UCI Indoor Locator

04/13/2018

Team :

Sreerag Mandakathil Sreenath 001838559

mandakathil.s@husky.neu.edu

Shreya Chudasama 001828562

chudasama.s@husky.neu.edu

Aahana Khajanchi 001824402

khajanchi.a@husky.neu.edu

Project Tools:

- Language : Python
- Pipeline : Luigi
- Framework : Flask
- Databse : SQL
- Tools used: Jupyter Notebooks and Docker , Xamp.

Github Link:

https://github.com/sreeragsreenath/AdavancedDataScience

Url Link:

http://198.199.74.32:5000/

Overview

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 wireless 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.

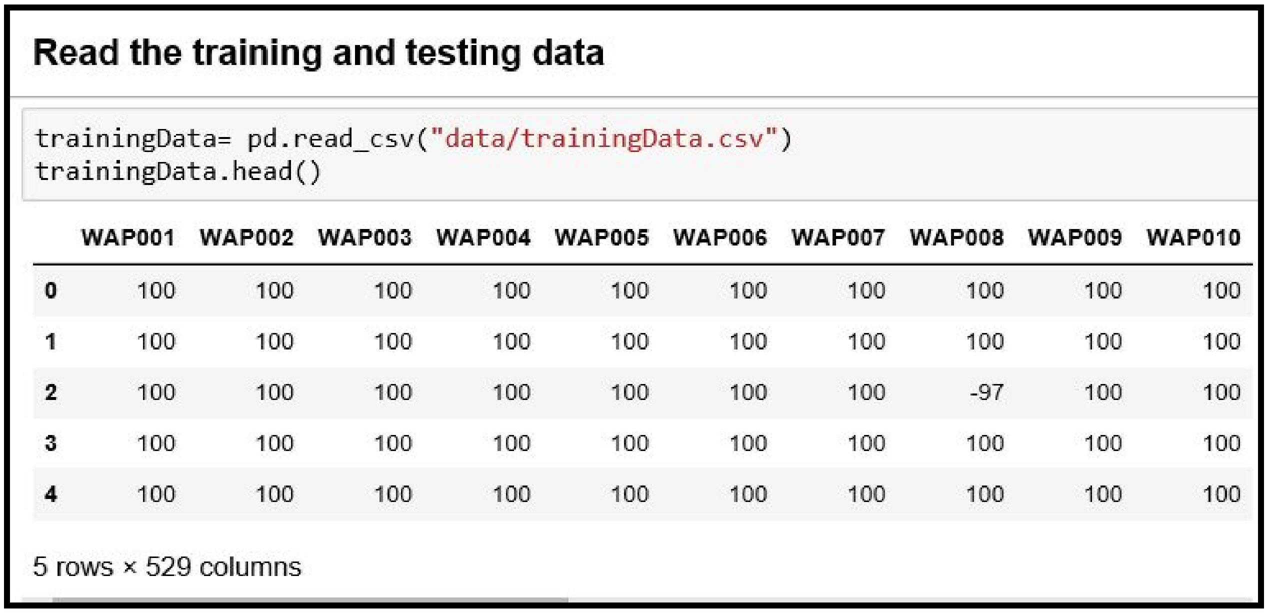
Dataset Description

Source: https://www.kaggle.com/giantuji/UjiIndoorLoc

- **WAP001-WAP520**: Intensity value for **Wireless Access Point** (AP). AP 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

Data Modeling

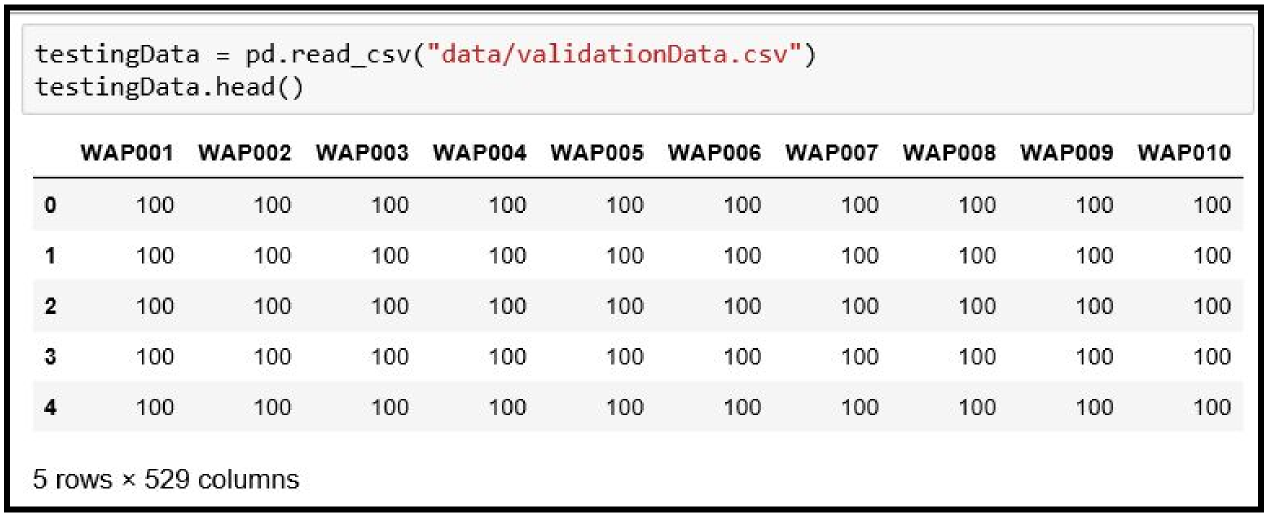


Read the training and testing data

```
trainingData= pd.read_csv("data/trainingData.csv")
trainingData.head()
```

	WAP001	WAP002	WAP003	WAP004	WAP005	WAP006	WAP007	WAP008	WAP009	WAP010
0	100	100	100	100	100	100	100	100	100	100
1	100	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	-97	100	100
3	100	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100	100

5 rows × 529 columns



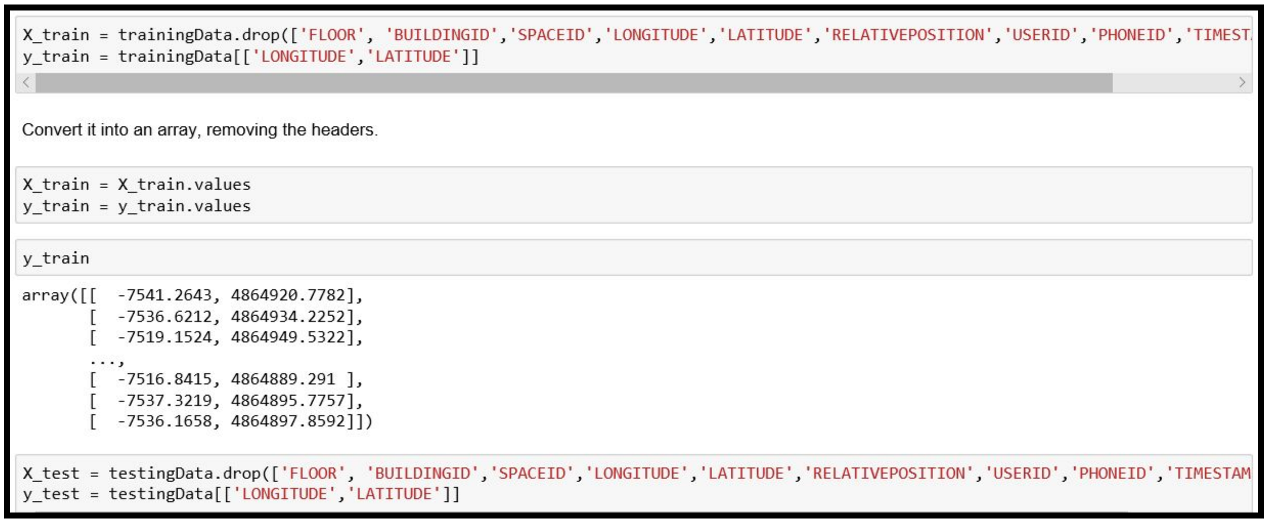
```
testingData = pd.read_csv("data/validationData.csv")
testingData.head()
```

	WAP001	WAP002	WAP003	WAP004	WAP005	WAP006	WAP007	WAP008	WAP009	WAP010
0	100	100	100	100	100	100	100	100	100	100
1	100	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100	100

5 rows × 529 columns

Dropped unnecessary columns.

