A SMART EARLY DETECTION AND PREDICTION SYSTEM FOR PARKINSON'S DISEASE USING DATA MINING AND MACHINE LEARNING

Audrey Han [1], Howard Lee [2]

[1] Walnut High School, 400 Pierre Rd, Walnut, CA 91789
[2] California State University, Long Beach, 1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

Parkinson's disease is a prevalent neurodegenerative disease that primarily affects the elderly. While it currently has no cure, early detection is essential in treating and reducing symptoms. Our project aims to make prediction of Parkinson's disease more accessible and convenient for patients by using keystroke data which can be easily gathered by any keyboard. In order to achieve this, we used machine learning, specifically predictive machine learning. We tested several different types of models including Random Forest Classifier, Logistic Regression, Gradient Boosting, and Support Vector Machines. In the end, our logistic regression model had the highest level of accuracy. We proceed to test the models with simulated data, once with a simulated sample of all positive cases and once with a sample of a mixed of positive and negative cases. These experiments supported our conclusion that logistic regression was our most reliable model. We then created a website that allows users to predict whether or not they have Parkinson's using their own keystroke data. This made our model accessible to the general public and easy to use, encouraging proactive testing and preventative measures.

KEYWORDS

Parkinson's, Machine Learning, Python, Scikit-learn

1. INTRODUCTION

With a rapidly aging population, neurological diseases have become the forefront of medical research. Alzheimer's, epilepsy, amyotrophic lateral sclerosis (ALS), and Bell's palsy are all conditions that threaten the health of the elderly population. Parkinson's disease, in particular, blah blah blah.

Parkinson's disease is a progressive neurodegenerative disorder that affects movement, causing neurons in the brain to weaken and die [5]. Symptoms of this disorder often include tremors, rigidity, and impaired movements [6]. With approximately 0.5-1% of people aged from 65 to 69 years old and 1-3% of people aged 80 and older suffering from Parkinson's, it is the second most common neurodegenerative disease following Alzheimer's disease. Parkinson's disease often impacts patients' quality of life, leading to physiological, physical, and mental problems [3].

David C. Wyld et al. (Eds): MLCL, CSITA, NMCO, AIRN, CONEDU – 2026
pp. 29-38, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160103

The first clear medical description of Parkinson's disease was in 1817 by James Parkinson. Since this discovery, many methods of treatment have been tested. Levodopa, for example, is one such treatment that converts to dopamine in the brain [2]. Other treatments like physical therapy and rehabilitation are also used to reduce symptoms. However, Parkinson's disease is a chronic illness with no current cure.

Additionally, despite the prevalence of the disease, there is no standardized test for detecting Parkinson's. Since it lacks a specific biomarker, diagnosis largely happens in the clinical setting based on common symptoms. As a result, the misdiagnosis rate for Parkinson's is high, with neurologists having a misdiagnosis rate of 10-20% and up to 25% for non-specialists [9][10].

Our solution was to create a Parkinson's disease prediction model using machine learning to analyze keystroke data of possible patients. Since Parkinson's disease affects motor functions, including causing tremors, it can alter patients' keystroke patterns [12]. These abnormalities can be detected through machine learning. After testing several different models such as Random Forests and Gradient Boosting, we found that a logistic model was the most accurate. By examining various features ranging from flight time between key presses and hold times, our model was able to determine with 78.5% accuracy whether or not a patient had Parkinson's disease [14].

Unlike many other methods of diagnosis that are not standardized and oftentimes expensive, our innovative method using keystroke data is cheap and accessible, requiring only a keyboard and device. This is especially important for elderly patients who cannot travel to hospitals or other medical facilities for specialized testing. We created a Streamlit that allows users to input their own keystroke data and predict whether or not they have Parkinson's disease using our model. This easy-to-use website makes diagnosis quick and efficient, returning a prediction in real time.

