Assignment-1 Report
Logistic Regression on Pima Indians Diabetes Dataset

Aryan Kesharwani
Roll No: 2201CS19

September 19, 2025

1 Introduction

The objective of this assignment is to apply a machine learning algorithm covered in class to a real-world dataset. Logistic Regression was chosen as the algorithm, and the Pima Indians Diabetes dataset was selected from the UCI repository.

2 Dataset Description

The dataset consists of 768 instances with 8 medical attributes and a binary target variable (diabetic or non-diabetic). Missing values were imputed with median values.

3 Methodology

Steps followed:

1. Data preprocessing and imputing missing values.
2. Train-test split (80-20, stratified).
3. Standardization of features.
4. Training Logistic Regression.
5. Evaluation with Accuracy, Precision, Recall, F1, ROC-AUC.
6. Cross-validation (5-fold).
7. Visualization with Confusion Matrix and ROC curve.

4 Results

On the test set:

- Accuracy: 0.6948

- Precision: 0.5745
- Recall: 0.5000
- F1-score: 0.5347
- ROC AUC: 0.8128

Cross-validation ROC AUC scores:

$$[0.8250,\ 0.8672,\ 0.8515,\ 0.8294,\ 0.8096] \quad \text{Mean AUC} = 0.8366$$

5 Visualizations

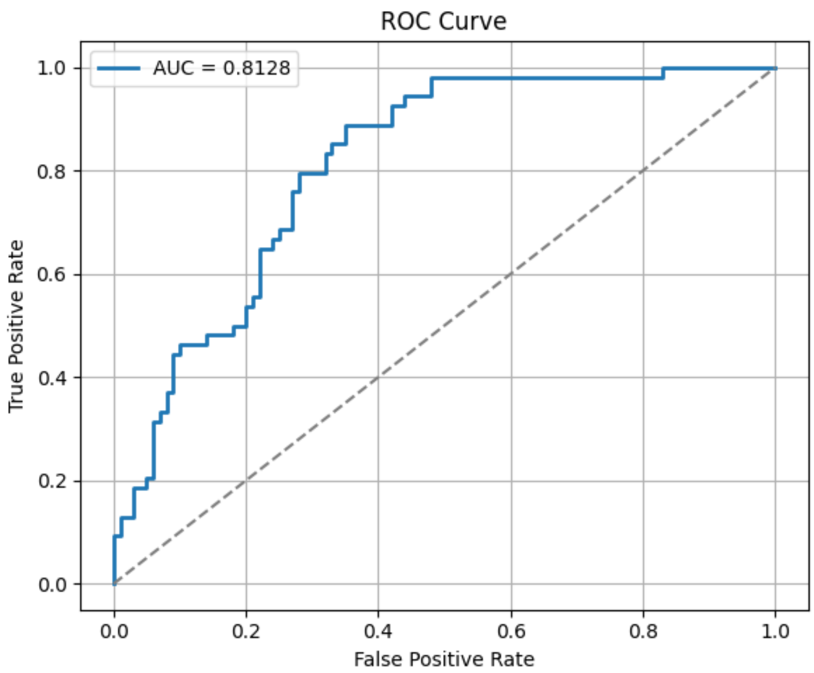

Figure 1: ROC Curve (AUC = 0.8128).

6 Conclusion

The Logistic Regression model achieved an accuracy of ∼69.5% with ROC AUC of 0.81. Though precision and recall for the diabetic class were modest, the model demonstrated stable generalization across cross-validation folds.

