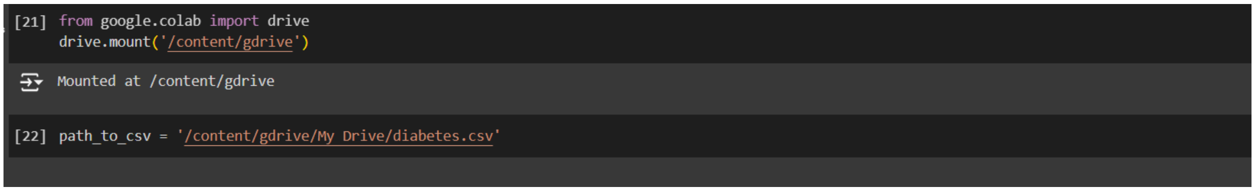
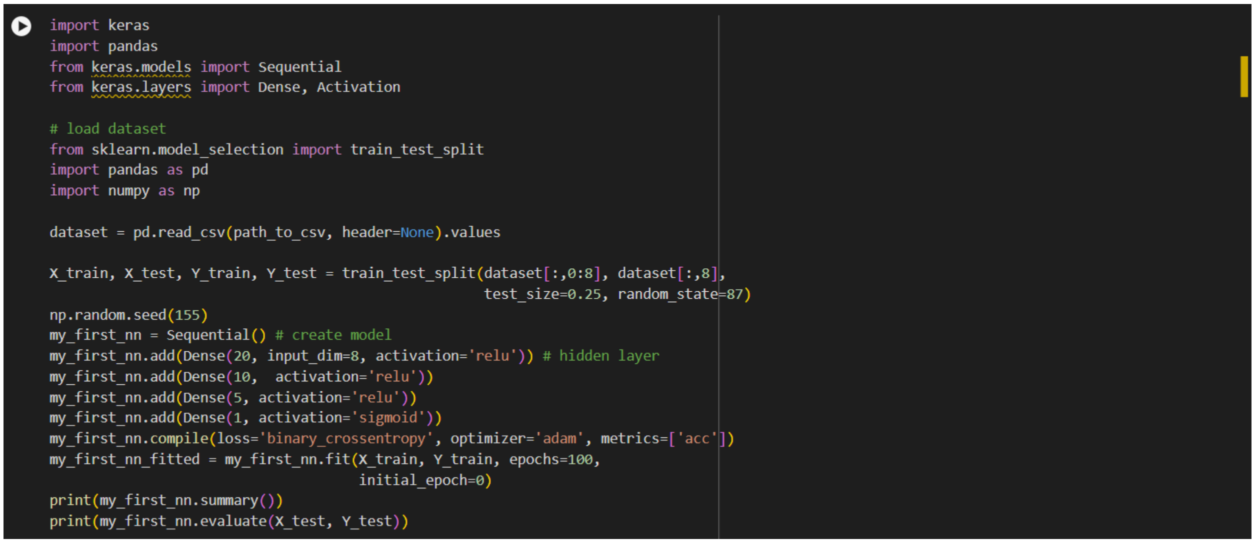
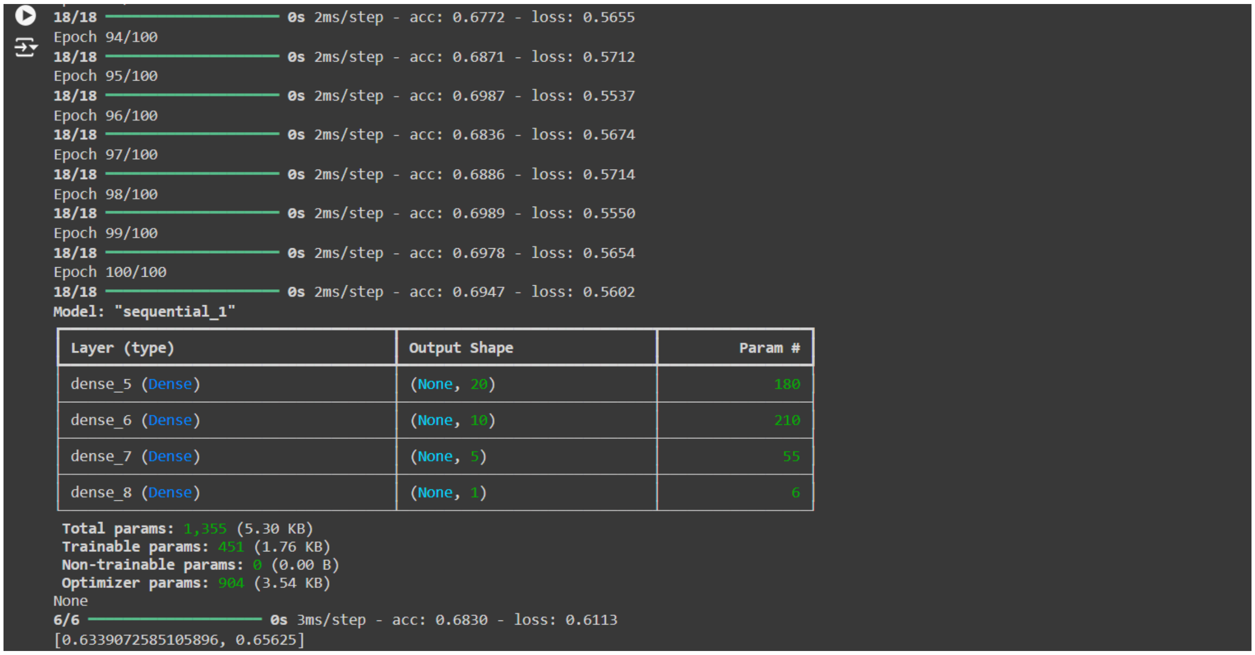
Icp5



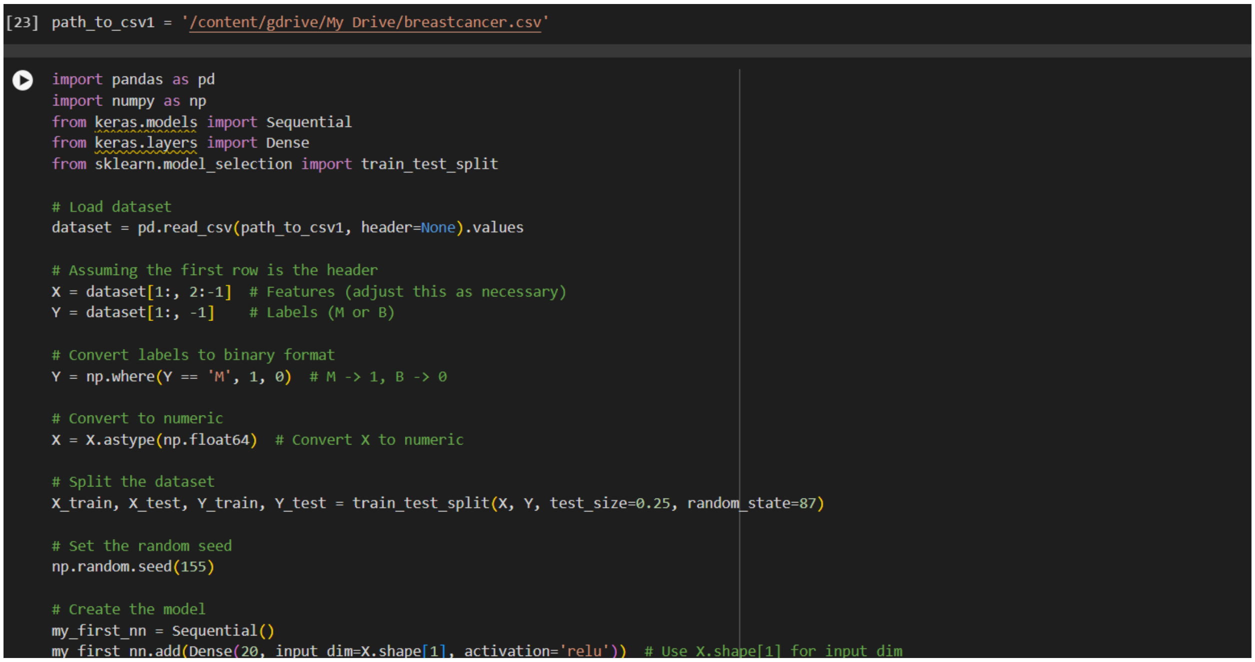
```
[21] from google.colab import drive
     drive.mount('/content/gdrive')

     Mounted at /content/gdrive

[22] path_to_csv = '/content/gdrive/My Drive/diabetes.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
import pandas as pd
import numpy as np

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

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(10,  activation='relu'))
my_first_nn.add(Dense(5, activation='relu'))
my_first_nn.add(Dense(1, activation='sigmoid'))
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))
```

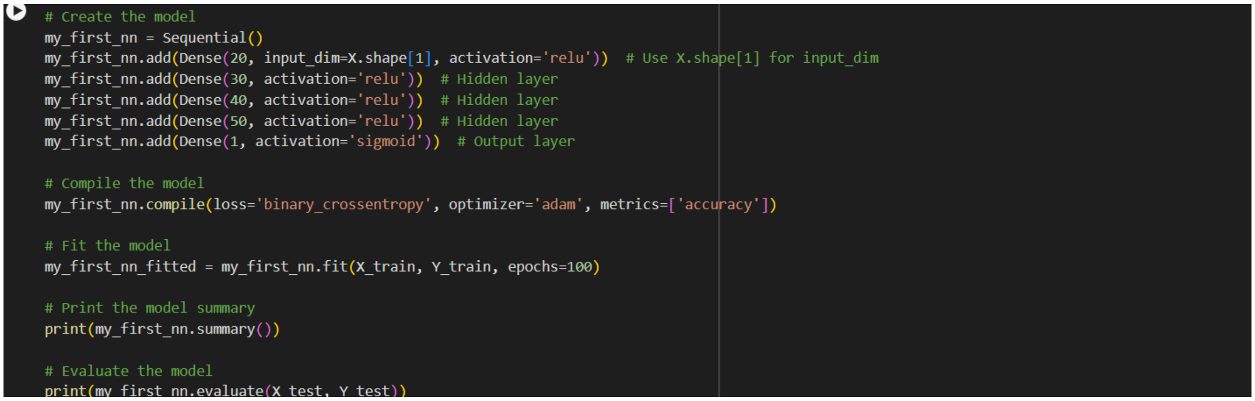


```
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6772 - loss: 0.5655
Epoch 94/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6871 - loss: 0.5712
Epoch 95/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6987 - loss: 0.5537
Epoch 96/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6836 - loss: 0.5674