Although Parkinson's currently has no cure, early diagnosis is important in slowing and mitigating the effects of the disease. Our model makes testing convenient and noninvasive, encouraging testing and increasing the likelihood of reaching a diagnosis in the early stages.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Database

Our first issue was cleaning and organizing our data. The data we used originally came in several different unlabeled text files, one set containing the user data and another set containing the keystroke data associated with each user. To solve this, I could parse through both sets of files, matching the users with their related keystroke data through their user IDs. Then, I could organize these separate files into one CSV file, containing all the users, their user data, and the averages and standard deviations of their keystroke features. Additionally, I could remove any missing entries and replace them with the median of that feature.

2.2. Machine Learning Model

Another issue we ran into was building our machine learning model. Initially, we trained a Random Forest Classifier model using only their average and standard deviations of the users' hold time, flight time, and latency. However, this first model suffered from poor accuracy. In order to improve our model, we added a scalar to increase the amount of training data available.

Additionally, we added more features including age, sidedness, and gender. We then tested this with several different models such as Gradient Boosting, Support Vector Machine, and Logistic Regression. We found that logistic regression had the highest accuracy of 78%.

2.3. Useable Web

While adapting our model to a usable website, we ran into several issues. First, the features we needed to analyze for our machine learning model such as their hold time, flight time, and latency were unable to be recorded by our website. Therefore, users had to calculate and input these values themselves before using our model. Additionally, when attempting to add our model into our Streamlit website, we realized that our model was too big to be supported. As a result, we had to compress our model by using a simpler training algorithm. This, consequently, reduced the accuracy of our model.

3. SOLUTION

Our product consists of three major components. Initially, we begin with our dataset of users with and without Parkinson's Disease and their keystroke data. Our dataset originally came segmented in different files and in an usable format to our code. The user's ID and demographic information such as age and gender was separated from the user's keystroke data. As a result, we utilized the Pandas library in Python to convert our scattered data into dataframes. We then dedicated one component of our project to merging and cleaning the dataset to group the users' data into a usable format.

The second major component of our project is our machine learning model. We created this model using Python as it is the popular choice among machine learning scientists along with Scikit-learn for its numerous built-in packages. We tested several different types of models such as Random Forest Classifier, Gradient Boosting Classifier, and Support Vector Machine. Eventually, we settled on Logistic Regression as the most accurate model. While tuning our model, we had to add a scalar that multiplied the amount of data and an imputer that replaced empty columns with the median for the feature to increase our accuracy.

To make our model accessible to users, we implemented our third component: our website. Using Streamlit, a quick platform to create websites in Python without the necessity of HTML, we created a page that allows users to input their own data and run it through our model to get a real-time prediction of their likelihood of currently having Parkison's Disease.

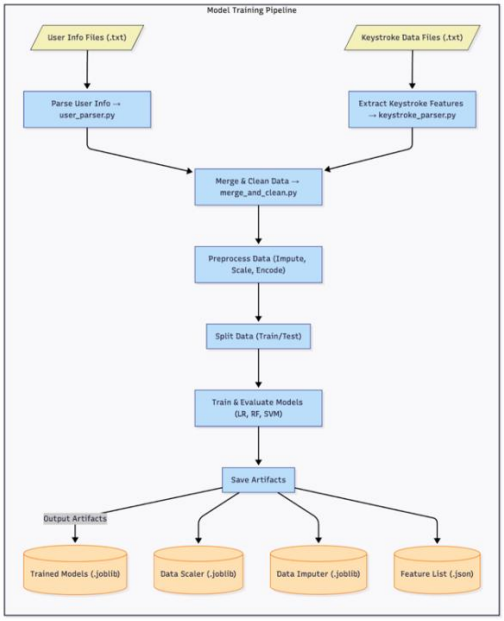

Figure 1. Overview of the solution

The authentication component allows users to create secure accounts by verifying and storing passwords in Firebase. It ensures that only authorized users can access the app's features. By providing a secure gateway, authentication protects user data and enables other processes, such as posting shots and interacting with the ML model.

