

İTÜ



UCK358E : Term Project

CRN: 22828

Due on Thursday, May 23, 2024

Asst. Prof. Barış BAŞPINAR

Nisa Nur ELMAS

110210031

Table of Contents

1. Introduction.....	2
2. Data Preprocessing.....	3
2.1 GNSS Data Preprocessing	3
2.1.1 Dropping Missing Columns.....	3
2.1.2 Imputing Missing Values.....	4
2.1.3 Detecting Correlated Features.....	4
2.2 Ground Truth Data Preprocessing	5
2.3 IMU Data Preprocessing.....	6
2.3.1 Dropping Missing Columns.....	6
2.3.2 Filtering Message.....	6
2.4 Merging Datasets	7
2.4.1 Merging Train Datasets.....	7
2.4.2 Merging Test Datasets	8
3. Models Used	8
3.1 Random Forest	8
3.2 XGBoost	9
3.3 LightGBM.....	9
3.4 Neural Networks	10
3.4.1 Overfitting Problem	11
4. Future Improvements	11
5. Results.....	12
6. Conclusion	14

1. Introduction

This project involves analyzing and predicting the latitude and longitude coordinates using smartphone decimeter data. The goal is to preprocess the data, engineer relevant features, and apply different machine learning models to make accurate predictions. The performance of the models is evaluated using training, validation, and test sets, and the best model's results are submitted to a Kaggle competition.

Google selected more than 60 datasets collected from highways in the San Francisco Bay Area, USA, during the summer of 2020 as the dataset for the competition. Google provides datasets that include raw GNSS measurements, raw inertial sensor readings, and accurate location data. These datasets are provided along with Matlab source code that can be used to evaluate positioning performance in specific scenarios. The data was collected on highways in the San Francisco Bay Area, USA, and is provided along with correction data.

In the paper published by Google regarding their measurement study, many details are explained, including the sensor data collected from the points where the phones were placed inside the vehicle in relation to the GT Antenna, as shown in the figure 1. To make latitude and longitude estimations with this data, the first step was to thoroughly determine what the data signifies.

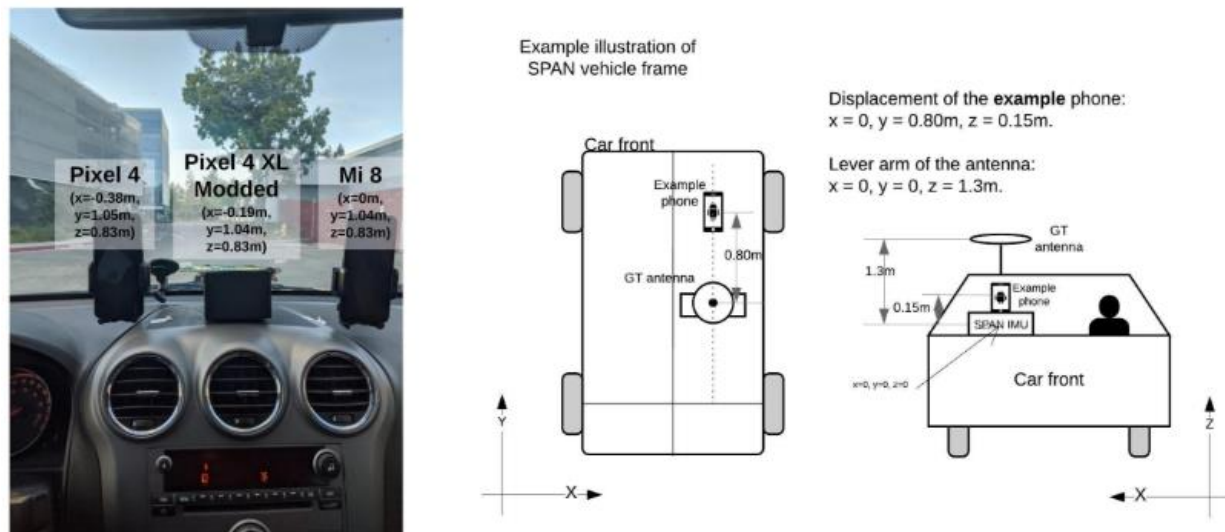


Figure 1: Measurement of sensor values

2. Data Preprocessing

Data preprocessing is a crucial step in the data science pipeline, especially when working with real-world datasets that are often noisy and incomplete. For this project, GNSS (Global Navigation Satellite System) data, IMU (Inertial Measurement Unit) data, and ground truth data were used. The preprocessing steps were meticulously carried out to ensure the data is clean, consistent, and ready for machine learning model training.

2.1 GNSS Data Preprocessing

The GNSS data contains various measurements related to satellite navigation. The preprocessing steps for this data contains below steps.

2.1.1 Dropping Missing Columns

Columns that were missing for all rows were removed. To see which columns are missing, `.isnull()` function is used and also visualized with Seaborn's heatmap as shown in the figure 2.

