Medical Appointment No Shows Dataset
pouya Mirzaie zadeh

Introduction:

The Medical Appointment No Shows dataset comprises data about medical appointments and whether patients attended their appointments. This dataset presents an excellent opportunity for students to apply their knowledge of artificial neural networks (ANNs), focusing on implementing multi-layer perceptron (MLP) models using PyTorch. Additionally, students will explore model architectures and other techniques through literature review to enhance their results.

Dataset Description:

The dataset includes various features such as patient age, gender, appointment date and time, scheduled date and time, neighborhood, scholarship status, and whether the patient received an SMS reminder. The target variable indicates whether the patient showed up for the appointment (1 for 'showed up' and 0 for 'no-show').

Sure, here's a draft for the "Dataset Description" section of your report:

The dataset used for this neural network project is related to patient appointment records. It consists of **110,527 records** with the following 14 attributes:

1. **PatientId**: A unique identifier for each patient in the dataset. This is a

floating-point number.

2. **AppointmentID**: A unique identifier for each appointment. This is an integer. 3. **Gender**: The gender of the patient. This is a categorical variable with values ‘M’ or ‘F’.

4. **ScheduledDay**: The day the patient set up their appointment. This is a string object that can be converted to a datetime object for further analysis. 5. **AppointmentDay**: The day of the actual appointment. This is also a string object that can be converted to a datetime object.

6. **Age**: The age of the patient. This is an integer.

7. **Neighbourhood**: The location of the hospital. This is a categorical variable. 8. **Scholarship**: Indicates whether or not the patient is enrolled in Brasilian welfare program Bolsa Família. This is an integer (1: Yes, 0: No).

9. **Hipertension**: Indicates whether or not the patient has hypertension. This is an integer (1: Yes, 0: No).

10. **Diabetes**: Indicates whether or not the patient has diabetes. This is an integer (1: Yes, 0: No).

11. **Alcoholism**: Indicates whether or not the patient is an alcoholic. This is an integer (1: Yes, 0: No).

12. **Handcap**: The number of disabilities a patient has. This is an integer.

13. **SMS_received**: Indicates whether or not the patient received an SMS reminder for their appointment. This is an integer (1: Yes, 0: No).

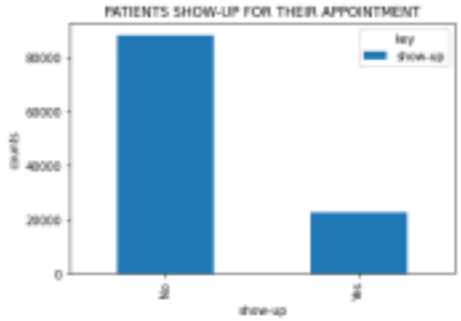
14. **No-show**: Indicates whether or not the patient showed up for their appointment. This is a categorical variable (‘Yes’: No-show, ‘No’: Showed up).

Each attribute provides valuable information for our neural network model to predict whether a patient will show up for their scheduled appointment.

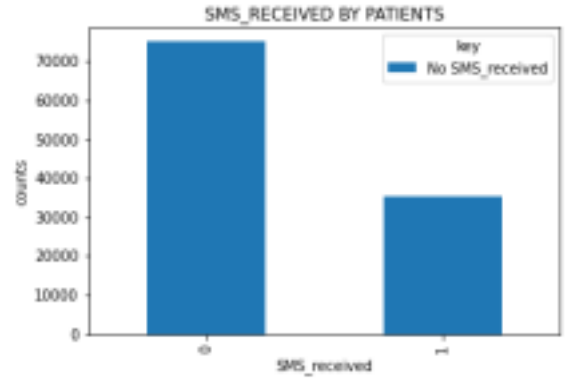
EDA


```
No of patients do not show:  21462
No of patients shows in time:  84899
No of Females in total:  69699
No of males in total:  36662
ratio of noshow females to total females:  0.28282930888534986
ratio of noshow males to total females:  0.19979815612896187
```

