# **A Machine Learning Approach to Wi-Fi Fingerprint based Localization**

Automatic user localization consists of estimating the position of the user (latitude, longitude and altitude) by using an electronic device, usually a mobile phone. Outdoor localization problem can be solved very accurately thanks to the inclusion of GPS sensors into the mobile devices. However, indoor localization is still an open problem mainly due to the loss of GPS signal in indoor environments. With the widespread use of Wi-Fi communication in indoor environments, Wi-Fi or wirless local area network (WLAN) based positioning gained popularity to solve indoor localization.

WLAN-based positioning systems utilize the Wi-Fi received signal strength indicator (RSSI) value. In this project, we focus on fingerprint-based localization. Fingerprinting technique consists of *two phases*: calibration and positioning. In the calibration phase, an extensive radio map is built consisting of RSSI values from multiple Wi-Fi Access Points (APs) at different *known*locations. This calibration data is used to train the localization algorithm. In the positioning phase, when a user reports the RSSI measurements for the multiple APs, the fit algorithm predicts the user position.

A key challenge in wireless localization is that RSSI value at a given location can have large fluctuations due to Wi-Fi interference, user mobility, environmental mobility etc. In this project, we design, implement and evaluate machine learning algorithms for WLAN fingerprint-based localization.

For comparison, we utilize the [UJIIndoorLoc dataset](https://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc)[1], the first publicly available dataset created for benchmarking WLAN fingerprint localization algorithms. The dataset consists of 19937 calibration records and 1111 positioning records. Each Wi-Fi fingerprint record consists of user ID, timestamp, received signal strength intensity of 520 Access Points and location information including latitude, longitude, floor etc.

**Project Tools:**

● Language: Python

● Process: Review Research Paper, Exploratory Data Analysis, Prediction Algorithms, Feature Selection, Model Validation and Selection, Pipelining

● Tools used: Jupyter Notebook, Docker, Sublime text

**Team:**

Sreerag Mandakathil Sreenath

001838559

[mandakathil.s@husky.neu.edu](mailto:mandakathil.s@husky.neu.edu)

Shreya Chudasama

001828562

[chudasama.s@husky.neu.edu](mailto:chudasama.s@husky.neu.edu)

Aahana Khajanchi

001824402

[khajanchi.a@husky.neu.edu](mailto:khajanchi.a@husky.neu.edu)

**GitHub Link:**

**1.b Libraries Used –**

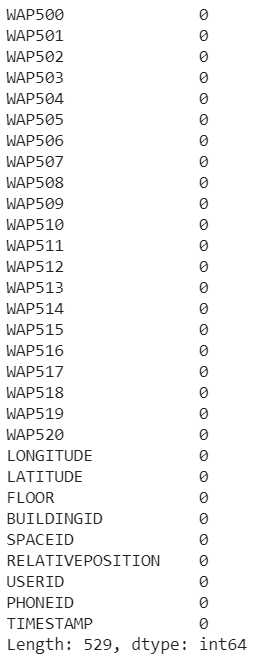
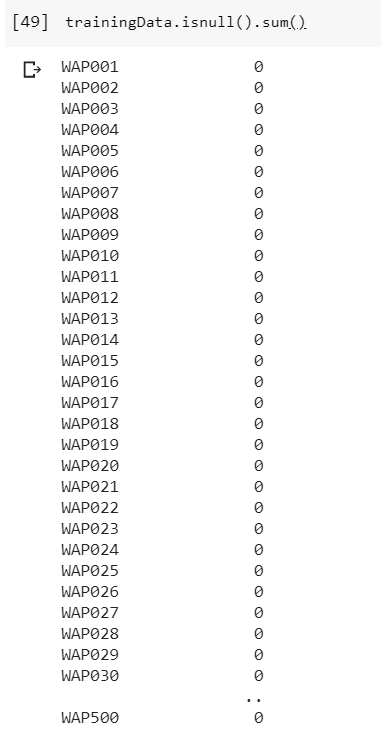
|  |  |  |  |
| --- | --- | --- | --- |
| **Plotting** | **Wrangling** | **EDA Tools** | **Others** |
| Seaborn | Numpy | MissingNo | Pickle |
| Plotly | Pandas | Pivottablejs |  |
| Scipy | Sklearn | PandasProfiling |  |
| Matplotlib |  |  |  |

**1.c Description of the Data columns –**Source: <https://www.kaggle.com/giantuji/UjiIndoorLoc>

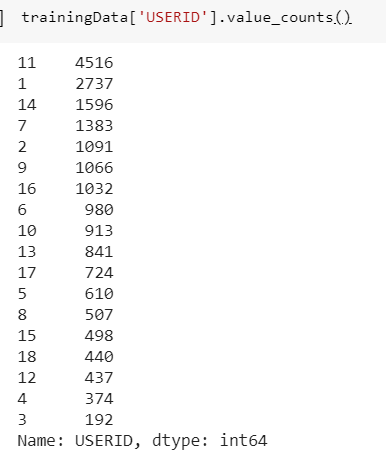
* **WAP001-WAP520**: Intensity value for **Wireless Access Point** (WAP). WAP will be the acronym used for rest of this notebook. Negative integer values from -104 to 0 and +100. **Censored data:** Positive value 100 used if WAP was not detected.
* **Longitude**: Longitude. Negative real values from -7695.9387549299299000 to -7299.786516730871000
* **Latitude**: Latitude. Positive real values from 4864745.7450159714 to 4865017.3646842018.
* **Floor**: Altitude in floors inside the building. Integer values from 0 to 4.
* **BuildingID**: ID to identify the building. Measures were taken in three different buildings. Categorical integer values from 0 to 2.
* **SpaceID**: Internal ID number to identify the Space (office, corridor, classroom) where the capture was taken. Categorical integer values.
* **RelativePosition**: Relative position with respect to the Space (1 - Inside, 2 - Outside in Front of the door). Categorical integer values.
* **UserID**: User identifier (see below). Categorical integer values.
* **PhoneID**: Android device identifier (see below). Categorical integer values.
* **Timestamp**: UNIX Time when the capture was taken. Integer value.

**1.d Missing Data Handling –**

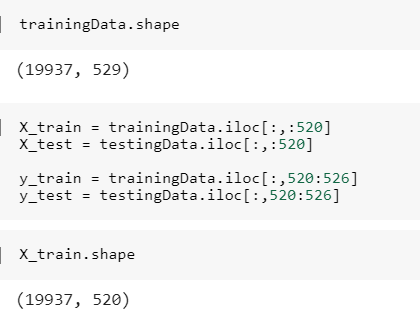
In this dataset, there is no missing data. The dataset is very clean.



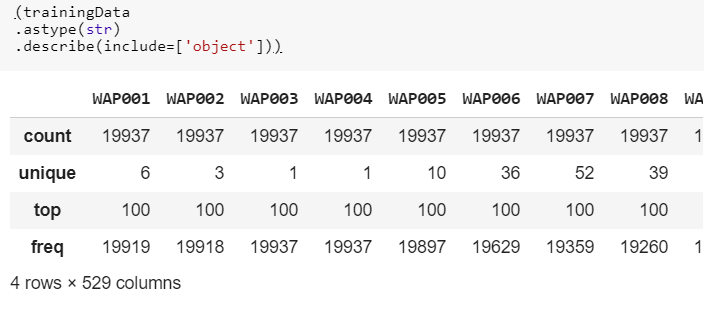
Calculated the number of users.