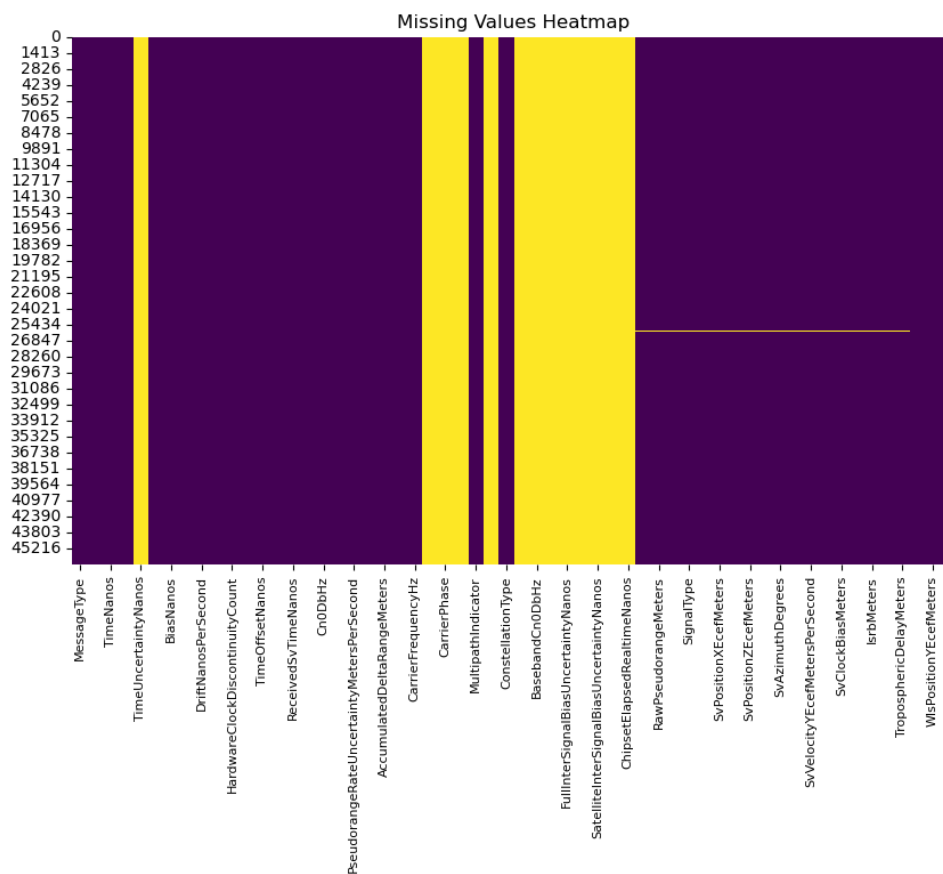


Figure 2: Representation of missing values of GNSS data

2.1.2 Imputing Missing Values

Many columns had missing values which needed to be imputed to make the dataset usable. Numeric columns were imputed with the mean of the column values and non-numeric columns with the most frequent value.

2.1.3 Detecting Correlated Features

WlsPositionXEcefMeters, WlsPositionYEcefMeters and WlsPositionZEcefMeters features are crucial because of the project aim is to determine the smartphone's location. Therefore, correlation matrix is projected with the help of Seaborn's heatmap() function as shown in figure 3. As a result, utcTimeMillis, TimeNanos, FullBiasNanos and ReceivedSvTimeNanosSinceGpsEpoch features are detected that they have negative and positive correlation between the WLS Position datas.

