ECG HEARTBEAT

Students: Dilnura G., Nurzhas A., Aktoty A.
Group: BDA-2103

Instructor: Akmaral Kuatbayeva

TODAY'S AGENDA

- Dataset Description
- Linear Regression
- Logistic Regression
- Binary Classification
- Regularization
- Results
- Comparison
- Conclusion

ECG Heartbeat Categorization Dataset

- MIT-BIH Arrhythmia Dataset
- The PTB Diagnostic ECG Dataset.

The dataset contains electrocardiogram (ECG) waveforms that show different types of heartbeats. Some are normal, and others are not because of different heart conditions like arrhythmias and heart attacks.

ECG Heartbeat Categorization Dataset

Looking at how many classes are there in each dataset The MIT dataset has 5 classes:

- 0 = N (Normal Beat)
- 1 = S (Supraventricular premature beat)
- 2 = V (Premature ventricular contraction)
- 3 = F (Fusion of ventricular and normal beat)
- 4 = Q (Unclassifiable beat)

Compared to the PTB dataset which is 1 for abnormal and 0 for normal

LINEAR REGRESSION

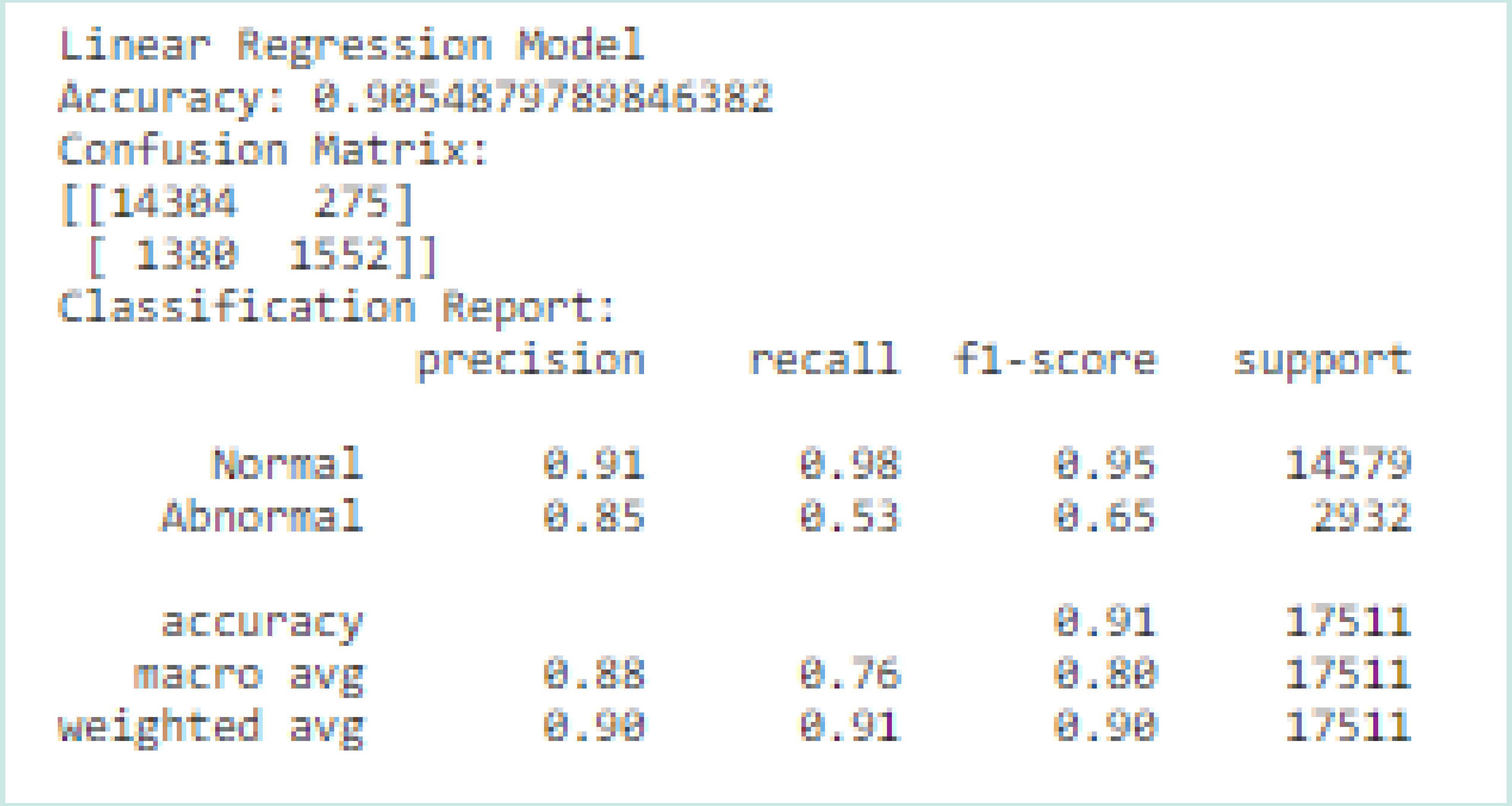
Objective:

Predict whether a heartbeat is normal or abnormal using a linear regression approach (not the standard method for binary classification).

Model Performance:

Accuracy: 90.55%

Demonstrates linear regression's application in classification tasks.- High precision in identifying 'Normal' beats.- Moderate recall for 'Abnormal' beats.



```
Linear Regression Model
Accuracy: 0.9054879789846382
Confusion Matrix:
[[14304   275]
 [ 1380  1552]]
Classification Report:
              precision    recall  f1-score   support

      Normal       0.91      0.98      0.95     14579
    Abnormal       0.85      0.53      0.65      2932

    accuracy                           0.91     17511
   macro avg       0.88      0.76      0.80     17511
weighted avg       0.90      0.91      0.90     17511
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import normalize

# Select the MIT dataset for classification (assuming mit_train is already loaded)
# Let's consider 'Normal Beat' (0) vs 'Abnormal Beat' (1)
mit_train['Binary_Class'] = mit_train['Class'].apply(lambda x: 0 if x == 0 else 1)

# Features (X) and target variable (y)
X = mit_train.drop(['Class', 'Binary_Class'], axis=1)
y = mit_train['Binary_Class']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize the data (optional, depending on your use case)
X_train = normalize(X_train, axis=0, norm='max')
X_test = normalize(X_test, axis=0, norm='max')

# Create and fit the linear regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)

# Predictions
y_pred = linear_reg.predict(X_test)

# Convert probabilities to binary predictions
y_pred_binary = (y_pred > 0.5).astype(int)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred_binary)
conf_matrix = confusion_matrix(y_test, y_pred_binary)
classification_report_str = classification_report(y_test, y_pred_binary, target_names=['Normal', 'Abnormal'])

print("Linear Regression Model")
print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_report_str)
```