In our first component, we used Pandas in order to clean and merge our scattered data into a usable dataframe for our future model. We needed to do so because our original data was split across several different files that separated user demographic and our keystroke data.

```python
def get_keystroke_features_dataframe(user_df, data_directory="dataset/Archived Data/"):
    def create_features_for_user(user_key, data_directory):
        hold_times = []
        latency_times = []
        flight_times = []

        for filename in os.listdir(data_directory):
            if filename.endswith('.txt') and filename.startswith(user_key):
                filepath = os.path.join(data_directory, filename)
                try:
                    temp_df = pd.read_csv(
                        filepath,
                        sep='\t',
                        header=None,
                        usecols=range(8),
                        dtype=str,
                        low_memory=False
                    )

                    temp_df.columns = [
                        'UserKey', 'Date', 'Timestamp', 'Hand', 'Hold Time', 'Direction', 'Latency Time', 'Flight Time'
                    ]

                    temp_df['Hold Time'] = pd.to_numeric(temp_df['Hold Time'].str.replace(',', '.'), errors='coerce')
                    temp_df['Latency Time'] = pd.to_numeric(temp_df['Latency Time'].str.replace(',', '.'), errors='coerce')
                    temp_df['Flight Time'] = pd.to_numeric(temp_df['Flight Time'].str.replace(',', '.'), errors='coerce')

                    hold_times.extend(temp_df['Hold Time'].dropna().tolist())
                    latency_times.extend(temp_df['Latency Time'].dropna().tolist())
                    flight_times.extend(temp_df['Flight Time'].dropna().tolist())
                except Exception as e:
                    print(f"Error {e}")
                    continue
        if not hold_times or not latency_times:
            return None

        features = {
            'UserKey': user_key,
            'mean_hold': pd.Series(hold_times).mean(),
            'std_hold': pd.Series(hold_times).std(),
            'median_hold': pd.Series(hold_times).median(),
            'mean_latency': pd.Series(latency_times).mean(),
            'std_latency': pd.Series(latency_times).std(),
            'median_latency': pd.Series(latency_times).median(),
            'mean_flight': pd.Series(flight_times).mean() if flight_times else 0,
            'std_flight': pd.Series(flight_times).std() if flight_times else 0,
            'keystroke_count': len(hold_times)
        }

        return features

    all_user_features = []
    all_users = user_df['UserKey'].tolist()

    for i, user_key in enumerate(all_users):
        user_features = create_features_for_user(user_key, data_directory)
        if user_features:
            all_user_features.append(user_features)

    features_df = pd.DataFrame(all_user_features)
    return features_df
```

Figure 2. Code Example

In this code snippet, we have a function that parses through every keystroke data file for each user. Using Pandas, we created an individual dataframe containing all the keystroke data in each user file. We also converted the values within the data from strings into numeric data types. We utilized Pandas to convert our hold time, latency, and flight time lists into Series. From there, the program then condenses the rows of keystroke data down into one row by calculating the mean and standard deviation for the hold time, latency, and flight time. This information, now stored easily into one row separated by user IDs, is then added to a larger dataframe containing this data for all users. Additionally, we dropped any empty rows from our dataframe. This code runs when the model is first created to prepare the training data for processing.

After cleaning our data, we moved onto our model. Our model is a Logistic Regression Model, the most accurate out of the several models we tested. The model, the foundation of the project, predicts whether or not a patient has Parkinson's Disease based on the features we set from the dataframe.