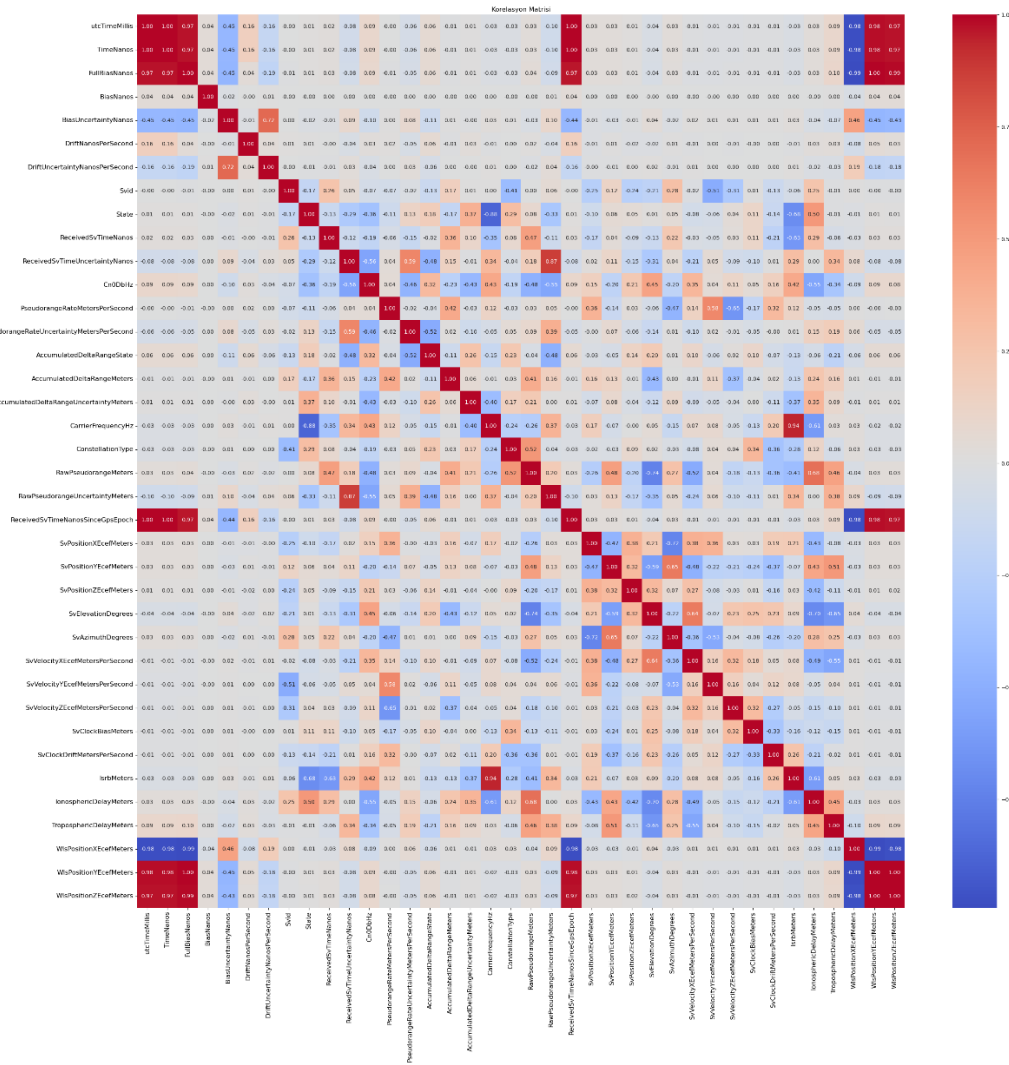


Figure 3: Correlation of cleaned GNSS data

2.1.4 Cleaning Duplicates

Duplicate rows based on the utcTimeMillis column were removed to ensure that each timestamp has a unique set of measurements. This step is crucial to make the submission file correctly.

2.2 Ground Truth Data Preprocessing

The ground truth data provides the actual positions and is essential for training and validating the models. The preprocessing steps for this data contains unnecessary and missing column dropping.

2.2.1 Dropping Unnecessary and Missing Columns

Columns that were missing for all rows and useless columns like MessageType and Provider columns were removed. To see which columns are missing, isnull() function is used and also visualized with Seaborn's heatmap as shown in the figure 4.

