WORCESTER POLYTECHNIC INSTITUTE

PROJECT 3 – REGRESSION

Enhanced Indoor Positioning: Comparing LSTM Neural Networks with Traditional Regression Models for Accurate RSSI-Based Localization

Jhih-ci Chao & Sheroz Shaikh

CS 548 – Knowledge Discovery & Data Mining

Prof. Roee Shraga

11/04/2023

Task 1 - BYOD

INTRODUCTION

In the burgeoning realm of Ambient Intelligence (AmI), the confluence of wearables and personal devices has paved the way for the Smart Environment (SE) revolution. Central to this revolution is the challenge of indoor localization—a domain demanding precision, adaptability, and innovation. Our project seeks to address this challenge head-on by leveraging a comprehensive dataset designed for the development and benchmarking of indoor localization systems.

The dataset at hand, retrieved from the UCI Machine Learning Repository, encapsulates a rich tapestry of multivariate, sequential, time-series data collected from smartwatches and smartphones. Key components of the dataset include:

- 153,540 instances spread over 25 features.
- Sensor data from both smartwatches and smartphones, including acceleration, magnetic field, angular position, and gyroscope readings.
- Wi-Fi signal strength measurements from various wireless access points (WAPs), with a default signal value for undetected WAPs.
- Timestamps of user location, providing sequences for movement tracking.
- A mapping file that correlates unique place identifiers to local X-Y coordinates.

The significance of this dataset cannot be overstated. Indoor localization stands as a cornerstone of SE, with applications ranging from navigation assistance for the visually impaired to asset tracking in vast industrial spaces. The meticulous assembly of this dataset addresses a critical gap—a standardized framework that allows for the comparison and validation of different localization algorithms.

Our motivation for choosing this dataset is twofold. First, it provides a realistic representation of user-device interaction within an indoor setting, and second, it offers a fertile ground for testing the robustness of localization solutions in the face of real-world complexities.

POTENTIAL IMPACT & CHALLENGES

The implications of refining indoor localization are profound. From enhancing personal navigation to optimizing space utilization in smart buildings, the applications are as diverse as they are impactful. The success of this project could lead to breakthroughs in user experience and operational efficiency, signaling a leap forward in the integration of AmI into our lives.

Yet, the path ahead is strewn with obstacles. Signal fluctuation, sensor discrepancies, and the ever-changing indoor dynamics pose significant hurdles. Crafting solutions that are both precise and swift enough for real-time application demands a delicate balance between algorithmic complexity and computational feasibility.

Embarking on this project, we stand at the frontier of innovation, ready to tackle the intricacies of indoor localization. Our efforts are set to contribute a pivotal chapter to the narrative of smart environment evolution.

Task 2 – EDA [10 points]

1. INITIAL STRUCTURE ANALYSIS

The dataset consists of multiple CSV files, each containing different types of data relevant to indoor localization via sensors and Wi-Fi signal strength. Here's a detailed breakdown:

Points Mapping (*PointsMapping.ods*): Maps physical locations in a space to a coordinate system.

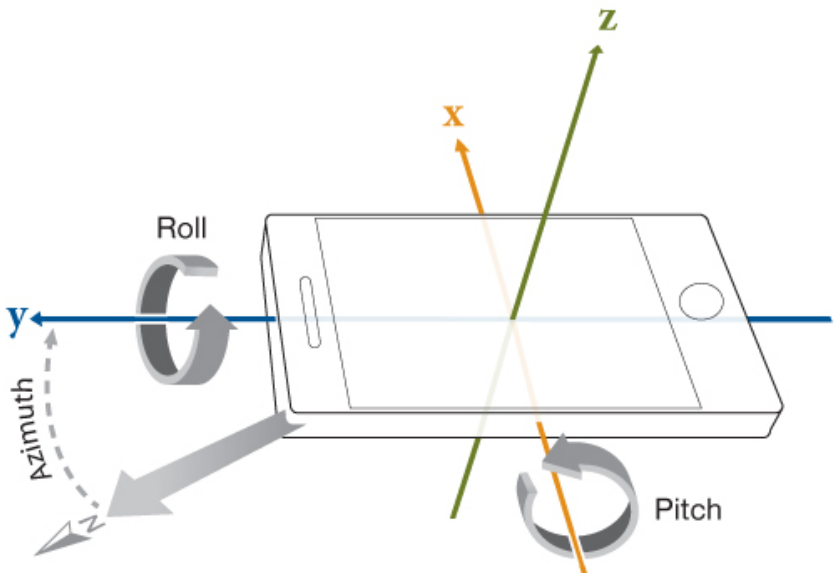
- `ID`: Unique identifier for a specific location within the indoor map.
- `X`: The x-coordinate of the location in the local coordinate system.
- `Y`: The y-coordinate of the location in the local coordinate system.

Timestamp ID (*measure1(2)_timestamp_id.csv*): Contain timestamps marking the user's arrival and departure times at specific locations, associated with Place IDs.

- `arrival`: Unix timestamp indicating when the user arrived at a location.
- `departure`: Unix timestamp indicating when the user left the location.
- `place_id`: The unique identifier of the place corresponding to the ID in the Points Mapping file, ranging from 0 to 324.

Smartphone Sensors (*measure1(2)_smartphone_sens.csv*): Provide sensor data collected from the smartphone, with each row corresponding to a sensor reading at a specific timestamp.

- `timestamp`: Unix timestamp of when the sensor reading was taken.
- `AccelerationX/Y/Z`: Acceleration readings along the X, Y, and Z axes, respectively.
- `MagneticFieldX/Y/Z`: Magnetic field strength along the X, Y, and Z axes, respectively.
- `Z-AxisAngle(Azimuth)`, `X-AxisAngle(Pitch)`, `Y-AxisAngle(Roll)`: Angular position of the device in degrees.
- `GyroX/Y/Z`: Gyroscope readings along the X, Y, and Z axes, respectively, indicating the device's rotational movement.



Smartwatch Sensors (*measure1(2)_smarwatcg_sens.csv*): Similar to the smartphone sensor data, these files record sensor data from a smartwatch.

- The columns are identical to those of the smartphone sensor data above.

Smartphone Wi-Fi (*measure1(2)_smartphone_wifi.csv*): These files detail the Wi-Fi signal strength readings from various wireless access points (WAPs) at different locations.

- The first column represents the PlaceId, which corresponds to the ID in the Points Mapping file.
- The subsequent 127 columns represent the Received Signal Strength Indicator (RSSI) levels for different WAPs. These are typically in dBm (decibels relative to a milliwatt), with -100 dBm indicating no detection of a WAP.

Each CSV file beginning with "`measure1`" or "`measure2`" indicates a separate measurement campaign, conducted at different times or under different conditions to provide a diverse set of data for robust model training and testing.

Additional Insights:

- The acceleration components (AccelerationX, AccelerationY, and AccelerationZ) may influence the orientation angles (Azimuth, Pitch, Roll) and, consequently, the X and Y coordinates. For example, changes in acceleration may result in changes in pitch or roll angles, affecting the object's position.