So here we need to predict the longitude and latitude of GPS, which can be done using the WAP columns.



```
X_train = trainingData.drop(['FLOOR', 'BUILDINGID','SPACEID','LONGITUDE','LATITUDE','RELATIVEPOSITION','USERID','PHONEID','TIMEST
y_train = trainingData[['LONGITUDE','LATITUDE']]
```

Convert it into an array, removing the headers.

```
X_train = X_train.values
y_train = y_train.values
```

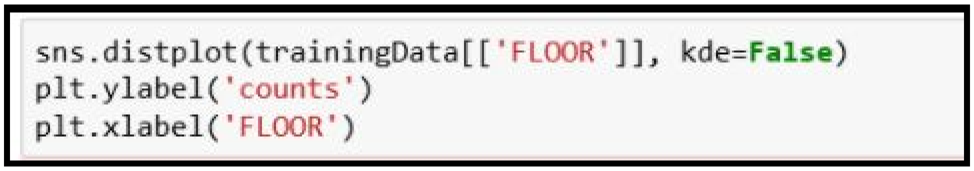
```
y_train
```

```
array([[  -7541.2643, 4864920.7782],
       [  -7536.6212, 4864934.2252],
       [  -7519.1524, 4864949.5322],
       ...,
       [  -7516.8415, 4864889.291 ],
       [  -7537.3219, 4864895.7757],
       [  -7536.1658, 4864897.8592]])
```

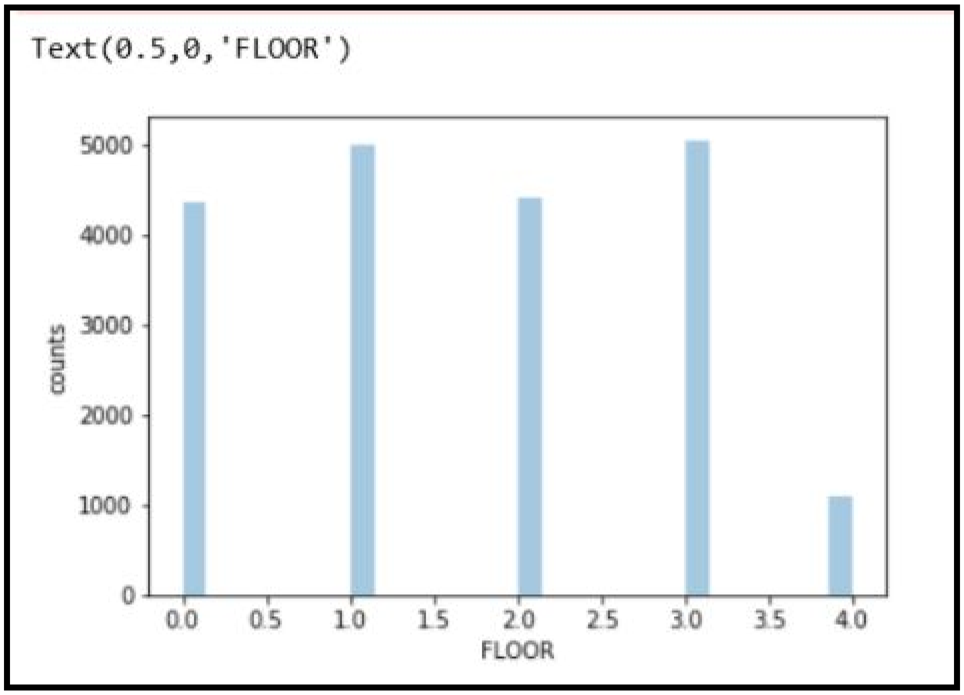
```
X_test = testingData.drop(['FLOOR', 'BUILDINGID','SPACEID','LONGITUDE','LATITUDE','RELATIVEPOSITION','USERID','PHONEID','TIMESTAM
y_test = testingData[['LONGITUDE','LATITUDE']]
```

We will use only WAP columns for predicting Latitude and Longitude.

Plotted Graphs

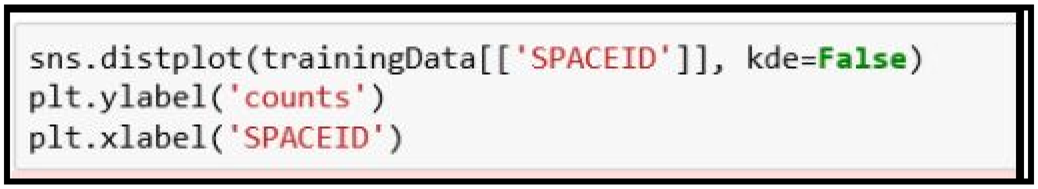


```
sns.distplot(trainingData[['FLOOR']], kde=False)
plt.ylabel('counts')
plt.xlabel('FLOOR')
```



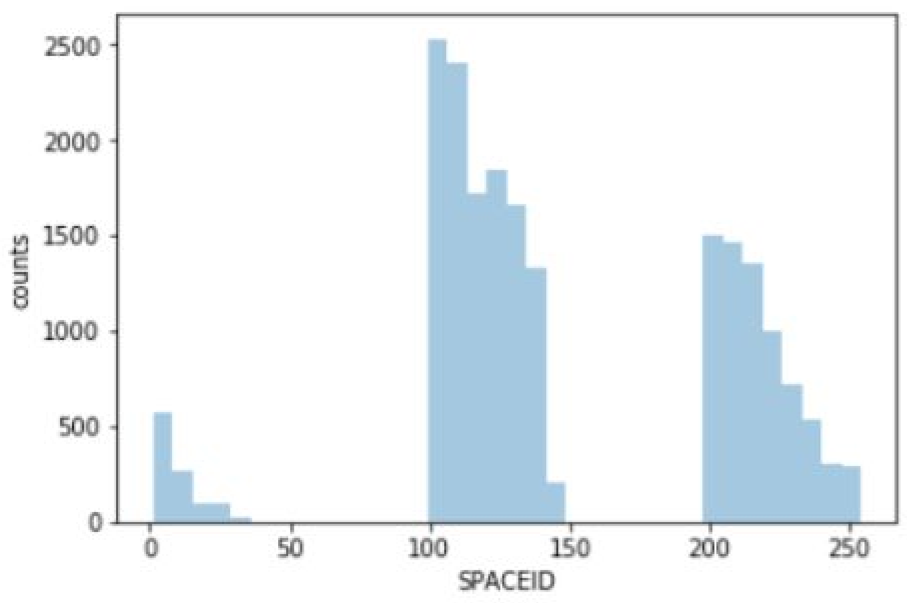
```
Text(0.5,0,'FLOOR')
```

Plotted the number of counts of the floor

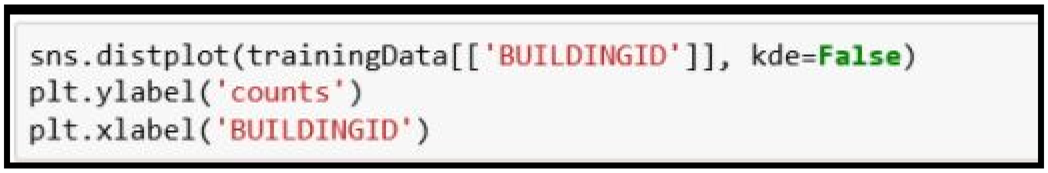


```
sns.distplot(trainingData[['SPACEID']], kde=False)
plt.ylabel('counts')
plt.xlabel('SPACEID')
```

```
Text(0.5,0,'SPACEID')
```

