Programming Steps

Data Wrangling

The first step is data wrangling. As mentioned, this is synthetic data. Mostly data is clean. And there are no outliers present. I check all the steps of data wrangling. This is time series data and regression problem.

Here is the code for data wrangling.

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/Assignment(sensors)/EDA.ipynb

The data is converted into csv data for better analysis

Training Supervised Machine Learning Algorithms

Different machine learning algorithm is tested and here is the summary.

Algorithm	Mean Absolute Error for Systolic BP	Mean Absolute Error for Diastolic BP
Ridge Regression	44.536	10.205
Linear Regression	25.27	83.8463
SVR	36.243	10.2026
SVR (after scaling data)	24.58	48.70

Here is the code. https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/Assignment(sensors)/supervised_ML.ipynb

Training Ensemble Algorithm and Neural Network

Ensemble algorithm performs well on time series data. Here is the result after training algorithm

Algorithm	Mean Absolute Error for Systolic BP	Mean Absolute Error for Diastolic BP
Ensemble Algorithm	35.0619	10.11075
Neural Network	44.269	10.206
Neural Network (tunning parameters)	44.497	10.202

Code for Ensemble Algorithm

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/EnsembleAlgorithm.ipynb

File for Neural Network algorithms

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/Assignment(sensors)/tensorflow_nn.ipynb

Some Code Snippets

```python
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error

# Training models for systolic and diastolic pressures
model_systolic = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, random_state=42)
model_diastolic = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, random_state=42)

# Fit the models
model_systolic.fit(X_train, y_train_systolic)
model_diastolic.fit(X_train, y_train_diastolic)

# Predicting on the test set
y_pred_systolic = model_systolic.predict(X_test)
y_pred_diastolic = model_diastolic.predict(X_test)

# Calculate the MAE for both systolic and diastolic predictions
mae_systolic = mean_absolute_error(y_test_systolic, y_pred_systolic)
mae_diastolic = mean_absolute_error(y_test_diastolic, y_pred_diastolic)

mae_systolic, mae_diastolic
```

```
(35.06195809453011, 10.110753746633229)
```

Neural Network

```python
# Build the model
model = keras.Sequential([
    layers.Dense(512, activation='relu', input_shape=(1000,)),
    layers.Dropout(0.3),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(2)  # Two outputs: Systolic and Diastolic
])
```

✓ 0.1s

```
# Train the model on your training data
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Get the model predictions on test data
y_pred = model.predict(X_test)

# Separate the predictions for Systolic and Diastolic BP
y_pred_S = y_pred[:, 0]
y_pred_D = y_pred[:, 1]

# Calculate MAE for Systolic BP
mae_S = np.mean(np.abs(y_pred_S - y_test_systolic))

# Calculate MAE for Diastolic BP
mae_D = np.mean(np.abs(y_pred_D - y_test_diastolic))

# Print the MAEs
print(f"Test MAE for Systolic BP: {mae_S}")
print(f"Test MAE for Diastolic BP: {mae_D}")
```

Test MAE for Systolic BP: 44.26901359863281, Test MAE for Diastolic BP: 10.206892533874512

Deep Learning Algorithms

Now we train advanced deep learning algorithms. Here are the results attached.

CNN

Now I started testing Deep learning algorithms. Mostly they train on GPU. The link of google colab code file is uploaded on GitHub and attached below.

MAE was achieved with CNN algorithm.

Algorithm	**Mean Absolute Error for Systolic BP**	**Mean Absolute Error for Diastolic BP**
CNN (Base model)	159.4602992392715	79.91103181238296
After fine tunning		
CNN (10 epochs)	45.12046623	10.34711438
CNN (40 epochs)	45.96306772	10.21365163
CNN (50 epochs)	46.15780802	10.2061519

This is the CNN algorithm results tested on google colab

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/sensors_data_analytics_cnn_f.ipynb

LSTM Model

Algorithm	Mean Absolute Error for Systolic BP	Mean Absolute Error for Diastolic BP
LSTM	351	938
LSTM (10 epochs)	45.6076	10.2580
LSTM (50 epochs)	44.4879	10.2097

The link of google colab file is uploaded on GitHub and attached below.

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/sensors_data_analytics_LSTM_f.ipynb

Both LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) are powerful neural network architectures, but they are optimized for different types of data and tasks. LSTM designed for sequential data. It can remember patterns over long sequences. Primarily designed for grid-like data such as images. CNNs can identify hierarchical patterns using convolution layers.

LSTMs are inherently designed for such tasks, It can capture short term and long term patterns in the data which allow them to remember or forget information selectively so LSTM performance is somewhat better than CNN.

