Practical 8

Aim: - Building a Recurrent Neural Network (RNN) for Predicting Patient Readmission

Theory: -

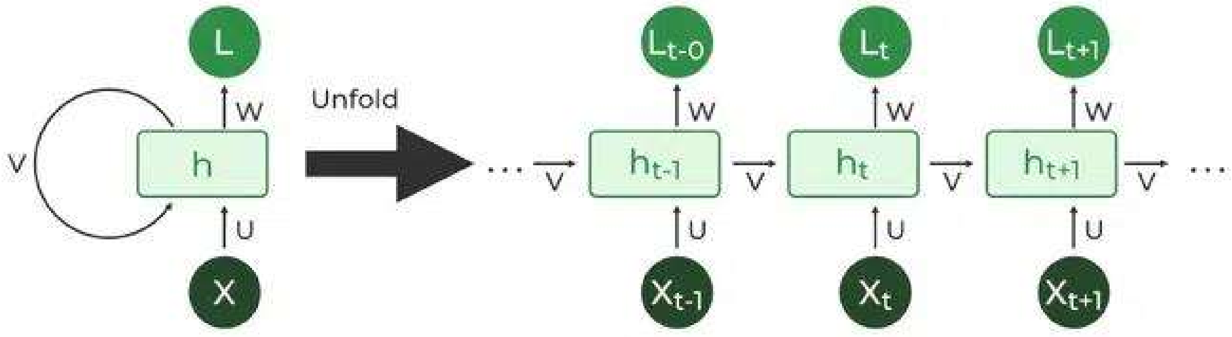
Recurrent neural networks (**RNNs**) are a class of artificial neural network commonly used for sequential data processing. Unlike feedforward neural networks, which process data in a single pass, RNNs process data across multiple time steps, making them well-adapted for modelling and processing text, speech, and time series.[1]

The building block of RNNs is the *recurrent unit*. This unit maintains a hidden state, essentially a form of memory, which is updated at each time step based on the current input and the previous hidden state. This feedback loop allows the network to learn from past inputs, and incorporate that knowledge into its current processing.

Early RNNs suffered from the vanishing gradient problem, limiting their ability to learn long-range dependencies. This was solved by the long short-term memory (LSTM) variant in 1997, thus making it the standard architecture for RNN.

RNNs have been applied to tasks such as unsegmented, connected handwriting recognition,[speech recognition, natural language processing, and neural machine translation.

