Application of Machine Learning Algorithms to Smartphone Satellite Navigation Data for Precise Positioning

Karolina Tchilinguirova, Supervisor: Sunil Bisnath

GNSS Lab, Deparment of Earth and Space Enginnering, Lassonde School or Enginnering, York Univeristy

Introduction

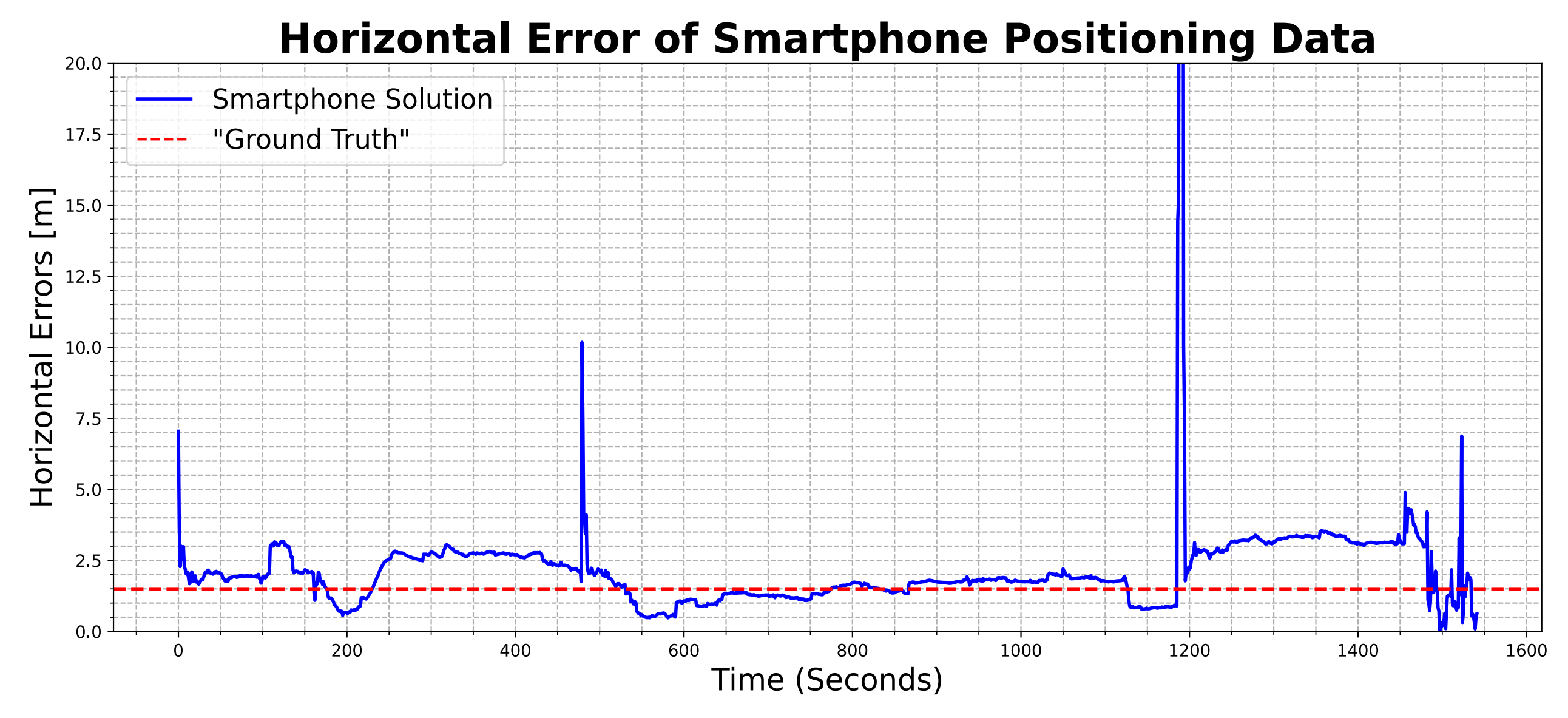
Smartphone-based navigation has become an indispensable tool for most people. Despite the advanced technology squeezed into modern smartphones, the Global Navigation Satellite Systems (GNSS) antennas and chipsets are low-cost, leading to significant positional accuracy limitations. Thus, software plays a major role in enhancing positioning accuracy caused by the nature of the low-cost chipsets combined with environment satellite signal blockages.

