In [8]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import statsmodels.api as sm
from scipy.stats import norm
from sklearn.svm import SVC
from scipy import stats
import seaborn as sns
import numpy as np
```

In [24]:

```
train_df = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/train.cs
seg_id = train_df['segment_id'][0]
df_seg_id = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/train/'
```

Analyzing seismic data for detection systems.

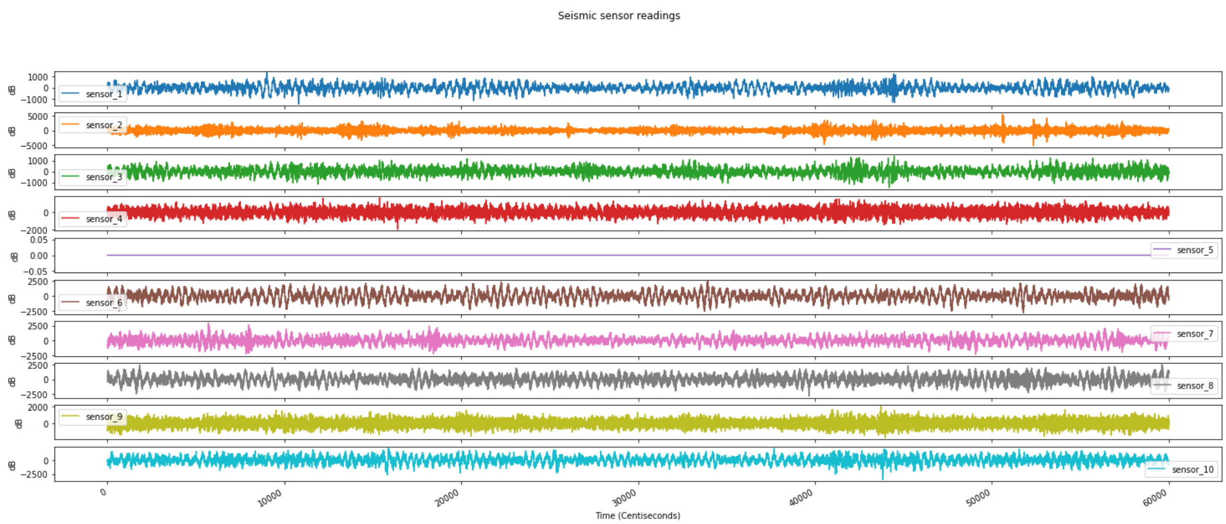
Introduction

In this project we will investigate whether we can interpret seismic activity of a volcano to predict eruptions within a timeframe (0-2 day, 2-4 days, 4-5 days). Seismic activity in this project refers to the tremors and earthquakes that occur within a close range to a volcano (within 10km, of the surface). Usually an eruption is preceded by dozens to hundreds of seismic events[1].

In our study, we have a curated dataset of seismic activity readings from 10 sensors at unspecified locations around a volcano. Each set of reading contains 10 mins of data form each sensor.

The ability to interpret seismic data as to predict a time to eruption would of course be beneficial toward disaster management.

