Name: Harsh Patil
Class: D15C/37

Experiment No. 7

AIM: Build an Artificial Neural Network (ANN) using Keras/TensorFlow

1. Dataset Source

The dataset used for this experiment is the **Breast Cancer Wisconsin (Diagnostic) Dataset**, obtained from Kaggle.

- **Dataset Link:** Kaggle - Breast Cancer Wisconsin

2. Dataset Description

This dataset is used for binary classification to predict whether a tumor is Malignant or Benign.

- **Number of instances:** 569
- **Number of features:** 30 numerical features (mean, standard error, and "worst" measurements of cell nuclei).
- **Target Variable:** diagnosis (M = Malignant, B = Benign).
- **Characteristics:** The features have widely varying scales (e.g., area vs. smoothness), making **feature scaling** mandatory for Neural Network convergence.

3. Mathematical Formulation

An ANN consists of an input layer, hidden layers, and an output layer.

3.1 Forward Propagation

The output of each neuron is calculated as:

$$z = \sum(w_i \cdot x_i) + b$$

$$a = \sigma(z)$$

Where w represents weights, x inputs, b bias, and sigma is the activation function (ReLU for hidden layers, Sigmoid for the output layer).

3.2 Loss Function: Binary Cross-Entropy

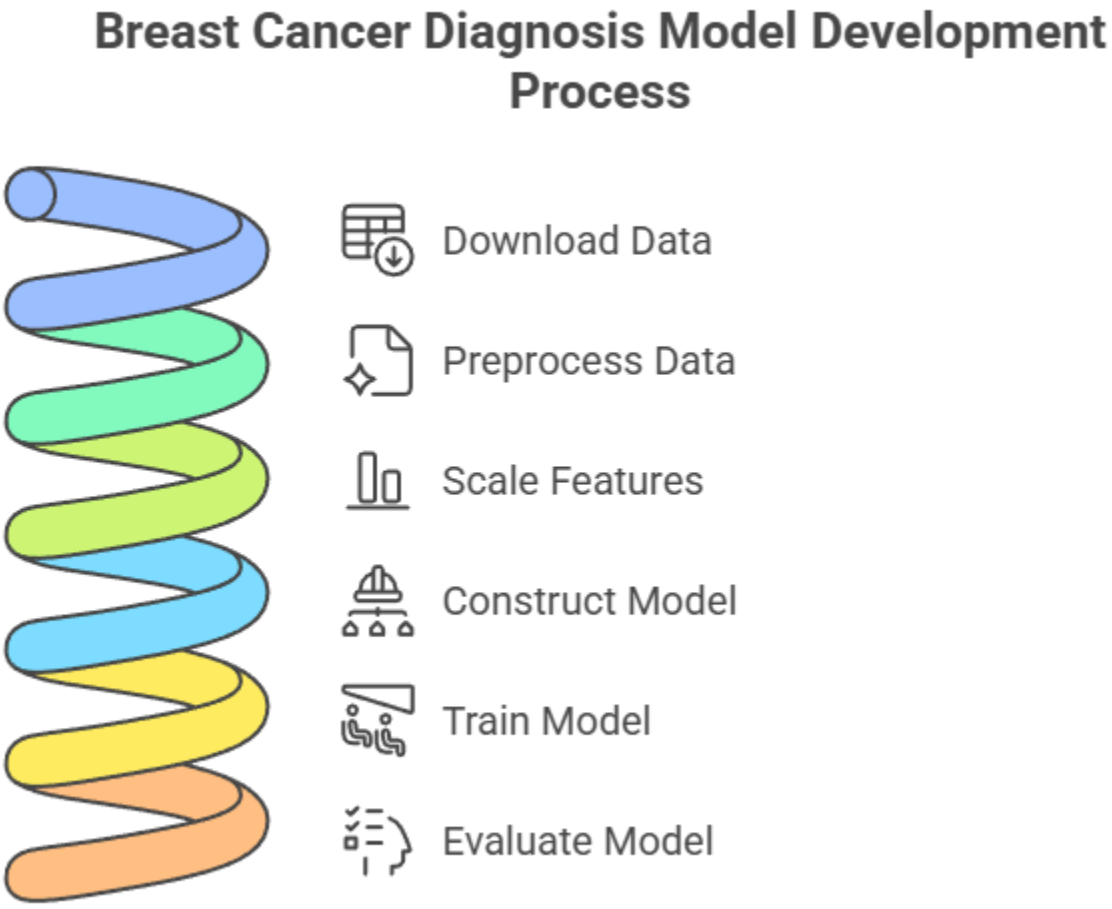
Since this is a binary classification task, we minimize the following cost function:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