When making our model, we started with defining the features from the keystroke data that would help determine whether or not a patient has Parkinson's. We altered some features like gender into binary categories to better help our model process the data. We then split our data—80% used for training and 20% used for testing—to allow some of it to be used to develop the model. After we split our data, we used an imputer to fill empty cells with median values and a scaler to increase the volume of our data, allowing for our model to have more data to train from and improving its accuracy. To find the best model, we tested four different types of models: Logistic Regression, Random Forest Classifier, Support Vector Machine, and Gradient Boosting Classifier. We analyzed the accuracy and precision of each model and found that Logistic Regression was generally the most accurate.

```python
def run_classification_pipeline(df):
    model_df = df.dropna(subset=['Parkinsons']).copy()
    model_df['Parkinsons'] = model_df['Parkinsons'].astype(int)

    categorical_features = ['Gender', 'Sided']
    model_df = pd.get_dummies(model_df, columns=categorical_features, drop_first=True)

    features_to_use = ['mean_hold', 'std_hold', 'mean_latency', 'std_latency', 'Age',
                       'mean_flight', 'std_flight', 'median_hold',
                       'Gender_Male', 'Gender_Female', 'Sided_Left', 'Sided_Right', 'Impact', 'keystroke_count']

    features_to_use = [f for f in features_to_use if f in model_df.columns]


    X = model_df[features_to_use]
    y = model_df['Parkinsons']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

    imputer = SimpleImputer(strategy='median')
    X_train = imputer.fit_transform(X_train)
    X_test = imputer.transform(X_test)

    scalar = StandardScaler()
    X_train_scaled = scalar.fit_transform(X_train)
    X_test_scaled = scalar.transform(X_test)

    models = {
        "Logistic Regression": LogisticRegression(class_weight='balanced', random_state=42, max_iter=1000),
        "Random Forest": RandomForestClassifier(class_weight='balanced', random_state=42, n_estimators=100),
        "Support Vector Machine": SVC(class_weight='balanced', random_state=42),
        "Gradient Boosting": GradientBoostingClassifier(random_state=42)
    }

    results_list = []

    for name, model in models.items():
        model.fit(X_train_scaled, y_train)
        predictions = model.predict(X_test_scaled)
        accuracy = accuracy_score(y_test, predictions)
        report = classification_report(y_test, predictions, output_dict=True)

        joblib.dump(model, f"{name}.joblib")

        results_list.append({
            'Model': name,
            'Accuracy': accuracy,
            'Healthy_Recall (0)': report['0']['recall'],
            'PD_Recall (1)': report['1']['recall'],
            'Healthy_Precision (0)': report['0']['precision'],
            'PD_Precision (1)': report['1']['precision'],
        })

    results_df = pd.DataFrame(results_list)
    results_df = results_df.sort_values(by='Healthy_Recall (0)', ascending=False)
    return results_df
```

Figure 3. Code Example

Our third component was our Streamlit Python website which allows users to access our model and run the prediction on their own keystroke data through an easy-to-use interface, allowing users to access our model from anywhere. Users are able to receive immediate feedback from our model once entering their data. In this code, we collect the user's data through a form. We utilized Streamlit's built-in functions for creating buttons and selectable options that are easy to navigate for users. Once we collect the data and the form is submitted, we convert it into usable forms for our model. For example, we convert the severity feature which is initially inputted as a string into numerical values and convert the user's gender into a binary response. Then, we filter our features to align with the features that are used in our model and convert this data into a dataframe. This dataframe is subsequently processed through our model which returns a prediction of whether or not the user has Parkinson's Disease and a probability that shows the accuracy of our model to highlight that the model's output is only a machine learning prediction and not medical advice.

```
with st.form(key="parkinsons_form"):
    gender = st.radio(
        "Gender", ('Male', 'Female'),
        horizontal=True,
        help="Pick a gender"
    )
    age = st.number_input('Age', 0, 100)
    mean_hold = st.number_input('Mean Hold Time', 0.0, help='Average amount of tine each key is held down.')
    std_hold = st.number_input('Standard Deviation of Hold Time', 0.0)
    mean_latency = st.number_input('Mean Latency', 0.0)
    std_latency = st.number_input('Standard Deviation of Latency', 0.0)
    mean_flight = st.number_input('Mean Flight Time', 0.0, help='Average time between key presses.')
    std_flight = st.number_input('Standard Deviation of Flight Time', 0.0)
    median_hold = st.number_input('Median Hold Time', 0.0)


    sided = st.radio(
        "Sided", ("Left", "Right"),
        horizontal=True
    )

    impact = st.radio(
        "Impact", ("Mild", "Medium", "Severe"),
        horizontal=True
    )

    keystroke_count = st.number_input('Keystroke Count', 0)

    is_pressed = st.form_submit_button(
        label="Predict if I have Parkinson's"
    )

if is_pressed:
    impact_map = {'Mild': 1.0, 'Medium': 2.0, 'Severe': 3.0}
    input_data = {
        'mean_hold': mean_hold,
        'std_hold': std_hold,
        'mean_latency': mean_latency,
        'std_latency': std_latency,
        'Age': age,
        'mean_flight': mean_flight,
        'std_flight': std_flight,
        'median_hold': median_hold,
        'Gender_Male': 1 if gender == 'Male' else 0,
        'Sided_Left': 1 if sided == 'Left' else 0,
        'Impact': impact_map[impact],
        'keystroke_count': keystroke_count
    }

    features_to_use = ['mean_hold', 'std_hold', 'mean_latency', 'std_latency', 'Age',
                       'mean_flight', 'std_flight', 'median_hold',
                       'Gender_Male', 'Sided_Left', 'Impact', 'keystroke_count']

    df = pd.DataFrame([input_data])
    df = df[features_to_use]

    prediction, probability = predict(df, model)

    prob_parkinsons = probability[1]

    if prediction == 1:
        st.error('Prediction: You might have Parkinson\'s Disease.' , icon="😟")
    else:
        st.success('Prediction: You do not have Parkinson\'s Disease.', icon='😊')

    st.progress(prob_parkinsons)
    st.info("Disclaimer: This is a machine learning prediction and not a substitute for professional medical advice.")
```