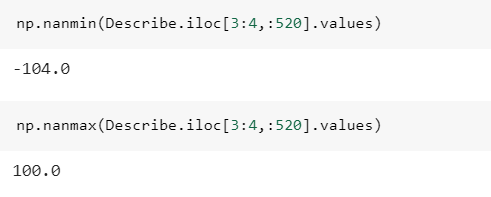
Splitting the dataset into 2. The X\_train will contain all the WAP i.e. Access points and the rest in y\_train.



For object data (e.g. strings or timestamps), the results index will include count, unique, top, and freq. The top is the most common value. The freq is the most common value’s frequency. Timestamps also include the first and last items.



The WAP ranges from -104 to +100



**Skewness and Kurtosis –**

• **Skewness**: measure of lack of symmetry. Skewness for normal distribution should be zero. Negative skewness indicates skew left and positive skewness indicates skew right.

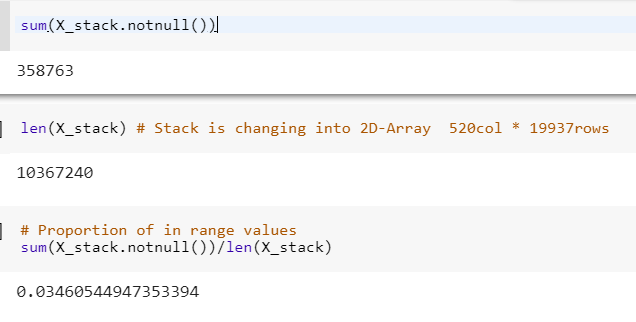
• **Kurtosis**: measure of whether the data are heavily tailed relative to the normaldistribution.



**Observation**

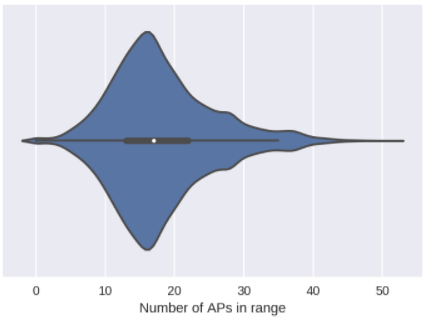
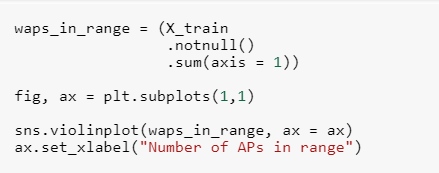
The distribution above appears right-skewed with majority of the values being on the left side of the distribution. We will perform a log/ Box-Cox transformation to overcome the skewness.

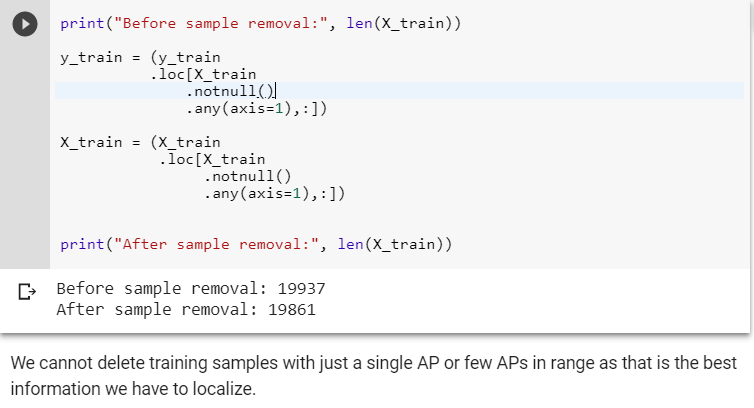
Calculation of percentage of overall out-of-range values .

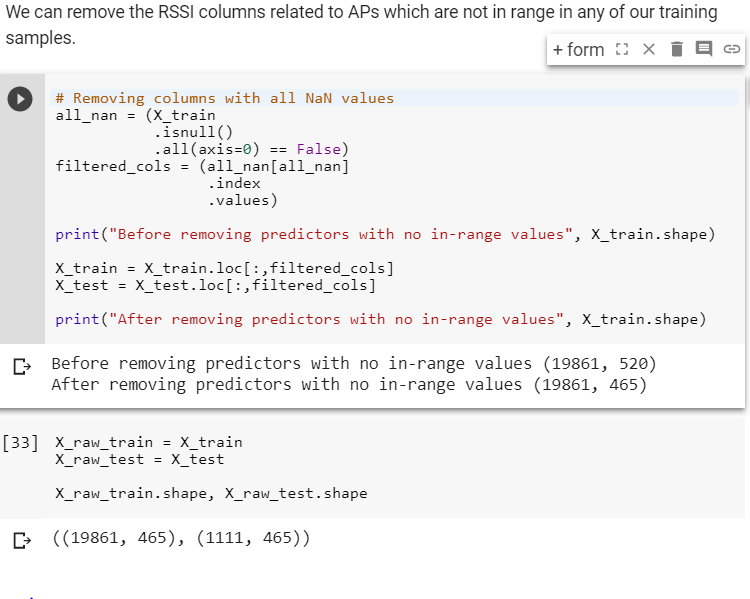


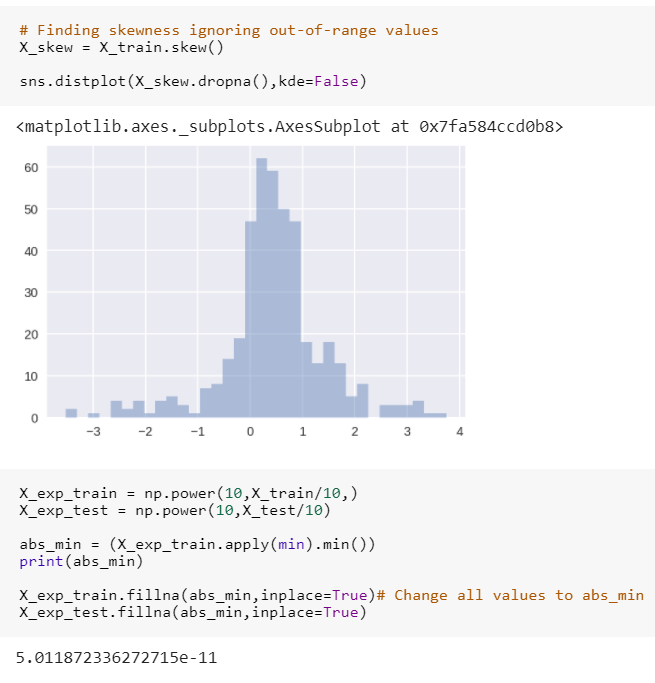
96.1% of the values in the matrix represent Out-of-Range. This is expected as for any given measurement, only a subset of the APs might be in reach of the mobile device.

**Plotting the graph for the distribution of number of APs in range for the training data samples.**



Interestingly, majority of the samples have over 13 APs in range with the maximum of 51 APs. We do observe some of the training samples with 0 APs in range. Let's remove these samples from the training data.





As you can see above majority of the WAP columns have a low to medium positive skewness in the region (0,1). There are still a few columns outside the (1,-1) range. (1,-1) is considered an acceptable range of skewness.

Next, before we apply the Normality tests, we need to fill in the out-of-range values which are currently NaN. Box-Cox transformation requires all values to be positive. For this purpose, let's transform our predictors to normal scale from the dBm scale.

