Assignment 1

Diabetes Prediction Using Logistic Regression: A Machine Learning Approach with Diagnostic Metrics

Problem Statement

This study's goal is to use a set of diagnostic parameters to predict whether a patient has diabetes. The dataset that was supplied, The aim is to create a classification model that can reliably predict the Outcome variable, where 1 denotes a positive diabetes diagnosis and 0 denotes no diabetes. diabetes2.csv contains a variety of health indicators for multiple patients.

Abstract

This article describes a machine learning project that uses a collection of diagnostic metrics to predict diabetes in patients. The main objective is to create a predictive model that uses a collection of characteristics to categorise patients as either diabetes or non-diabetic. A logistic regression model, an appropriate approach for binary classification tasks, is used in this research. To determine how well the model identifies diabetes cases, the data is pre-processed and its performance is assessed using important metrics like accuracy, recall, and F1 score in addition to a confusion matrix.

Methodology

The methodology followed a standard machine learning workflow:

1. **Data Loading**: The diabetes2.csv dataset is loaded into a Pandas Data Frame. The dataset includes features such as Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age, with 'Outcome' as the target variable.

    ```
    df = pd.read_csv('/content/diabetes2.csv')
    df.sample(5)
    ```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
631	0	102	78	40	90	34.5	0.238	24	0
411	1	112	72	30	176	34.4	0.528	25	0
640	0	102	86	17	105	29.3	0.695	27	0
16	0	118	84	47	230	45.8	0.551	31	1
638	7	97	76	32	91	40.9	0.871	32	1

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
df.isnull().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

