Logistic Regression on Indian Diabetes Dataset

Step 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, 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, confusion_matrix, roc_curve, auc
```

Step 2: Load Dataset

```
df = pd.read_csv("diabetes.csv")
print(df.head())
print(df.info())
print(df.describe())
```

Step 3: Exploratory Data Analysis (EDA)

```
sns.countplot(x='Outcome', data=df)
plt.title("Target Distribution (0 = No Diabetes, 1 = Diabetes)")
plt.show()

plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

Step 4: Handle Missing/Invalid Values

```
cols_with_zero =
['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
for col in cols_with_zero:
df[col] = df[col].replace(0, np.nan)
df[col].fillna(df[col].median(), inplace=True)
```

Step 5: Train-Test Split

```
X = df.drop("Outcome", axis=1)
y = df["Outcome"]
```

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

Step 6: Feature Scaling

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 7: Model Training

```
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
y_pred_prob = log_reg.predict_proba(X_test)[:,1]
```

Step 8: Evaluation Metrics

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.show()
```

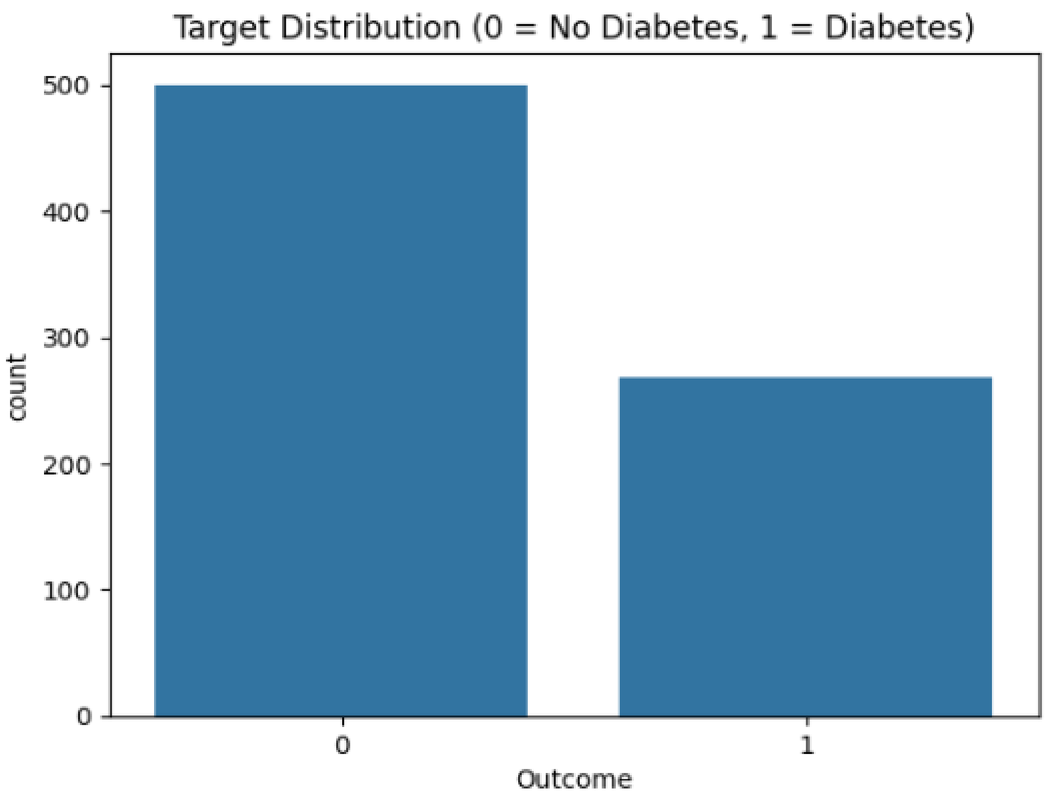
Step 9: ROC Curve

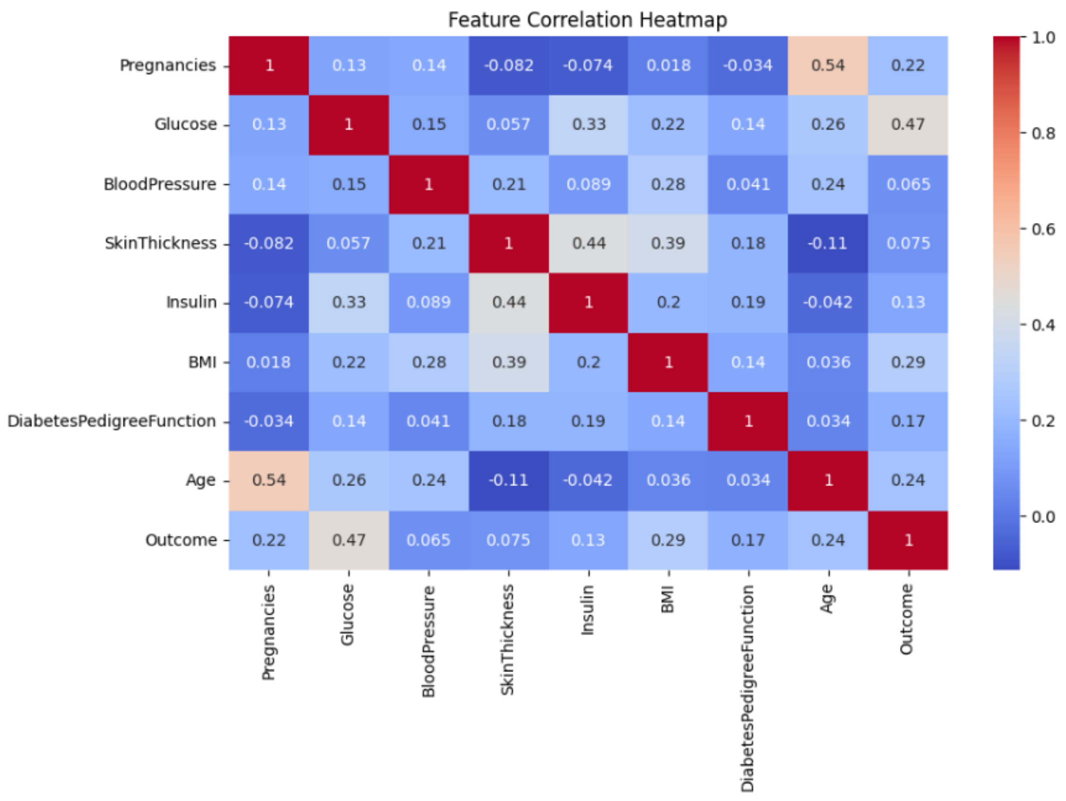
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='blue', label=f"ROC Curve (AUC =
{roc_auc:.2f})")
plt.plot([0,1],[0,1], color='red', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

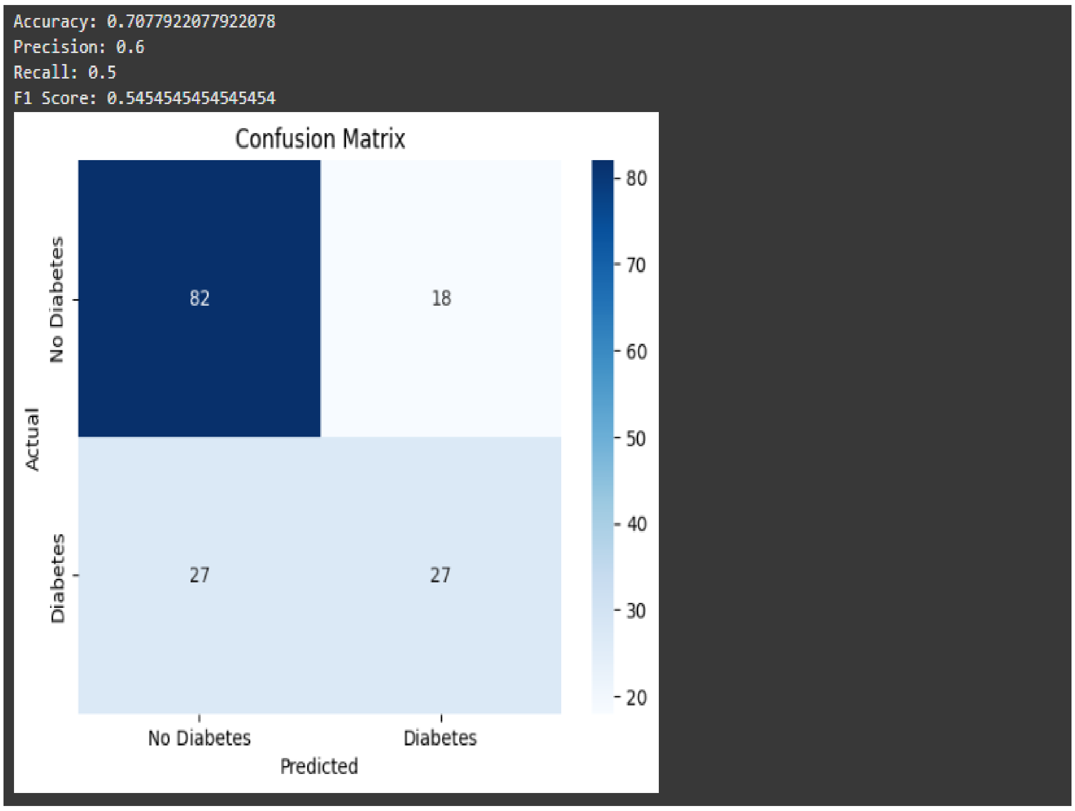
Step 10: Cross Validation

