Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

Table of Contents

---

1. Introduction

Diabetes is a growing public health concern. Early detection and risk prediction are critical for effective intervention. This project demonstrates how to use the classic Logistic Regression algorithm to predict diabetes risk from patient health indicators, using the famous Pima Indians Diabetes Dataset.

This notebook is suitable for beginners who want to learn hands-on machine learning, as well as intermediate learners aiming to polish their data science portfolios with real medical problems.

---

2. Project Objectives

- Build a logistic regression model for diabetes prediction.
- Explore and preprocess data (handling missing values, scaling, etc.).

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

- Evaluate the model with industry-standard metrics and visualizations.
- Interpret and communicate results to both technical and non-technical audiences.
- Learn best practices in data science for medical and health applications.

---

3. Dataset Overview

- Rows: 768 patients
- Columns: 9 (8 health indicators + 1 outcome)
- Features:
  - Pregnancies: Number of times pregnant
  - Glucose: Plasma glucose concentration
  - BloodPressure: Diastolic blood pressure (mm Hg)
  - SkinThickness: Triceps skin fold thickness (mm)
  - Insulin: 2-Hour serum insulin (mu U/ml)
  - BMI: Body Mass Index (weight in kg/(height in m)^2)
  - DiabetesPedigreeFunction: Likelihood of diabetes based on family history
  - Age: Age (years)
  - Outcome: 1 = Diabetes, 0 = No diabetes

---

4. Project Workflow

4.1. Import Libraries

```python
# 1. Import Libraries
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
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, confusion_matrix, classification_report, roc_curve, auc)
```

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

Explanation:

- pandas, numpy: Data handling
- matplotlib, seaborn: Visualization
- sklearn: Machine learning and evaluation tools

4.2. Data Loading and Exploration

```
# 2. Load Dataset
url = 'diabetes.csv'
df = pd.read_csv(url, header=None)
df = pd.read_csv(url)  # No header=None, just the default
print(df.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
   Pregnancies  Glucose  BloodPressure  ...  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72  ...                     0.627   50        1
1            1       85             66  ...                     0.351   31        0
2            8      183             64  ...                     0.672   32        1
3            1       89             66  ...                     0.167   21        0
4            0      137             40  ...                     2.288   33        1

[5 rows x 9 columns]
```

```
<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   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

```
       Pregnancies     Glucose  ...         Age     Outcome
count   768.000000  768.000000  ...  768.000000  768.000000
mean      3.845052  120.894531  ...   33.240885    0.348958
std       3.369578   31.972618  ...   11.760232    0.476951
min       0.000000    0.000000  ...   21.000000    0.000000
25%       1.000000   99.000000  ...   24.000000    0.000000
50%       3.000000  117.000000  ...   29.000000    0.000000
75%       6.000000  140.250000  ...   41.000000    1.000000
max      17.000000  199.000000  ...   81.000000    1.000000

[8 rows x 9 columns]
```

Note: Zeros in Glucose, BloodPressure, etc. are not physically possible (will handle this next).

4.3. Handling Missing Values

Problem: Some features use 0 as "missing" values.

```python
# 4. Handle Missing Values
# Replace zeros with NaN for certain features
cols_with_zeros = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[cols_with_zeros] = df[cols_with_zeros].replace(0, np.nan)
print(df.isnull().sum())
```

```python
# Impute missing values with median (safer for medical data)
df.fillna(df.median(), inplace=True)
```

Explanation:

- Why do we use the Median here?
  - Because it is more robust to outliers than the mean, this is crucial for medical data.

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Solution: Median imputation fills missing values safely.

4.4. Train-Test Split

```
# 5. Feature/Target Split
X = df.drop( labels: 'Outcome', axis=1)
y = df['Outcome']

# 6. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.3, random_state=42, stratify=y)
```

- 70% of the data is used for training and 30% of the data is for unbiased evaluation.
- **Stratify:** Keeps class distribution balanced.

4.5. Feature Scaling

```
# 7. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- Logistic regression expects all features to be on similar scales.
- Standardization = 0 mean, 1 variance.

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

4.6. Model Training

```
# 8. Model Training
model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)
```

- Simple and interpretable binary classifier.
- random_state ensures reproducibility.

4.7. Evaluation Matrix

```
# 9. Predictions and Evaluation
y_pred = model.predict(X_test_scaled)
y_pred_prob = model.predict_proba(X_test_scaled)[:, 1]

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7445887445887446
Confusion Matrix:
 [[129  21]
 [ 38  43]]
```

Accuracy: 0.744

- The model correctly predicts about 74% of test cases.

Confusion Matrix

- True Negatives (TN): 129 – Model correctly predicts non-diabetes
- False Positives (FP): 21 – Model predicts diabetes but is wrong (false alarm)
- False Negatives (FN): 38 – Model misses actual diabetes cases (important in healthcare)
- True Positives (TP): 43 – Model correctly predicts diabetes

Medical Takeaway:

False negatives (FN = 38) are critical; missing a diabetes diagnosis could be dangerous.