LOGISTIC REGRESSION

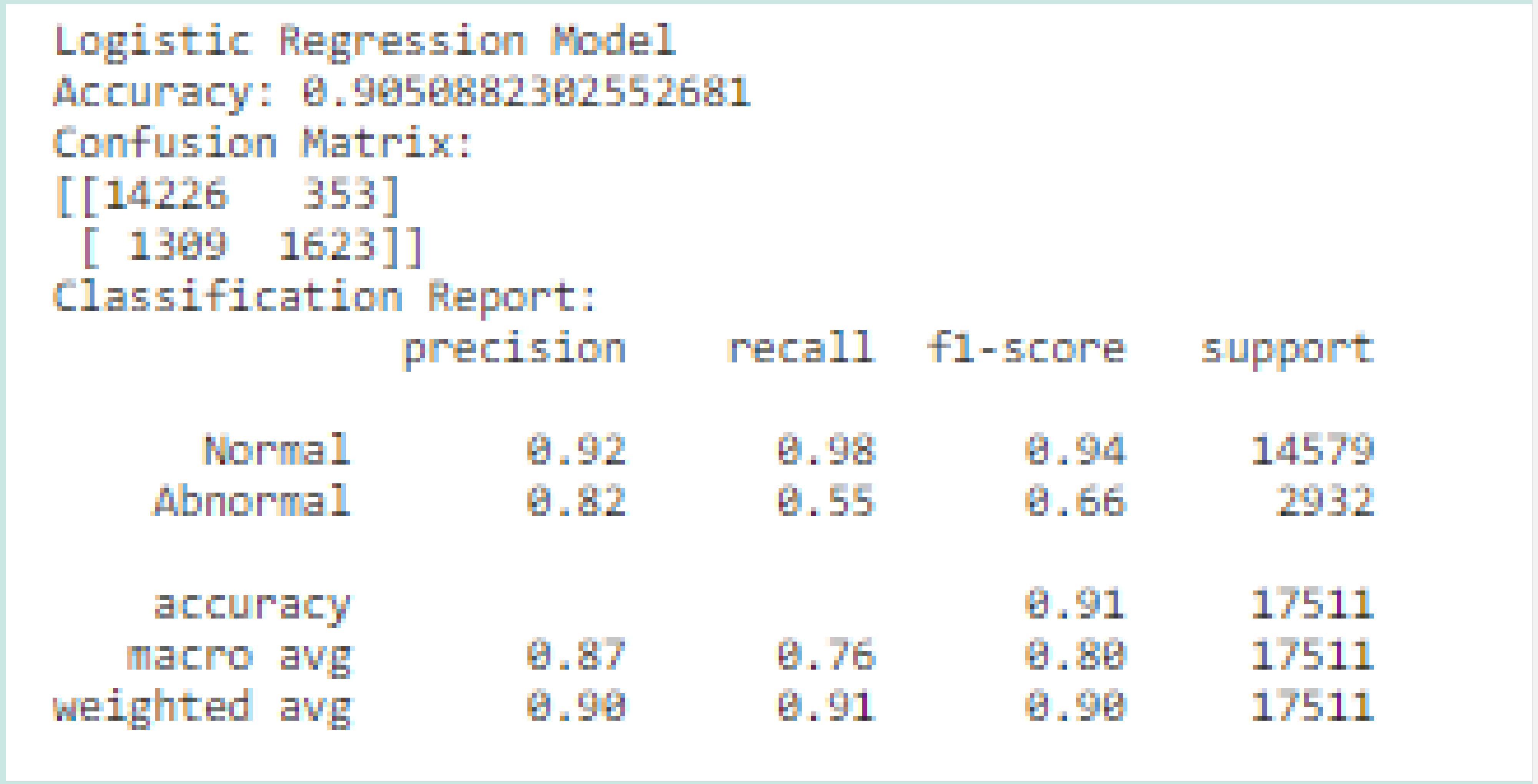
Objective:

Predict whether a heartbeat is normal or abnormal using logistic regression.

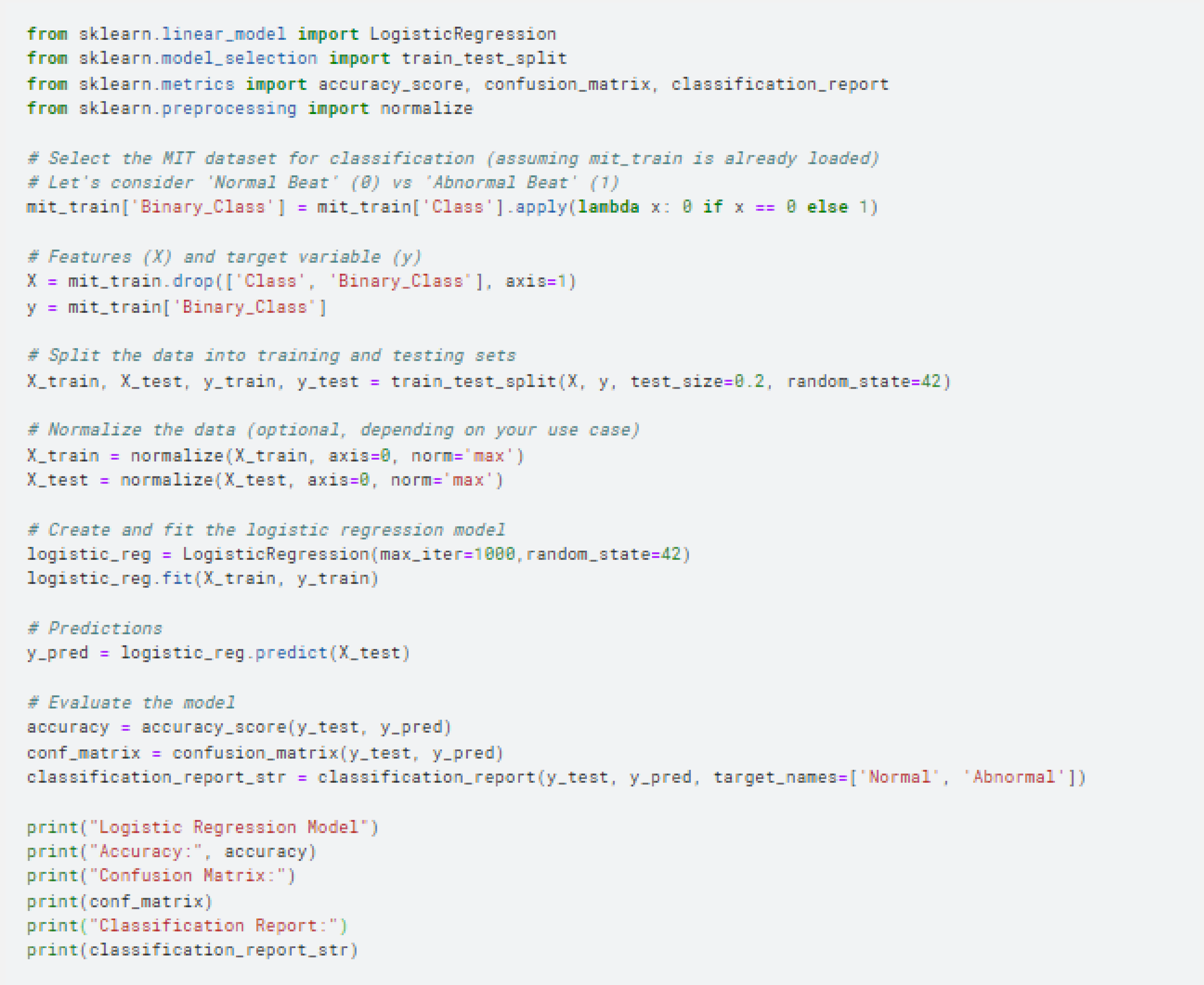
Model Performance:

