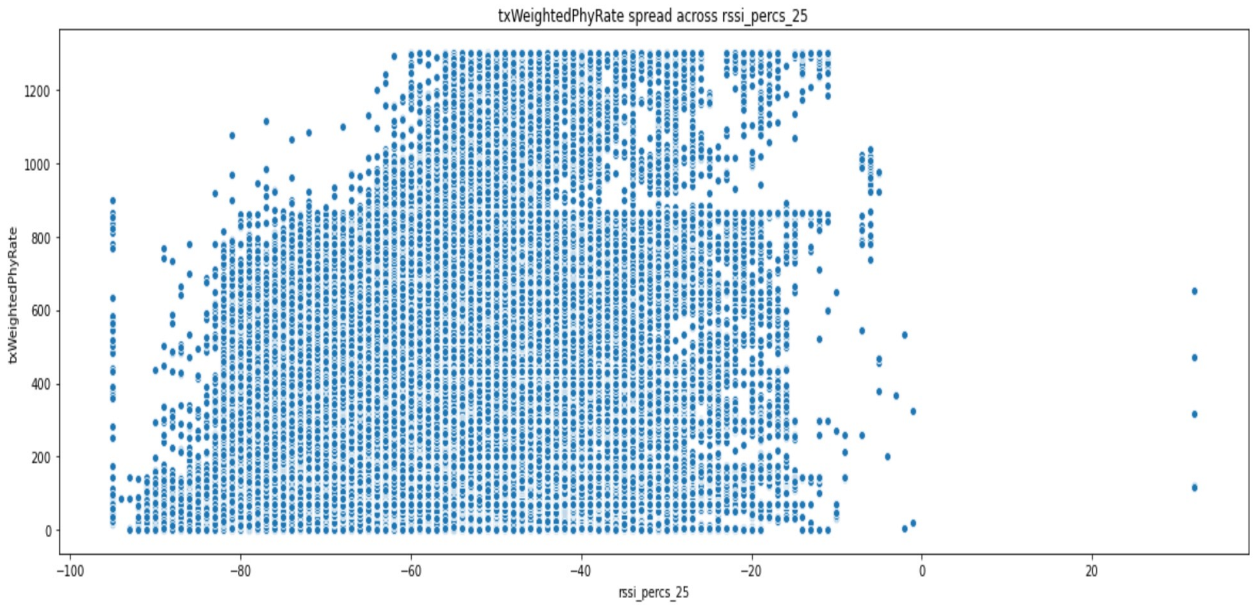
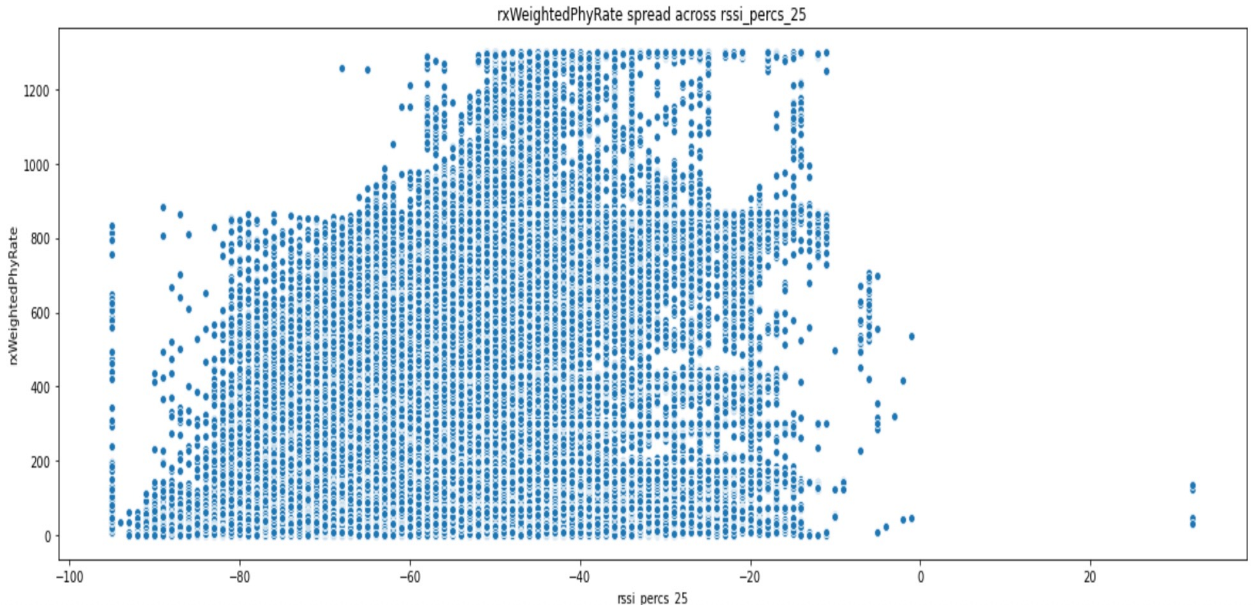
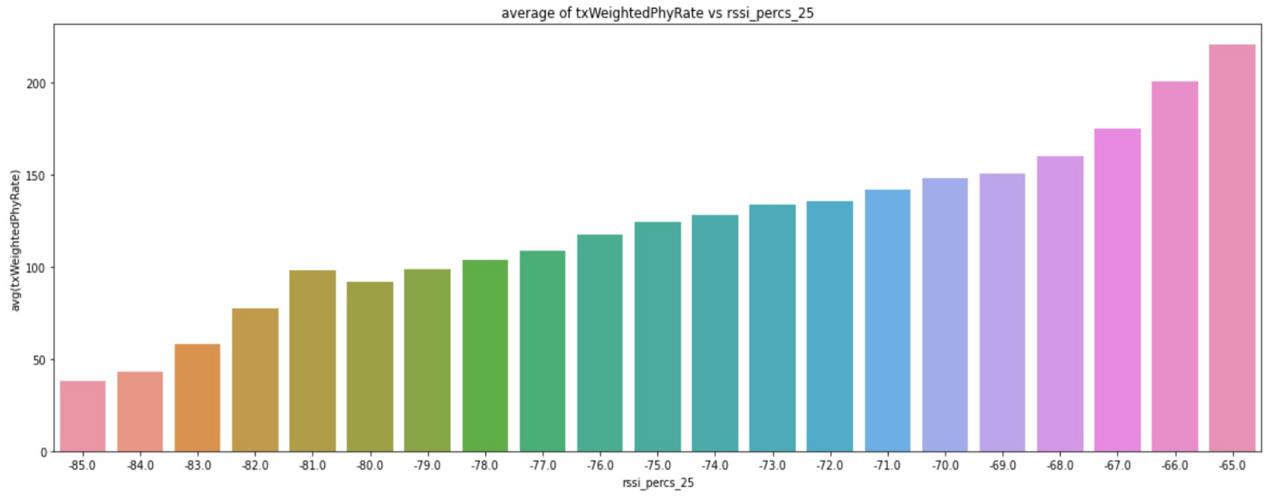
txWeightedPhyRate and rxWeightedPhyRate has similar spread across rssi_percs_25