```
cv_scores = cross_val_score(log_reg, X, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

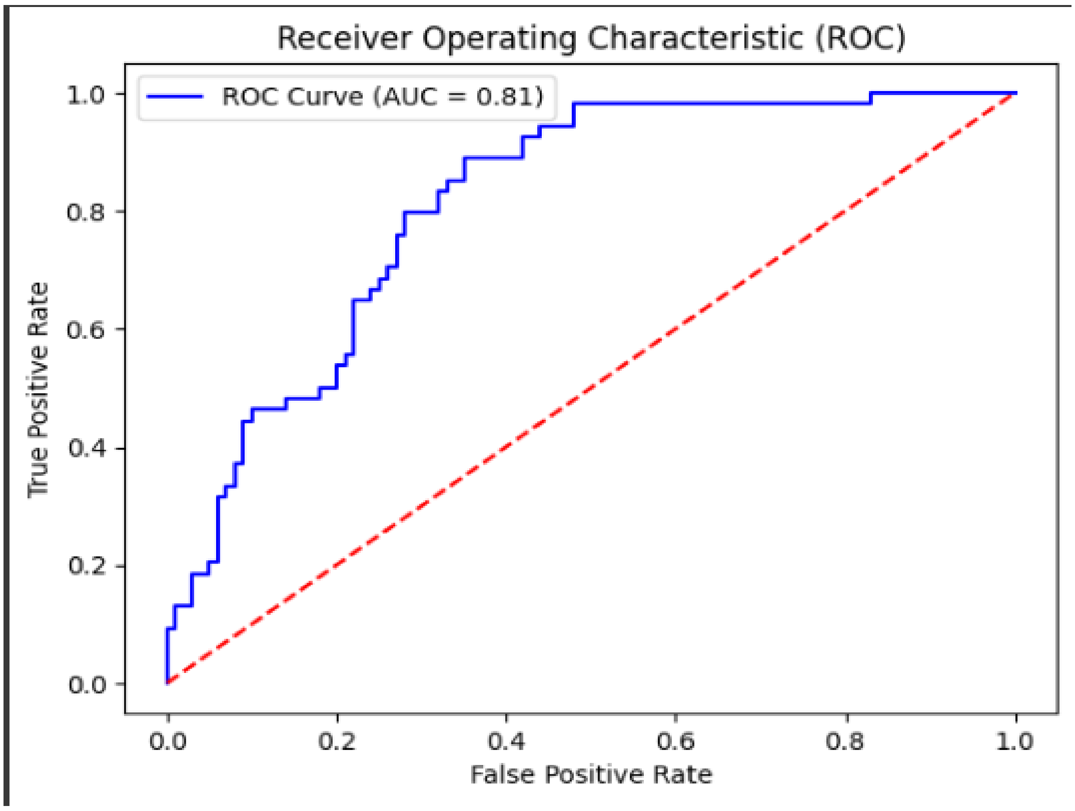
Results and Visualizations







```
Accuracy: 0.7077922077922078
Precision: 0.6
Recall: 0.5
F1 Score: 0.5454545454545454
```


Receiver Operating Characteristic (ROC)
ROC Curve (AUC = 0.81)
True Positive Rate
False Positive Rate