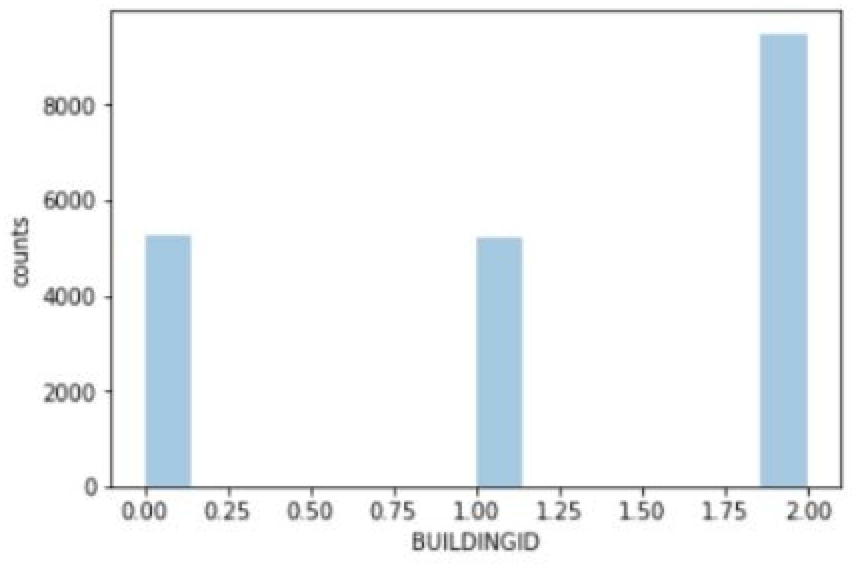


Plotted the number of counts of the spaceid

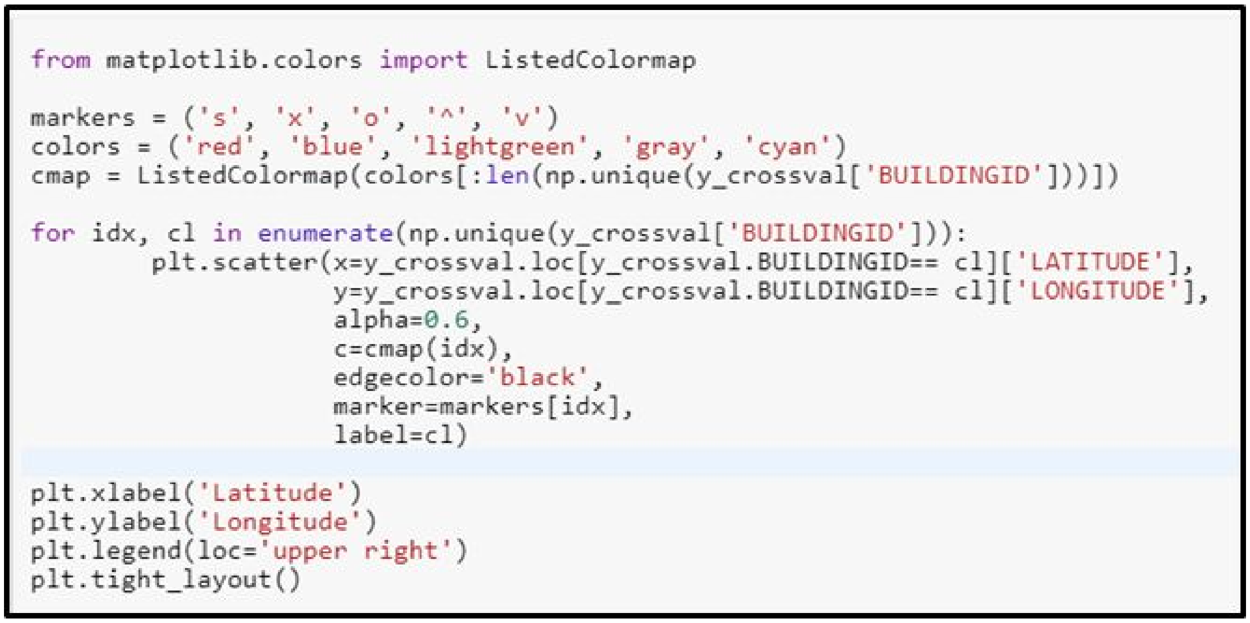


```
sns.distplot(trainingData[['BUILDINGID']], kde=False)
plt.ylabel('counts')
plt.xlabel('BUILDINGID')
```

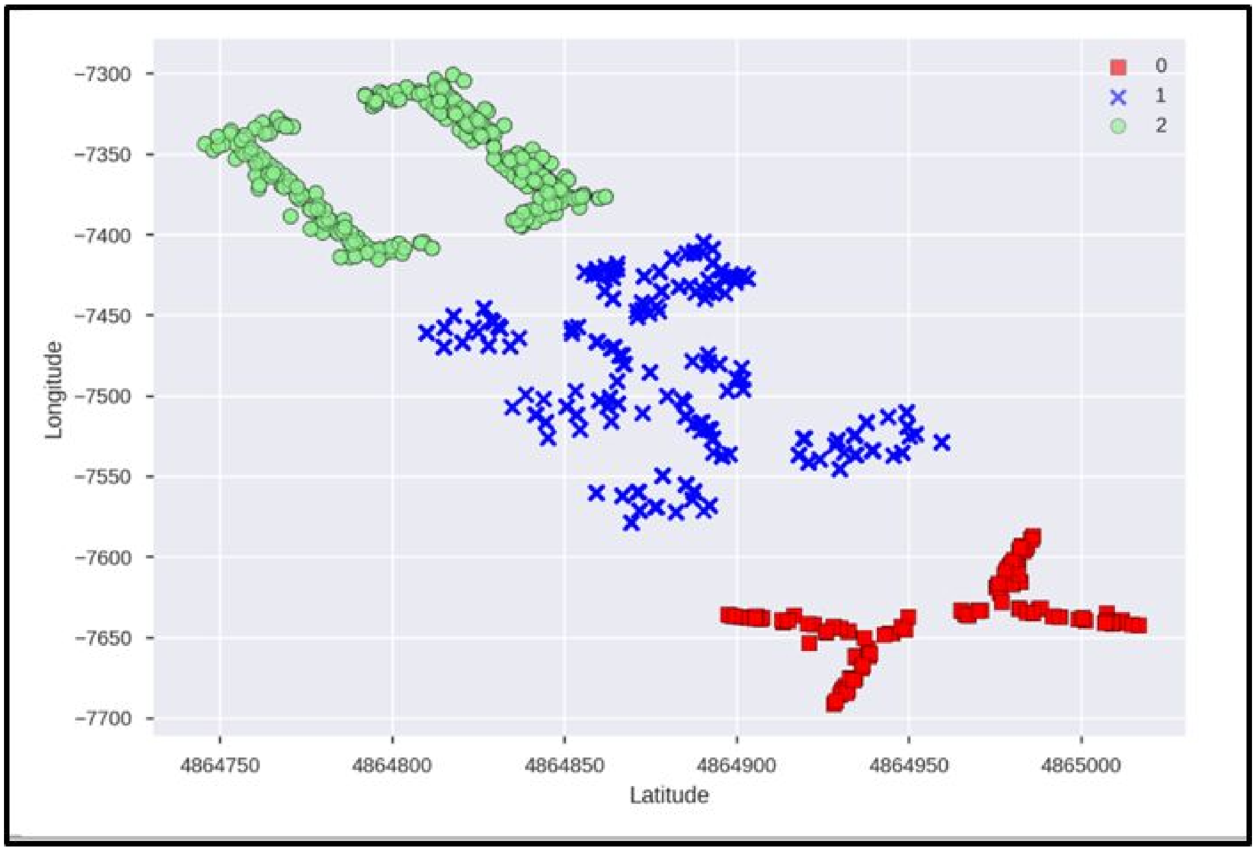
```
Text(0.5,0,'BUILDINGID')
```



Plotted the number of counts of BuildingId

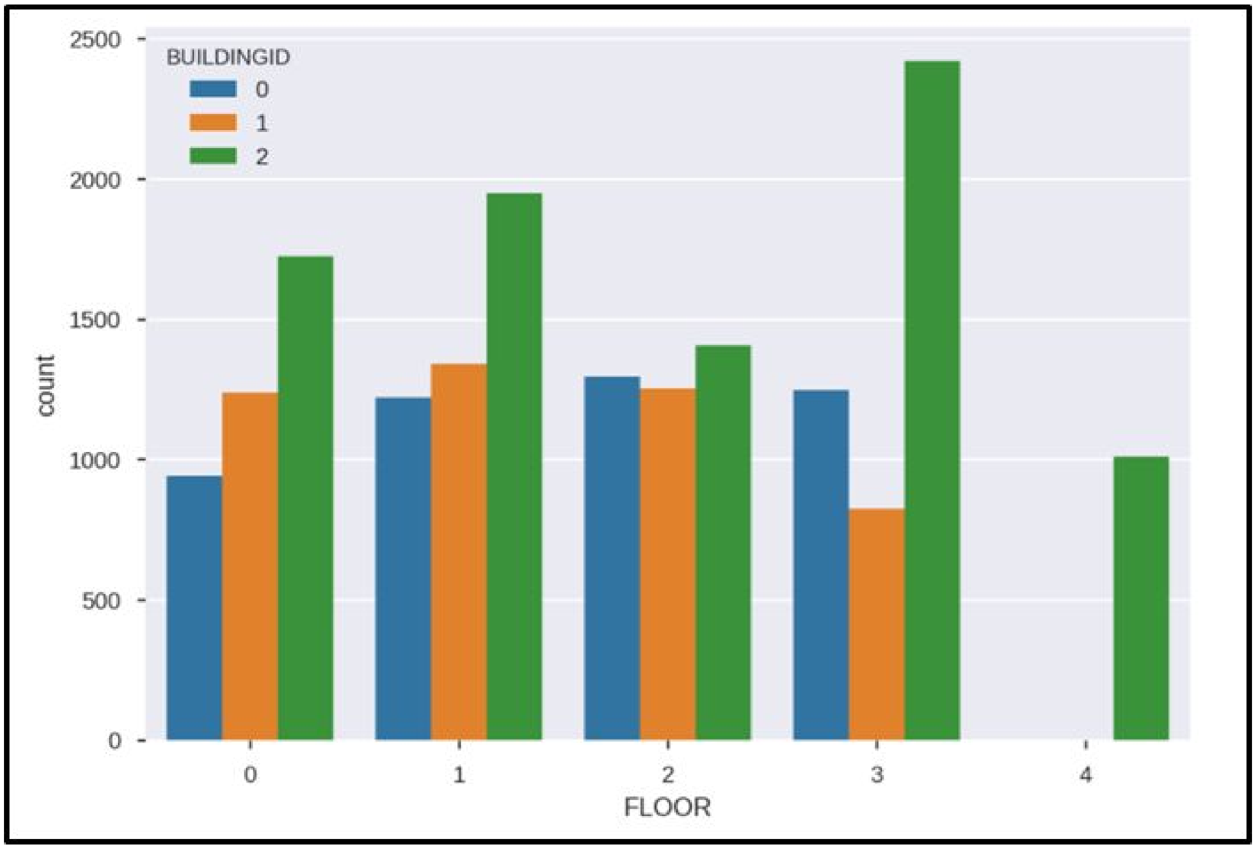


```
from matplotlib.colors import ListedColormap

markers = ('s', 'x', 'o', '^', 'v')
colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
cmap = ListedColormap(colors[:len(np.unique(y_crossval['BUILDINGID']))])

for idx, cl in enumerate(np.unique(y_crossval['BUILDINGID'])):
        plt.scatter(x=y_crossval.loc[y_crossval.BUILDINGID== cl]['LATITUDE'],
                    y=y_crossval.loc[y_crossval.BUILDINGID== cl]['LONGITUDE'],
                    alpha=0.6,
                    c=cmap(idx),
                    edgecolor='black',
                    marker=markers[idx],
                    label=cl)

plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.legend(loc='upper right')
plt.tight_layout()
```



