Comprehensive Assessment : Deep Learning - Predicting Diabetes Progression using Artificial Neural Networks#

In [ ]:
```
#1. Loading and Preprocessing
#load the Diabetes dataset from sklearn, handle missing values and normalize the features for better performance with the ne
```

In [1]:
```
# Import necessary libraries
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
```

In [2]:
```
# Load the dataset
diabetes_data = load_diabetes()
```

In [3]:
```
# Convert to a DataFrame for better visualization
data = pd.DataFrame(diabetes_data.data, columns=diabetes_data.feature_names)
target = pd.Series(diabetes_data.target, name='target')
```

In [4]:
```
# Check for missing values (if any)
print(data.isnull().sum())
```

```
age    0
sex    0
bmi    0
bp     0
s1     0
s2     0
s3     0
s4     0
s5     0
s6     0
dtype: int64
```

In [5]:
```
# Normalize the features using StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

In [6]:
```
# Splitting data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(scaled_data, target, test_size=0.2, random_state=42)
```

In [ ]:
```
#2. Exploratory Data Analysis (EDA)
#perform basic EDA to understand the dataset and visualize relationships between features and the target variable.
```

In [7]:

```python
# Basic statistics
print(data.describe())
```

```
                age           sex           bmi            bp            s1  \
count  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02
mean  -2.511817e-19  1.230790e-17 -2.245564e-16 -4.797570e-17 -1.381499e-17
std    4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02
min   -1.072256e-01 -4.464164e-02 -9.027530e-02 -1.123988e-01 -1.267807e-01
25%   -3.729927e-02 -4.464164e-02 -3.422907e-02 -3.665608e-02 -3.424784e-02
50%    5.383060e-03 -4.464164e-02 -7.283766e-03 -5.670422e-03 -4.320866e-03
75%    3.807591e-02  5.068012e-02  3.124802e-02  3.564379e-02  2.835801e-02
max    1.107267e-01  5.068012e-02  1.705552e-01  1.320436e-01  1.539137e-01

                 s2            s3            s4            s5            s6
count  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02
mean   3.918434e-17 -5.777179e-18 -9.042540e-18  9.293722e-17  1.130318e-17
std    4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02
min   -1.156131e-01 -1.023071e-01 -7.639450e-02 -1.260971e-01 -1.377672e-01
25%   -3.035840e-02 -3.511716e-02 -3.949338e-02 -3.324559e-02 -3.317903e-02
50%   -3.819065e-03 -6.584468e-03 -2.592262e-03 -1.947171e-03 -1.077698e-03
75%    2.984439e-02  2.931150e-02  3.430886e-02  3.243232e-02  2.791705e-02
max    1.987880e-01  1.811791e-01  1.852344e-01  1.335973e-01  1.356118e-01
```

In [8]:

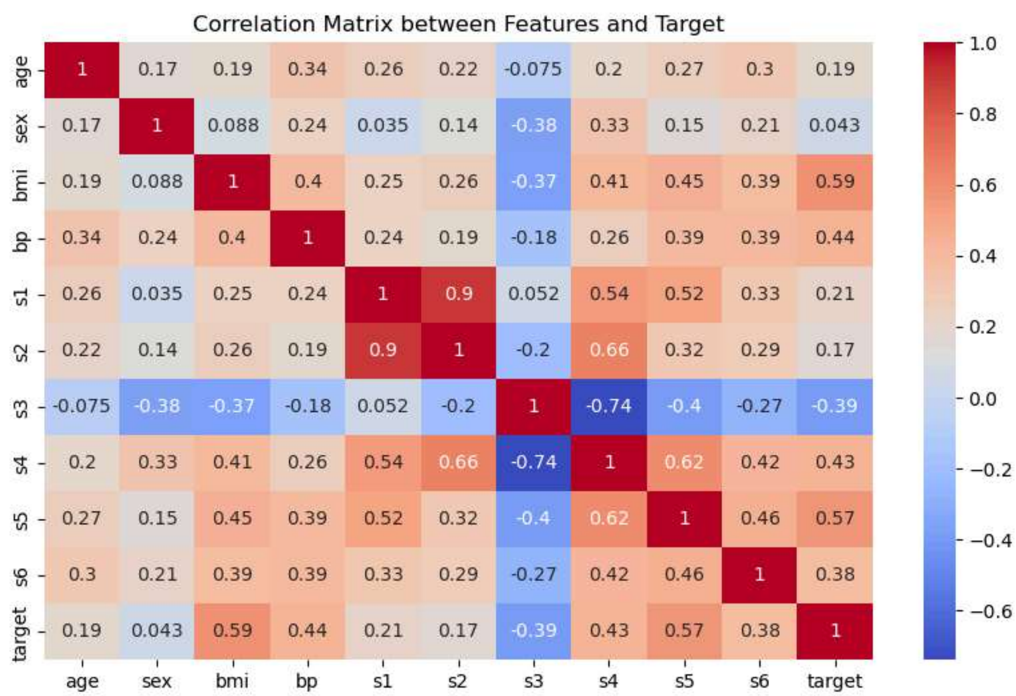
```python
# Pairplot to understand relationships between features and target
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:

```python
# Create a dataframe with scaled data for visualization
scaled_df = pd.DataFrame(scaled_data, columns=diabetes_data.feature_names)
scaled_df['target'] = target.values
```