Figure 4. Code Example

4. EXPERIMENT

4.1. Model

Since we are using a simple classifier and all of our training data is from one data source, we have not been able to test the accuracy of our model for patients with Parkinson's with data beyond the training dataset.For this experiment, we began with creating fake data, all realistically simulated as having Parkinson's disease. In total, we generated 200 samples of synthetic data. We created this data by generating random values for each feature in the original dataset that we used to train the model. The random values are based on the mean values of the original data. By doing this, we were able to test our models on data beyond our initial training dataset. Using this simulated data, we were able to find the accuracy of our different models for a dataset with patients that all have Parkinson's disease.

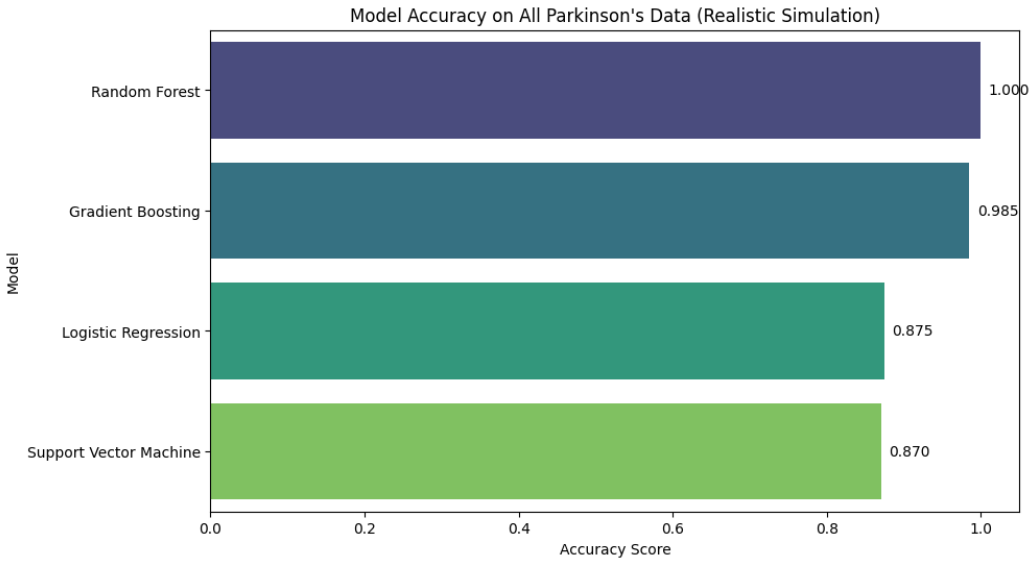

Figure 5. Data Table

From the experiment, we were able to get the accuracy of each model when all patients had Parkinson's in the dataset. The mean of the models' accuracy was 0.9325 and the median was 0.93. Our Random Forest Classifier had the highest accuracy of 1.000 while our Support Vector Machine had the lowest accuracy of 0.870. The success of the Random Forest Classifier was surprising because in the initial tests between models, Random Forest did not have the highest overall accuracy. This could suggest that our Random Forest Classifier has high success rates in identifying individuals with Parkinson's disease, however, struggles to accurately identify individuals that do not have Parkinson's disease. Additionally, our Logistic Regression model fell below our expectations since it was our highest ranking in overall accuracy during our initial tests. Other models may be more successful in classifying purely positive cases or negative cases, while Logistic Regression does well in both.