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

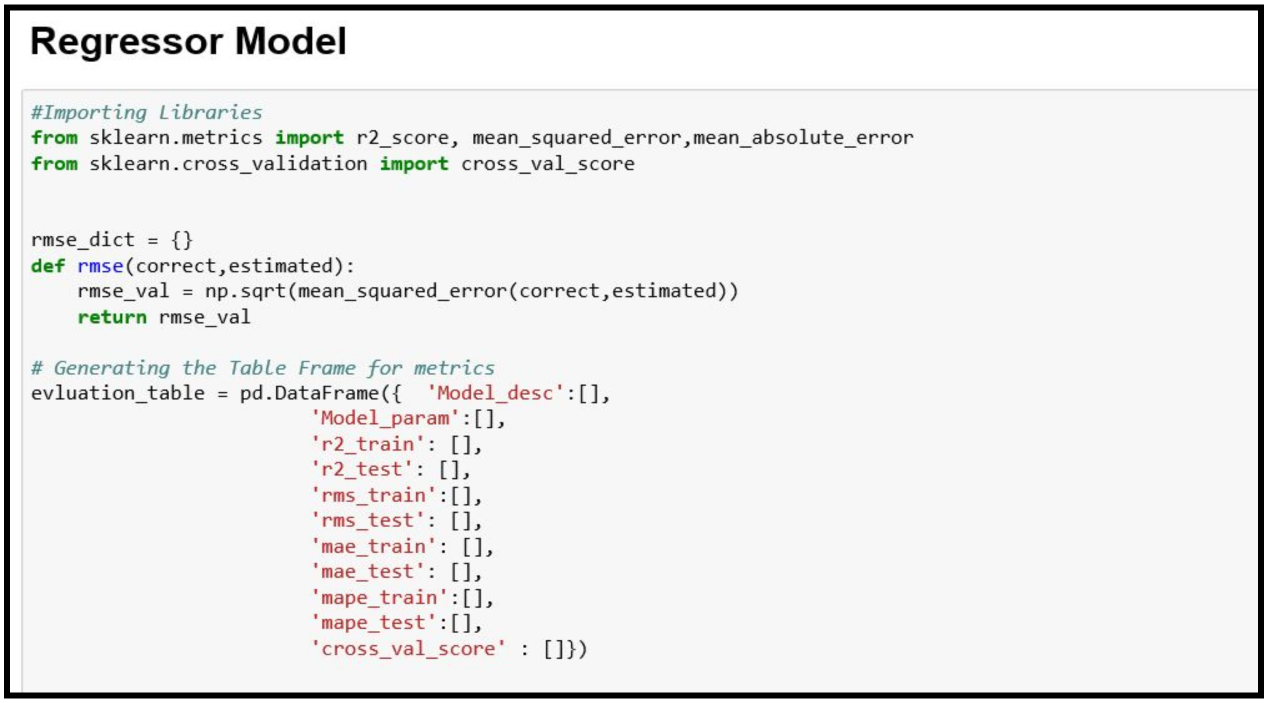
```
sns.countplot(x="FLOOR", hue="BUILDINGID", data=y_crossval,orient="v")
```



Observations:

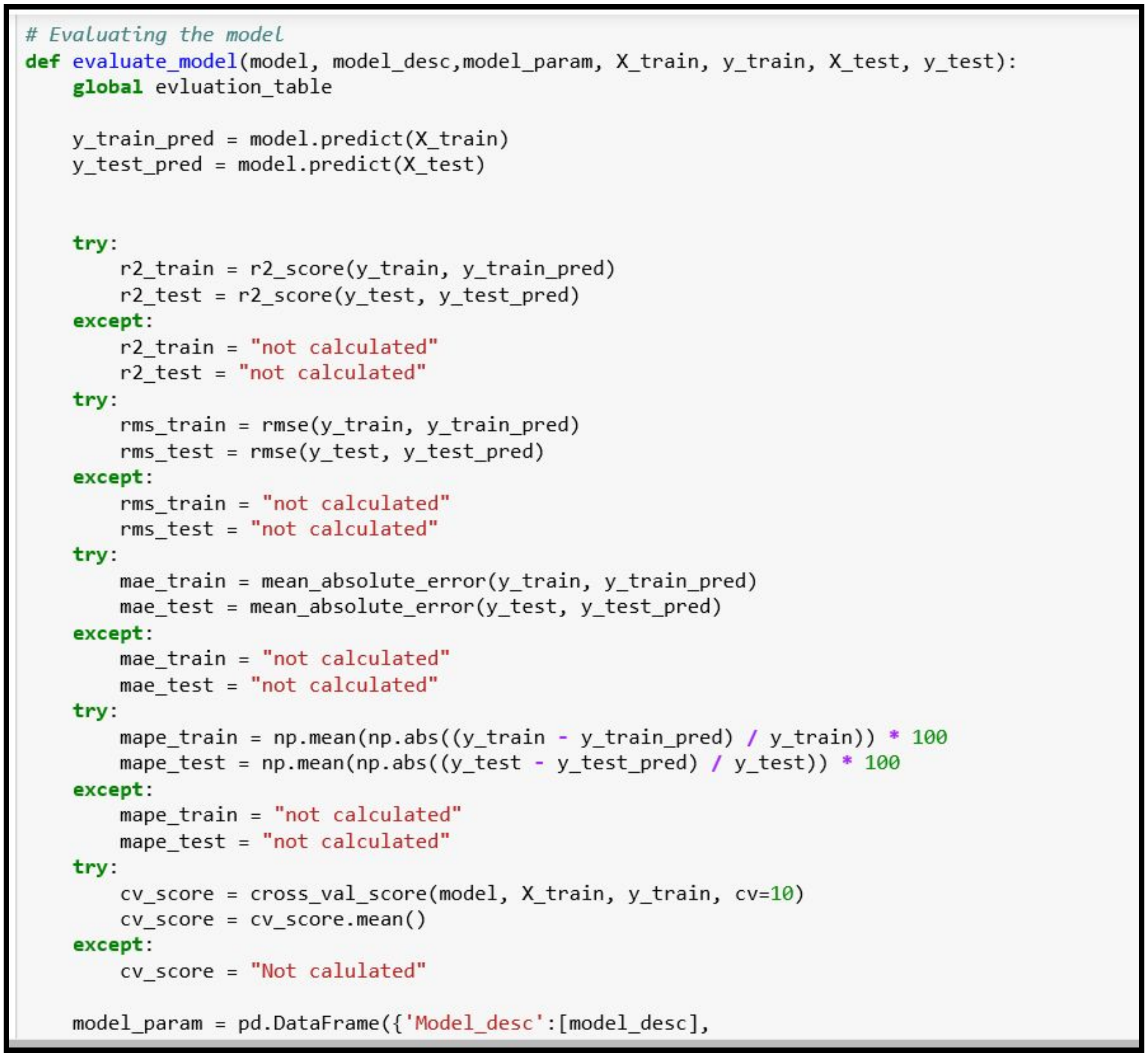
Buildings 0 and 1 have 4 floors whereas Building 2 has 5 floors.

