Model Selection

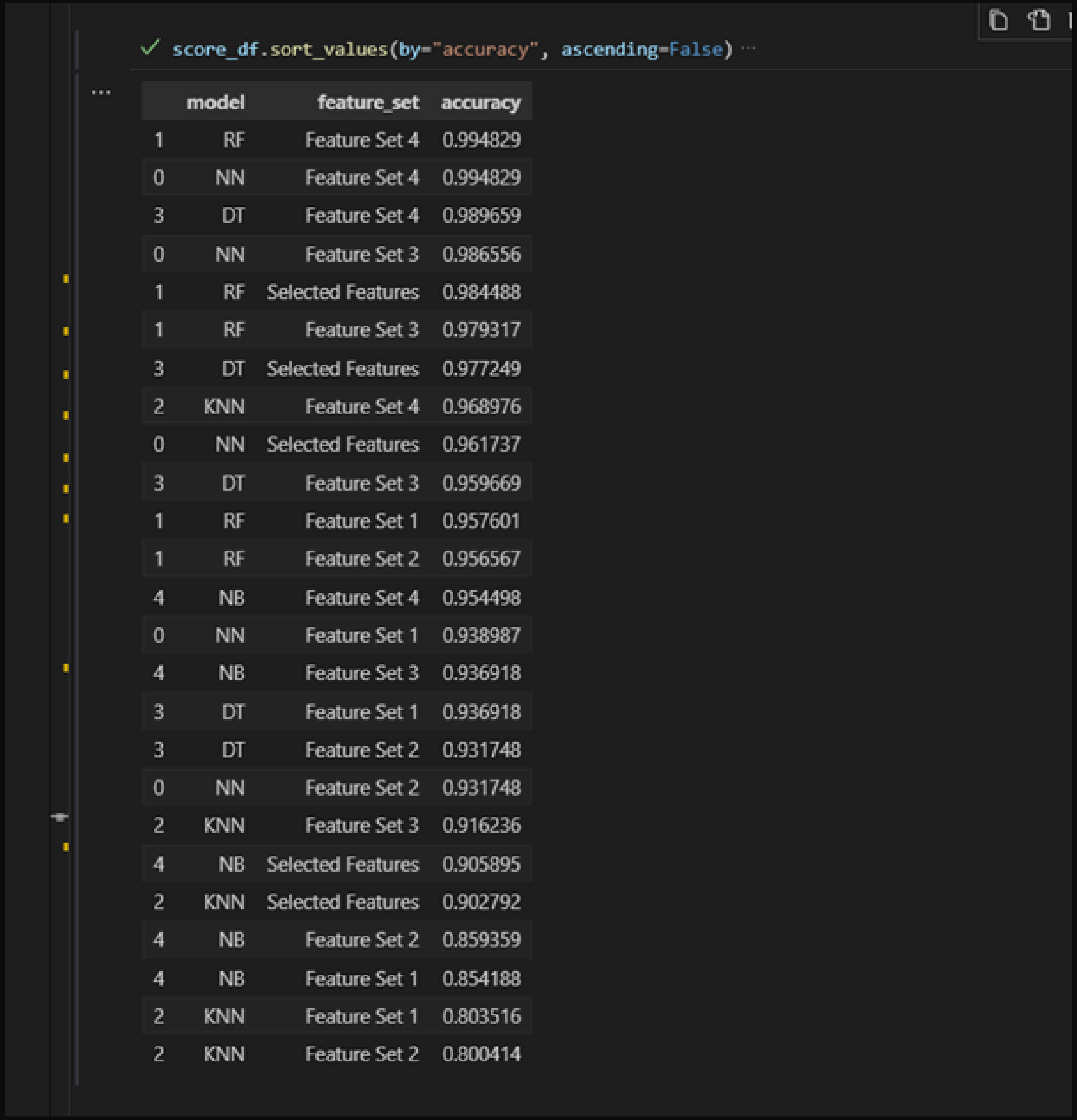
Model training with different set of features

Here's the revised content with Random Forest instead of k-NN:
To find the best-performing model, I conducted a thorough model selection process. I tested various classification models, including:

- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Feedforward Neural Networks

In addition to this, I evaluated each model using five different feature sets. After careful analysis, I found that Feature Set 4 provided the best results. The best-performing models were Random Forest and a Feedforward Neural Network. These models offered the highest accuracy and performance when used with the optimal feature set.

