```python
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
```

```
pip install ucimlrepo
```

```
     Collecting ucimlrepo
       Downloading ucimlrepo-0.0.3-py3-none-any.whl (7.0 kB)
     Installing collected packages: ucimlrepo
     Successfully installed ucimlrepo-0.0.3
```

```python
from ucimlrepo import fetch_ucirepo

# fetch dataset
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# data (as pandas dataframes)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# metadata
print(breast_cancer_wisconsin_diagnostic.metadata)

# variable information
print(breast_cancer_wisconsin_diagnostic.variables)
```

```
     9      concave_points1  Feature   Continuous        None        None  None
     10           symmetry1  Feature   Continuous        None        None  None
     11  fractal_dimension1  Feature   Continuous        None        None  None
     12             radius2  Feature   Continuous        None        None  None
     13            texture2  Feature   Continuous        None        None  None
     14          perimeter2  Feature   Continuous        None        None  None
     15               area2  Feature   Continuous        None        None  None
     16         smoothness2  Feature   Continuous        None        None  None
     17        compactness2  Feature   Continuous        None        None  None
     18          concavity2  Feature   Continuous        None        None  None
     19     concave_points2  Feature   Continuous        None        None  None
     20           symmetry2  Feature   Continuous        None        None  None
     21  fractal_dimension2  Feature   Continuous        None        None  None
     22             radius3  Feature   Continuous        None        None  None
     23            texture3  Feature   Continuous        None        None  None
     24          perimeter3  Feature   Continuous        None        None  None
     25               area3  Feature   Continuous        None        None  None
     26         smoothness3  Feature   Continuous        None        None  None
     27        compactness3  Feature   Continuous        None        None  None
     28          concavity3  Feature   Continuous        None        None  None
     29     concave_points3  Feature   Continuous        None        None  None
     30           symmetry3  Feature   Continuous        None        None  None
     31  fractal_dimension3  Feature   Continuous        None        None  None

        missing_values
     0              no
     1              no
     2              no
     3              no
     4              no
     5              no
     6              no
     7              no
     8              no
     9              no
     10             no
     11             no
     12             no
     13             no
     14             no
     15             no
     16             no
     17             no
     18             no
     19             no
     20             no
     21             no
     22             no
     23             no
     24             no
     25             no
     26             no
     27             no
     28             no
     29             no
     30             no
     31             no
```

```
data = pd.read_csv('wdbc.data')
```

```
data.columns = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness', 'Mean Compactness',
```

```
data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})
```

```
data.head()
```

	ID	Diagnosis	Mean Radius	Mean Texture	Mean Perimeter	Mean Area	Mean Smoothness	Mean Compactness
0	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864
1	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990
2	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390
3	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280
4	843786	1	12.45	15.70	82.57	477.1	0.12780	0.17000

5 rows × 32 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568 entries, 0 to 567
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   ID                       568 non-null    int64  
 1   Diagnosis                568 non-null    int64  
 2   Mean Radius              568 non-null    float64
 3   Mean Texture             568 non-null    float64
 4   Mean Perimeter           568 non-null    float64
 5   Mean Area                568 non-null    float64
 6   Mean Smoothness          568 non-null    float64
 7   Mean Compactness         568 non-null    float64
 8   Mean Concavity           568 non-null    float64
 9   Mean Concave Points      568 non-null    float64
 10  Mean Symmetry            568 non-null    float64
 11  Mean Fractal Dimension   568 non-null    float64
 12  SE Radius                568 non-null    float64
 13  SE Texture               568 non-null    float64
 14  SE Perimeter             568 non-null    float64
 15  SE Area                  568 non-null    float64
 16  SE Smoothness            568 non-null    float64
 17  SE Compactness           568 non-null    float64
 18  SE Concavity             568 non-null    float64
 19  SE Concave Points        568 non-null    float64
 20  SE Symmetry              568 non-null    float64
 21  SE Fractal Dimension     568 non-null    float64
 22  Worst Radius             568 non-null    float64
 23  Worst Texture            568 non-null    float64
 24  Worst Perimeter          568 non-null    float64
 25  Worst Area               568 non-null    float64
 26  Worst Smoothness         568 non-null    float64
 27  Worst Compactness        568 non-null    float64
 28  Worst Concavity          568 non-null    float64
 29  Worst Concave Points     568 non-null    float64
 30  Worst Symmetry           568 non-null    float64
 31  Worst Fractal Dimension  568 non-null    float64
dtypes: float64(30), int64(2)
memory usage: 142.1 KB
```

DATA NORMALIZATION