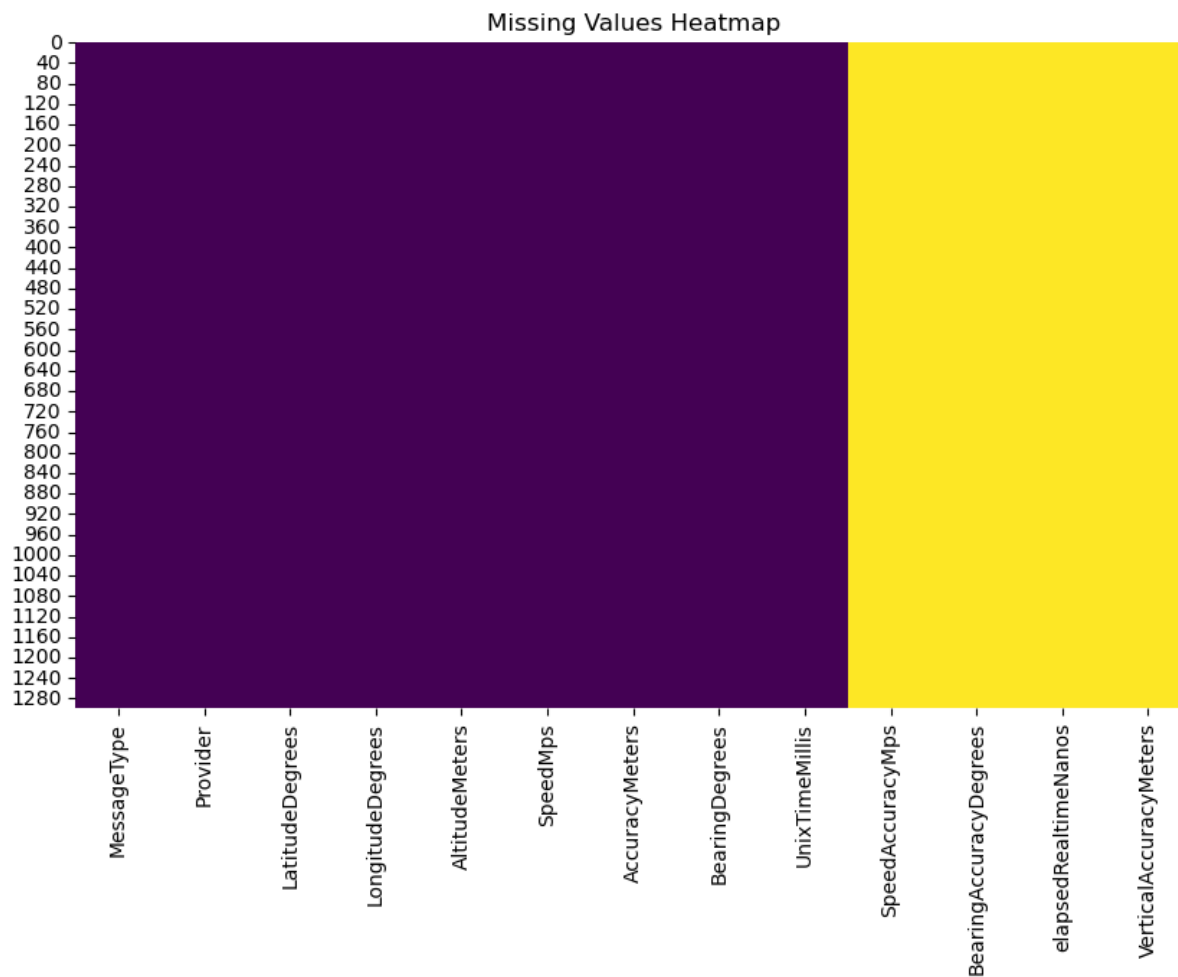


Figure 4: Representation of missing values of Ground Truth data

2.3 IMU Data Preprocessing

The IMU data consists of measurements from various sensors (accelerometers, gyroscopes, etc.) and provides additional context to the GNSS data.

2.3.1 Dropping Missing Columns

Columns that were missing for all rows were removed. To see which columns are missing, `isnull()` function is used and also visualized with Seaborn's heatmap as shown in the figure 5.

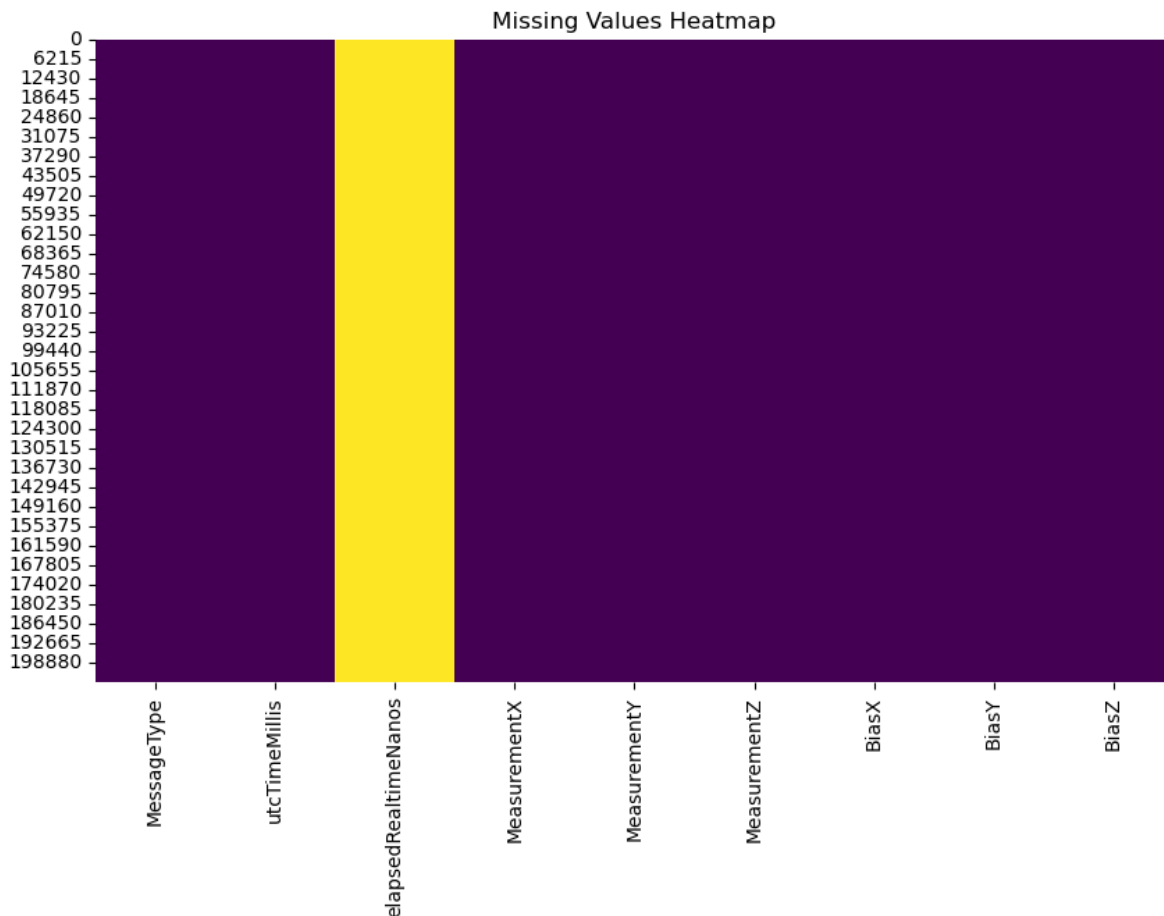


Figure 5: Representation of missing values of IMU data

2.3.2 Filtering Message

MessageType column has measurements of magnetometer gyroscope and accelerometer. In this project, only acceleration values were used, because of its relation with displacement. So, in the MessageType column, the rows that consists UncalGyro and UncalMag were dropped.

