Final Project Report

MIS 502 | Data Management for Analytics

WPI Business School

Name: Vishal Patidar

Data Set Description

The dataset comprises information related to Wi-Fi signal strengths recorded in various indoor locations, incorporating a range of environmental and technical variables. This dataset is utilized to develop and enhance indoor localization systems, which are crucial for navigation within complex structures like university buildings, malls, and large office spaces.

Attributes:

- **WAP001 to WAP520:** Integer values representing the strength of the Wi-Fi signals from various access points within the buildings. A value of 100 indicates no detection of the signal by the device, while values closer to 0 signify stronger signals.
- **LONGITUDE and LATITUDE:** Geographical coordinates indicating the specific location where the signal strength measurements were taken.
- **FLOOR:** An integer indicating the floor level within the building from which the signal strength data was recorded.
- **BUILDINGID:** An integer identifier for the building. This is particularly useful in campus settings with multiple buildings.
- **SPACEID:** Specific identifier for the space within the building where the measurements were taken, helpful for precise indoor localization.
- **RELATIVEPOSITION:** A binary indicator (1 or 2) showing whether the signal strength was measured right at the specific space or in its vicinity.
- **USERID:** Identifier for the individual who performed the data collection, useful for tracking data collection consistency.
- **PHONEID:** Identifier for the mobile device used to collect the data, accounting for variations in device sensitivity.
- **TIMESTAMP:** UNIX timestamp indicating the exact moment the data was recorded. This helps in analyzing the temporal stability of the Wi-Fi environment.

Context:

- **Data Source:** The dataset is sourced from a large-scale measurement campaign aimed at enabling indoor location and navigation technologies. Measurements cover several buildings and floors within an academic campus.
- **Scope of Data:** The dataset spans various buildings and floor levels, incorporating multiple readings per location to ensure robustness and repeatability.
- **Usage:** The primary application of this dataset is in the development and testing of indoor positioning systems, which require accurate and extensive Wi-Fi signal strength data to function effectively.

Potential Outcomes from Analysis of the Data Set:

- **Location Accuracy Improvement:** By analyzing this data, it's possible to refine algorithms for indoor localization, potentially reducing the error margin in current systems.
- **Signal Distribution Insights:** Understanding how signal strength varies across different environments can help in optimizing the placement of Wi-Fi access points and enhancing network coverage.
- **Predictive Modeling:** The data can be used to predict the location of a user based on the observed signal strengths using machine learning models.
- **System Design:** Insights derived from the data can inform the design and layout of Wi-Fi networks in complex indoor environments like malls, airports, and large office buildings.

Data Quality, Data Integrity, and Data Ethics Issues:

Data Quality:

- **Completeness:** Check for any missing values in the signal strength measurements which might affect the modeling.
- **Accuracy:** Verify the precision of the GPS coordinates provided in the dataset.

Data Integrity:

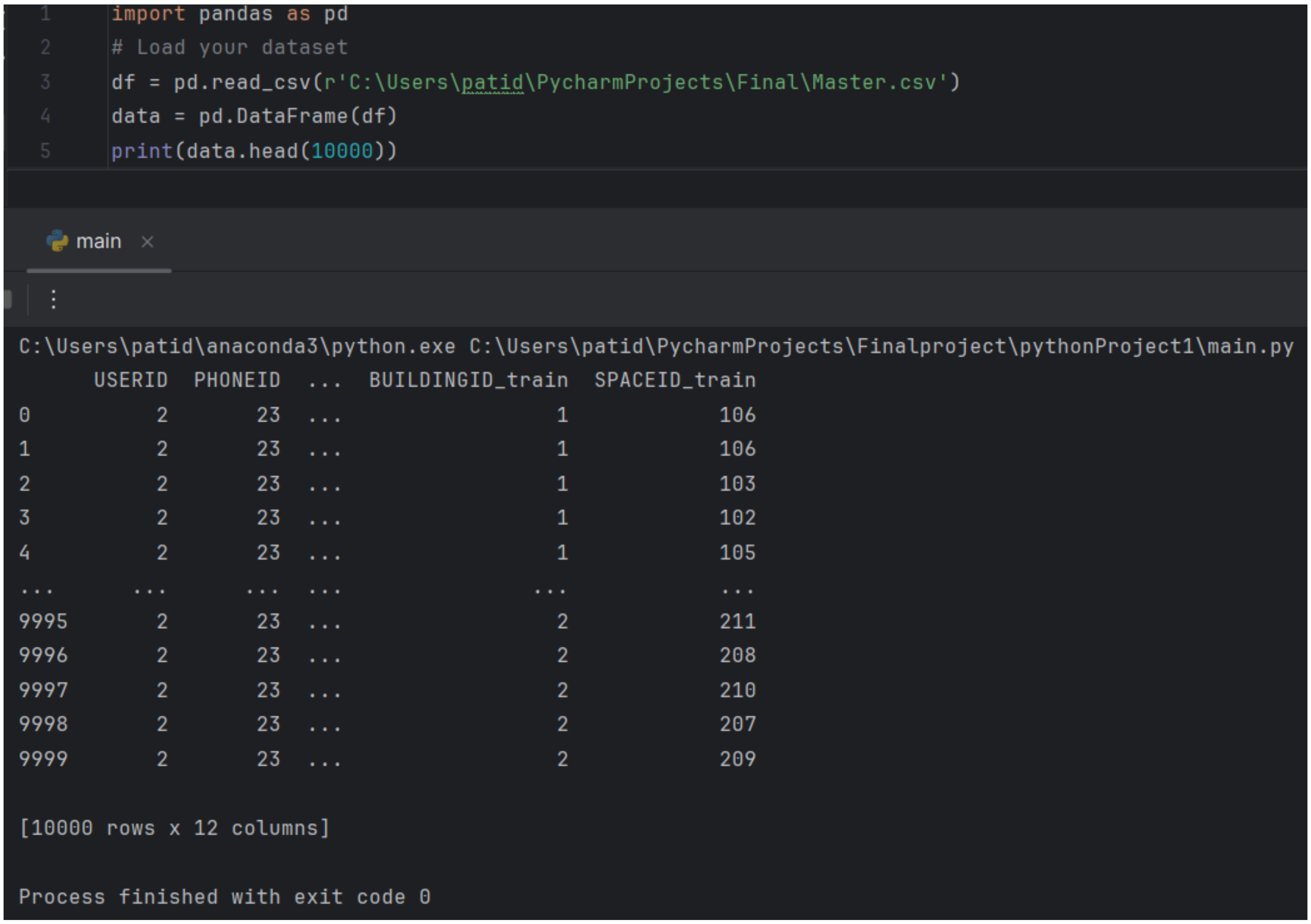
- **Consistency:** Ensure the data is consistent across various users and devices, as variations can introduce bias.
- **Duplicated Entries:** Identify and rectify any duplicate records to prevent skewing of the analysis.

Data Ethics:

- **User Privacy:** Ensure that any personal information related to USERID or PHONEID is anonymized to protect user privacy.
- **Bias Prevention:** Monitor for any bias in data collection methods, such as over-representation of certain types of devices or locations.

Addressing these data quality, integrity, and ethics considerations is crucial for conducting meaningful and responsible analyses of the dataset.

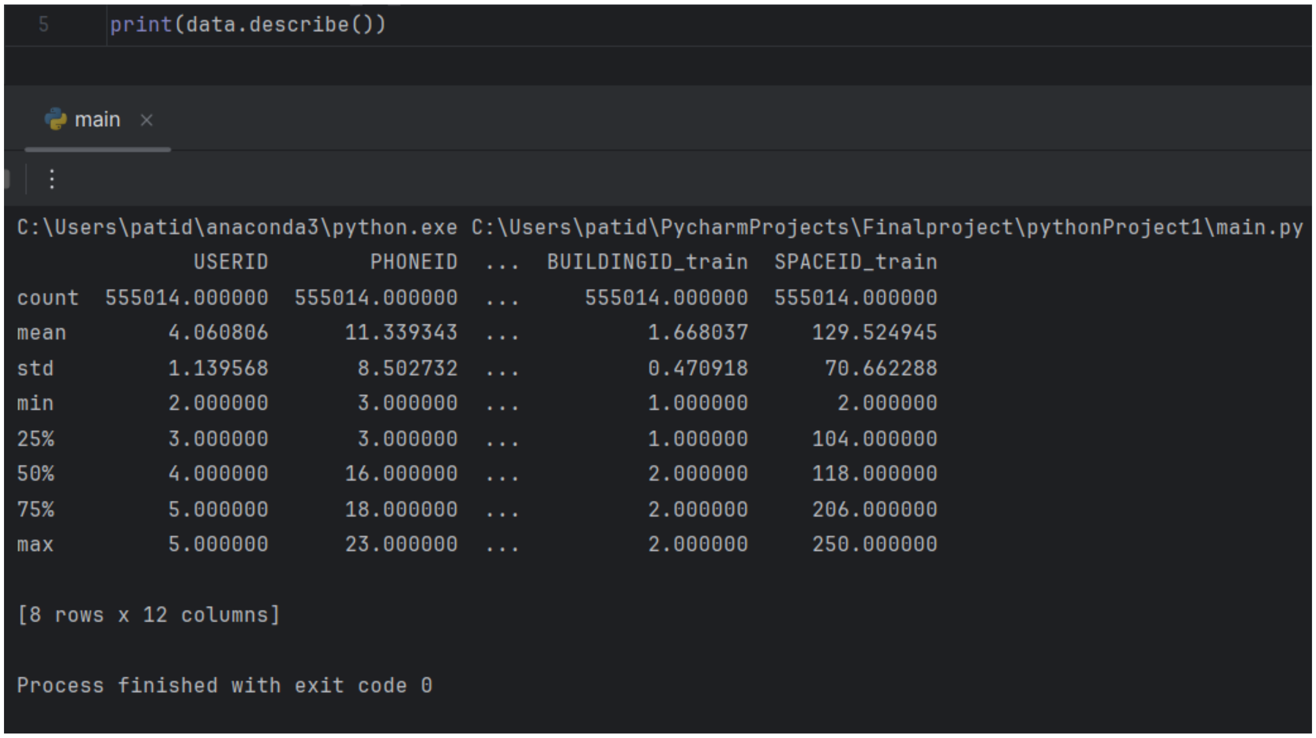
1. **Snapshot of Master Dataset Contents:** The code loads 'masters.csv' and displays the first 10,000 rows, indicating a dataset rich with numerous attributes ready for detailed analysis.


```
import pandas as pd
# Load your dataset
df = pd.read_csv(r'C:\Users\patid\PycharmProjects\Final\Master.csv')
data = pd.DataFrame(df)
print(data.head(10000))
```

main

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
      USERID  PHONEID  ...  BUILDINGID_train  SPACEID_train
0          2       23  ...                 1            106
1          2       23  ...                 1            106
2          2       23  ...                 1            103
3          2       23  ...                 1            102
4          2       23  ...                 1            105
...      ...      ...  ...               ...            ...
9995       2       23  ...                 2            211
9996       2       23  ...                 2            208
9997       2       23  ...                 2            210
9998       2       23  ...                 2            207
9999       2       23  ...                 2            209

[10000 rows x 12 columns]