Also, the out-of-range values are transformed to 1/hundreth of the absolute minimum among all in-range values. Therefore, the transformed out-of-range value represents the minimum RSSI value in the dataset.

# **Normality test**

In this sub-section, I will explore various normality tests and explore the distributions of RSSI predictors.

Skew test: Tests whether the skew is different from the normal distribution. The Z-score of the test in which the null hypothesis states that that the skewness of the population that the sample was drawn from is the same as that of a corresponding normal distribution.

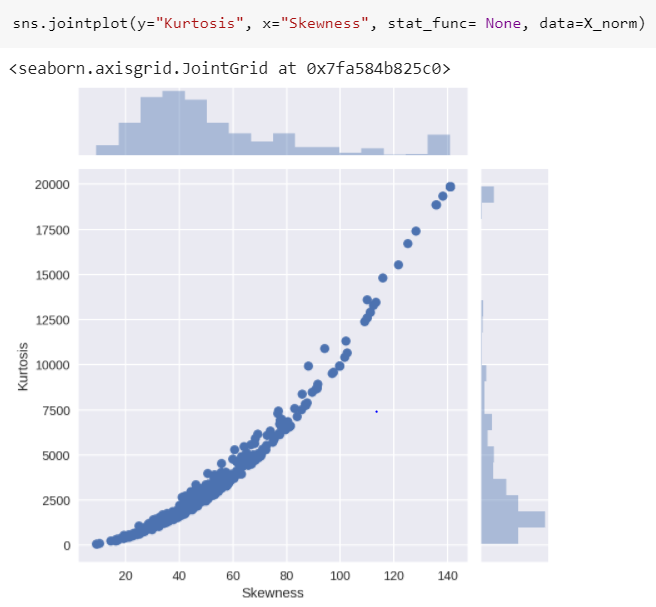
Kurtosis test: Z-score of the test in which the null hypothesis states that the kurtosis of the population from which the sample was drawn is that of the normal distribution: kurtosis = 3(n-1)/(n+1).

k2: s2+k2, where s is the z-score returned by skew test and k is the z-score returned by kurtosistest.

p\_value: A 2-sided chi squared probability for the hypothesis test that the sample comes from a normal distribution. Same test as k2.



**The relationship between Kurtosis scores and Skew scores.**



In the above figure, for the columns with a higher skewness score, the kurtosis is also more extreme.

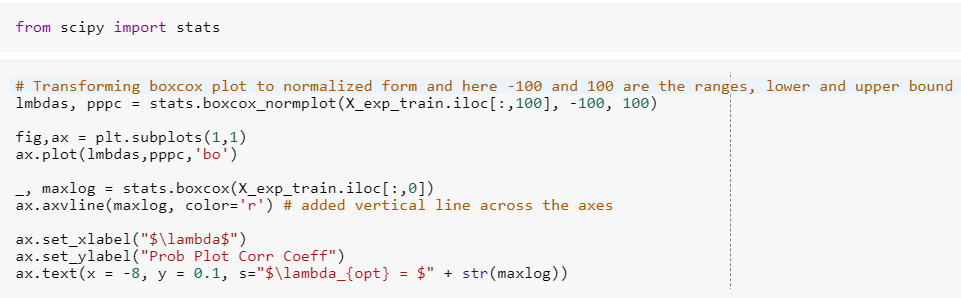
# 

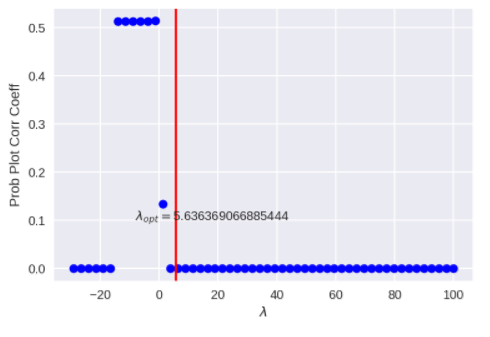
# 

# **Box-Cox Transformation**

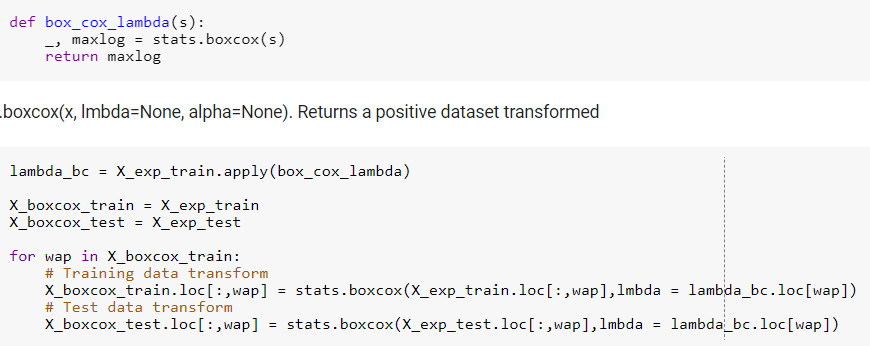
Data that doesn't follow a known distribution, such as the normal distribution, is often difficult to manage. A popular strategy to get control of the data is to apply the Box-Cox transformation. The transformation relies primarily on a lambda (ƛ) variable that holds a value between -1 and 1 that is automatically calculated to be optimal for your data.

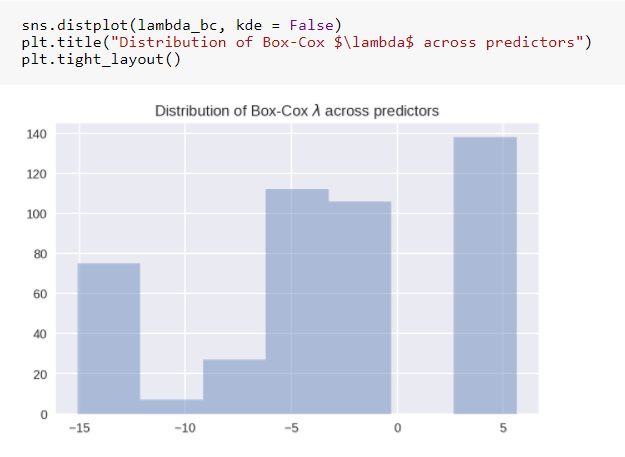
A Box-Cox normality plot shows graphically what the best transformation parameter is to use in boxcox to obtain a distribution that is close to normal. Probability Plot Correlation Coefficient, as obtained from probplot when fitting the Box-Cox transformed input x against a normal distribution.



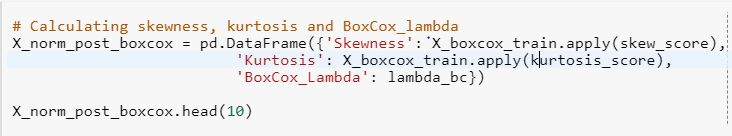


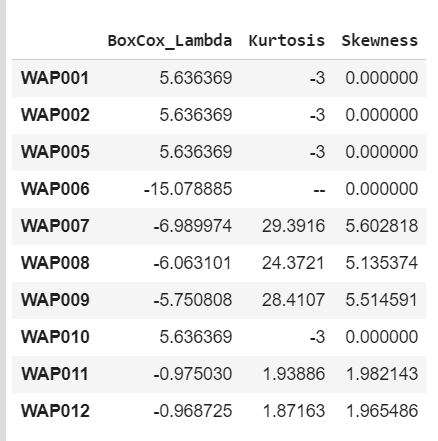
The maximum log likelihood function peaks at λ=1.1437.