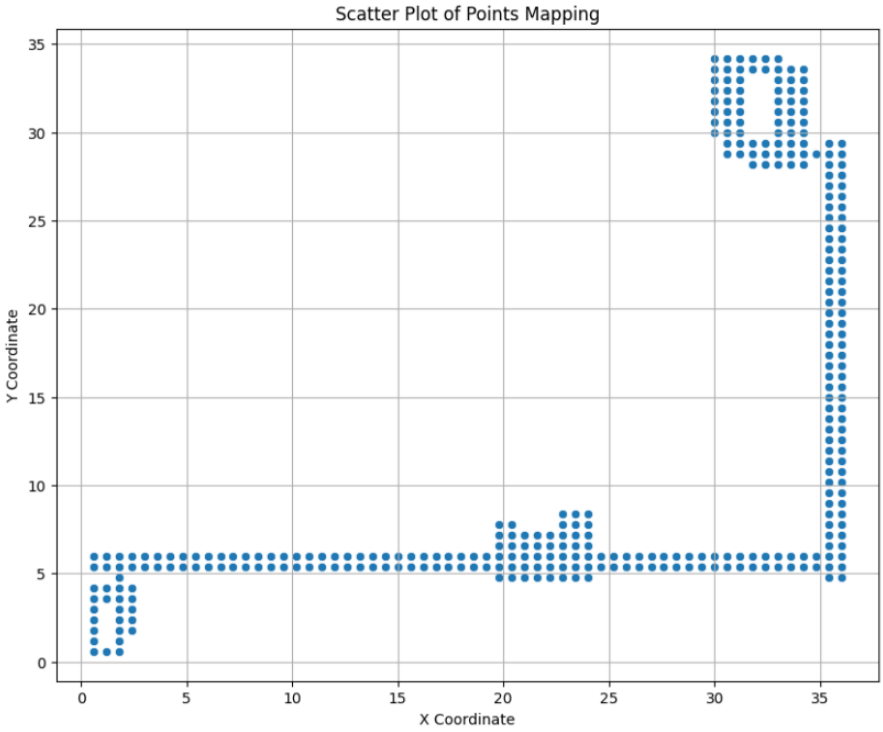
- The magnetic field components (MagneticFieldX, MagneticFieldY, MagneticFieldZ) may also influence the orientation angles, particularly the azimuth angle, which depends on the Earth's magnetic field. This, in turn, can affect the X and Y coordinates.

2. POINTS MAPPING ANALYSIS

To perform an exploratory data analysis (EDA) on the points mapping data, we'll start by examining the structure and distribution of the location points within the given indoor space. We'll look for any anomalies or patterns in the distribution of points, such as clustering or gaps that could indicate areas of interest or potential data collection issues.te

The statistical summary provides the following insights into the points mapping data:

- There are 325 unique location points (IDs).
- The X coordinates range from 0.6 to 36, with a mean of approximately 24.76.
- The Y coordinates range from 0.6 to 34.2, with a mean of approximately 14.02.
- The standard deviations for X and Y are around 11.70 and 10.97, respectively, suggesting a wide spread of points.

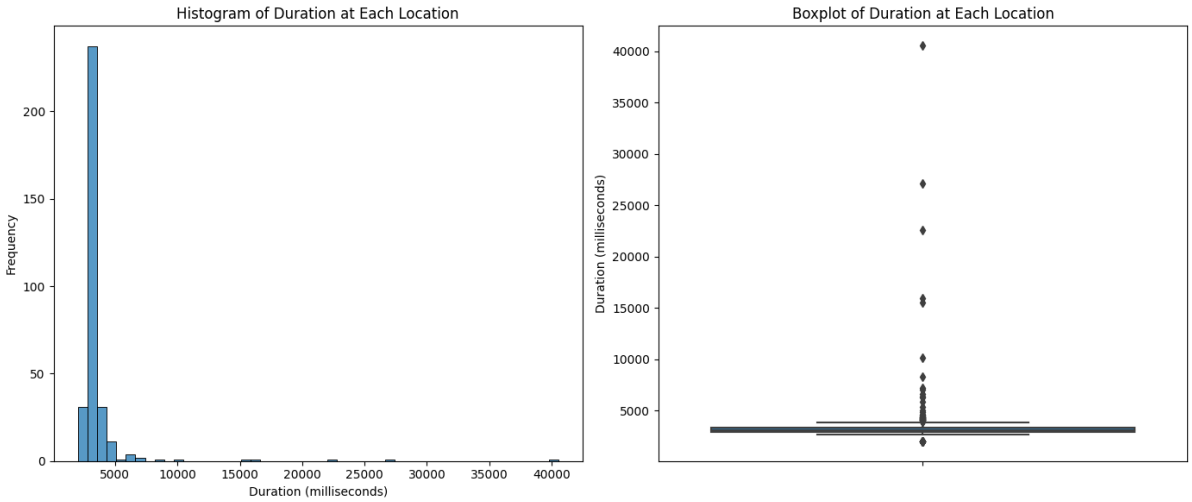


The scatter plot above visualizes the distribution of these points in the local coordinate system, showing how they are spread out. There don't appear to be any anomalies at first glance, as the points seem uniformly distributed without any obvious outliers. However, there's a visible pattern of higher density in certain regions, which could correlate to important areas within the indoor environment, such as entry points, rooms, or corridors.

While mapping the place IDs with the timestamp place IDs, we noticed that there is a place ID of 0 at row 316 of both timestamp_id datasets, which is not supposed to be a valid place ID. Since it appeared in the middle of the dataset instead of at the beginning or the end, we simply removed that row assuming it is an error.

3. TIMESTAMP ID ANALYSIS

In this section, we focus on the timestamp ID data after being correctly loaded with the proper headers (arrival, departure, and place_id). We will first look at the distribution of the time spent at each location by calculating the duration of stay (departure – arrival) and also check the frequency of visits to different place_ids. Additionally, we'll plot the durations to visually inspect any outliers or anomalies. Let's proceed with these steps.



The duration data reveals the following:

- The average duration of stay at a location is around 3559 milliseconds, with a standard deviation of approximately 2964 milliseconds.
- The minimum duration recorded is 2000 milliseconds, and the maximum is quite high at 40536 milliseconds, suggesting some longer stays or potential outliers.
- The majority of durations fall within a narrower range, as indicated by the 25th to 75th percentile values (2895 to 3291 milliseconds).

The histogram shows the distribution of durations, with a clear concentration in the lower range, while the boxplot indicates the presence of outliers on the higher end.

Regarding the frequency of visits to different place_ids, it appears that most locations are visited once, with a few locations being visited more frequently (up to four times for the most visited place_id). This data could be crucial for understanding traffic patterns and user behaviors within the indoor environment. The outliers in duration might be areas where users tend to stay longer, which could be points of interest or waiting areas.