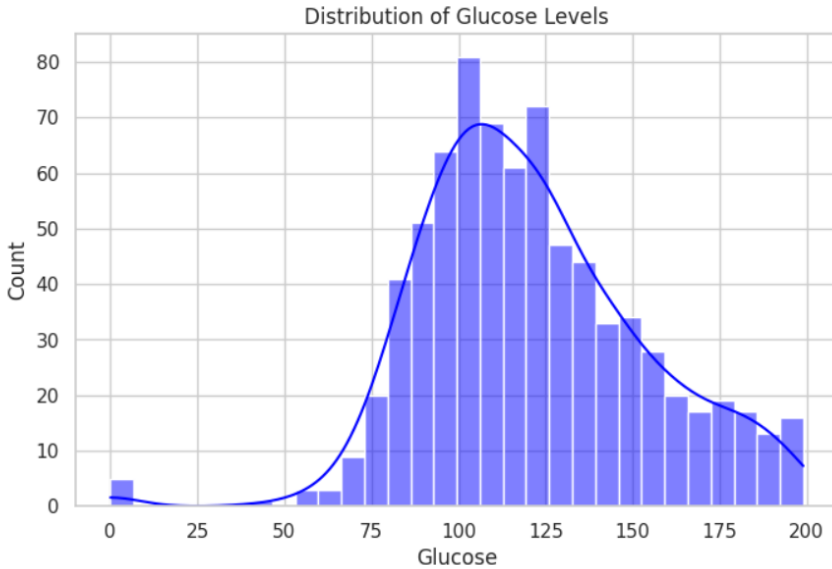
dtype: int64

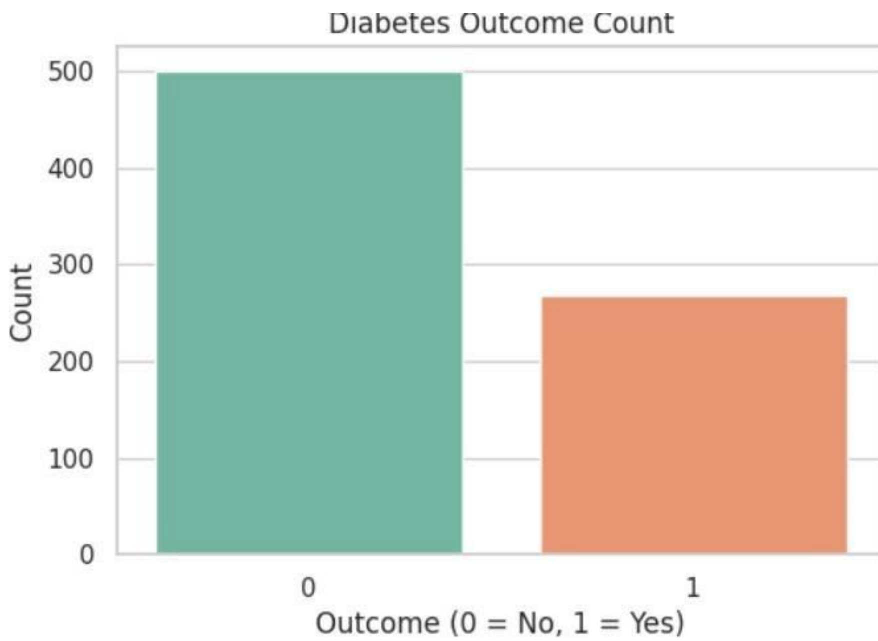
```
df.duplicated().sum()
```

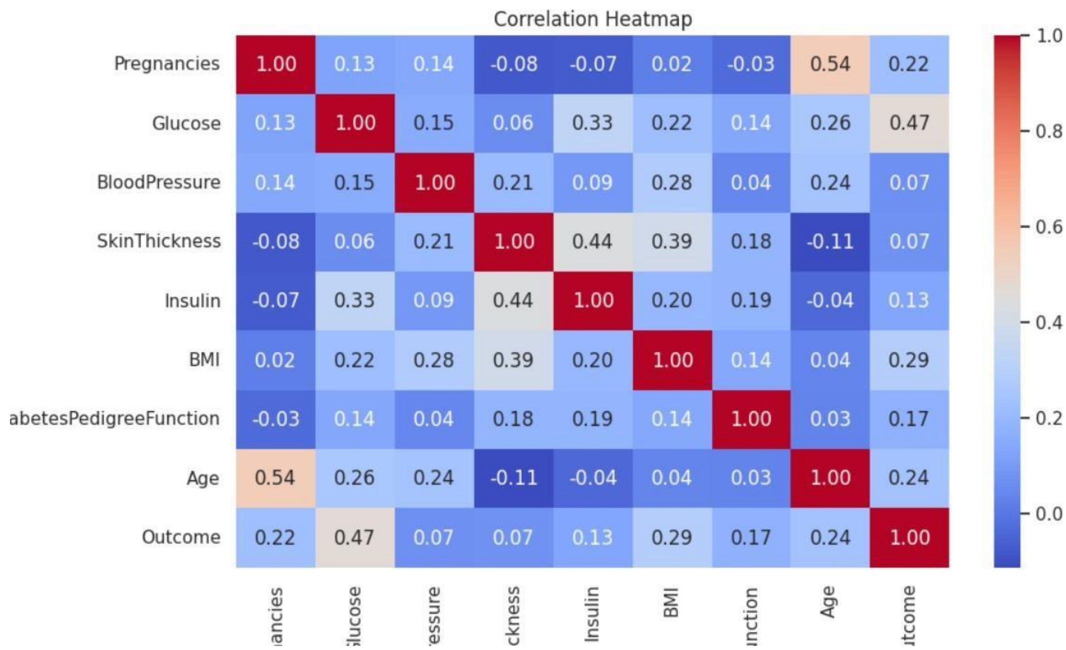
```
np.int64(0)
```

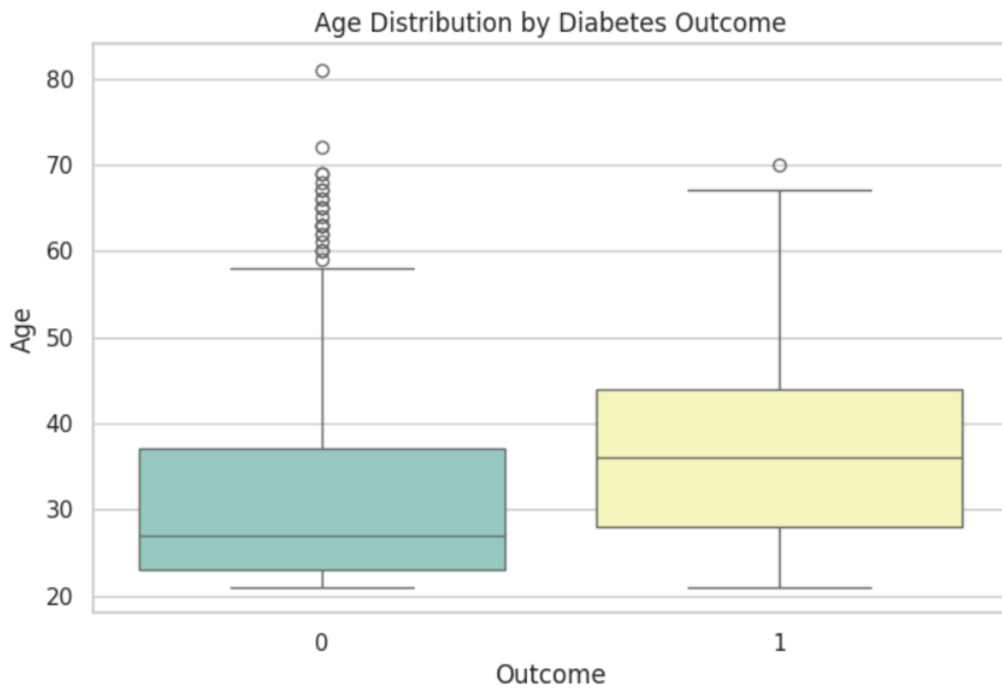
2. **Data Visualization**: It generates several plots to visualize the data, including a histogram of Glucose levels, a bar chart of the diabetes outcome count, a correlation heatmap of the variables, and a boxplot showing the age distribution by diabetes outcome.

```python
import seaborn as sns

sns.set(style="whitegrid")

# Distribution of Glucose levels
plt.figure(figsize=(8,5))
sns.histplot(df['Glucose'], kde=True, bins=30, color="blue")
plt.title("Distribution of Glucose Levels")
plt.xlabel("Glucose")
plt.ylabel("Count")
plt.show()

# Outcome count (0 = Non-diabetic, 1 = Diabetic)
plt.figure(figsize=(6,4))
sns.countplot(x='Outcome', data=df, palette='Set2')
plt.title("Diabetes Outcome Count")
plt.xlabel("Outcome (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()

# Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()

# Age distribution by outcome
plt.figure(figsize=(8,5))
sns.boxplot(x="Outcome", y="Age", data=df, palette="Set3")
plt.title("Age Distribution by Diabetes Outcome")
plt.show()
```


Distribution of Glucose Levels
Count
Glucose
0
10
20
30
40
50
60
70
80
0
25
50
75
100
125
150
175
200


Diabetes Outcome Count
Count
0
100
200
300
400
500
0
1
Outcome (0 = No, 1 = Yes)





3. **Data Splitting**: The dataset is split into training and testing sets to prepare for model training and evaluation. The training set is used to train the model, while the test set is used to evaluate its performance on unseen data.