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

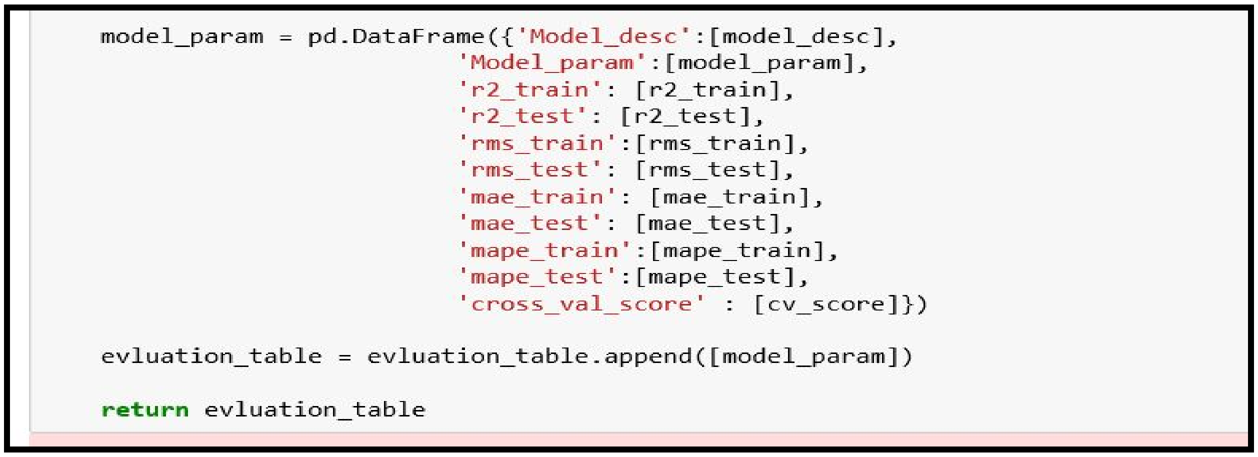


Regressor Model

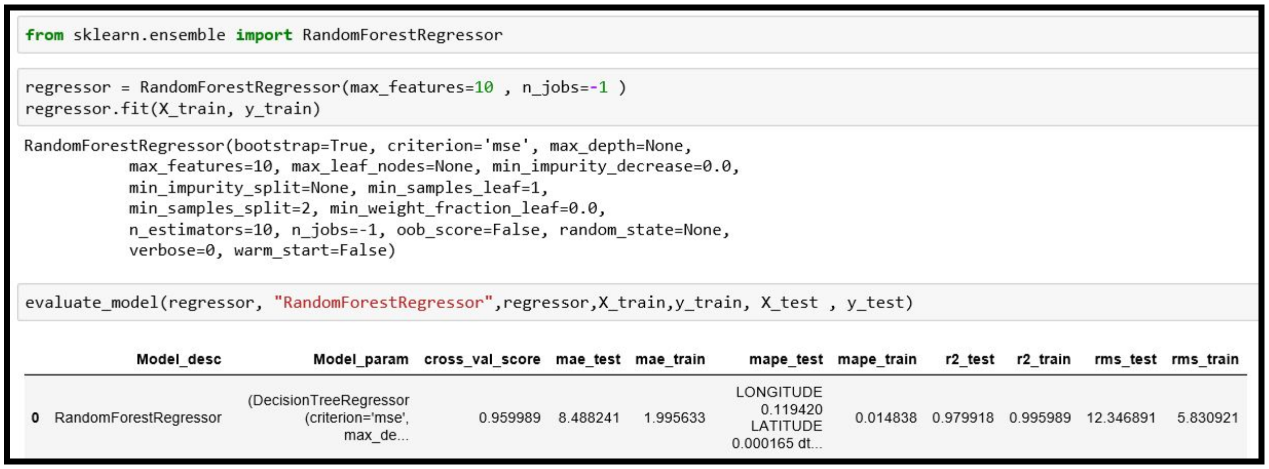
```
#Importing Libraries
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
from sklearn.cross_validation import cross_val_score


rmse_dict = {}
def rmse(correct,estimated):
    rmse_val = np.sqrt(mean_squared_error(correct,estimated))
    return rmse_val

# Generating the Table Frame for metrics
evluation_table = pd.DataFrame({  'Model_desc':[],
                        'Model_param':[],
                        'r2_train': [],
                        'r2_test': [],
                        'rms_train':[],
                        'rms_test': [],
                        'mae_train': [],
                        'mae_test': [],
                        'mape_train':[],
                        'mape_test':[],
                        'cross_val_score' : []})
```



```
# Evaluating the model
def evaluate_model(model, model_desc,model_param, X_train, y_train, X_test, y_test):
    global evluation_table

    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)


    try:
        r2_train = r2_score(y_train, y_train_pred)
        r2_test = r2_score(y_test, y_test_pred)
    except:
        r2_train = "not calculated"
        r2_test = "not calculated"
    try:
        rms_train = rmse(y_train, y_train_pred)
        rms_test = rmse(y_test, y_test_pred)
    except:
        rms_train = "not calculated"
        rms_test = "not calculated"
    try:
        mae_train = mean_absolute_error(y_train, y_train_pred)
        mae_test = mean_absolute_error(y_test, y_test_pred)
    except:
        mae_train = "not calculated"
        mae_test = "not calculated"
    try:
        mape_train = np.mean(np.abs((y_train - y_train_pred) / y_train)) * 100
        mape_test = np.mean(np.abs((y_test - y_test_pred) / y_test)) * 100
    except:
        mape_train = "not calculated"
        mape_test = "not calculated"
    try:
        cv_score = cross_val_score(model, X_train, y_train, cv=10)
        cv_score = cv_score.mean()
    except:
        cv_score = "Not calulated"

    model_param = pd.DataFrame({'Model_desc':[model_desc],
```



```
    model_param = pd.DataFrame({'Model_desc':[model_desc],
                                'Model_param':[model_param],
                                'r2_train': [r2_train],
                                'r2_test': [r2_test],
                                'rms_train':[rms_train],
                                'rms_test': [rms_test],
                                'mae_train': [mae_train],
                                'mae_test': [mae_test],
                                'mape_train':[mape_train],
                                'mape_test':[mape_test],
                                'cross_val_score' : [cv_score]})

    evluation_table = evluation_table.append([model_param])

    return evluation_table
```



```
from sklearn.ensemble import RandomForestRegressor
```

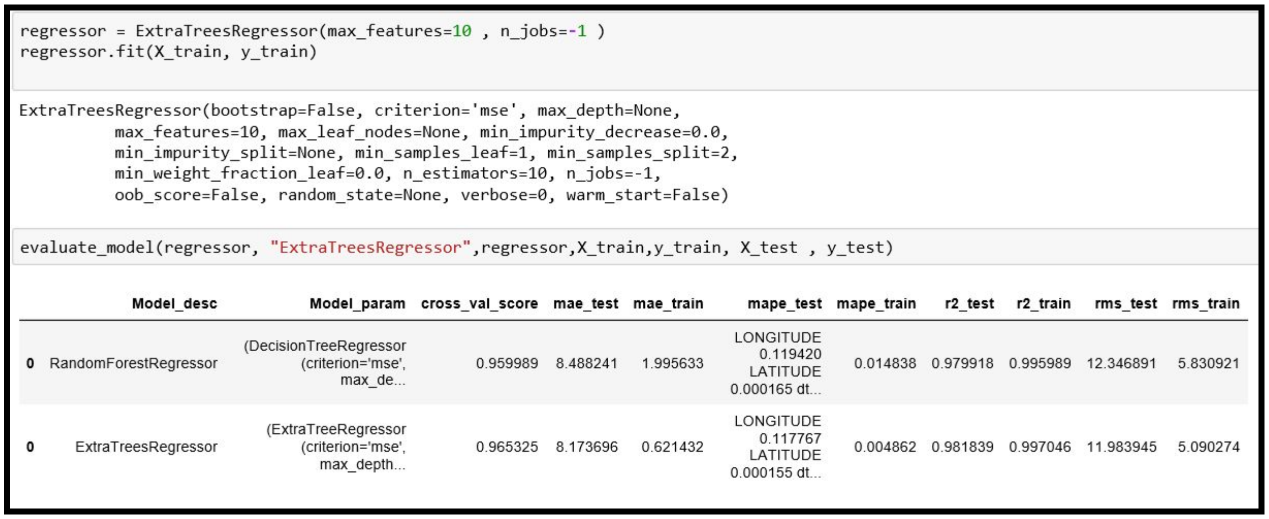
```
regressor = RandomForestRegressor(max_features=10 , n_jobs=-1 )
regressor.fit(X_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features=10, max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=10, n_jobs=-1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

```
evaluate_model(regressor, "RandomForestRegressor",regressor,X_train,y_train, X_test , y_test)
```

	Model_desc	Model_param	cross_val_score	mae_test	mae_train	mape_test	mape_train	r2_test	r2_train	rms_test	rms_train
0	RandomForestRegressor	(DecisionTreeRegressor (criterion='mse', max_de...	0.959989	8.488241	1.995633	LONGITUDE 0.119420 LATITUDE 0.000165 dt...	0.014838	0.979918	0.995989	12.346891	5.830921

From Random Forest Regressor we got 99% r-squared for training data and 97% r-squared for testing Data