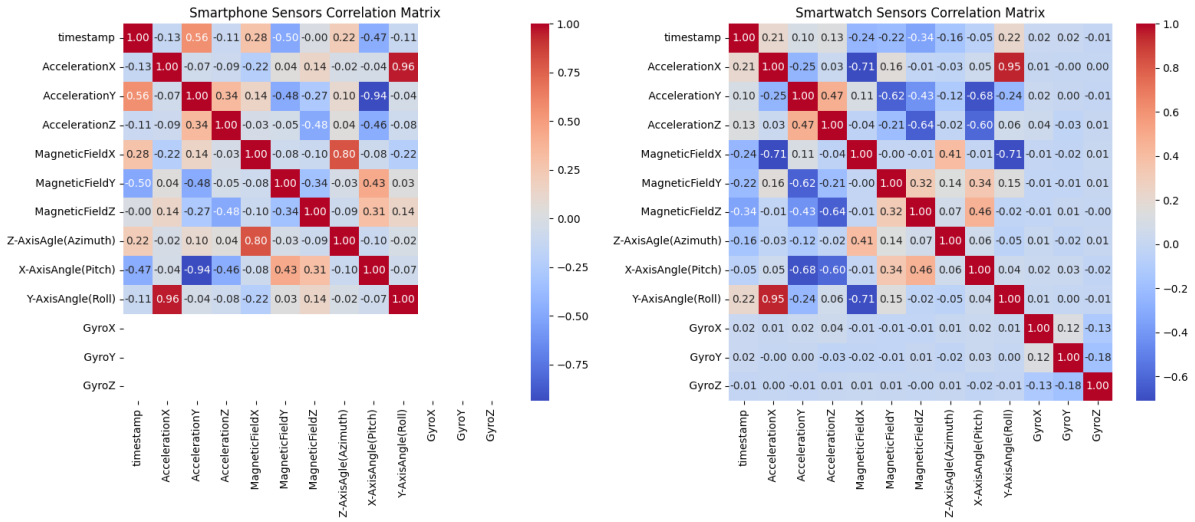
In our exploration of the Timestamp ID data, we have quantified the durations of stay across various indoor locations and observed a range of visit frequencies. While extended stays have been noted, in the absence of contextual information about the indoor environment, we have refrained from conjecture about the reasons behind these patterns. Recognizing that such variabilities in human movement are typical in real-world settings, we consider this variance a natural aspect of the dataset that our localization models must accommodate. With this understanding, we now turn our attention to the sensor and Wi-Fi data, which hold the key to developing a robust and accurate indoor localization system.

4. SENSOR DATA ANALYSIS

We will first start with some Statistical Summary:

- The smartphone sensor data has 18,354 entries. Acceleration values vary around a mean close to zero for X, and positive values for Y and Z, indicating gravity's effect. Magnetic field readings and angles vary widely.
- The smartwatch sensor data has 58,374 entries. Similar to the smartphone, acceleration shows gravity's influence. Gyroscope readings are floats with means close to zero, indicating stationary or smooth movement.

We will now proceed with some correlation and distribution analysis with the two types of sensor data.



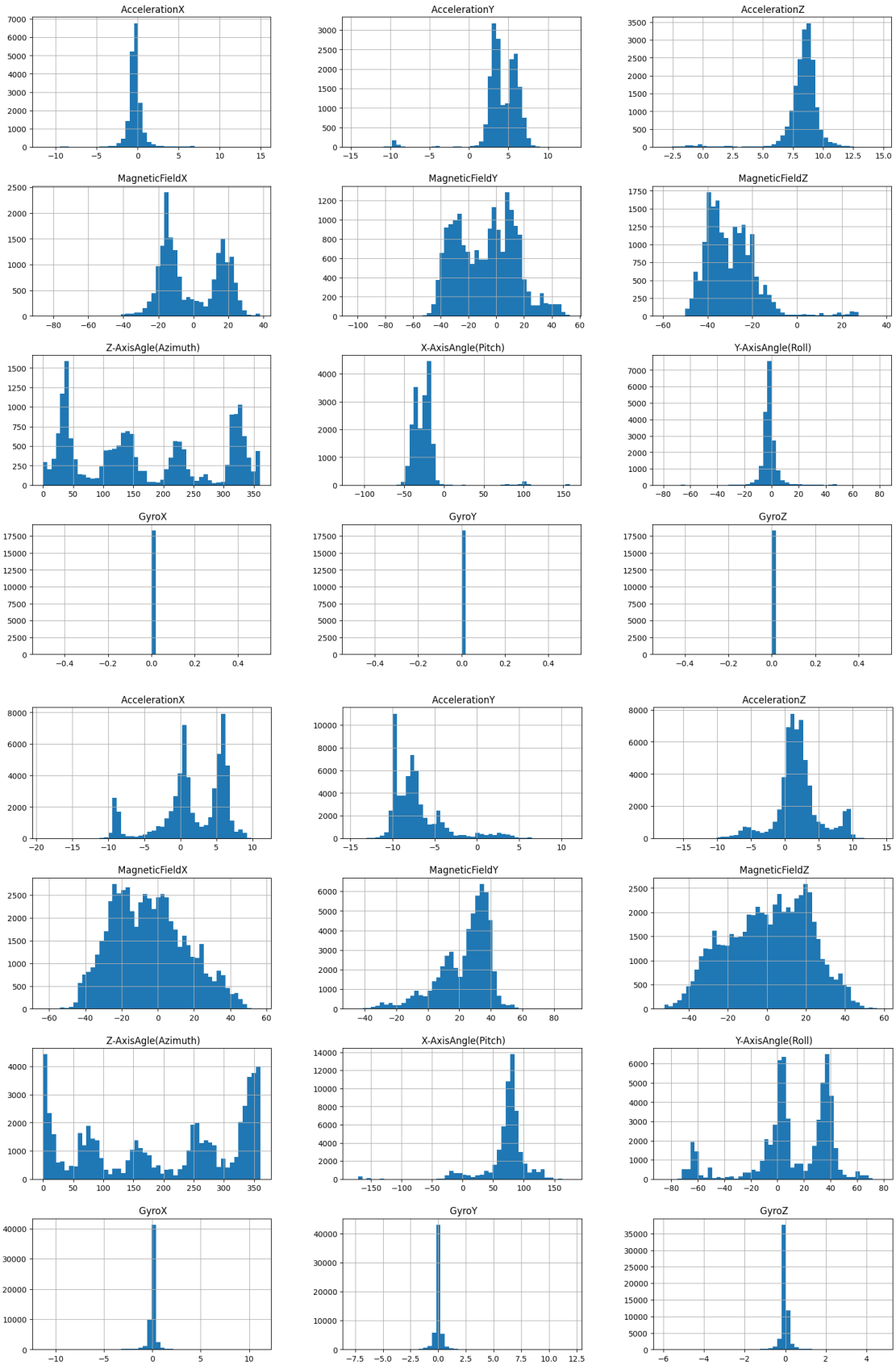
The correlation heatmaps for both smartphone and smartwatch sensors are displayed. Correlation coefficients range from -1 to 1, where 1 indicates a perfect positive correlation, -1 a perfect negative correlation, and values close to 0 no correlation.

Let's summarize the key correlations:

- Acceleration axes show some degree of correlation, expected due to the movement in 3D space.
- Magnetic field readings show varying degrees of correlation; they can be influenced by the orientation of the device relative to the Earth's magnetic field.
- Gyroscope readings (GyroX, GyroY, GyroZ) show low correlation with other sensors, which is typical since they measure rotation, not linear acceleration or magnetic fields.