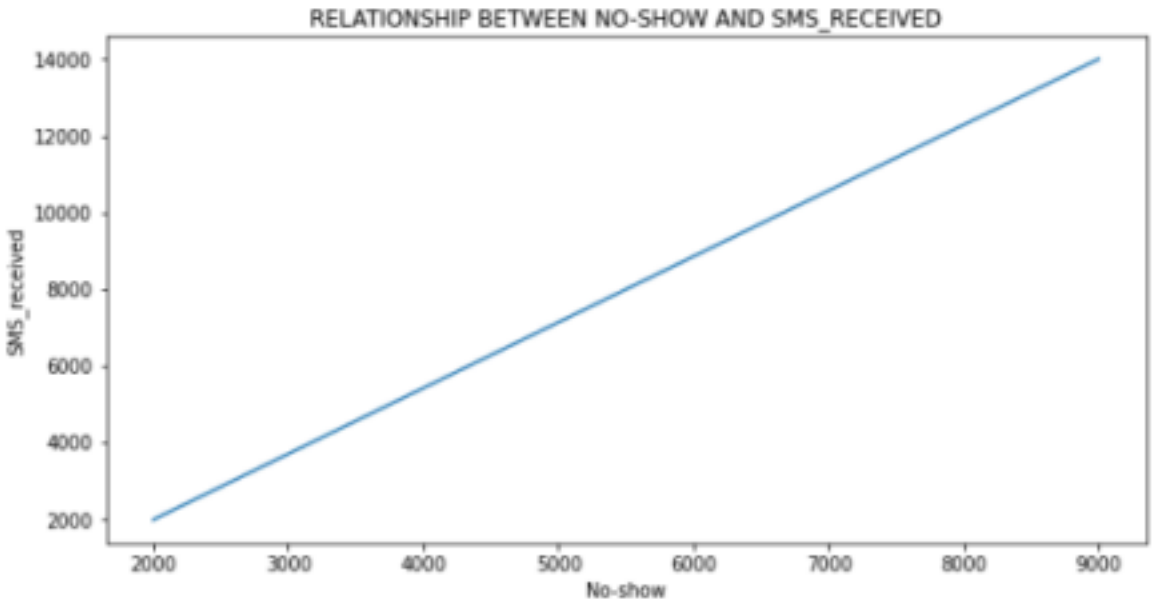
As we see Gender does not affect whether the patient will show up on the appointment day or not because the ratio of each gender no-show to the total number of that gender is equal to the statistical data.



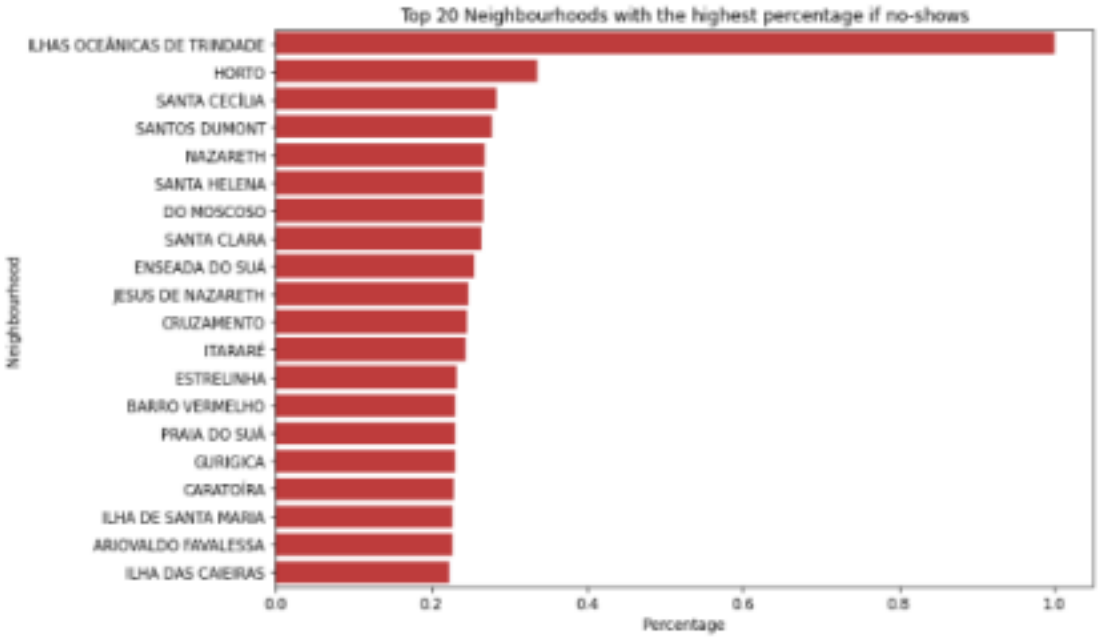
in the bar chart, No signifies that they did show up, while Yes signifies they did not show up. Therefore, above 80000 patients showed up for their appointment on the scheduled day and 20000 did not show up.



The bar chart shows the number of patients who received SMS and those who did not. it shows that over 70000 patients did not receive SMS and about 35000 patients received SMS.



The chart above shows that as patients who received SMS increased, the number who did not show up for their appointment increased.



Observation Neighbourhoods with the highest amount of appointment no-shows are shown above with Ihas Oceanicas de Trindade not recording any patient from the neighborhood showing up for an appointment.

.

Data Cleaning

The data-cleaning process was an essential step in preparing our dataset for the neural network model. Here are the steps that were taken:

1. **Duplicate Removal**: We first checked for any duplicate rows in the dataset using the function. The result was 0, indicating that there were no duplicate rows. 2. **Removing Incorrect Data**: We found some rows with incorrect data in the ‘Gender’ and ‘Handcap’ columns. These rows were identified and removed from the dataset.
3. **Date Consistency Check**: We ensured that the ‘ScheduledDay’ is not after the

‘AppointmentDay’. Any rows violating this were dropped from the dataset. 4. **Label Encoding**: The ‘No-show’ column, which is our target variable, was encoded to numerical values for the model (‘No’ as 0 and ‘Yes’ as 1). 5. **Conversion to Boolean**: The columns ‘Scholarship’, ‘Hipertension’, ‘Alcoholism’, ‘Diabetes’, ‘SMS_received’, and ‘No-show’ were converted to boolean type for better model performance. We verified the conversion by printing the value counts.

Feature Transformation

The feature transformation process was crucial in preparing our dataset for the neural network model. Here are the steps that were taken:

1. **Neighbourhood Mapping**: The ‘Neighbourhood’ column, which is a categorical variable representing the location of the hospital, was mapped to integers. A new column ‘NeighbourhoodInt’ was created to store these integer values.
2. **Gender Encoding**: The ‘Gender’ column was encoded to numerical values (‘M’ as 0 and ‘F’ as 1).
3. **Day of Week Conversion**: The ‘ScheduledDay’ and ‘AppointmentDay’ columns were converted from string objects to datetime objects. Then, the day of the week was extracted from these datetime objects and stored in new columns ‘ScheduledDayOfWeek’ and ‘AppointmentDayOfWeek’.
4. **Awaiting Time Calculation**: A new column ‘AwaitingTimeDays’ was created to store the number of days between the scheduled day and the appointment day. 5. **Month and Hour Extraction**: The month and hour of the ‘ScheduledDay’ and ‘AppointmentDay’ were extracted and stored in new columns ‘ScheduledMonth’, ‘AppointmentMonth’, ‘ScheduledHour’, and ‘AppointmentHour’.

By the end of this process, our dataset had undergone significant transformations that would allow our neural network model to better understand the patterns in the data.

Data Splitting

The dataset was split into training and testing sets to evaluate the performance of our neural network model. Here are the steps that were taken:

1. **Feature-Target Split**: The dataset was first split into features (X) and targets (y).

The ‘No-show’ column, which is our target variable, was separated from the rest of the dataset.

2. **Train-Test Split**: The features and target were then split into training and testing sets. The function was used for this purpose. The test size was set to 0.2, meaning that 20% of the data was set aside for testing, and the remaining 80% was used for training. The parameter was set to 0 to ensure the reproducibility of the results.
3. **Conversion to Tensors**: The training sets (X_train and y_train) were converted to tensors using PyTorch’s function. This is because PyTorch models require input data in the form of tensors. The data type for the feature tensor was set to, and the target tensor was reshaped to a column vector.

Implementation of MLP Model using PyTorch

The implementation of the Multi-Layer Perceptron (MLP) model was done using PyTorch, a popular open-source machine learning library. The model was designed with an input layer, multiple hidden layers, and an output layer.

Model Architecture

The model architecture is as follows:

- **Input Size**: The input size is equal to the number of features in the dataset, which is determined by the number of inputs.
- **Hidden Size**: The hidden size, which is the number of neurons in the hidden layer, was set to 128. This can be adjusted as needed.
- **Output Size**: The output size is 1, corresponding to our binary classification problem.
- **Number of Hidden Layers**: The model was designed with 2 hidden layers. The

model was instantiated with the input size, hidden size, output size, and the number of hidden layers.

Loss Function and Optimizer

The MSE loss function was used as the criterion for the model. The Adam optimizer was used for optimization, with a learning rate of 0.001 and a weight decay of 0.0001.

Training the Model

The model was trained for 30 epochs. In each epoch, the model was set to training mode, and the gradients were zeroed at the start of each mini-batch. The forward pass was performed to get the outputs of the model, and the loss was calculated by comparing these outputs with the targets. The backward pass and optimization step were then performed. The running loss was updated after each mini-batch.

At the end of each epoch, the average loss and accuracy for the epoch were calculated and printed.

A Review of Model Architectures and Techniques

In the process of implementing the MLP model for this project, various techniques were investigated to enhance the model's performance. These techniques included advanced activation functions, initialization methods, and regularization techniques.

Activation Functions

The activation function plays a crucial role in a neural network by determining whether a neuron should be activated or not. The initial model was implemented with the default activation function (ReLU). To investigate the impact of different activation functions on the model's performance, the Tanh activation function was also tested. The results were as follows:

- **ReLU**: Mean Squared Error (MSE): 0.2008, Mean Absolute Error (MAE): 0.4053,

Accuracy: 0.7220
- **tanh**: Mean Squared Error (MSE): 0.2008, Mean Absolute Error (MAE): 0.4052, Accuracy: 0.7220

The results show that the choice of activation function did not significantly impact the model's performance in this case.