```
regressor = ExtraTreesRegressor(max_features=10 , n_jobs=-1 )
regressor.fit(X_train, y_train)
```

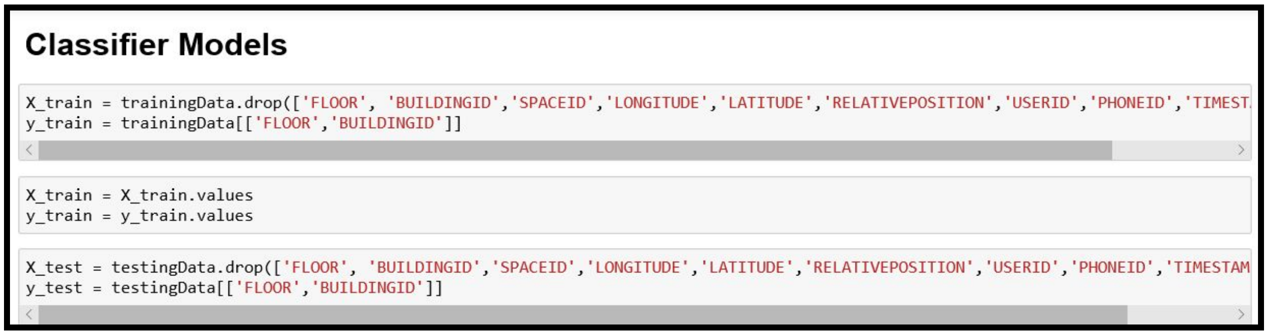
```
ExtraTreesRegressor(bootstrap=False, criterion='mse', max_depth=None,
         max_features=10, max_leaf_nodes=None, min_impurity_decrease=0.0,
         min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
         min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
         oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
evaluate_model(regressor, "ExtraTreesRegressor",regressor,X_train,y_train, X_test , y_test)
```

	Model_desc	Model_param	cross_val_score	mae_test	mae_train	mape_test	mape_train	r2_test	r2_train	rms_test	rms_train
0	RandomForestRegressor	(DecisionTreeRegressor (criterion='mse', max_de...	0.959989	8.488241	1.995633	LONGITUDE 0.119420 LATITUDE 0.000165 dt...	0.014838	0.979918	0.995989	12.346891	5.830921
0	ExtraTreesRegressor	(ExtraTreeRegressor (criterion='mse', max_depth...	0.965325	8.173696	0.621432	LONGITUDE 0.117767 LATITUDE 0.000155 dt...	0.004862	0.981839	0.997046	11.983945	5.090274

Extra Trees Regressor gives 99% r-squared for training Data and RMSE value is 5.09

It gives 98% for testing Data and RMSE value is 11.9

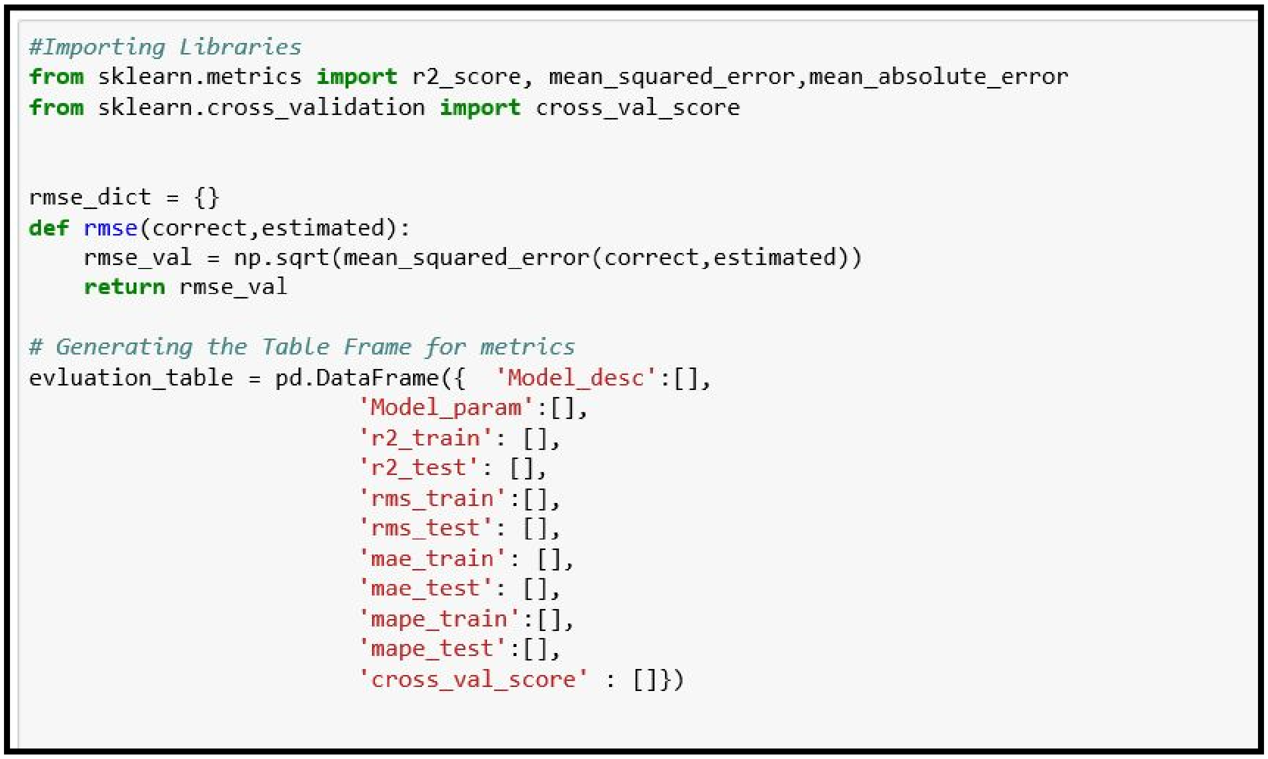


Classifier Models

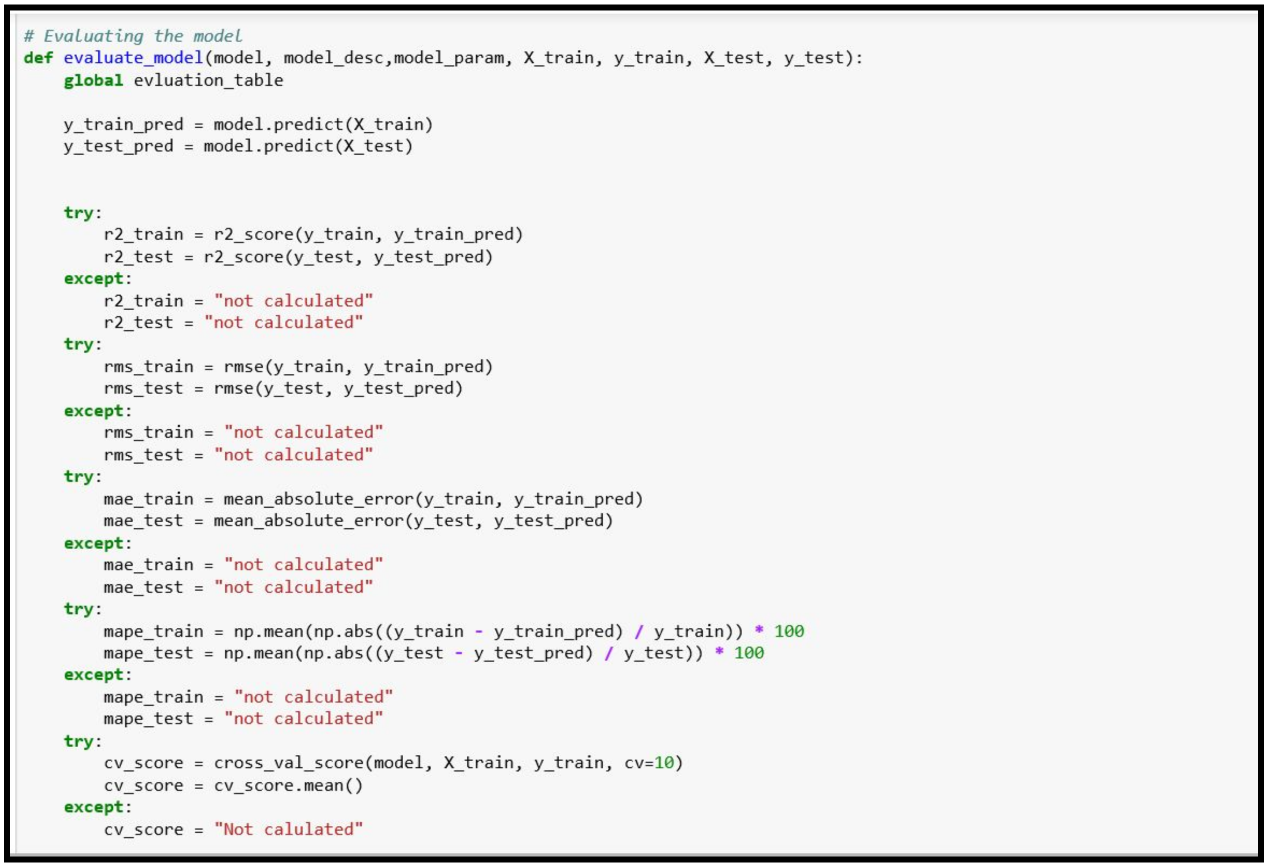
```
X_train = trainingData.drop(['FLOOR', 'BUILDINGID','SPACEID','LONGITUDE','LATITUDE','RELATIVEPOSITION','USERID','PHONEID','TIMEST
y_train = trainingData[['FLOOR','BUILDINGID']]
```

