**SUMMARY**

**Data Collection and Analysis:**

* Reading the data from the 'AR\_df.csv' file into a DataFrame: data\_df = pd.read\_csv('AR\_df.csv').
* We have checked the length of our dataframe which is 5598130 by using this code: len(data\_df).
* Checked our dataframe columns by using this code: data\_df.columns. We have 23 columns in our dataframe including: ['TIME', 'Acceleration\_X', 'Acceleration\_Y', 'Acceleration\_Z',

'Gyroscope\_X', 'Gyroscope\_Y', 'Gyroscope\_Z', 'Magnetometer\_X',

'Magnetometer\_Y', 'Magnetometer\_Z', 'Orientation\_w', 'Orientation\_x', 'Orientation\_y', 'Orientation\_z', 'Gravity\_X', 'Gravity\_Y', 'Gravity\_Z', 'LAcceleration\_X', 'LAcceleration\_Y', 'LAcceleration\_Z', 'Pressure', 'Coarse\_Label', 'User'].

* Checking the categories of Course Label by using this code :data\_df.Coarse\_Label.value\_counts(). And got the result:  
  5 2160100

4 1572007

6 864002

2 672010

3 330011

* Checking any null values in our dataframe by using this code:  
  data\_df.isnull().sum(). And we have 0 null values in our dataframe.

**Preprocessing the Data:**

* Dropping irrelevant columns ('TIME', 'User'): data\_df = data\_df.drop(columns=['TIME', 'User'])
* Getting sensor columns by using this code:  
  sensor\_df = data\_df.drop('Coarse\_Label', axis=1)

sensor\_columns = sensor\_df.columns

This is useful to extract our features in a feature extraction part.

* Normalizing the sensor columns: data\_df[sensor\_columns] = (data\_df[sensor\_columns] - data\_df[sensor\_columns].mean()) / data\_df[sensor\_columns].std().

We do normalizing to keep our data at one scale.

* Defining labels based on 'Coarse\_Label' column: data\_df['Label'] = np.where(data\_df['Coarse\_Label'] == 4, 1, 0). In this code we labelling our course Label data if there is 4 it means it is ‘Bike’ otherwise it is ‘Others’.

**Windowing and Feature Extraction:**

1. Initializing the feature dataframe: features = pd.DataFrame()
2. Calculating the number of windows: num\_windows = len(data\_df) // window\_size – overlap
3. Iterating through each window:

* Extracting the window data: window = data\_df.iloc[start\_idx:end\_idx]
* Initializing the window\_features DataFrame: window\_features = pd.DataFrame()
* Feature extraction for each axis:
* Calculating Jitter, Mean Crossing Rate, Mean, Standard Deviation, Variance, Minimum, Maximum, Autocorrelation, Auto Covariance, Skewness, Kurtosis, and RMSE.
* Adding the label column for the window: window\_features["Label"] = [window["Label"].iloc[0]]
* Concatenating window\_features with features: features = pd.concat([features, window\_features], ignore\_index=True)

Saving the features to a CSV file: features.to\_csv('features1.csv')

Our resulting features dataframe contains 281 columns with 933 rows. From 281 columns the 280 columns are for our calculated features and 1 column containing Label for our each window. We have also checked the categories in our calculated Label column and we got 262 bike riders and 671 non-bike sources after iterating through each window of 1 minute.

**Training and Testing:**

* Before splitting the data into training and testing we have created X variable which contains all our 280 features with 933 observation and our y variable have 933 Label which are calculated during feature extraction.
* Splitting the features into training and testing sets: X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42). X\_train has a observation of 746 rows while X\_test has a observation of 187 rows.
* Training and evaluating the Logistic Regression model: logreg\_model = LogisticRegression()

logreg\_model.fit(X\_train, y\_train)

* Training and evaluating the Random Forest model:

rf\_model = RandomForestClassifier()

rf\_model.fit(X\_train, y\_train)

* Training and evaluating the XGBoost model:

xgb\_model = XGBClassifier()

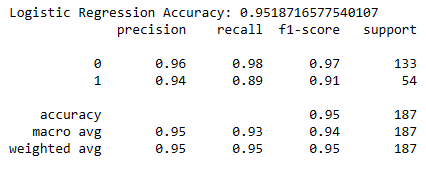
xgb\_model.fit(X\_train, y\_train)

**Printing the Accuracy:**

* Printing the accuracy of the Logistic Regression model on the test data: print("Logistic Regression Accuracy:", logreg\_score).

We got 95.18% accuracy on the test data and 97.72% accuracy on the training data.

Also I have added classification report to check the accuracy, macro average, weighted average.



* Printing the accuracy of the Random Forest model on the test data: print("Random Forest Accuracy:", rf\_score).

We got 100% accuracy on the test data and also 100% accuracy on the training data.

Also I have added classification report to check the accuracy, macro average, weighted average.

A picture containing text, screenshot, font, number

Description automatically generated

* Printing the accuracy of the XGBoost model on the test data: print("XGBoost Accuracy:", xgb\_score)

We got 100% accuracy on the test data and also 100% accuracy on the training data.

Also I have added classification report to check the accuracy, macro average, weighted average.

A picture containing text, screenshot, font, number

Description automatically generated