Figure A: Seismic sensor data

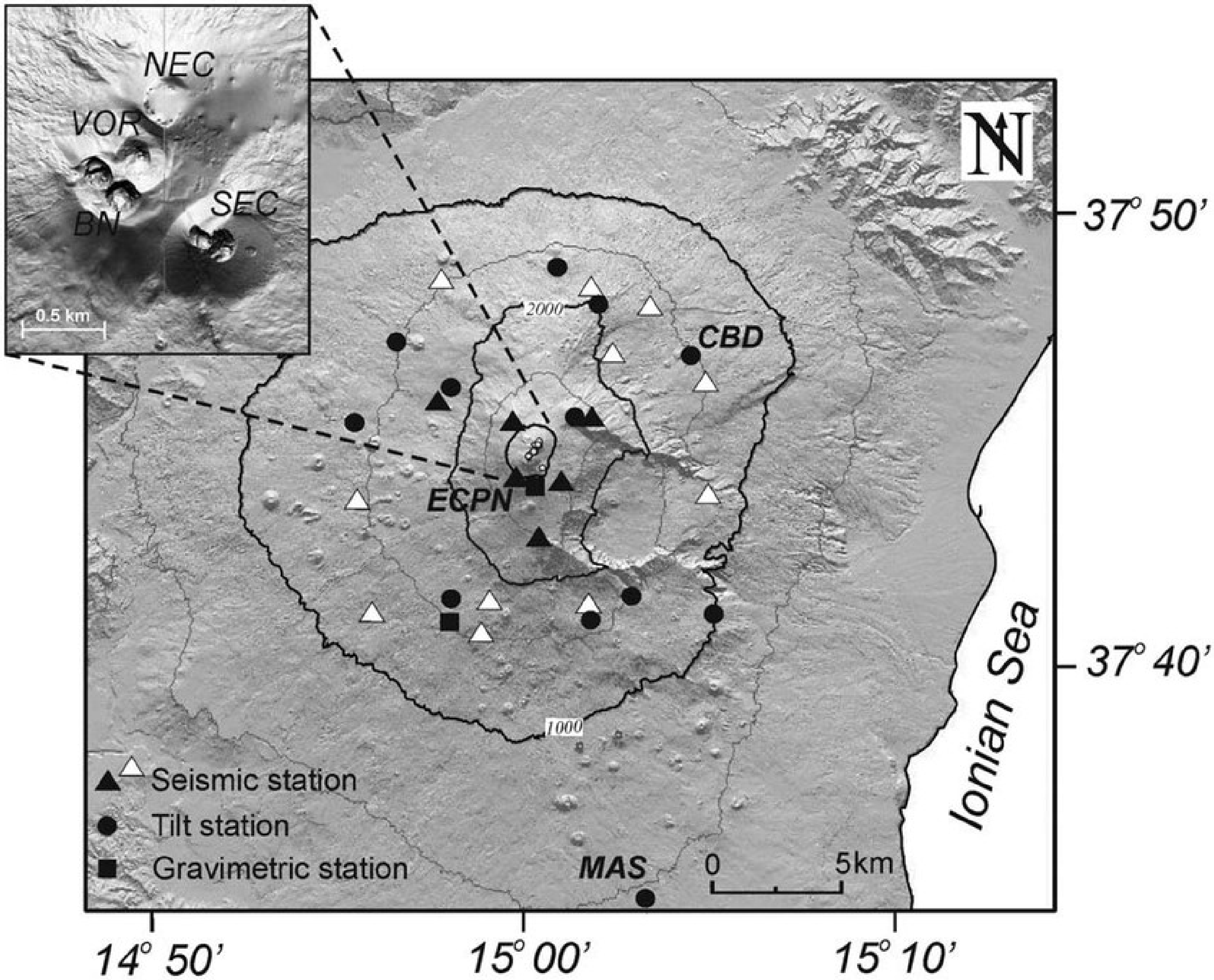


Data specification

We use a publicly available dataset distributed by Kaggle, which is provided by the The National Institute of Geophysics and Volcanology, a research institute in Italy. Downloaded the data from: [https://www.kaggle.com/c/predict-volcanic-eruptions-ingv-oe/data (https://www.kaggle.com/c/predict-volcanic-eruptions-ingv-oe/data)](https://www.kaggle.com/c/predict-volcanic-eruptions-ingv-oe/data)

We are restricted as to precise data collection methods and metadata from the sensors. We have only been provided raw seismic data from 10 unspecified location around a volcano. However we were able to establish that the dataset some from monitored events around Mt. Etna.

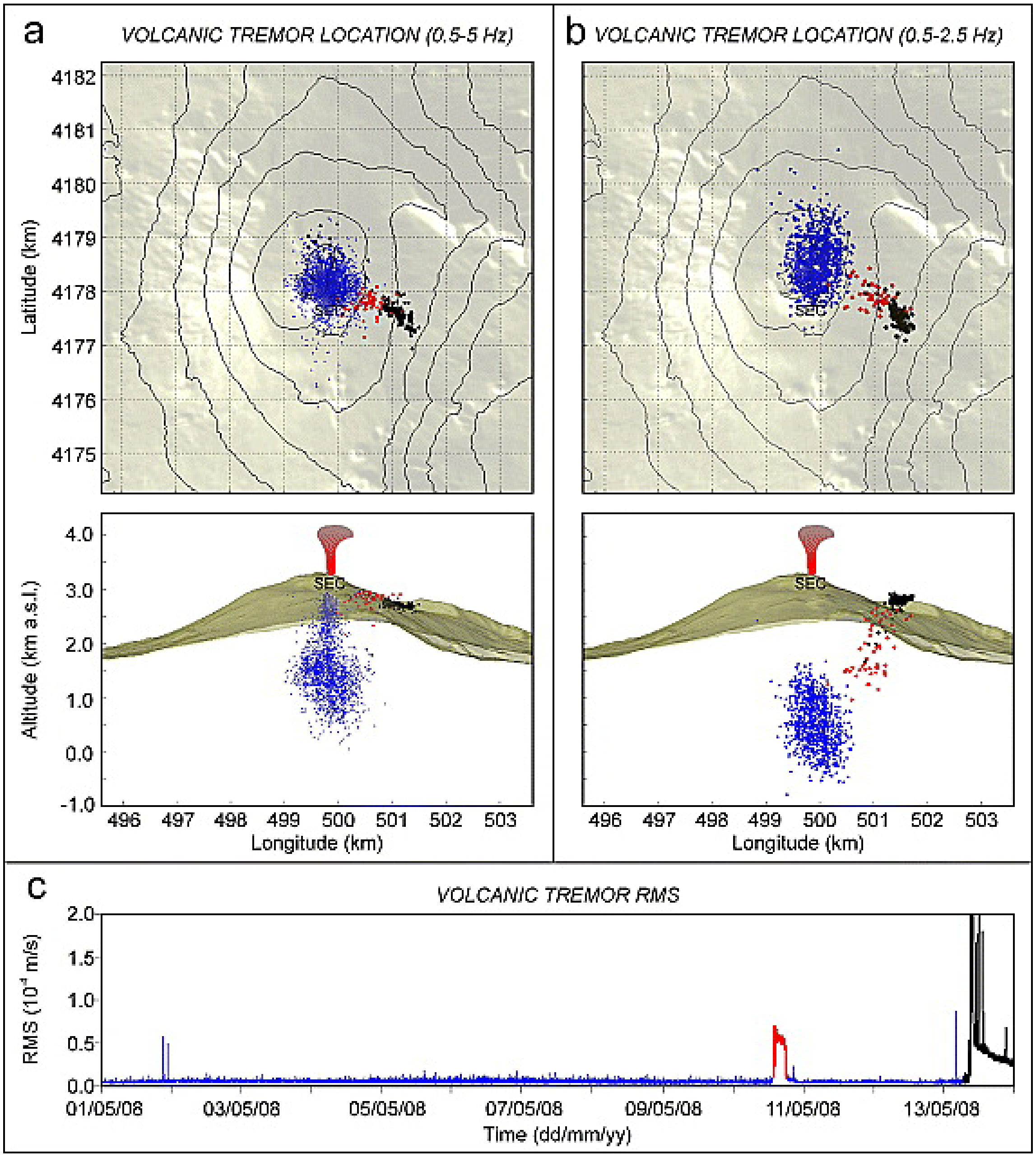
Image A:"Figure 1, A. Bonaccorso (2008)" [2]



Digital elevation model of Mount Etna with seismic stations used to locate volcanic tremor (black and white triangles) and LP events (black triangles), tiltmeters (black dots), and gravimeters (black squares) used in this work. The inset in the top left corner shows the distribution of the four summit craters (VOR, Voragine; BN, Bocca Nuova; SEC, Southeast Crater; NEC, Northeast Crater).