Epoch 97/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6886 - loss: 0.5714
Epoch 98/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6989 - loss: 0.5550
Epoch 99/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6978 - loss: 0.5654
Epoch 100/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6947 - loss: 0.5602
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ dense_5 (Dense)                      │ (None, 20)                  │             180 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_6 (Dense)                      │ (None, 10)                  │             210 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_7 (Dense)                      │ (None, 5)                   │              55 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_8 (Dense)                      │ (None, 1)                   │               6 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 1,355 (5.30 KB)
 Trainable params: 451 (1.76 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 904 (3.54 KB)
None
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 0.6830 - loss: 0.6113
[0.6339072585105896, 0.65625]
```

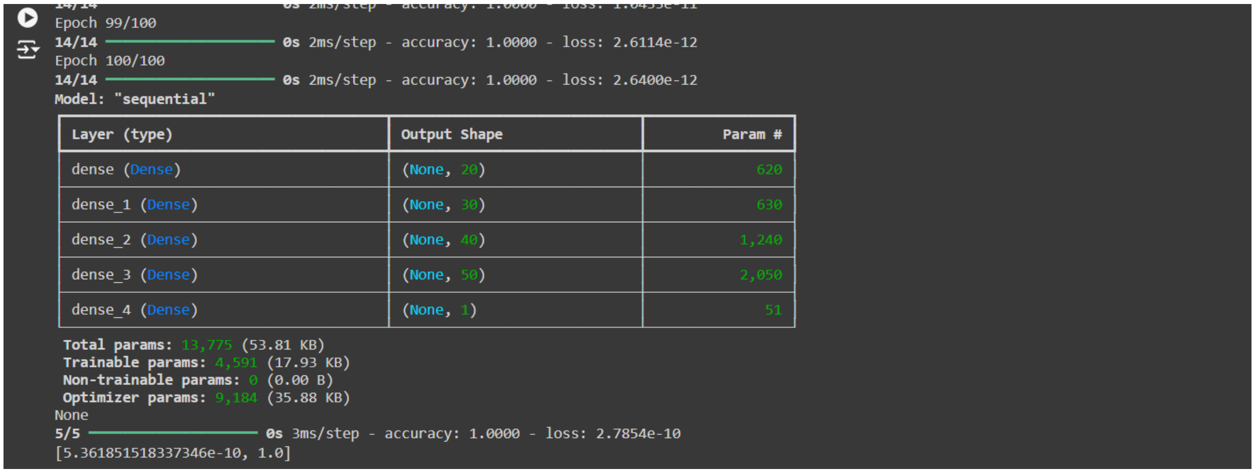


```
[23] path_to_csv1 = '/content/gdrive/My Drive/breastcancer.csv'
```

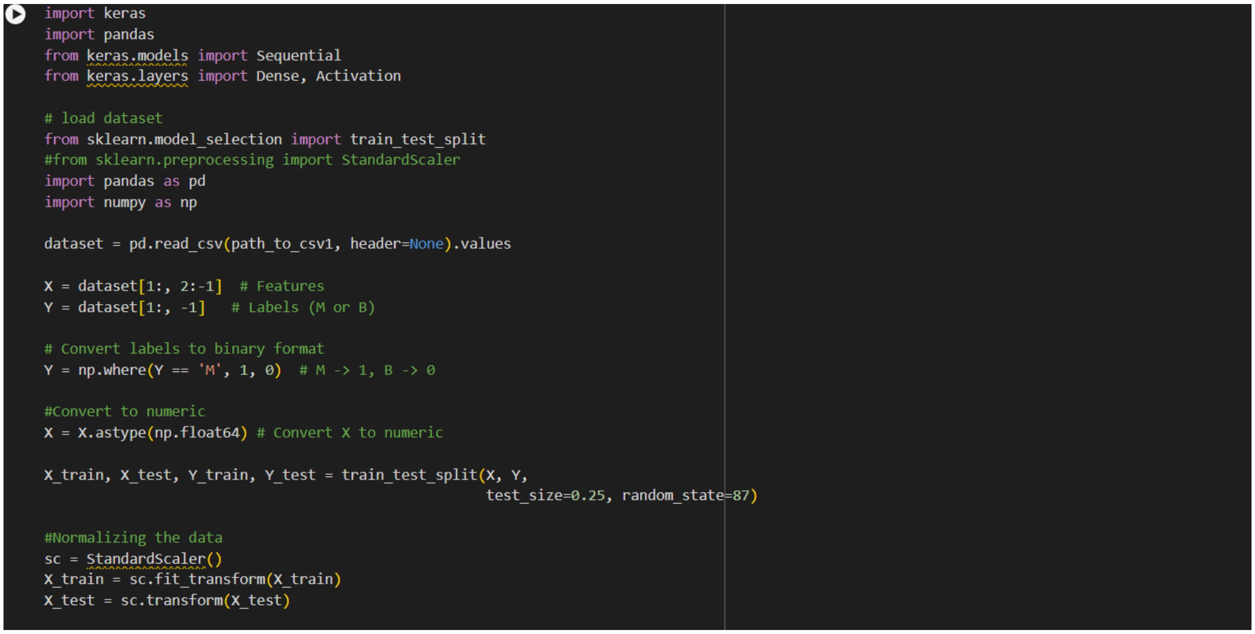
```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

# Load dataset
dataset = pd.read_csv(path_to_csv1, header=None).values

# Assuming the first row is the header
X = dataset[1:, 2:-1]  # Features (adjust this as necessary)
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

# Split the dataset
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)

# Set the random seed
np.random.seed(155)

# Create the model
my_first_nn = Sequential()
my_first_nn.add(Dense(20, input_dim=X.shape[1], activation='relu'))  # Use X.shape[1] for input_dim
```



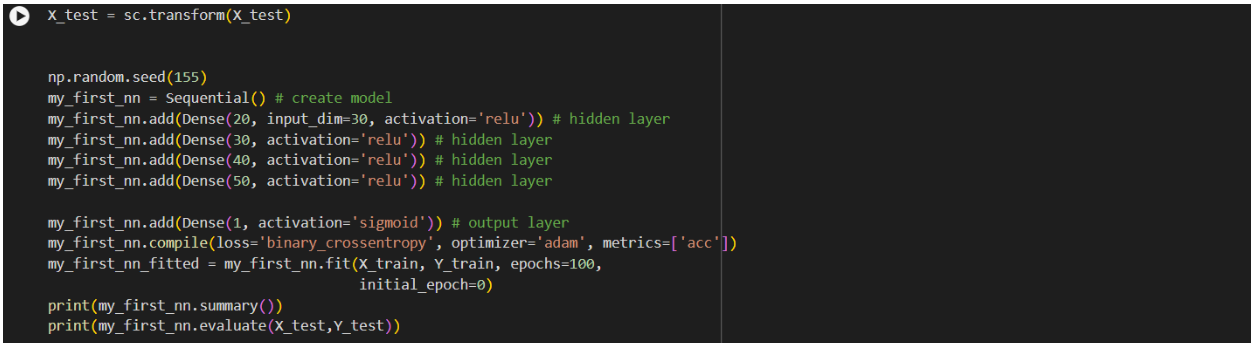
```
# Create the model
my_first_nn = Sequential()
my_first_nn.add(Dense(20, input_dim=X.shape[1], activation='relu'))  # Use X.shape[1] for input_dim
my_first_nn.add(Dense(30, activation='relu'))  # Hidden layer