Accuracy: 90.51%

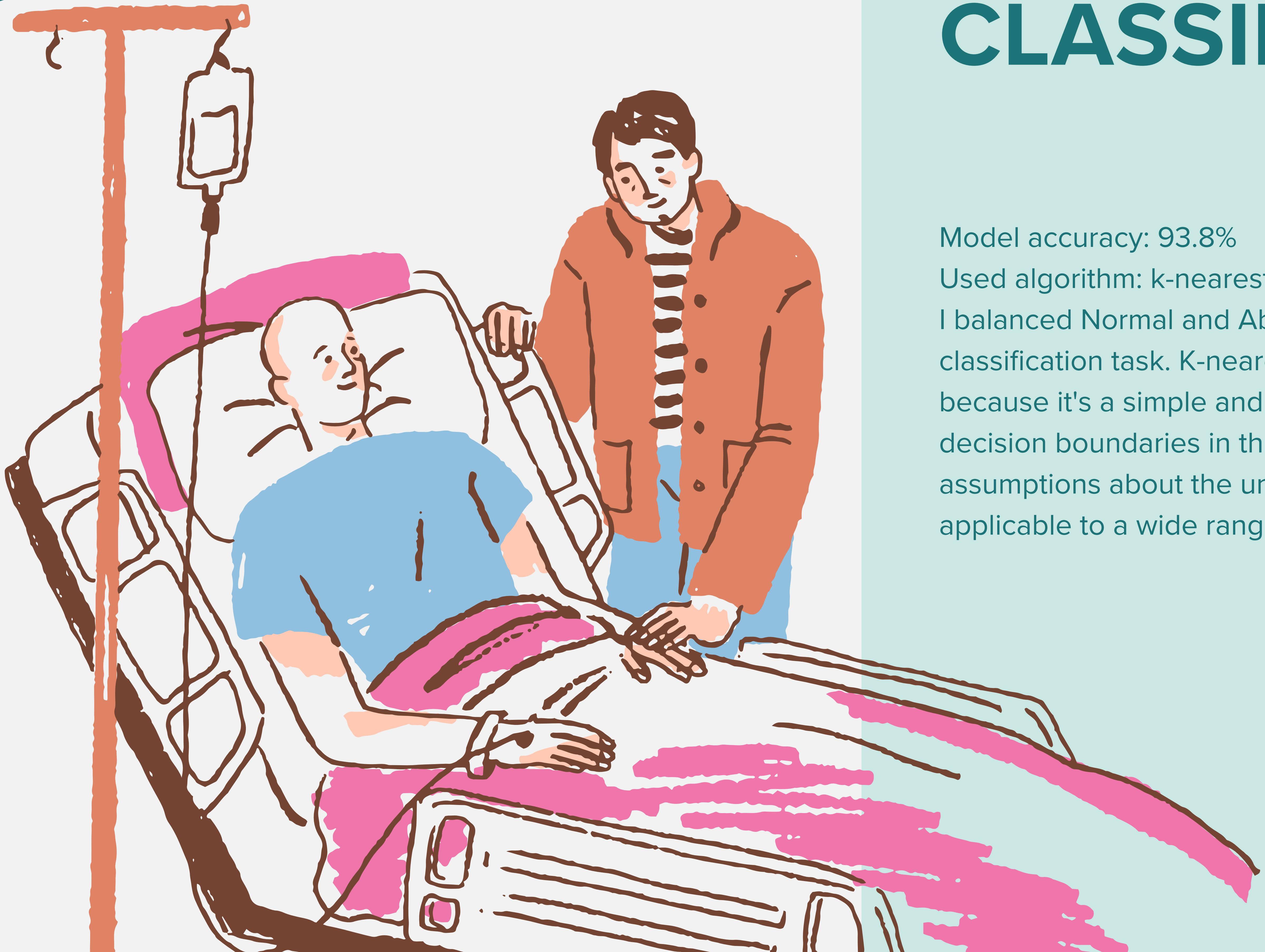
Logistic regression is a versatile approach for classification.- It provides a probabilistic output suitable for both binary and multiclass tasks.- Regularization (L1, L2) can be applied to control model complexity and prevent overfitting.



```
Logistic Regression Model
Accuracy: 0.9050882302552681
Confusion Matrix:
[[14226   353]
 [ 1309  1623]]
Classification Report:
              precision    recall  f1-score   support

      Normal       0.92      0.98      0.94     14579
    Abnormal       0.82      0.55      0.66      2932

    accuracy                           0.91     17511
   macro avg       0.87      0.76      0.80     17511
weighted avg       0.90      0.91      0.90     17511
```