Below are the distribution histograms (smartphones, smartwatches). Key observations include:

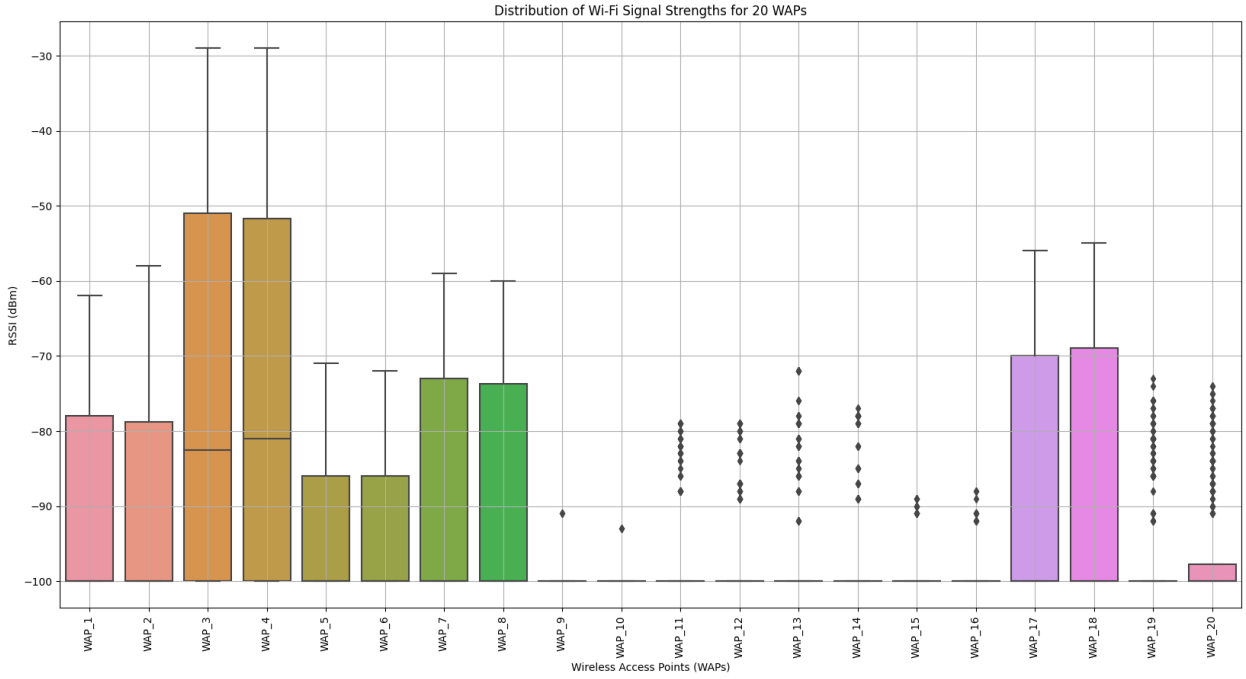
- Acceleration readings typically show a Gaussian-like distribution, centered around the gravity value for the Z-axis.
- Magnetic field readings are spread out, which could be due to various orientations of the device.
- Gyroscope readings in the smartphone data are all zero, which is likely an anomaly or a result of the device being stationary. For the smartwatch, the readings are more spread out, indicating movement.


AccelerationX
AccelerationY
AccelerationZ
MagneticFieldX
MagneticFieldY
MagneticFieldZ
Z-AxisAgle(Azimuth)
X-AxisAngle(Pitch)
Y-AxisAngle(Roll)
GyroX
GyroY
GyroZ
AccelerationX
AccelerationY
AccelerationZ
MagneticFieldX
MagneticFieldY
MagneticFieldZ
Z-AxisAgle(Azimuth)
X-AxisAngle(Pitch)
Y-AxisAngle(Roll)
GyroX
GyroY
GyroZ

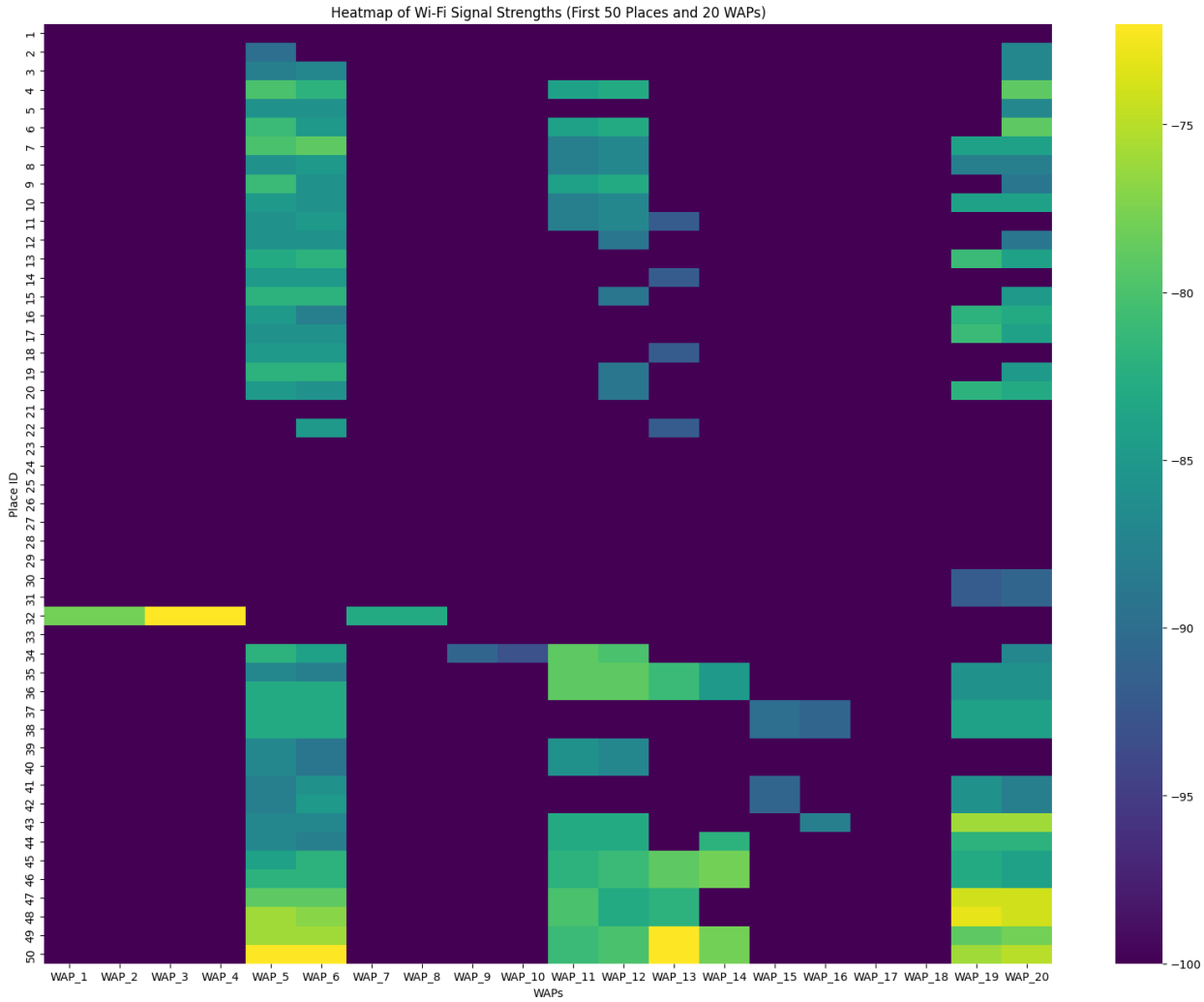
5. Wi-Fi DATA ANALYSIS

The Wi-Fi data consists of rows representing different Place IDs with 127 columns of RSSI values for various Wireless Access Points (WAPs). A value of -100 dBm indicates a WAP that was not detected.

