Breast Cancer Detection Using Neural Networks

By-Antarjita Adhya

B.Tech, CSE

Introduction

Breast cancer remains one of the most prevalent and deadly cancers among women globally. Early detection and diagnosis are critical for improving survival rates and treatment efficacy. Machine learning, particularly neural networks, offers powerful tools for developing diagnostic models that can analyze medical data and provide accurate predictions. This project leverages neural networks to classify breast cancer tumors, aiming to support early diagnosis and timely intervention.

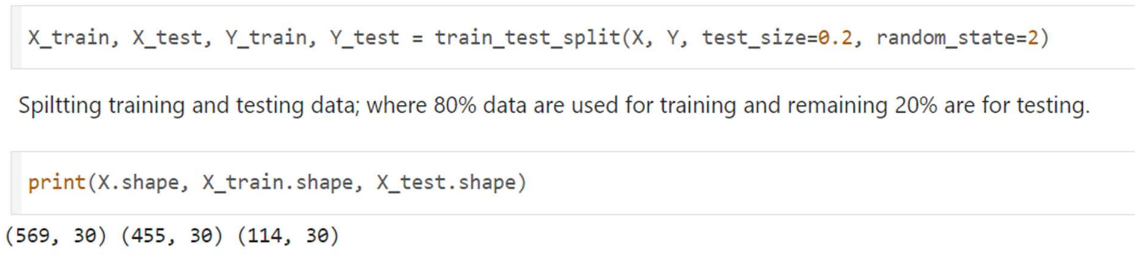
Methodology:

Data Collection and Preprocessing:

1. **Load Data**: The Breast Cancer Wisconsin dataset was loaded from the sklearn library.
2. **Data Frame Creation**: The dataset was converted into a Pandas DataFrame, allowing for easy manipulation and analysis.
3. **Data Exploration**: Initial exploration was performed to understand the dataset's structure and characteristics. This included printing the DataFrame, checking the first and last few rows, and displaying the shape of the dataset.
4. **Handling Missing Values**: A check for missing values ensured the dataset was complete and ready for analysis.
5. **Statistical Analysis**: Statistical measures and distribution of the target variable were analyzed to gain insights into the data.

Feature Engineering:

1. **Label Encoding**: The target label (benign or malignant) was added to the DataFrame.
2. **Train-Test Split**: The data was split into training (80%) and testing (20%) sets to evaluate the model's performance on unseen data.



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

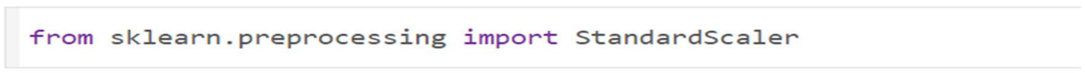
Spiltting training and testing data; where 80% data are used for training and remaining 20% are for testing.

```
print(X.shape, X_train.shape, X_test.shape)
```

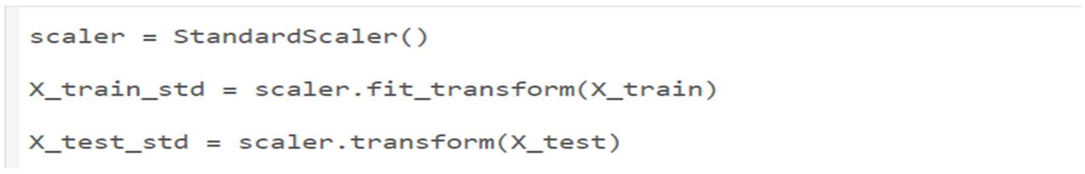
```
(569, 30) (455, 30) (114, 30)
```

Data Standardization:

1. **Scaling**: Features were standardized using the StandardScaler to improve model performance and convergence.



```
from sklearn.preprocessing import StandardScaler
```



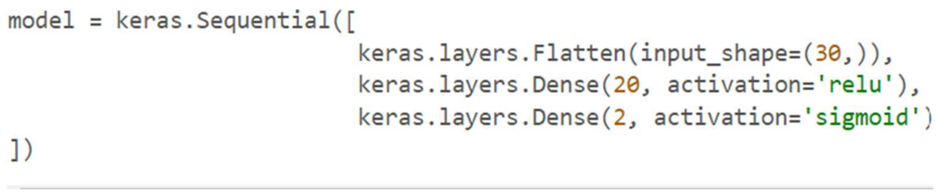
```
scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)

X_test_std = scaler.transform(X_test)
```

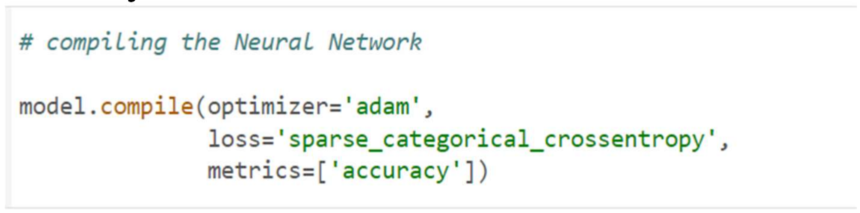
Model Building:

1. **Define Model Architecture**: A neural network model was created with the following layers:
   - Input layer: Flatten the input features.
   - Hidden layer: Dense layer with 20 neurons and ReLU activation.
   - Output layer: Dense layer with 2 neurons and sigmoid activation.



```
model = keras.Sequential([
                          keras.layers.Flatten(input_shape=(30,)),
                          keras.layers.Dense(20, activation='relu'),
                          keras.layers.Dense(2, activation='sigmoid')
])
```

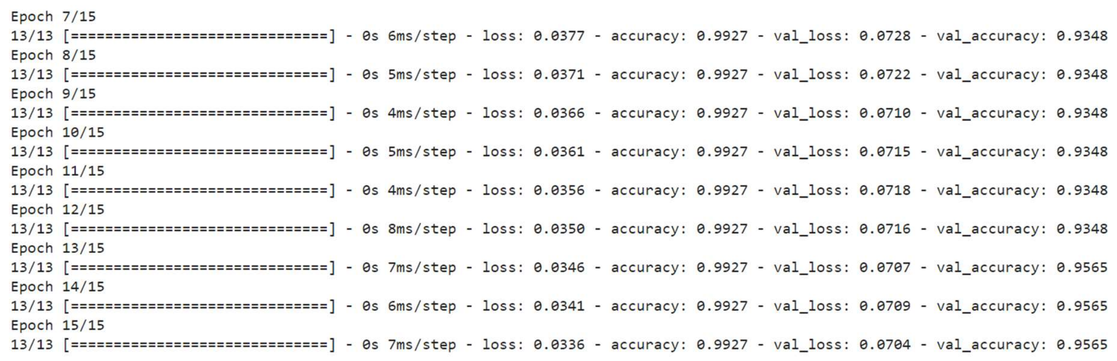
2. **Compile Model**: The model was compiled using the Adam optimizer, sparse categorical crossentropy loss function, and accuracy as the evaluation metric.



```
# compiling the Neural Network

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

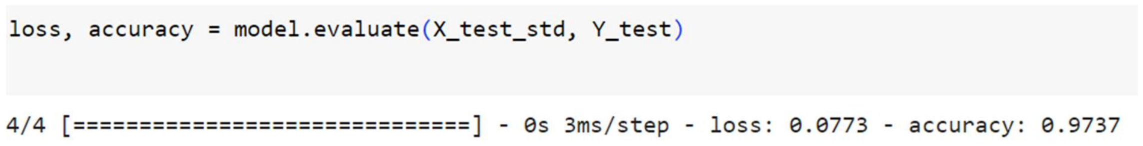
3. **Train Model**: The model was trained on the standardized training data with a validation split of 10% over 15 epochs.



