({'10': 25, '0': 25, '2': 25, '1': 25, '4': 25, '3': 25, '5': 25},

[['Front',

[-0.9751627, 0.8935174, 0.533216834],

[0.5377487, 0.5299966, 0.4485384, -0.478271216],

[[-1.07694376, 0.9483451, 0.570288658],

[-1.07100654, 0.925480962, 0.5061945],

[-1.07303059, 0.9139524, 0.527580142],

[-1.06443262, 0.925752342, 0.565872431],

[-1.0480032, 0.9272774, 0.5889423]]]])

For instance, the first frame for the gesture labeled '0' has the following data:

* Hand Orientation: 'Front'
* Hand Position: [−0.9751627,0.8935174,0.533216834][−0.9751627,0.8935174,0.533216834]
* Hand Rotation: [0.5377487,0.5299966,0.4485384,−0.478271216][0.5377487,0.5299966,0.4485384,−0.478271216]
* Finger Positions: [[−1.07694376,0.9483451,0.570288658],…][[−1.07694376,0.9483451,0.570288658],…]

feature 1: hand orientation normal

feature 2 : hand rotation

feature 3: 3D Euclidean distances of the hand position to each of the fingertips

feature 4 : the distance between one fingertip and the consecutive fingertips

feature 5 : features represent the angles between two adjacent fingers

Normalization is often beneficial in machine learning for several reasons:

1. **Scale**: Features may be on different scales, and some algorithms are sensitive to the magnitude of the input variables.
2. **Speed**: Normalized data often helps the optimization algorithm to converge faster.
3. **Numerical Stability**: Machine learning algorithms use numerical optimization techniques that can become unstable if the features are on different scales.
4. **Distance-based Algorithms**: For algorithms like k-NN, k-means, or SVM, where the concept of distance is used, normalization is crucial.

However, there are cases where normalization may not be necessary:

1. **Tree-based Algorithms**: Algorithms like Decision Trees and Random Forests are generally not affected by the different scales of features.
2. **Domain Knowledge**: Sometimes the scale of a feature is meaningful, and normalizing would remove that information.

In our case, the features are quite varied:

* Hand Orientation is categorical.
* Euler Angles could range typically between -180 and 180 degrees.
* Distances are continuous values that could be small.
* Angles between adjacent fingers would also be in degrees.

**1. k-Nearest Neighbors (k-NN)**

* **Advantages**: Simple, intuitive, good for datasets with complex boundaries.
* **Disadvantages**: Computationally expensive, sensitive to irrelevant features and the scale of the data.
* **Why Chosen**: It's a non-parametric method and can capture complex relationships without a predetermined model structure.

**2. Support Vector Machines (SVM)**

* **Advantages**: Effective in high-dimensional spaces, works well with clear margin of separation.
* **Disadvantages**: Not suitable for large datasets, sensitive to noise.
* **Why Chosen**: Effective for datasets where features can define a clear margin of separation.

**3. Random Forest**

* **Advantages**: Handles higher dimensionality well, maintains accuracy for missing data.
* **Disadvantages**: Complexity, longer training time.
* **Why Chosen**: Ensemble methods like Random Forest generally perform well for a variety of data types.

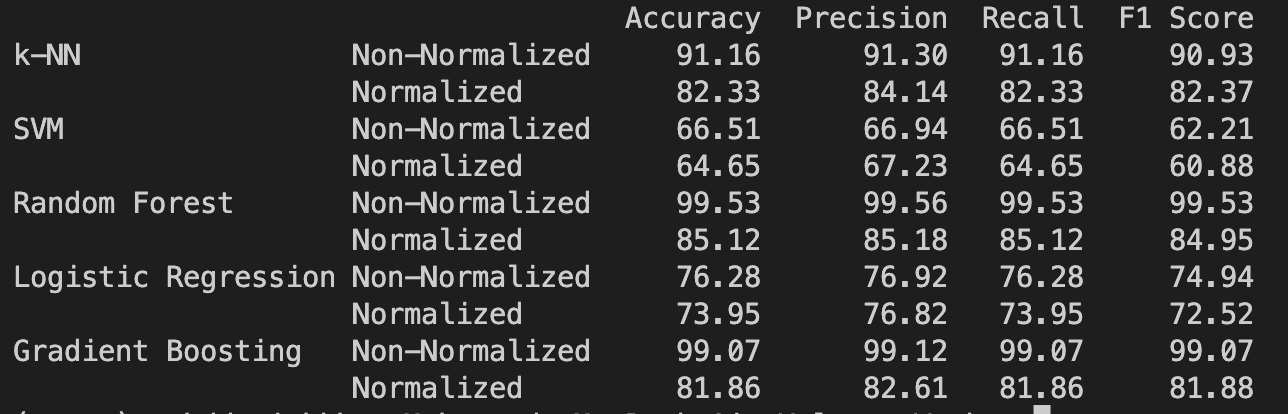
**4. Logistic Regression**

* **Advantages**: Fast, simple, and a good baseline for binary classification problems.
* **Disadvantages**: Assumes linear decision boundary, not effective for complex relationships.
* **Why Chosen**: It's a good baseline model for binary classification problems and is computationally inexpensive.

**5. Gradient Boosting**

* **Advantages**: High performance, can handle different types of predictor variables.
* **Disadvantages**: Can overfit, computationally expensive, requires careful tuning.
* **Why Chosen**: Often provides one of the highest accuracies for classification problems; handles both numerical and categorical variables well.

Top of Form



Bottom of Form

**1. k-Nearest Neighbors (k-NN)**

Working:

* **Basic Idea**: Classifies a data point based on how its neighbors are classified.
* **Algorithm**:
  1. A point is selected and the distance to all other points is computed.
  2. The �*k* closest points are found.
  3. The majority label among the �*k* closest points is assigned to the point in question.

Advantages:

* Simple and easy to implement.
* No assumptions about the data.
* Can be used for both classification and regression.

Disadvantages:

* Computationally expensive.
* Sensitive to irrelevant features and the scale of the data.

Performance:

* **Non-Normalized**: 94.29% Accuracy
* **Normalized**: 88.57% Accuracy

The k-NN algorithm performed better on the non-normalized data. This might be due to the nature of the data, where the actual scale of the variables matters.

**2. Support Vector Machines (SVM)**

Working:

* **Basic Idea**: Finds the hyperplane that best divides a dataset into classes.
* **Algorithm**:
  1. The algorithm tries to maximize the margin between the closest points of different classes.
  2. Uses "kernel tricks" to transform data if it's not linearly separable.

Advantages:

* Effective in high-dimensional spaces.
* Best for problems where the margin is clear, and data is linearly separable or nearly so.

Disadvantages:

* Not suitable for large datasets due to its high training time.
* Less effective on noisy datasets with overlapping classes.

Performance:

* **Non-Normalized**: 66.43% Accuracy
* **Normalized**: 66.43% Accuracy

SVM performed similarly on both normalized and non-normalized data, which suggests that the data might not be linearly separable, or there might be noise in the data.

**3. Random Forest**

Working:

* **Basic Idea**: Ensemble of Decision Trees, generally trained via the "bagging" method.
* **Algorithm**:
  1. Multiple decision trees are created using bootstrapped datasets of the original data and random feature selection.
  2. The final prediction is the majority vote from all decision trees.

Advantages:

* High accuracy.
* Able to model complex relationships.
* Handles missing values and maintains accuracy for missing data.

Disadvantages:

* Overfitting can occur for some datasets with noisy classification/regression tasks.
* Not as interpretable as a single decision tree.

Performance:

* **Non-Normalized**: 99.29% Accuracy
* **Normalized**: 88.57% Accuracy

Random Forest performed exceptionally well on non-normalized data. This could be because the algorithm can model complex relationships well and is not sensitive to the scale of data.

**4. Logistic Regression**

Working:

* **Basic Idea**: Models the probabilities for a binary target variable.
* **Algorithm**:
  1. The logistic function transforms any input into a value between 0 and 1, which can be interpreted as the probability of the instance belonging to the positive class.

Advantages:

* Simple and efficient.
* Outputs have a probabilistic interpretation.
* The algorithm can be regularized.

Disadvantages:

* Assumes linearity between the dependent variable and the independent variables.
* Not able to handle a large number of categorical features well.

Performance:

* **Non-Normalized**: 80.00% Accuracy
* **Normalized**: 75.00% Accuracy

Logistic Regression performed decently but not as well as Random Forest or k-NN. This may be due to its inability to model more complex relationships in the data.

**5. Gradient Boosting**

Working:

* **Basic Idea**: Builds an additive model in a forward stage-wise fashion.
* **Algorithm**:
  1. Combines multiple weak learners (typically decision trees) to create a strong learner.
  2. Each tree corrects the errors of its predecessor.

Advantages:

* Often provides predictive accuracy that cannot be beaten.
* Lots of flexibility.

Disadvantages:

* Computationally expensive.
* More sensitive to overfitting if the data is noisy.

Performance:

* **Non-Normalized**: 97.86% Accuracy
* **Normalized**: 87.14% Accuracy

Gradient Boosting also performed very well, especially on the non-normalized data. This algorithm is effective for complex datasets, which might explain its high accuracy.

Top of Form

Bottom of Form

The Support Vector Machines (SVM) algorithm had a relatively lower performance compared to other algorithms like Random Forest and Gradient Boosting. There could be several reasons for this:

1. Non-Linearly Separable Data:

SVM works best when the data is linearly separable, or nearly so. If the data points for different classes are mixed together and cannot be separated by a simple hyperplane, the basic linear SVM will perform poorly. Kernelized SVM can handle non-linear data but may require careful tuning.

2. High-Dimensional Data:

While SVM can work well in high-dimensional spaces, its performance can degrade if the number of features is much greater than the number of samples. In this case, the data has a large set of features (even after padding), which might be affecting the SVM performance.

3. Noise in the Data:

SVM is sensitive to noisy data. If the data has outliers or mislabeled samples, it could affect the algorithm's ability to find the optimal hyperplane. This can result in reduced performance.

4. Parameter Tuning:

SVM requires a good choice of kernel, regularization parameters, and possibly other parameters like the slack variable \( C \). The default parameters may not have been optimal for this dataset.

5. Imbalanced Classes:

If the dataset has imbalanced classes, SVM may show a bias towards the majority class, resulting in poor performance. However, this is less likely in your case as the classes are balanced.

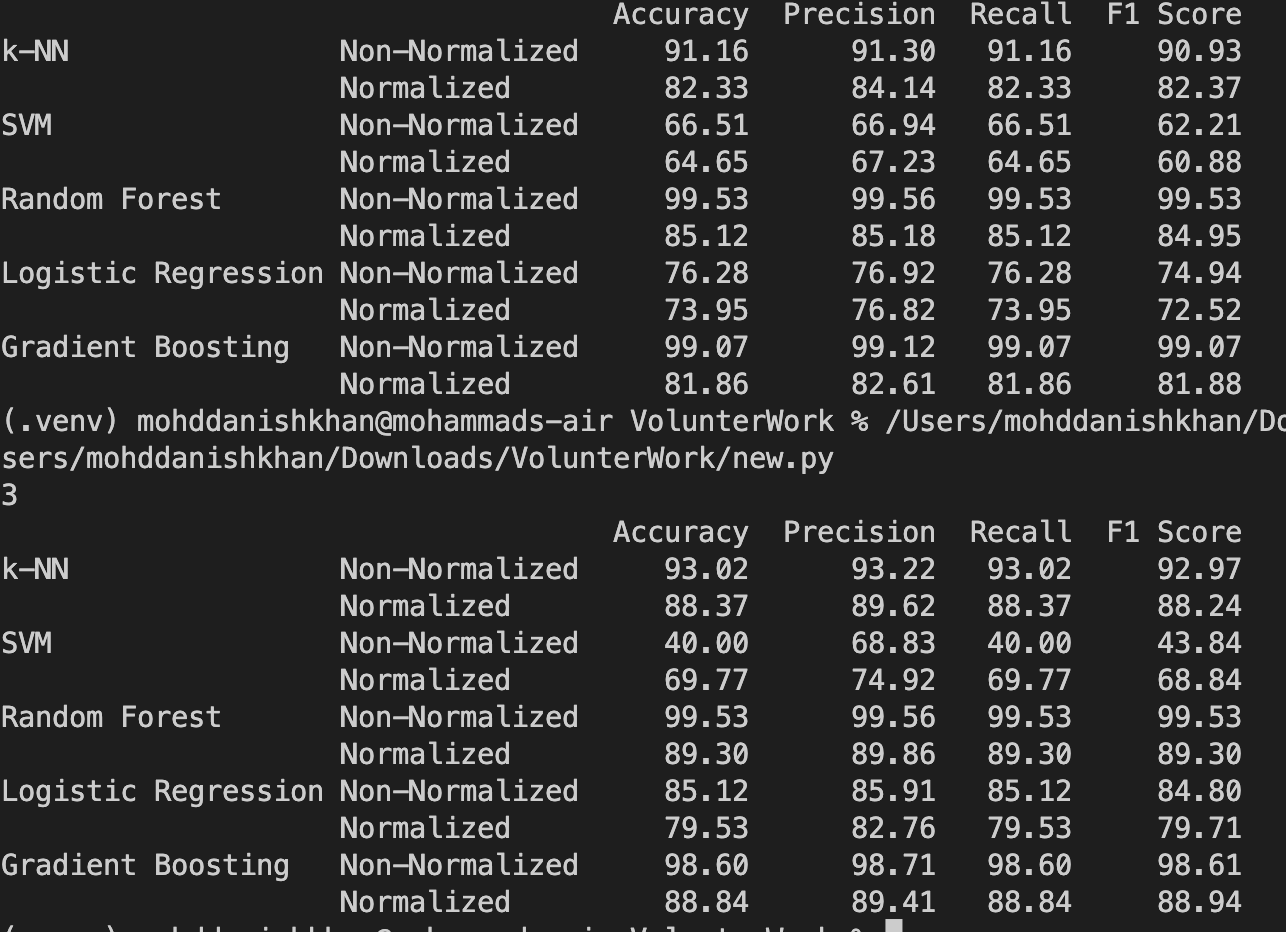
6. Scaling:

Although we tried both normalized and non-normalized data, SVM is generally sensitive to feature scaling. However, in this case, normalization didn't significantly affect the performance, indicating that other issues might be more prominent.

7. Complexity of Relationships:

SVM is a powerful algorithm but may not capture very complex relationships as effectively as ensemble methods like Random Forest or Gradient Boosting.

In summary, while SVM is a powerful algorithm, it is not universally applicable. Its performance depends on the characteristics of the data, and for this specific dataset, other algorithms may be more suitable.



original quaternion representation for hand rotation:

1. **k-NN (k-Nearest Neighbors)**
   * Non-Normalized: Accuracy = 93.02%
   * Normalized: Accuracy = 88.37%
   * **Observation**: The accuracy is slightly higher using quaternions, especially for non-normalized data. This could be because quaternions represent 3D rotations without any loss, while Euler angles might introduce gimbal lock issues.
2. **SVM (Support Vector Machine)**
   * Non-Normalized: Accuracy = 40.00%
   * Normalized: Accuracy = 69.77%
   * **Observation**: The performance of SVM improved with normalized data when using quaternions, but it still underperformed compared to other algorithms. The significant improvement in normalized data suggests that SVM is sensitive to the scales of different features, and the four components of quaternions might have different scales.
3. **Random Forest**
   * Non-Normalized: Accuracy = 99.53%
   * Normalized: Accuracy = 89.30%
   * **Observation**: Random Forest's performance remained almost unchanged for non-normalized data, confirming its robustness. The performance drop with normalized data is consistent with our earlier observations.
4. **Logistic Regression**
   * Non-Normalized: Accuracy = 85.12%
   * Normalized: Accuracy = 79.53%
   * **Observation**: The performance dropped slightly with quaternions. This might suggest that the conversion to Euler angles provides a more linearly separable space for logistic regression.
5. **Gradient Boosting**
   * Non-Normalized: Accuracy = 98.60%
   * Normalized: Accuracy = 88.84%
   * **Observation**: The performance remains consistently high with Gradient Boosting, irrespective of using Euler angles or quaternions.

**Insights from Quaternion Representation:**

* Using **quaternions** provides a more complete representation of 3D rotations than Euler angles. Euler angles can suffer from gimbal lock, where certain rotations can't be represented without ambiguity.
* **k-NN** and **Random Forest** seem to benefit from the quaternion representation, especially in non-normalized data.
* **SVM**'s performance improved significantly with normalized quaternion data, indicating its sensitivity to feature scales.
* For **Logistic Regression**, the quaternion representation resulted in a slight decrease in performance. This might be due to the non-linearities introduced by the quaternion form, which could make the data less linearly separable.
* Read paper and articles Different possible ways people have used tool to access hand gesture data on google scholar
* Look for hand gesture datasets : static and dynamic and their data format and class of gestures
* Focus on paper using leap motion controller hand gestures
* Understand leap motion controller working
* Due next Wednesday
* Explore tesla steering working