Statistics Summary: The RSSI values for the WAPs show a lot of variability. Many WAPs have a 25th percentile value of -100 dBm, indicating a high prevalence of non-detections. The mean values suggest that for most WAPs, the signal strength is weak (closer to -100 dBm).



The boxplot for a subset of 20 WAPs shows a wide range of RSSI values, with many outliers. This suggests that signal strength can vary significantly even within the same WAP.



The heatmap for the first 50 places and 20 WAPs shows the variation in signal strength across different places. Darker areas (closer to -100 dBm) indicate weaker or no signals, while lighter areas indicate stronger signals. There's a clear pattern of signal presence and absence across the Place IDs, which could be critical for localization algorithms.

Task 3 – Problem Definition [5 points]

1. PROBLEM STATEMENT AND OBSERVATIONS

Indoor localization systems are fundamental components of Smart Environments, with their efficiency hinged on the precise interpretation of signal strengths from various WAPs. However, the EDA of the provided dataset reveals a complex landscape where RSSI values exhibit marked fluctuations, indicating a highly dynamic signal environment.

These fluctuations, along with numerous non-detections and signal outliers, suggest that the Wi-Fi signal strength is affected by factors such as moving objects, the opening and closing of doors, and the reconfiguration of indoor spaces. The current localization models, which often assume static signal conditions, struggle to accommodate this variability, leading to reduced accuracy in real-time applications.

The observation of temporal patterns in the signal strength across different WAPs further underscores the intricate nature of the problem. There is a clear pattern of signal variation over time, with certain areas exhibiting stronger signals than others at different times. This variability is a significant source of error in indoor localization efforts and points to the need for a predictive model that can intelligently adapt to these temporal changes.

2. MOTIVATION

The motivation for tackling the dynamic nature of indoor signal propagation is to enhance the adaptability and accuracy of localization systems, crucial for the AmI vision where environments are not just aware but also predictive of human presence and behavior. There's a pressing need for models that can forecast Wi-Fi signal variations, driven by the demand for real-time, reliable data in applications ranging from personal navigation to emergency response. By leveraging the rich dataset to predict signal strength over time, the goal is to create robust localization systems that are effective amidst the variable indoor settings, thereby supporting the practical and operational demands of SE.

3. SIGNIFICANCE

The development of a time-dependent signal strength prediction model carries immense significance for the future of SE. It is a key step towards achieving environments that are genuinely responsive to their inhabitants, thereby elevating the user experience to new heights. For instance, in scenarios like emergency evacuation or navigation for the visually impaired, the ability to rely on accurate, real-time localization is not just beneficial but potentially life-saving.

Moreover, this model holds the promise of optimizing space utilization and operational efficiencies in commercial and industrial settings. By accurately predicting signal strength dynamics, it can support a wide range of applications, including targeted advertising, energy management, and intelligent security surveillance. Thus, this work is not only a technical endeavor but also a foundational element for the practical realization of the AmI vision, with far-reaching implications for smart building technology and beyond.

Task 4 – Preprocessing [15 points]

I. DATA MERGING

For simplicity, we first made a comprehensive compilation of sensor and Wi-Fi signal strength data, merged with location coordinates. Here's a summary of the observed structure and changes made:

1. **Sensor Data Integration:** We consolidated sensor readings from both smartwatches and smartphones to capture a complete picture of user-device interaction. This includes acceleration, magnetic field, and orientation data, which are critical for detecting user movements and orientations within the indoor space. Merging these modalities provides a richer feature set for the localization models.

2. **Wi-Fi Signal Strength Inclusion:** The dataset encompasses an extensive array of Wi-Fi signal strength features, ranging from `RSSI_1` to `RSSI_127`. We retained the full spectrum of Wi-Fi data to maximize the information available for detecting signal patterns, which are vital for Wi-Fi fingerprinting approaches in indoor localization.

3. **Location Mapping:** Each record in the dataset is associated with a `place_id` that corresponds to a specific location within the monitored environment. The inclusion of X and Y coordinates allows for a direct mapping to physical space, providing ground truth for supervised learning and enabling the models to predict actual locations as output.

II. FEATURE GENERATION

Time Features: By converting the timestamp to datetime and extracting the hour, minute, day of the week, and month, we are capturing the time-related fluctuations in signal strength. These features can help the model learn patterns that occur at specific times or days.

Fourier Transforms: Applying the Fourier transform to the sensor data is a way of identifying frequency components within the signal. This could be useful for capturing periodic patterns in the sensor data which might correlate with the RSSI fluctuations.

Radians and Trigonometric Features: Converting angles to radians and then creating sine and cosine features can help in capturing the orientation more effectively. This is especially important for models that require continuous input features like LSTMs.

RSSI Statistics: Computing the mean and variance of RSSI values across all WAPs can provide general statistics that summarize the overall signal strength and variability at each observation, which is critical in understanding signal dynamics.

Interaction Features: Multiplying different sensor readings aims to capture interaction effects between them. This can be particularly useful when the relationship between sensor inputs is not simply additive.

Distance to Origin: Calculating the Euclidean distance to a reference point (origin) could be relevant if the position within the space correlates with signal strength patterns.

Numerical Features List: Keeping track of which features are numerical is a good practice, especially when it comes to scaling or normalizing the data before feeding it into machine learning models.

Each of these feature sets attempts to capture different aspects of the data that could potentially influence RSSI values:

- **Time features** could help understand how signal strength varies throughout the day or week.
- **Fourier transforms** of sensor data could uncover hidden periodicities.
- **Trigonometric features** help model the cyclical nature of angles.
- **RSSI statistics** provide aggregate information on signal strength.
- **Interaction features** attempt to model the combined effect of different sensors on the signal.
- **Distance to origin** might show a relationship between a device's location in space and the signal strength.

These feature engineering steps show a thoughtful approach to modeling a complex problem like indoor localization using time-series data aligned with the problem statement's emphasis on the dynamic and temporal nature of the signal environment, suggesting that the feature generation steps are appropriate for the task at hand.

III. FEATURE SELECTION

1. **Data Sampling:**

   To manage computational resources, a 10% random sample of the dataset was taken, using sample(frac=0.1, random_state=1). This ensures a representative subset of the data while maintaining computational efficiency.