2.4 Merging Datasets

Merging datasets is a pivotal step in the data preprocessing pipeline, especially when working with multimodal data sources like GNSS, IMU, and ground truth data. The merging process aligns these diverse datasets into a unified format, ensuring that each timestamp has corresponding measurements from all data sources. Below are the detailed steps and considerations taken during the merging process.

2.4.1 Merging Train Datasets

In the train datasets given by the competition, there are 3 type of dataframes for one smartphone location measurement: GNSS, IMU and Ground Truth data. These 3 dataframes were collected for training the machine learning model.

Both GNSS and ground truth data contain a `utcTimeMillis` column that records the timestamp of each measurement. Ensuring that these timestamps are aligned is the first step in merging the datasets. If necessary, the timestamps were rounded or interpolated to match each other more closely.

Each row in the GNSS data was matched with the corresponding row in the ground truth data based on the timestamp. This ensured that every GNSS measurement was supplemented with the corresponding ground truth position and other related metrics. In addition, in the post-merge step, there might be rows with missing ground truth data due to timestamp mismatches or gaps in the ground truth data. These missing values were handled using interpolation techniques to ensure continuity in the dataset.

The IMU data often has a higher sampling rate compared to GNSS data. Therefore, the IMU data was resampled to match the timestamp frequency of the GNSS data. This was done by averaging or interpolating IMU measurements within the time windows defined by the GNSS timestamps.

Similar to the previous merge, the `utcTimeMillis` column was used to align the IMU data with the already merged GNSS and ground truth data. Again, this involved ensuring that timestamps were synchronized and any necessary rounding or interpolation was performed. After merging, the dataset was checked for any remaining missing values. Interpolation was applied where necessary to fill in gaps, ensuring that the dataset was complete and ready for the subsequent feature engineering and modeling phases.

2.4.2 Merging Test Datasets

In the test datasets given by the competition, there are 2 type of dataframes for one smartphone location measurement: GNSS and IMU data. These 2 dataframes were collected for testing the machine learning model.

Nearly every step of the merging test datasets was same. The only difference is that IMU data was not matched with GNSS-Ground Truth data. Instead, it matched with only preprocessed GNSS data. The difference is because test datasets of one smartphone location measurement do not have ground truth data.

3. Models Used

For this project, Random Forest, LightGBM and XGBoost algorithms and Neural Networks were used.

3.1 Random Forest

To explore the dataset's limitations, ability and features, random forest algorithm was used as a first try. The advantage of random forest was the ability to reduce the risk of overfitting among other decision tree algorithms. In addition, two of the disadvantage of random forest are that it took up so much memory space and long training time.

Random Forest Regressor demonstrated solid performance in capturing the underlying patterns in the data, thanks to its ability to handle high-dimensional feature spaces and interactions among features.

The score for the submission dataset that predicted by random forest in the Kaggle site was 92045.967 as shown in figure 6.



Figure 6: Score of the random forest algorithm

3.2 XGBoost

In this project, XGBoost was chosen for its robustness in handling diverse datasets with complex relationships and large feature spaces. It excels in capturing both linear and nonlinear patterns in the data and is less prone to overfitting compared to other ensemble methods.

One notable advantage of XGBoost is its flexibility in handling missing data, which reduces the need for extensive data preprocessing. However, it may require careful tuning of hyperparameters to achieve optimal performance.

The XGBoost Regressor exhibited impressive performance in training time. However, probably because of the training dataset's size, it was overfitted as we can understand from the final score.

Upon evaluation on the submission dataset in the Kaggle competition, the XGBoost model achieved a commendable score of 432411.829 as shown in figure 7.

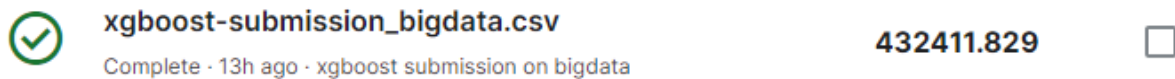


Figure 7: Score of the XGBoost algorithm

3.3 LightGBM

To leverage the dataset's complexity and large feature space efficiently, LightGBM was employed as it can handle large datasets with high dimensionality and provides faster training times compared to other gradient boosting algorithms.

One of the main advantages of LightGBM is its ability to handle categorical features directly, without requiring one-hot encoding, thereby reducing the preprocessing overhead. However, it may be more sensitive to overfitting compared to other algorithms if not properly tuned.

The LightGBM Regressor demonstrated superior performance in capturing intricate patterns within the data, owing to its ability to handle nonlinear relationships and interactions among features effectively.

The model achieved a competitive score of 73485.153 when evaluated on the submission dataset in the Kaggle competition as shown in figure 8.

	lightgbm-submission_bigdata.csv Complete · 13h ago · lightgbm with all train data	73485.153	<input type="checkbox"/>
---	---	------------------	--------------------------

Figure 8: Score of the LightGBM algorithm

3.4 Neural Networks

In this project, a neural network is employed to predict latitude and longitude coordinates from a dataset of various features. The network architecture included several dense layers with ReLU activations, optimizing the model with the Adam optimizer and mean squared error loss.