The data we analyze in this project, we assume, is that collected from seismic sensors that monitor 'Long Period Events'(LP). LP events are low frequencies emitted from volcano-related earthquakes and is typically used to predict eruptions[4]. Our dataset contains a reading from 10 sensors recording in parallel from such LP sensors around a volcano. Figure C shows in subplot C, the timeframe of recorded data leading to an eruption. Our dataset set contains reading between 1 and 5 days preceding an earthquake.

Image 2: "Figure 8, A. Bonaccorso (2008)" [2]:

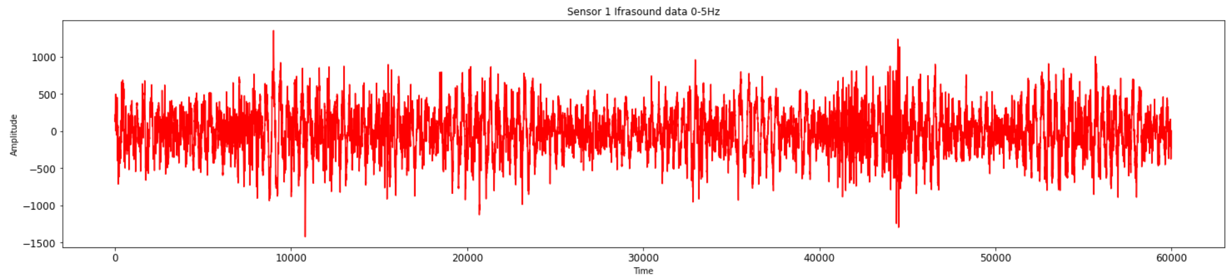


(a) WB and (b) NB volcanic tremor source locations in map view and cross section and (c) RMS during 1–13 May 2008. The colors of the dots in Figures 8a and 8b indicate the different time intervals of the locations, also reported in the RMS time series. In particular, the red, black, and blue dots indicate the tremor source locations during the 10 May lava fountain, the very first days of the 2008–2009 eruption, and the remaining period, respectively.

Seismic data file analysis

Each seismic data file, contains 10 sensor readings each a the length of 60,000 timesteps. The frequency of the sample rate is assumed at 100Hz, and therefore the segment time is 10 minutes. The X axis is time and the y axis is amplitude. The figure below shows a plot of one sensor reading from a segment_id.csv file.

Figure B:



Some sensor readings for in some segments have no data, shown below as NaN. This data is filled with Zeros.

In [25]:
```
1 df_seg_id.head()
```

Out[25]:

	sensor_1	sensor_2	sensor_3	sensor_4	sensor_5	sensor_6	sensor_7	sensor_8	sensor_
0	220.0	339.0	-336.0	364.0	NaN	492.0	-796.0	-423.0	-582.
1	178.0	221.0	-317.0	366.0	NaN	525.0	-754.0	-415.0	-932.
2	151.0	42.0	-280.0	250.0	NaN	463.0	-772.0	-229.0	-257.
3	162.0	-123.0	-243.0	288.0	NaN	303.0	-899.0	212.0	-295.
4	158.0	-287.0	-300.0	372.0	NaN	169.0	-769.0	755.0	169.

The downloaded dataset has also following file of interest, **train.csv** which contains the name of a seismic data file labelled 'segment_id' and it's respective time to eruption metadata measured in Centiseconds (1/100th of a second).

Train.csv

a) segment_id (Discrete): ID code for the data segment. Matches the name of an associated .csv data file containing 10 minutes of logs from 10 different sensors configured around a volcano. The readings have been normalized within each segment.

b) time_to_eruption (Continuous): Time to eruption-the time until the next eruption(in centiseconds - 1/100th of a second). The range of time to eruption is up to 5 days.