my_first_nn.add(Dense(40, activation='relu'))  # Hidden layer
my_first_nn.add(Dense(50, activation='relu'))  # Hidden layer
my_first_nn.add(Dense(1, activation='sigmoid'))  # Output layer

# Compile the model
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100)

# Print the model summary
print(my_first_nn.summary())

# Evaluate the model
print(my_first_nn.evaluate(X_test, Y_test))
```



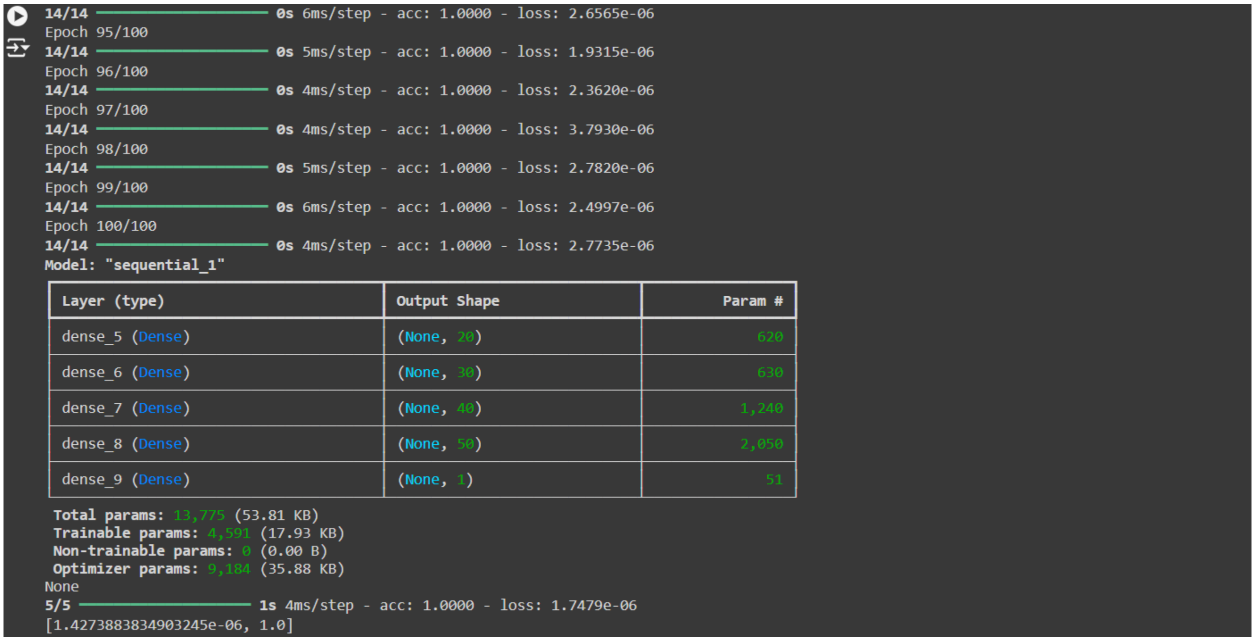
```
14/14                    0s 2ms/step - accuracy: 1.0000 - loss: 1.0433e-11
Epoch 99/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 2.6114e-12
Epoch 100/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 2.6400e-12
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ dense (Dense)                        │ (None, 20)                  │             620 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 30)                  │             630 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 40)                  │           1,240 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 