Assignment-2-Jose-Zacarias

August 6, 2024

Student Name: Jose Zacarias

Student Id: N01663659

1 Assignment 2

1.0.1 Step 1: Import Libraries

[13]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (
    confusion_matrix,
    ConfusionMatrixDisplay,
    precision_score,
    recall_score,
    f1_score,
)
```

[3]:
```
# loading Dataset
df = pd.read_csv('Breast cancer Wisconsin.csv')
df.head()
```

[3]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
```

```
4        0.10030          0.13280        0.1980               0.10430

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33           184.60      2019.0            0.1622
1  …          23.41           158.80      1956.0            0.1238
2  …          25.53           152.50      1709.0            0.1444
3  …          26.50            98.87       567.7            0.2098
4  …          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[4]:
```python
# checking types
df.dtypes
```

[4]:
```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
```

```
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

[5]:
```
# Dropping the 'id' column and the unnamed last column as they are not useful␣
↪for prediction
df.drop(['id', 'Unnamed: 32'], axis=1, inplace=True)
```

1.0.2 Step 2: Encoding Diagnosis, is recommended to use "LabelEncoder" instead of "get_dummies" because this is the target variable

[7]:
```
label_encoder = LabelEncoder()
df['diagnosis'] = label_encoder.fit_transform(df['diagnosis'])
df
```

[7]:
```
     diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0            1        17.99         10.38          122.80     1001.0
1            1        20.57         17.77          132.90     1326.0
2            1        19.69         21.25          130.00     1203.0
3            1        11.42         20.38           77.58      386.1
4            1        20.29         14.34          135.10     1297.0
..         …          …            …             …        …
564          1        21.56         22.39          142.00     1479.0
565          1        20.13         28.25          131.20     1261.0
566          1        16.60         28.08          108.30      858.1
567          1        20.60         29.33          140.10     1265.0
568          0         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710
1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
3            0.14250           0.28390         0.24140              0.10520
4            0.10030           0.13280         0.19800              0.10430
..               …               …              …                  …
```

```
564          0.11100           0.11590         0.24390                0.13890
565          0.09780           0.10340         0.14400                0.09791
566          0.08455           0.10230         0.09251                0.05302
567          0.11780           0.27700         0.35140                0.15200
568          0.05263           0.04362         0.00000                0.00000

     symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0           0.2419  …        25.380          17.33           184.60
1           0.1812  …        24.990          23.41           158.80
2           0.2069  …        23.570          25.53           152.50
3           0.2597  …        14.910          26.50            98.87
4           0.1809  …        22.540          16.67           152.20
..             …  …           …             …               …
564         0.1726  …        25.450          26.40           166.10
565         0.1752  …        23.690          38.25           155.00
566         0.1590  …        18.980          34.12           126.70
567         0.2397  …        25.740          39.42           184.60
568         0.1587  …         9.456          30.37            59.16

     area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0        2019.0           0.16220            0.66560           0.7119
1        1956.0           0.12380            0.18660           0.2416
2        1709.0           0.14440            0.42450           0.4504
3         567.7           0.20980            0.86630           0.6869
4        1575.0           0.13740            0.20500           0.4000
..          …               …                  …               …
564      2027.0           0.14100            0.21130           0.4107
565      1731.0           0.11660            0.19220           0.3215
566      1124.0           0.11390            0.30940           0.3403
567      1821.0           0.16500            0.86810           0.9387
568       268.6           0.08996            0.06444           0.0000

     concave points_worst  symmetry_worst  fractal_dimension_worst
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
3                  0.2575          0.6638                  0.17300
4                  0.1625          0.2364                  0.07678
..                    …             …                      …
564                0.2216          0.2060                  0.07115
565                0.1628          0.2572                  0.06637
566                0.1418          0.2218                  0.07820
567                0.2650          0.4087                  0.12400
568                0.0000          0.2871                  0.07039

[569 rows x 31 columns]
```

[8]:
```
# defining features and target variable
X = df.drop('diagnosis', axis=1)
y = df['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Standardizing the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

1.0.3 Step 3: Defining and fitting the model

[9]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

[9]:
```
0.9736842105263158
```

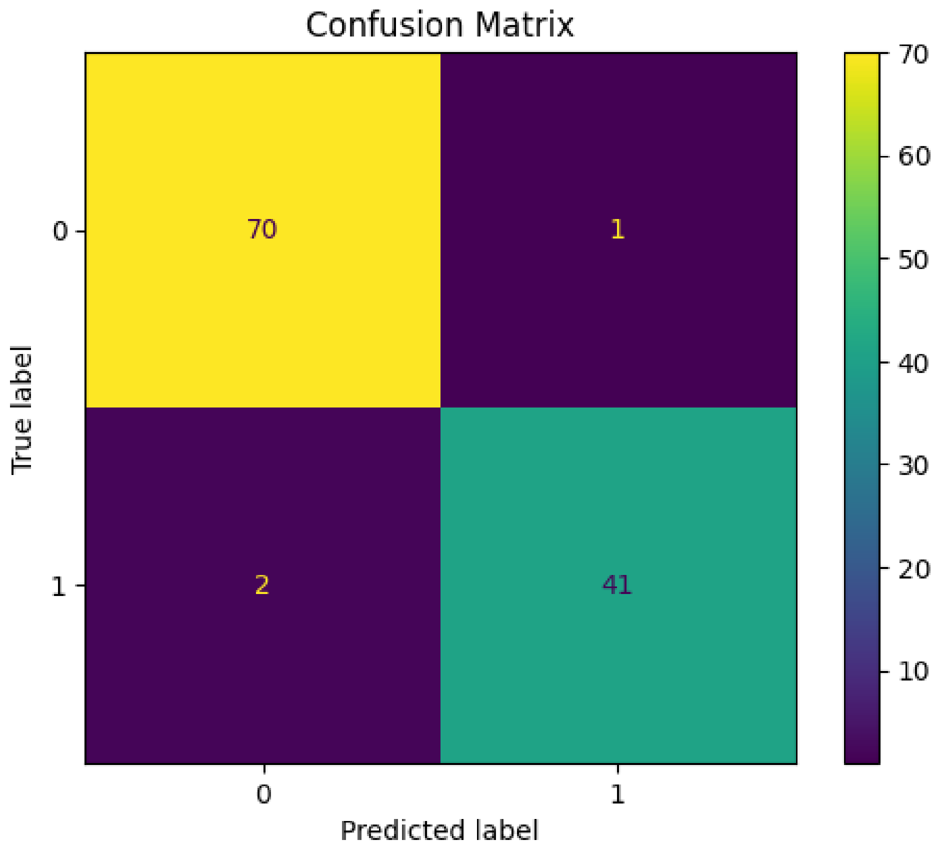
1.0.4 Step 4: Evaluate the moodel

[15]:
```
# Make predictions on the testing set
y_pred = model.predict(X_test)
```

[16]:
```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix")
plt.show()

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```




```
Accuracy: 0.97
Precision: 0.98
Recall: 0.95
F1 Score: 0.96
```

1.0.5 Conclusion: The model performs well and its reliable for scenarios where both false positives and false negatives are critical considerations like for cancer prediction data like this one.