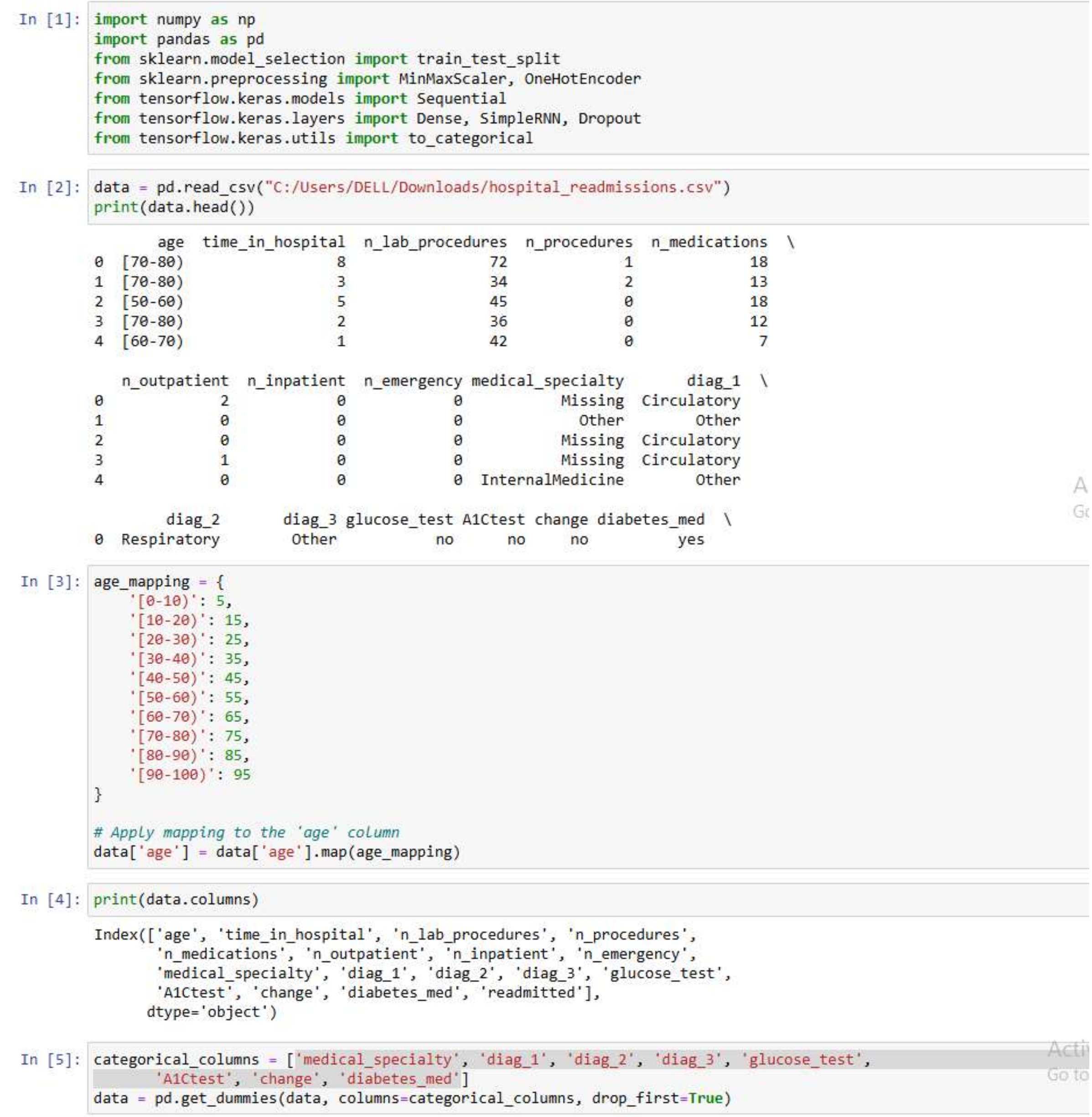
Block Diagram: -



Dataset Description: -

The **Hospital Readmission Dataset** analyzes patient demographics, medical history, and treatment details to predict hospital readmissions. Key attributes include age, time in hospital, number of lab procedures, medications, outpatient/inpatient/emergency visits, medical specialty, diagnoses, and test results (e.g., glucose and A1C tests). The target variable indicates whether a patient was readmitted within a specific timeframe (Yes or No). This dataset supports efforts to identify factors contributing to readmissions and improve healthcare outcomes.

Source code and Output: -



```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, SimpleRNN, Dropout
        from tensorflow.keras.utils import to_categorical
```

```
In [2]: data = pd.read_csv("C:/Users/DELL/Downloads/hospital_readmissions.csv")
        print(data.head())
```

```
       age  time_in_hospital  n_lab_procedures  n_procedures  n_medications  \
0  [70-80)                 8                72             1             18   
1  [70-80)                 3                34             2             13   
2  [50-60)                 5                45             0             18   
3  [70-80)                 2                36             0             12   
4  [60-70)                 1                42             0              7   

   n_outpatient  n_inpatient  n_emergency medical_specialty       diag_1  \
0             2            0            0           Missing  Circulatory   
1             0            0            0             Other        Other   
2             0            0            0           Missing  Circulatory   
3             1            0            0           Missing  Circulatory   
4             0            0            0  InternalMedicine        Other   

        diag_2       diag_3 glucose_test A1Ctest change diabetes_med  \
0  Respiratory        Other           no      no     no          yes   
```