Since artificial intelligence is on the rise and impacting all industries, this project aims to explore the utility of machine learning (ML) algorithms in improving the positioning accuracy of smartphones. This is approached through a preliminary investigation of the capabilities of machine learning algorithms to predict the precision of positioning data produced by a smartphone.

This research can lead to implementing ML into smartphone position processing to improve user accuracy while maintaining the current hardware's low cost and small size. This project aligns with the United Nations' 9th Sustainable Development Goals; Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation. Ultimately, satellite navigation is essential in modern-day society, and working to provide precise positioning with mass market equipment means building resilient infrastructure for the global population's safety and economic development.

Objective

This project aims to learn about the feasibility of implementing AI tools, specifically machine learning, into the processing of smartphone positioning data to improve its accuracy for driving navigation. Fig. 1 shows the wide range of errors in smartphone positioning. The smartphone solution is compared against a corresponding reference ("true") trajectory coordinates determined with high-precision equipment. This chart shows that the average smartphone horizontal error is 3.8 meters. During interrupted satellite signals, such as when the car moves under an overpass or bridge, the error can shoot up beyond 10 meters.



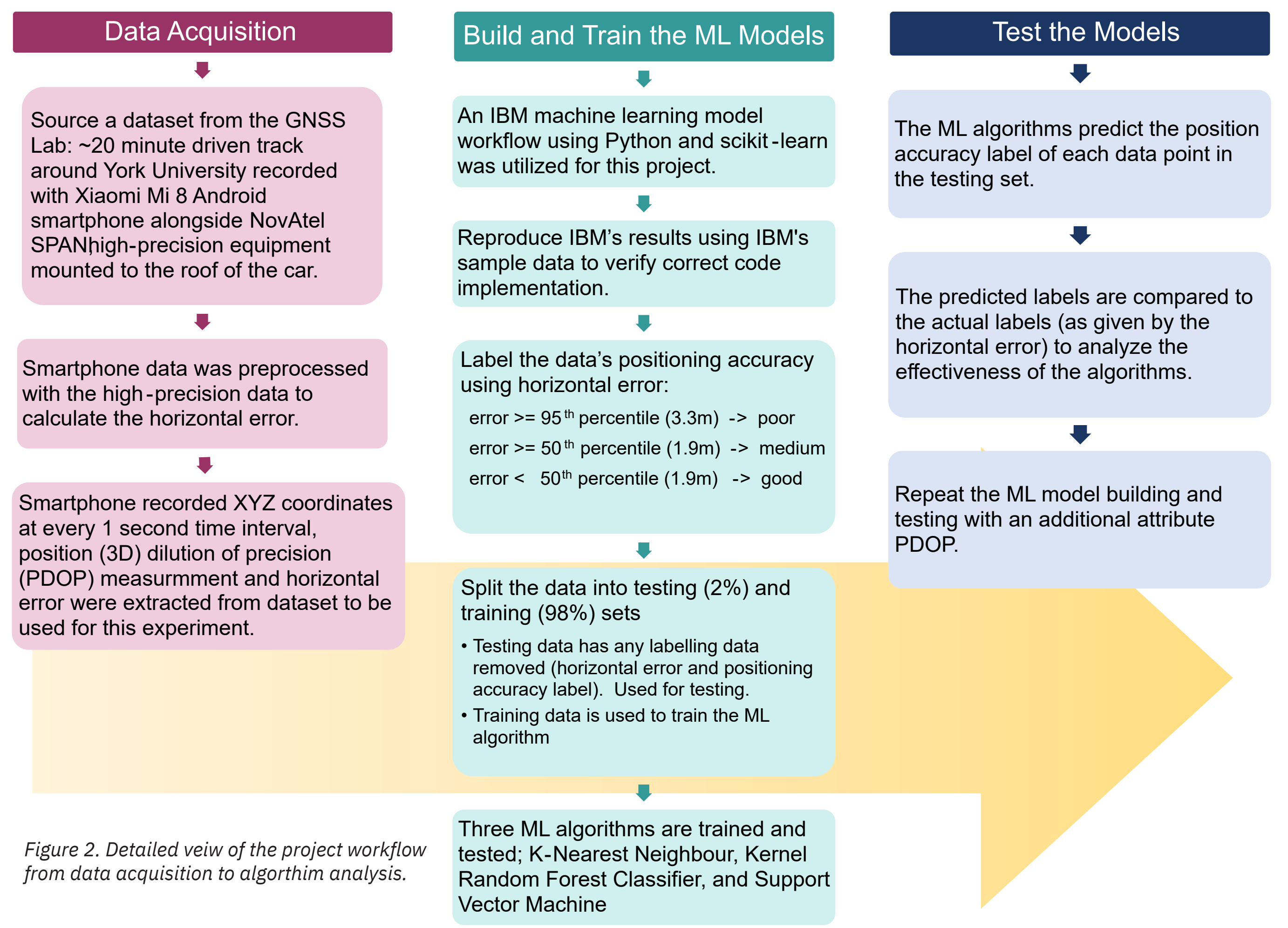
	50th percentile errors (m)	95th percentile errors (m)	Root mean square error (m)
Smartphone Solution	1.9	3.3	3.8