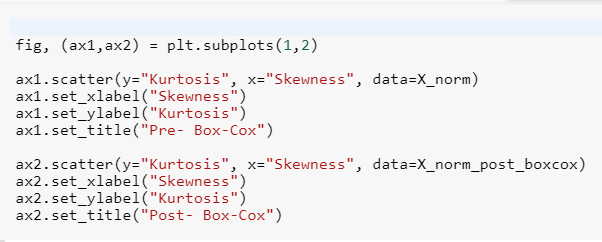


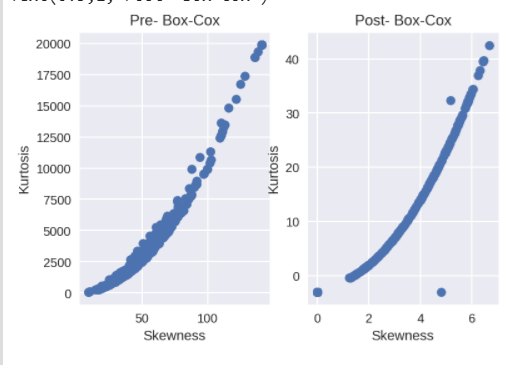
The above figure shows the distribution of λs that maximize log-likelihood function for each predictor. We can observe the two biggest bars are located at +5 and -2.5.





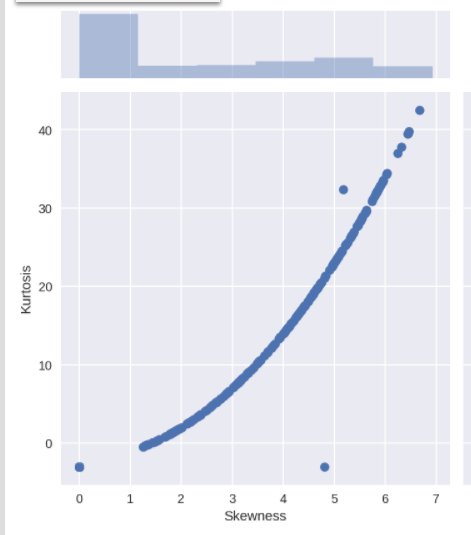
From the above table, we observed that the BoxCox-parameter values lie between -2.5 to +5. Also,The kurtosis and skewness seems to have greatly reduced compared to before the Box-Cox transformation.





From the above two, we found that the distribution is normalized which reduced the skewness as well as Kurtosis.





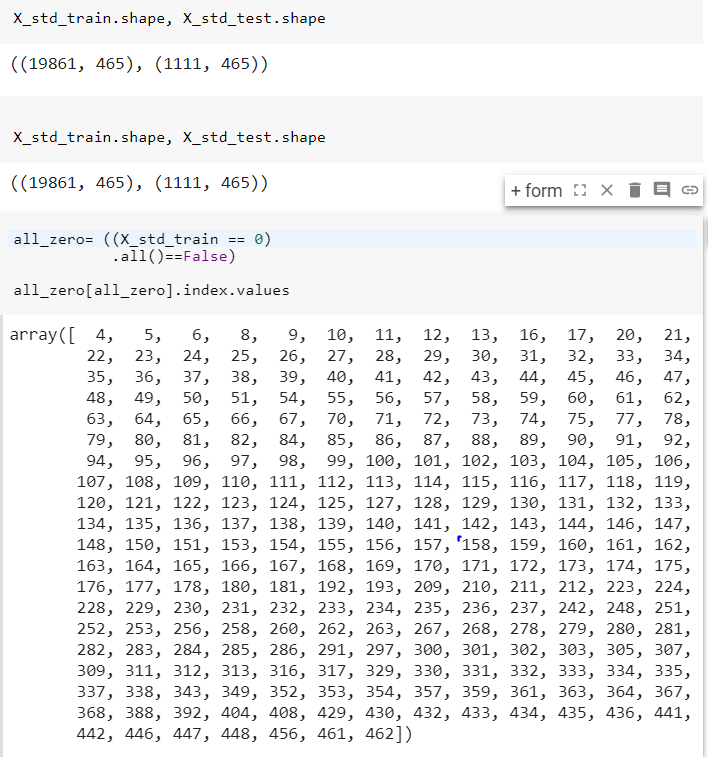
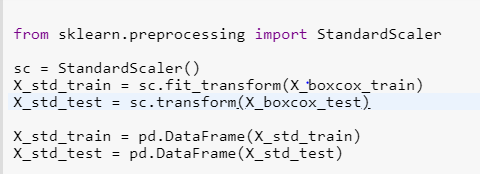
We can observe the biggest bars are located in the region [0,1) for skewness and [0,-3) for kurtosis.

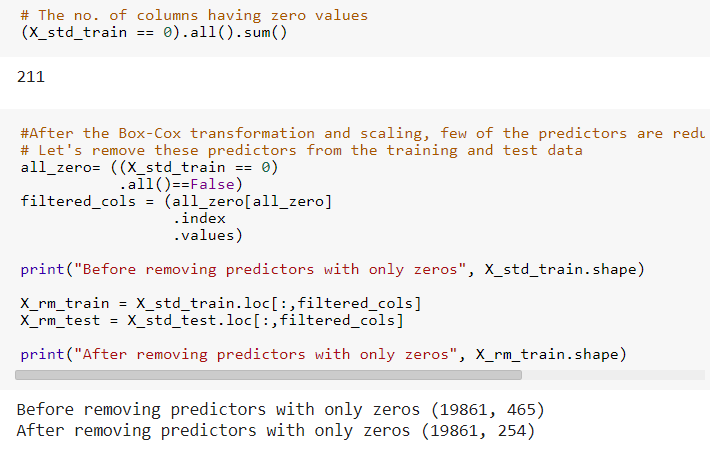
At this point, we can remove the predictors that have high skewness (> 3) or high kurtosis (> 1). However, as predictor corresponds to RSSI distribution of a single WAP, there might be samples for which one or more of these WAPs might be the closest and correspondingly have the highest measure. For this purpose, I do not perform any predictor selection at this stage.

Instead, we can explore how much of the variance in the dataset is explained by the predictors using Principal Component Analysis (PCA).

**Dimensionality Reduction**

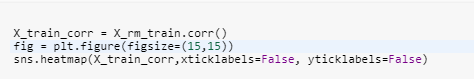
Analyses the correlations between the predictors and remove highly correlated predictors. This is because adjoining nearly correlated variables increases the contribution of their common underlying factor to the PCA. We can remove highly correlated predictors algorithmically or removing the correlations by whitening the data (conversion to Identity Covariance Matrix).