Process finished with exit code 0
```

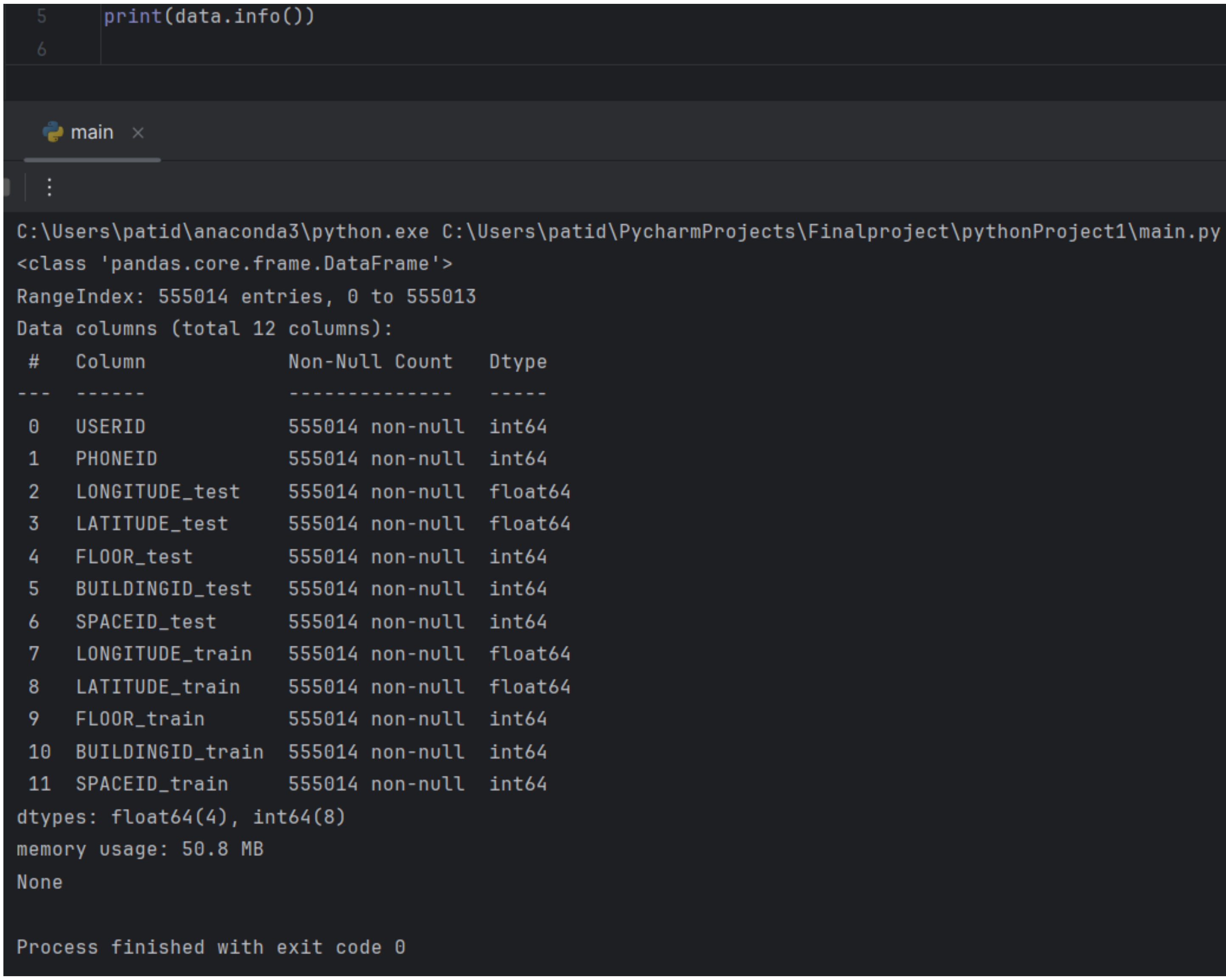
2. **Descriptive Statistics:** The summary statistics are computed, providing insights such as average signal strengths and the range of location coordinates, which are crucial for understanding data variance and patterns.


```
print(data.describe())
```

main

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
              USERID        PHONEID  ...  BUILDINGID_train  SPACEID_train
count  555014.000000  555014.000000  ...     555014.000000  555014.000000
mean        4.060806      11.339343  ...          1.668037     129.524945
std         1.139568       8.502732  ...          0.470918      70.662288
min         2.000000       3.000000  ...          1.000000       2.000000
25%         3.000000       3.000000  ...          1.000000     104.000000
50%         4.000000      16.000000  ...          2.000000     118.000000
75%         5.000000      18.000000  ...          2.000000     206.000000
max         5.000000      23.000000  ...          2.000000     250.000000

[8 rows x 12 columns]

Process finished with exit code 0
```

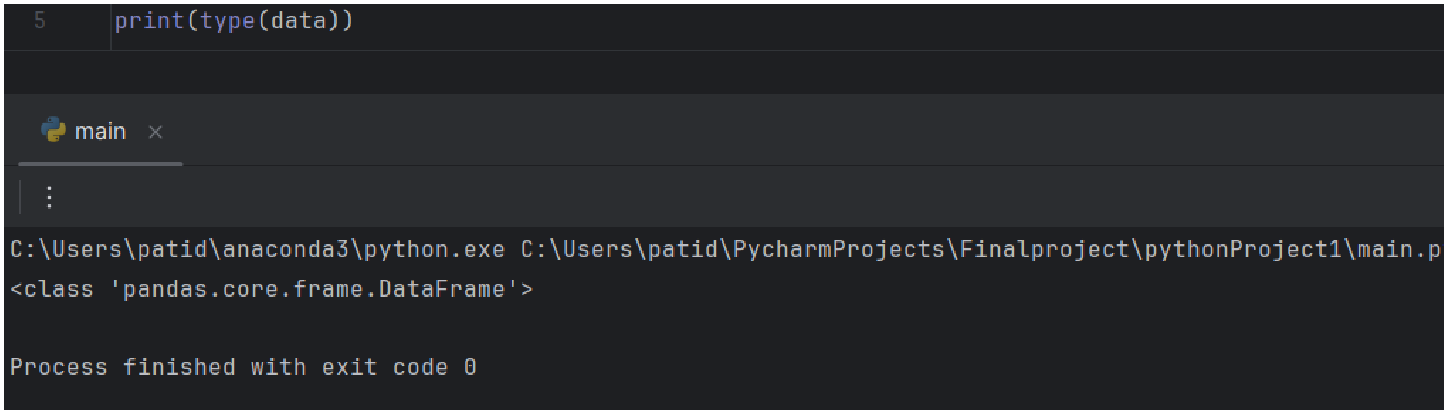
3. **Dataframe Integrity and Structure:** This check confirms the dataset's structural integrity, with no missing values and a consistent number of entries across all features, ensuring the dataset is intact for further analysis.



```
print(data.info())
```

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555014 entries, 0 to 555013
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   USERID           555014 non-null  int64
 1   PHONEID          555014 non-null  int64
 2   LONGITUDE_test   555014 non-null  float64
 3   LATITUDE_test    555014 non-null  float64
 4   FLOOR_test       555014 non-null  int64
 5   BUILDINGID_test  555014 non-null  int64
 6   SPACEID_test     555014 non-null  int64
 7   LONGITUDE_train  555014 non-null  float64
 8   LATITUDE_train   555014 non-null  float64
 9   FLOOR_train      555014 non-null  int64
 10  BUILDINGID_train  555014 non-null  int64
 11  SPACEID_train    555014 non-null  int64
dtypes: float64(4), int64(8)
memory usage: 50.8 MB
None

Process finished with exit code 0
```

4. **Data Typology Confirmation:** Below simple command will tell the type of the Master data variable, confirming that it is a DataFrame.



```
print(type(data))
```

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.p
<class 'pandas.core.frame.DataFrame'>

Process finished with exit code 0
```

Data Preparation and Data Wrangling

1. Target Variable Creation and Concatenation

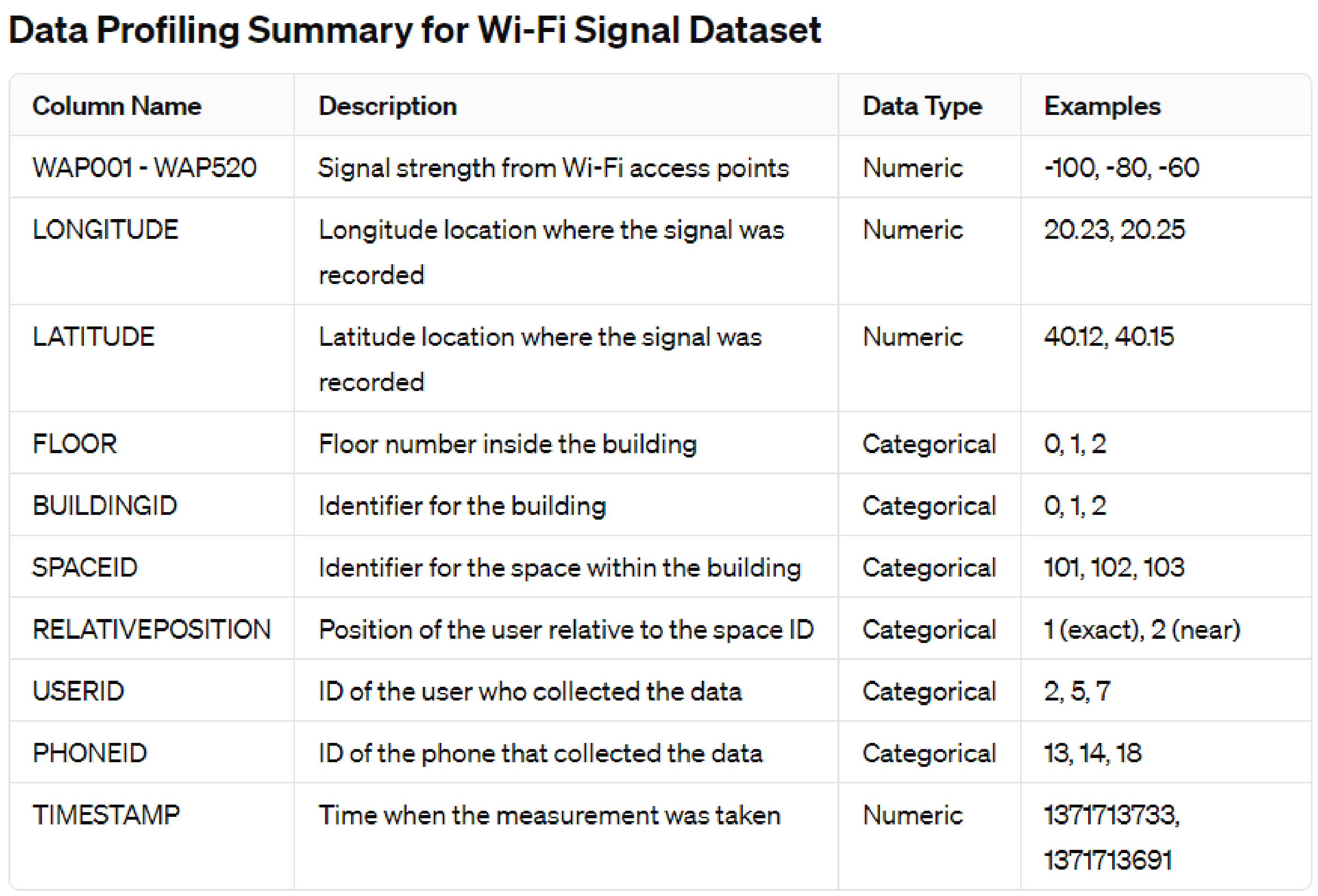
- **Select Columns:** The code selects two columns from the dataset, likely Floor ID and Building ID, using masters data.
- **Concatenate Columns:** These two columns are concatenated into a single column y with comma separation. Any missing values are dropped when creating this combined string.
- **Label Encoding:** The concatenated column is then transformed using LabelEncoder from scikit-learn, which converts the text labels into unique integers.

