AIM - To create a ML model to identify FM stations

Python version used -> python 3

Step - 1

Run RTL_power command

Install GQRX software then connect RTL_SDR dongle and open terminal.

Note : Following command only works for Linux and Mac OS.

COMMAND -> rtl_power -f min:max:bin -g gain -i interval -e runtime filename.ext where min is initial frequency max is terminal frequency bin is frequency interval interval in seconds

COMMAND I USED -

rtl_power -f 87M:108M:1k -g 20 -i 10 -e 5m logfile.csv

All the data is stored in a csv file logfile.csv.

Step - 2

Data cleaning

We will now convert obtained csv into a desireable pandas dataframe

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("logfile.csv", header=None)
```

In [2]:

```
df.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	4093
0	2018-03-30	22:31:10	87000000	89625000	640.87	40	-45.44	-50.61	-52.59	-52.59	...	-53.91
1	2018-03-30	22:31:10	89625000	92250000	640.87	40	-55.90	-57.27	-57.36	-56.05	...	-56.50
2	2018-03-30	22:31:10	92250000	94875000	640.87	40	-40.56	-41.09	-40.24	-41.16	...	-41.91
3	2018-03-30	22:31:10	94875000	97500000	640.87	40	-41.38	-40.05	-39.69	-40.90	...	-44.02
4	2018-03-30	22:31:10	97500000	100125000	640.87	40	-52.95	-55.07	-54.32	-56.54	...	-52.79

5 rows × 4103 columns

In [3]:

```
x=[] # Stores all the frequencies
y=[] # Stores corresponding power value
z=[] # Stores is_FM, if 1 then yes, if 0 then no

# following array is obtained from "https://en.wikipedia.org/wiki/Category:Lists_of_radio_s
arr = [76.1, 76.5, 77.1, 78.8, 80.0, 80.2, 81.3, 82.5, 84.7, 87.5, 87.6, 87.7, 87.8, 87.9,

for j in range(0,len(df)):
    for i in range(6,4103):
        y.append(df[i][j])
        r = (df[3][j]-df[2][j])/4096
        temp = df[3][j]+(r*(i-6))
        x.append(temp)
        check = round((temp/1000000),1)
        n=0
        for k in arr:
            if(check == k):
                n=1
        z.append(n)
```

In [4]:

```
dfs = pd.DataFrame({"Frequency":x,"Power":y,"Is_FM":z})
dfs.head()
```

Out[4]:

	Frequency	Is_FM	Power
0	8.962500e+07	1	-45.44
1	8.962564e+07	1	-50.61
2	8.962628e+07	1	-52.59
3	8.962692e+07	1	-52.59
4	8.962756e+07	1	-53.53

In [5]:

```
dfs.isnull().sum()
```

Out[5]:

```
Frequency    0
Is_FM        0
Power        0
dtype: int64
```

In [6]:

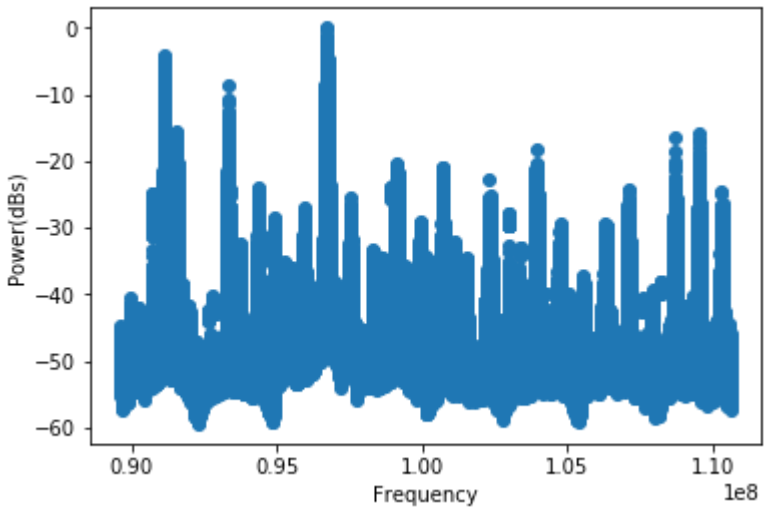
```
dfs.describe()
```

Out[6]:

	Frequency	Is_FM	Power
count	9.832800e+05	983280.000000	983280.000000
mean	1.001250e+08	0.577435	-47.791829
std	6.062204e+06	0.493968	7.172532
min	8.962500e+07	0.000000	-59.420000
25%	9.487500e+07	0.000000	-51.920000
50%	1.001250e+08	1.000000	-50.200000
75%	1.053750e+08	1.000000	-46.330000
max	1.106250e+08	1.000000	0.010000

In [7]:

```
plt.scatter(x,y)
plt.ylabel('Power(dBs)')
plt.xlabel('Frequency')
plt.show()
```



In [8]:

```
dfs.plot.scatter('Frequency', 'Power', c='Is_FM', colormap='jet')
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1848a8dec88>
```

In [9]:

```
dfs.groupby('Is_FM').count()
```

Out[9]:

Is_FM	Frequency	Power
0	415500	415500
1	567780	567780

Step 3 -

Train all the classification models to find the best one

In [10]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import Perceptron
import xgboost as xgb

from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
D:\python\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarni
ng: This module was deprecated in version 0.18 in favor of the model_selecti
on module into which all the refactored classes and functions are moved. Als
o note that the interface of the new CV iterators are different from that of
this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [11]:

```
X = dfs.drop('Is_FM', axis=1)
y = dfs['Is_FM']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.12, random_state=0)
```