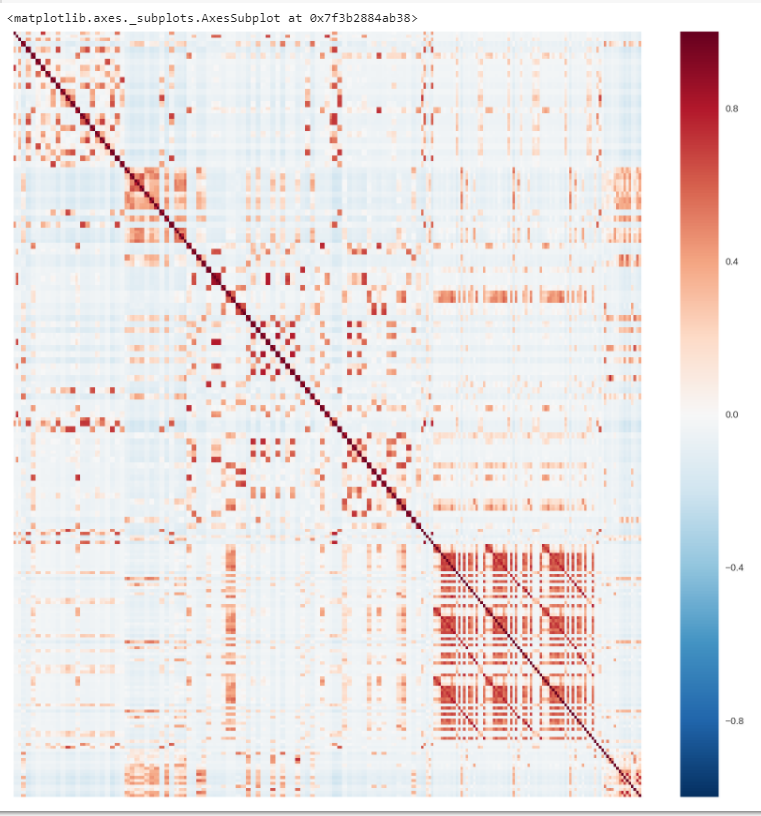




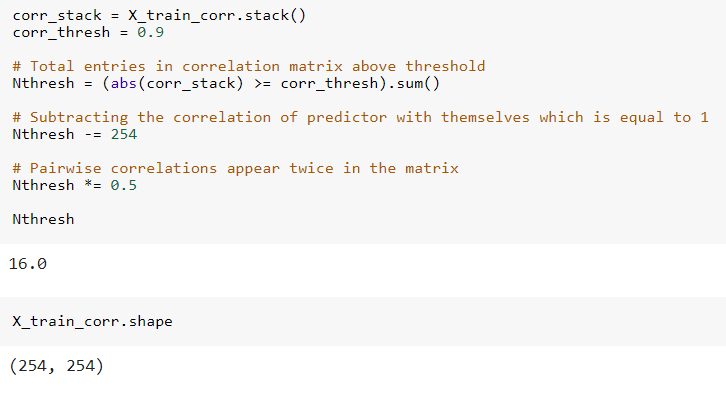
Removed all the fields having zero values.

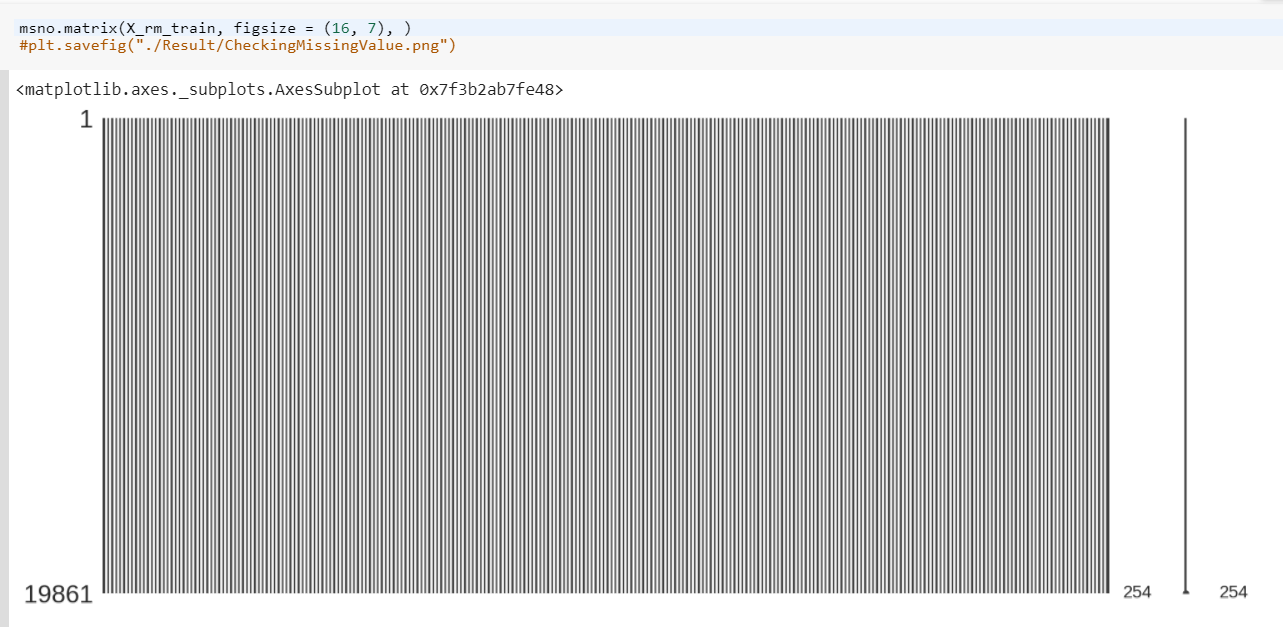
Calculating Correlations





Clearly, we observe clusters of predictors that are highly correlated. Let's assign a threshold of 0.8 and see how many predictor pairs have correlation above this threshold.





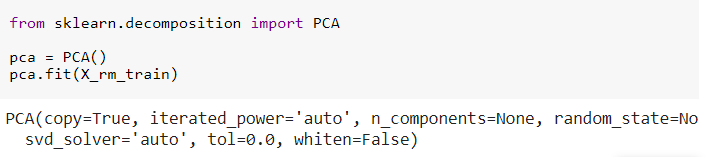
There are no missing values. Our dataset is clean.

**Principal Component Analysis (PCA)**

Dimensionality reduction is one of the key techniques to reduce the complexity.

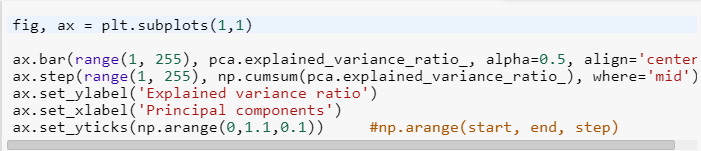
PCA is a simple dimensionality reduction technique that applies linear transformations on the original space. Among all the orthogonal linear projections, PCA minimizes the reconstruction error, which is the distance between the instance and its reconstruction from the lower-dimensional space. That is sum of the distances between points in original space and the corresponding points in lower-dimensional space.