In [26]:

```
# train.csv containing the segment_ids and time_of_eruption
train_df = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/train.c

# Added new column to display days_to_eruption by converting time in cen
train_df['days_to_eruption']=train_df['time_to_eruption']/(100*60*60*24)
train_df.tail()
# Plotting the histogram with time to eruption data
```

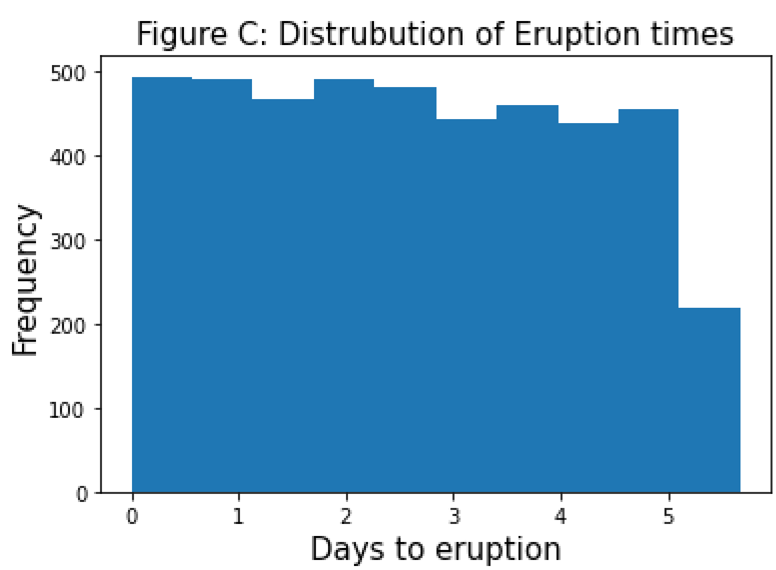
Out[26]:

	segment_id	time_to_eruption	days_to_eruption
4426	873340274	15695097	1.816562
4427	1297437712	35659379	4.127243
4428	694853998	31206935	3.611914
4429	1886987043	9598270	1.110911
4430	1100632800	20128938	2.329738

In [27]:

```
fig = plt.hist(train_df["days_to_eruption"], bins=10)
plt.xlabel('Days to eruption', size=15)
plt.ylabel('Frequency', size=15)
plt.title('Figure C: Distrubution of Eruption times', size=15)
```

Out[27]: `Text(0.5, 1.0, 'Figure C: Distrubution of Eruption times')`



1.4 Put a testing set aside and do not look at it before you test your model. Split the rest of the data into a training set and a validation set.

In [28]:

```
# Put a testing set aside and do not look at it before you test your mode
# Split the rest of the data into a training set and a validation set.

# First split data to train, test and then split train again into valida
df_consolidated = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/
X = df_consolidated
y = df_consolidated['time_to_eruption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print(len(X_train),len(X_test))

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_
print(len(X_train),len(X_val),len(X_test))
```

```
3544 887
2835 709 887
```

In [58]:

```
X_train['sensor_1_mean']
```

Out[58]:

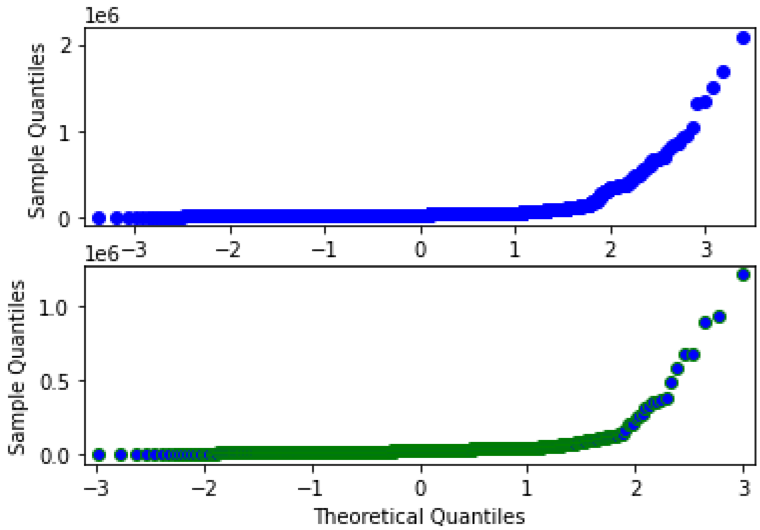
```
2990     26051.163681
1122    219274.472607
3453     15947.764163
3373     64121.001126
1776     24524.850257
            ...
2435     13827.475810
1370    479871.526249
2119     22731.269095
544      49787.728628
1868     80975.555521
Name: sensor_1_mean, Length: 2835, dtype: float64
```

1.5 Do they follow the same distribution? Use a Q-Q plot to show their relations.

Figure D: Distrubution Train and Validation sets

In [59]:

```
fig, ax = plt.subplots(2, 1, figsize=(6,4))
X_train['sensor_1_mean']
sm.qqplot(X_train['sensor_1_mean'],ax=ax[0])
sm.qqplot(X_val['sensor_1_mean'],ax=ax[1], color="green")
#sm.qqplot(X_test['sensor_1_mean'],ax=ax[1], color="red")
plt.show()
```



2. Define a problem

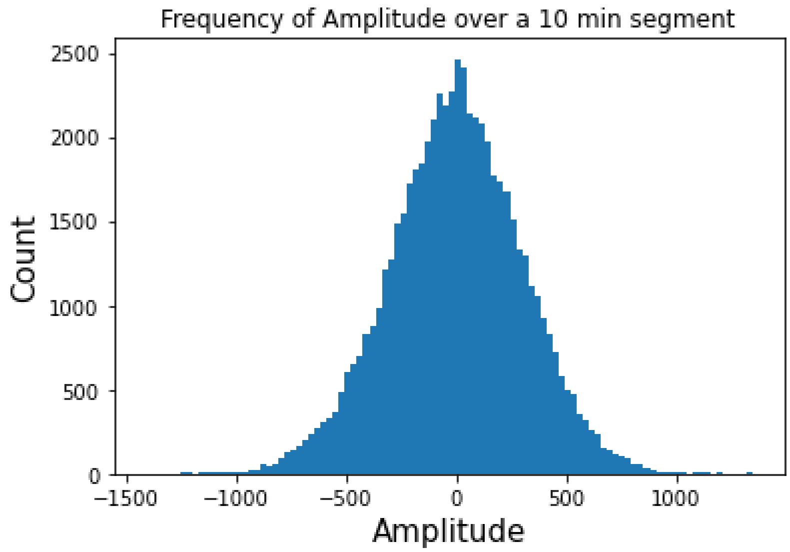
In this problem we are aiming to classify the time to eruption intervals into less than two days, between two to four days or more than 4 days given the sound signals received from the sensors configured around Mount Etna.