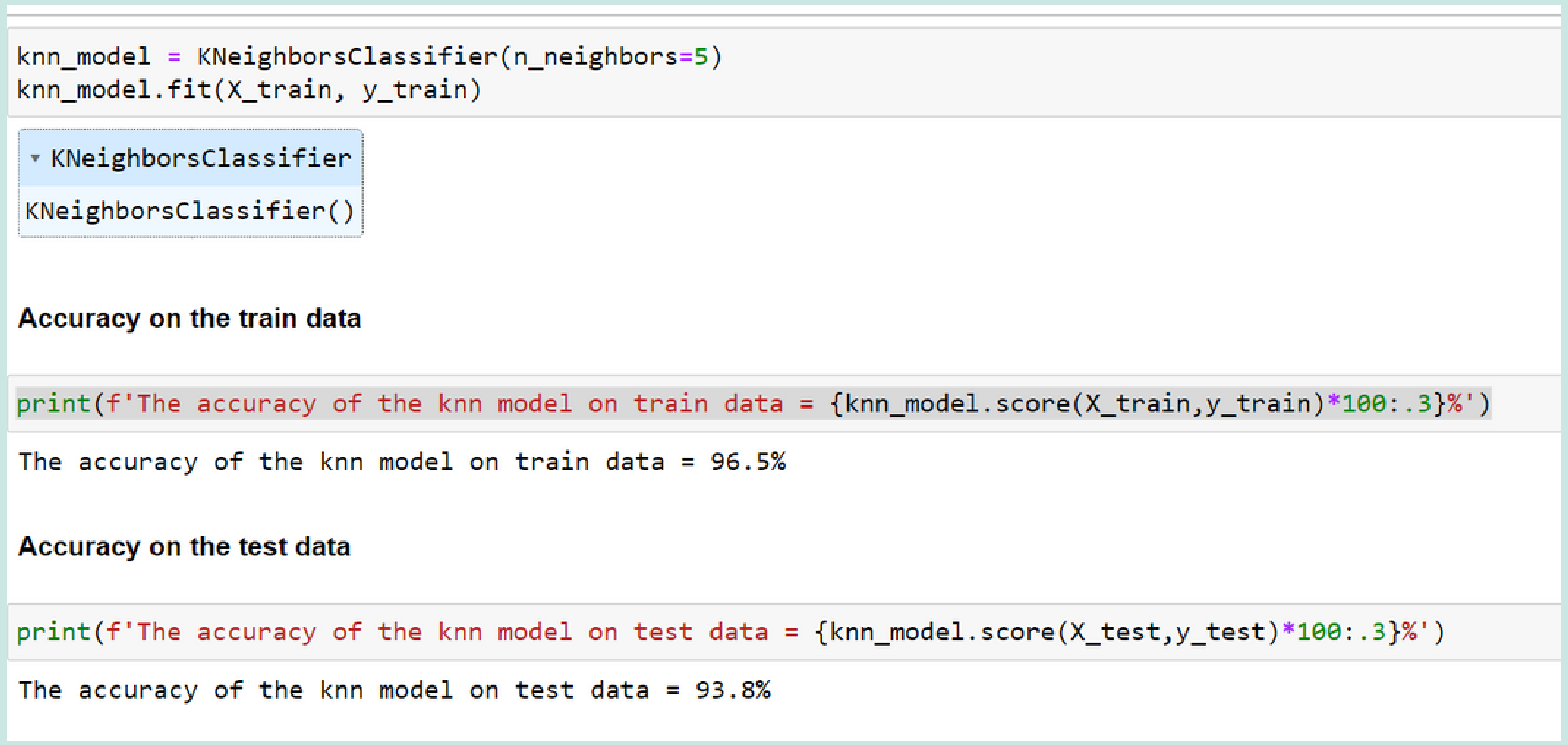


```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import normalize

# Select the MIT dataset for classification (assuming mit_train is already loaded)
# Let's consider 'Normal Beat' (0) vs 'Abnormal Beat' (1)
mit_train['Binary_Class'] = mit_train['Class'].apply(lambda x: 0 if x == 0 else 1)

# Features (X) and target variable (y)
X = mit_train.drop(['Class', 'Binary_Class'], axis=1)
y = mit_train['Binary_Class']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize the data (optional, depending on your use case)
X_train = normalize(X_train, axis=0, norm='max')
X_test = normalize(X_test, axis=0, norm='max')

# Create and fit the logistic regression model
logistic_reg = LogisticRegression(max_iter=1000,random_state=42)
logistic_reg.fit(X_train, y_train)

# Predictions
y_pred = logistic_reg.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred, target_names=['Normal', 'Abnormal'])

print("Logistic Regression Model")