4. **Feature Scaling**: The features in the dataset are scaled using Standard Scaler. This is an important preprocessing step for logistic regression to ensure that all features contribute equally to the model, as they are on different scales.

5. **Model Training**: A logistic regression model is initialized and trained on the scaled training data. This model learns the relationship between the features and the target variable to make predictions.
6. **Model Evaluation**: The trained model's performance is evaluated on the test set using several metrics.

Results and Output

The performance of the logistic regression model on the test data is evaluated using accuracy, recall, F1 score, and a confusion matrix. The results are as follows:

- **Accuracy**: 0.7532
- **Recall**: 0.7311
- **F1 Score**: 0.63
- **Confusion Matrix**: The confusion matrix provides a detailed breakdown of the model's predictions.
- **Classification report**

The confusion matrix shows the following:

- **True Negatives (TN)**: 84 (Correctly predicted non-diabetic)
- **False Positives (FP)**: 25 (Incorrectly predicted as diabetic)
- **False Negatives (FN)**: 13 (Incorrectly predicted as non-diabetic)
- **True Positives (TP)**: 32 (Correctly predicted as diabetic)

```python
X_train, X_test, y_train, y_test = train_test_split(df.drop(['Outcome'], axis=1),
                                                    df['Outcome'],
                                                    test_size=0.2,
                                                    random_state=2)
```

```python
std = StandardScaler()
```

```python
X_train = std.fit_transform(X_train)
X_test = std.transform(X_test)
```

```python
model = LogisticRegression(max_iter=1000, class_weight='balanced')
```

```python
model.fit(X_train, y_train)
```