One of the key advantages of neural networks is their ability to model complex, nonlinear relationships in data, which was particularly beneficial in capturing intricate patterns and interactions within the dataset. This capability led to the neural network achieving the best score for predicting the coordinates, surpassing other methods tested.

On the other hand, training neural networks can be computationally intensive and time-consuming, often requiring specialized hardware like GPUs. Despite these challenges, the neural network demonstrated remarkable success in this project, efficiently handling the high-dimensional data and yielding highly accurate predictions. The Neural Network model achieved a score of 3278.005 when evaluated on the submission dataset in the Kaggle competition as shown in figure 9.






	grafik.csv Complete · 5h ago · neural network no hyperparameter tuning no regularization	4538.144	<input type="checkbox"/>
	nn_6.csv Complete · 7h ago · neural network 6 14 digits, not regularized but hyperpara...	10087.515	<input type="checkbox"/>
	nn_4.csv Complete · 7h ago · neural network 4 hyperparameter tuned and regularized	9199.218	<input type="checkbox"/>
	nn_3.csv Complete · 15h ago · neural network submission 3 (regularized)	20309.510	<input type="checkbox"/>
	nn_2.csv Complete · 16h ago · neural network submission 2	3278.005	<input checked="" type="checkbox"/>

Figure 9: Score of all Neural Network algorithms

3.4.1 Overfitting Problem

While developing the Neural Network algorithm, after training the initial model, a validation and train loss graph was plotted as shown in Figure 10. However, it was identified that there might be an overfitting issue because the validation loss value was significantly higher than the train loss value. Various methods were tried to address this issue. Ridge regression was applied but did not resolve the problem. Hyperparameter tuning, early stopping, and reducing the dataset's row size were also attempted. However, none of these methods yielded better results than the initial model. Therefore, despite the overfitting, the best score was achieved with the initial model.

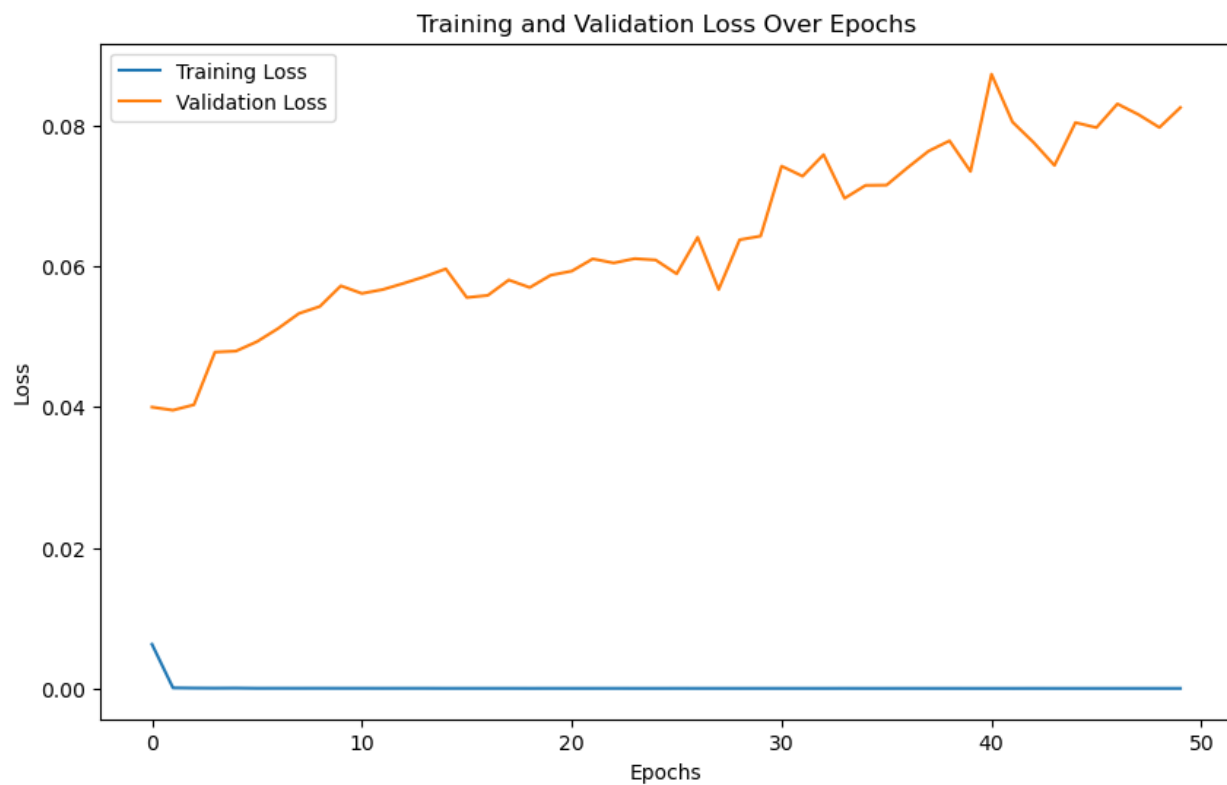


Figure 10: Training and Validation Loss over Epochs

4. Future Improvements

Exploring advanced feature engineering techniques, such as dimensionality reduction, feature selection algorithms, or embedding representations, would help extract more relevant information from the data. For example, acceleration values from IMU data would be used as a new feature as increment of displacement and used with WlsPositionEcefX/Y/Z parameters to detect current location and change to longitude and latitude form.