In [10]:

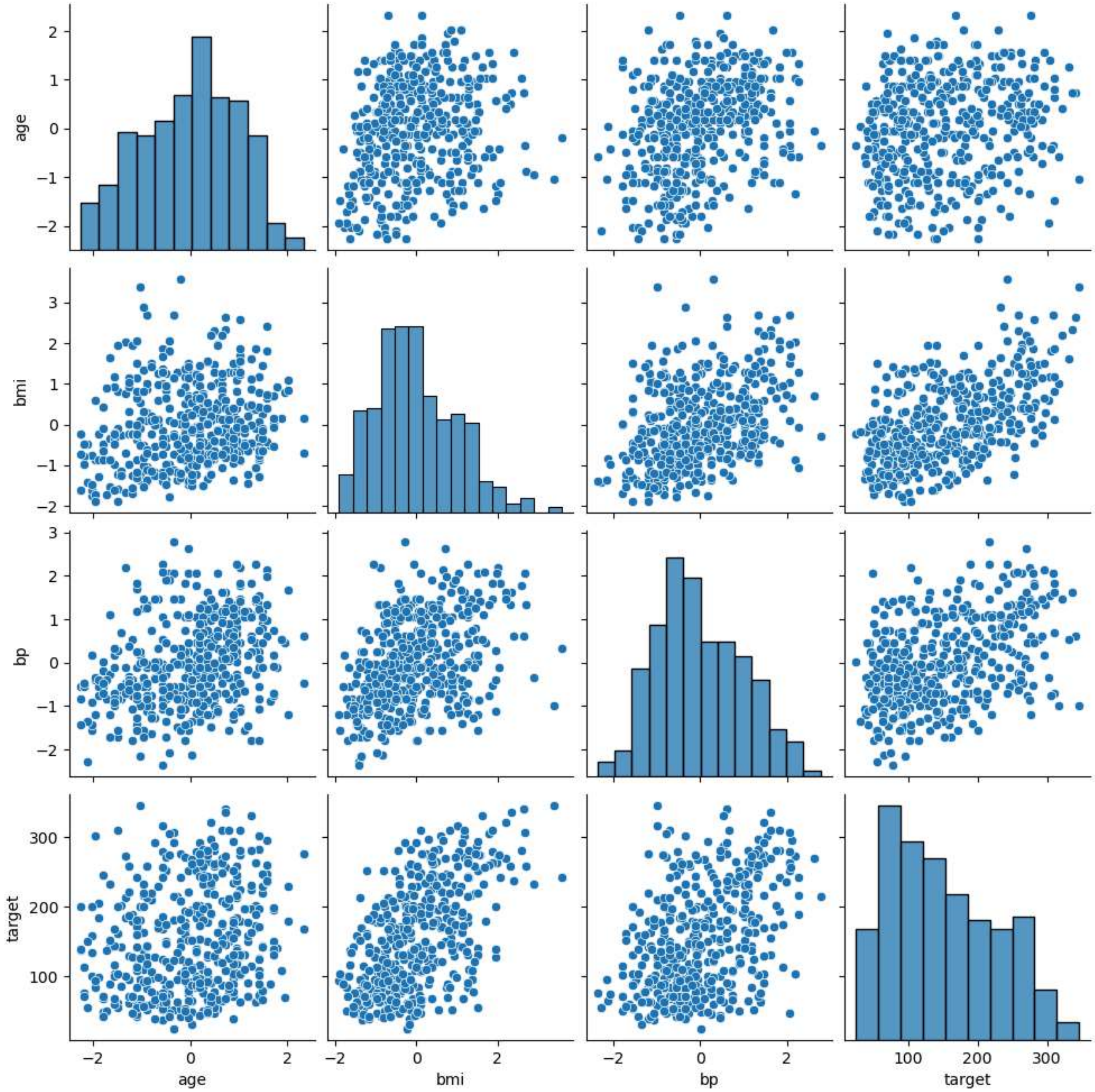
```python
# Visualize correlations between features and the target
plt.figure(figsize=(10, 6))
sns.heatmap(scaled_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix between Features and Target')
plt.show()
```



In [11]:

```python
# Scatterplot of some features with target to explore relationships
sns.pairplot(scaled_df[['age', 'bmi', 'bp', 'target']])
plt.show()

```



In [ ]:

```python
#3. Building the ANN Model
#design a simple ANN model using Keras, with at least one hidden layer and appropriate activation functions.
```

In [12]:

```python
# Import necessary libraries for building the ANN
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


```

In [13]:

```python
# Build the ANN model
model = Sequential()


```

In [14]:
```python
# Input layer and first hidden layer with ReLU activation
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))