2. **RSSI Column Selection:**

   A subset of 10 RSSI columns was randomly selected from the dataset for the analysis. This approach allows focusing on a diverse yet manageable set of signal strength features.

3. **Feature Importance Accumulation:**

   A dictionary, feature_importance_sum, was initialized to accumulate the importances of features across all selected RSSI columns.

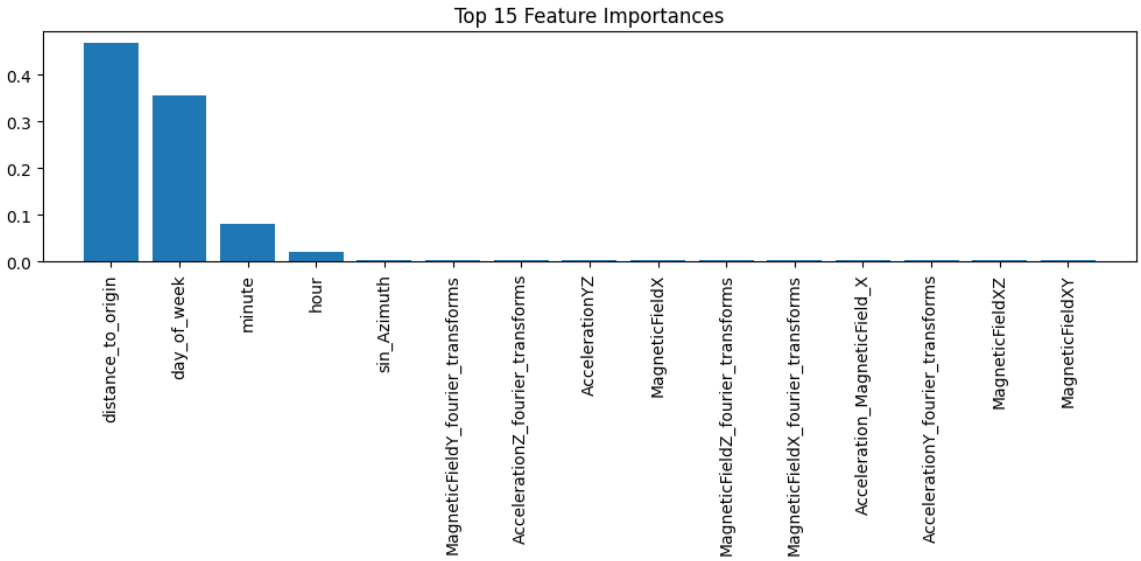
4. **Iterative Analysis for Each RSSI Column:**

   For each RSSI column, the following process was executed:

   - **Data Preparation**: Non-predictor columns, including other RSSI columns, timestamps, wifi_tag, and location identifiers (X, Y, place_id), were excluded to isolate predictor variables.
   - **Train-Test Split**: The dataset was split into training and testing sets (80-20 split) using train_test_split to validate model performance on unseen data.
   - **Random Forest Model**: A Random Forest Regressor with 100 trees (n_estimators=100) was trained, leveraging its capability to handle numerous features and capture complex relationships.
   - **Importance Aggregation**: Feature importances were aggregated across all analyzed RSSI columns to gain a comprehensive understanding of each feature's relevance.

5. **Average Importance Calculation:**

The average importance of each feature was calculated by dividing the cumulative importance by the number of RSSI columns analyzed. This step provides a balanced assessment of each feature's influence. The feature ranking and visualization is as follows:



Features were ranked based on their average importance and visualized to highlight the top 15 features. The visualization, created using Matplotlib, offers a clear, concise view of the most critical features for predicting RSSI values in indoor environments.

The analysis revealed the top features contributing to the accuracy of RSSI predictions, with distance to origin and day of week being the most important information. By focusing on the most influential features, the efficiency and accuracy of localization algorithms may be significantly enhanced.

IV. FEATURE NORMALIZATION

1. **Selection of Features for Normalization**: Features selected for normalization exclude timestamp, place_id, wifi_tag, X/Y, as these are either categorical or location-specific identifiers that do not require scaling. The selection ran merged_data_sorted.columns.difference([...]), ensuring that only relevant numerical features are included for normalization.

2. **Application of Min-Max Scaling**: The MinMaxScaler from the sklearn.preprocessing module is employed, with a feature range set to (0, 1). This scaler transforms each feature individually, scaling them to a specified range, here between 0 and 1. The transformation, fit_transform, is applied to the selected features of the merged_data_sorted DataFrame, ensuring that all relevant numerical data are scaled uniformly.

V. DATA SPLITTING

The data is split into training (80%) and testing (20%) sets using train_test_split for validation.

Task 5 – Model Selection, Training, and Optimization [30 points]

I. Traditional Model Selection and Training

In our approach to developing an indoor localization system, we've elected to utilize a dual-model strategy, encompassing both traditional tabular regression models and LSTM, a time-series specific model. This decision stems from our dataset's diverse nature, which includes both static features (such as RSSI readings and sensor data) and dynamic, sequential data.

Traditional regression models are adept at capturing relationships within static datasets, while time-series models, like LSTM (Long Short-Term Memory) networks, excel in understanding and predicting patterns in sequential data. By leveraging the strengths of both types, we aim to create a comprehensive and robust system capable of high accuracy and adaptability in indoor localization. This strategy allows us to exploit the full potential of our rich dataset, ensuring that we can effectively address the complexities inherent in indoor localization tasks.

Each model was trained on the preprocessed training data, which underwent data cleaning, transformation, normalization, and feature scaling. The training process involved fitting the models to learn the underlying patterns in the data, using RSSI values as target variables.

1. **Linear Regression**: A baseline model used for its simplicity and interpretability.
2. **Ridge Regression**: Introduced L2 regularization to handle multicollinearity.
3. **Lasso Regression**: Applied L1 regularization to induce sparsity in feature selection.
4. **Random Forest Regression**: An ensemble method leveraging decision trees for robust predictions.
5. **Gradient Boosting Regression**: Boosted decision trees to improve predictive performance.
6. **Elastic Net Regression**: Elastic Net combines L1 (Lasso) and L2 (Ridge) regularization, striking a balance between feature selection and handling multicollinearity. It addresses the limitations of Lasso by introducing a hyperparameter (l1_ratio) to control the mix of L1 and L2 penalties.
7. **Bayesian Ridge Regression**: It is a Bayesian counterpart to classical linear regression. It incorporates a probabilistic framework, allowing for the estimation of model parameters with uncertainty. It introduces hyperparameters (alpha_1, alpha_2, lambda_1, lambda_2) that play a role in the prior distribution, providing more flexibility.
8. **Decision Tree Regression**: Utilizes a tree-like model structure to make decisions based on feature splits. It is non-parametric and handles complex relationships in the data. The model partitions the feature space, assigning a constant value to each partition, making it a versatile choice for both simple and intricate patterns.
9. **K-Nearest Neighbors (KNN) Regression**: Predicts the target variable based on the average (or weighted average) of the k-nearest neighbors in the feature space. It is a non-parametric, instance-based learning algorithm. The choice of 'k' and the distance metric are crucial parameters influencing model performance.
10. **Huber Regressor**: Is a robust regression algorithm that combines the best of mean squared error and mean absolute error. It introduces an epsilon parameter to distinguish between inliers and outliers, making it less sensitive to extreme values. Huber loss provides a compromise between the sensitivity of MSE and the robustness of MAE.