```
Epoch 7/15
13/13 [==============================] - 0s 6ms/step - loss: 0.0377 - accuracy: 0.9927 - val_loss: 0.0728 - val_accuracy: 0.9348
Epoch 8/15
13/13 [==============================] - 0s 5ms/step - loss: 0.0371 - accuracy: 0.9927 - val_loss: 0.0722 - val_accuracy: 0.9348
Epoch 9/15
13/13 [==============================] - 0s 4ms/step - loss: 0.0366 - accuracy: 0.9927 - val_loss: 0.0710 - val_accuracy: 0.9348
Epoch 10/15
13/13 [==============================] - 0s 5ms/step - loss: 0.0361 - accuracy: 0.9927 - val_loss: 0.0715 - val_accuracy: 0.9348
Epoch 11/15
13/13 [==============================] - 0s 4ms/step - loss: 0.0356 - accuracy: 0.9927 - val_loss: 0.0718 - val_accuracy: 0.9348
Epoch 12/15
13/13 [==============================] - 0s 8ms/step - loss: 0.0350 - accuracy: 0.9927 - val_loss: 0.0716 - val_accuracy: 0.9348
Epoch 13/15
13/13 [==============================] - 0s 7ms/step - loss: 0.0346 - accuracy: 0.9927 - val_loss: 0.0707 - val_accuracy: 0.9565
Epoch 14/15
13/13 [==============================] - 0s 6ms/step - loss: 0.0341 - accuracy: 0.9927 - val_loss: 0.0709 - val_accuracy: 0.9565
Epoch 15/15
13/13 [==============================] - 0s 7ms/step - loss: 0.0336 - accuracy: 0.9927 - val_loss: 0.0704 - val_accuracy: 0.9565
```

Evaluation and Prediction

1. **Model Evaluation:** The model's performance was assessed on the test set, achieving a high accuracy rate(97.37%).
2. **Prediction:** Predictions were made on new data points to demonstrate the model's practical application.

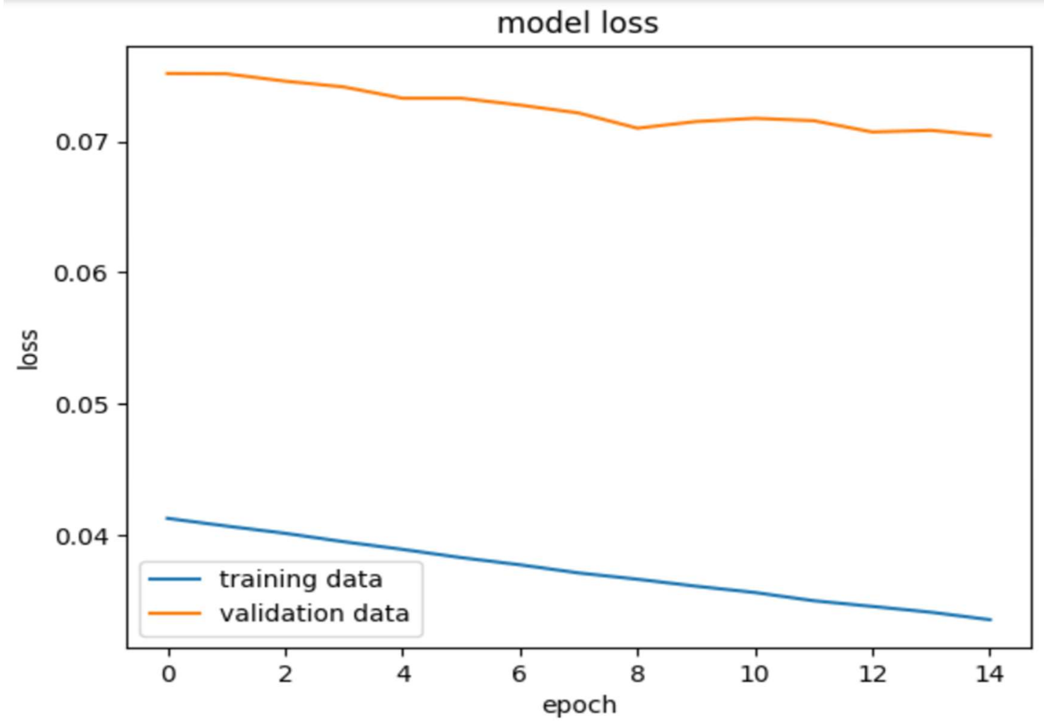


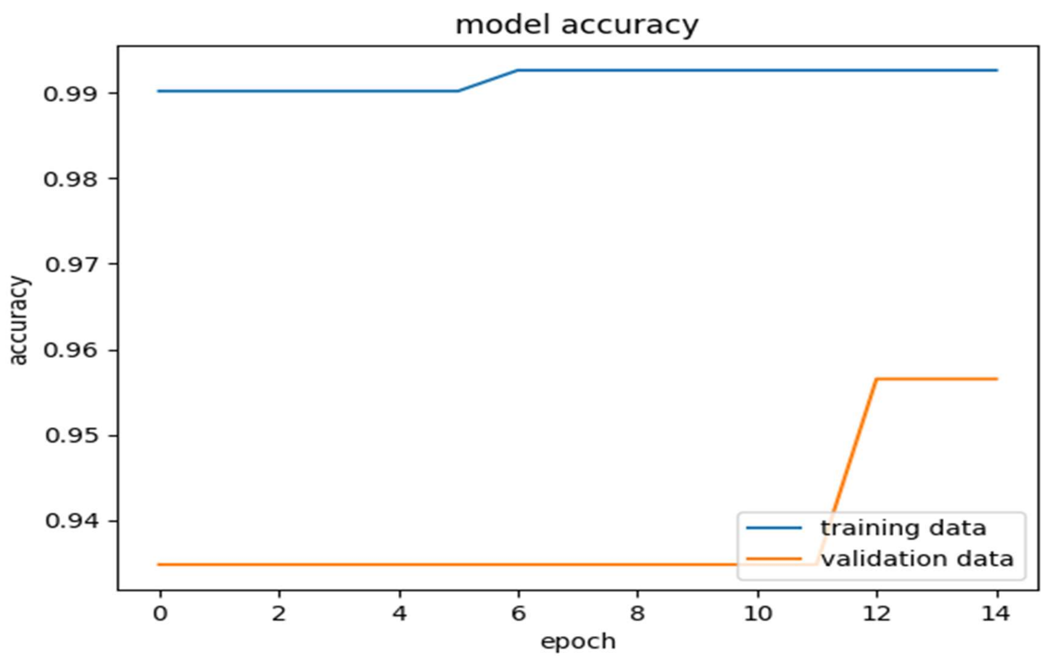
```
loss, accuracy = model.evaluate(X_test_std, Y_test)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.0773 - accuracy: 0.9737
```

Results

- **Model Accuracy:** The neural network model achieved an impressive accuracy of 97.37% on the test set.
- **Visualization:** Plots of training and validation accuracy and loss over epochs were generated, showing the model's learning progress and stability.





```
1/1 [==============================] - 0s 55ms/step
[[0.04421783 0.99968433]]
[1]
The tumor is Benign
```

References:

1.Research Paper: An Artificial Neural Network Based Approach for Breast Cancer Diagnosis

- Authors: Edin Gamberger, Nada Lavrac, and Mitja Krstajic

2. Research Paper: Breast Cancer Diagnosis and Prognosis via Linear Programming

- Authors: William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian

3. Book: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron