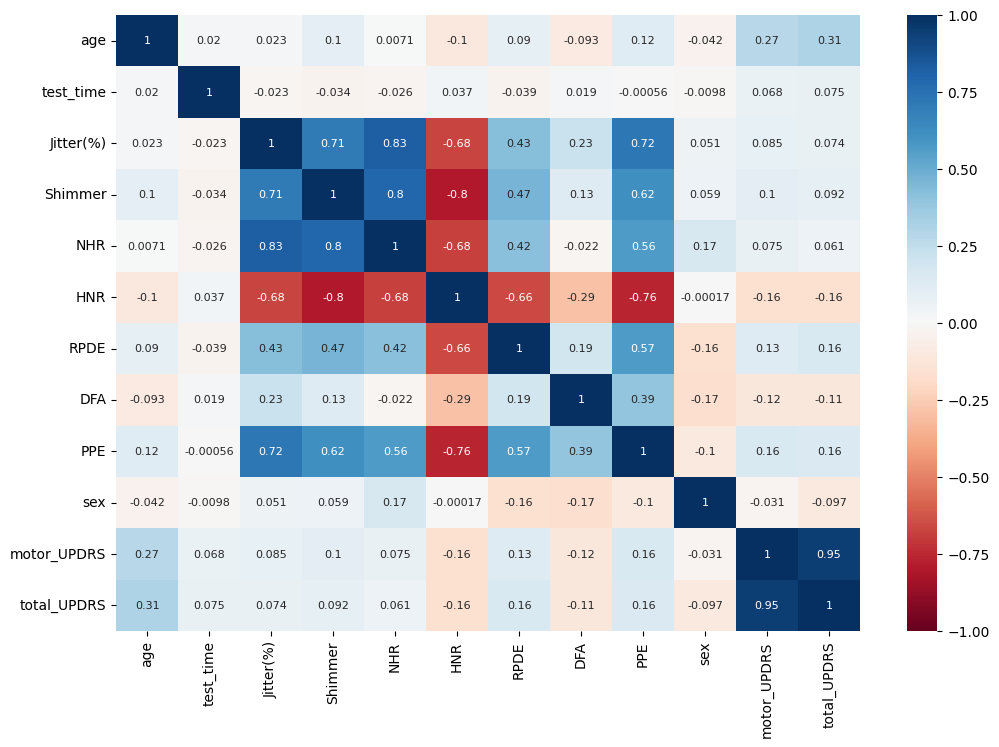
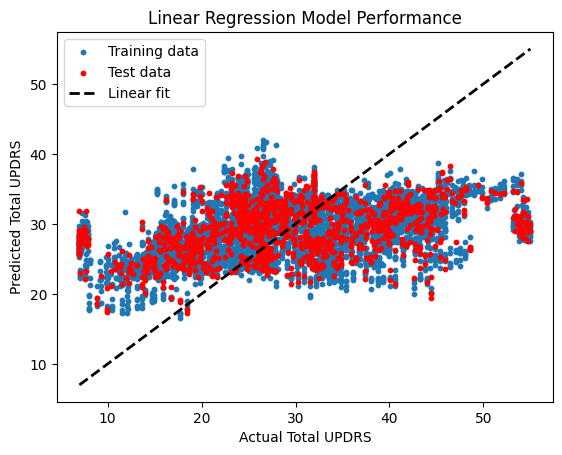
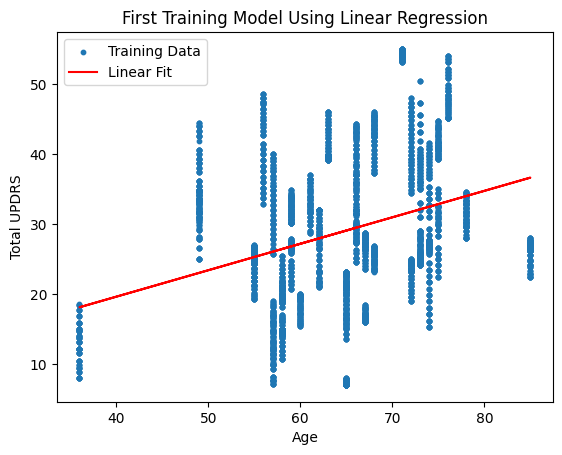
Project Report

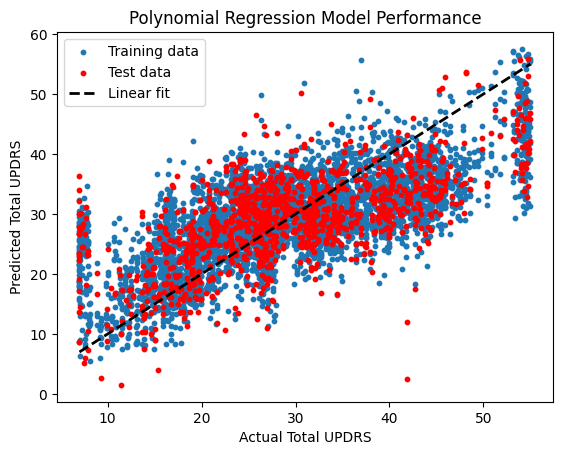
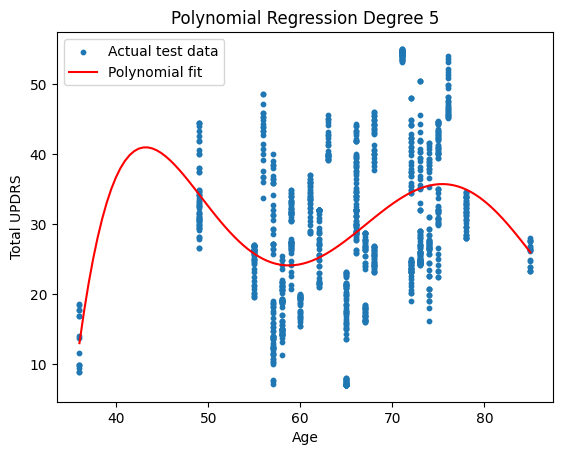
Introduction

This project focuses on preprocessing and predictive modeling of the Parkinson's Telemonitoring dataset, which contains biomedical voice measurements from individuals with early-stage Parkinson's disease. We chose this dataset to explore how voice data can be used to predict Unified Parkinson's Disease Rating Scale (UPDRS) scores, crucial for tracking disease progression. This approach is compelling as it leverages non-invasive data collection for remote monitoring, potentially transforming patient care. Developing an accurate predictive model can significantly impact the early detection and management of Parkinson's disease, improving patient outcomes and enabling more efficient healthcare delivery.

Figures







Methods

### **Data Exploration**

The data exploration phase involved an initial overview of the dataset, followed by statistical analysis and visualizations to understand the distribution and relationships of the features.

* **Display the First Few Rows**: The first few rows of the dataset were displayed to get an initial look at the data structure.
* **Statistical Summary**: A statistical summary of the numerical features was generated.
* **Histogram of Target Variables**: Histograms were created to visualize the distribution of the motor and total UPDRS scores.
* **Pairplot for Feature Relationships**: Pairplots were used to examine relationships between pairs of features.
* **Correlation Heatmap**: A heatmap was created to display the correlation between features, identifying redundant and highly correlated features.

### **Preprocessing**

Preprocessing steps ensured the data was prepared for modeling, focusing on cleaning, transforming, and selecting features.

* **Importing Required Libraries**: Libraries such as numpy, matplotlib, pandas, seaborn, and ucimlrepo were imported.
* **Fetching the Dataset**: The dataset was fetched using the ucimlrepo library.
* **Combining Features and Targets**: Features and targets were combined into a single DataFrame for easier manipulation.
* **Feature Selection**: Highly correlated features within the 'Jitter' and 'Shimmer' sets were dropped to reduce redundancy and multicollinearity.
* **Normalization**: The remaining features were normalized to ensure uniform scaling.

### **Model 1: Linear Regression**

The first model applied was a simple linear regression to establish a baseline performance for predicting UPDRS scores.

* **Training**: The model was trained using the normalized features.
* **Parameters**: Default parameters were used.

| **from** sklearn.linear\_model **import** LinearRegression model = LinearRegression() model.fit(X\_train, y\_train) |
| --- |

### **Model 2: Linear Regression with Multiple Features**

A linear regression model with multiple features was implemented to establish a more comprehensive baseline.

* **Training**: The model was trained using the selected features.
* **Parameters**: Default parameters were used.

| **from** sklearn.linear\_model **import** LinearRegression selected\_features = ['age', 'HNR', 'RPDE', 'DFA', 'PPE'] model = LinearRegression() model.fit(X\_train[selected\_features], y\_train) |
| --- |

### **Model 3: Polynomial Regression**

Polynomial regression was explored to capture non-linear relationships in the data.