Figure 1. The plotted horizontal error of the smartphone positioning solution of driven ~20-minute route around the campus of York University. The "true" value is the trajectory coordinates determined with high-precision equipment mounted on the vehicle. Horizontal error is calculated as the deviation of the smartphone measurements away from the high-precision measurements. Spikes in error are caused by interruptions of the satellite signals when the vehicle passes under bridges. The 95th and 50th percentile errors are included in the table below.

It was hypothesized that given a driven track dataset of positioning coordinates as measured by a smartphone; ML algorithms would successfully predict the quality of the positioning measurements. Furthermore, it is hypothesized that the prediction accuracy will increase if additional information, such as satellite geometry, is provided to the ML model.

Methods

This experiment was programmed in Python using multiple data science and machine learning packages, including NumPy, SciKit-Learn, Pandas, and Matplotlib. The workflow follows a basic structure of labelling the data, splitting the data into two sets; training and testing. Next, the machine learning model is trained with the training set, followed by an analysis of its effectiveness with the testing set. More details can be seen in Figure 2.



Data Acquisition

Source a dataset from the GNSS Lab: ~20 minute driven track around York University recorded with Xiaomi Mi 8 Android smartphone alongside NovAtel SPANhigh-precision equipment mounted to the roof of the car.

Smartphone data was preprocessed with the high-precision data to calculate the horizontal error.

Smartphone recorded XYZ coordinates at every 1 second time interval, position (3D) dilution of precision (PDOP) measurmment and horizontal error were extracted from dataset to be used for this experiment.

Build and Train the ML Models

An IBM machine learning model workflow using Python and scikit-learn was utilized for this project.

Reproduce IBM's results using IBM's sample data to verify correct code implementation.

Label the data's positioning accuracy using horizontal error:

error >= 95th percentile (3.3m) -> poor
error >= 50th percentile (1.9m) -> medium
error < 50th percentile (1.9m) -> good

Split the data into testing (2%) and training (98%) sets
- Testing data has any labelling data removed (horizontal error and positioning accuracy label). Used for testing.
- Training data is used to train the ML algorithm

Three ML algorithms are trained and tested; K-Nearest Neighbour, Kernel Random Forest Classifier, and Support Vector Machine

Test the Models

The ML algorithms predict the position accuracy label of each data point in the testing set.

The predicted labels are compared to the actual labels (as given by the horizontal error) to analyze the effectiveness of the algorithms.

Repeat the ML model building and testing with an additional attribute PDOP.

Figure 2. Detailed veiw of the project workflow from data acquisition to algorthim analysis.

The full dataset used in this study is visually represented in both two dimensions and three dimensions in Figure 4. Each point in the graph represents a 1-second measurement taken by the smartphone along the track. The red, yellow and green coloured points represent the labelling of data. The calculated horizontal error is available since this track was simultaneously measured with the NovAtel SPAN (high-precision GNSS equipment). This feature labels the data points as good, medium or poor in position accuracy. *Position accuracy* is the element which the ML models predict. This data is split into two subsets; the training and testing sets. The training data is used to train the ML model, and the testing data is used to compare against the predictions of the ML model, as shown in Fig.5, Fig.6 and Fig.7.