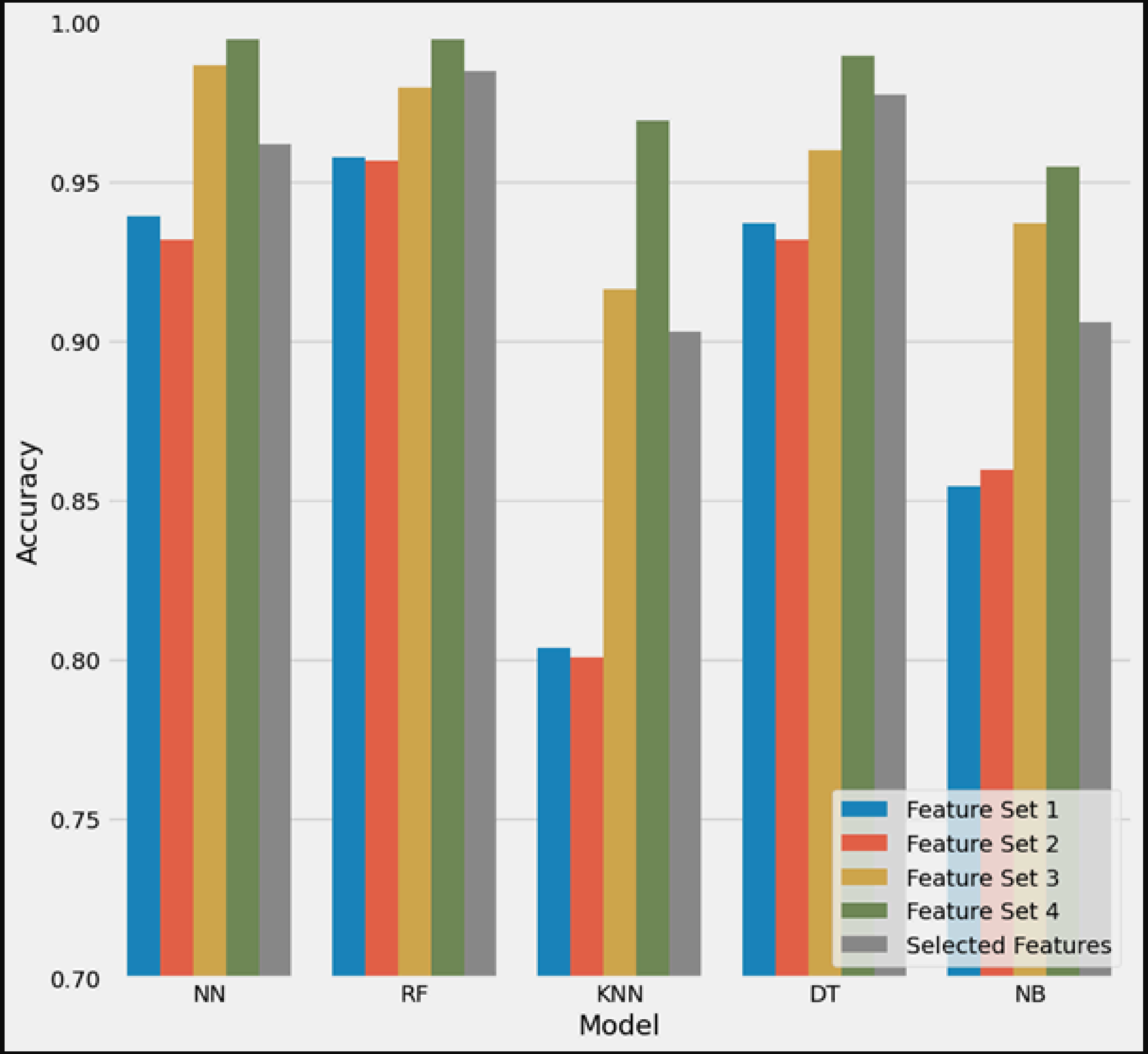
Best models accuracy : 99.48%



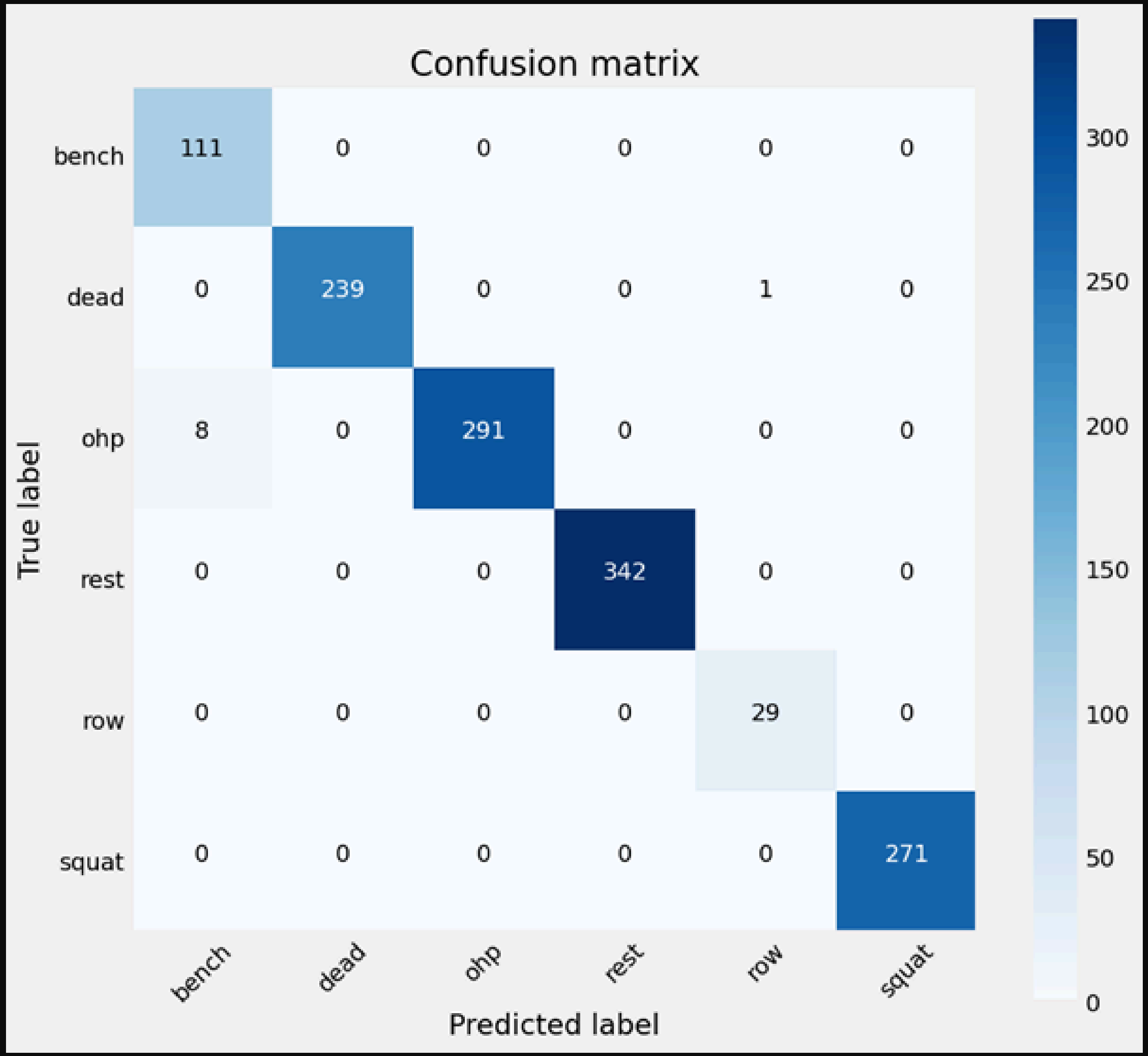
```
score_df.sort_values(by="accuracy", ascending=False)
```

	model	feature_set	accuracy
1	RF	Feature Set 4	0.994829
0	NN	Feature Set 4	0.994829
3	DT	Feature Set 4	0.989659
0	NN	Feature Set 3	0.986556
1	RF	Selected Features	0.984488
1	RF	Feature Set 3	0.979317
3	DT	Selected Features	0.977249
2	KNN	Feature Set 4	0.968976
0	NN	Selected Features	0.961737
3	DT	Feature Set 3	0.959669
1	RF	Feature Set 1	0.957601
1	RF	Feature Set 2	0.956567
4	NB	Feature Set 4	0.954498
0	NN	Feature Set 1	0.938987
4	NB	Feature Set 3	0.936918
3	DT	Feature Set 1	0.936918
3	DT	Feature Set 2	0.931748
0	NN	Feature Set 2	0.931748
2	KNN	Feature Set 3	0.916236
4	NB	Selected Features	0.905895
2	KNN	Selected Features	0.902792
4	NB	Feature Set 2	0.859359
4	NB	Feature Set 1	0.854188
2	KNN	Feature Set 1	0.803516
2	KNN	Feature Set 2	0.800414