4.2. A Potential Blind Spot

Similarly, since we have not tested the accuracy of the model with outside data, we are unable to test the model on external data sources for people both with and without Parkinson's. For our second experiment, we again created a synthetic dataset of 200 samples to simulate a 50/50 split of patients with Parkinson's disease and patients without Parkinson's disease. When generating our data, we utilized an imputer and scaler to create new patient data with the values given based on the original dataset. We then ran this data through several different models to test for the models' accuracies when identifying Parkinson's in a mixed dataset. We used the Python Matplotlib library to then graph the models' results to help visualize the outcomes.

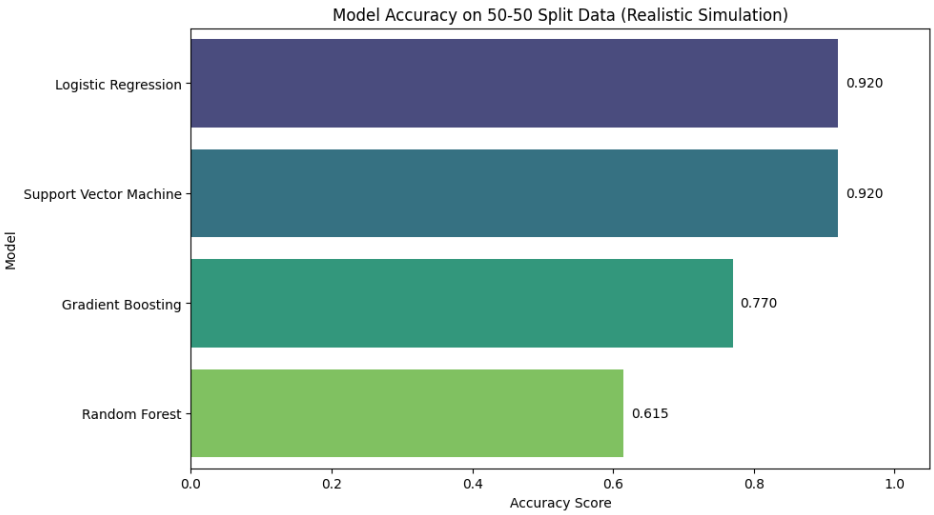

Figure 6. Data Table

From our second experiment, we were able to graph the accuracies of the different models when they were tasked with classifying a dataset with a 50/50 mix of patients with Parkinson's disease and patients without Parkinson's disease. Overall, the mean accuracy of 0.806 and median accuracy of 0.845 were lower than the mean and median accuracies when the dataset consisted purely of patients with Parkinson's disease. Shockingly, our Random Forest Classifier, which was the most accurate model when identifying patients with Parkinson's, was the poorest performing model with the mixed dataset while the previously least accurate model Logistic Regression was the highest performing model alongside Support Vector Machine. This could suggest that in realistic settings where there is more variance in the data, Logistic Regression is a better choice. These experiments also show that our Support Vector Machine is much closer in accuracy to our Logistic Regression model than we had initially inferred.