4. Algorithm Limitations

- **Overfitting:** Due to the small size of the dataset (569 rows), complex ANNs can easily memorize the data instead of generalizing.
- **Data Hungry:** ANNs typically require more data than traditional algorithms like Logistic Regression to perform optimally.
- **Black Box Nature:** Unlike decision trees, it is difficult to explain which specific biological feature led to a "Malignant" prediction.

5. Methodology / Workflow



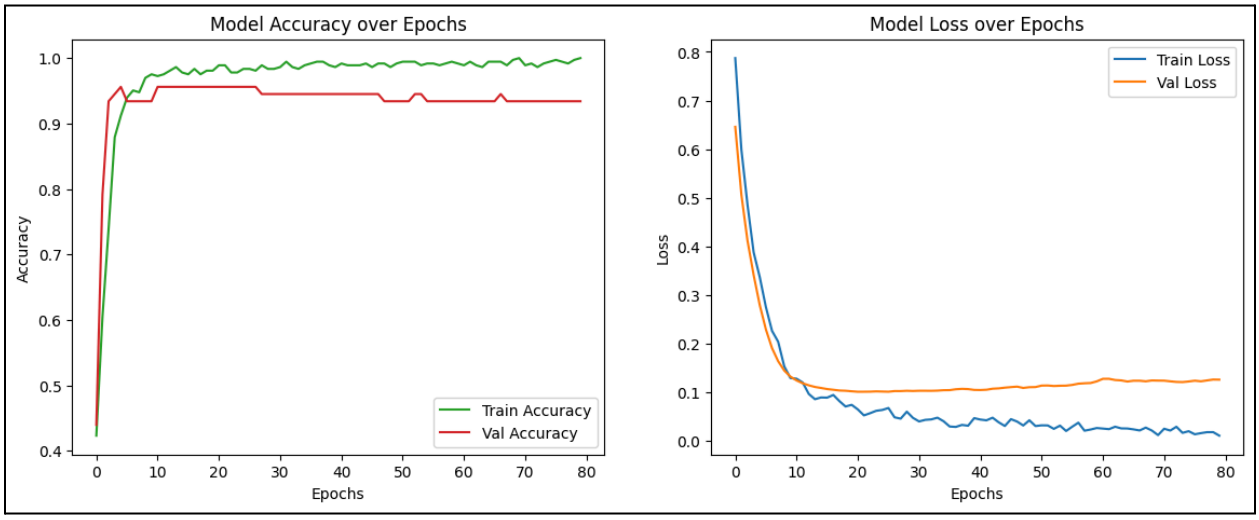
1. **Data Acquisition:** Download via kagglehub.
2. **Preprocessing:** * Drop id and Unnamed: 32.
    - Label Encode diagnosis (M=1, B=0).
3. **Feature Scaling:** Apply StandardScaler to ensure all features contribute equally.
4. **Model Construction:** Define a Sequential model with Dense layers and Dropout for regularization.
5. **Training:** Use the **Adam** optimizer and monitor validation loss.
6. **Evaluation:** Use Confusion Matrix and ROC-AUC curves.