```
X_train = X_train.values
y_train = y_train.values
```

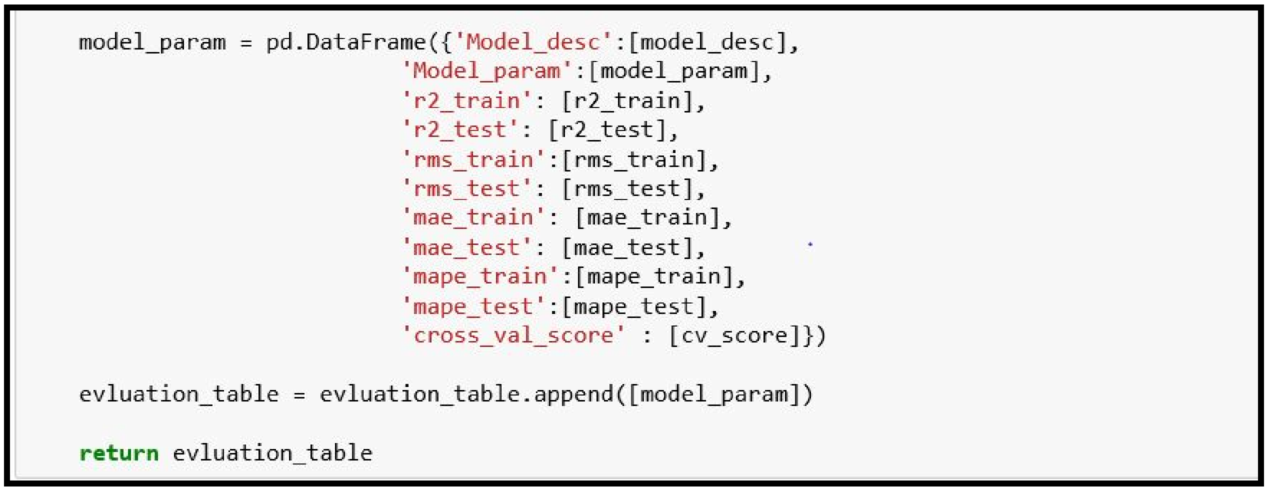
```
X_test = testingData.drop(['FLOOR', 'BUILDINGID','SPACEID','LONGITUDE','LATITUDE','RELATIVEPOSITION','USERID','PHONEID','TIMESTAM
y_test = testingData[['FLOOR','BUILDINGID']]
```



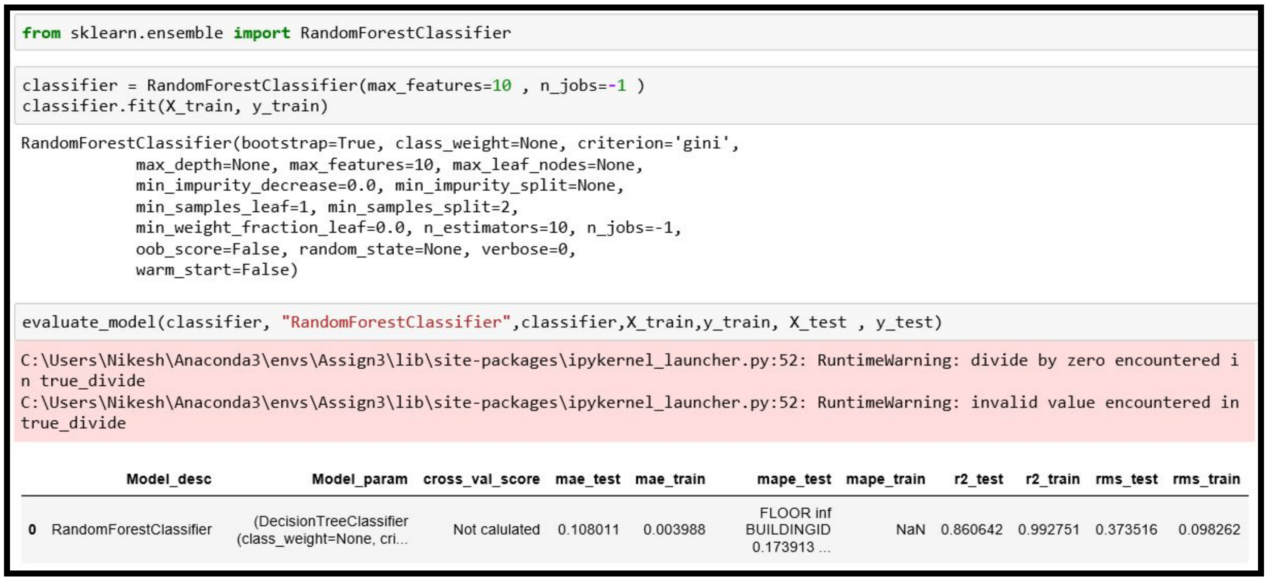
```
#Importing Libraries
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
from sklearn.cross_validation import cross_val_score


rmse_dict = {}
def rmse(correct,estimated):
    rmse_val = np.sqrt(mean_squared_error(correct,estimated))
    return rmse_val

# Generating the Table Frame for metrics
evluation_table = pd.DataFrame({  'Model_desc':[],
                        'Model_param':[],
                        'r2_train': [],
                        'r2_test': [],
                        'rms_train':[],
                        'rms_test': [],
                        'mae_train': [],
                        'mae_test': [],
                        'mape_train':[],
                        'mape_test':[],
                        'cross_val_score' : []})
```



```python
# Evaluating the model
def evaluate_model(model, model_desc,model_param, X_train, y_train, X_test, y_test):
    global evluation_table

    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)


    try:
        r2_train = r2_score(y_train, y_train_pred)
        r2_test = r2_score(y_test, y_test_pred)
    except:
        r2_train = "not calculated"
        r2_test = "not calculated"
    try:
        rms_train = rmse(y_train, y_train_pred)
        rms_test = rmse(y_test, y_test_pred)
    except:
        rms_train = "not calculated"
        rms_test = "not calculated"
    try:
        mae_train = mean_absolute_error(y_train, y_train_pred)
        mae_test = mean_absolute_error(y_test, y_test_pred)
    except:
        mae_train = "not calculated"
        mae_test = "not calculated"
    try:
        mape_train = np.mean(np.abs((y_train - y_train_pred) / y_train)) * 100
        mape_test = np.mean(np.abs((y_test - y_test_pred) / y_test)) * 100
    except:
        mape_train = "not calculated"
        mape_test = "not calculated"
    try:
        cv_score = cross_val_score(model, X_train, y_train, cv=10)
        cv_score = cv_score.mean()
    except:
        cv_score = "Not calulated"
```



```python
    model_param = pd.DataFrame({'Model_desc':[model_desc],
                                'Model_param':[model_param],
                                'r2_train': [r2_train],
                                'r2_test': [r2_test],
                                'rms_train':[rms_train],
                                'rms_test': [rms_test],
                                'mae_train': [mae_train],
                                'mae_test': [mae_test],
                                'mape_train':[mape_train],
                                'mape_test':[mape_test],
                                'cross_val_score' : [cv_score]})

    evluation_table = evluation_table.append([model_param])

    return evluation_table
```



```
from sklearn.ensemble import RandomForestClassifier
```

```
classifier = RandomForestClassifier(max_features=10 , n_jobs=-1 )
classifier.fit(X_train, y_train)
```

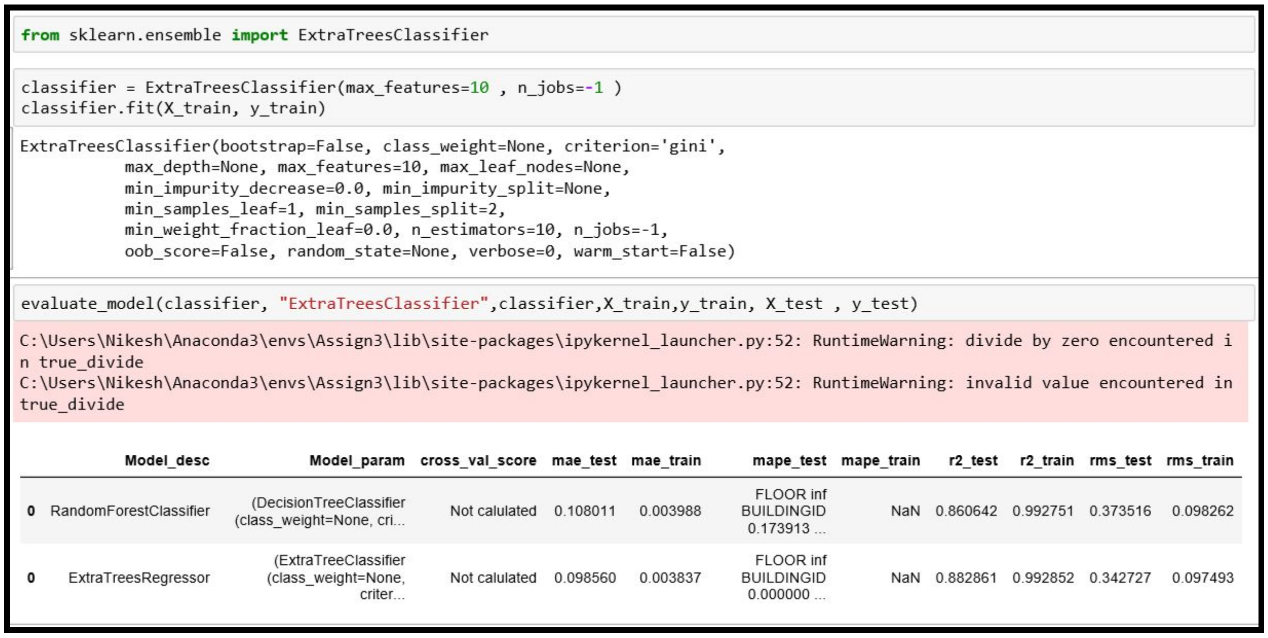
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features=10, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
evaluate_model(classifier, "RandomForestClassifier",classifier,X_train,y_train, X_test , y_test)
```

```
C:\Users\Nikesh\Anaconda3\envs\Assign3\lib\site-packages\ipykernel_launcher.py:52: RuntimeWarning: divide by zero encountered i
n true_divide
C:\Users\Nikesh\Anaconda3\envs\Assign3\lib\site-packages\ipykernel_launcher.py:52: RuntimeWarning: invalid value encountered in
true_divide
```