3. Descriptive analysis

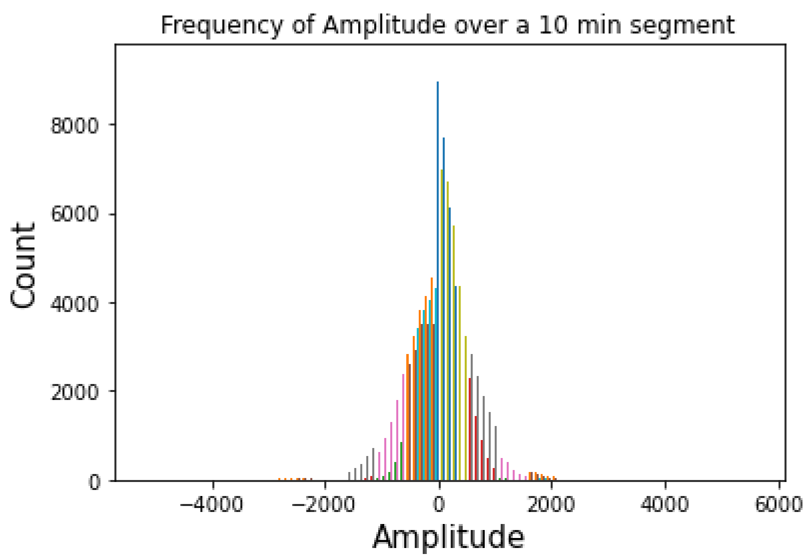
3.1 Show the histogram of some selected variables and describe what you conclude.

Sensor readings

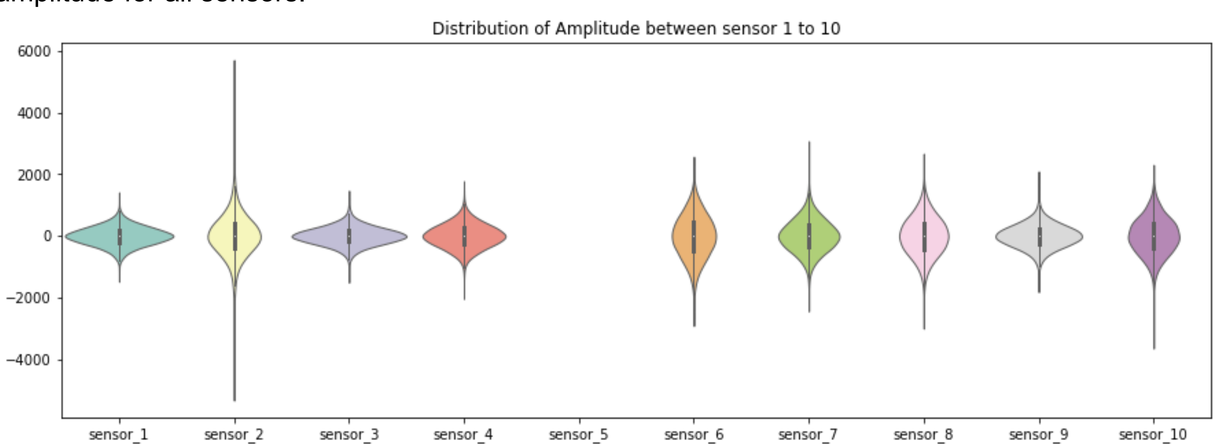
The sensor data, shown in Figure D, plotted as a histogram below that the over the 10 minute segment a gaussian distribution of the amplitude is observed. This is expected as we notice that over the segment some LP event is captured.



The Figure below shows the frequency of amplitude of all sensors in a 10 minute segment file.



Visualizing a group on sensors as violin plots, we can observe the individual distributions of amplitude for all sensors.