50)                  │           2,050 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 1)                   │              51 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 13,775 (53.81 KB)
 Trainable params: 4,591 (17.93 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 9,184 (35.88 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 2.7854e-10
[5.361851518337346e-10, 1.0]
```



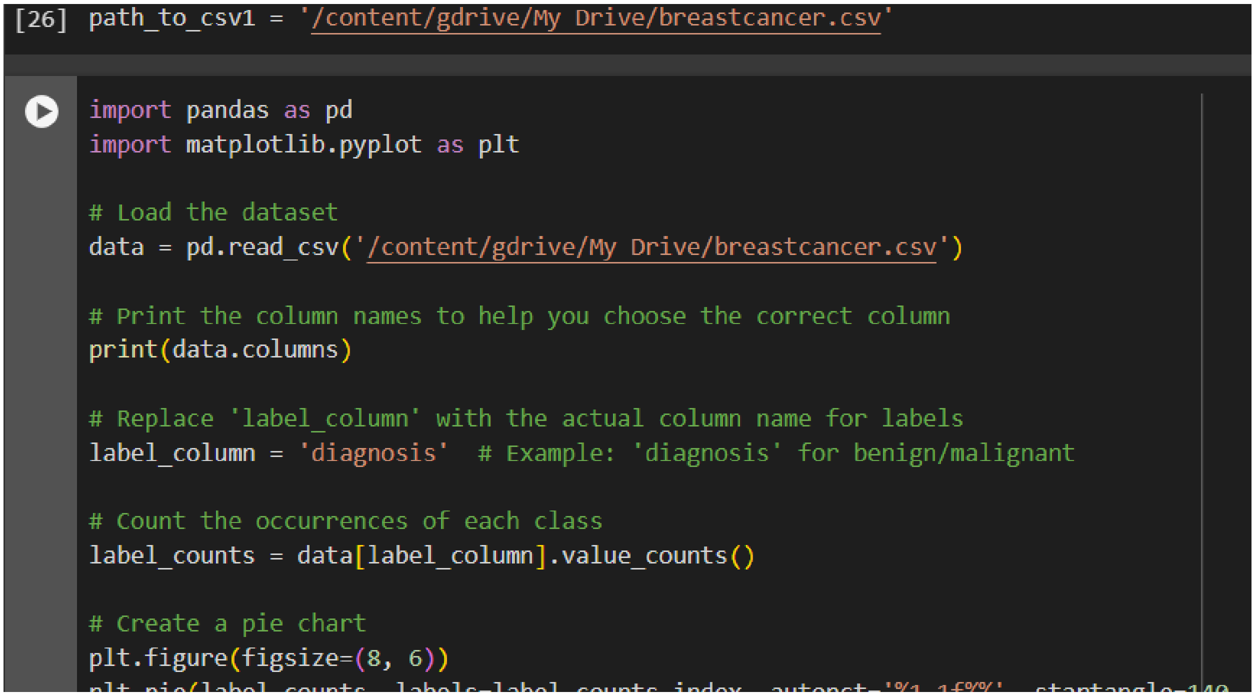
```python
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# 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_csv1, 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)
```



```python
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(30, activation='relu')) # hidden layer
my_first_nn.add(Dense(40, activation='relu')) # hidden layer
my_first_nn.add(Dense(50, 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))
```

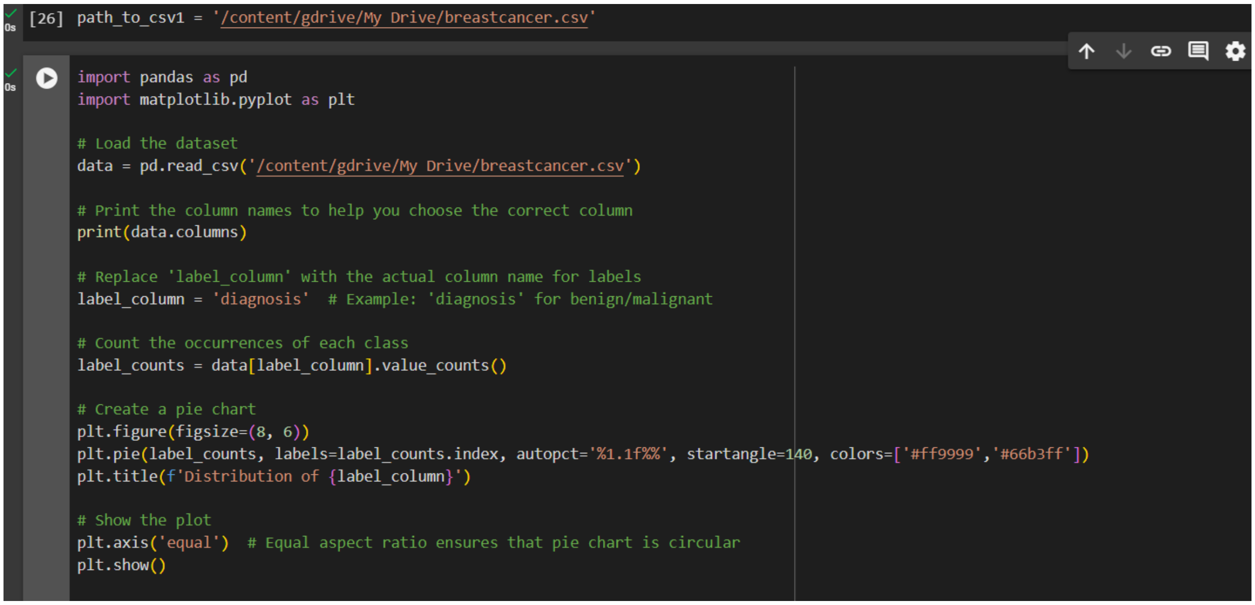


```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - acc: 1.0000 - loss: 2.6565e-06