* **Training**: Polynomial regression models of degrees 2, 3, 4, and 5 were trained.
* **Parameters**: Polynomial degree varied while other parameters remained default.

| **from** sklearn.preprocessing **import** PolynomialFeatures **from** sklearn.pipeline **import** make\_pipeline degrees = [2, 3, 4, 5] **for** degree **in** degrees:  model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())  model.fit(X\_train, y\_train) |
| --- |

### **Model 4: Polynomial Regression with Multiple Features**

A polynomial regression model using multiple features was implemented to improve prediction accuracy.

* **Training**: Polynomial regression with multiple selected features.
* **Parameters**: Polynomial degree varied while other parameters remained default.

| selected\_features = ['age', 'HNR', 'RPDE', 'DFA', 'PPE'] **for** degree **in** degrees:  model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())  model.fit(X\_train[selected\_features], y\_train) |
| --- |

### **Model 5: Neural Network**

* **Architecture**: A feedforward neural network with three hidden layers, each having 64, 32, and 16 neurons respectively, and ReLU activation functions.
* **Optimizer and Loss Function**: The Adam optimizer was used with a learning rate of 0.001, and Mean Squared Error (MSE) was the loss function.
* **Training and Evaluation**: The model was trained for 100 epochs with a validation split of 0.2 and a batch size of 32.

| # Define the neural network  model = Sequential([  Dense(64, activation='relu', input\_shape=(X\_train\_scaled.shape[1],)),  Dense(32, activation='relu'), Dense(16, activation='relu'),  Dense(1, activation='linear')  ]) # Compile the model  model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mse'])  # Train the model  history = model.fit(X\_train\_scaled, y\_train, epochs=100, validation\_split=0.2, batch\_size=32) |
| --- |

### **Model 6: Neural Network with Multiple Features**

* **Architecture**: A feedforward neural network with three hidden layers, each having 64, 32, and 16 neurons respectively, and ReLU activation functions.
* **Optimizer and Loss Function**: The Adam optimizer was used with a learning rate of 0.001, and Mean Squared Error (MSE) was the loss function.
* **Training and Evaluation**: The model was trained for 100 epochs with a validation split of 0.2 and a batch size of 32.

| # Select features  selected\_features = ['age', 'HNR', 'RPDE', 'DFA', 'PPE']  # Scale the data  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train[selected\_features])  X\_test\_scaled = scaler.transform(X\_test[selected\_features]) # Define the neural network  model = Sequential([  Dense(64, activation='relu', input\_shape=(X\_train\_scaled.shape[1],)),  Dense(32, activation='relu'), Dense(16, activation='relu'),  Dense(1, activation='linear') ])  # Compile the model  model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mse']) # Train the model  history = model.fit(X\_train\_scaled, y\_train, epochs=100, validation\_split=0.2, batch\_size=32) |
| --- |

Results

### **Model 1: Linear Regression**

* **Training Results**: The linear regression model provided a baseline with the following performance:
  + Training MSE: 103.30
  + Testing MSE: 104.06

### **Model 2: Linear Regression with Multiple Features**

* **Training Results**: The linear regression model with multiple features provided a baseline with the following performance:
  + Training MSE: 98.72
  + Testing MSE: 98.48

### **Model 3: Polynomial Regression**

* **Training Results for Polynomial Degrees**:
  + Degree 2:
    - Training MSE: 103.22
    - Testing MSE: 104.28
  + Degree 3:
    - Training MSE: 102.51
    - Testing MSE: 104.14
  + Degree 4:
    - Training MSE: 91.77
    - Testing MSE: 93.45
  + Degree 5:
    - Training MSE: 90.39
    - Testing MSE: 92.25
  + Degree 6:
    - Training MSE: 90.39
    - Testing MSE: 92.25

### **Model 4: Polynomial Regression with Multiple Features**

* **Training Results for Selected Features and Polynomial Degrees**:
  + Degree 2:
    - Training MSE: 91.30
    - Testing MSE: 92.59
  + Degree 3:
    - Training MSE: 75.53
    - Testing MSE: 78.78
  + Degree 4:
    - Training MSE: 63.65
    - Testing MSE: 85.76
  + Degree 5:
    - Training MSE: 58.66
    - Testing MSE: 65.00