5. METHODOLOGY COMPARISON

Using mPower: Mobile Parkinson Disease Study, an iOS app created by Sage Bionetworks, scientists collected samples of ten second voice samples from participants which were then processed by a machine learning model to predict if the participant has Parkinson's Disease [1]. Their model utilizes Mel-Frequency Cepstral Coefficients (MFCC) for feature extraction and a Support Vector Machine classifier which was able to achieve a 99% diagnosis accuracy. While our method used keystroke data, theirs used audio data. However, in the end, in both methods we trained machine learning models to use these features to create a prediction for Parkinson's diagnosis. Additionally, since other diseases like multiple system atrophy (MSA) can also cause abnormal speech patterns, there is a risk of misdiagnosis [4].

Much like the previous study, this paper also outlines the use of machine learning for Parkinson's Disease prediction. Since there is currently no objective test for the detection of Parkinson's Disease, machine learning offers an opportunity for early detection through prediction models. This team analyzed MRI images from DaTscans and achieved an AUC of 0.996 with Neural Networks [8]. Although this method and our method both utilize machine learning, requiring MRI images and relying on DaTscans which limit accessibility and relies on stable access to healthcare facilities [15]. Our solution, which only requires a keyboard, can be done anywhere, granting greater access especially to vulnerable populations like the elderly.

Similar to the last two, this study also utilizes machine learning for Parkinson's prediction. It differs, however, in its method. While the other papers used audio snippets and MRI images, this team focused on the increasingly studied relationship between the brain and gut, analyzing gut microbiota instead. One of the early symptoms of Parkinson's is constipation due to a disruption in gut health [11]. Braak's hypothesis theorizes that non-familial forms of Parkinson's disease begin as a pathogen in the gut [7]. Through this method of analysis, they achieved a mean accuracy of 0.85 and an AUC of 0.92 [13]. Like the prior study, the main limitation of this method is its accessibility. Extracting gut microbiota data can be more invasive than other methods and require access to medical facilities. In comparison, our method using keystroke data offers a noninvasive alternative.

Through our two experiments, we tested our model for possible errors in predicting Parkinson's disease. For our first experiment, we generated 200 samples of simulated data of all patients that are diagnosed with Parkinson's disease to then allow us to test how well our model performs when the sample consists of all positive cases. Shockingly, we found that in this scenario, our Random Forest classifier and Gradient Boosting model perform the best which contrasts our accuracy score that ranks Logistic Regression as the best. However, in our second experiment, we again generated 200 samples. However, this time the samples consisted of a 50/50 split of positive and negative cases. When tested with this new mixed data, Logistic Regression and Support Vector Machine were more accurate. Our experiments suggests that while certain models like Random Forest may be accurate in specific scenarios, Logistic Regression is more balanced and reliable when working with mixed data.

6. CONCLUSIONS

Although our project currently offers an accurate, affordable method of early Parkinson's detection, there are several improvements we can implement to better our solution. One limitation we ran into was a limited amount of training data. We only had a single dataset resulting in 647 users. Since we were using an external dataset we could not recreate the experiment to gather more data than the amount given. This impacted our machine learning model's accuracy and precision in predicting patients with Parkison's Disease. With more resources, we could recreate the study and gather more data. Another way we can remedy this issue is reaching out to more patients with Parkinson's to increase our amount of data and give us more up to date information about Parkinson's than our current data from 2019.

Our projects demonstrate how keystroke data can provide an alternative, less-invasive, more cost-effective, and affordable option to Parkinson's disease detection. While further data needs to be collected for our models, our results show potential to help early detection that will benefit patients worldwide through convenience and accessibility.

The first method implemented an application that used short audio recordings to predict if the user has Parkinson's disease through a machine learning model. However, for people without access to a mobile device, a problem even more common in elderly populations, it may be difficult to access this tool. Our project only requires a computer, a device that is more commonly accessible in public spaces like libraries.

The second method utilizes MRI images from DaTscans alongside a neural network to perform these predictions. While this method has a high accuracy, it is a costly and inaccessible process especially for elderly patients that do not have convenient access to healthcare centers. By only requiring a keyboard, our project attempts to make this process more accessible.