print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_report_str)
```



BINARY CLASSIFICATION

Model accuracy: 93.8%
Used algorithm: k-nearest neighbors (KNN) classifier with k=5
I balanced Normal and Abnormal target data since it is preferred for binary classification task. K-nearest neighbors (KNN) is good for binary classification because it's a simple and intuitive algorithm that can capture complex decision boundaries in the data. Additionally, KNN doesn't make strong assumptions about the underlying data distribution, making it versatile and applicable to a wide range of binary classification problems.



```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

Accuracy on the train data

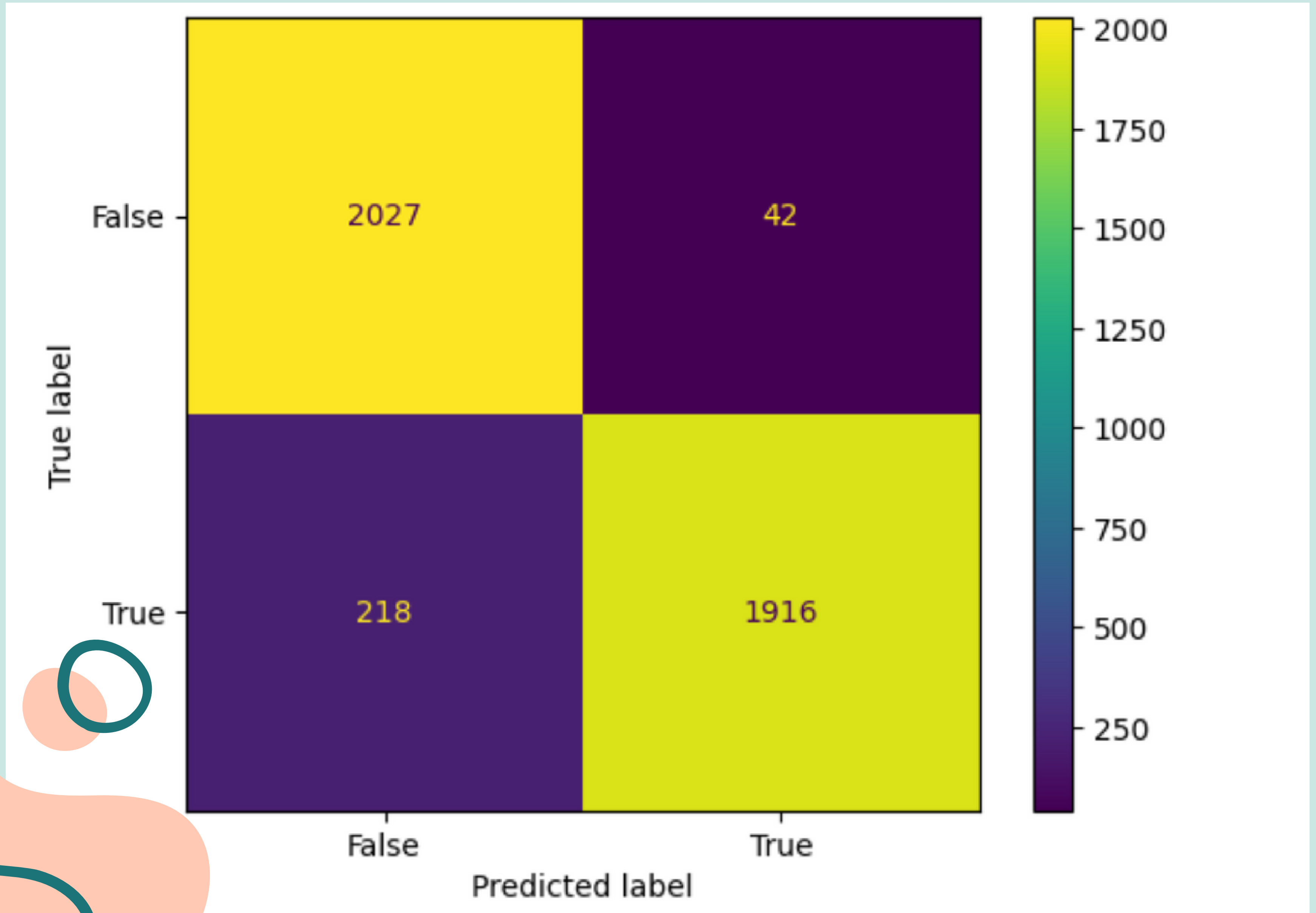
```
print(f'The accuracy of the knn model on train data = {knn_model.score(X_train,y_train)*100:.3}%')
```

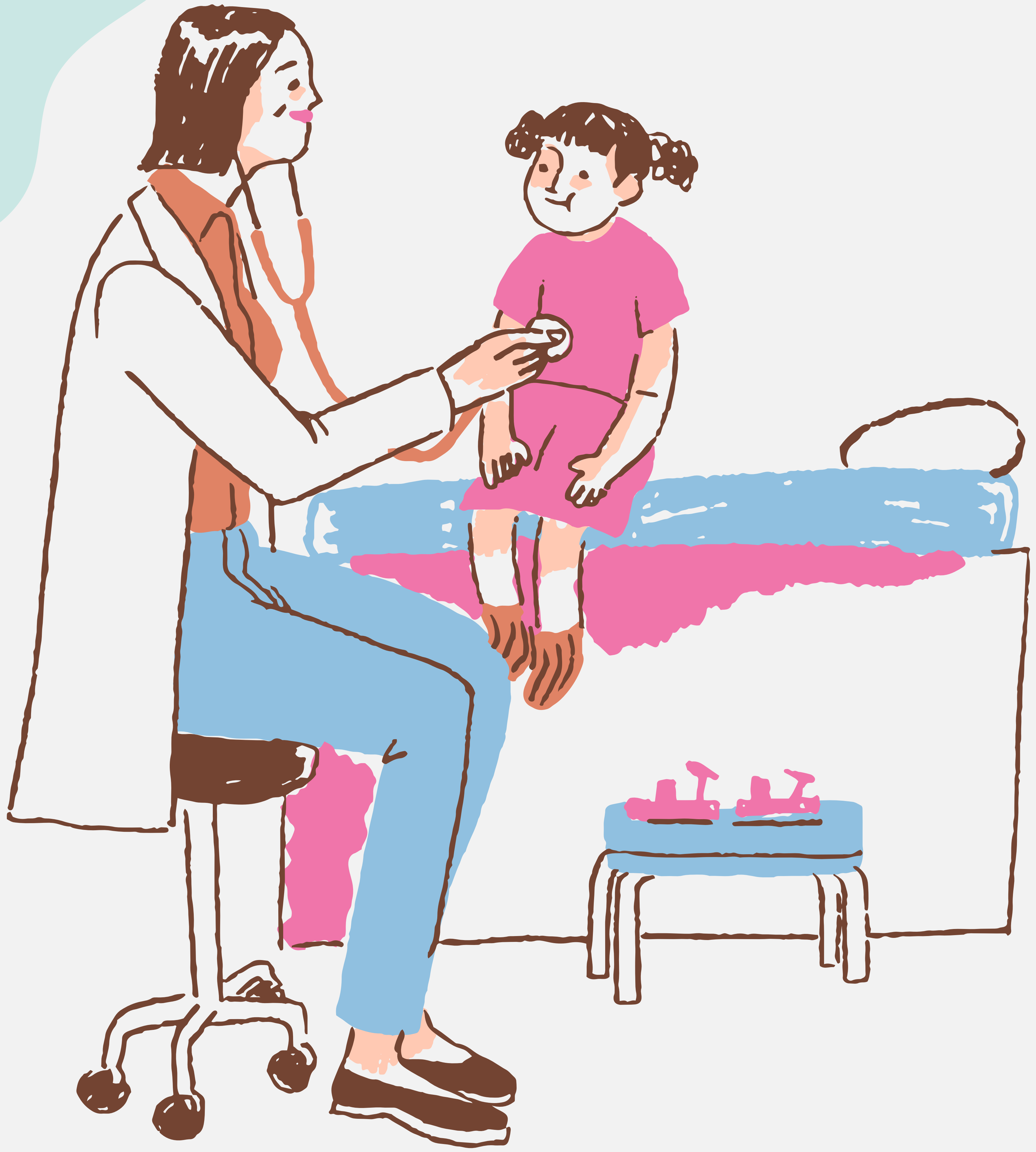
```
The accuracy of the knn model on train data = 96.5%
```

Accuracy on the test data