All Models performances

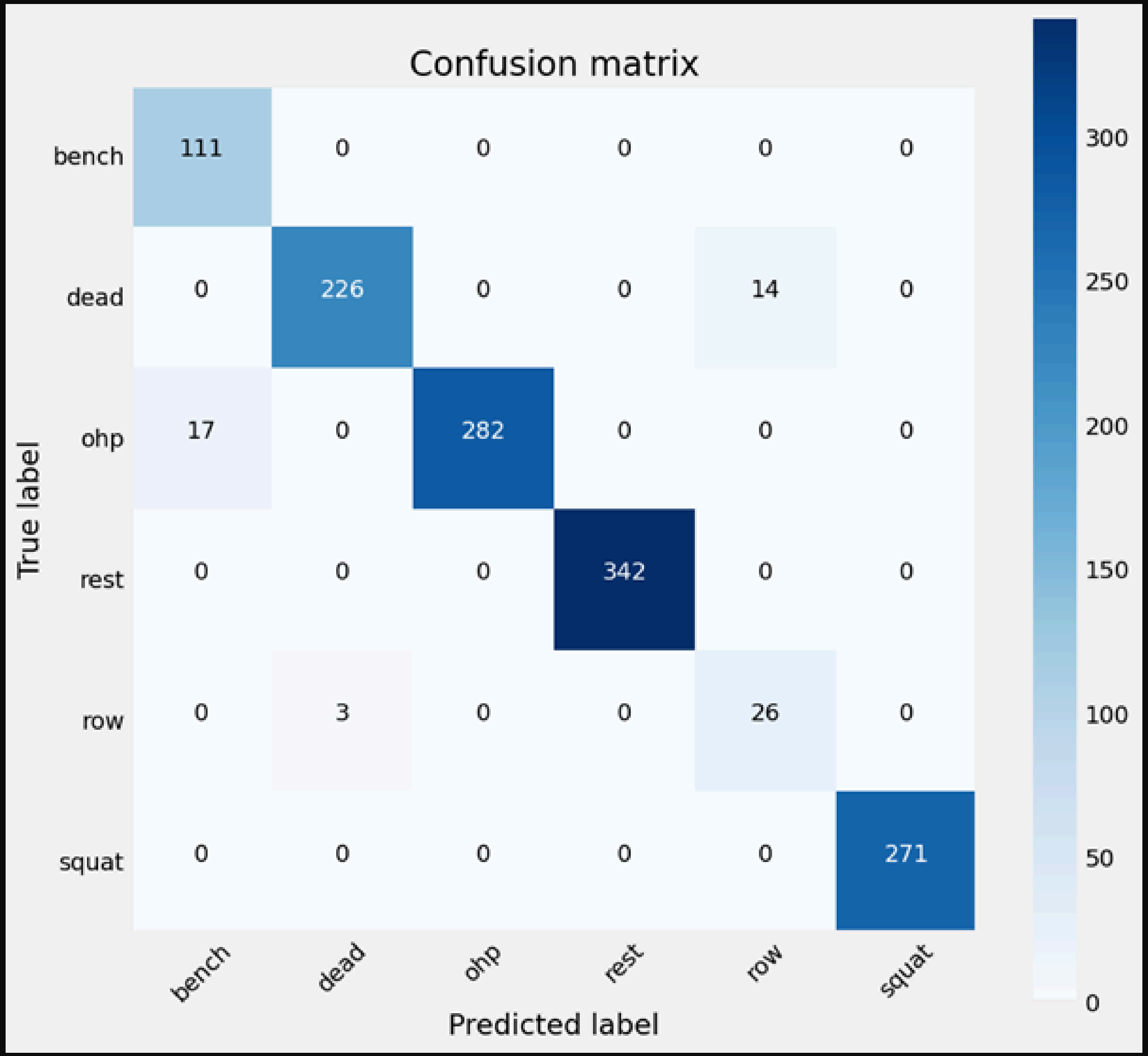


Performance visualization with True and Predicted classifications



	bench	dead	ohp	rest	row	squat
bench	111	0	0	0	0	0
dead	0	239	0	0	1	0
ohp	8	0	291	0	0	0
rest	0	0	0	342	0	0
row	0	0	0	0	29	0
squat	0	0	0	0	0	271

Random Forest

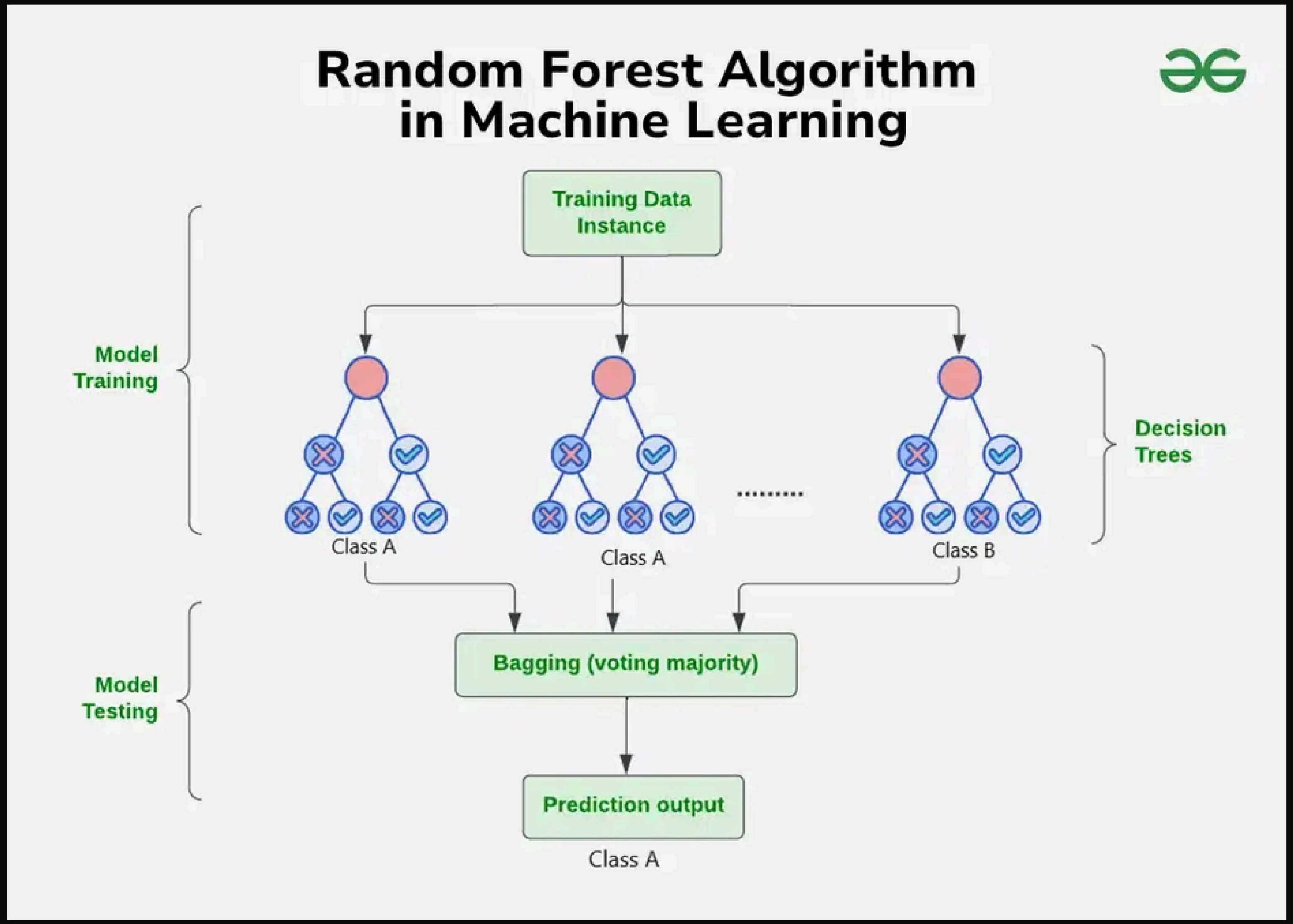


	bench	dead	ohp	rest	row	squat
bench	111	0	0	0	0	0
dead	0	226	0	0	14	0
ohp	17	0	282	0	0	0
rest	0	0	0	342	0	0
row	0	3	0	0	26	0
squat	0	0	0	0	0	271

Feed Forward Neural Network

Observations

- Through the model selection process, I found that the Random Forest model, while simpler in its approach compared to a Neural Network (NN), outperformed not only the NN but also other models tested. Random Forest's simplicity comes from its design: it is an ensemble of decision trees, each making predictions independently. The final prediction is based on the majority vote or average from the trees. This straightforward ensemble approach allows Random Forest to reduce overfitting and handle noisy data, offering a balance of simplicity and robustness.
- Feature Set 4, with its high dimensionality and the most features among all sets, provided the Random Forest model with diverse and detailed data. The model's ability to handle a large number of features, combined with its built-in feature selection mechanism, allowed it to focus on the most relevant features and ignore irrelevant ones. This is one of the key reasons why it performed better than the NN model and other algorithms like k-NN, Support Vector Machines, or Decision Trees.
- By averaging the predictions of multiple decision trees, Random Forest achieved a high degree of stability and accuracy. Its capability to handle complex datasets with many features in a relatively simple way made it the best-performing model, even surpassing the NN model, which is typically more resource-intensive.