```

```
C:\Users\SHONIMA S\AppData\Roaming\Python\Python310\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass a
n `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [15]:
```python
# Second hidden layer
model.add(Dense(32, activation='relu'))


```

In [16]:
```python
# Output layer (for regression, no activation)
model.add(Dense(1))


```

In [17]:
```python
# Compile the model with mean squared error loss function and Adam optimizer
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse'])

```

In [ ]:
```python
#4. Training the ANN Model
#split the dataset into training and testing sets and train the model using the Adam optimizer and mean squared error loss f
```

In [18]:
```python
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2, verbose=1)

```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 3328.5027 - mse: 3328.5027 - val_loss: 3038.6499 - val_mse: 3038.6499
Epoch 54/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 3325.8879 - mse: 3325.8879 - val_loss: 3024.3296 - val_mse: 3024.3296
Epoch 55/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 3616.9775 - mse: 3616.9775 - val_loss: 3009.3462 - val_mse: 3009.3462
Epoch 56/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 3207.6099 - mse: 3207.6099 - val_loss: 2995.4495 - val_mse: 2995.4495
Epoch 57/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 3102.1714 - mse: 3102.1714 - val_loss: 2981.8621 - val_mse: 2981.8621
Epoch 58/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 3500.3020 - mse: 3500.3020 - val_loss: 2966.5056 - val_mse: 2966.5056
Epoch 59/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 3149.5830 - mse: 3149.5830 - val_loss: 2959.3887 - val_mse: 2959.3887
Epoch 60/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 3475.3552 - mse: 3475.3552 - val_loss: 2947.0354 - val_mse: 2947.0354
Epoch 61/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 3178.6528 - mse: 3178.6528 - val_loss: 2938.1443 - val_mse: 2938.1443
Epoch 62/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 3084.7791 - mse: 3084.7791 - val_loss: 2926.0896 - val_mse: 2926.0896
Epoch 63/100
```

In [ ]:
```python
#5. Evaluating the Model
#evaluate the model on the test data and report the performance metrics, such as Mean Squared Error and R² Score.
```

In [19]:
```python
# Import metrics for evaluation
from sklearn.metrics import mean_squared_error, r2_score


```

In [20]:
```python
# Predict on test set
y_pred = model.predict(X_test)


```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
```

In [21]:
```python
# Calculate Mean Squared Error and R² Score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R² Score: {r2}')

```

```
Mean Squared Error: 2889.3804901232
R² Score: 0.4546436894843223
```

In [ ]:

```python
#6. Improving the Model
#different architectures, such as increasing the number of hidden layers, neurons, or using different activation functions l
```

In [22]:

```python
# Rebuilding the model with more layers and neurons
model_improved = Sequential()
```

In [23]:

```python
# Input layer and first hidden layer with tanh activation
model_improved.add(Dense(128, input_dim=X_train.shape[1], activation='tanh'))
```

```
C:\Users\SHONIMA S\AppData\Roaming\Python\Python310\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass a
n `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [24]:

```python
# Second hidden layer with more neurons
model_improved.add(Dense(64, activation='tanh'))
```

In [25]:

```python
# Third hidden layer
model_improved.add(Dense(32, activation='relu'))
```

In [26]:

```python
# Output layer
model_improved.add(Dense(1))
```

In [27]:

```python
# Compile the improved model
model_improved.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), metrics=['mse'])
```

In [28]:

```python
# Train the improved model
history_improved = model_improved.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2, verbose=1)
```

```
Epoch 1/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - loss: 30462.8984 - mse: 30462.8984 - val_loss: 22408.2734 - val_mse: 22408.2734
Epoch 2/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 30698.2129 - mse: 30698.2129 - val_loss: 22238.0391 - val_mse: 22238.0391
Epoch 3/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 31248.4121 - mse: 31248.4121 - val_loss: 22045.9570 - val_mse: 22045.9570
Epoch 4/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 31262.0195 - mse: 31262.0195 - val_loss: 21833.4648 - val_mse: 21833.4648
Epoch 5/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 29341.2539 - mse: 29341.2539 - val_loss: 21590.6133 - val_mse: 21590.6133
Epoch 6/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 27960.4062 - mse: 27960.4062 - val_loss: 21326.8066 - val_mse: 21326.8066
Epoch 7/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 27215.1543 - mse: 27215.1543 - val_loss: 21045.5098 - val_mse: 21045.5098
Epoch 8/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 28403.4883 - mse: 28403.4883 - val_loss: 20734.9473 - val_mse: 20734.9473
Epoch 9/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 29112.6328 - mse: 29112.6328 - val_loss: 20383.4082 - val_mse: 20383.4082
Epoch 10/100
```

In [30]:

```python
# Evaluate the improved model
y_pred_improved = model_improved.predict(X_test)
mse_improved = mean_squared_error(y_test, y_pred_improved)
r2_improved = r2_score(y_test, y_pred_improved)

print(f'Improved Mean Squared Error: {mse_improved}')
print(f'Improved R² Score: {r2_improved}')
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Improved Mean Squared Error: 2759.9279688401743
Improved R² Score: 0.4790772141222813
```

In [ ]:

```
 
```