Data Preparation and Feature Selection

- **Features Used**: Acceleration (X, Y, Z), Magnetic Field (X, Y, Z), Axis Angles (Azimuth, Pitch, Roll), and several RSSI values.
- **Preprocessing**: StandardScaler for acceleration and angle features; MinMaxScaler for magnetic field features.

Model Development

- **Regression Models**: Linear, Ridge, Lasso, ElasticNet, Bayesian Ridge, Decision Tree, Random Forest, Gradient Boosting, KNN, and Huber Regressors.
- **Training Approach**: Each RSSI value (RSSI_2 to RSSI_10) is predicted separately, employing various regression models.
- **Performance Metrics**: MSE, RMSE, NRMSE, R^2, MAE, MAPE, MBD, MASE, EVS.

Model Optimization

To enhance model performance, we employed several optimization techniques:

1. **Feature Engineering**: We considered feature importance from tree-based models to select relevant features, eliminating noise and improving model efficiency.
2. **Cross-Validation**: Utilized cross-validation to assess the models' performance across different subsets of the training data, ensuring robustness and generalization.
3. **Hyperparameter Tuning**: Employed random grid search to find optimal hyperparameters for selected models, fine-tuning them for improved performance.

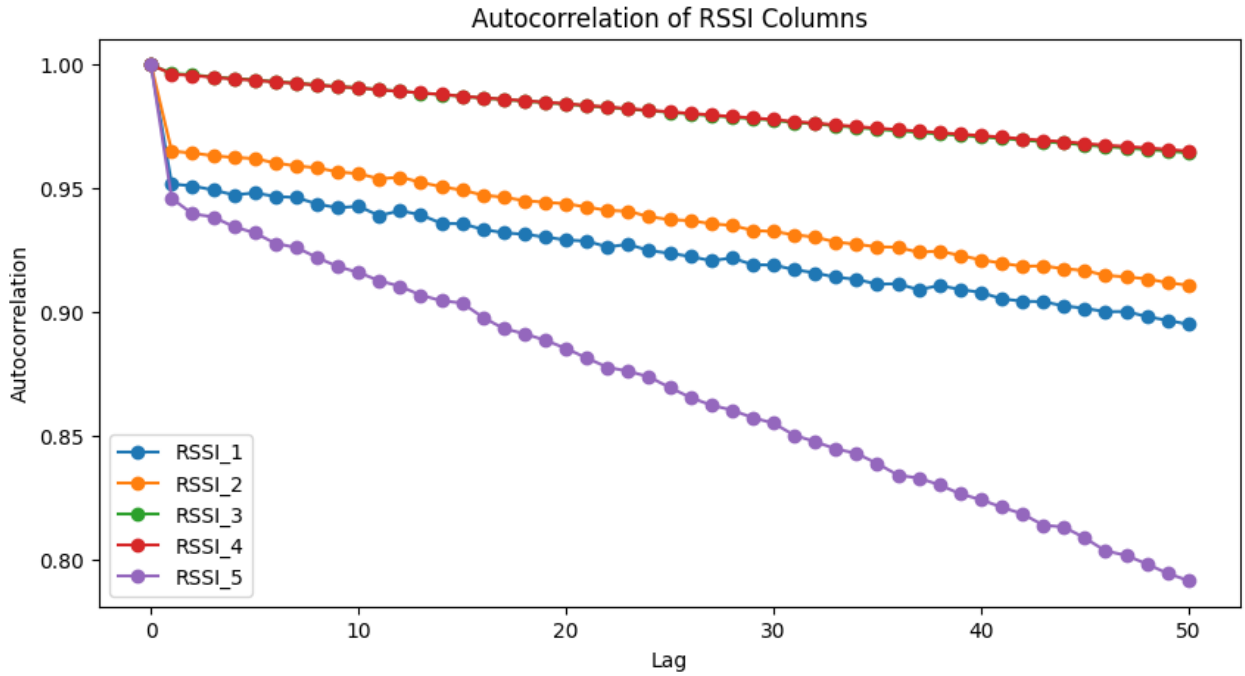
We compared the performance of each model before and after optimization. This involved evaluating their predictive power, robustness, and efficiency in capturing the underlying patterns in the dataset.

Model selection, training, and optimization are crucial steps in building an effective regression model. By considering various algorithms and optimization techniques, we aim to identify the model that best captures the relationships within our dataset. This process enhances the model's ability to make near to accurate predictions on unseen data, contributing to the success of our RSSI prediction task. The comparative analysis will guide us in selecting the most suitable model for our specific use case.

II. LSTM MODEL FOR TIME-SERIES

Autocorrelation Analysis

To understand the temporal relationship within the RSSI data by examining the autocorrelation at different lags, we calculated the autocorrelation for the first five RSSI columns up to 50 lags.



The initial drop from 1 to around 0.95~0.99 suggests that while there is a slight decrease in correlation, the RSSI values still maintain a relatively strong correlation with their immediate past. This indicates a notable, but not drastic, change in the temporal relationship from one lag to the next.

The slow decline in autocorrelation values over subsequent lags, and the maintenance of relatively high values even after 50 lags, particularly in RSSI_3 and RSSI_4, points to the presence of strong long-term dependencies in the data.

Model Configuration

- Sequence Length: 12

  This length captures sufficient historical context to predict future values.

- Prediction Length: 5

  The model predicts values for the next 5 time steps, providing short-term future insights.

- Feature Columns:'distance_to_origin', 'day_of_week', 'minute', and 'hour'.

  These were selected based on their importance in influencing the target variable.

Sequence Generation

The function `create_sequences` processes the data into sequences of the specified length. It uses numpy arrays for efficient computation and pre-allocates memory for X (input features) and y (target values) to enhance performance.

LSTM Model

- **Architecture:** The LSTM model comprises two LSTM layers with 50 units each. The first LSTM layer returns sequences to maintain temporal information, while the second compresses these sequences into a single context vector. A dense layer reshapes the output to match the prediction length and feature dimensions.
- **Compilation:** The model uses the Adam optimizer and mean squared error as the loss function, suitable for regression tasks.
- **Training:** Training occurs over 8 epochs with a batch size of 64. Validation data is used to monitor the model's performance on unseen data.

The LSTM model is designed to capture temporal dependencies and predict future values based on historical data. The chosen features, sequence length, and prediction length are aligned with the goal of accurately forecasting short-term future values. The training process is structured to optimize the model's ability to generalize well to new data.

Model Optimization

The optimization in this section took a slightly different approach. We start with a rough estimate of sequence and prediction lengths using the autocorrelation analysis of the first five RSSI columns, and test a few different values to find the best model configuration.

Task 6 – Model Evaluation [30 points]

Evaluation Metrics

In the final phase of the project, the trained machine learning models are evaluated using a range of metrics. This step is crucial for understanding the efficacy of each model in a multi-faceted manner, which in turn informs the decision-making process for model deployment or further optimization. For comprehensive evaluation, the metrics selected are shown below:

1. **Mean Squared Error (MSE):** MSE measures the average of the squared differences between the actual and predicted values. It provides a measure of how well the model's predictions match the actual data. Lower MSE values indicate better model performance.

