ML Pipeline Analysis and Porting Report by Anthony Zhang and Amit Adate

Feature Selection: Explanation of the feature selection methods used (Univariate, RFE, and one additional method). Include which features were selected and why.

We used the following feature selection methods:

- **Univariate Selection:** Uses the ANOVA F-test to score each feature independently. **accel_x_std** had the highest F-score, making it the most important.
- **Recursive Feature Elimination (RFE):** Iteratively removes the least important features using a RandomForestClassifier. RFE selected **accel_x_std**, **accel_z_std**, **accel_z_min**, and **accel_z_max** as the top features.
- **Decision Tree-Based Selection:** A shallow Decision Tree was used to quickly compute feature importances. This method highlighted **accel_x_std** (again the top feature) along with **accel_y_median**, **accel_z_mean**, **accel_z_median**, **gyr_y_min**, and **gyr_z_std**.

And for our final feature set, we combined the results from all three methods, and chose **accel_x_std**, **accel_y_std**, **gyr_z_std**, **accel_z_min**, **accel_z_max**, and **accel_z_std**, as these consistently demonstrated strong predictive power.

Feature	Ranking
accel_z_std	1
accel_x_std	1
accel_z_max	1
accel_z_min	1
gyr_z_std	2
accel_y_std	2
gyr_y_mean	3
gyr_z_min	3
gyr_z_median	3
accel_z_median	3

Feature	Importance
accel_z_std	0.096195
accel_x_std	0.079252
gyr_x_max	0.068131
accel_y_std	0.062602
accel_x_min	0.059013
gyr_z_min	0.057935
gyr_y_std	0.054514
gyr_z_max	0.049660
accel_z_max	0.047877
accel_z_mean	0.040771

Model Training: Description of the models trained (Decision Tree, Gradient Boosting, and 4 additional models). Explain why you chose each model and report their performance.

We trained 6 different models and profiled their performance, here is every model and a rationale for their selection

Decision Tree Classifier: A non-parametric supervised learning method that creates a model that predicts the target by learning simple decision rules inferred from the data features. It's intuitive, easy to interpret, and serves as a good baseline for comparison.

Gradient Boosting Classifier: An ensemble technique that builds trees sequentially, with each tree correcting the errors of its predecessors. It's powerful for capturing complex patterns and often achieves high accuracy on structured data like sensor readings.

Random Forest Classifier: An ensemble of decision trees that reduces overfitting through bagging and feature randomization. We chose this because it works well with the standard deviation features in your dataset and is robust to outliers common in sensor data.

K-Nearest Neighbors: A simple yet effective instance-based learning algorithm that classifies based on similarity measures. This is particularly suitable for your accelerometer and gyroscope data as activity patterns often cluster together in feature space.

Logistic Regression: A linear model that estimates probabilities for classification. I included this as it provides good interpretability of feature importance and serves as a useful baseline to compare against more complex models.

LightGBM Classifier: A gradient-boosting framework that uses tree-based learning algorithms. It's designed for efficiency and scalability with innovations like Gradient-based One-Side Sampling and Exclusive Feature Bundling, making it faster than traditional boosting implementations. LightGBM performs well on time-series sensor data by effectively capturing complex non-linear patterns while maintaining reasonable computation requirements. The model's leaf-wise tree growth strategy gives it robust performance on structured mHealth data, balancing accuracy and speed.

Model Performance:

Model Performance Summary

Model	Training Time (s)	Prediction Time (s)	Accuracy
DecisionTreeClassifier	17.13	0.09	49.14%
GradientBoostingClassifier	864.18	3.14	51.59%
RandomForestClassifier	154.43	3.63	51.62%
KNeighborsClassifier (1)	8.16	72.79	81.37%
KNeighborsClassifier (2)	7.38	307.92	83.90%
LGBMClassifier	131.94	47.59	73.17%

Key Observations:

1. **Highest Accuracy**: The second KNeighborsClassifier implementation achieved the best accuracy at 83.90%, followed by the first KNN implementation at 81.37%.
2. **Training Time**:
   - GradientBoostingClassifier was by far the slowest to train (864.18s)
   - KNN models were fastest to train (7-8s)
   - RandomForest and LightGBM had moderate training times (131-154s)
3. **Prediction Time**:
   - DecisionTreeClassifier had the fastest prediction time (0.09s)
   - KNN models had extremely slow prediction times (73-308s)
   - LGBMClassifier had a moderate prediction time (47.59s)
4. **Efficiency vs. Accuracy Trade-off**:
   - LGBMClassifier offers a good balance between accuracy (73.17%) and moderate prediction time
   - KNN models achieve highest accuracy but at the cost of very slow prediction times
   - Tree-based models (Decision Tree, Random Forest, Gradient Boosting) offer modest accuracy (49-52%) with varied computational demands

LOSO Evaluation: **P9 vs P10**

The Leave-One-Subject-Out evaluation shows substantial performance variation between participants. P10's accuracy (59.95%) greatly exceeded P9's (28.60%), demonstrating significant individual differences in activity recognition.

This variation highlights the challenge of creating person-independent activity recognition models. The consistent feature importance suggests the selected features capture relevant motion patterns, but individual movement styles significantly impact classification success.

LOSO Evaluation Comparison: P9 vs P10

Overall Performance Metrics

Metric	Participant P9	Participant P10	Difference
Accuracy	28.60%	59.95%	+31.35%
Training Time (s)	161.89	169.55	+7.66
Prediction Time (s)	3.24	3.82	+0.58
Weighted Avg F1-score	0.39	0.60	+0.21

Per-Class Performance (Precision)

Activity Class	P9 Precision	P10 Precision	Difference
Ascending stairs	0.05	0.27	+0.22
Dancing	0.00	0.00	0.00
Descending stairs	0.08	0.40	+0.32
Jumping	0.96	0.44	-0.52
Rest	0.00	0.48	+0.48
Running	0.90	0.59	-0.31
Sitting	0.39	0.95	+0.56
Standing	0.97	0.89	-0.08
Walking	0.47	0.44	-0.03

Feature Importance Ranking

Rank	Feature	P9 Importance	P10 Importance
1	accel_z_min	5910	5899
2	accel_z_max	5591	5713
3	accel_x_std	4346	4270
4	gyr_z_std	4051	4031
5	accel_y_std	3652	3637
6	accel_z_std	3450	3450

Conclusion

Feature selection was the smoothest part of this project. We were able to quickly zone in on the six features that we were going to use later. In terms of model performance, we found that KNN outperformed the likes of linear regression and tree-based algorithms. This was a little surprising to us since we expected the performance difference to be less drastic. To our understanding, this may be because the data point we are classifying requires a “smoother” curve that only KNN were able to produce. We found a good tradeoff between performance and runtime when we chose LGBM which is a Light Gradient-Boosting Machine. This was able to get us 73% accuracy on the test set with the requested 33% test data split.

We performed LOSO on two participants instead of one, P9 and P10. P9 showed better precision for jumping, running, and standing, while P10 performed better on all other activities. P10's overall accuracy was 31.35% higher than P9's, showing significant inter-participant variability. The feature importance ranking remained consistent between participants, suggesting stable feature relevance.

These results highlight the challenge of person-independent activity recognition and demonstrate that model performance can vary drastically depending on individual movement patterns.

Finally, it was tragic for us to find out that the provided library (micromlgen) does not support the porting of KNN to C. So we ported the random forest model into c code instead.