Results

- **Model Accuracy : 99.38%**

Counting Repetitions

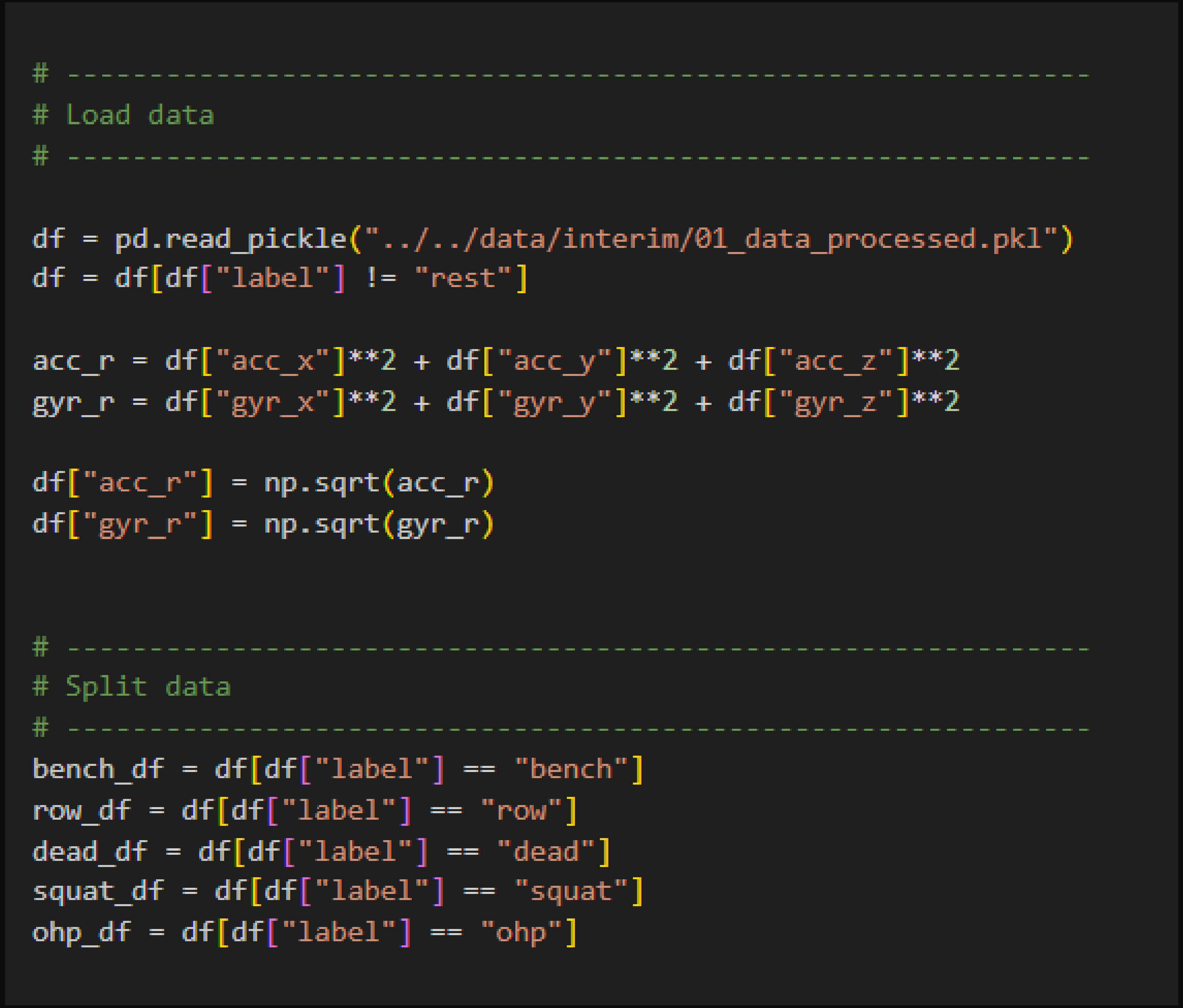
Pre-Processing Step

I processed the acceleration and gyroscope data from the x, y, and z axes for accurate repetition counting.

- The unit vector formula was applied to calculate:
  - acc_r (resultant acceleration)
  - gyr_r (resultant gyroscope data)
- This formula combines the three-axis data into a single magnitude, representing the overall movement more effectively.

Why this approach is better:

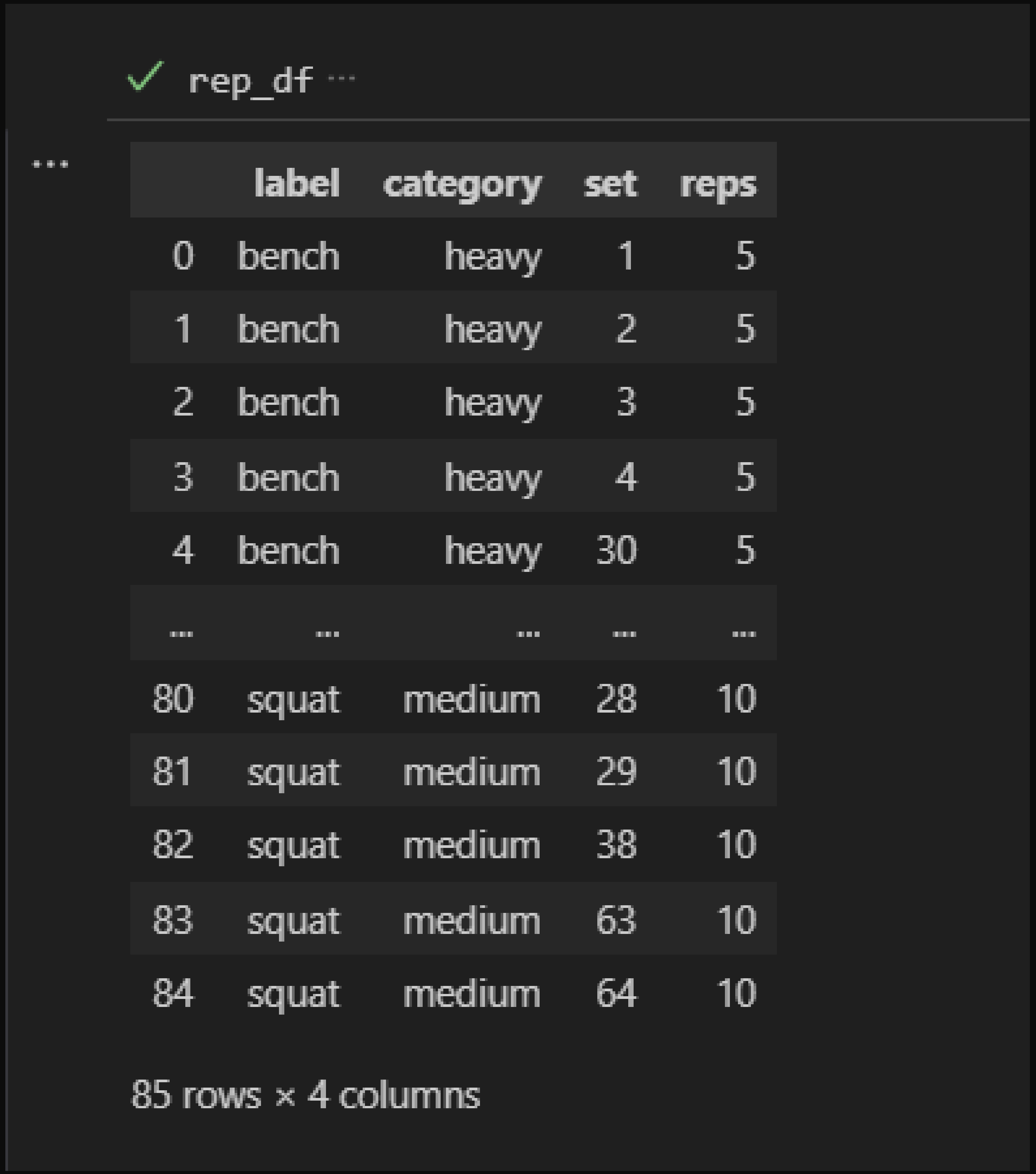
- Normalizes multi-axis data into a single directional measurement.
- Reduces noise caused by inconsistent movements on individual axes.
- Provides a consistent and reliable count by focusing on the overall magnitude of motion, regardless of orientation changes.



```python
# --------------------------------------------------------------
# Load data
# --------------------------------------------------------------

df = pd.read_pickle("../../data/interim/01_data_processed.pkl")
df = df[df["label"] != "rest"]

acc_r = df["acc_x"]**2 + df["acc_y"]**2 + df["acc_z"]**2
gyr_r = df["gyr_x"]**2 + df["gyr_y"]**2 + df["gyr_z"]**2

df["acc_r"] = np.sqrt(acc_r)
df["gyr_r"] = np.sqrt(gyr_r)


# --------------------------------------------------------------
# Split data
# --------------------------------------------------------------
bench_df = df[df["label"] == "bench"]
row_df = df[df["label"] == "row"]
dead_df = df[df["label"] == "dead"]
squat_df = df[df["label"] == "squat"]
ohp_df = df[df["label"] == "ohp"]
```