3.2 Show the dependence of some selected variables and describe what you conclude.

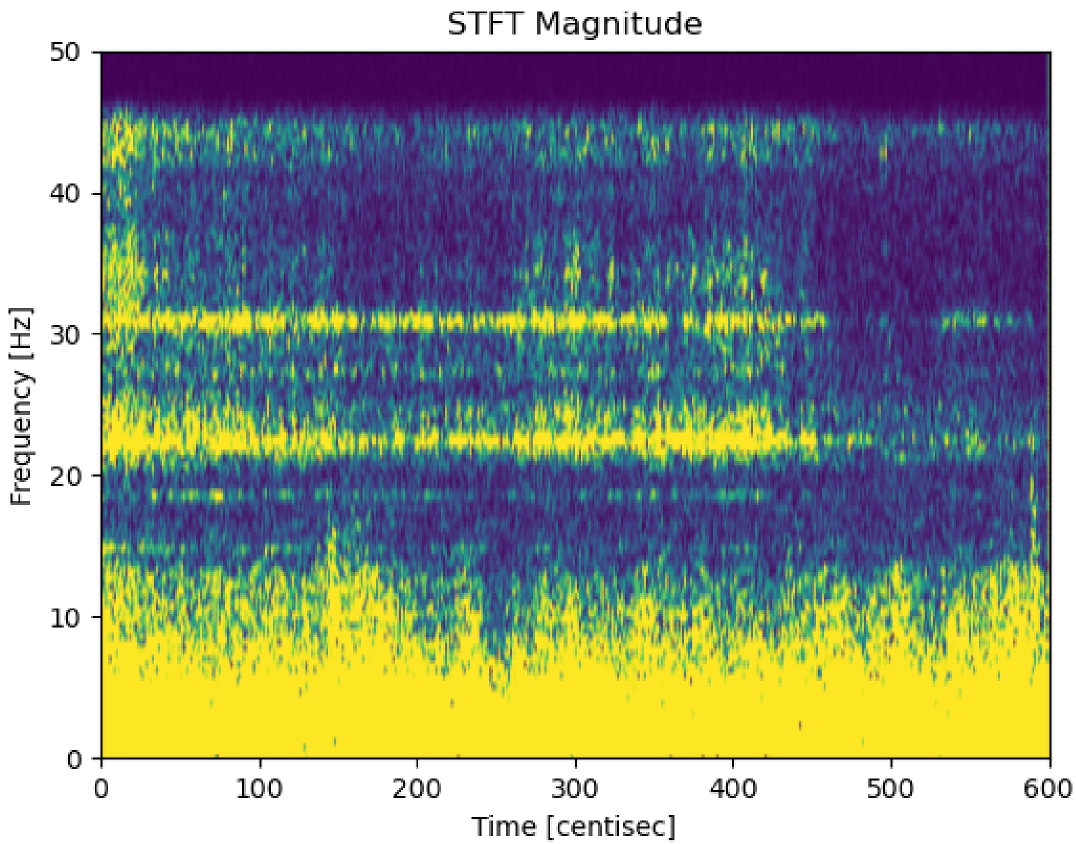
Correlation between sensors

Our raw dataset contains 10 sensors per segment data file, and where each sensor contains 60,000 data points. The visualization of the data is shown in Figure A.
We want to know if there is a correlation between the sensors, with the purpose to find out of some sensor recording the same data. We start with plotting the raw data and find that there is no apparent correlation between sensors. This is somewhat unexpected that all sensors so near zero correlation figures.

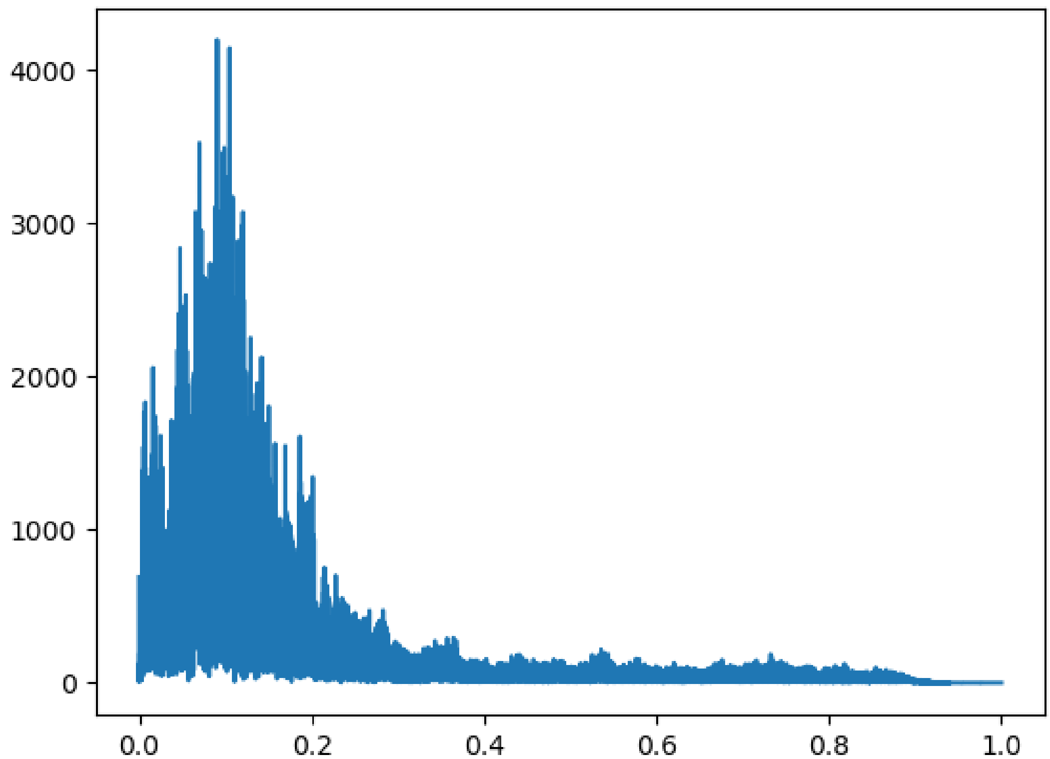
Correlation of raw sensor data over a 10 min segment

	sensor_2	sensor_3	sensor_4	sensor_5	sensor_6	sensor_7	sensor_8	sensor_9	sensor_10
sensor_2	1.00	0.02	0.02	0.05	0.01	0.01	0.01	0.02	0.02
sensor_3	0.02	1.00	0.03	0.03	0.04	0.02	0.02	0.02	0.02
sensor_4	0.02	0.03	1.00	0.01	0.03	0.03	0.05	0.04	0.03
sensor_5	0.05	0.03	0.01	1.00	0.02	0.02	0.01	0.03	0.01
sensor_6	0.01	0.04	0.03	0.02	1.00	0.00	0.04	0.03	0.01
sensor_7	0.01	0.02	0.03	0.02	0.00	1.00	0.02	0.04	0.03
sensor_8	0.01	0.02	0.05	0.01	0.04	0.02	1.00	0.04	0.02
sensor_9	0.02	0.02	0.04	0.03	0.03	0.04	0.04	1.00	0.02
sensor_10	0.02	0.02	0.03	0.01	0.01	0.03	0.02	0.02	1.00

The raw in this form however, may not be the best representation of our input data to model. The seismic data has captured the frequency of the infrasound, to visualize we should transform our data. To do this we use a Short-time Fourier transform (STFT) apply 100Hz as the sampling frequency. What we observe is a constant low frequency emission which we are interested in 0-5Hz typical of LP events. The higher frequencies, we assume to be noise data.