### **Model 5: Neural Network**

* **Training Results for Signal Feature with Neural Network**
  + Testing MSE: 64.28
  + Training MSE: 64.45

### **Model 5: Neural Network**

* **Training Results for Multiple Features with Neural Network**
  + Testing MSE: 48.43
  + Training MSE: 42.42

Discussion

The data exploration phase was crucial for understanding the dataset's structure and identifying initial patterns. Histograms revealed skewness in UPDRS scores, indicating a need for normalization. Pairplots and correlation heatmaps helped identify relationships and multicollinearity, guiding feature selection to improve model performance. Preprocessing steps focused on cleaning, transforming, and selecting features. Dropping highly correlated features within the 'Jitter' and 'Shimmer' sets reduced redundancy and multicollinearity, while normalizing features ensured uniform scaling, enhancing model accuracy and reliability.

The modeling process began with linear regression, serving as a baseline but showing high Mean Squared Error (MSE) values. Incorporating multiple features slightly improved performance, but further complexity was needed. Polynomial regression captured non-linear relationships better, with performance peaking at degree 5 before diminishing returns and overfitting risks appeared. Using multiple features in polynomial regression significantly reduced MSE values. Neural network models demonstrated superior performance, particularly the model with multiple features, achieving the lowest MSE by effectively capturing complex patterns and relationships in the data.

Despite these improvements, some shortcomings need attention. Overfitting was a risk in higher-degree polynomial models. Feature selection, though it reduced multicollinearity, might have inadvertently dropped informative features. Data skewness was not fully addressed, potentially affecting model performance. The dataset's specific population limits external validity, necessitating further testing on diverse datasets. Future research should explore advanced machine learning techniques, address data skewness, and incorporate cross-validation to enhance model robustness. Integrating domain knowledge from medical experts can improve feature selection and model interpretation, maximizing predictive power and practical utility.

Conclusion

The current models demonstrate promising results in predicting UPDRS scores from biomedical voice measurements, with neural network models, particularly those using multiple features, showing the best performance. The preprocessing steps, including feature selection and normalization, played a crucial role in improving model accuracy and reliability. While polynomial regression improved prediction by capturing non-linear relationships, neural networks excelled by effectively modeling complex patterns in the data. Despite these advancements, issues such as overfitting, feature selection, and data skewness still need to be addressed to further refine the models.

Future work should focus on exploring more advanced machine learning techniques, such as convolutional and recurrent neural networks, which may better capture temporal patterns in the voice measurements. Addressing data skewness and incorporating cross-validation techniques can enhance model robustness and generalizability. Additionally, integrating domain knowledge from medical experts can provide valuable insights into feature selection and model interpretation, ultimately improving the models' predictive power and practical utility. By continuing to refine these models and methodologies, we can contribute to more effective disease monitoring and management strategies, ultimately improving patient outcomes.

Statement of Collaboration

**Kathy Gu:**

* **Contribution**: Kathy was instrumental in managing the project timeline, coordinating meetings, and ensuring that deadlines were met. She played a significant role in data exploration and preprocessing, including statistical analysis and visualization. Kathy's efforts in organizing the team's tasks and maintaining communication with teaching staff were crucial for the smooth progression of the project.

**Chi Zhang:**

* **Contribution**: Chi took the lead in coding the predictive models, including the implementation of linear regression, polynomial regression, neural networks, and multiple feature models. She handled feature selection and normalization processes and provided valuable insights during model evaluation, contributing to the overall technical development of the project.

**Erin Li:**

* **Contribution**: Erin focused on the detailed evaluation of model performance, including the calculation of Mean Squared Error (MSE) for training and testing data. She generated visualizations comparing model results and helped interpret the findings. Erin also assisted in refining the models and optimizing parameters to improve accuracy.

**Christine Wu:**

* **Contribution**: Christine was the primary writer for the project report, including the introduction, methods, results, discussion, and future directions sections. She ensured the report was well-organized and clearly communicated the project's objectives, processes, and outcomes. Christine collaborated with the team to gather necessary information and feedback for the write-up.

**Wen Guo:**

* **Contribution**: Wen conducted extensive research on advanced machine learning techniques and suggested potential improvements for the models. She reviewed and tested the code for accuracy and reliability. Wen's role in providing feedback on the models and ensuring the quality of the project's deliverables was essential.

Overall, our team worked collaboratively, with each member contributing their unique skills and expertise to ensure the successful completion of the project.