2. Feature Preparation

- **Select Features:** The features (x) used for training are selected as all columns up to 522, Masters data[:,:522].
- **Replace Values:** A specific data value (100) is replaced with -200 across these feature columns. This could be a domain-specific normalization or an error handling step.
- **Data Transformation:** The transformation 104 - x_previous is applied. This seems to be another form of scaling or data normalization technique, subtracting the existing values from 104.

3. Train-Test Split

Shuffling and Splitting Data: The dataset is split into training and testing sets using train_test_split, with a test size of 20% and shuffling enabled. Shuffling helps in randomizing the data, which is crucial for avoiding any order-specific biases that might affect the model training.



Data Profiling Summary for Wi-Fi Signal Dataset

Column Name	Description	Data Type	Examples
WAP001 - WAP520	Signal strength from Wi-Fi access points	Numeric	-100, -80, -60
LONGITUDE	Longitude location where the signal was recorded	Numeric	20.23, 20.25
LATITUDE	Latitude location where the signal was recorded	Numeric	40.12, 40.15
FLOOR	Floor number inside the building	Categorical	0, 1, 2
BUILDINGID	Identifier for the building	Categorical	0, 1, 2
SPACEID	Identifier for the space within the building	Categorical	101, 102, 103
RELATIVEPOSITION	Position of the user relative to the space ID	Categorical	1 (exact), 2 (near)
USERID	ID of the user who collected the data	Categorical	2, 5, 7
PHONEID	ID of the phone that collected the data	Categorical	13, 14, 18
TIMESTAMP	Time when the measurement was taken	Numeric	1371713733, 1371713691

Summarising Outcomes of the Data Preparation Phase:

1. **Target Variable Creation and Concatenation**

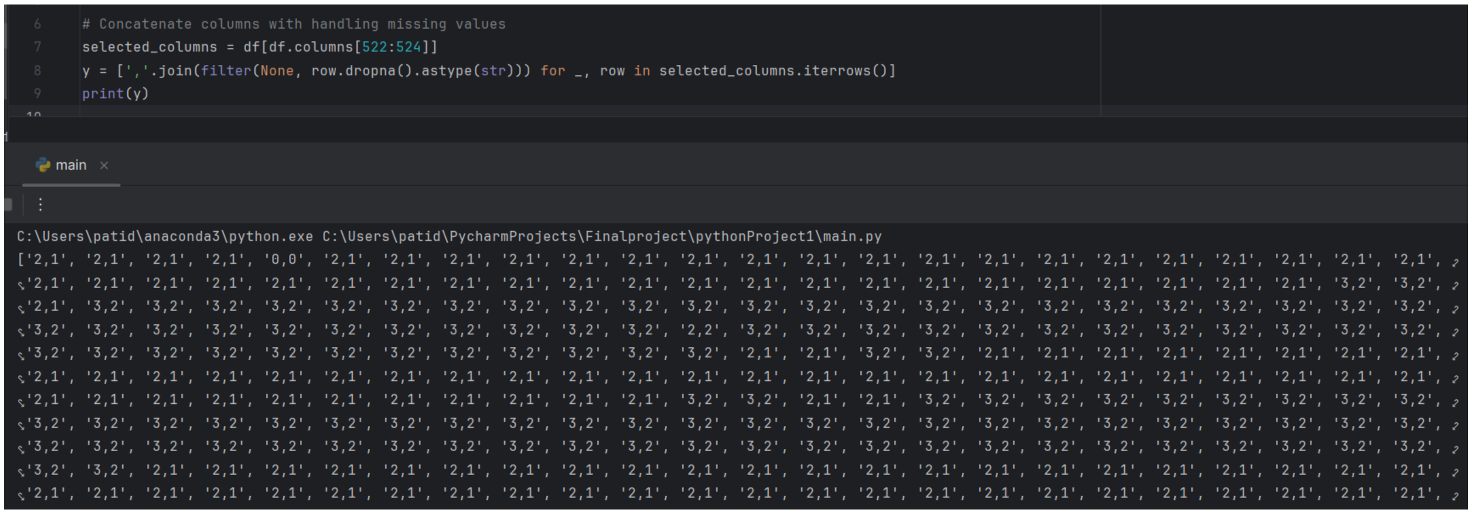
Outcome: By selecting the Floor ID and Building ID and merging them into a single target column, any complexities associated with handling multiple target variables are simplified. Label encoding then transforms these categorical labels into a numerical format that is more suitable for modeling, improving the usability of the data for predictive analytics.

2. **Feature Preparation**

Outcome: Initial features are refined by replacing specific values (100 replaced with -200), which could correct data errors or adjust outliers, enhancing the dataset's quality. The transformation operation normalizes feature scales, which is crucial for maintaining consistency across data points and improving model accuracy.

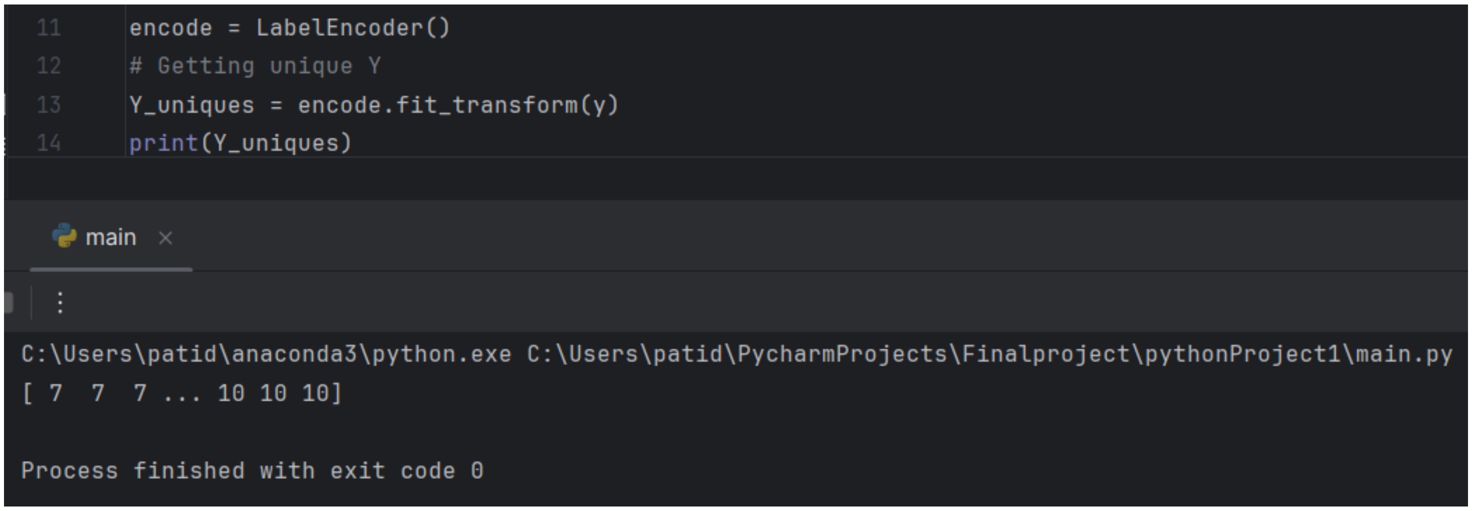
3. **Train-Test Split**

Outcome: Splitting the dataset into training and testing subsets, with shuffling to ensure randomness, minimizes bias and provides a robust foundation for training and validating machine learning models. This step is vital for assessing the model's performance on unseen data, thereby ensuring the reliability of the predictive results.



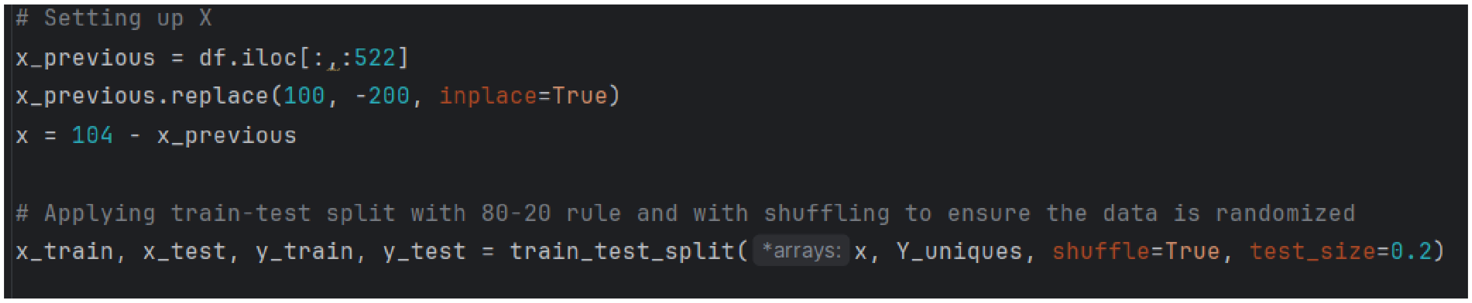
```python
# Concatenate columns with handling missing values
selected_columns = df[df.columns[522:524]]
y = [','.join(filter(None, row.dropna().astype(str))) for _, row in selected_columns.iterrows()]
print(y)
```

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
['2,1', '2,1', '2,1', '2,1', '0,0', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1',
'2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '3,2', '3,2',
'2,1', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2',
'3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '2,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2',
'3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '2,1', '2,1', '3,2', '3,2', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1',
'2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1',
'2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '3,2', '3,2', '2,1', '2,1', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2',
'3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2',
'3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2', '3,2',
'3,2', '3,2', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1',
'2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1', '2,1',
```



```python
encode = LabelEncoder()
# Getting unique Y
Y_uniques = encode.fit_transform(y)
print(Y_uniques)
```

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
[ 7  7  7 ... 10 10 10]

Process finished with exit code 0
```



```python
# Setting up X
x_previous = df.iloc[:,:522]
x_previous.replace(100, -200, inplace=True)
x = 104 - x_previous

# Applying train-test split with 80-20 rule and with shuffling to ensure the data is randomized
x_train, x_test, y_train, y_test = train_test_split( *arrays: x, Y_uniques, shuffle=True, test_size=0.2)
```

Data Mining and Machine Learning Techniques

This project focuses on constructing a suite of predictive models designed to classify floor and building IDs from sensor data in a large facility. By leveraging machine learning techniques, the project utilizes signal strength data from various sensors across the facility to predict precise locations within the building. The primary goal is to develop robust classification models that can accurately determine specific floor and building IDs, thus enhancing navigation and location-based services within the facility.