The performance of machine learning models is highly sensitive to hyperparameters, which are settings that control the learning process. While efforts were made to tune the hyperparameters of the models, the search space may not have been exhaustively explored. Fine-tuning hyperparameters using techniques like grid search or Bayesian optimization could potentially yield better results.

In the Neural Network model with the best score, the feature importance values were examined, and the importance of the features given for training was observed as shown in Figure 11. In future studies, a feature importance table can be generated for each feature provided in the test set, and feature selection can be performed using this table.

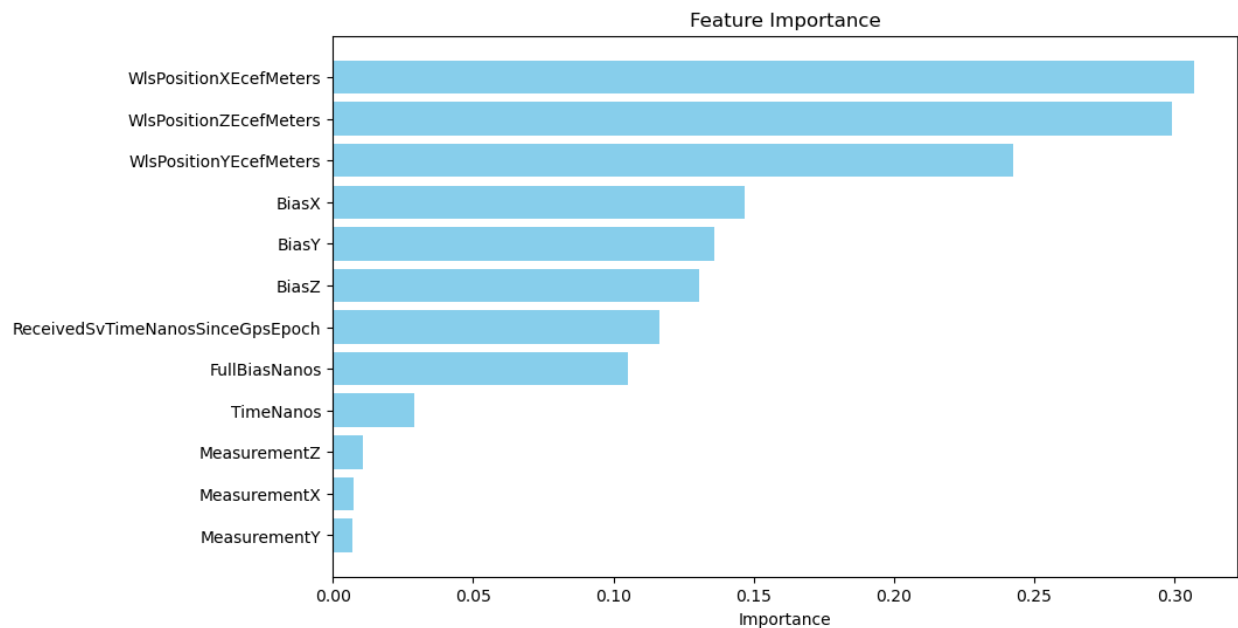


Figure 11: Feature importance of training dataset

Also, using IonosphericDelayMeters, TroposphericDelayMeters, SvVelocity [X/Y/Z]Ecef MetersPerSecond values can be helpful for mathematically calculating the location, and using it as a meaningful feature.

5. Results

All submissions made using Random Forest, LightGBM, XGBoost, and Neural Networks were uploaded to Kaggle, and their scores were compared internally. As seen in figure 12, Neural Network algorithms performed significantly better than algorithms like Random Forest,

LightGBM, and XGBoost. Due to the superior performance of the Neural Network algorithm compared to the other algorithms, efforts were focused on improving the Neural Network algorithm. Techniques such as scaling, hyperparameter tuning, and regularization were tried, and the number of features in the training data was adjusted. Surprisingly, despite the model overfitting, the Neural Network with applied regularization and hyperparameter tuning performed worse than the one without these adjustments.

Submission	Algorithm	Scaling	Regularize	Hyperparam Tuning	Number of Features	Score
Sample_deneme1	Random Forest	✗	✗	✗	21	92045
Lightgbm-bigdata	LightGBM	✗	✗	✗	19	73485
Xgboost-bigdata	XGBoost	✗	✗	✗	19	432411
nn_2	Neural Networks	✓	✗	✗	12	3278
nn_3	Neural Networks	✓	✓	✗	12	20309
nn_4	Neural Networks	✓	✓	✓	12	9199
nn_6	Neural Networks	✓	✗	✓	12	10087
grafik	Neural Networks	✓	✗	✗	9	4538

Figure 12: Score table of all algorithms

As a result, the best model was identified and its predictions were successfully submitted to the Kaggle competition. As seen in the figure 13 from the leaderboard the best model was Neural Network with no regularization and no hyperparameter tuning model with 3278.005 score.