Advance DL Algorithms

A lot of deep learning algorithms are tried, the results of some algorithms are in this file. Including inception time

https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/AdvanceDL.ipynb

TCN Algorithm

Algorithm	Mean Absolute Error for Systolic BP	Mean Absolute Error for Diastolic BP
TCN (50 epochs, Shuffle=ON)	21.79	9.79
TCN (80 epochs, Shuffle=Off)	25.8	12.01
TabTransformer Model	50	11.17
Inception time Model	61	13

Both Temporal Convolutional Networks (TCN) and Fully Convolutional Networks (FCN) are types of neural network architectures used for sequence modeling tasks, but they have different design principles and use-cases. **TCN** usually employs dilated convolutions to capture long-range dependencies without increasing the number of parameters or computation. **FCNs** generally have a simpler architecture consisting of a stack of convolutional layers followed by a global pooling layer.

FCN Algorithms

Algorithm	Mean Absolute Error for Systolic BP	Mean Absolute Error for Diastolic BP
FCN (50 epochs)	21.79	9.79
FCN (80 epochs)	25.8	12.01
FCN Plus (50 epochs)	8.913	5.0389
FCN Plus (80 epochs)	4.393	4.3867

Final TCN, FCN and FCN_plus algorithms are trained in this document. Different scaling methods are tried with different algorithms and a lot of hyperparameters are tuned in all the algorithms to get the best results.

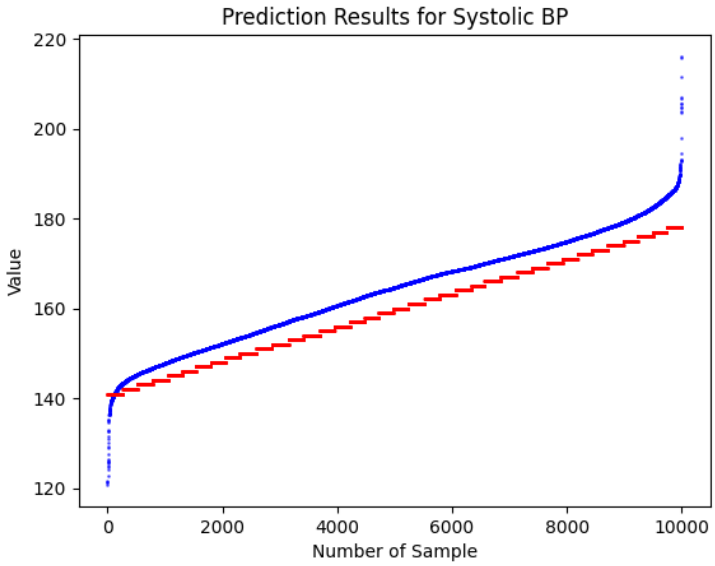
https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/Advance_DL3.ipynb

Testing file

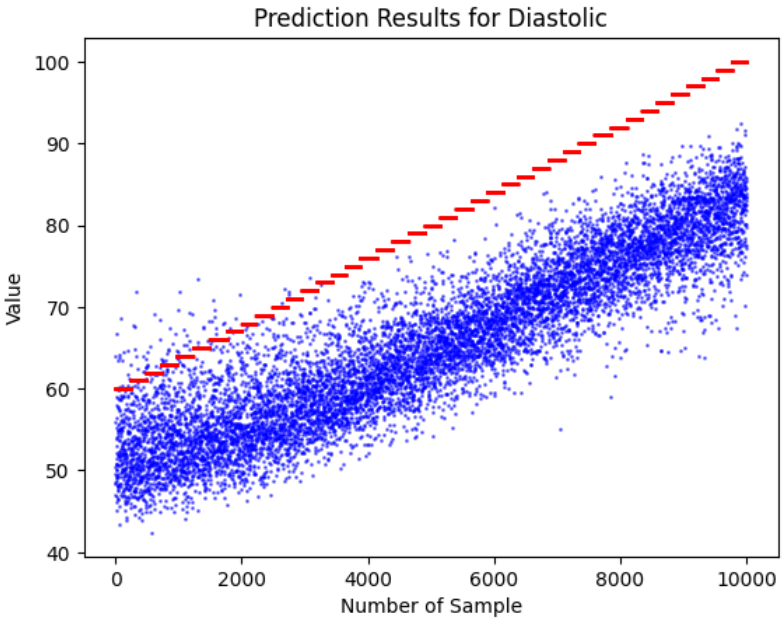
https://github.com/rumman-adnan/Assignment-Sensors-Data/blob/main/FCN_FCN_PLUS_testing.ipynb

Results:

Systolic BP



Diastolic Blood Pressure



Conclusion:

The choice of architecture is problem specific. FCN-Plus is an extended version of the Fully Convolutional Network (FCN) that typically includes additional features like attention mechanisms or other sophisticated layers to improve performance. FCN plus have more complex layers and mechanism, such as attention layers, to better capture the important features in the sequence.

The added complexity could help the model learn to better capture dependencies in the sequence, making it more suited for tasks like time-series forecasting. The extra layers in FCN-Plus could be more effective in extracting useful features from your specific dataset, thereby improving the model's performance.