Organizing Data

- To prevent overlap of different exercises and sets, I arranged all data row-wise and implemented a grouping strategy.
- To ensure accurate plotting and analysis, I used groupby on the following columns:
- Label: Identifies the specific exercise.
- Category: Distinguishes between different types of movements or exercises.
- Set: Groups repetitions within each exercise.
- I also added a "reps" column to categorize the sets into:
- Heavy Sets: 5 reps
- Medium Sets: 10 reps
- These "true rep" columns will be used for comparison with the predicted rep columns later.
- Benefits:
- Organized Data: Helps in distinguishing and analyzing each exercise and set separately.
- Accurate Visualization: Facilitates plotting graphs for different exercises and sets without data overlap.
- Comparison with Predictions: The "reps" column allows for direct comparison between actual and predicted repetitions to assess model accuracy.



```
rep_df
```

	label	category	set	reps
0	bench	heavy	1	5
1	bench	heavy	2	5
2	bench	heavy	3	5
3	bench	heavy	4	5
4	bench	heavy	30	5
...	...	...	...	...
80	squat	medium	28	10
81	squat	medium	29	10
82	squat	medium	38	10
83	squat	medium	63	10
84	squat	medium	64	10

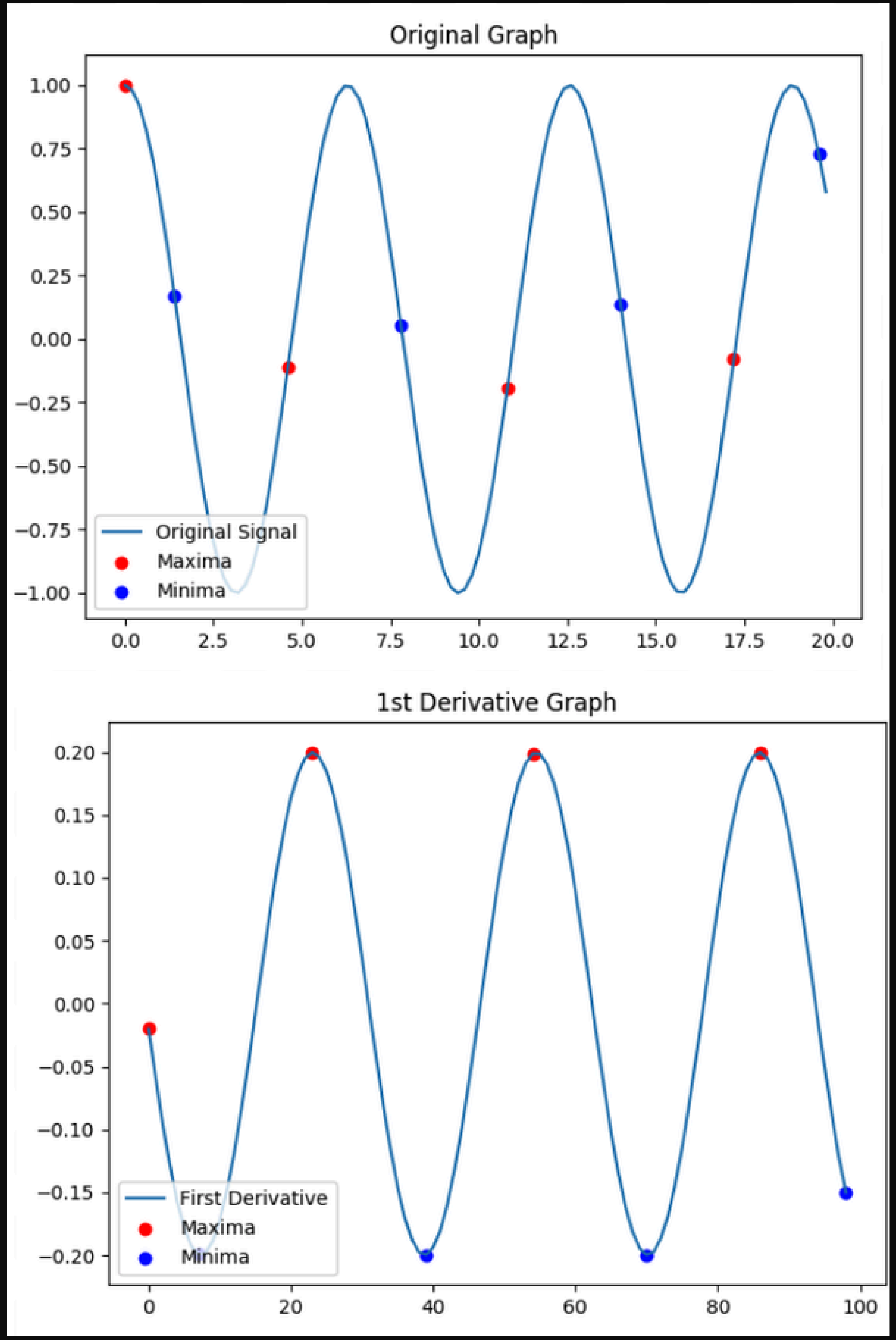
85 rows × 4 columns

Counting Repetitions(argrelextrema method)

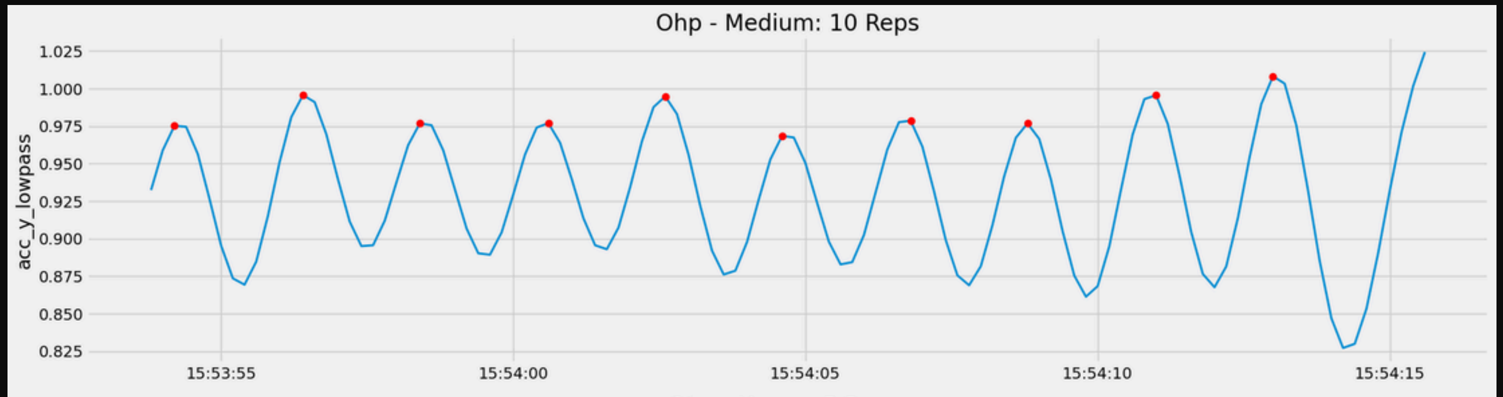
- Using argrelextrema to Identify Extrema for Repetition Counting
- MATHEMATICAL BACKGROUND
- The argrelextrema method is used to identify local extrema (maxima and minima) in a dataset. It operates based on the concept of local neighborhoods:
- Local Maximum: A point xix_ixi in a sequence is a local maximum if it is greater than its neighboring points.
- Local Minimum: A point xix_ixi is a local minimum if it is smaller than its neighboring points.
- Mathematically, if you have a function f(x)f(x)f(x), a point xix_ixi is considered a local maximum if f(xi)>f(xi−1)f(x_i) > f(x_{i-1})f(xi)>f(xi−1) and f(xi)>f(xi+1)f(x_i) > f(x_{i+1})f(xi)>f(xi+1). Similarly, for a local minimum, f(xi)<f(xi−1)f(x_i) < f(x_{i-1})f(xi)<f(xi−1) and f(xi)<f(xi+1)f(x_i) < f(x_{i+1})f(xi)<f(xi+1).