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

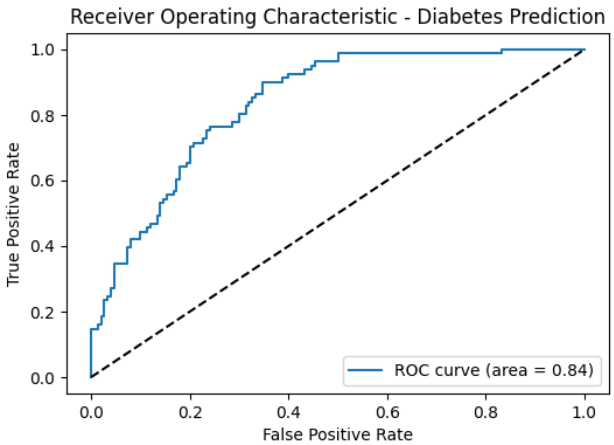
Classification Report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81       150
           1       0.67      0.53      0.59        81

    accuracy                           0.74       231
   macro avg       0.72      0.70      0.70       231
weighted avg       0.74      0.74      0.74       231
```

- **Precision (class 1):** 67% of predicted diabetes are actually correct.
- **Recall (class 1):** 53% of actual diabetes cases detected.
  - In healthcare, high recall is often more important than high precision.
- **F1-score:** Harmonic mean of precision/recall.

ROC Curve and AUC

```python
# 10. ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6,4))
plt.plot( *args: fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot( *args: [0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Diabetes Prediction')
plt.legend(loc='lower right')
plt.show()
```



Predicting Diabetes Risk with Logistic Regression

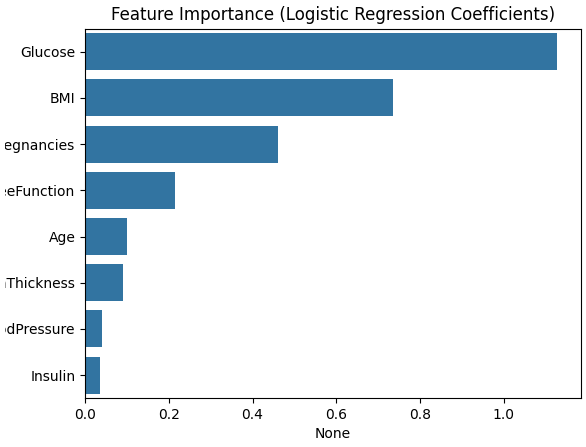
By:- Shivesh Raj Sahu

AUC = 0.84:

- **AUC (Area Under Curve) tells you the probability the model will rank a random positive instance higher than a random negative one.**
- **0.5 = random guess; 1.0 = perfect model.**
- **0.84 is considered "good" for medical diagnostics.**

Feature Importance:

```
# 11. Feature Importance (Coefficients)
feature_importance = pd.Series(model.coef_[0], index=X.columns)
feature_importance = feature_importance.abs().sort_values(ascending=False)
print("\nFeature importance (absolute value):")
print(feature_importance)

sns.barplot(x=feature_importance, y=feature_importance.index)
plt.title("Feature Importance (Logistic Regression Coefficients)")
plt.show()
```



Most Important Features:

- Glucose – Strongest influence on diabetes prediction (biologically sensible).
- BMI
- Pregnancies
- Diabetes Pedigree Function

Least important: Insulin, BloodPressure in this dataset.

Predicting Diabetes Risk with Logistic Regression

By:- Shivesh Raj Sahu

4.8. Interpretation of the results

What do these numbers mean in real life?

- The model can pick up on strong risk factors (e.g., high glucose, high BMI).
- Good overall accuracy but less sensitive for picking up ALL true diabetes cases (lower recall for class 1).
- There is a trade-off between false positives (unnecessary further testing) and false negatives (missed cases).

4.9. Visualization

ROC Curve

- Shows the trade-off between sensitivity and specificity.
- The curve bows towards the upper left (ideal), and AUC of 0.84 shows model is strong.

Feature Importance

- Bar plot quickly tells which factors matter most in prediction.
- Useful for clinicians, as it aligns with medical knowledge.

---

5. Key Challenges and Solutions

Challenge: Many features had zero values instead of missing values.

- **Solution:** Replaced zeroes with NaN, then imputed using median.

Challenge: Possible class imbalance (more negatives than positives).

- **Solution:** Used stratified split; could try oversampling/undersampling or class weighting for even better recall.

Challenge: Linear Model Limitations.

- **Solution:** For complex patterns, could try nonlinear models (Random Forests, SVMs, Neural Nets), but always start with interpretable baselines.

---

6. Conclusion

- Logistic Regression is a robust, interpretable first step in medical classification.
- This pipeline achieved 74% accuracy and 0.84 AUC using only simple, classic tools.
- The most predictive features matched established medical risk factors for diabetes.
- Limitations: Sensitivity/recall for detecting all true cases was moderate; tuning threshold or using other models may help.
- This workflow is widely applicable to other health datasets and binary classification problems.

---

7. Future Work & Best Practices

- Tune the threshold to optimize recall vs. precision, depending on the application.
- Try advanced models: Random Forest, XGBoost, SVM, or Deep Learning for further improvement.
- Perform cross-validation for more robust estimates.
- Advanced feature engineering (e.g., interaction terms, domain features).
- Regularization tuning (C parameter) to avoid overfitting.

---

8. References

- Pima Indians Diabetes Database - Kaggle
- scikit-learn Documentation
- Kuhn & Johnson, Applied Predictive Modeling