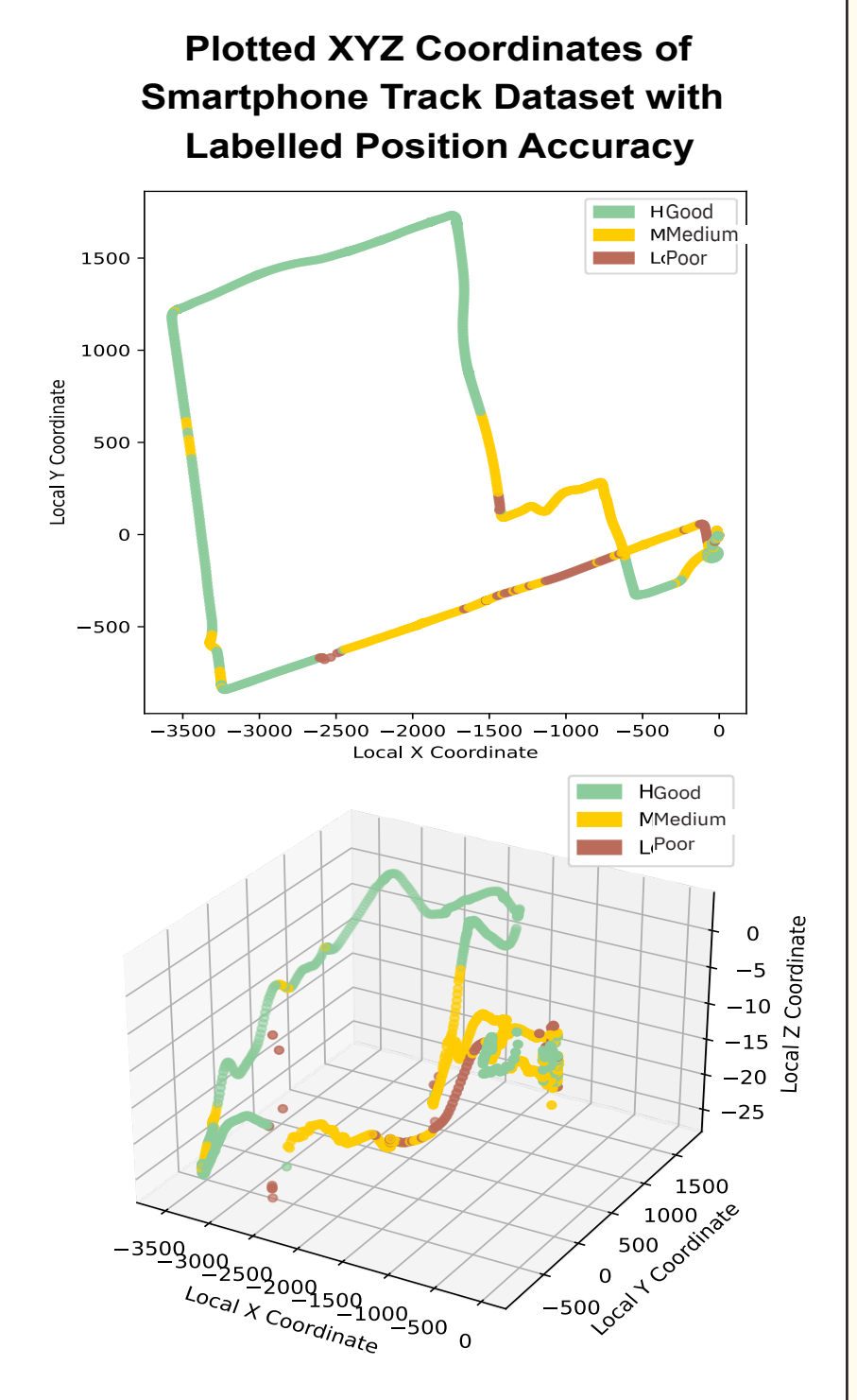


Figure. 4. A spatial representation of the full dataset.

Three rounds of model building and testing:

Round 1: The models were only provided the smartphone's measured XYZ coordinates to make predictions.