	Model_desc	Model_param	cross_val_score	mae_test	mae_train	mape_test	mape_train	r2_test	r2_train	rms_test	rms_train
0	RandomForestClassifier	(DecisionTreeClassifier (class_weight=None, cri...	Not calulated	0.108011	0.003988	FLOOR inf BUILDINGID 0.173913 ...	NaN	0.860642	0.992751	0.373516	0.098262

Random Forest Classifier gives 99% r-squared for training data and 86% r-squared for testing Data



```
from sklearn.ensemble import ExtraTreesClassifier
```

```
classifier = ExtraTreesClassifier(max_features=10 , n_jobs=-1 )
classifier.fit(X_train, y_train)
```

```
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=None, max_features=10, max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
evaluate_model(classifier, "ExtraTreesClassifier",classifier,X_train,y_train, X_test , y_test)
```

```
C:\Users\Nikesh\Anaconda3\envs\Assign3\lib\site-packages\ipykernel_launcher.py:52: RuntimeWarning: divide by zero encountered i
n true_divide
C:\Users\Nikesh\Anaconda3\envs\Assign3\lib\site-packages\ipykernel_launcher.py:52: RuntimeWarning: invalid value encountered in
true_divide
```

	Model_desc	Model_param	cross_val_score	mae_test	mae_train	mape_test	mape_train	r2_test	r2_train	rms_test	rms_train
0	RandomForestClassifier	(DecisionTreeClassifier (class_weight=None, cri...	Not calulated	0.108011	0.003988	FLOOR inf BUILDINGID 0.173913 ...	NaN	0.860642	0.992751	0.373516	0.098262
0	ExtraTreesRegressor	(ExtraTreeClassifier (class_weight=None, criter...	Not calulated	0.098560	0.003837	FLOOR inf BUILDINGID 0.000000 ...	NaN	0.882861	0.992852	0.342727	0.097493

Extra Trees Classifier gives 99% r-squared for training data and 88% r-squared for testing Data

Pipeline

Luigi

The purpose of Luigi is to address all the plumbing typically associated with long-running batch processes.

Conceptually, Luigi is similar to GNU Make where you have certain tasks and these tasks in turn may have dependencies on other tasks

The Luigi server comes with a web interface too, so you can search and filter among all your tasks.

Steps to Perform for Luigi

Requirement : Python 2.7.*

1. Run pip install luigi to install Luigi
2. Run the build.sh, it will create a virtual environment in the root directory of the project with all the dependencies.
3. It will also setup the luigi Central Scheduler and run it as a daemon process.
4. The Luigi Task Visualiser can be accessed by http://localhost:8082 which will give visualisation of all the running tasks.

Python Object Serialization

Pickle

It is used for serializing and de-serializing a Python object structure.

Any object in python can be pickled so that it can be saved on disk.

What pickle does is that it "serialize" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream.

The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Dockerized

Docker

Docker is an open source tool that automates the deployment of the application inside software container.

When you develop an application, you need to provide your code alongside with all possible dependencies like libraries, web server, databases, etc. You may end up in a situation when the application is working on your computer but won't even start on stage server, dev or a QA's machine.

This challenge can be addressed by isolating the app to make it independent of the system.

Steps to Perform for Docker

1. We have used an Ubuntu 16.4 image and updated its libraries.
2. Then we installed Python3 and Python pip to execute the python script for web scraping.
3. Required libraries has been stated in the dockerfiles itself which include : ● Boto
4. docker build -f Dockerfile . -t sreerag/as3
5. docker run -it -p 5000:5000 sreerag/as3

Model Deployment

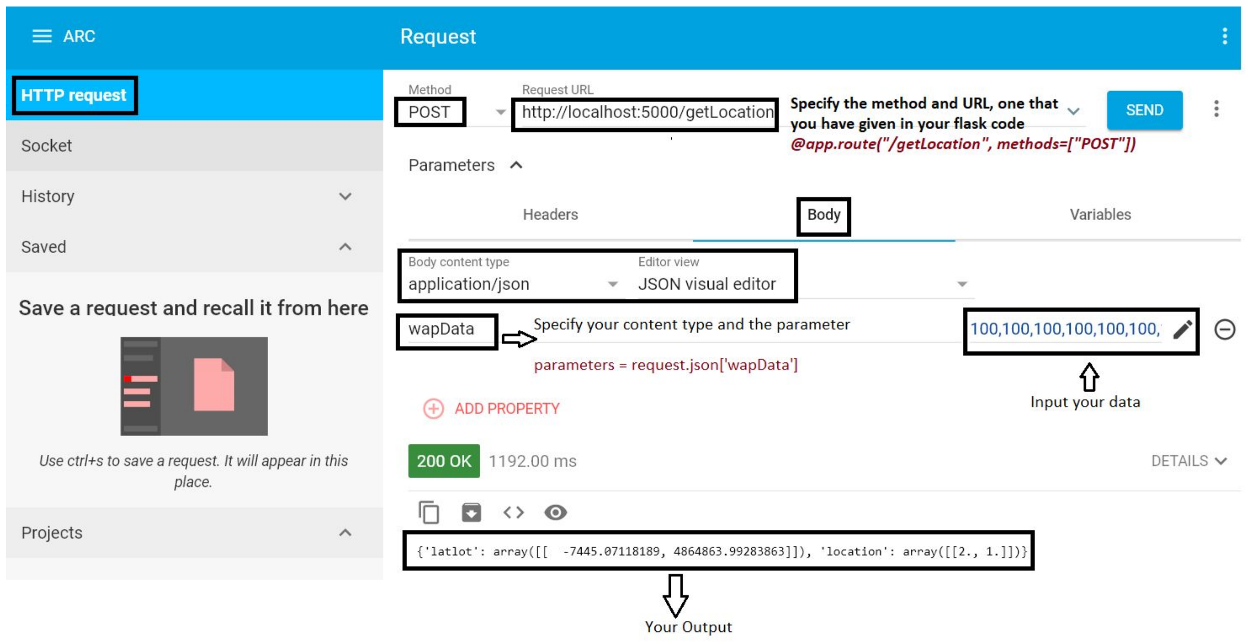
Web Application using Flask

Requirement steps :

1. pip install flask
2. pip install flask_mysql
3. pip install flask_restful
4. Python app.py
5. Start the Apache server and connect it with mySql using Xamp

Steps to perform for REST Api calls using JSON

Requirement : Install Advanced REST Client



Save a request and recall it from here

Use ctrl+s to save a request. It will appear in this place.

Method: POST

Request URL: http://localhost:5000/getLocation

Specify the method and URL, one that you have given in your flask code *@app.route("/getLocation", methods=["POST"])*

Body content type: application/json

Editor view: JSON visual editor

wapData ⇨ Specify your content type and the parameter

parameters = request.json['wapData']

100,100,100,100,100,100,

Input your data

200 OK 1192.00 ms

```
{'latlot': array([[  -7445.07118189, 4864863.99283863]]), 'location': array([[2., 1.]])}
```

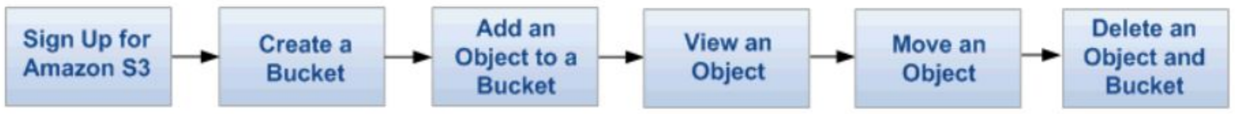
Your Output

Amazon S3

Amazon Simple Storage Service (Amazon S3) is storage for the Internet.

You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

Steps to Perform for Amazon S3



References :

- Luigi : https://github.com/Atreya22/luigi_rosmann_sales
- Pickle : https://pythontips.com/2013/08/02/what-is-pickle-in-python/
- Amazon S3 :https://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html