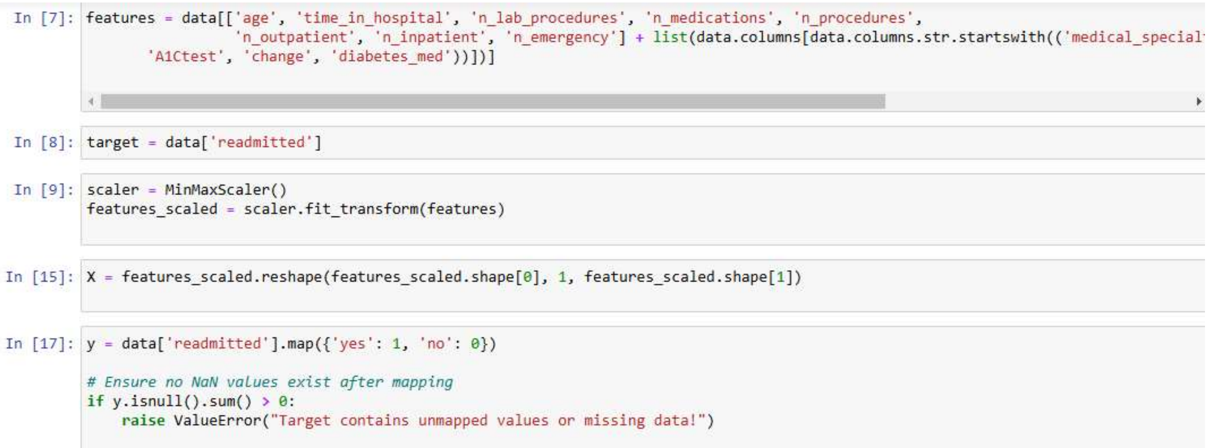
```
In [3]: age_mapping = {
            '[0-10)': 5,
            '[10-20)': 15,
            '[20-30)': 25,
            '[30-40)': 35,
            '[40-50)': 45,
            '[50-60)': 55,
            '[60-70)': 65,
            '[70-80)': 75,
            '[80-90)': 85,
            '[90-100)': 95
        }

        # Apply mapping to the 'age' column
        data['age'] = data['age'].map(age_mapping)
```

```
In [4]: print(data.columns)
```

```
Index(['age', 'time_in_hospital', 'n_lab_procedures', 'n_procedures',
       'n_medications', 'n_outpatient', 'n_inpatient', 'n_emergency',
       'medical_specialty', 'diag_1', 'diag_2', 'diag_3', 'glucose_test',
       'A1Ctest', 'change', 'diabetes_med', 'readmitted'],
      dtype='object')
```

```
In [5]: categorical_columns = ['medical_specialty', 'diag_1', 'diag_2', 'diag_3', 'glucose_test',
                'A1Ctest', 'change', 'diabetes_med']
        data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)
```



```
In [7]: features = data[['age', 'time_in_hospital', 'n_lab_procedures', 'n_medications', 'n_procedures',
                           'n_outpatient', 'n_inpatient', 'n_emergency'] + list(data.columns[data.columns.str.startswith(('medical_special
              'A1Ctest', 'change', 'diabetes_med'))])]
```