```
X = data.drop(['ID','Diagnosis'],axis=1)
Y =  data['Diagnosis']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, random_state=0)
```

```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

```
((454, 30), (114, 30), (454,), (114,))
```

```
model = Sequential()
model.add(Dense(16, input_dim=30, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Make predictions on the test data
y_pred = model.predict(X_test)

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

# Print the binary predictions
print(y_pred_binary)
```

```
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]]
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.fit(X_train, Y_train, epochs=100, batch_size=10)
```

```
Epoch 77/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1401 - accuracy: 0.9537
Epoch 78/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1509 - accuracy: 0.9427
Epoch 79/100
46/46 [==============================] - 0s 2ms/step - loss: 0.2025 - accuracy: 0.9295
Epoch 80/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1416 - accuracy: 0.9405
Epoch 81/100
46/46 [==============================] - 0s 2ms/step - loss: 0.2272 - accuracy: 0.9207
Epoch 82/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1395 - accuracy: 0.9449
Epoch 83/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1579 - accuracy: 0.9427
Epoch 84/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1425 - accuracy: 0.9493
Epoch 85/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1531 - accuracy: 0.9405
Epoch 86/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1285 - accuracy: 0.9515
Epoch 87/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1301 - accuracy: 0.9515
Epoch 88/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1358 - accuracy: 0.9493
Epoch 89/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1584 - accuracy: 0.9295
Epoch 90/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1343 - accuracy: 0.9559
Epoch 91/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1320 - accuracy: 0.9559
Epoch 92/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1386 - accuracy: 0.9405
Epoch 93/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1577 - accuracy: 0.9361
Epoch 94/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1430 - accuracy: 0.9471
Epoch 95/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1282 - accuracy: 0.9581
Epoch 96/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1767 - accuracy: 0.9185
Epoch 97/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1385 - accuracy: 0.9449
Epoch 98/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1359 - accuracy: 0.9537
Epoch 99/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1534 - accuracy: 0.9295
Epoch 100/100
46/46 [==============================] - 0s 2ms/step - loss: 0.1337 - accuracy: 0.9405
<keras.src.callbacks.History at 0x79d08010e050>
```

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(acccuracy_score(Y_test, y_pred))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-91-ca41f24a880d> in <cell line: 2>()
      1 from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
----> 2 print(acccuracy_score(Y_test, y_pred))

NameError: name 'acccuracy_score' is not defined
```

LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=5000)
```

```
model.fit(X_train,Y_train)
```

```
▾        LogisticRegression
LogisticRegression(max_iter=5000)
```

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

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           B       0.93      0.99      0.96        71
```

```
                   M       0.97      0.88      0.93        43

            accuracy                           0.95       114
           macro avg       0.95      0.93      0.94       114
        weighted avg       0.95      0.95      0.95       114
```

```
confusion_matrix(Y_test,y_pred)
```

```
     array([[70,  1],
            [ 5, 38]])
```

```
accuracy_score(Y_test, y_pred)
```

```
     0.9473684210526315
```

SUPPORT VECTOR MACHINE

```
from sklearn.svm import SVC
svm = SVC()
```

```
svm.fit(X_train,Y_train)
```

```
▾ SVC
SVC()
```

```
y_preds = svm.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(classification_report(Y_test, y_preds))
```

```
                   precision    recall  f1-score   support

                   B       0.87      1.00      0.93        71
                   M       1.00      0.74      0.85        43

            accuracy                           0.90       114
           macro avg       0.93      0.87      0.89       114
        weighted avg       0.92      0.90      0.90       114
```

```
confusion_matrix(Y_test,y_preds)
```

```
     array([[71,  0],
            [11, 32]])
```

```
accuracy_score(Y_test,y_preds)
```

```
     0.9035087719298246
```