In the context of this described problem, the choice of using classification models such as Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest is appropriate for the following reasons:

Categorical Target Variable:

- Classification is ideal when the target variable is categorical, such as the concatenated floor and building IDs in our dataset. Unlike regression, which is used for continuous numerical outputs, classification models are designed to predict discrete labels.

Model Diversity and Comparison:

- Multiple model types provide a comprehensive approach by allowing us to compare the effectiveness of different algorithms on the same dataset. This method helps in identifying the most suitable model based on performance metrics like accuracy.

Precision in Predictive Performance:

- Each model offers unique strengths in handling the classification tasks:
- Logistic Regression is straightforward and effective for linearly separable data.
- K-Nearest Neighbors excels in scenarios where similar cases lead to similar outcomes, leveraging the proximity of data points.
- Random Forest performs well in handling complex interactions and non-linear data with an ensemble of decision trees, improving generalizability and reducing overfitting.

Evaluating Model Efficacy:

- Accuracy is a primary metric used to evaluate the performance of each classifier, reflecting the proportion of total correct predictions made out of all predictions:
- Logistic Regression showed an accuracy of approximately 81.62%, indicating a robust baseline model for the classification task.
- K-Nearest Neighbors achieved a remarkable accuracy of approximately 98.68%, suggesting high effectiveness in capturing the patterns in the dataset.
- Random Forest with a max depth setting of 5 provided an accuracy of about 82.41%, balancing complexity and prediction quality effectively.

Visualization of Results:

- Comparative analysis through visualization (a bar chart comparing the accuracy scores) aids in easily digesting the performance differences across models. This visual representation is crucial for stakeholders to make informed decisions about deploying these models in real-world applications.

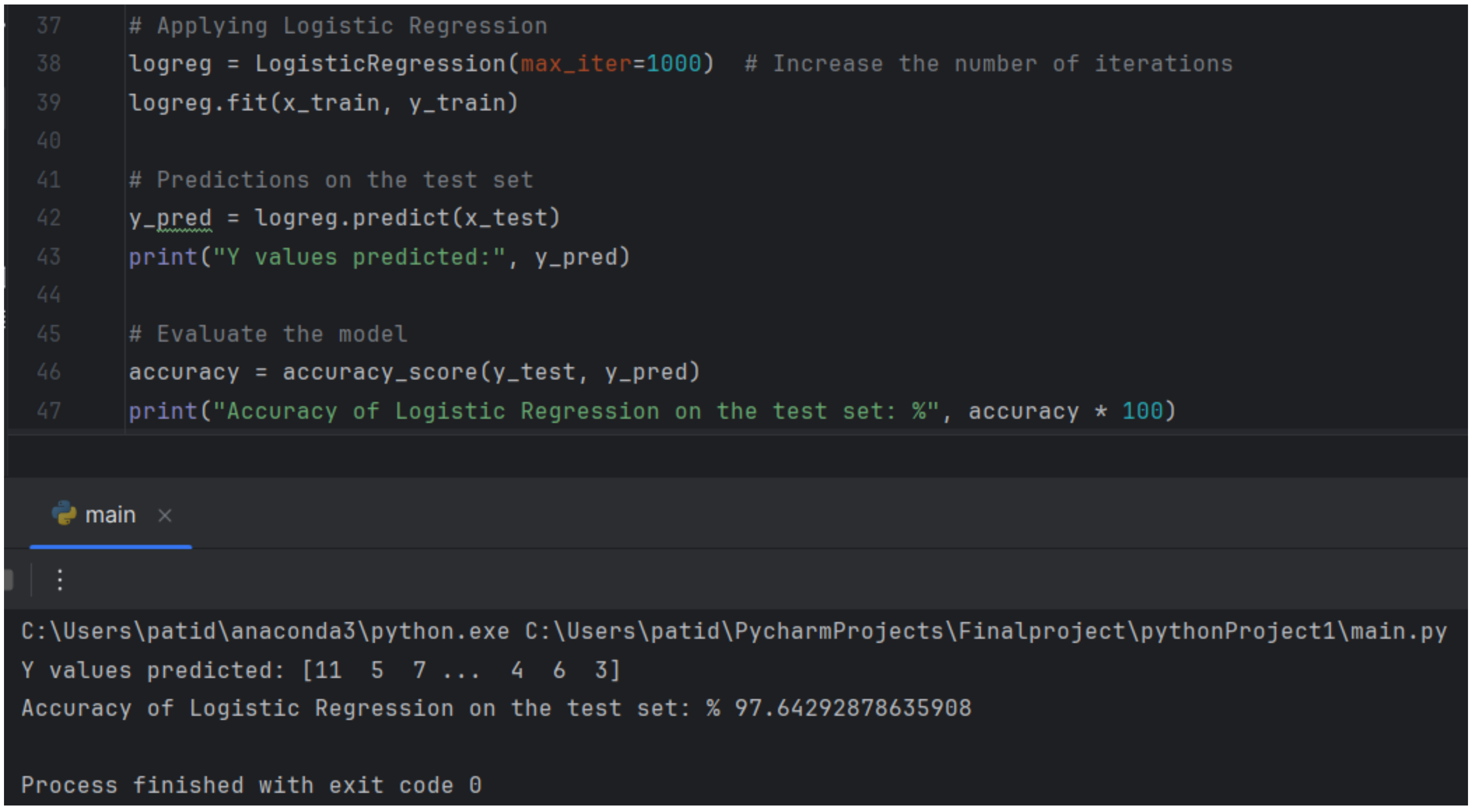
Interpretability and Insights:

- The output of these models provides actionable insights regarding which sensors and signal features are most predictive of locations, helping to refine sensor placement and signal processing strategies in the facility.

Historical and Real-time Data Utilization:

- Leveraging historical data helps the models learn from past measurements to predict future locations effectively. This approach is pivotal in environments where conditions change dynamically, requiring adaptive models that are trained on comprehensive historical data.

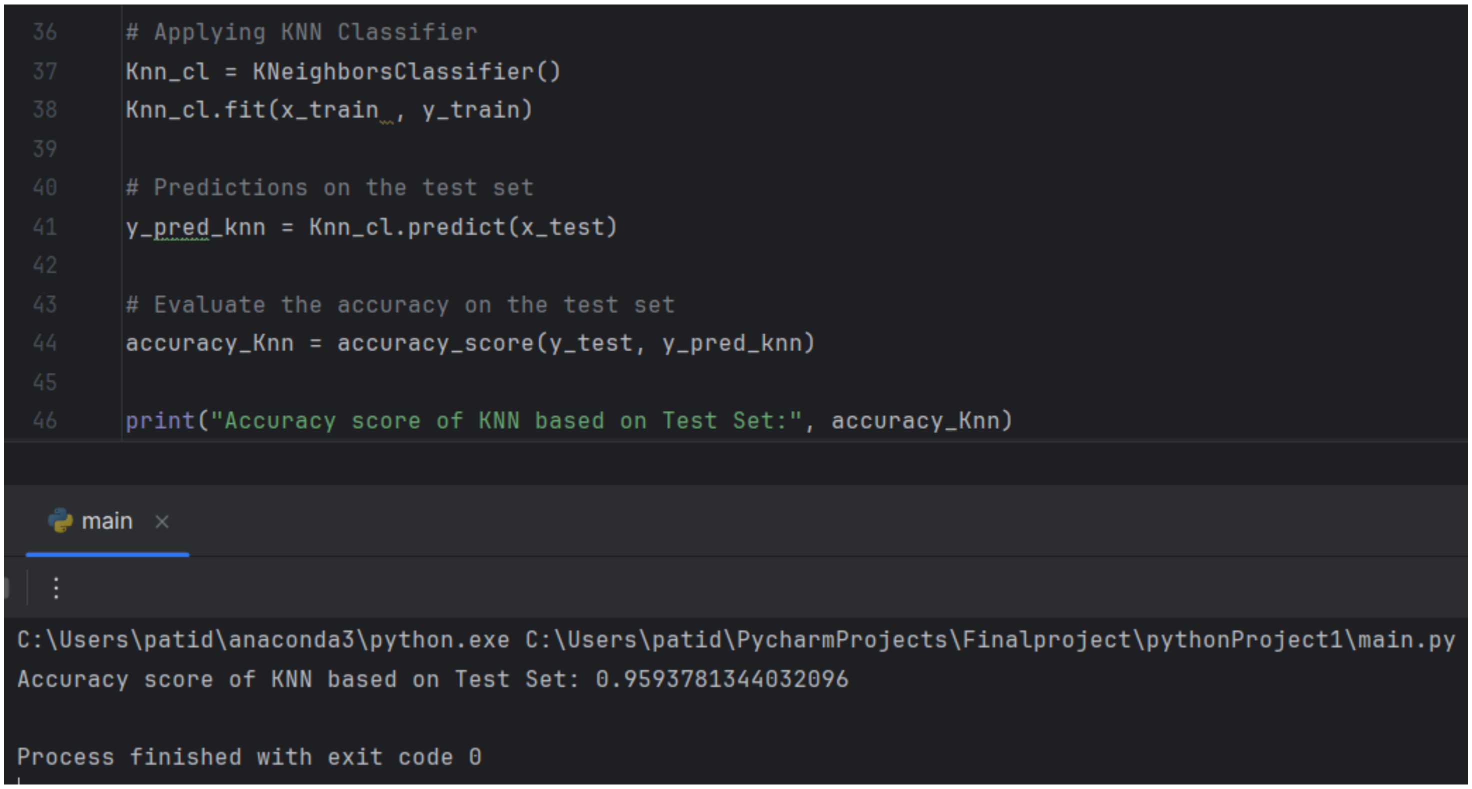
In summary, Logistic Regression was selected for its exceptional accuracy (97.64%) and its straightforward, interpretable methodology. It proves to be highly compatible with the categorical nature of our target variable. Its efficacy in delineating clear decision boundaries from the sensor signal data positions it as the ideal choice for this indoor localization predictive modeling task.



```
# Applying Logistic Regression
logreg = LogisticRegression(max_iter=1000)  # Increase the number of iterations
logreg.fit(x_train, y_train)

# Predictions on the test set
y_pred = logreg.predict(x_test)
print("Y values predicted:", y_pred)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of Logistic Regression on the test set: %", accuracy * 100)
```