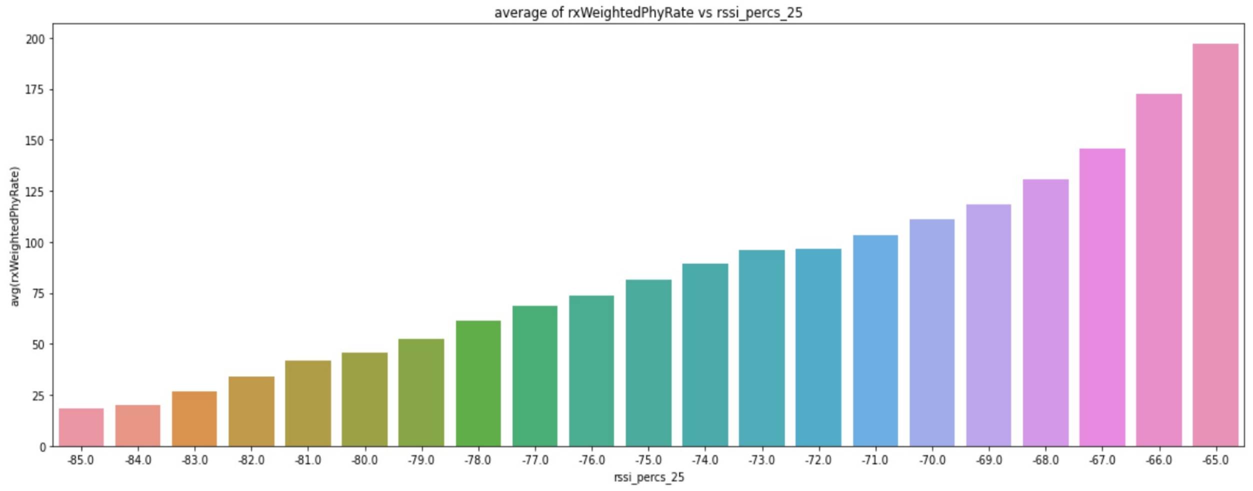




- Majority data points for both tx and rx weighted phy rate lies between -80 dB to -20 dB rssi_percs_25

With increase in rssi_percs_25 both tx and rx average weighted phy rate increases