Round 2: The models were only provided the smartphone's measured position dilution of precision (PDOP) value to make predicitions.

Round 3: The models were given both the smartphone's measured XYZ and PDOP values to make predictions.

The dataset is run through three supervised ML Models:

K-Nearest Neighbour (KNN): A class label is assigned on the basis of a majority vote within a proximity—i.e. the label that is most frequently represented around a given data point is used (IBM).

Random Forest Classifier (RFC):): This algorithm creates a collection of random decisions trees from subsets of the data. The class label is assigned based on the averaging of these tree's decisions (IBM).

Kernel Support Vector Machine (SVM): This model maps the data into a high-dimensional feature space, and the classes are separated by a hyperplane. The class label is assigned based on where the data point is located within this feature space (IBM).

Results

The following figures each represent a separate round of ML algorithm runs. Each figure consists of 4 plots; the actual classification (leftmost plot) and the predictions made by the three different ML models. The actual classification represents the testing subset of the entire dataset. Therefore, it shows the classification based on the horizontal error. This subset is removed when training the models. The output is the model's attempt to predict these actual values. As labelled above each plot, the accuracy metric states the percentage of correct predictions. Additionally, the errors can be seen visually when comparing the prediction plots to the leftmost Actual Classification plot.

Round 1: In the first round of ML runs, the XYZ coordinates are the only GNSS artifact used to create predictions. As shown in Fig.5, the KNN algorithm performed the best of the three, correctly predicting 92% of data points. Its major struggle was pushing data too far into the 'medium' class, therefore, both overestimating the 'poor' measurements and underestimating the 'good' measurements. RFC and SVM performed similarly, with RFC being the least accurate of the three. Both algorithms consistently overestimate measurements, pushing 'poor' into the 'medium' class and 'medium' into the 'good' class.

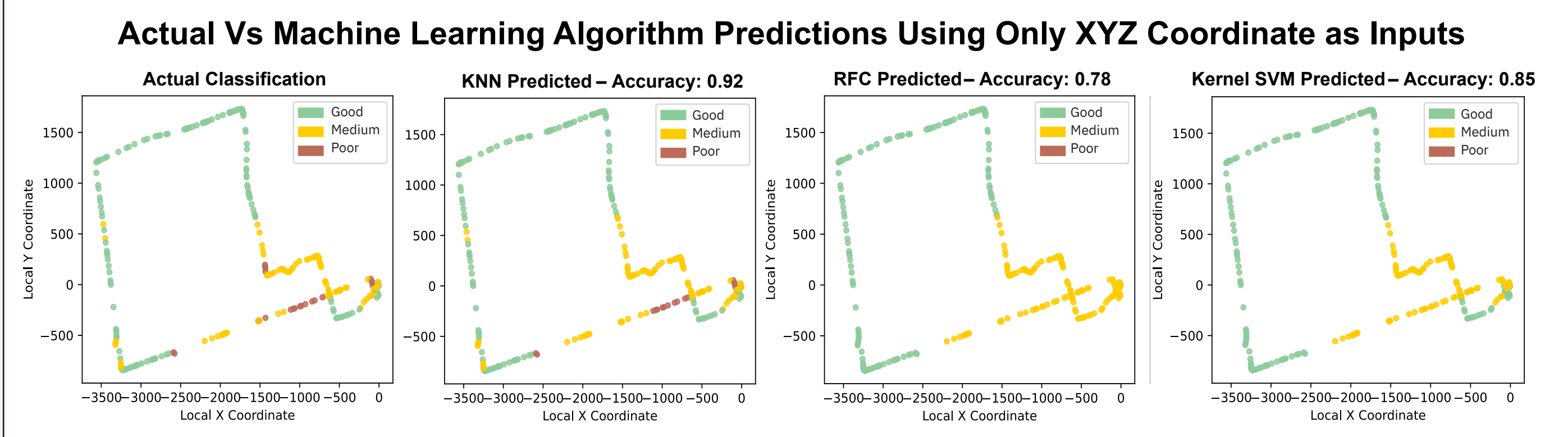


Figure. 5. The results from the 1st round of ML model runs.