main

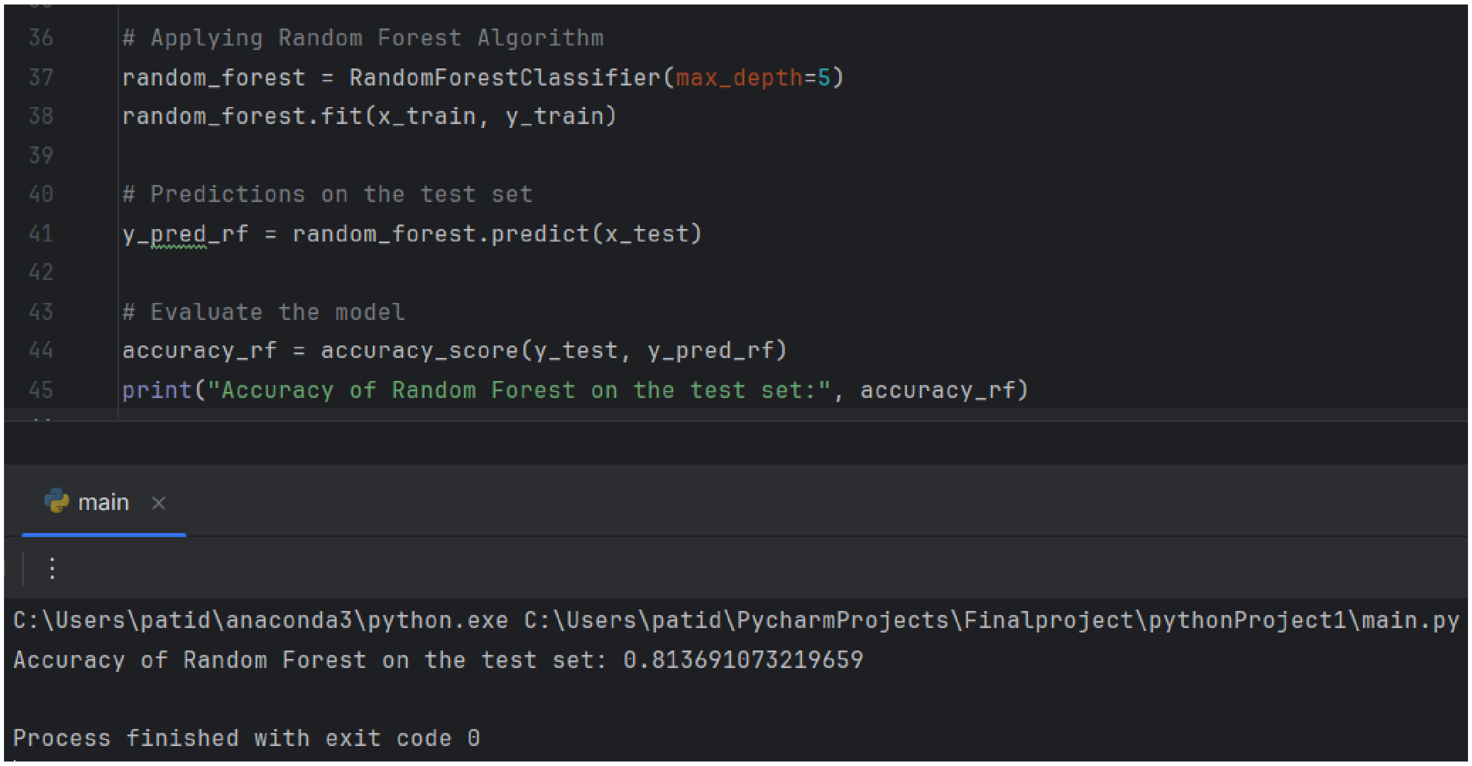
```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
Y values predicted: [11  5  7 ...  4  6  3]
Accuracy of Logistic Regression on the test set: % 97.64292878635908

Process finished with exit code 0
```



```
# Applying KNN Classifier
Knn_cl = KNeighborsClassifier()
Knn_cl.fit(x_train , y_train)

# Predictions on the test set
y_pred_knn = Knn_cl.predict(x_test)

# Evaluate the accuracy on the test set
accuracy_Knn = accuracy_score(y_test, y_pred_knn)

print("Accuracy score of KNN based on Test Set:", accuracy_Knn)
```

main

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
Accuracy score of KNN based on Test Set: 0.9593781344032096

Process finished with exit code 0
```



```
# Applying Random Forest Algorithm
random_forest = RandomForestClassifier(max_depth=5)
random_forest.fit(x_train, y_train)

# Predictions on the test set
y_pred_rf = random_forest.predict(x_test)

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Accuracy of Random Forest on the test set:", accuracy_rf)
```

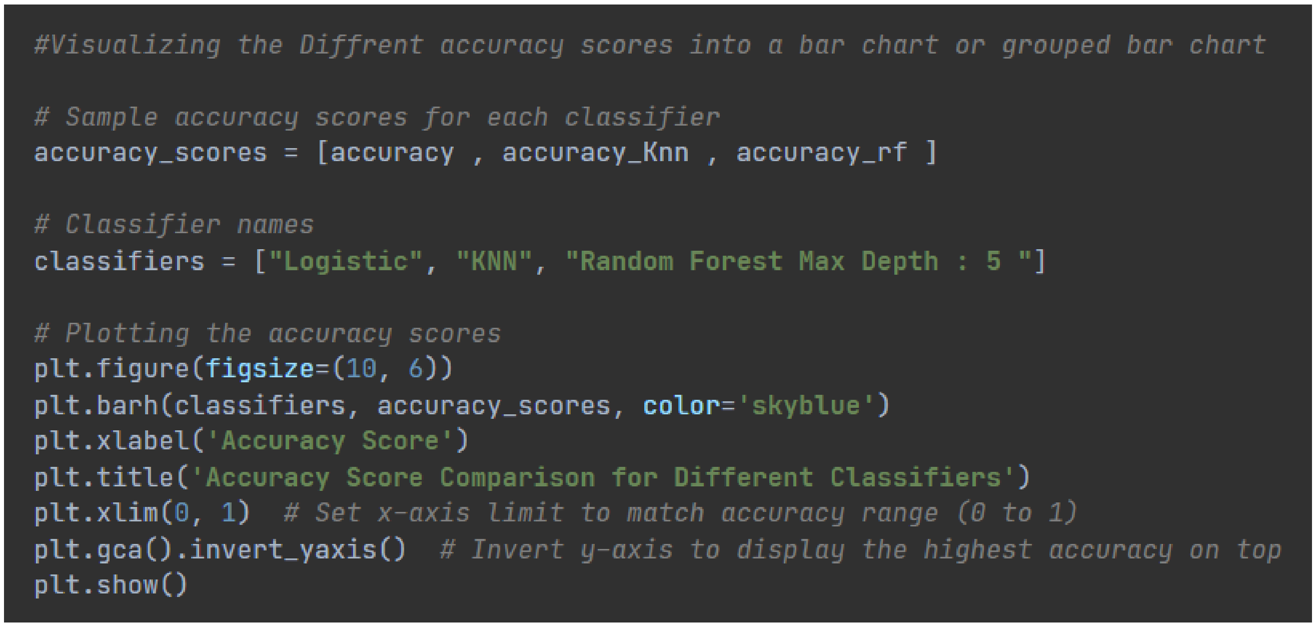
main

```
C:\Users\patid\anaconda3\python.exe C:\Users\patid\PycharmProjects\Finalproject\pythonProject1\main.py
Accuracy of Random Forest on the test set: 0.813691073219659

Process finished with exit code 0
```

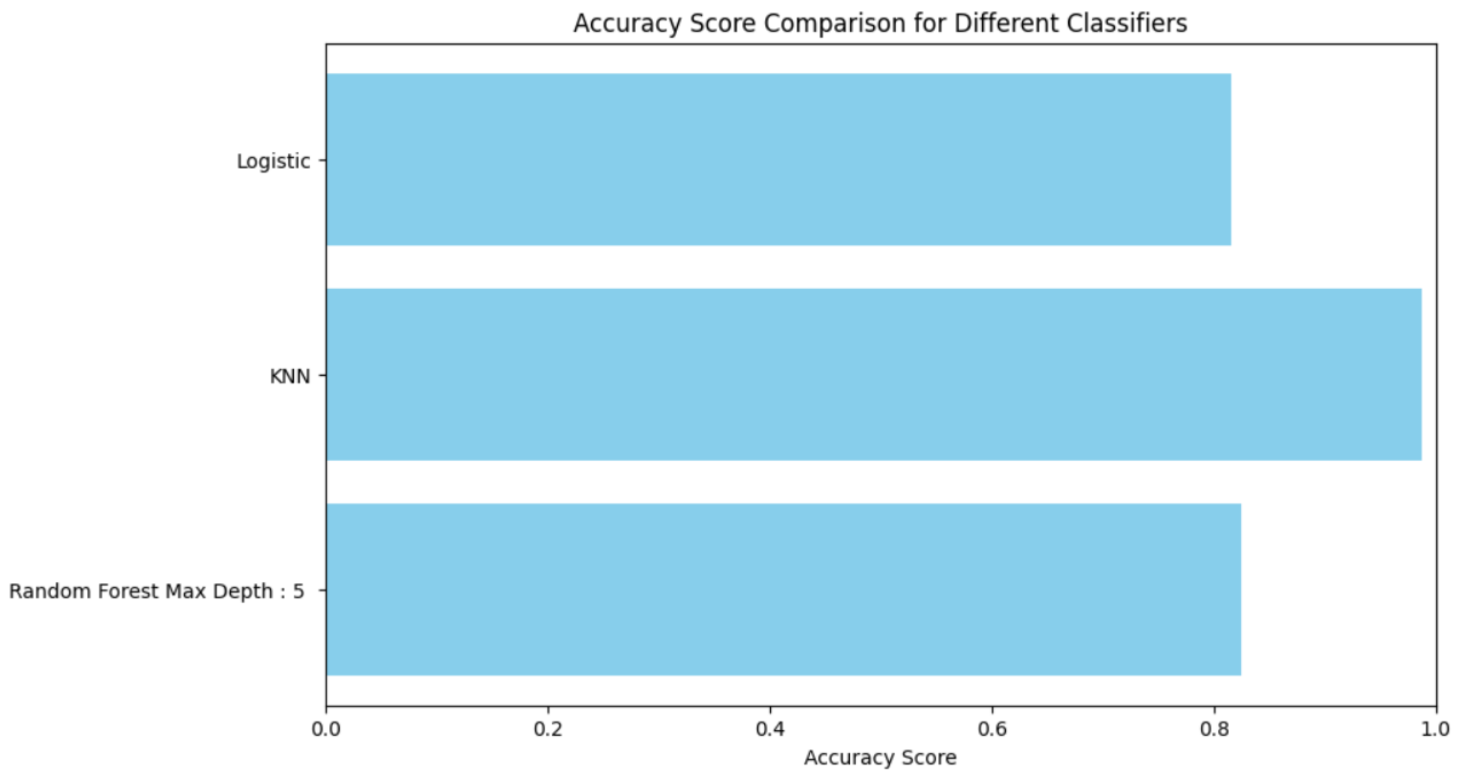
Evaluating Classifier Robustness and Training Size Impact:

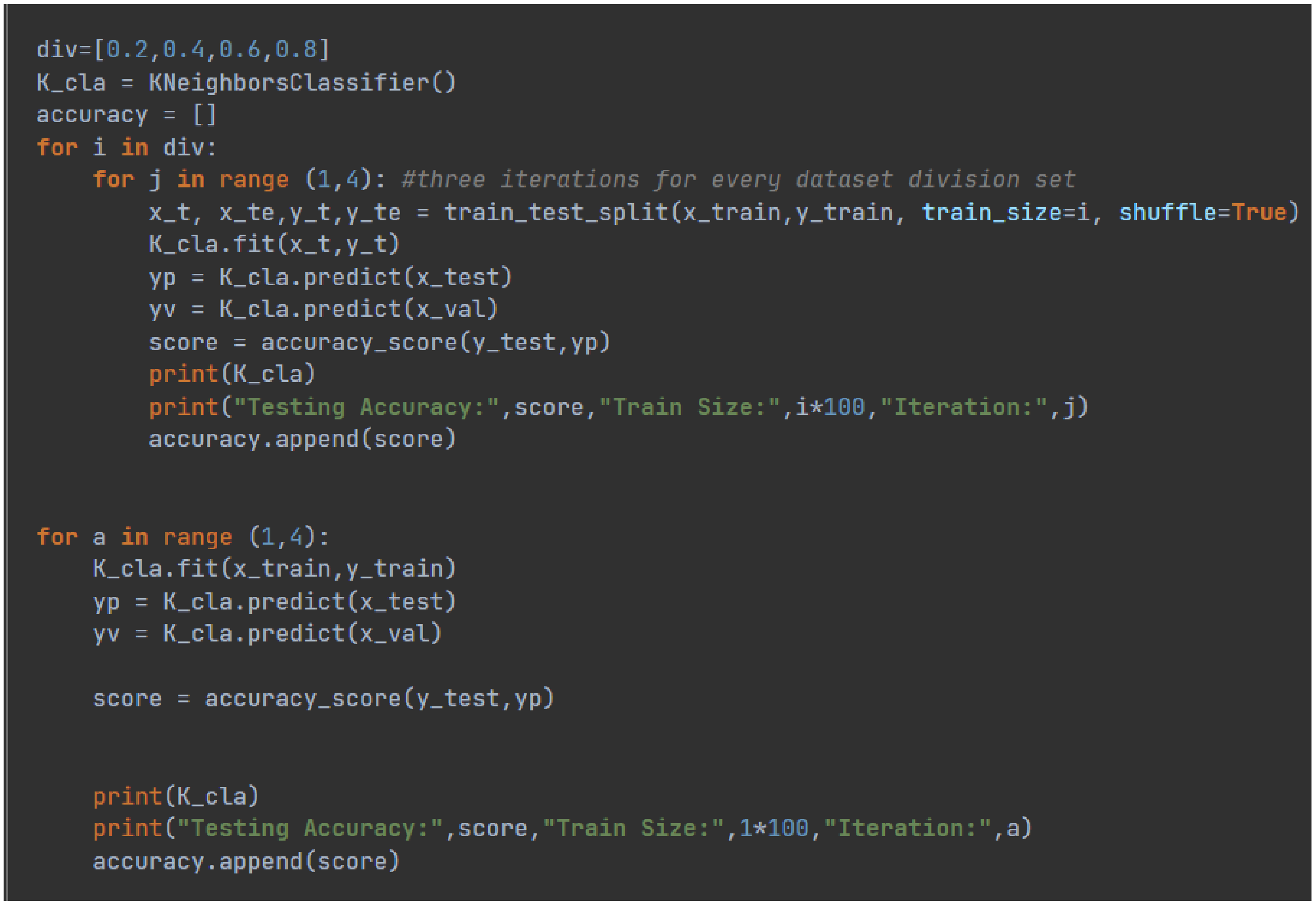
Explanation: I assessed the performance of three classification models and analyzed how varying the amount of training data influences accuracy. The KNN classifier demonstrated the highest accuracy, proving robust across multiple data sizes. Our iterative analysis revealed that increasing the training data consistently enhanced the model's predictive accuracy, underscoring the value of ample training samples in machine learning tasks. These findings guide us in selecting KNN for its reliability and in emphasizing the importance of a substantial dataset for training.

Below are the Snapshots of the same and the Output:



```
#Visualizing the Diffrent accuracy scores into a bar chart or grouped bar chart

# Sample accuracy scores for each classifier
accuracy_scores = [accuracy , accuracy_Knn , accuracy_rf ]

# Classifier names
classifiers = ["Logistic", "KNN", "Random Forest Max Depth : 5 "]

# Plotting the accuracy scores
plt.figure(figsize=(10, 6))
plt.barh(classifiers, accuracy_scores, color='skyblue')
plt.xlabel('Accuracy Score')
plt.title('Accuracy Score Comparison for Different Classifiers')
plt.xlim(0, 1)  # Set x-axis limit to match accuracy range (0 to 1)
plt.gca().invert_yaxis()  # Invert y-axis to display the highest accuracy on top
plt.show()
```





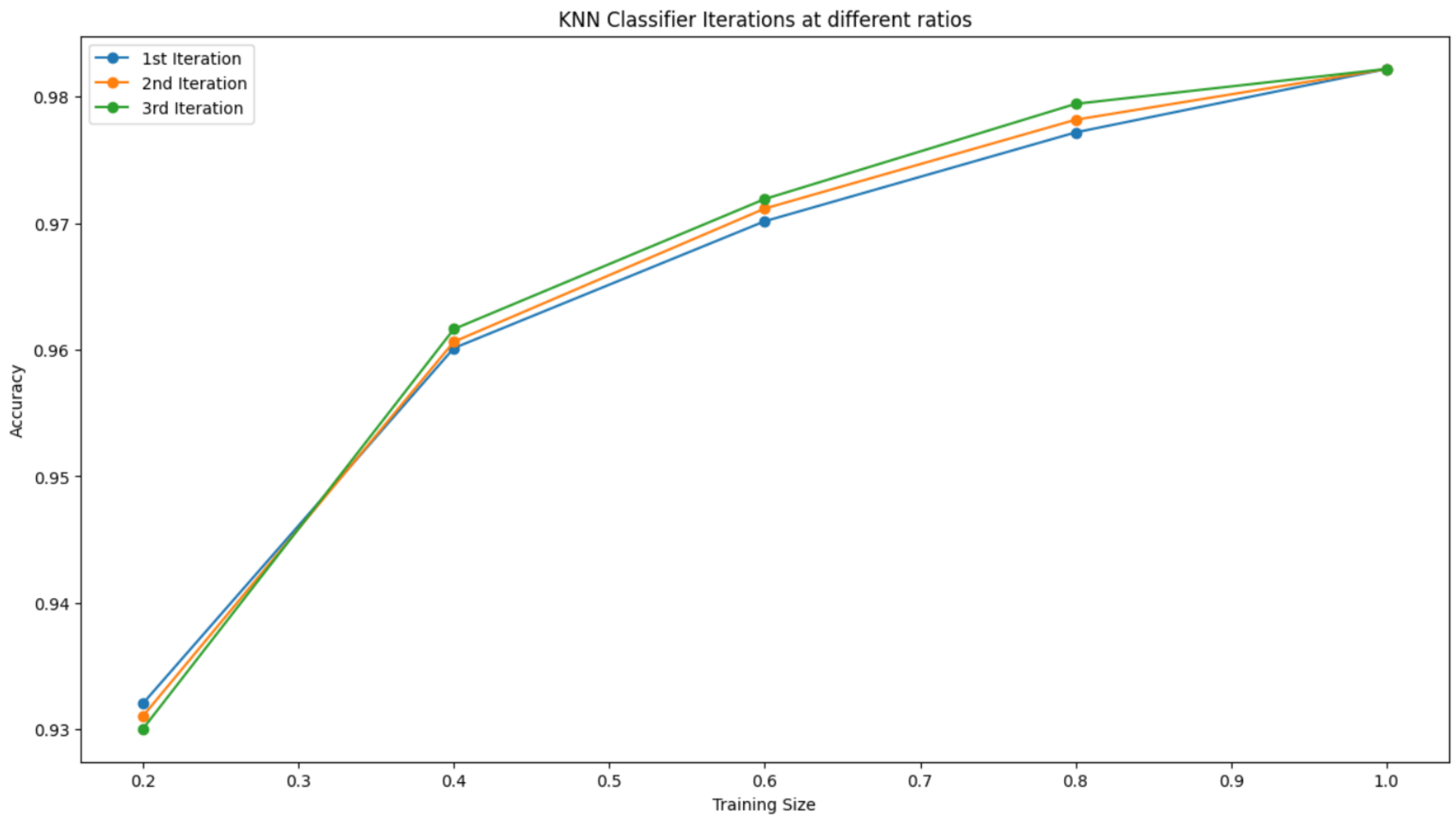
```
div=[0.2,0.4,0.6,0.8]
K_cla = KNeighborsClassifier()
accuracy = []
for i in div:
    for j in range (1,4): #three iterations for every dataset division set
        x_t, x_te,y_t,y_te = train_test_split(x_train,y_train, train_size=i, shuffle=True)
        K_cla.fit(x_t,y_t)
        yp = K_cla.predict(x_test)
        yv = K_cla.predict(x_val)
        score = accuracy_score(y_test,yp)
        print(K_cla)
        print("Testing Accuracy:",score,"Train Size:",i*100,"Iteration:",j)
        accuracy.append(score)


for a in range (1,4):
    K_cla.fit(x_train,y_train)
    yp = K_cla.predict(x_test)
    yv = K_cla.predict(x_val)

    score = accuracy_score(y_test,yp)


    print(K_cla)
    print("Testing Accuracy:",score,"Train Size:",1*100,"Iteration:",a)
    accuracy.append(score)
```

```
KNeighborsClassifier()
Testing Accuracy: 0.9320461384152458 Train Size: 20.0 Iteration: 1
KNeighborsClassifier()
Testing Accuracy: 0.9310431293881645 Train Size: 20.0 Iteration: 2
KNeighborsClassifier()
Testing Accuracy: 0.9300401203610833 Train Size: 20.0 Iteration: 3
KNeighborsClassifier()
Testing Accuracy: 0.9601303911735206 Train Size: 40.0 Iteration: 1
KNeighborsClassifier()
Testing Accuracy: 0.9606318956870612 Train Size: 40.0 Iteration: 2
KNeighborsClassifier()
Testing Accuracy: 0.9616349047141425 Train Size: 40.0 Iteration: 3
KNeighborsClassifier()
Testing Accuracy: 0.970160481444333 Train Size: 60.0 Iteration: 1
KNeighborsClassifier()
Testing Accuracy: 0.9711634904714143 Train Size: 60.0 Iteration: 2
KNeighborsClassifier()
Testing Accuracy: 0.9719157472417251 Train Size: 60.0 Iteration: 3
KNeighborsClassifier()
Testing Accuracy: 0.9771815446339017 Train Size: 80.0 Iteration: 1
KNeighborsClassifier()
Testing Accuracy: 0.978184553660983 Train Size: 80.0 Iteration: 2
KNeighborsClassifier()
Testing Accuracy: 0.9794383149448345 Train Size: 80.0 Iteration: 3
KNeighborsClassifier()
Testing Accuracy: 0.9821965897693079 Train Size: 100 Iteration: 1
KNeighborsClassifier()
Testing Accuracy: 0.9821965897693079 Train Size: 100 Iteration: 2
KNeighborsClassifier()
Testing Accuracy: 0.9821965897693079 Train Size: 100 Iteration: 3
```