In [12]:

```
regression_model = LogisticRegression()
regression_model.fit(X_train, y_train)

y_predict = regression_model.predict(X_train)
print("Training accuracy :",accuracy_score(y_train,y_predict))


y_predict = regression_model.predict(X_test)
print("accuracy score of y_test :",accuracy_score(y_test,y_predict))
```

```
Training accuracy : 0.577290052075
accuracy score of y_test : 0.578495516721
```

In [13]:

```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_predict = knn.predict(X_train)
print("Using KNN -\naccuracy score of y_train :",accuracy_score(y_train,y_predict))
y_predict = knn.predict(X_test)
print("accuracy score of y_test :",accuracy_score(y_test,y_predict))
```

```
Using KNN -
accuracy score of y_train : 1.0
accuracy score of y_test : 1.0
```

In [14]:

```
perceptron = Perceptron()
perceptron.fit(X_train, y_train)
y_predict = perceptron.predict(X_train)
print("Using Perceptron -\naccuracy score of y_train :",accuracy_score(y_train,y_predict))
y_predict = perceptron.predict(X_test)
print("accuracy score of y_test :",accuracy_score(y_test,y_predict))
```

```
D:\python\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:128:
FutureWarning: max_iter and tol parameters have been added in <class 'sklear
n.linear_model.perceptron.Perceptron'> in 0.19. If both are left unset, they
default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to
max_iter=1000. From 0.21, default max_iter will be 1000, and default tol wil
l be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
Using Perceptron -
accuracy score of y_train : 0.422709947925
accuracy score of y_test : 0.421504483279
```

In [15]:

```
gradboost = xgb.XGBClassifier(n_estimators=1000)
gradboost.fit(X_train, y_train)
y_predict = gradboost.predict(X_train)
print("Using XGBoost -\naccuracy score of y_train :",accuracy_score(y_train,y_predict))
y_predict = gradboost.predict(X_test)
print("accuracy score of y_test :",accuracy_score(y_test,y_predict))
```

```
Using XGBoost -
accuracy score of y_train : 1.0
accuracy score of y_test : 1.0
```

Step - 4

Testing with test data

In [16]:

```
df = pd.read_csv("lgfile.csv", header=None)
```

In [17]:

```
df.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	...	4093
0	2018-04-15	16:06:13	87000000	89625000	640.87	38	-45.84	-55.77	-56.08	-57.61	...	-56.32
1	2018-04-15	16:06:13	89625000	92250000	640.87	38	-55.28	-57.92	-56.26	-55.92	...	-56.83
2	2018-04-15	16:06:13	92250000	94875000	640.87	38	-37.90	-37.65	-38.25	-39.42	...	-37.16
3	2018-04-15	16:06:13	94875000	97500000	640.87	38	-35.69	-39.10	-37.89	-38.95	...	-34.19
4	2018-04-15	16:06:13	97500000	100125000	640.87	38	-53.26	-52.24	-52.38	-53.54	...	-48.42

5 rows × 4103 columns

In [18]:

```
x=[] # Stores all the frequencies
y=[] # Stores corresponding power value
z=[] # Stores is_FM, if 1 then yes, if 0 then no

# following array is obtained from "https://en.wikipedia.org/wiki/Category:Lists_of_radio_s
arr = [76.1, 76.5, 77.1, 78.8, 80.0, 80.2, 81.3, 82.5, 84.7, 87.5, 87.6, 87.7, 87.8, 87.9,

for j in range(0,len(df)):
    for i in range(6,4103):
        y.append(df[i][j])
        r = (df[3][j]-df[2][j])/4096
        temp = df[3][j]+(r*(i-6))
        x.append(temp)
        check = round((temp/1000000),1)
        n=0
        for k in arr:
            if(check == k):
                n=1
        z.append(n)
```

In [19]:

```
dfs = pd.DataFrame({"Frequency":x,"Power":y,"Is_FM":z})
dfs.head()
```

Out[19]:

	Frequency	Is_FM	Power
0	8.962500e+07	1	-45.84
1	8.962564e+07	1	-55.77
2	8.962628e+07	1	-56.08
3	8.962692e+07	1	-57.61
4	8.962756e+07	1	-54.90

In [20]:

```
dfs.isnull().sum()
```

Out[20]:

```
Frequency    0
Is_FM        0
Power        0
dtype: int64
```

In [21]:

```
dfs.describe()
```

Out[21]:

	Frequency	Is_FM	Power
count	9.832800e+05	983280.000000	983280.000000
mean	1.001250e+08	0.577435	-45.524167
std	6.062204e+06	0.493968	9.412274
min	8.962500e+07	0.000000	-60.340000
25%	9.487500e+07	0.000000	-51.890000
50%	1.001250e+08	1.000000	-49.530000
75%	1.053750e+08	1.000000	-41.270000
max	1.106250e+08	1.000000	9.470000

In [22]:

```
X = dfs.drop('Is_FM', axis=1)
y = dfs['Is_FM']
```

In [23]:

```
y_predict = regression_model.predict(X)
print("accuracy score :",accuracy_score(y,y_predict))
```

```
accuracy score : 0.577434708323
```

In [24]:

```
y_predict = knn.predict(X)
print("accuracy score :",accuracy_score(y,y_predict))
```

```
accuracy score : 1.0
```

In [25]:

```
y_predict = perceptron.predict(X)
print("accuracy score :",accuracy_score(y,y_predict))
```

```
accuracy score : 0.422565291677
```

In [28]:

```
y_predict = gradboost.predict(X)
print("accuracy score :",accuracy_score(y,y_predict))
```

```
accuracy score : 0.999908469612
```

In [ ]:

```

```