```
print(f'The accuracy of the knn model on test data = {knn_model.score(X_test,y_test)*100:.3}%')
```

```
The accuracy of the knn model on test data = 93.8%
```



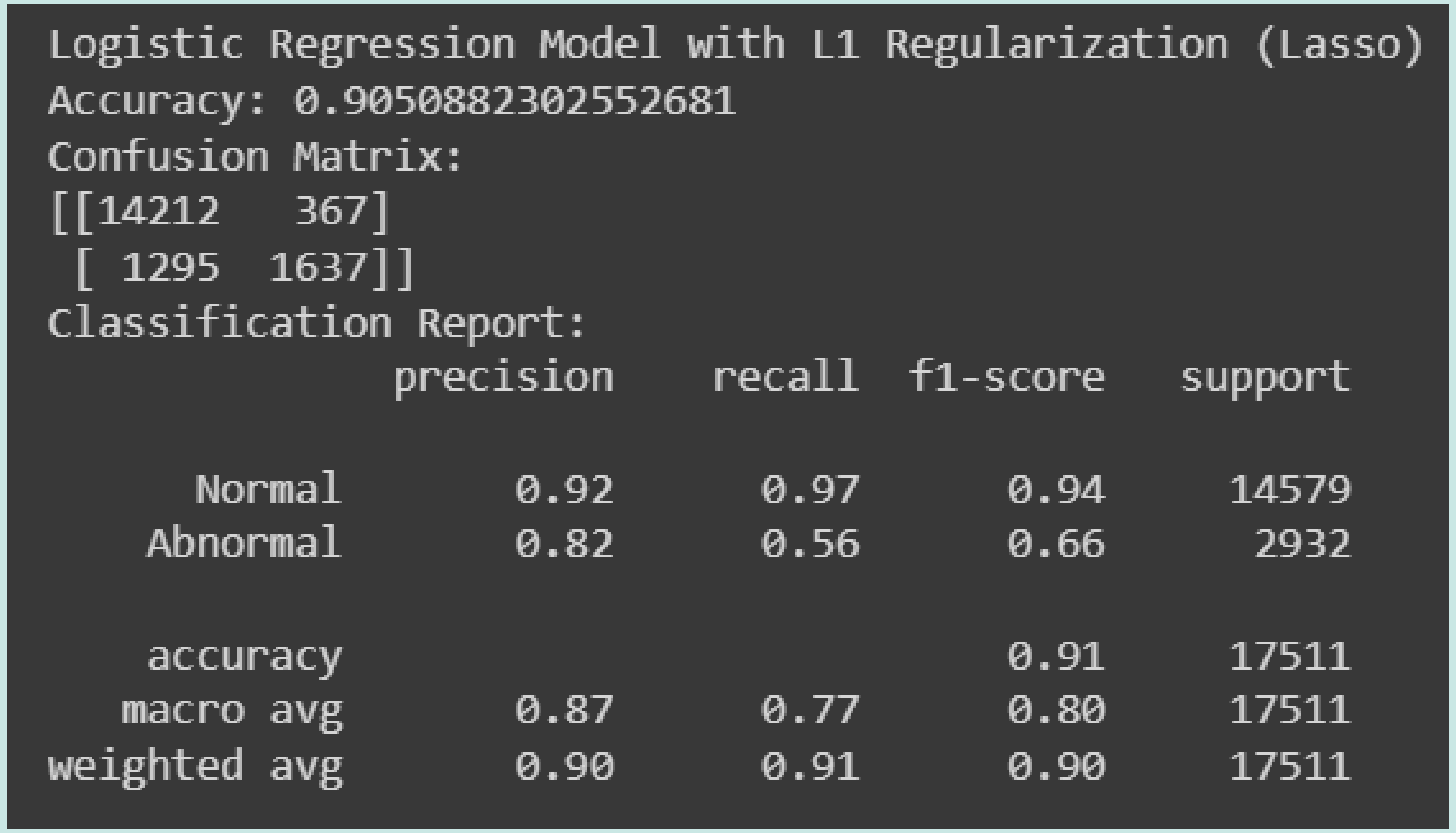


REGULARIZATION

-

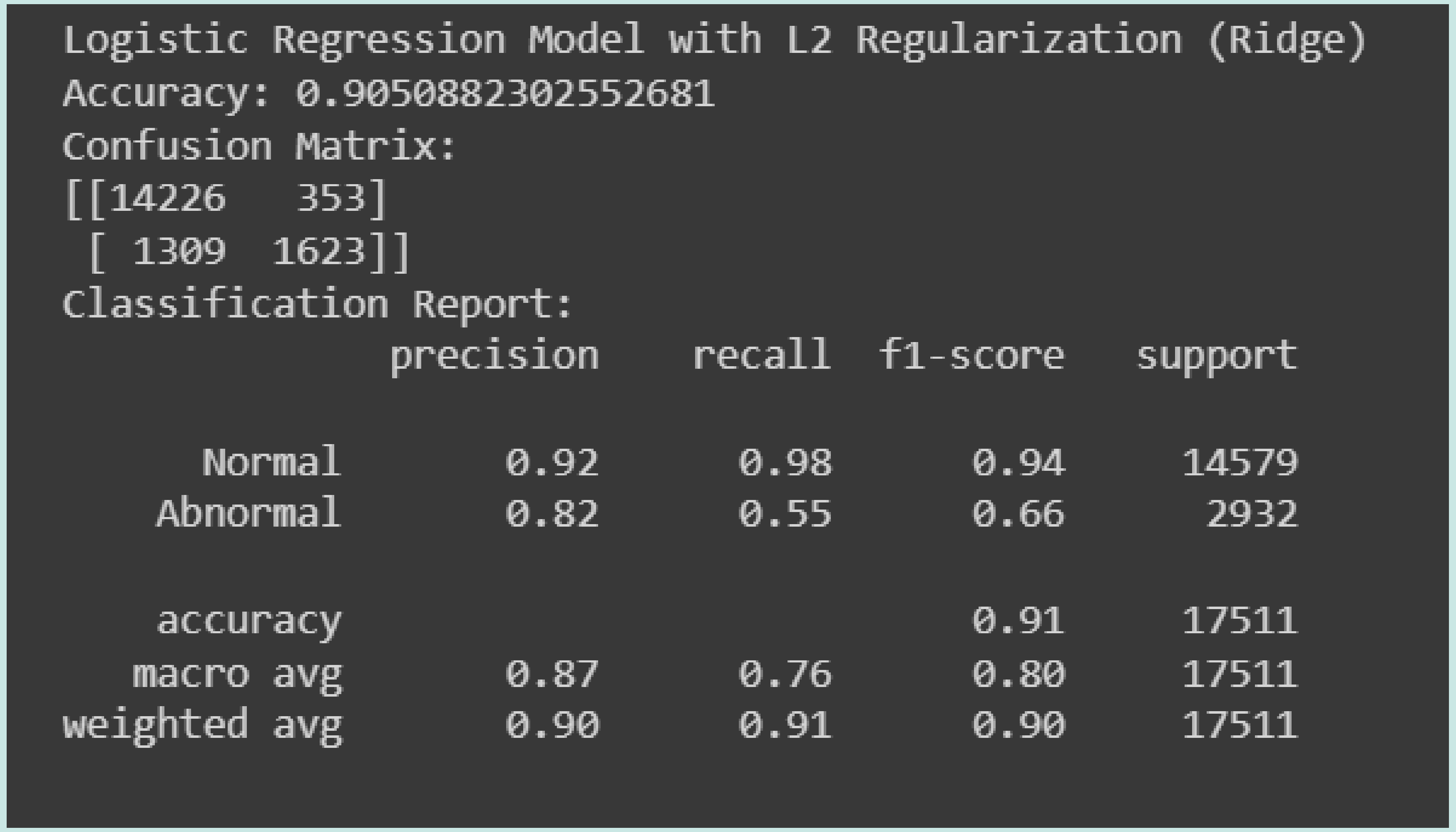
"refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting" (Simplilearn, 2023, para. 3).

LASSO (L1)

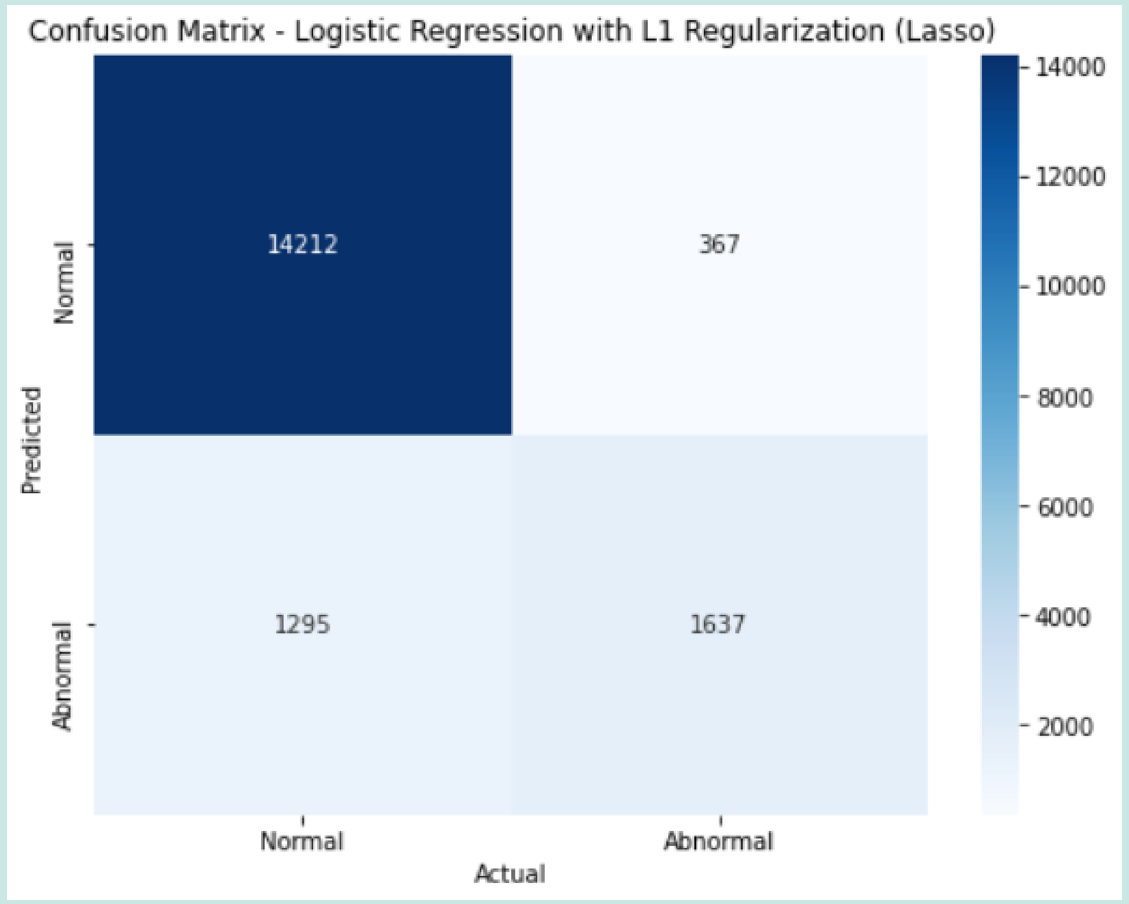