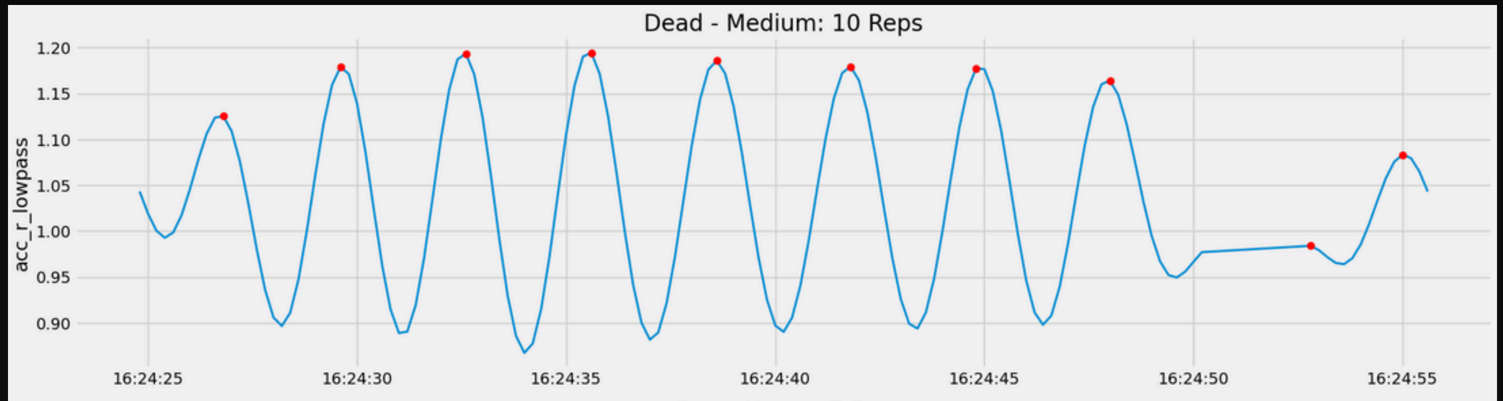
- ARGRELEXTREMA METHOD:
- Function: The argrelextrema function from scipy.signal is used to find relative extrema in a dataset.
- Parameters: It takes the array of data and a comparator function (e.g., np.greater for maxima or np.less for minima).
- Output: It returns the indices of the local extrema in the dataset.
- How It Works
- Data Preparation: The data, which is typically a time-series of measurements or a signal, is prepared for analysis.
- Applying argrelextrema:
- Import the necessary function: from scipy.signal import argrelextrema
- Apply it to the data to find indices of local maxima and/or minima.

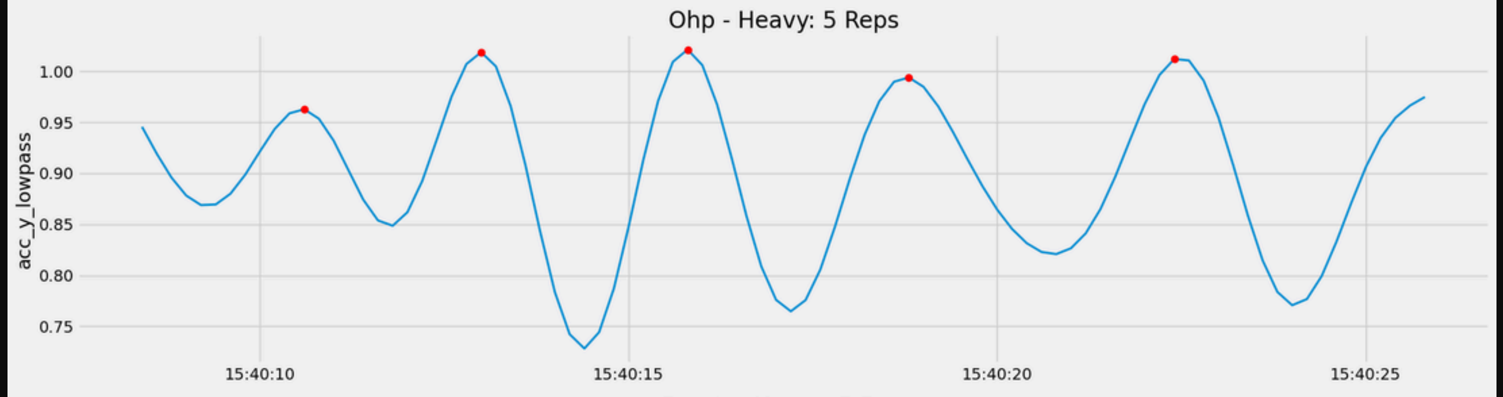
- REPETITION COUNTING:
- Extrema Identification: Use argrelextrema to identify peaks (local maxima) in the signal. Each peak corresponds to a potential repetition in the exercise data.
- Counting Reps: Count the number of identified peaks to determine the number of repetitions.

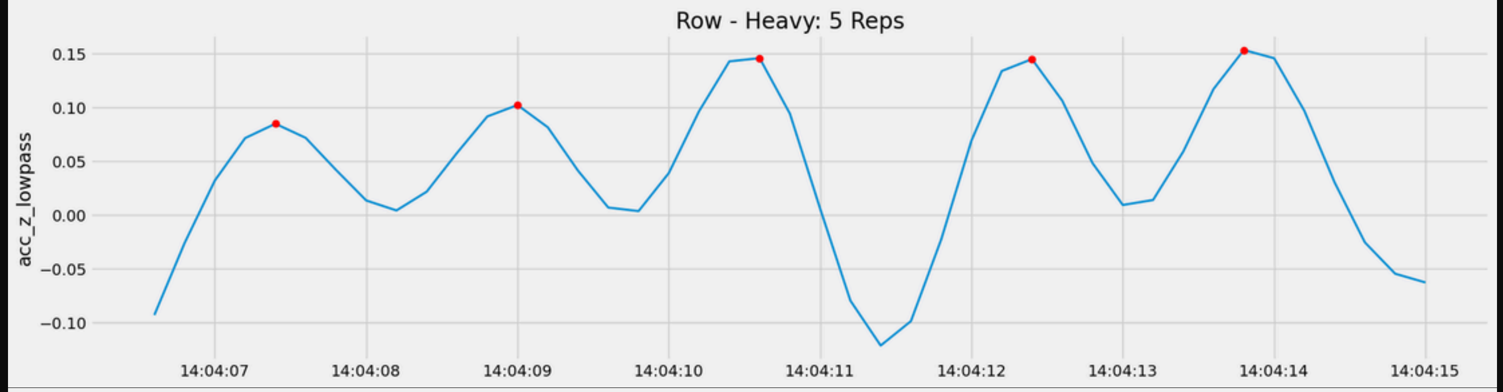


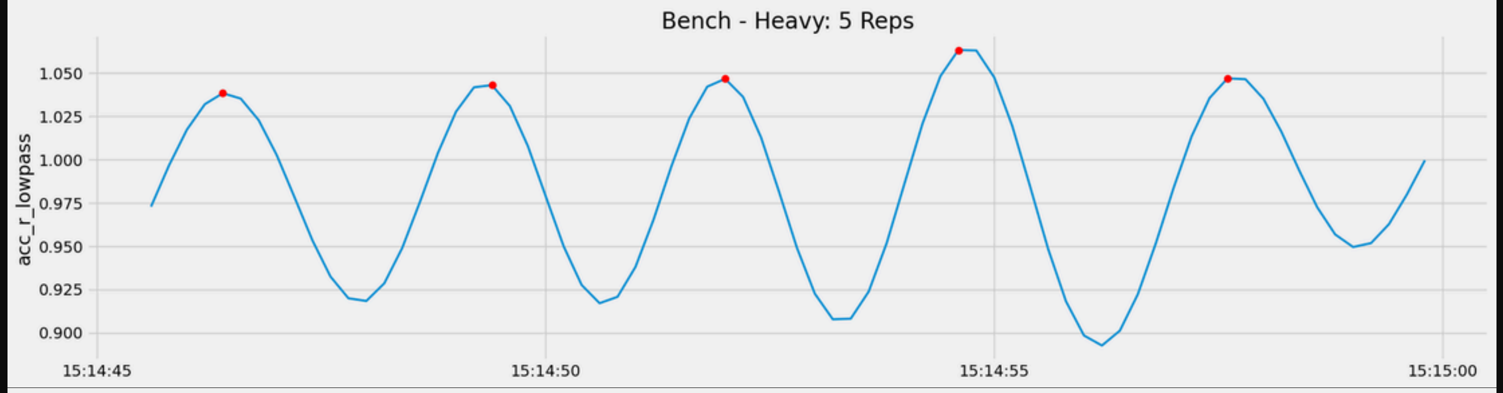
Few Results of using the argrelextrema method