Epoch 95/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - acc: 1.0000 - loss: 1.9315e-06
Epoch 96/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 2.3620e-06
Epoch 97/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 3.7930e-06
Epoch 98/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - acc: 1.0000 - loss: 2.7820e-06
Epoch 99/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - acc: 1.0000 - loss: 2.4997e-06
Epoch 100/100
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - acc: 1.0000 - loss: 2.7735e-06
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ dense_5 (Dense)                      │ (None, 20)                  │             620 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_6 (Dense)                      │ (None, 30)                  │             630 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_7 (Dense)                      │ (None, 40)                  │           1,240 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_8 (Dense)                      │ (None, 50)                  │           2,050 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_9 (Dense)                      │ (None, 1)                   │              51 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 13,775 (53.81 KB)
 Trainable params: 4,591 (17.93 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 9,184 (35.88 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - acc: 1.0000 - loss: 1.7479e-06
[1.4273883834903245e-06, 1.0]
```



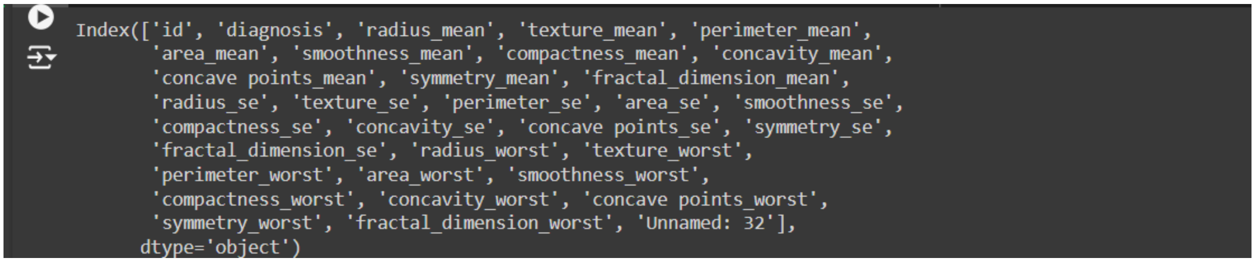
```
[26] path_to_csv1 = '/content/gdrive/My Drive/breastcancer.csv'
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('/content/gdrive/My Drive/breastcancer.csv')

# Print the column names to help you choose the correct column
print(data.columns)

# Replace 'label_column' with the actual column name for labels
label_column = 'diagnosis'  # Example: 'diagnosis' for benign/malignant

# Count the occurrences of each class
label_counts = data[label_column].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 6))
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%', startangle=140
```