```
Logistic Regression Model with L1 Regularization (Lasso)
Accuracy: 0.9050882302552681
Confusion Matrix:
[[14212   367]
 [ 1295  1637]]
Classification Report:
              precision    recall  f1-score   support

      Normal       0.92      0.97      0.94     14579
    Abnormal       0.82      0.56      0.66      2932

    accuracy                           0.91     17511
   macro avg       0.87      0.77      0.80     17511
weighted avg       0.90      0.91      0.90     17511
```

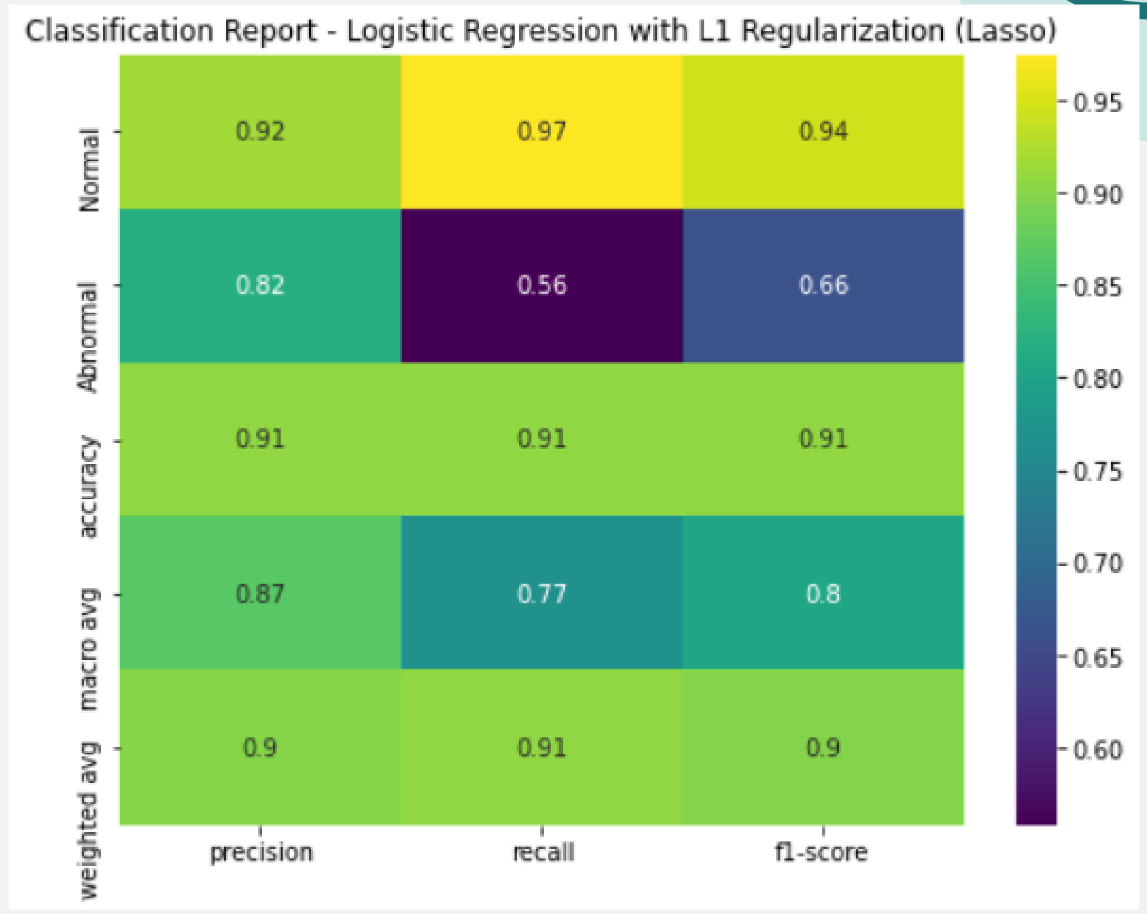
RIDGE (L2)

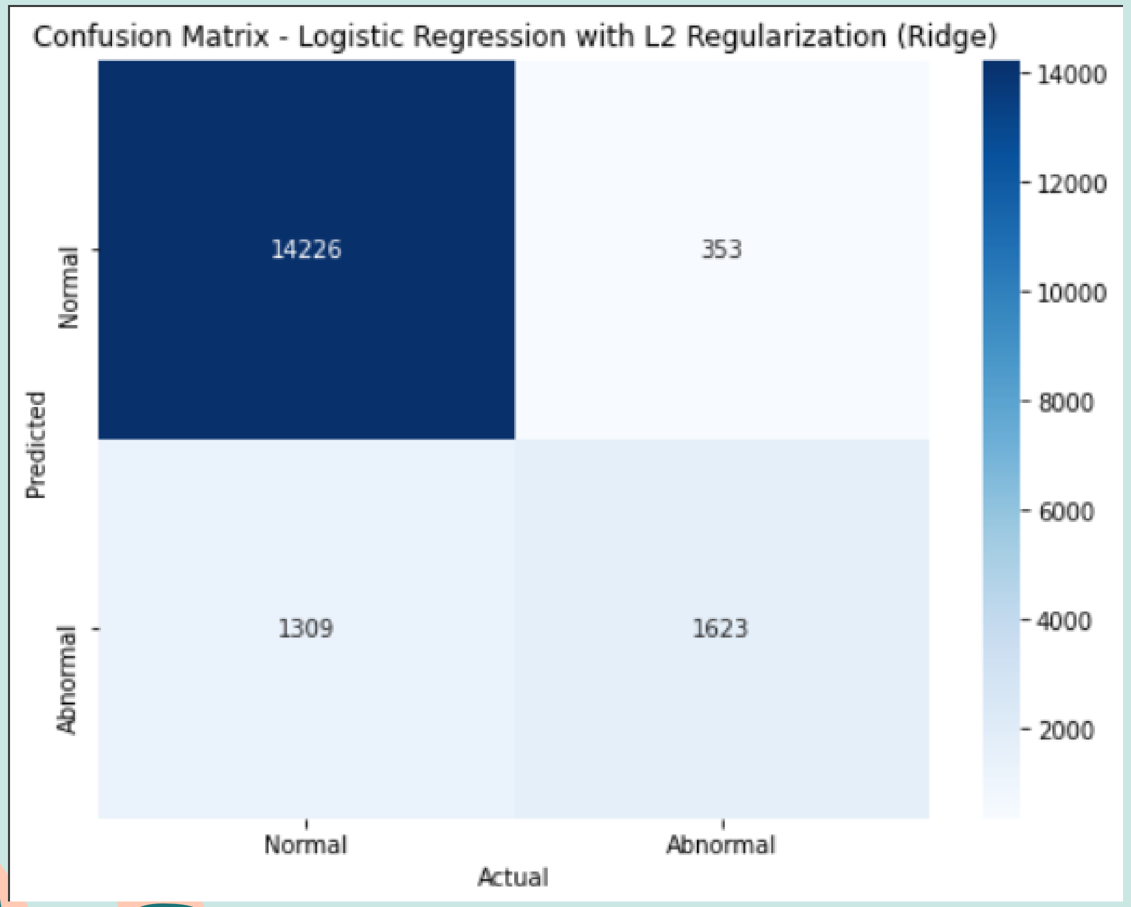