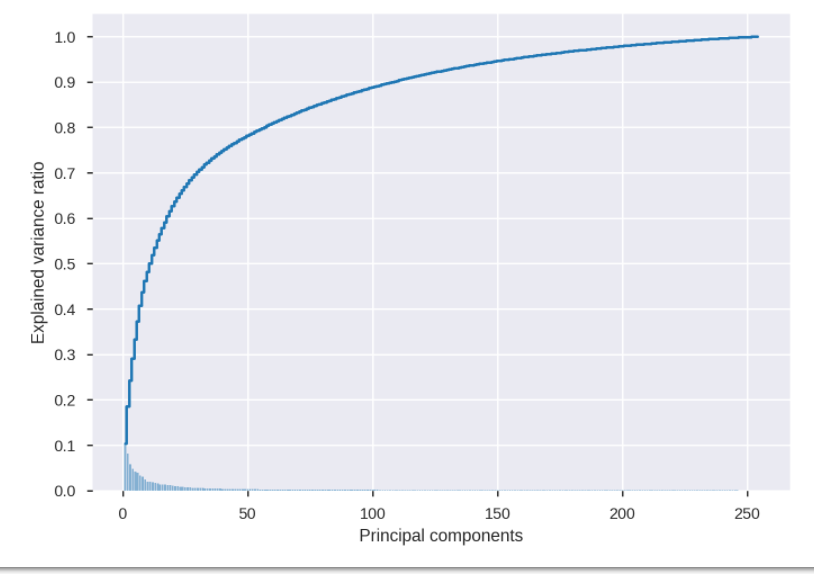
An important point to remember about PCA is that it is an unsupervised form of dimensionality reduction. This means the response variables are not taken into consideration at any point of the transformation. sci-kit learn provides convenient methods to perform PCA which I'll be using directly.



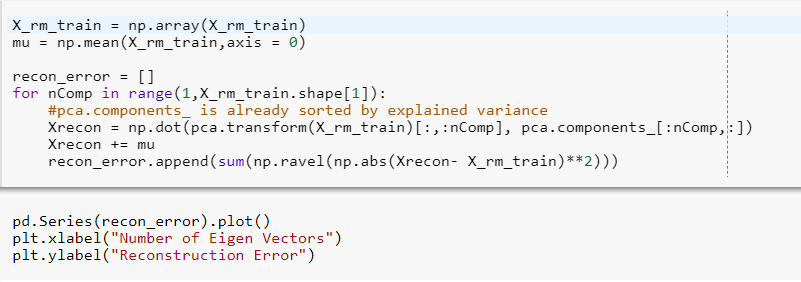
The pca.explained\_variance*ratio* parameter returns a vector of the variance explained by each dimension. Thus pca.explained\_variance*ratio*[i] gives the variance explained solely by the i+1st dimension.

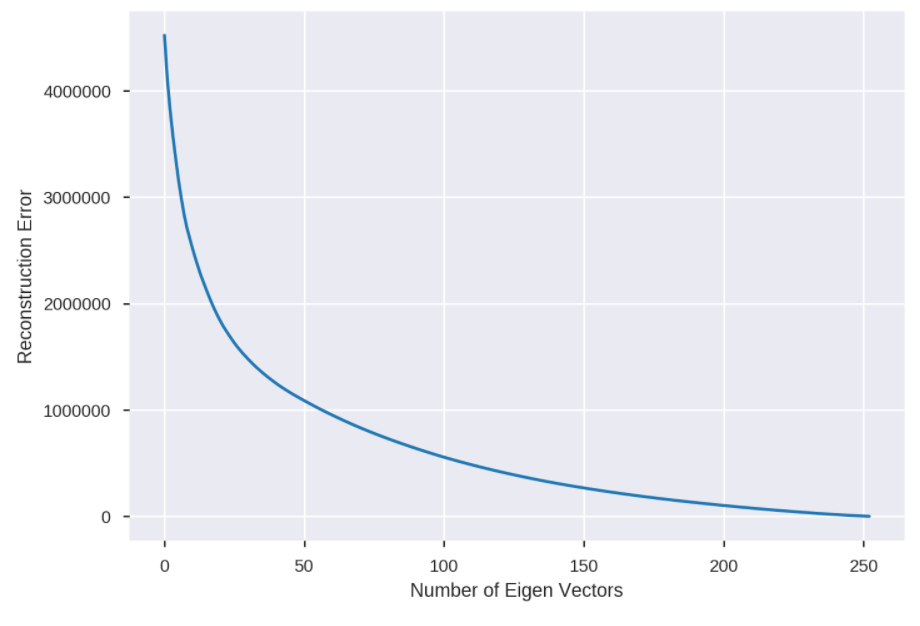
We will do pca.explained\_variance*ratio*.cumsum(). That will return a vector x such that x[i] returns the cumulative variance explained by the first i+1 dimensions.



****

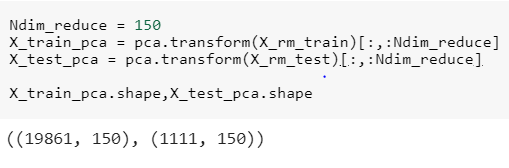
Roughly 95% of the variance is explained by first 150 eigen vectors. Before, we perform the dimensionality reduction on our data, let's analyze the reconstruction error as a function of the dimensions.

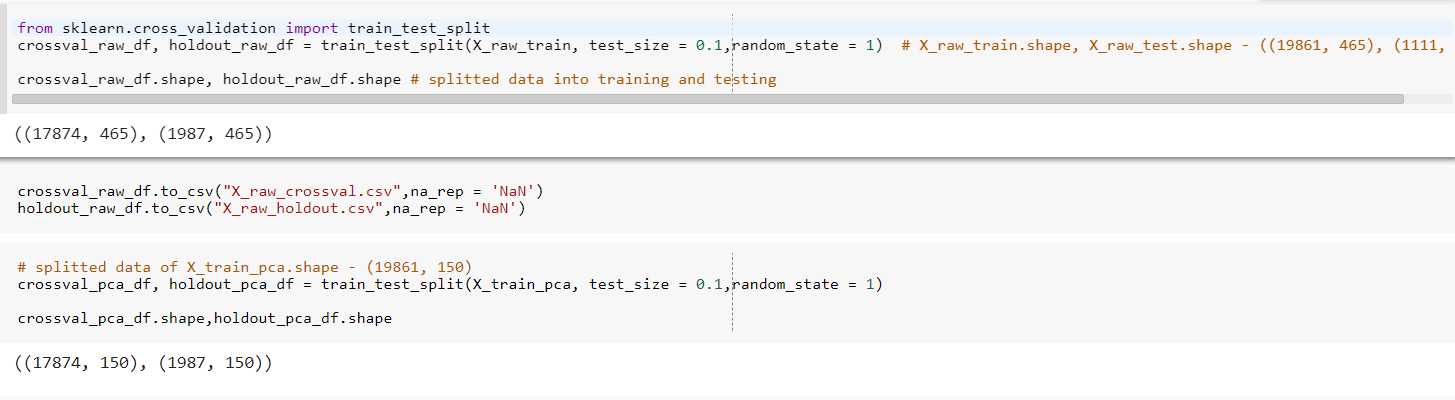


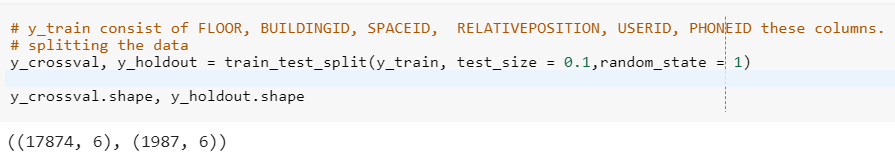


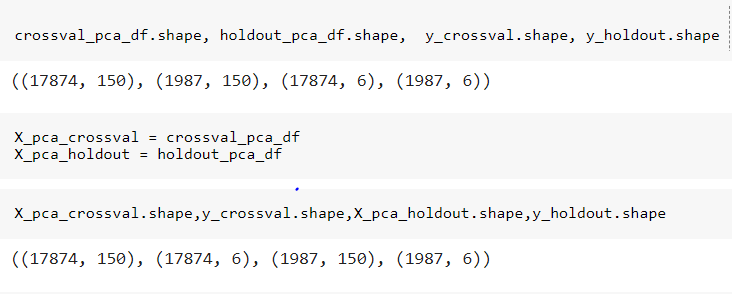
As the number of principal components used for the reconstruction increases, the reconstruction error expectedly decreases. This figure is a mirror image of the previous explained variance ratio figure.