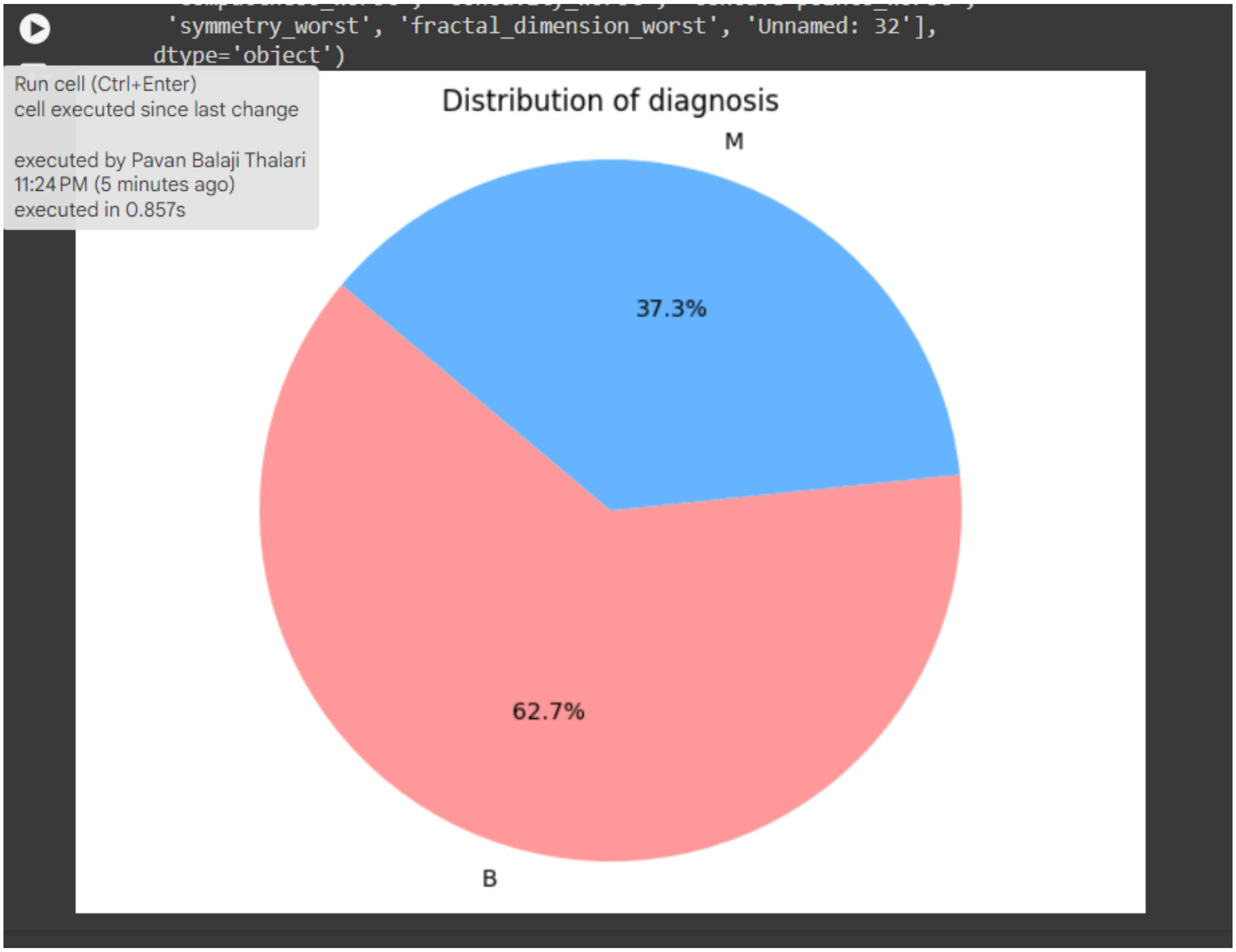


```
[26] path_to_csv1 = '/content/gdrive/My Drive/breastcancer.csv'
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('/content/gdrive/My Drive/breastcancer.csv')

# Print the column names to help you choose the correct column
print(data.columns)

# Replace 'label_column' with the actual column name for labels
label_column = 'diagnosis'  # Example: 'diagnosis' for benign/malignant

# Count the occurrences of each class
label_counts = data[label_column].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 6))
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff'])
plt.title(f'Distribution of {label_column}')

# Show the plot
plt.axis('equal')  # Equal aspect ratio ensures that pie chart is circular
plt.show()
```



```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```



```
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

Run cell (Ctrl+Enter)
cell executed since last change

executed by Pavan Balaji Thalari
11:24 PM (5 minutes ago)
executed in 0.857s

https://github.com/pavan7036/bda.git