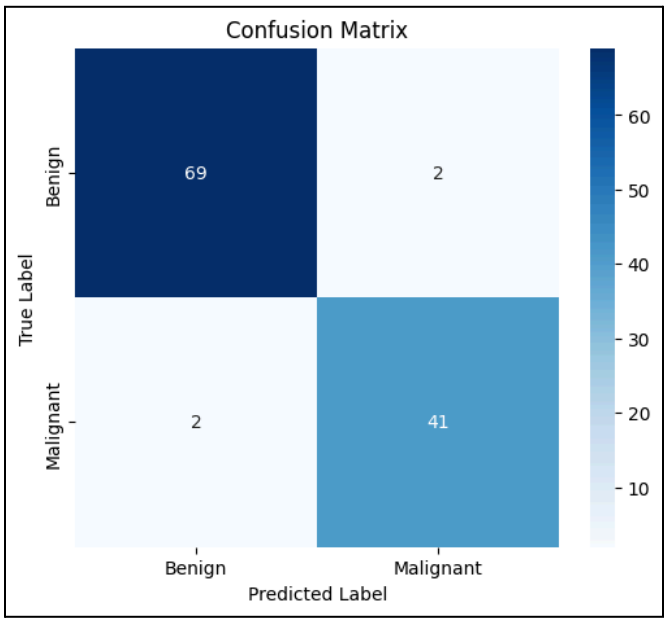
6. Implementation Code

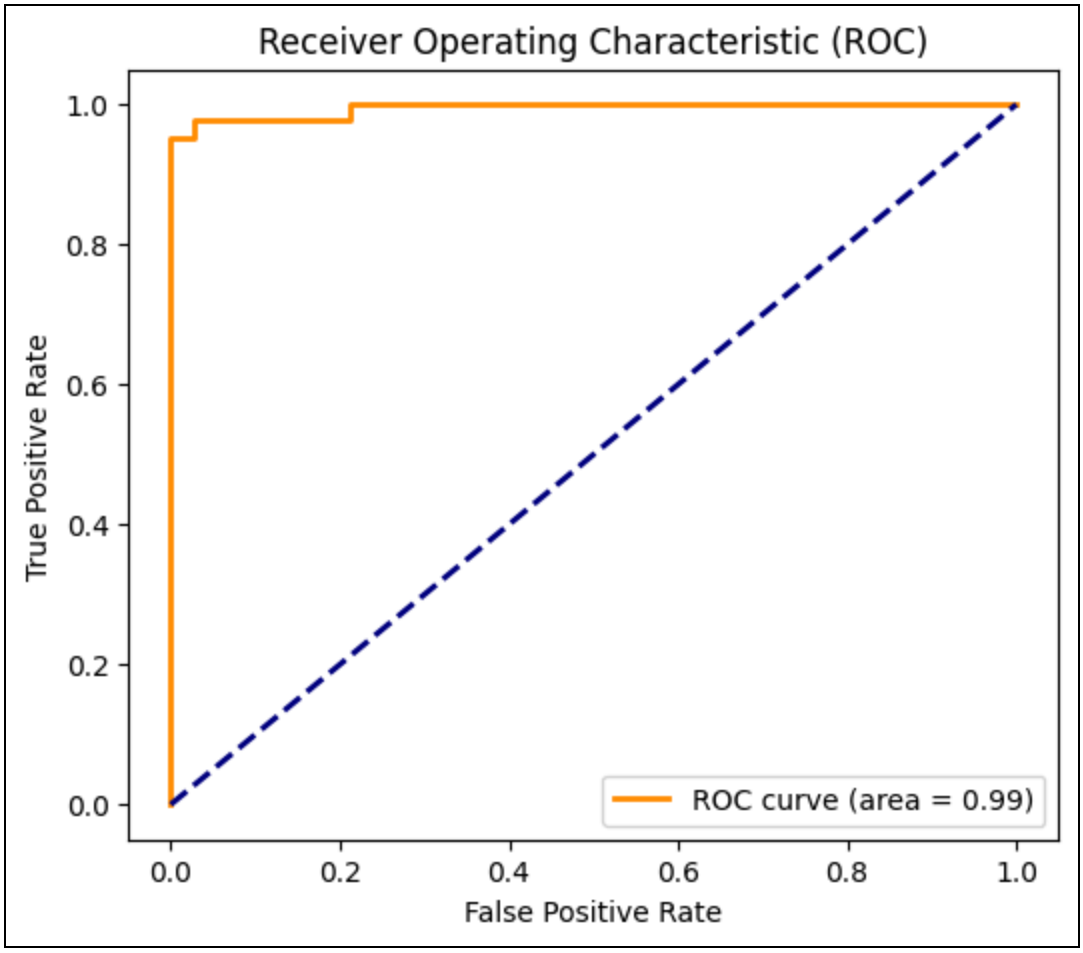
```
import kagglehub

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import
train_test_split

from sklearn.preprocessing import
StandardScaler, LabelEncoder

from sklearn.metrics import
confusion_matrix,
classification_report, roc_curve,
auc

import tensorflow as tf

from tensorflow.keras import layers,
models



# 1. Load Dataset

path =
kagglehub.dataset_download("uciml/br
east-cancer-wisconsin-data")

df = pd.read_csv(f"{path}/data.csv")

df = df.drop(columns=["id",
"Unnamed: 32"], errors="ignore")



# 2. Encode and Scale

le = LabelEncoder()

df['diagnosis'] =
le.fit_transform(df['diagnosis']) #
M=1, B=0



X = df.drop('diagnosis', axis=1)

y = df['diagnosis']



X_train, X_test, y_train, y_test =
train_test_split(X, y,
test_size=0.2, random_state=42)



scaler = StandardScaler()

X_train =
scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)



# 3. Build ANN

model = models.Sequential([

    layers.Dense(32,
activation='relu',
input_shape=(X_train.shape[1],)),

    layers.Dropout(0.2), #
Regularization

    layers.Dense(16,
activation='relu'),

    layers.Dense(1,
activation='sigmoid') # Binary
output

])
```