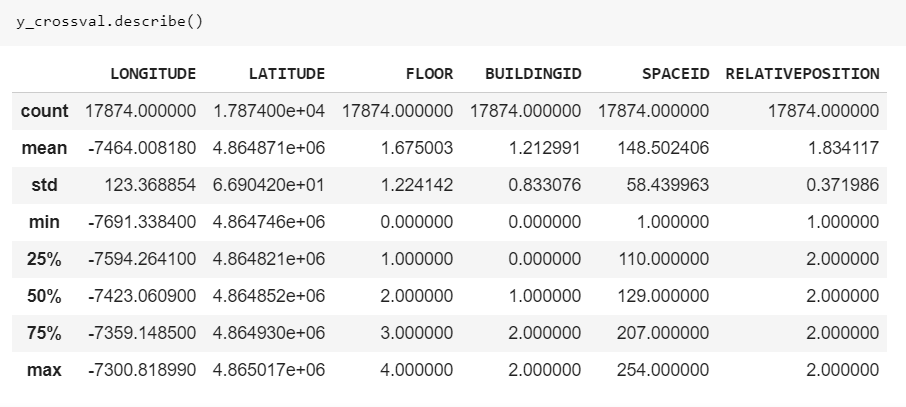
As 95% of the explained variance is explained by top 150 components, we have reduced training and test data to 150 dimensions.

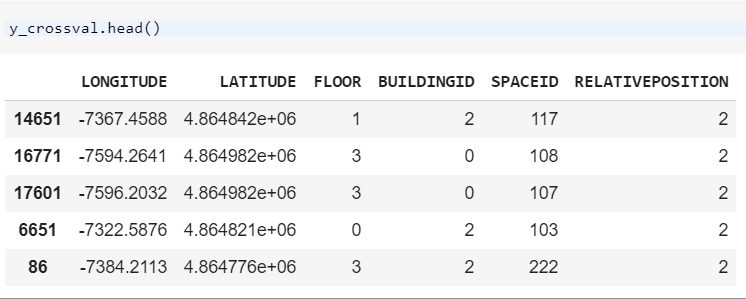






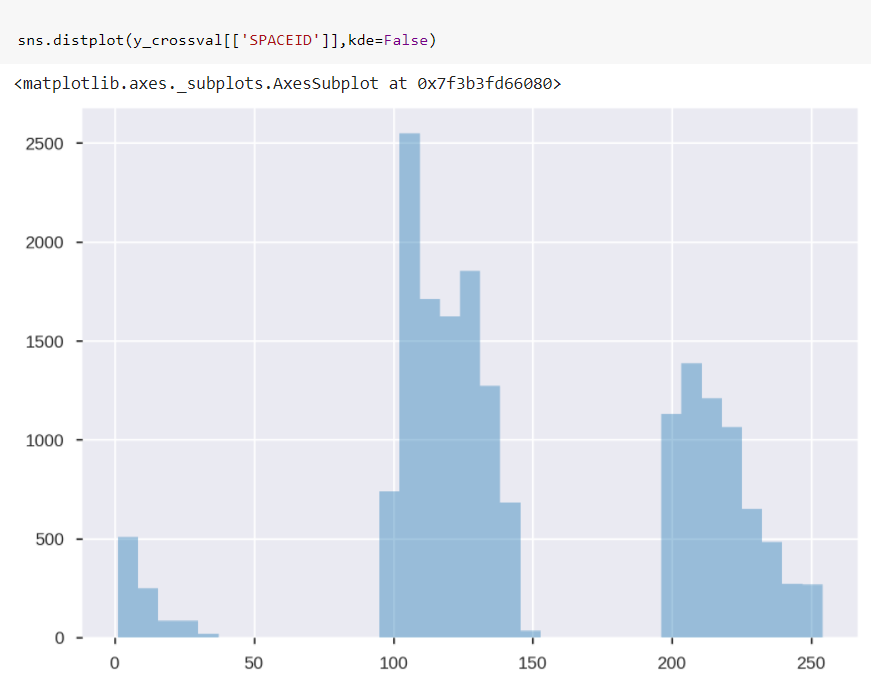


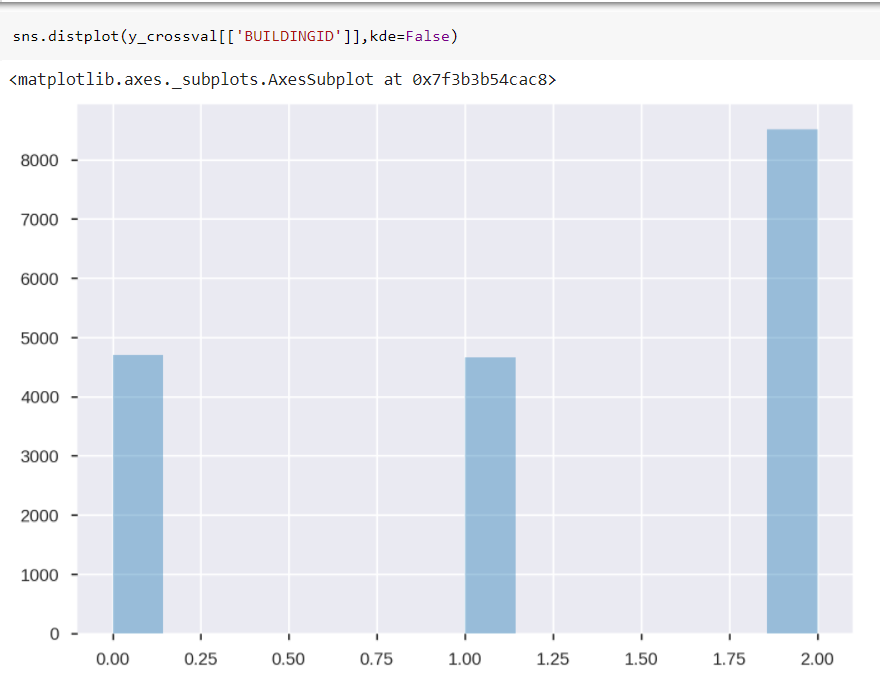




In the next few sections, we explore the characteristics of the different response variables.