Round 2: In this round of ML runs, the only GNSS artifact used to create predictions is the position dilution of precision (PDOP). This measurement reflects the geometry of the satellites. The lower the PDOP, the more favourable the satellite distribution, which correlates to higher accuracy position measurements (Novatel). As shown in Fig.6, KNN produced the highest accuracy of the three ML algorithms. KNN produced similar results to round 1, where data is pushed towards the middle. Additionally, RFC and SVM produced poor results, where RFC greatly overestimated the 'good' class, and SVM overestimated the 'medium' class.

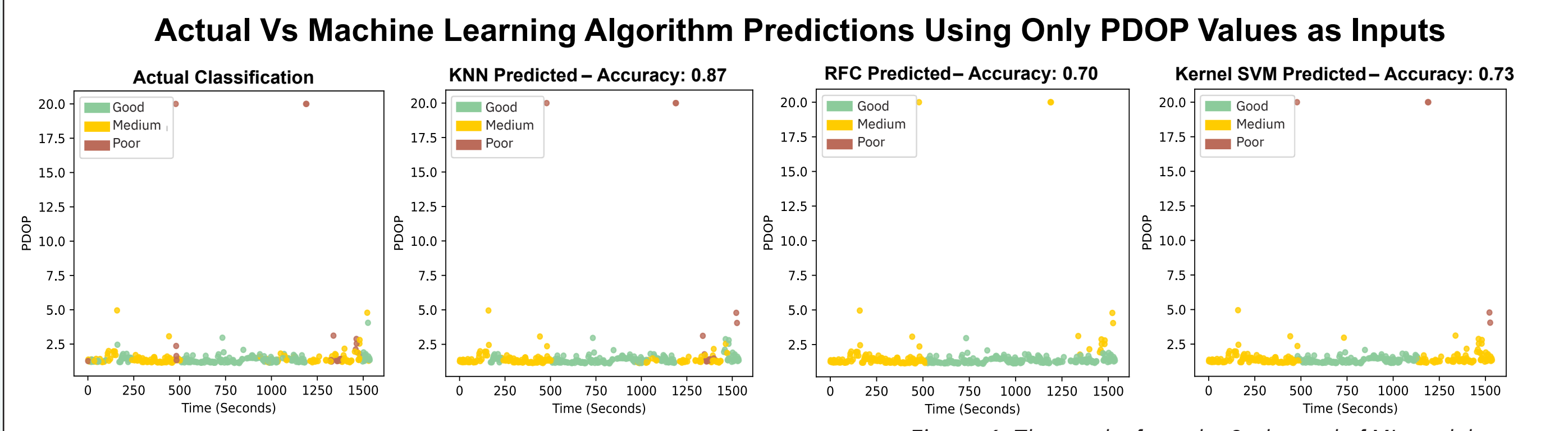


Figure. 6. The results from the 2nd round of ML model runs.

Round 3: In the final round of ML runs, the ML models used both XYZ coordinates and PDOP artifacts to form predictions. Following previous rounds, each model's performance concerning each other was the same; KNN produced the greatest accuracy, followed by SVM and RFC, with the lowest prediction accuracy. As shown in Fig.7, KNN's prediction slightly favoured the 'good' class, with the most significant incorrect predictions of the 'poor' class. RFC significantly favoured the 'medium' class, pushing both 'poor' and 'good' measurements into the 'medium' class. SVM performed similarly to KNN, with a slightly more significant bias towards the 'medium' class.