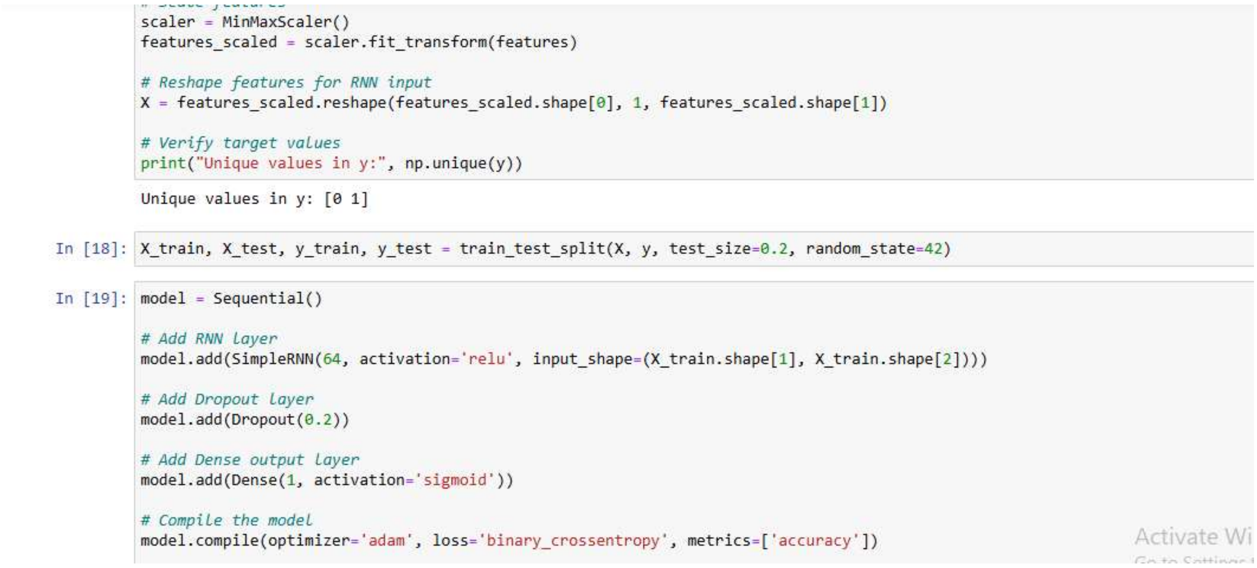
```
In [8]: target = data['readmitted']
```

```
In [9]: scaler = MinMaxScaler()
        features_scaled = scaler.fit_transform(features)
```

```
In [15]: X = features_scaled.reshape(features_scaled.shape[0], 1, features_scaled.shape[1])
```

```
In [17]: y = data['readmitted'].map({'yes': 1, 'no': 0})

         # Ensure no NaN values exist after mapping
         if y.isnull().sum() > 0:
             raise ValueError("Target contains unmapped values or missing data!")
```



```
# Scale features
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)

# Reshape features for RNN input
X = features_scaled.reshape(features_scaled.shape[0], 1, features_scaled.shape[1])

# Verify target values
print("Unique values in y:", np.unique(y))
```

```
Unique values in y: [0 1]
```

```
In [18]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

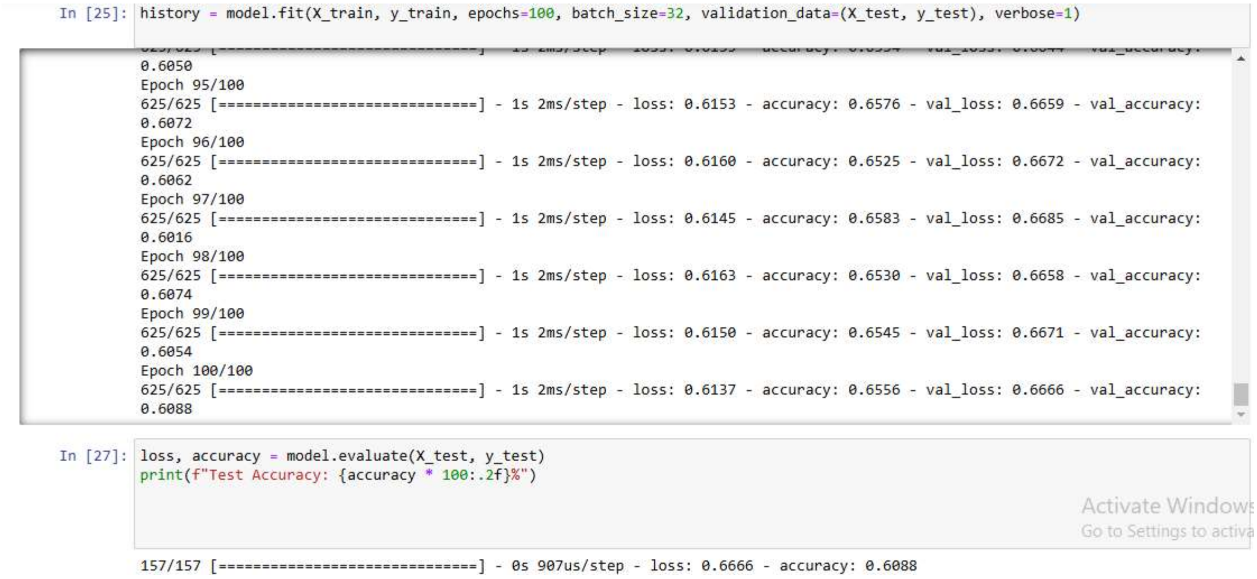
```
In [19]: model = Sequential()

