ICP-5 REPORT

```
[1] from google.colab import drive
    drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
[2] path_to_csv = '/content/gdrive/My Drive/diabetes.csv'
```

```
[4] import keras
    import pandas as pd
    import numpy as np
    from keras.models import Sequential
    from keras.layers import Dense
    from sklearn.model_selection import train_test_split

    # Load dataset
    dataset = pd.read_csv(path_to_csv, header=None).values

    # Split the dataset into training and testing sets
    X_train, X_test, Y_train, Y_test = train_test_split(dataset[:, 0:8], dataset[:, 8], test_size=0.25, random_state=87)

    # Set random seed for reproducibility
    np.random.seed(155)
```

✓ 12s completed at 7:34 PM

```
# Create a Sequential model
model = Sequential()

# Add Dense layers with 'relu' activation for hidden layers
model.add(Dense(20, input_dim=8, activation='relu'))  # First hidden layer
model.add(Dense(16, activation='relu'))  # Second hidden layer
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))   # Third hidden layer

# Add output layer with 'sigmoid' activation
model.add(Dense(1, activation='sigmoid'))

# Compile the model using binary crossentropy and adam optimizer
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])

# Train the model
model_fitted = model.fit(X_train, Y_train, epochs=100, initial_epoch=0)

# Print model summary and evaluate accuracy on the test set
print(model.summary())
print(model.evaluate(X_test, Y_test))
```

+ Code + Text

13s

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 0.7364 - loss: 0.5236
Epoch 98/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 0.7584 - loss: 0.4937
Epoch 99/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 0.7512 - loss: 0.5181
Epoch 100/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - acc: 0.7490 - loss: 0.5218
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 20)	180
dense_5 (Dense)	(None, 16)	336
dense_6 (Dense)	(None, 12)	204
dense_7 (Dense)	(None, 8)	104
dense_8 (Dense)	(None, 1)	9

```
 Total params: 2,501 (9.77 KB)
 Trainable params: 833 (3.25 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,668 (6.52 KB)
None
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 0.7258 - loss: 0.5765
[0.5843656659126282, 0.71875]
```

```python
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load dataset
dataset = pd.read_csv(path_to_csv, header=None).values

# Split the dataset into features (X) and target (Y)
X = dataset[:, 0:8]
Y = dataset[:, 8]

# Normalize the feature data
sc = StandardScaler()
X = sc.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)

# Set random seed for reproducibility
np.random.seed(155)
```

```python
# Create a Sequential model
model = Sequential()

# Add Dense layers with 'relu' activation for hidden layers
model.add(Dense(20, input_dim=8, activation='relu'))  # First hidden layer
model.add(Dense(16, activation='relu'))  # Second hidden layer
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
# Add output layer with 'sigmoid' activation
model.add(Dense(1, activation='sigmoid'))

# Compile the model using binary crossentropy and adam optimizer
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])

# Train the model
model_fitted = model.fit(X_train, Y_train, epochs=100, initial_epoch=0)

# Print model summary and evaluate accuracy on the test set
print(model.summary())
print(model.evaluate(X_test, Y_test))
```

```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.9262 - loss: 0.2247
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_18 (Dense)	(None, 20)	180
dense_19 (Dense)	(None, 16)	336
dense_20 (Dense)	(None, 12)	204
dense_21 (Dense)	(None, 8)	104
dense_22 (Dense)	(None, 1)	9

```
 Total params: 2,501 (9.77 KB)
 Trainable params: 833 (3.25 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,668 (6.52 KB)
None
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.7348 - loss: 0.6546
[0.6039299368858337, 0.7552083134651184]
```

```
15] path_to_csv2 = '/content/gdrive/My Drive/breastcancer.csv'
```

```
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
#from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

dataset = pd.read_csv(path_to_csv2, header=None).values

X = dataset[1:, 2:-1]  # Features
Y = dataset[1:, -1]   # Labels (M or B)

# Convert labels to binary format
Y = np.where(Y == 'M', 1, 0)  # M -> 1, B -> 0

#Convert to numeric
X = X.astype(np.float64) # Convert X to numeric
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.25, random_state=87)

#Normalizing the data
#sc = StandardScaler()
#X_train = sc.fit_transform(X_train)
#X_test = sc.transform(X_test)


np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(15, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu')) # hidden layer
my_first_nn.add(Dense(5, activation='relu')) # hidden layer

my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test,Y_test))
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 2.2242e-09
Epoch 100/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 1.0000 - loss: 1.0543e-09
Model: "sequential_10"
```

Layer (type)	Output Shape	Param #
dense_48 (Dense)	(None, 20)	620
dense_49 (Dense)	(None, 15)	315
dense_50 (Dense)	(None, 10)	160
dense_51 (Dense)	(None, 5)	55
dense_52 (Dense)	(None, 1)	6

```
 Total params: 3,470 (13.56 KB)
 Trainable params: 1,156 (4.52 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,314 (9.04 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 1.0000 - loss: 6.9380e-09
[1.3352716266012976e-08, 1.0]
```

```python
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
#from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

dataset = pd.read_csv(path_to_csv2, header=None).values

X = dataset[1:, 2:-1]  # Features
Y = dataset[1:, -1]   # Labels (M or B)

# Convert labels to binary format
Y = np.where(Y == 'M', 1, 0)  # M -> 1, B -> 0

#Convert to numeric
X = X.astype(np.float64) # Convert X to numeric

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.25, random_state=87)
```

```
#Normalizing the data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(15, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu')) # hidden layer
my_first_nn.add(Dense(5, activation='relu')) # hidden layer

my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test,Y_test))
```

```
Epoch 99/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - acc: 1.0000 - loss: 2.3211e-05
Epoch 100/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 1.8389e-05
Model: "sequential_11"
```

Layer (type)	Output Shape	Param #
dense_53 (Dense)	(None, 20)	620
dense_54 (Dense)	(None, 15)	315
dense_55 (Dense)	(None, 10)	160
dense_56 (Dense)	(None, 5)	55
dense_57 (Dense)	(None, 1)	6

```
 Total params: 3,470 (13.56 KB)
 Trainable params: 1,156 (4.52 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,314 (9.04 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 1.7065e-05
[1.4555774214386474e-05, 1.0]
```

My Github link : https://github.com/PraveenDondapati/bda.git

```python
import matplotlib.pyplot as plt
import pandas as pd
data1 = pd.read_csv(path_to_csv2)
# Grouping data by Diagnosis
diagnosis_counts = data1.groupby('diagnosis')['diagnosis'].count()
# Plotting the pie chart
plt.figure(figsize=(2,2))
plt.pie(diagnosis_counts, labels=diagnosis_counts.index, autopct='%1.1f%%', startangle=10, colors=['#009299','#03bf00'])
# Adding title
plt.title('Distribution of Patients by Diagnosis', fontsize=8)
# Display the plot
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```