```
Logistic Regression Model with L2 Regularization (Ridge)
Accuracy: 0.9050882302552681
Confusion Matrix:
[[14226   353]
 [ 1309  1623]]
Classification Report:
              precision    recall  f1-score   support

      Normal       0.92      0.98      0.94     14579
    Abnormal       0.82      0.55      0.66      2932

    accuracy                           0.91     17511
   macro avg       0.87      0.76      0.80     17511
weighted avg       0.90      0.91      0.90     17511
```

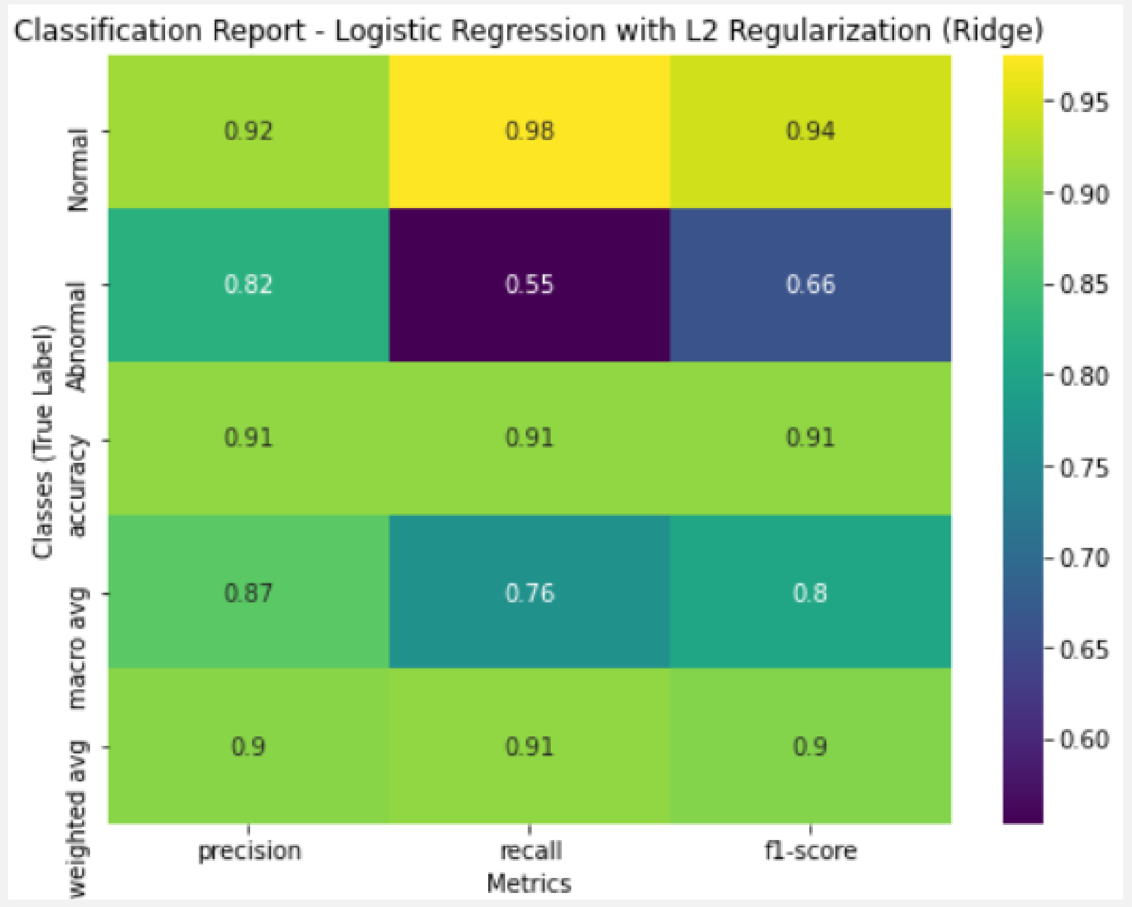

Confusion Matrix - Logistic Regression with L1 Regularization (Lasso)
Predicted
Normal
Abnormal
14212
367
1295
1637
Normal
Abnormal
Actual
14000
12000
10000
8000
6000
4000
2000

LASSO


Classification Report - Logistic Regression with L1 Regularization (Lasso)
Normal
Abnormal
accuracy
macro avg
weighted avg
0.92
0.97
0.94
0.82
0.56
0.66
0.91
0.91
0.91
0.87
0.77
0.8
0.9
0.91
0.9
precision
recall
f1-score
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60


Confusion Matrix - Logistic Regression with L2 Regularization (Ridge)
Predicted
Normal
Abnormal
14226
353
1309
1623
Normal
Abnormal
Actual
14000
12000
10000
8000
6000
4000
2000

RIDGE


Classification Report - Logistic Regression with L2 Regularization (Ridge)
Classes (True Label)
Normal
Abnormal
accuracy
macro avg
weighted avg
0.92
0.98
0.94
0.82
0.55
0.66
0.91
0.91
0.91
0.87
0.76
0.8
0.9
0.91
0.9
precision
recall
f1-score
Metrics
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60

COMPARISON

Linear Regression:
Accuracy: 90.55%
Strengths: Achieves high precision in identifying 'Normal' beats but has a moderate recall for 'Abnormal' beats.
Weaknesses: not designed for classification tasks, and its output may not fit well with binary classification problems.

Logistic Regression:
Accuracy: 90.51%
Strengths: versatile method for binary classification, providing probabilistic output suitable for both binary and multiclass tasks.
Weaknesses: can be affected by outliers.

K-nearest Neighbors (KNN) for Binary Classification:
Accuracy: 93.8%
Strengths: does not make strong assumptions about data distribution, making it versatile.
Weaknesses: can be affected by the scale of features, so feature scaling is often necessary.

COMPARISON

L1 Regularization (Lasso):
Accuracy: 91%
Strengths: effectively perform feature selection by pushing the coefficients of irrelevant features to zero.
Weaknesses: can produce a sparse solution with sharp changes in coefficients, making the model less stable and interpretable.

L2 Regularization (Ridge):
Accuracy: 91%
Strengths: encourages small but non-zero coefficients for all features.
Weaknesses: may not prevent overfitting.

Conclusion

QUESTIONS?

I hope not, else, ask creators of the presentation in person!

With all love, Nurzhas, Dilnura, Aktoty