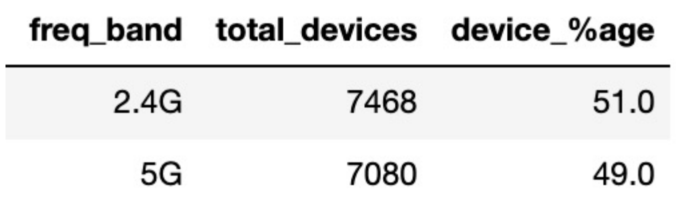




- Average txWeightedPhyRate becomes ~4X when rssi_percs_25 increases from -85dB to -65dB
- Average rxWeightedPhyRate becomes ~8X when rssi_percs_25 increases from -85dB to -65dB

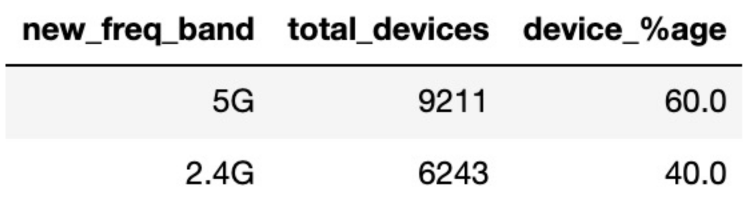
Post activation of new band steering, 5G band will see a device growth of almost 30%

Pre activation device distribution



freq_band	total_devices	device_%age
2.4G	7468	51.0
5G	7080	49.0

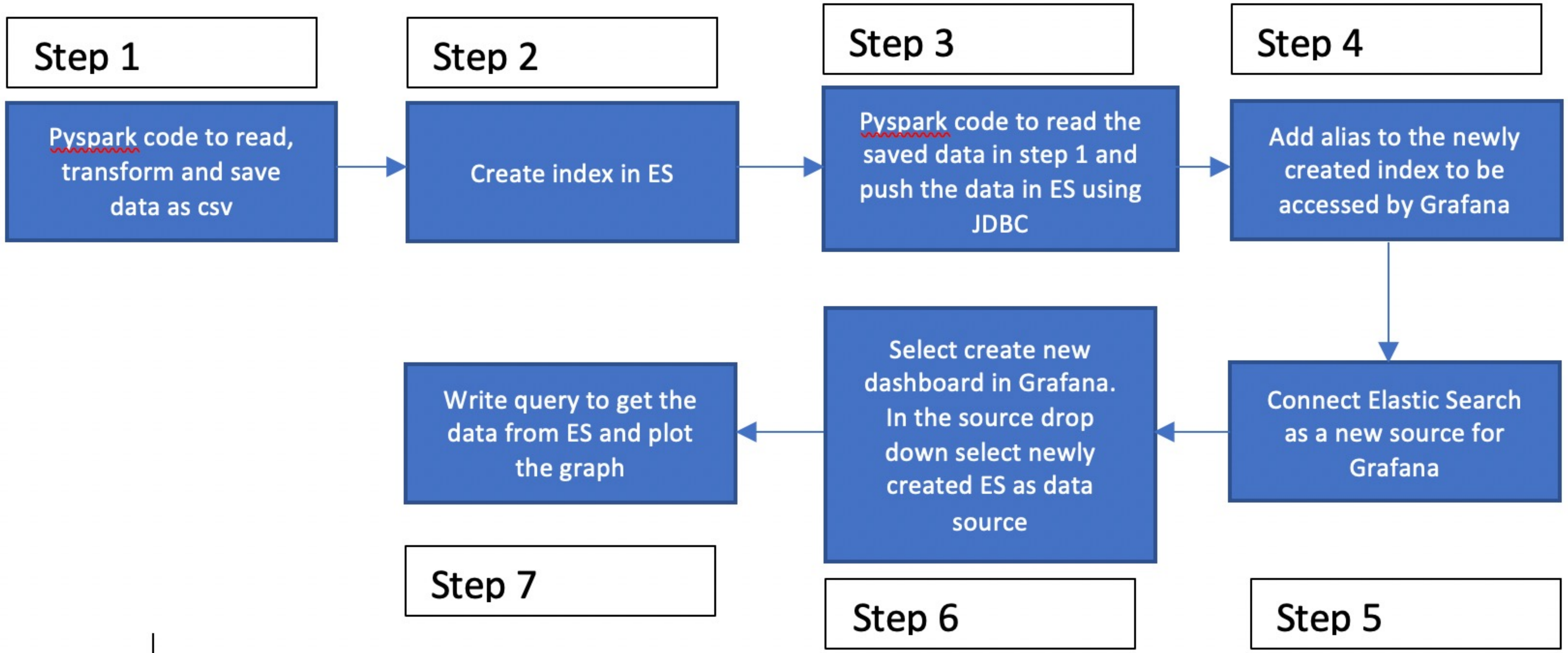
Post activation device distribution



new_freq_band	total_devices	device_%age
5G	9211	60.0
2.4G	6243	40.0

- Currently 51% of devices are connected to 2.4G and 49% of devices are connected to 5G
- Post activation of new band steering, 5G band will have 60% of total connected device where as 2.4G will go down to 40%