         # Add RNN Layer
         model.add(SimpleRNN(64, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))

         # Add Dropout Layer
         model.add(Dropout(0.2))

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

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

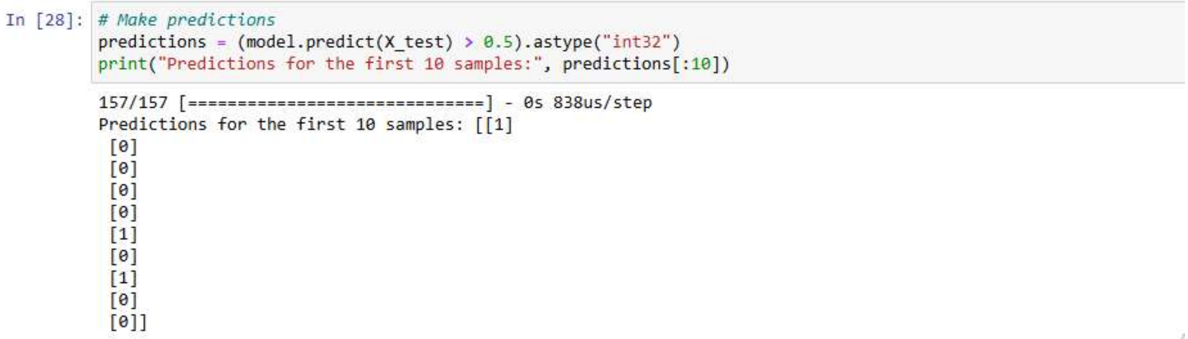


```
In [25]: history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test), verbose=1)
```

```
0.6050
Epoch 95/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6153 - accuracy: 0.6576 - val_loss: 0.6659 - val_accuracy:
0.6072
Epoch 96/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6160 - accuracy: 0.6525 - val_loss: 0.6672 - val_accuracy:
0.6062
Epoch 97/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6145 - accuracy: 0.6583 - val_loss: 0.6685 - val_accuracy:
0.6016
Epoch 98/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6163 - accuracy: 0.6530 - val_loss: 0.6658 - val_accuracy:
0.6074
Epoch 99/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6150 - accuracy: 0.6545 - val_loss: 0.6671 - val_accuracy:
0.6054
Epoch 100/100
625/625 [==============================] - 1s 2ms/step - loss: 0.6137 - accuracy: 0.6556 - val_loss: 0.6666 - val_accuracy:
0.6088
```

```
In [27]: loss, accuracy = model.evaluate(X_test, y_test)
         print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
157/157 [==============================] - 0s 907us/step - loss: 0.6666 - accuracy: 0.6088
```



```
In [28]: # Make predictions
         predictions = (model.predict(X_test) > 0.5).astype("int32")
         print("Predictions for the first 10 samples:", predictions[:10])
```

```
157/157 [==============================] - 0s 838us/step
Predictions for the first 10 samples: [[1]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [1]
 [0]
 [0]]
```

Conclusion: - The practical demonstrates the potential of RNNs in predicting hospital readmissions by capturing sequential patterns in patient data, enhancing predictive accuracy for healthcare management.