The last method analyzes gut microbiota as an indicator of Parkinson's. This method, however, can be overly invasive and discouraging to elderly patients that do not want to endure intensive testing. Our method using keystroke data is noninvasive and makes testing for Parkinson's a less daunting and more convenient task.

This project builds on previous methodologies in sports performance analysis, using computer vision and deep learning to provide precise, personalized feedback for basketball shot analysis.

In the first methodology, prior research used computer vision and deep learning to analyze sports performance by tracking key body movements. Although effective, this approach struggled with real-time analysis and generalization across diverse users. [16] My project advances this by incorporating refined feedback based on individual movement patterns, enhancing personalization.The second methodology employed pose estimation via convolutional neural networks (CNNs) to track athlete movements. However, issues with image quality reduced tracking accuracy. My project addresses this limitation by introducing image pre-processing techniques, ensuring reliable pose detection even with low-quality inputs.

REFERENCES

[1] Chahine, L. M., Stern, M. B., & Chen-Plotkin, A. (2020). Blood-based biomarkers for Parkinson's disease. Movement Disorders, 35(6), 883–891.
[2] Fearnley, J. M., & Lees, A. J. (1991). Ageing and Parkinson's disease: Substantia nigra regional selectivity. Brain, 114(5), 2283–2301. https://pmc.ncbi.nlm.nih.gov/articles/PMC2536542/
[3] Gandhi, K. R., &Saadabadi, A. (2024). Parkinson disease. In StatPearls. StatPearls Publishing.
[4] Kaur, A., Singh, H., & Sharma, R. (2024). A multimodal hybrid deep learning approach for Parkinson's disease diagnosis. Frontiers in Artificial Intelligence, 7, 11735538.
[5] Keshavarzian, A., & Shannon, K. M. (2024). Parkinson disease: Pathophysiology. In StatPearls. StatPearls Publishing.
[6] National Institute of Neurological Disorders and Stroke. (n.d.). Parkinson's disease. U.S. Department of Health and Human Services.
[7] Park, H., Kim, J., & Cho, S. (2022). Molecular signatures of Parkinson's disease progression revealed by single-cell transcriptomics. Nature Communications, 13, 7356.
[8] Patel, S., Gupta, N., & Kumar, V. (2024). Deep learning-based feature extraction and machine learning models for Parkinson's disease prediction using DaTSCAN image. Biomedical and Pharmacology Journal, 18(Special Edition March).
[9] Prashanth, R., Roy, S. D., Mandal, P. K., & Ghosh, S. (2017). Automatic classification and prediction models for early Parkinson's disease diagnosis using Parkinson's Progression Markers Initiative (PPMI) data. PLOS ONE, 12(2), e0188226.
[10] Rizzo, G., Copetti, M., Arcuti, S., Martino, D., Fontana, A., &Logroscino, G. (2016). Accuracy of clinical diagnosis of Parkinson disease: A systematic review and meta-analysis. Neurology, 86(6), 566–576.
[11] Sarasso, E., Agosta, F., &Filippi, M. (2022). Neuroimaging in Parkinson's disease: Recent advances and perspectives. Frontiers in Neurology, 13, 9263276.
[12] Tian, J., Liu, Y., Zhang, X., & Li, C. (2024). Machine learning–based classification of Parkinson's disease using multi-modal neuroimaging data. Frontiers in Neuroscience, 18, 11970477.
[13] Wang, Y., Tian, X., Zhang, T., Wang, H., Zhang, Z., & Zhang, Y. (2025). Multimodal imaging biomarkers for Parkinson's disease progression: A systematic review and meta-analysis. PLOS ONE, 20(1), e0310005.
[14] Wei, J., Zhang, T., & Chen, L. (2025). Alpha-synuclein seeding mechanisms in Parkinson's disease. npj Parkinson's Disease, 11(1), 25.
[15] Zhao, Y., Zhang, L., & Chen, W. (2024). Multi-sequence MRI-based deep learning framework for Parkinson's disease diagnosis. BMC Medical Imaging, 24, 1335.