**Exercise: 1 - Classification and Regression with Artificial Neural Networks (ANN)**

**Files:**

## Install Required Libraries:

|  |
| --- |
| pip install numpy pandas scikit-learn tensorflow streamlit |

## 1. Feature Transformation Using Scikit-Learn with ANN

**1.1. Import Libraries and Load Data**

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.datasets import load\_iris # Example dataset |

**1.2. Load and Split Data**

|  |
| --- |
| # Load dataset (e.g., Iris dataset)  data = load\_iris()  X = data.data  y = data.target  # Split dataset into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) |

**1.3 Feature Transformation**

|  |
| --- |
| # Scale features  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test) |

**2. Step-by-Step Training with ANN Using TensorFlow**

**2.1. Import Libraries**

|  |
| --- |
| import tensorflow as tf  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense |

**2.2. Define ANN Architecture**

|  |
| --- |
| # Define the ANN model  model = Sequential([  Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),  Dense(32, activation='relu'),  Dense(3, activation='softmax') # Output layer for 3 classes  ]) |

**2.3. Compile the Model**

|  |
| --- |
| # Compile the model  model.compile(optimizer='adam',  loss='sparse\_categorical\_crossentropy',  metrics=['accuracy']) |

**2.4. Train the Model**

|  |
| --- |
| # Train the model  history = model.fit(X\_train, y\_train, epochs=20, batch\_size=32,  validation\_split=0.2, verbose=1) |

**2.5. Save the Model**

|  |
| --- |
| model.save('simple\_rnn\_imdb\_output.h5') |

**2.6. Save Training History**

|  |
| --- |
| import pandas as pd  history\_df = pd.DataFrame(history.history)  history\_df.to\_csv('training\_history.csv', index=False) |

**3. Prediction with Trained ANN Model**

**3.1. Load the Model**

|  |
| --- |
| from tensorflow.keras.models import load\_model  model = load\_model('simple\_rnn\_imdb\_output.h5') |

**3.2. Define Helper Functions**

* **Decode Review:**

|  |
| --- |
| * def decode\_review(encoded\_review): * return ' '.join([reverse\_word\_index.get(i - 3, '?') for i in encoded\_review]) |

* **Preprocess Text:**

|  |
| --- |
| * def preprocess\_text(text): * words = text.lower().split() * encoded\_review = [word\_index.get(word, 2) + 3 for word in words] * padded\_review = sequence.pad\_sequences([encoded\_review], maxlen=500) * return padded\_review |

* **Predict Sentiment:**

|  |
| --- |
| * def predict\_sentiment(review): * preprocessed\_input = preprocess\_text(review) * prediction = model.predict(preprocessed\_input) * sentiment = 'Positive' if prediction[0][0] > 0.5 else 'Negative' * return sentiment, prediction[0][0] |

**3.3. Example Review for Prediction**

|  |
| --- |
| example\_review = "This movie was fantastic! The acting was great and the plot was thrilling."  sentiment, score = predict\_sentiment(example\_review)  print(f'Review: {example\_review}')  print(f'Sentiment: {sentiment}')  print(f'Prediction Score: {score}') |

## 4. Streamlit Application

**4.1. Set Up Streamlit App(main.py)**

|  |
| --- |
| import streamlit as st  import numpy as np  import tensorflow as tf  from sklearn.preprocessing import StandardScaler  # Load the pre-trained model  model = tf.keras.models.load\_model('ann\_model.h5')  # Load the scaler used for feature scaling  scaler = StandardScaler()  # Define preprocessing function  def preprocess\_input(input\_data):  scaled\_data = scaler.transform([input\_data])  return scaled\_data  # Define prediction function  def predict\_class(input\_data):  preprocessed\_input = preprocess\_input(input\_data)  prediction = model.predict(preprocessed\_input)  return np.argmax(prediction)  # Streamlit UI  st.title('ANN Classification App')  st.write('Enter feature values to classify.')  # User input  input\_features = st.text\_input('Enter features (comma-separated):')  if st.button('Predict'):  features = list(map(float, input\_features.split(',')))  prediction = predict\_class(features)  st.write(f'Predicted Class: {prediction}') |

**4.3. Run Streamlit App**

|  |
| --- |
| streamlit run main.py |

**Execution Links:**

|  |
| --- |
|  **Network link:** http://192.168.0.199:8501/   **Localhost:** http://localhost:8501/ |