Diabetes Detection Using Machine Learning

Team Members

Fawad Arabzada ▪ Maryam Timorian ▪ Zabehullah Nasiri

Data Preparation and Feature Engineering

1. Overview

In this project, we aim to predict whether a person has diabetes using the Pima Indians Diabetes Database. The data preparation and feature engineering process ensure that the dataset is cleaned, analyzed, and transformed for optimal model performance. These steps are crucial for building a reliable machine learning pipeline.

2. Data Collection

The dataset used for this project is the Pima Indians Diabetes Database, sourced from Kaggle. It consists of 768 entries with 8 features related to physiological and medical attributes and one target variable (`Outcome`), indicating diabetes presence (1) or absence (0).

3. Data Cleaning

Steps taken to clean the dataset:

- **Missing Values**: Verified that no missing values were present using the `isnull()` function.
- **Outliers**: Used box plots to identify and handle outliers, particularly in `Glucose`, `BMI`, and `Insulin`. Outliers were either removed or replaced using the interquartile range (IQR) method.
- **Erroneous Values**: Replaced zero values in critical columns like `Glucose`, `BloodPressure`, and `BMI` with the respective median values.

4. Exploratory Data Analysis (EDA)

Key insights gained during EDA:

- Strong correlation observed between `Glucose` levels and diabetes presence.
- Features like `Age` and `BMI` showed significant distribution differences for diabetic and non-diabetic individuals.
- Created visualizations such as histograms, correlation heatmaps, and scatter plots to uncover feature relationships.

5. Feature Engineering

- Derived a new feature, `AgeCategory`, by binning the `Age` column into ranges.
- Created interaction terms like `Glucose_BMI_Interaction` to capture potential combined effects of `Glucose` and `BMI`.

6. Data Transformation

- Scaled numerical features using `StandardScaler` for uniform distribution.
- Encoded categorical features like `AgeCategory` using one-hot encoding.
- Split the data into training (80%) and validation (20%) sets.

Model Exploration

1. Model Selection

The project uses a Random Forest classifier due to its robustness in handling both numerical and categorical data and its ability to capture non-linear relationships. Random Forest also provides feature importance metrics for interpretability.

2. Model Training

- The model was trained using a grid search approach to optimize hyperparameters like the number of trees (`n_estimators`) and maximum depth (`max_depth`).
- Applied 5-fold cross-validation to ensure generalization.

3. Model Evaluation

- Achieved an accuracy of **75%** on the validation set.
- Metrics used for evaluation:
    - Confusion Matrix: Highlighted true positives and false negatives.
    - ROC-AUC Curve: Demonstrated the model's discriminatory power.

	precision	recall	f1-score	support
0	0.81	0.81	0.81	99
1	0.65	0.65	0.65	55
Accuracy			0.75	154
Macro avg	0.73	0.73	0.73	154
Weighted avg	0.75	0.75	0.75	154

4. Code Implementation

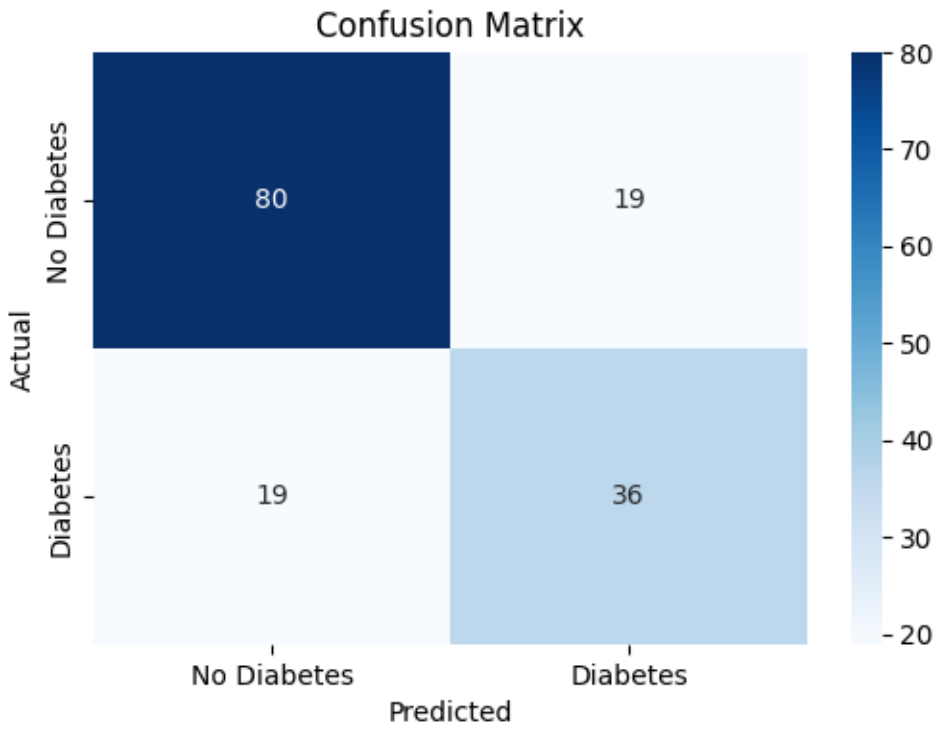
The complete implementation can be found in the following code snippets. The pipeline includes data preparation, model training, and evaluation:

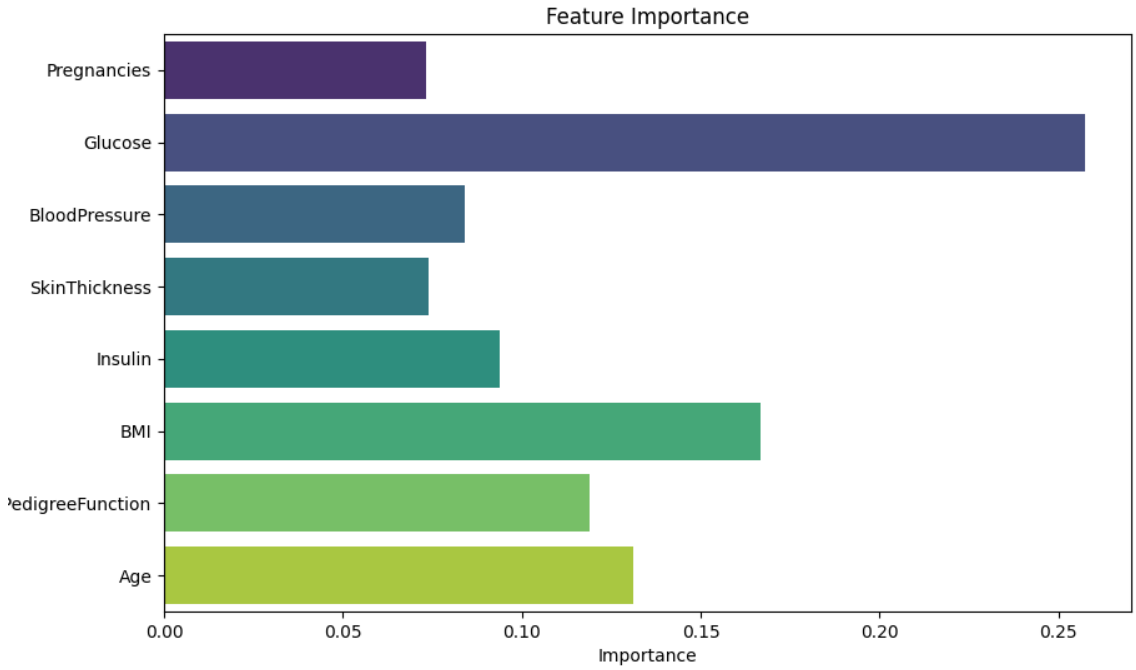
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("diabetes.csv")

print(data.head())
print(data.info())
print(data.describe())

print("Missing values:\n", data.isnull().sum())

columns_with_zeros = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI']
for col in columns_with_zeros:
    data[col] = data[col].replace(0, np.nan)
    data[col].fillna(data[col].mean(), inplace=True)

print("Missing values after handling:\n", data.isnull().sum())

X = data.drop('Outcome', axis=1)
y = data['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
roc_auc = roc_auc_score(y_test, y_proba)
print(f"ROC AUC Score: {roc_auc:.2f}")
```

```
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
xticklabels=['No Diabetes', 'Diabetes'], yticklabels=['No Diabetes', 'Diabetes'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

importances = model.feature_importances_
feature_names = X.columns
plt.figure(figsize=(10, 6))
sns.barplot(x=importances, y=feature_names, palette="viridis")
plt.title("Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

Visualizations




Feature Importance
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
edigreeFunction
Age
0.00
0.05
0.10
0.15
0.20
0.25
Importance