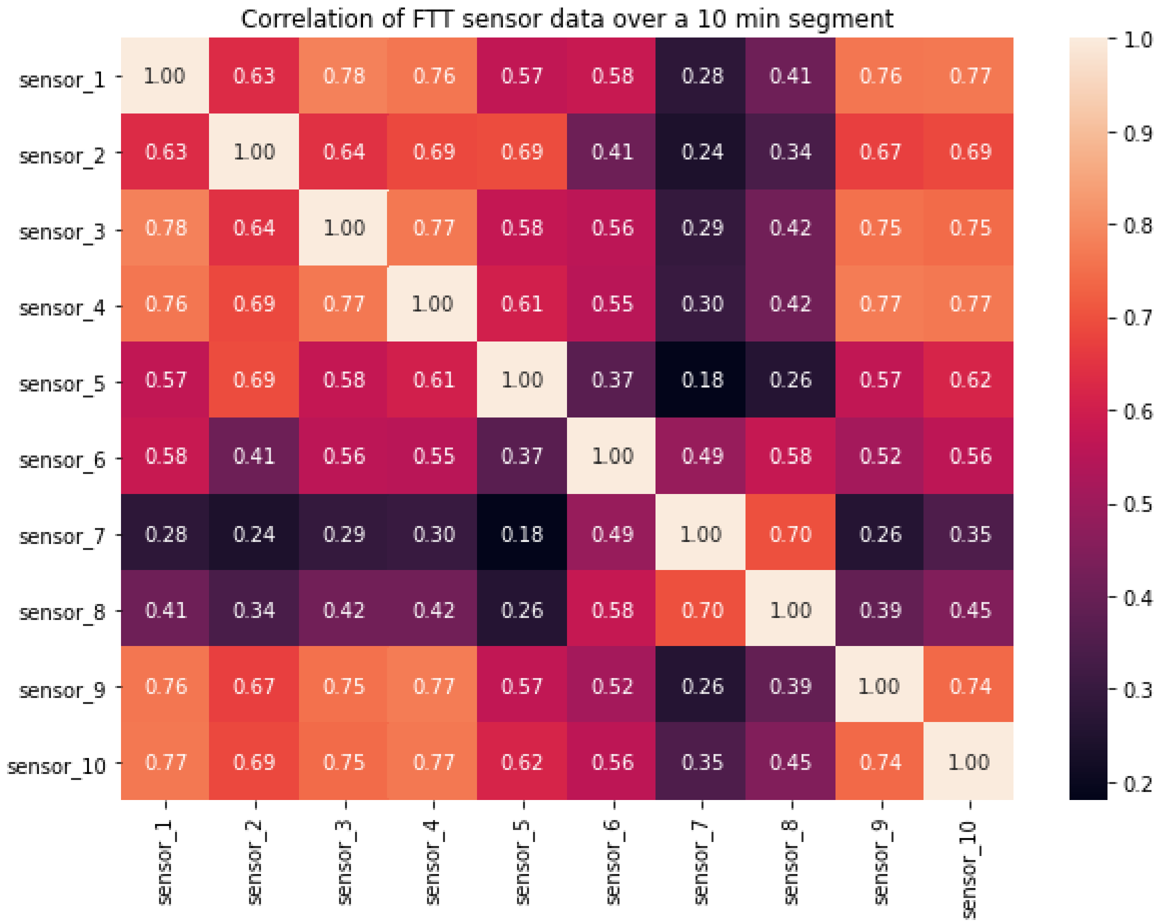


We will transform our data from time to frequency using FFT which will look at the entire 10 min sequence.



Correlation of FTT sensor data over a 10 min segment

We the assume that rather than to observe the amplitude over time, given in the raw data; we can observe the frequency and amplitude. Figure H, below, shows the correlation between sensors taking frequency in account. This shows a higher degree of correlation between some sensors of an average reading of 100 segments.



3.3 Describe the data using its range, sample mean, sample standard deviation and some quantiles.

Data preprocessing

We have processed our data using: pipe_fft.py : Performs Fast-Fourier Transform on raw data.

pipe_data_stats.py : Reduces data 60,000 per sensor to collected the mean and std of each segment.