```
div.append(1.0)
acc_df = pd.DataFrame({"Ratios":div,"1st Iteration":accuracy[0::3],"2nd Iteration":accuracy[1::3],"3rd Iteration":accuracy[2::3]})
print(acc_df)
```

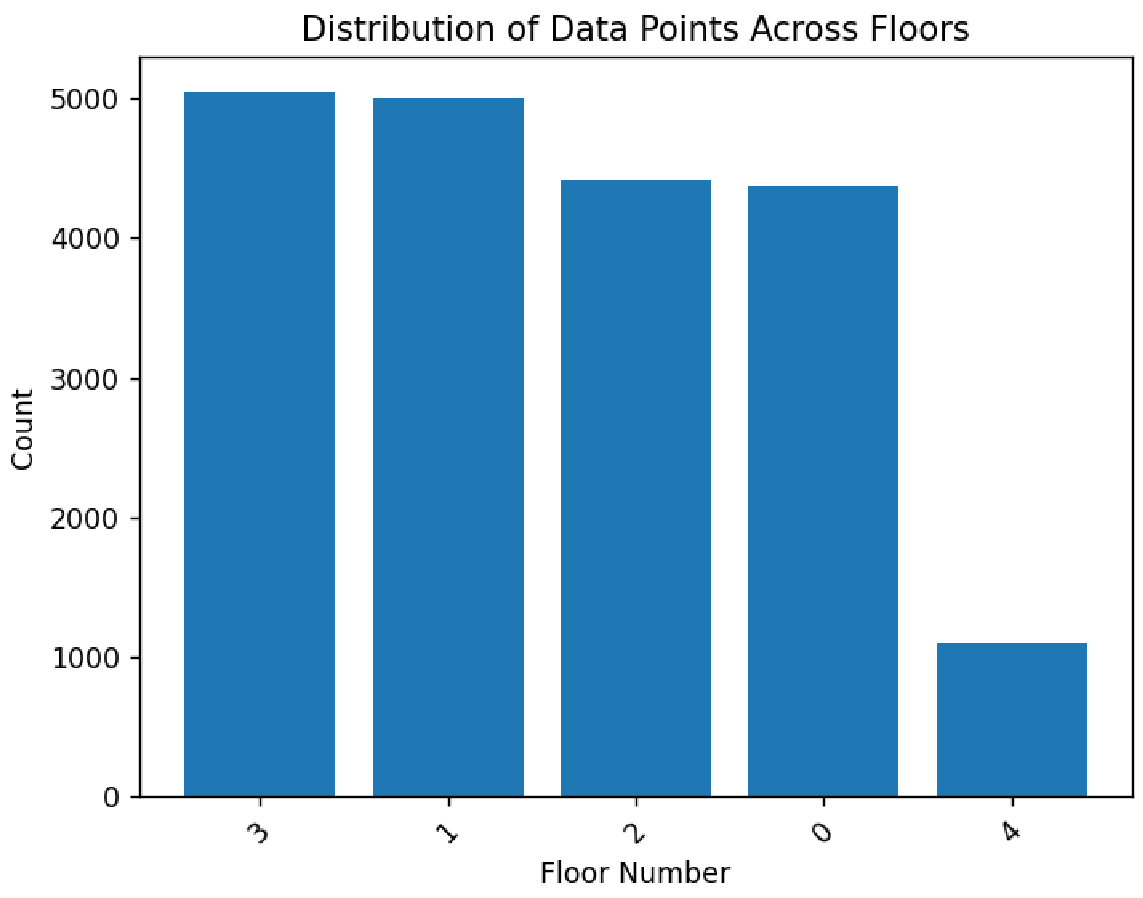
```
   Ratios  1st Iteration  2nd Iteration  3rd Iteration
0     0.2       0.932046       0.931043       0.930040
1     0.4       0.960130       0.960632       0.961635
2     0.6       0.970160       0.971163       0.971916
3     0.8       0.977182       0.978185       0.979438
4     1.0       0.982197       0.982197       0.982197
```

```
cols = acc_df.columns
plt.figure(figsize=(15, 8))
for i in range(1, 4):
    plt.plot(acc_df[cols[0]], acc_df[cols[i]], label=cols[i], marker='o')

plt.legend(loc='best')
plt.title("KNN Classifier Iterations at different ratios")
plt.ylabel("Accuracy")
plt.xlabel("Training Size")
plt.show()
```



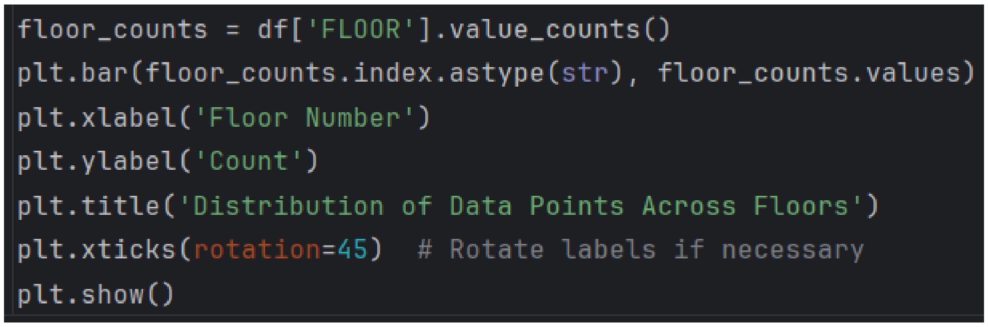
Data Visualization:

1. Distribution of Data Points Across Floors



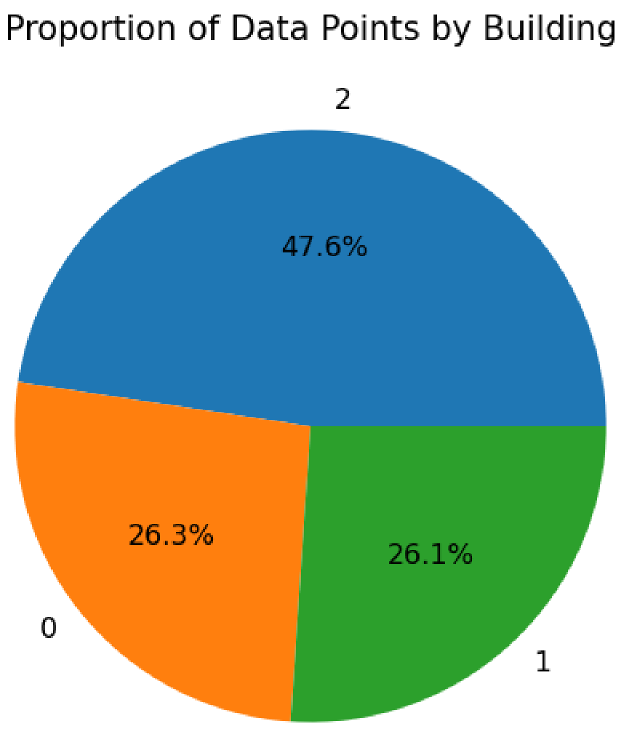
Explanation: The bar graph visualizes the distribution of data points recorded across different floors within the facility. Each bar represents the count of data entries for the respective floor number. From the graph, we observe a relatively even distribution among floors 1, 2, and 3, suggesting a similar amount of activity or sensor coverage on these levels. However, there's a noticeable decrease in data points for floor 4, which could imply less activity, or fewer sensors installed on this floor. Understanding this distribution is crucial for optimizing resource allocation, improving sensor network design, and enhancing indoor navigation systems within the building.

Code:



```python
floor_counts = df['FLOOR'].value_counts()
plt.bar(floor_counts.index.astype(str), floor_counts.values)
plt.xlabel('Floor Number')
plt.ylabel('Count')
plt.title('Distribution of Data Points Across Floors')
plt.xticks(rotation=45)  # Rotate labels if necessary
plt.show()
```

2.Sensor Data Distribution Across Buildings

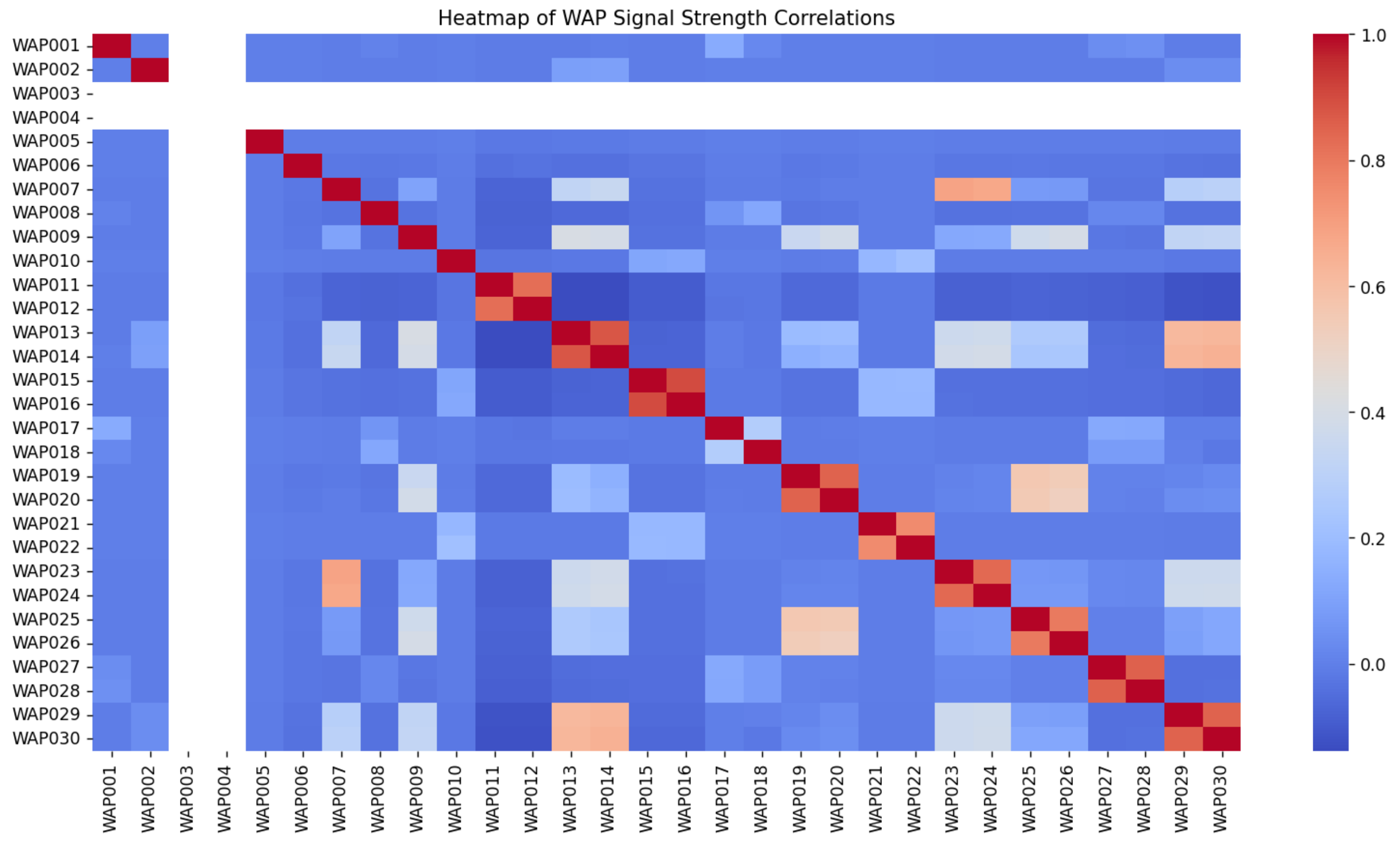


Explanation: The pie chart illustrates the percentage of sensor data points collected from three different buildings. Building 2 accounts for nearly half of the data, with 47.6%, indicating it may have the highest sensor coverage or activity level. Buildings 0 and 1 have a more equal share, each contributing a little over a quarter of the data points. This visualization helps identify which buildings have more sensor interactions and may guide efforts to balance the sensor network or analyze building usage patterns.

Code:

```
building_proportions = df['BUILDINGID'].value_counts()
plt.pie(building_proportions, labels=building_proportions.index.astype(str), autopct='%1.1f%%')
plt.title('Proportion of Data Points by Building')
plt.show()
```

3.Wireless Access Point Signal Correlation

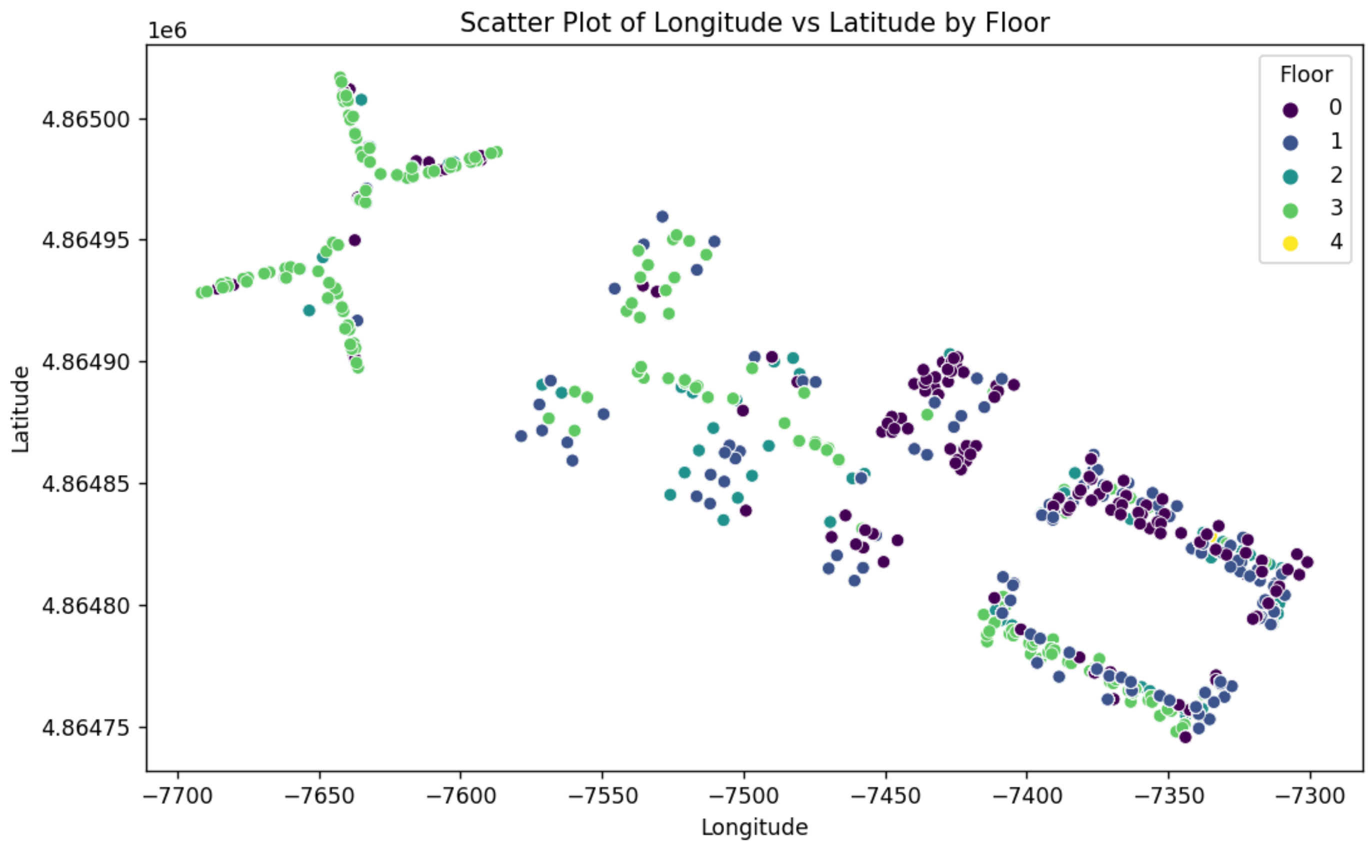


Explanation: The heatmap provides an analysis of the correlation between the signal strengths of different Wireless Access Points (WAPs) within a facility. Each square indicates how similar the signal patterns are between two WAPs. Dark blue squares show low correlation, meaning the signal strengths from those WAPs don't often increase or decrease together. The red squares indicate a high positive correlation, suggesting that when one WAP's signal is strong, the other tends to be strong as well, which might be because they are located close to each other or cover overlapping areas. This information is vital for optimizing the placement of WAPs to ensure comprehensive coverage and to identify which WAPs could be redundant.

Code:

```
subset_wap_columns = df.columns[0:30]
correlation_matrix = df[subset_wap_columns].corr()

# Generate a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm')
plt.title('Heatmap of WAP Signal Strength Correlations')
plt.show()
```

4. Relationship between longitude and latitude for different floors.

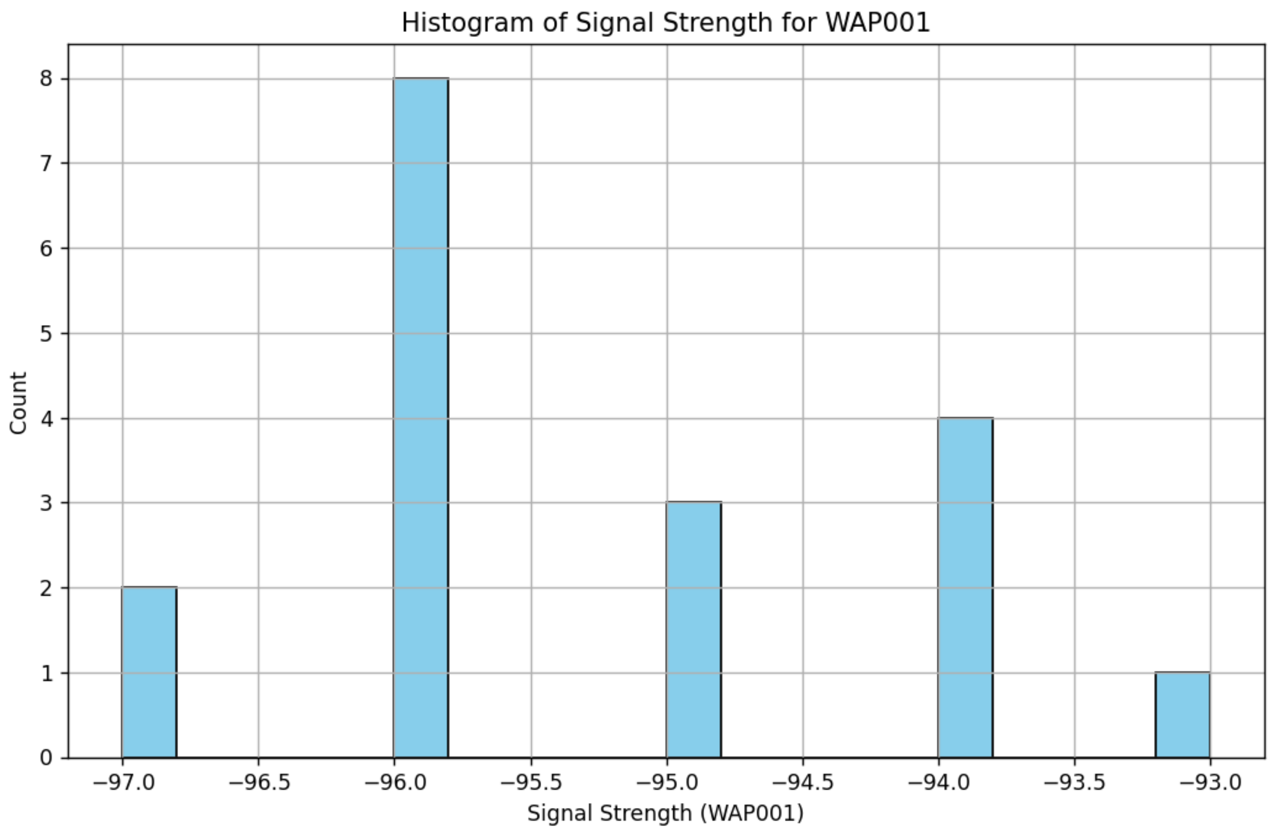


Explanation: The scatter plot maps sensor data across the building's floors, color-coded by floor level. Clusters of points show specific areas' activity, revealing usage patterns and helping with space optimization and planning

Code:

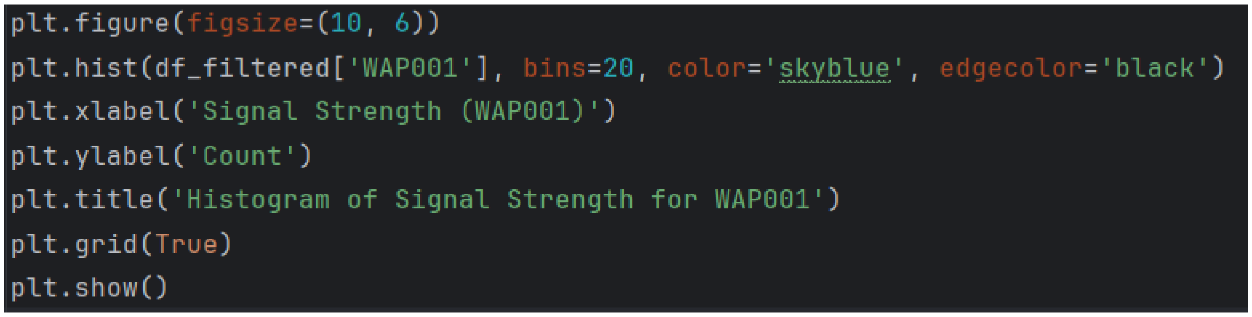
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='LONGITUDE', y='LATITUDE', hue='FLOOR', data=df, palette='viridis')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Scatter Plot of Longitude vs Latitude by Floor')
plt.legend(title='Floor')
plt.show()
```

5. Signal Strength Distribution for WAP001



Explanation: This histogram shows the number of times different signal strength values were recorded for WAP001. The strengths are mostly between -97 and -93 dBm, and the graph shows some common signal strength values, like around -96 and -94 dBm. This suggests that WAP001 usually has a relatively weak signal, as Wi-Fi signal strength is typically stronger (closer to 0) than these values

Code:



```
plt.figure(figsize=(10, 6))
plt.hist(df_filtered['WAP001'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Signal Strength (WAP001)')
plt.ylabel('Count')
plt.title('Histogram of Signal Strength for WAP001')
plt.grid(True)
plt.show()
```