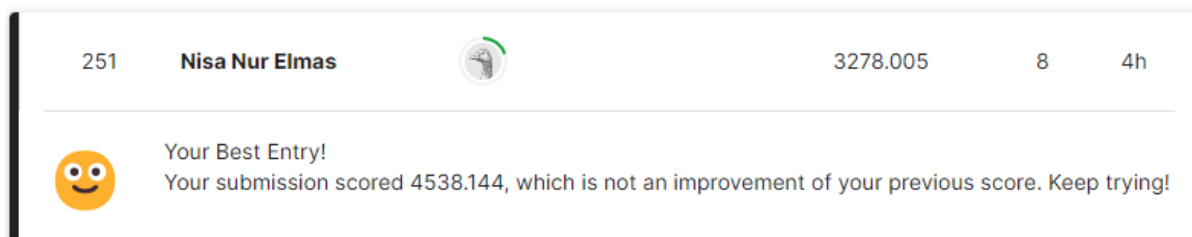


Figure 13: Screenshot from the competition leader board

Additionally, in Figure 14, the predicted and actual values for a measurement conducted in San Francisco using a Google Pixel 4 phone are shown on a map. The green dots represent the predicted values, while the blue dots represent the actual values. This allows for the observation of how the score value of 3278 corresponds on the map.

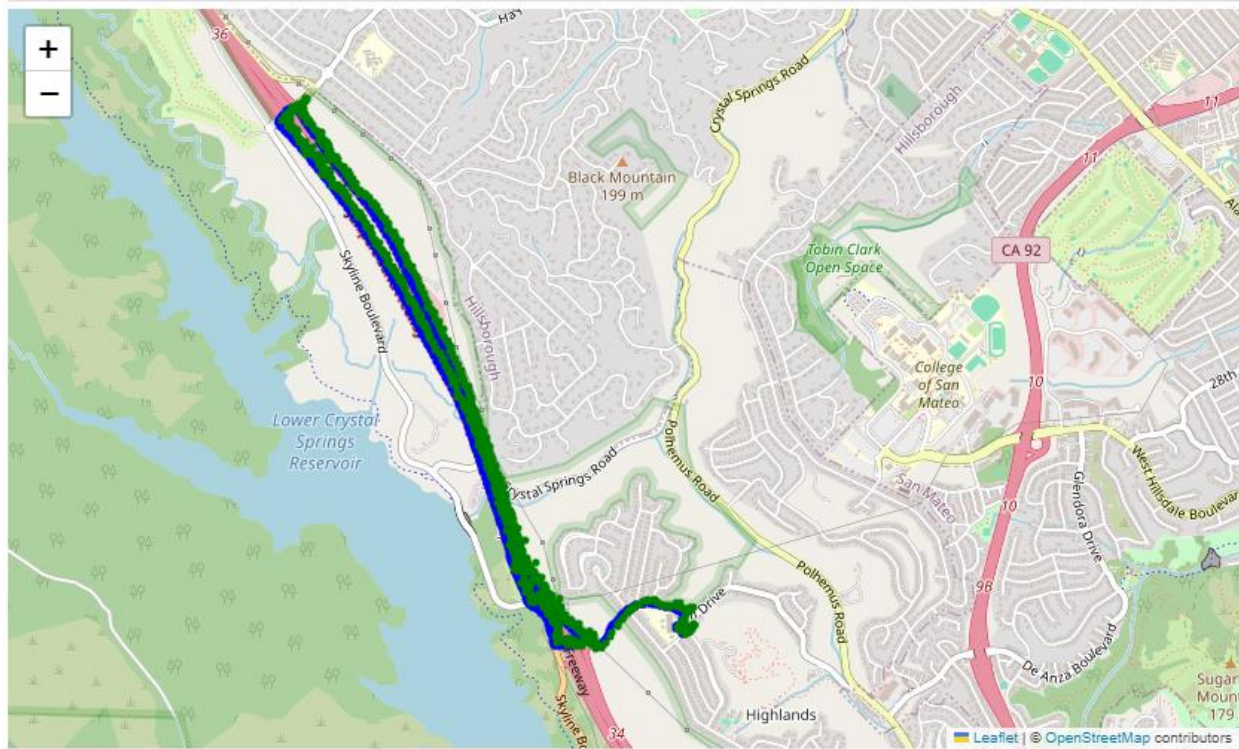


Figure 14: Map visualization of real and predicted location values

6. Conclusion

This project demonstrated the application of machine learning models to predict smartphone decimeter data. Through comprehensive data preprocessing, feature engineering, and model training, we achieved a robust predictive performance.

The project highlighted the importance of thorough data preprocessing and the benefits of using ensemble learning methods like Random Forest, LightGBM, and XGBoost for regression tasks and Neural Networks. Future work could explore advanced techniques and additional data sources to further enhance prediction accuracy.