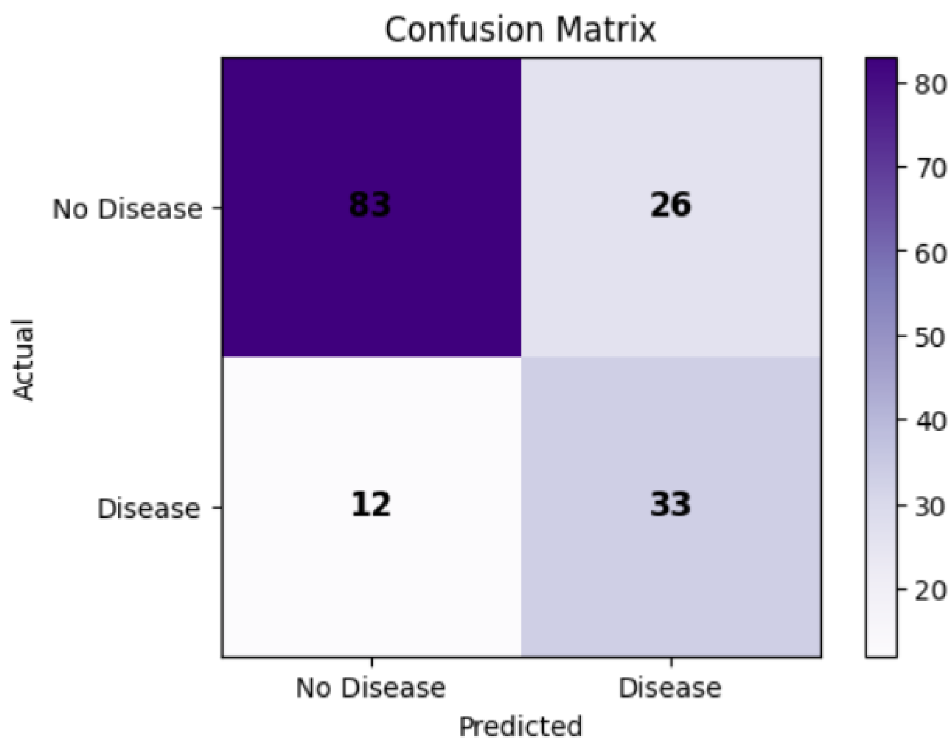
```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

```python
y_pred = model.predict(X_test)
```

```python
print('Accuracy: ', accuracy_score(y_test, y_pred))
print('Recall: ', recall_score(y_test, y_pred))
print('f1 score: ', f1_score(y_test, y_pred))
print('Confusion Matrix: ', confusion_matrix(y_test, y_pred))
```

```
Accuracy:  0.7532467532467533
Recall:  0.7333333333333333
f1 score:  0.6346153846153846
Confusion Matrix:  [[83 26]
 [12 33]]
```

```python
import numpy as np
import matplotlib.pyplot as plt


cm = np.array([[83, 26],
               [12, 33]])

plt.figure(figsize=(6,4))
plt.imshow(cm, cmap='Purples')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0, 1], ['No Disease', 'Disease'])
plt.yticks([0, 1], ['No Disease', 'Disease'])

for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, cm[i, j], ha='center', va='center', color='black', fontsize=12, fontweight='bold')

plt.colorbar()
plt.show()
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.76      0.81       109
           1       0.56      0.73      0.63        45

    accuracy                           0.75       154
   macro avg       0.72      0.75      0.72       154
weighted avg       0.78      0.75      0.76       154
```

Conclusion

Based on the given dataset, the logistic regression model performed admirably in predicting diabetes. The model accurately anticipated the result for three-quarters of the test instances, with an accuracy of roughly 75.3%. The model was able to accurately identify a sizable percentage of real diabetes cases, according to the recall score of 73.1%. The model's precision and recall are balanced by its F1 score of 63.7%. Although there is room for improvement by investigating alternative models or feature engineering approaches, the results indicate that the model is a practical tool for diabetes prediction.