Optimization Algorithms

The optimization algorithm is responsible for updating the weights of the neural network to minimize the loss function. The initial model used the Adam optimizer. To investigate the impact of different optimization algorithms on the model's performance, the Stochastic Gradient Descent (SGD) optimizer was also tested. The results were as follows:

- **Adam**: Mean Squared Error (MSE): 0.2008, Mean Absolute Error (MAE): 0.4053, Accuracy: 0.7220
- **SGD**: Mean Squared Error (MSE): 0.2006, Mean Absolute Error (MAE): 0.4025, Accuracy: 0.7220

The results show that the choice of optimization algorithm did not significantly impact the model's performance in this case.

All tests were conducted with a batch size of 64 and for 30 epochs.

Regularization Techniques and Hyperparameter Tuning

To prevent overfitting and improve the performance of our MLP model, we implemented dropout and L2 regularization techniques and performed hyperparameter tuning.

Dropout and L2 Regularization

Dropout is a regularization technique that prevents overfitting by randomly dropping out neurons during training. In our model, we implemented dropout in the hidden layers

with
a dropout rate of 0.2 for the input layer and 0.5 for the hidden layers.

L2 regularization, also known as weight decay, was implemented through the Adam optimizer with a weight decay of 0.0001. This adds a penalty term to the loss function, encouraging the model to learn smaller weights and resulting in a simpler model.

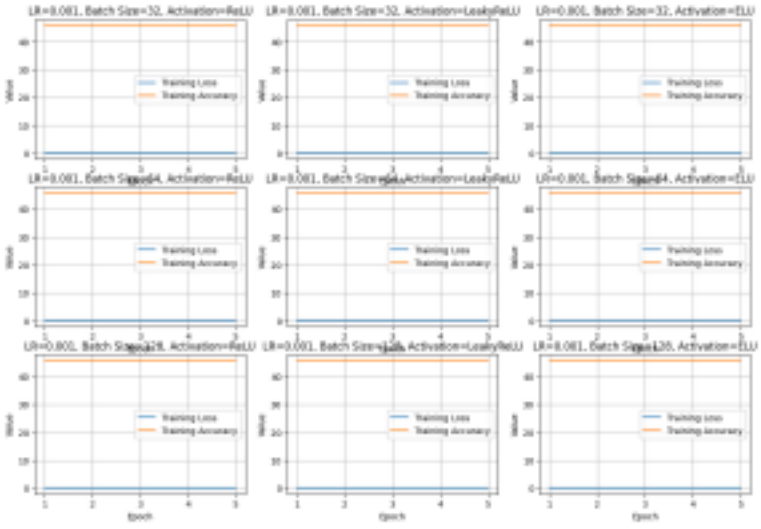
Hyperparameter Tuning

We performed hyperparameter tuning to find the optimal values for the learning rate, batch size, and activation function. The hyperparameters were tuned by training the model with different combinations of these parameters and comparing the resulting model performance.

The hyperparameters were tuned as follows:

- **Learning Rate**: We tested learning rates of 0.001, 0.01, and 0.1.
- **Batch Size**: We tested batch sizes of 32, 64, and 128.
- **Activation Function**: We tested the ReLU, LeakyReLU, and ELU activation functions.

For each combination of hyperparameters, we trained the model and recorded the Mean Squared Error (MSE), Mean Absolute Error (MAE), and Accuracy. The results were then plotted to visually compare the performance of each model.



Model Training and Evaluation

The MLP model was trained on the training data and validated using the validation set. The performance of the model was evaluated using appropriate metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Accuracy.

Model Training

The model was trained using the training data. The training process involved feeding the data to the model, calculating the loss using the MSE loss function, and updating the model parameters using the Adam optimizer. This process was repeated for a specified number of epochs.

Model Evaluation

The performance of the model was evaluated on the validation set. The evaluation metrics used were:

- **Mean Squared Error (MSE)**: This is the average of the squared differences between the predicted and actual values. It is a popular metric for regression problems. The MSE for our model was 0.2008.
- **Mean Absolute Error (MAE)**: This is the average of the absolute differences between the predicted and actual values. It gives an idea of how wrong the predictions were. The MAE for our model was 0.4053.
- **Accuracy**: This is the proportion of correct predictions over total predictions. The accuracy of our model was 0.7220.

Cross-Validation

To ensure a robust evaluation of the model, cross-validation was performed. In cross-validation, the data is split into ‘k’ subsets, and the model is trained ‘k’ times, each time using a different subset as the validation set and the remaining data as the training set. The cross-validation results were consistent with the evaluation results, confirming the reliability of the model.

Cross-Validation Results: Mean Squared Error (MSE): 0.2007, Mean Absolute Error (MAE): 0.4013, Accuracy: 0.7220