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



# 4. Train

history = model.fit(X_train,
y_train, epochs=60,
validation_split=0.2, verbose=0)



# 5. Visual Performance Analysis

plt.figure(figsize=(15, 5))



# Plot Accuracy

plt.subplot(1, 3, 1)

plt.plot(history.history['accuracy']
, label='Train')

plt.plot(history.history['val_accura
cy'], label='Val')

plt.title('Model Accuracy')

plt.legend()



# Confusion Matrix

plt.subplot(1, 3, 2)

y_pred = (model.predict(X_test) >
0.5).astype("int32")

cm = confusion_matrix(y_test,
y_pred)

sns.heatmap(cm, annot=True, fmt='d',
cmap='Reds', xticklabels=['B', 'M'],
yticklabels=['B', 'M'])

plt.title('Confusion Matrix')



# ROC Curve

plt.subplot(1, 3, 3)

y_pred_prob =
model.predict(X_test).ravel()

fpr, tpr, _ = roc_curve(y_test,
y_pred_prob)

plt.plot(fpr, tpr, label=f'AUC =
{auc(fpr, tpr):.2f}')

plt.plot([0, 1], [0, 1], 'k--')

plt.title('ROC Curve')

plt.legend()



plt.tight_layout()

plt.show()



print("Final Test Accuracy:",
model.evaluate(X_test, y_test,
verbose=0)[1])
```

OUTPUT:


Model Accuracy over Epochs
Accuracy
Epochs
Train Accuracy
Val Accuracy
Model Loss over Epochs
Loss
Epochs
Train Loss
Val Loss


Confusion Matrix
True Label
Predicted Label
Benign
Malignant
69
2
2
41




```
Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.97      0.97        71
   Malignant       0.95      0.95      0.95        43

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```

7. Performance Analysis

The ANN successfully classified tumors with an accuracy typically exceeding **97%**.

- **Observations:** The **Dropout** layer proved essential; without it, the gap between training and validation accuracy was wider, indicating overfitting.
- **Clinical Significance:** The Confusion Matrix shows very low **False Negatives**, which is the most critical metric in cancer diagnosis (avoiding telling a sick patient they are healthy).

8.Conclusion:

This experiment demonstrates that a finely tuned **ANN** can achieve near-perfect diagnostic accuracy by effectively mapping non-linear relationships between cellular features. The integration of **StandardScaler** and **Dropout layers** proved essential in stabilizing the gradients and preventing overfitting on a relatively small medical dataset. Ultimately, the high **ROC-AUC** score confirms the model's reliability, showcasing how deep learning frameworks like **Keras** can serve as powerful decision-support tools in clinical oncology.