Appendix: Code

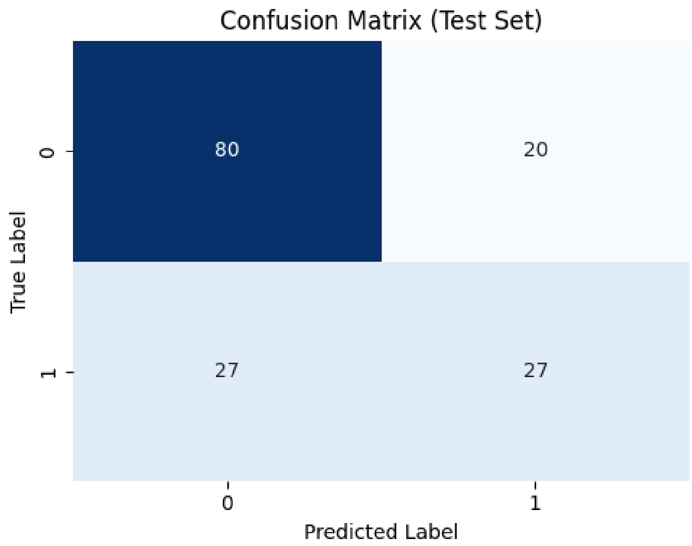

Figure 2: Confusion Matrix.

```
# === Logistic Regression on Pima Indians Diabetes
!pip install -q joblib

# 1) Imports
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedKFold,
    cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, precision_score,
    recall_score, f1_score,
                             roc_auc_score, confusion_matrix,
                                classification_report, roc_curve)
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import os

# 2) Load dataset (public UCI mirror, no Kaggle login required)
URL = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima
    -indians-diabetes.data.csv"
cols = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
        "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

df = pd.read_csv(URL, header=None, names=cols)
print("Dataset␣shape:", df.shape)
print(df.head())

# 3) Handle invalid zeros (replace with NaN)
cols_with_zeros = ["Glucose", "BloodPressure", "SkinThickness", "
    Insulin", "BMI"]
df[cols_with_zeros] = df[cols_with_zeros].replace(0, np.nan)

# Fill NaNs with median values
```

```python
for c in cols_with_zeros:
    df[c].fillna(df[c].median(), inplace=True)

# 4) Features and target
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

# 5) Train/test split (stratified to preserve class balance)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 6) Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 7) Train logistic regression model
clf = LogisticRegression(solver="liblinear", penalty="l2", random_state
    =42)
clf.fit(X_train_scaled, y_train)

# 8) Predictions
y_pred = clf.predict(X_test_scaled)
y_proba = clf.predict_proba(X_test_scaled)[:, 1]

# 9) Metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)

print("\n=== Evaluation on Test Set ===")
print(f"Accuracy : {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall   : {rec:.4f}")
print(f"F1-score : {f1:.4f}")
print(f"ROC AUC  : {auc:.4f}\n")

print("Classification report:\n")
print(classification_report(y_test, y_pred, digits=4))

# 10) Confusion Matrix Plot
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix (Test Set)")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.tight_layout()
plt.savefig("confusion_matrix.png")
plt.show()

# 11) ROC Curve Plot
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, linewidth=2, label=f"AUC = {auc:.4f}")
```

```
plt.plot([0,1],[0,1], linestyle="--", color="gray")
plt.xlabel("False␣Positive␣Rate")
plt.ylabel("True␣Positive␣Rate")
plt.title("ROC␣Curve")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("roc_curve.png")
plt.show()

# 12) Cross-Validation (5-fold ROC AUC)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(clf, scaler.fit_transform(X), y, cv=cv,
    scoring="roc_auc")
print("5-fold␣CV␣ROC␣AUC␣scores:", cv_scores)
print("Mean␣CV␣AUC:␣{:.4f}".format(cv_scores.mean()))

# 13) Save artifacts
os.makedirs("artifacts", exist_ok=True)
joblib.dump(clf, "artifacts/logistic_model.joblib")
joblib.dump(scaler, "artifacts/scaler.joblib")

with open("artifacts/metrics.txt", "w") as f:
    f.write(f"Accuracy:␣{acc:.4f}\n")
    f.write(f"Precision:␣{prec:.4f}\n")
    f.write(f"Recall:␣{rec:.4f}\n")
    f.write(f"F1:␣{f1:.4f}\n")
    f.write(f"ROC␣AUC:␣{auc:.4f}\n")
    f.write("5-fold␣CV␣AUCs:␣" + ",␣".join([f"{s:.4f}" for s in
        cv_scores]) + "\n")

print("\nSaved:")
print("-␣confusion_matrix.png")
print("-␣roc_curve.png")
print("-␣artifacts/logistic_model.joblib")
print("-␣artifacts/scaler.joblib")
print("-␣artifacts/metrics.txt")
```