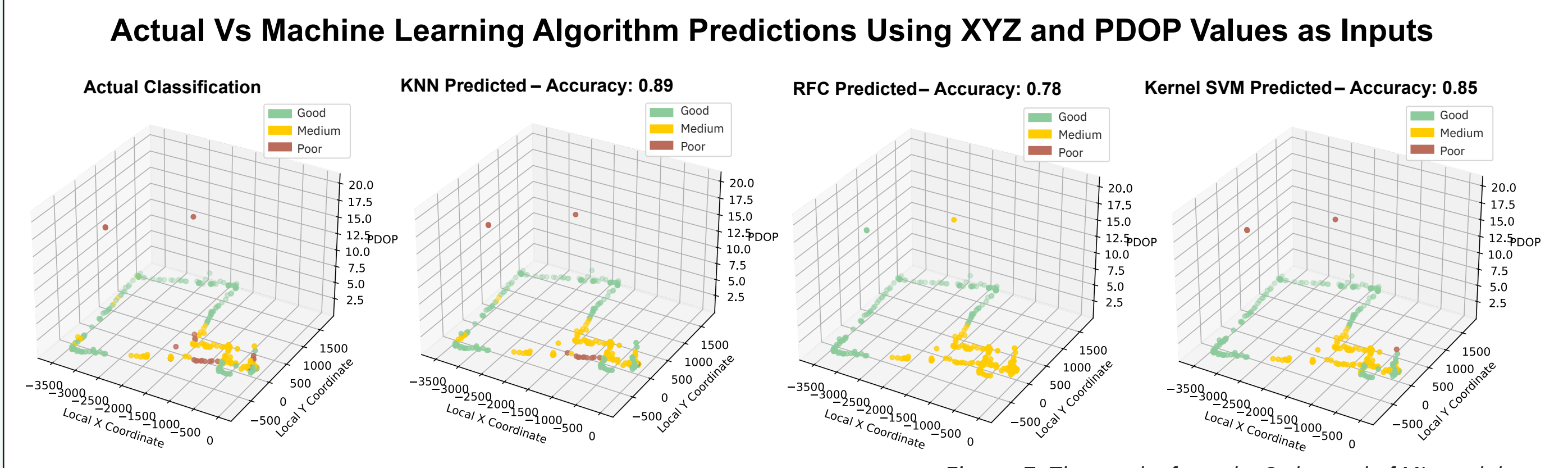


Figure. 7. The results from the 3rd round of ML model runs.