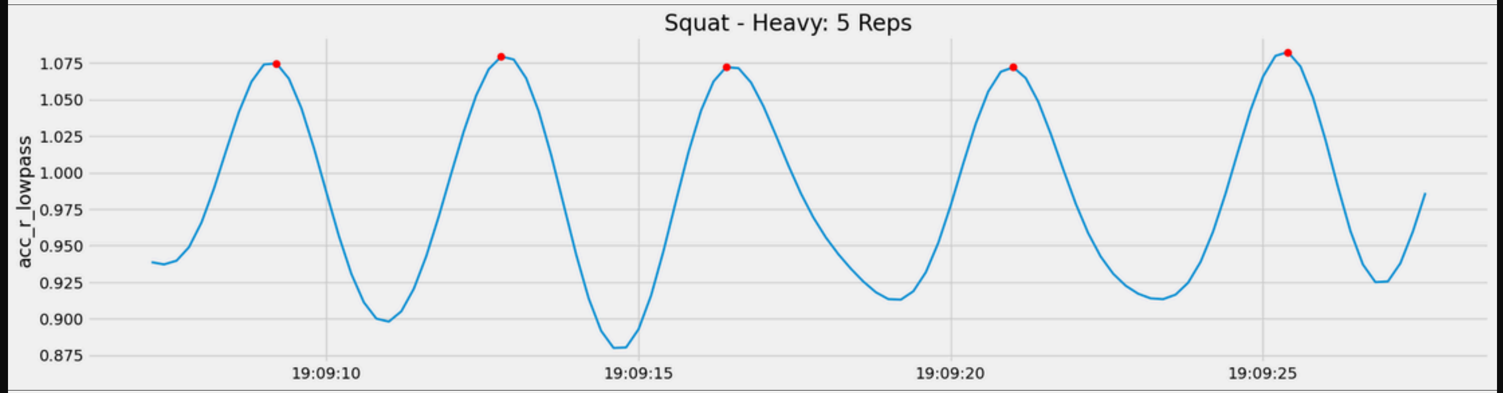


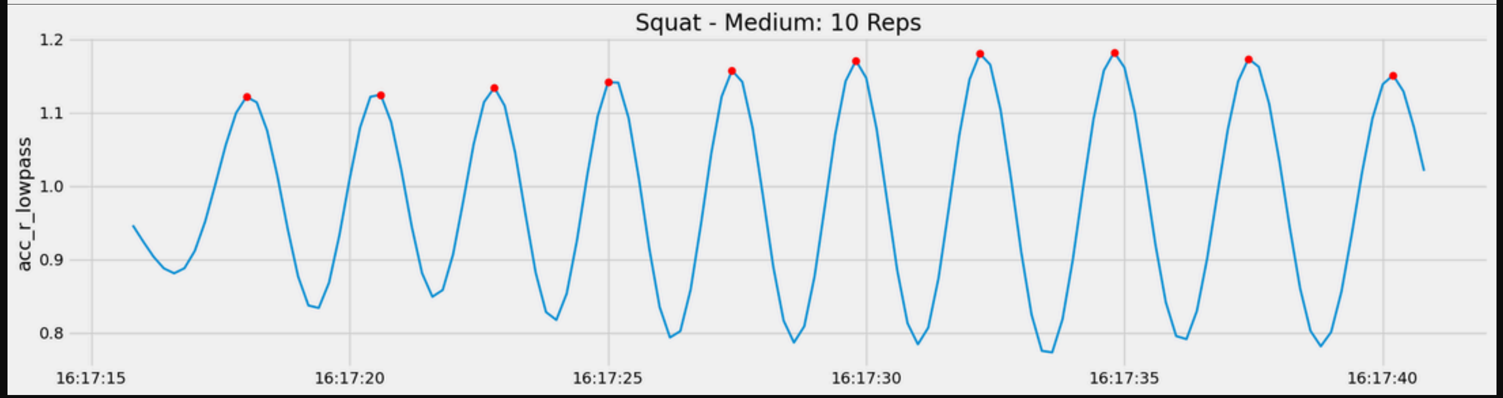


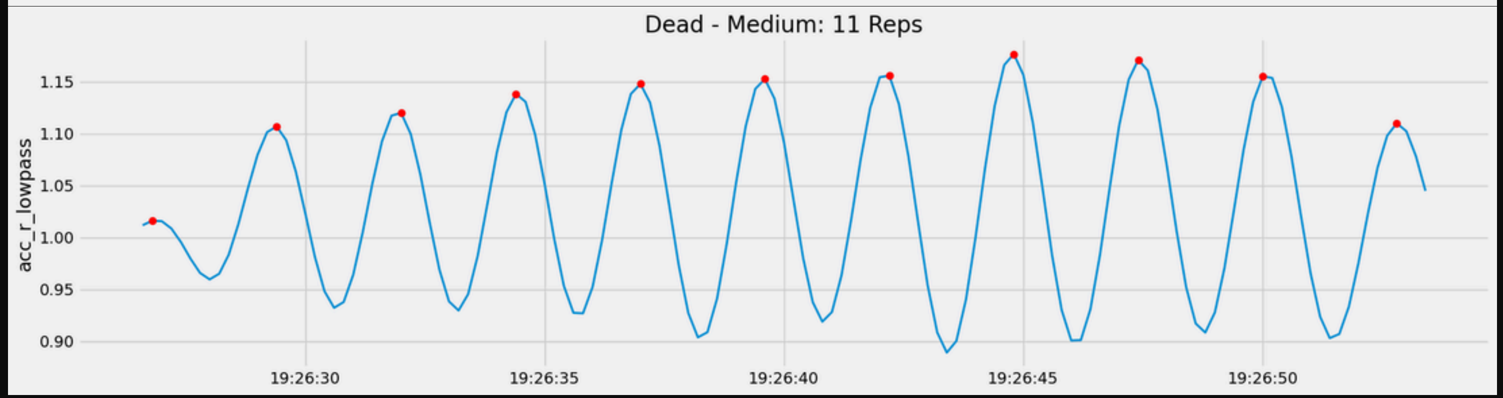




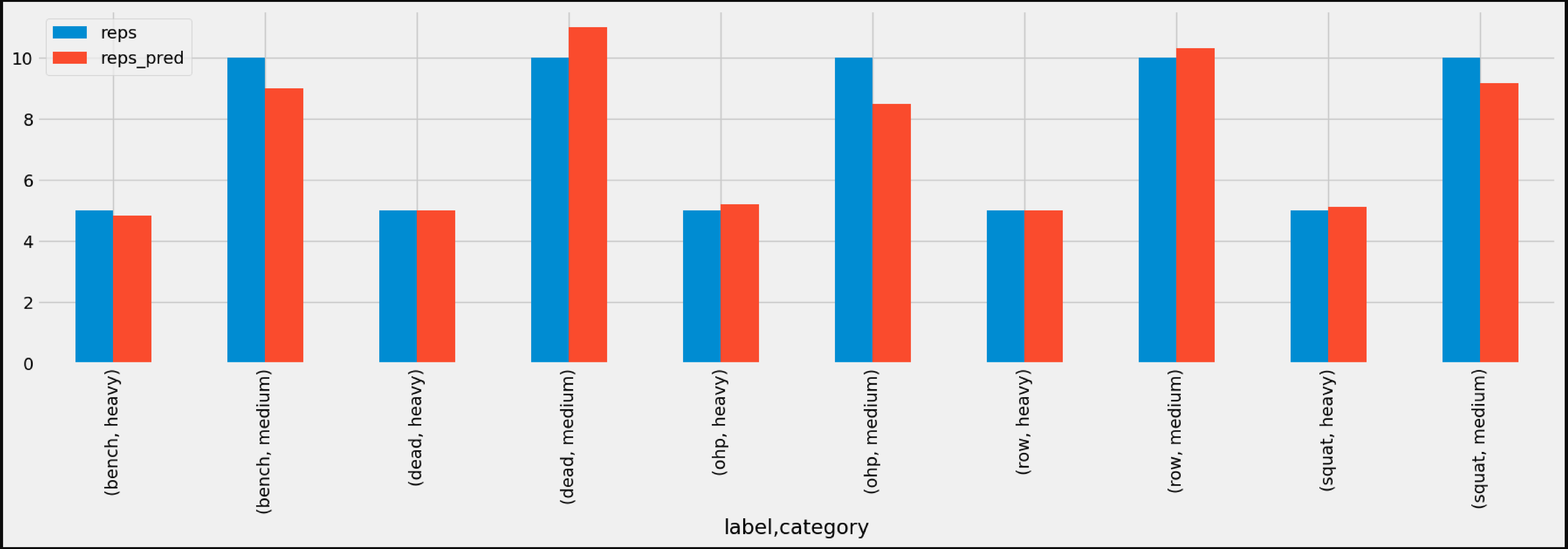








Comparison of true and predicted rep values for each exercise and its category



Observation & Conclusion

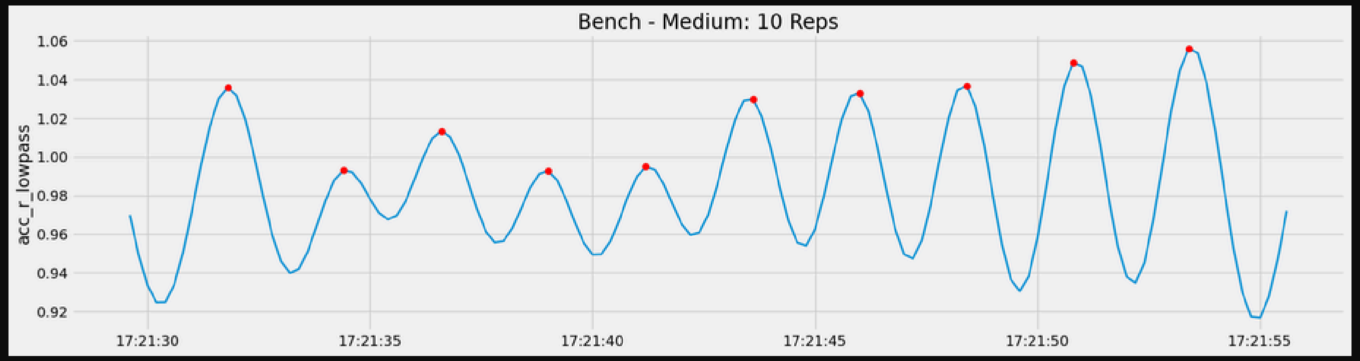
The implementation of the argrelextrema method for counting exercise repetitions yielded promising results. **The mean absolute error (MAE) of 0.64** indicates that the discrepancy between the predicted and actual number of repetitions was relatively small. This suggests that the method was generally effective in identifying the true count of repetitions.

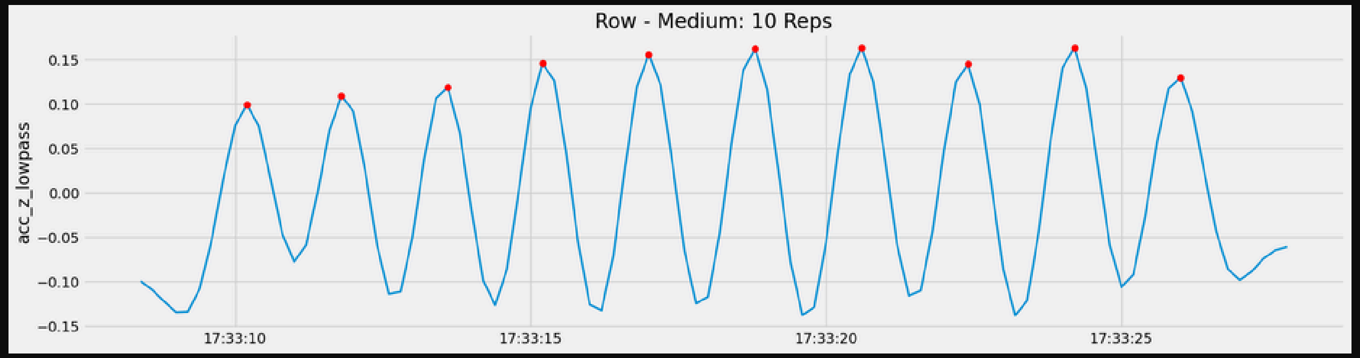
KEY POINTS:

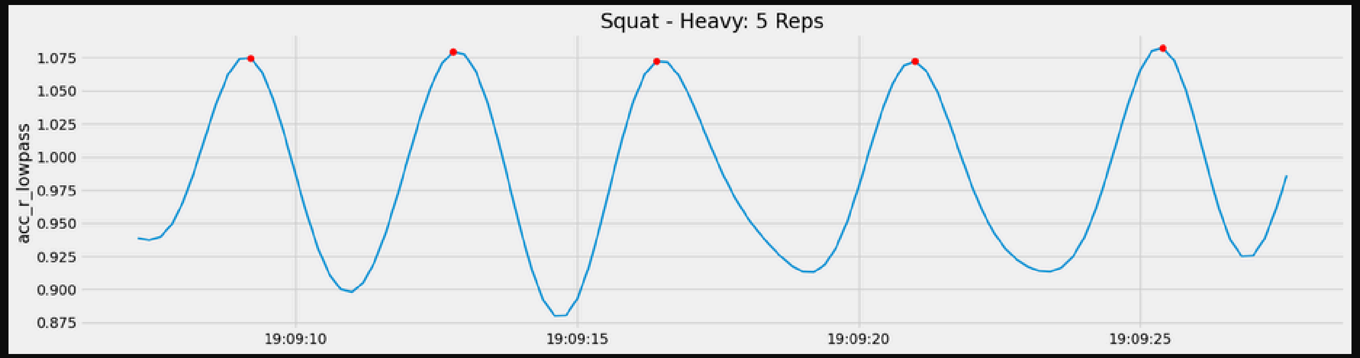
- Accuracy of Repetitions: Most sets of exercise repetitions were accurately counted. The method's ability to detect local maxima (peaks) in the signal data closely matched the actual number of repetitions. This accuracy is crucial for reliable tracking and analysis of exercise performance.
- Error Analysis: The MAE of 0.64 reflects a low average error across the dataset. This level of accuracy suggests that the peak detection algorithm was successful in differentiating between individual repetitions with minimal error. The small MAE implies that any discrepancies between predicted and true counts were minor, which is a positive outcome.
- Effectiveness of argrelextrema: The argrelextrema method effectively identified peaks in the processed acceleration or gyroscope data. By detecting local maxima, the method was able to determine the number of repetitions accurately. The performance of this method demonstrates its suitability for counting discrete events in a time-series signal.
- Areas for Improvement: While the results were generally accurate, there may be specific cases or noisy data where the method could be refined. Further analysis of edge cases and potential sources of error could help in enhancing the accuracy of repetition counting.

CONCLUSION:

- The argrelextrema approach provided a reliable means of counting exercise repetitions with a mean absolute error of 0.64. The results showed that most exercise sets were counted accurately, validating the effectiveness of this method for repetition tracking. The minor discrepancies observed suggest that while the method is robust, there is room for further optimization to ensure even greater precision in all scenario


Bench - Medium: 10 Reps
acc_r_lowpass
17:21:30
17:21:35
17:21:40
17:21:45
17:21:50
17:21:55


Row - Medium: 10 Reps
acc_z_lowpass
17:33:10
17:33:15
17:33:20
17:33:25


Squat - Heavy: 5 Reps
acc_r_lowpass
19:09:10
19:09:15
19:09:20
19:09:25