2. **Root Mean Squared Error (RMSE):** RMSE is the square root of the MSE. It represents the average magnitude of the errors in the same units as the target variable. RMSE is more interpretable than MSE and is commonly used for regression model evaluation.

3. **Normalized Root Mean Squared Error (NRMSE):** NRMSE is the RMSE (Root Mean Squared Error) normalized by the range of the target variable. It provides a standardized measure of the model's prediction error, making it useful when dealing with data on different scales.

4. **R-squared (R2) Score:** R2 measures the proportion of the variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating a better fit. An R2 score of 1 means the model perfectly predicts the target variable. Negative values in the test indicate a bad model.

5. **Mean Absolute Error (MAE):** MAE calculates the average absolute differences between the actual and predicted values. It provides a measure of the average magnitude of errors in the target variable.

6. **Mean Absolute Percentage Error (MAPE):** MAPE measures the average percentage difference between the actual and predicted values. It provides a relative measure of the model's accuracy and is particularly useful when we want to understand the prediction accuracy in terms of percentages.

7. **Mean Bias Deviation (MBD):** MBD quantifies the average difference between the actual and predicted values. It helps assess whether the model systematically overestimates or underestimates the target variable.

8. **Mean Absolute Scaled Error (MASE):** MASE is used to evaluate the performance of time series forecasting models. It measures the relative accuracy of the model's predictions compared to a simple, naïve forecasting method.

9. **Explained Variance Score (EVS):** The explained variance score measures the proportion of the variance in the target variable that the model explains. It is a variant of R2 that takes into account the mean squared error of the model's predictions.

Traditional Model Results

The comprehensive results of traditional models are shown in optimized_predictions - Google Sheets. Below is a high-level overview of the insights gathered from the report:

1. **Performance Comparison**: We observed that the Random Forest and Gradient Boosting Regression models outperformed linear models (Linear, Ridge, Lasso) in terms of MSE, RMSE, and R2 Score.
2. **Impact of Optimization**: Optimization techniques, including feature engineering and hyperparameter tuning, positively influenced model performance. The optimized models exhibited improved accuracy and robustness.
3. **Model Robustness**: Random Forest and Gradient Boosting demonstrated better resilience to variations in the dataset, contributing to their superior performance.
4. **Feature Importance**: Models with feature importance considerations showed enhanced predictive power by focusing on relevant features.

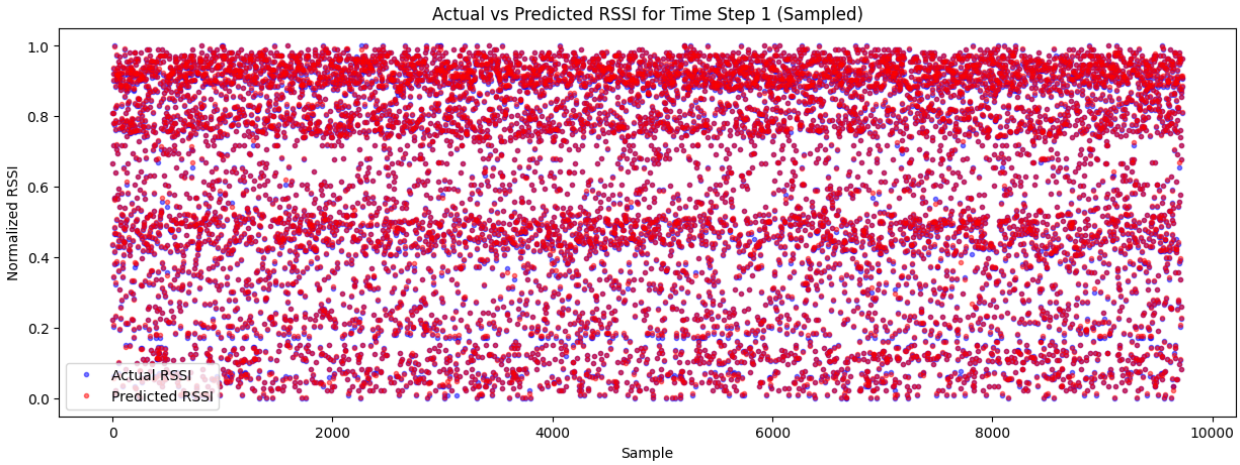
Our evaluation indicates that Random Forest and Gradient Boosting Regression are promising models for predicting RSSI values in our specific context. These models showcase improved accuracy and robustness, providing valuable insights for further refinement and deployment.

LSTM Results

The metrics from our LSTM model indicate excellent performance overall. A Mean Absolute Error (MAE) of 0.0023 and a Mean Squared Error (MSE) of 0.0001 are very low, suggesting that the model's predictions closely match the actual values. The Root Mean Squared Error (RMSE) of 0.0124 and Normalized RMSE (NRMSE) of 0.0080 further confirm the model's accuracy, with NRMSE particularly useful for comparing performance across different datasets. An R-squared (R2) value of 0.9994 is nearly perfect, indicating that the model explains almost all the variability in the data. The Mean Absolute Scaled Error (MASE) of 0.0312 being low suggests good forecasting accuracy, especially when compared to a naïve benchmark. The Explained Variance Score (EVS) of 0.9994 aligns with the R2 score, reinforcing the model's effectiveness.

The infinite Mean Absolute Percentage Error (MAPE), likely due to zeros in the dataset, is the only outlier. It doesn't detract from the model's overall performance but does signal the need for careful interpretation or alternative metrics in cases of zero actual values.

In comparison to traditional regression models, this LSTM appears to outperform in several key areas. Its ability to capture and learn from sequences in the data, a distinct advantage over standard regression techniques, is likely contributing to its superior performance in metrics like MAE, MSE, RMSE, R2, and EVS. These results are indicative of the LSTM's proficiency in handling complex, time-series data, where dependencies and patterns over time are crucial for accurate predictions. The only caveat is the undefined MAPE due to the presence of zeros in the data, an issue that is less about the LSTM's performance and more about the suitability of MAPE as a metric in this context. Traditional regression models, which may not handle time-series data as effectively, might show less impressive results in these metrics, emphasizing the LSTM's advanced capabilities for this particular dataset and problem.



CONCLUSION

In summary, the evaluation of different models in this project demonstrates a clear advantage of the LSTM model in handling complex, time-series data for predicting RSSI values, outperforming traditional regression models in key metrics. This superior performance underscores the LSTM's ability to capture temporal dependencies and patterns, which are critical in time-series forecasting. While the LSTM's performance is robust across most metrics, the infinite MAPE highlights the need for careful selection of evaluation metrics, especially in datasets with specific characteristics like zero values.

Future work could explore the integration of additional features, further optimization of the LSTM architecture, or the application of other advanced time-series models such as ARIMA to enhance predictive accuracy and robustness. This project lays a strong foundation for advanced, data-driven approaches in RSSI prediction, offering valuable insights for both model development and deployment in real-world scenarios.