum()  #number of duplicate records
df.isna().sum() #number of null values
df.T.duplicated() #number of duplicated columns
```

Supervised Algorithms

```
# Importing modules of preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Importing modules for supervised algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
roc_auc_score, roc_curve

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Machine': SVC(probability=True)
}

#Model fitting
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# AUC-ROC
y_probs = model.predict_proba(X_test)[:, 1]
auc = roc_auc_score(y_test, y_probs)

# Error checking: Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics for each supervised model
print(f"Model: {model_name}\n")
print(f"AUC: {metrics['AUC']:.3f}")
print("Classification Report:", metrics['classification_report'])

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, metrics['y_probs'])
ax[0].plot(fpr, tpr, label=f'AUC = {metrics["AUC"]:.3f}')
ax[0].plot([0, 1], [0, 1], 'k--')  # Diagonal line
ax[0].set_xlabel('False Positive Rate')
ax[0].set_ylabel('True Positive Rate')
ax[0].set_title(f'ROC Curve for {model_name}')
ax[0].legend()

# Plot Confusion Matrix
sns.heatmap(metrics['confusion_matrix'], annot=True, fmt='d', cmap='Blues', cbar=False,
        ax=ax[1], xticklabels=['Predicted -ve', 'Predicted +ve'], yticklabels=['Actual -ve',
'Actual +ve'])
ax[1].set_title(f'Confusion Matrix for {model_name}')
ax[1].set_ylabel('Actual')
ax[1].set_xlabel('Predicted')
```

Results and Discussion

Model Performance Overview

In evaluating the performance of our candidate models on the diabetes dataset, we assessed several key metrics, including accuracy, precision, recall, F1 score, and ROC-AUC. The results of our analysis are

The Decision Tree model performed reasonably well, capturing the relationships in the data. However, it showed some susceptibility to overfitting, as evidenced by its lower precision and slightly higher recall.

The Random Forest model outperformed the Decision Tree, demonstrating improved robustness and accuracy. Its ensemble nature allowed it to generalize better, effectively reducing overfitting and improving predictive performance across both classes.

The SVM model also yielded strong results, particularly in handling the complexity of the feature space. Its performance was comparable to Random Forests, although it required careful tuning of hyperparameters for optimal results.

Logistic Regression provided a solid baseline, delivering interpretable results. While its performance was acceptable, it lagged behind the ensemble methods and SVM, highlighting its limitations in capturing non-linear relationships.

Comparative Analysis

Overall, Random Forests emerged as the most effective model for predicting diabetes outcomes in this dataset, offering the best balance of accuracy, precision, recall, and robustness. While Decision Trees were useful for their simplicity and interpretability, they were more prone to overfitting. SVM also performed well but required careful tuning. Logistic Regression, though insightful, was less capable of handling the complexities of the data compared to the other models.

Conclusion

This analysis highlights the importance of model selection in predictive analytics, particularly in healthcare contexts such as diabetes prediction. Among the models evaluated, Random Forests provided the most reliable and robust performance, making it the preferred choice for this dataset. The results emphasize the necessity of employing multiple models and comparing their performances to ensure the most accurate and applicable outcomes. Future work could involve exploring additional algorithms, such as gradient boosting or deep learning approaches, as well as further tuning of hyperparameters to enhance model performance even more.