Conclusion

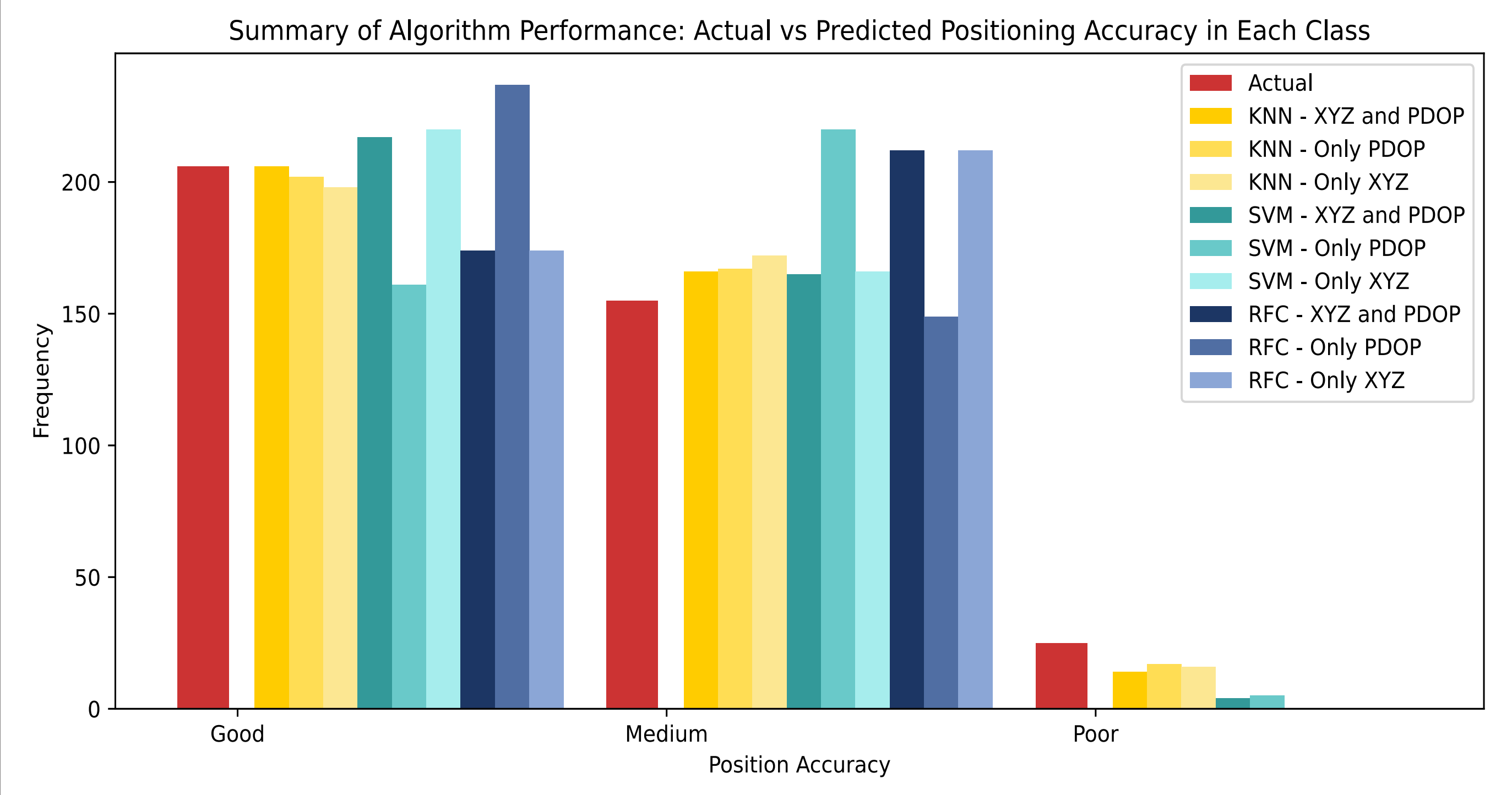


Figure. 8. Bar graph showing the number of datapoints within each classification from the actual testing dataset compared with each round and each machine learning algorithm's predictions.

A summary of this experiment, which highlights the differences in classification, is shown in Fig. 8; the closer the bar height is to the actual bars (red), the more accurate the model is at predicting the correct position accuracy class. Therefore, the three K-Nearest Neighbour rounds outperformed the other machine learning models, Random Forest Classifier and Kernal Support Vector Machine. Surprisingly, the first round of runs that only included the XYZ coordinates as GNSS artifact inputs for the model predictions performed the best. This is unexpected from the initial hypothesis because it was reasoned that adding additional information relevant to the quality of a positioning measurement would improve the model's ability to make correct predictions; however, this pattern was not observed. A potential explanation for this occurrence might result from the strong geometric interworkings of the KNN algorithm. Since KNN prediction is based on the distance between points and XYZ are concrete spatial measurements, the distance between coordinates may be more meaningful than the distance between coordinates and an arbitrary number in the form of a PDOP measurement.

This experiment was a preliminary exploration into implementing machine learning in GNSS applications, specifically in classifying position accuracy measurements of smartphone navigation solutions. This experiment showed that the ML model, specifically KNN is effective at classifying smartphone positioning accuracy, which can be used to reduce the weight of 'poor' measurements and improve the accuracy of positioning solutions. The major limitation of this experiment is that the models classified navigation data which was logged previously, therefore, leveraging known measurements from both previous and future locations. This would not be directly transferable to real-world applications as a user's future coordinates would be unknown and unavailable for the model's training. However, this opens an opportunity for future work where ML models are implemented live, such that the model is updated in real-time and is required to make predictions based only on a user's previously logged location. In conclusion, artificial intelligence applications are growing rapidly across all industries. Its implementation into GNSS applications will improve satellite navigation, ultimately contributing to building resilient infrastructure for the global population's safety and economic development.

References

DOP values for the satellites in the PDP solution. Novatel. (n.d.). https://docs.novatel.com/OEM7/Content/Logs/PDPDOP.htm

BM. (n.d.). How SVM works. https://www.ibm.com/docs/en/spss-modeler/18.2.2?topic=models-how-svm-works

Madhavan, S., & Sturdevant, M. (2019, December 4). *Build and test your first machine learning model using Python and scikit-learn.* IBM developer. https://developer.ibm.com/tutorials/build-and-test-your-first-machine-learning-model-using-python-and-scikit-learn/

Madhavan, S., & Sturdevant, M. (2019, December 4). *Learn classification algorithms using Python and scikit-learn.* IBM developer. https://developer.ibm.com/tutorials/learn-classification-algorithms-using-python-and-scikit-learn/

What is Random Forest?. IBM. (n.d.-a). https://www.ibm.com/topics/random-forest

What is the K-nearest neighbors algorithm?. IBM. (n.d.-b). https://www.ibm.com/topics/knn