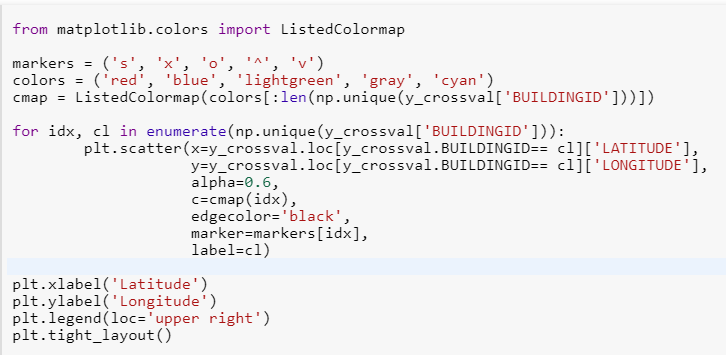


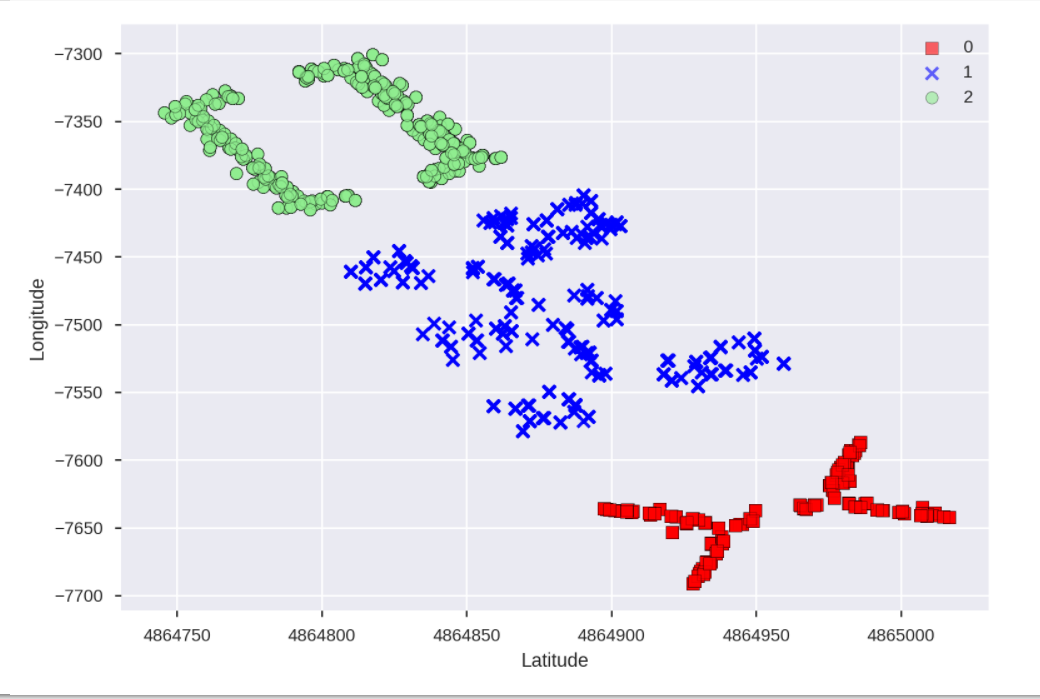
Observations:

In our training samples, building 2 has the clear majority with it's count being slightly lower than the sum of building 0 and building 1.

Building 0 and building 1 have roughly the same representation in the training data.

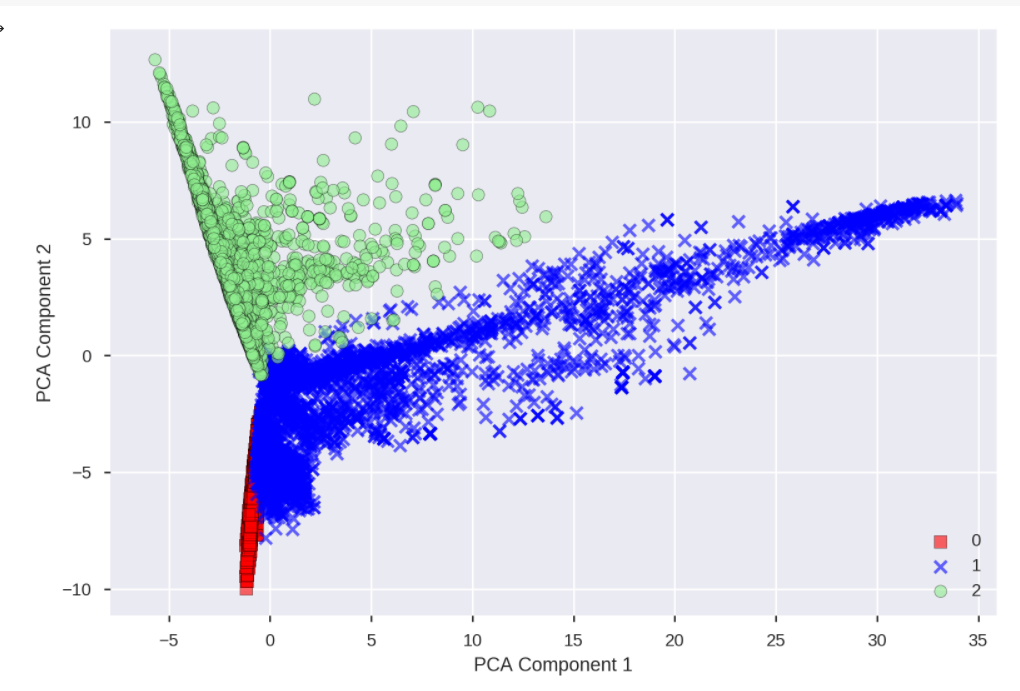
Clearly, there is an imbalance among the groups.





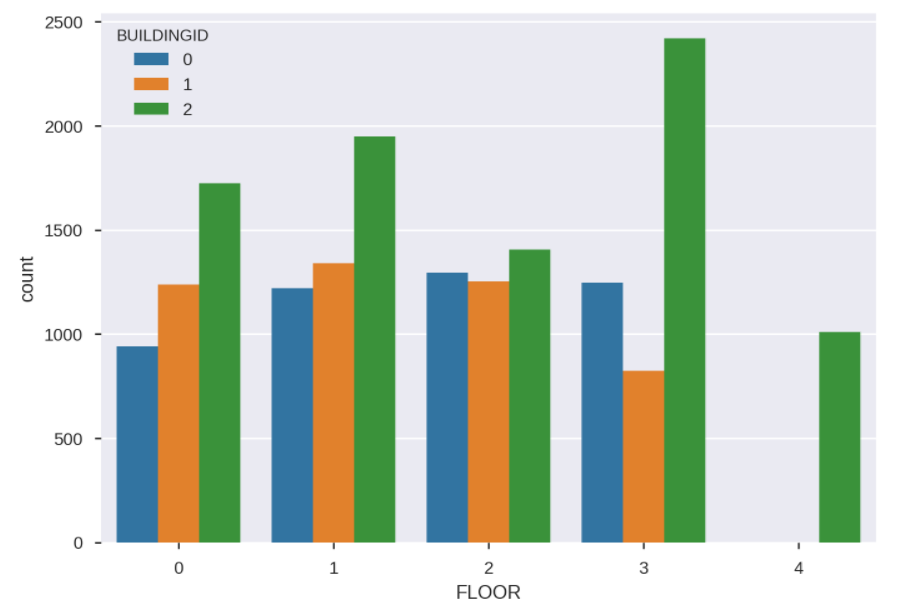
The above plot illustrates the locations of the buildings in the campus.



 The above plot illustrates how the buildingID are distributed across the top two PCA dimensions. Later, I explore the machine learning approaches for the building classification.

PCA is an unsupervised learning technique for dimensionality reduction. So, it is quite possible the two top PCA components might not have explained our response variable well.





Observations:

Buildings 0 and 1 have 4 floors whereas Building 2 has 5 floors.

Expectedly, the samples from Building 2 are consistently the highest across all the floors.