Data pipeline design and spec


Step 1
Pyspark code to read, transform and save data as csv
Step 2
Create index in ES
Step 3
Pyspark code to read the saved data in step 1 and push the data in ES using JDBC
Step 4
Add alias to the newly created index to be accessed by Grafana
Step 5
Connect Elastic Search as a new source for Grafana
Step 6
Select create new dashboard in Grafana. In the source drop down select newly created ES as data source
Step 7
Write query to get the data from ES and plot the graph

Pseudo code for data pipeline

Step 1 (run pyspark code):

```
spark_submit(‘pyspark_code.py’,application_resource,machine_type,args) #This will save the transformed data into HDFS
```

Step 2 (create new index on ES):

```
def create_index(elastic_settings,index_name,shards=1,replica=0,mappings=None,codec='best_compression',refresh_interval=-1):
    url = 'http://'+elastic_settings['url']+':'+elastic_settings['port']+'/'+str(index_name)
    print(url)
    es_user = elastic_settings['user']
    es_pwd = elastic_settings['pass']
    payload = {"settings" :
                {   "number_of_shards" : shards,
                    "codec": "best_compression",
                    "refresh_interval": "-1",
                    "number_of_replicas":replica  }
                ,"mappings": {
                    "doc": {
                        "properties":{}
                    }}}

    if mappings != None:
        payload['mappings']['doc']['properties']=mappings

    headers = {'Content-Type': "application/json"}
    response = requests.put(url, data=json.dumps(payload), headers=headers,auth = (es_user,es_pwd))
    return(response)

Create_index(elastic_setting,index_name,)
```

Pseudo code for data pipeline contd…

Step 3 (run pyspark code to read saved data in step 1 and push the read data to newly created index in step 2):

```
df = spark.read.option(“header”,”true”).csv(‘dataset_path.csv’)

df.write.option("spark.es.batch.write.refresh",es_index_settings[AirflowJobConstants.ES_SETTINGS_BATCH_WRITE_REFRESH_STRING]).option("spark.es.batch.size.bytes",es_index_settings[AirflowJob
Constants.ES_SETTINGS_BATCH_SIZE_STRING]).option("index.refresh_interval",str(es_index_settings[AirflowJobConstants.ES_SETTINGS_REFRESH_INTERVAL_STRING])).option("es.batch.size.entries",
es_index_settings[AirflowJobConstants.ES_SETTINGS_BATCH_SIZE_ENTRIES_STRING]).option("es.index.auto.create",str(es_index_settings[AirflowJobConstants.ES_SETTINGS_INDEX_AUTO_CREATE_S
TRING])).option("es.write.operation", "index").option("es.nodes", es_settings['url']).option("es.port",es_settings['port']).option("es.spark.dataframe.write.null",
str(es_index_settings[AirflowJobConstants.ES_SETTINGS_DATAFRAME_WRITE_NULL_STRING])).format("org.elasticsearch.spark.sql").mode("append").save(new_index_name + "/doc")
```

Step 4 (add alias):

```
def add_alias(elastic_settings,alias,index_name=[]):
    url = 'http://'+elastic_settings['url']+':'+elastic_settings['port']+'/_aliases'
    es_user = elastic_settings['user']
    es_pwd = elastic_settings['pass']

    actions = []
    for items in index_name:
        ac = { "add" : { "index" : str(items), "alias" : str(alias) } }
        actions.append(ac)

    payload = {
                "actions" : actions
              }
    headers = {'Content-Type': "application/json"}
    response = requests.post(url, data=json.dumps(payload), headers=headers,auth = (es_user,es_pwd))
    return(response)

add_alias(elastic_settings,alias_name,index_name)
```

Pseudo code for data pipeline contd…

Step 5 (connect a new data source in Grafana):

Login to Grafana web GUI and select “Data Sources” from the menu. Click on “Add data source”,
Type “elastic_search_data_source_name” for the name and select “Elasticsearch” from the type dropdown
Provide elastic search end point, username and password

Step 6 (create new dashboard in Grafana):

From the menu icon, select Dashboards > New, and select a “Graph” for the type
Under the “Metrics” tab, remove the test metric using the fake data source, and select “elastic_search_data_source_name” for the panel data source and press “Add Query”
Write the query for getting the data