The out of the pipe is labeled as dataset.csv and is separated in training set X_train and validation set y_train:

In [129]:

```
X_train.head()
```

Out[129]:

	segment_id	time_to_eruption	sensor_1_mean	sensor_1_std	sensor_2_mean	sensor_:
3515	1400929225	26164471	27218.702557	129179.999334	136301.300507	243456.78
3572	1380340436	21127299	10072.123801	25357.513082	45524.819614	84638.58
1938	612447943	25260944	23359.622724	75716.776378	67884.197697	151522.13
3864	1710136076	20853878	24953.684385	106643.545017	77728.862259	117583.10
1482	1774928274	13445802	26338.315345	50561.164457	0.000000	0.00

5 rows × 22 columns

In [137]:

```
X_train.describe()
```

Out[137]:

	segment_id	time_to_eruption	sensor_1_mean	sensor_1_std	sensor_2_mean	sensor_
count	2.835000e+03	2.835000e+03	2.835000e+03	2.835000e+03	2.835000e+03	2.83500
mean	1.074869e+09	2.299516e+07	4.868628e+04	1.573934e+05	1.011687e+05	2.19503
std	6.197539e+08	1.358718e+07	1.050485e+05	3.836291e+05	1.569529e+05	4.12168
min	5.131810e+05	2.692900e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.00000
25%	5.509362e+08	1.123808e+07	2.055007e+04	6.279642e+04	4.490200e+04	8.61827
50%	1.073020e+09	2.267582e+07	2.646204e+04	8.246519e+04	6.629467e+04	1.21012
75%	1.607460e+09	3.442823e+07	3.993235e+04	1.268622e+05	1.114328e+05	2.18904
max	2.146939e+09	4.881429e+07	2.088111e+06	5.072697e+06	1.835810e+06	4.62482

8 rows × 22 columns

3.4 Choose a visualization method to explore the data set.

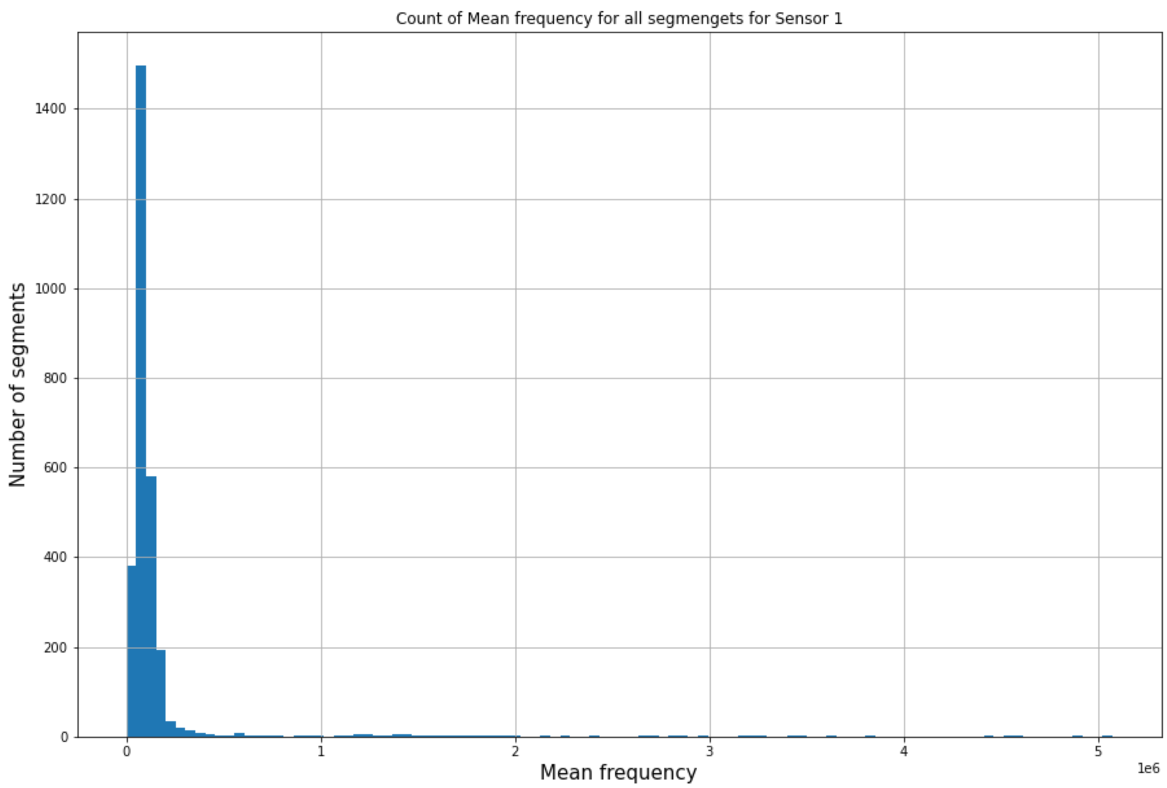
After preprocessing our raw data using fft, we reduced our data to mean and std of the infra-sound frequency and amplitude of the data for each segment. In doing so we can observe and summarize the frequency of the entire dataset. As we visualize sensor data this data, via a histogram and scatter plots, we observe that most of the 2835 training sample segments have a scattering of potential outliers.

In [29]:

```
sensor_1 = X_train['sensor_1_std']
plt.figure(figsize=(15,10))
plt.title('Count of Mean frequency for all segmengets for Sensor 1')
#fig = sns.boxplot(data=sensor_1, palette="Set3")
#fig = sns.violinplot(data=sensor_1, palette="Set3", bw=.2, cut=1, linew
plt.xlabel('Mean frequency', size=15)
plt.ylabel('Number of segments', size=15)
sensor_1.hist(bins=100)

```

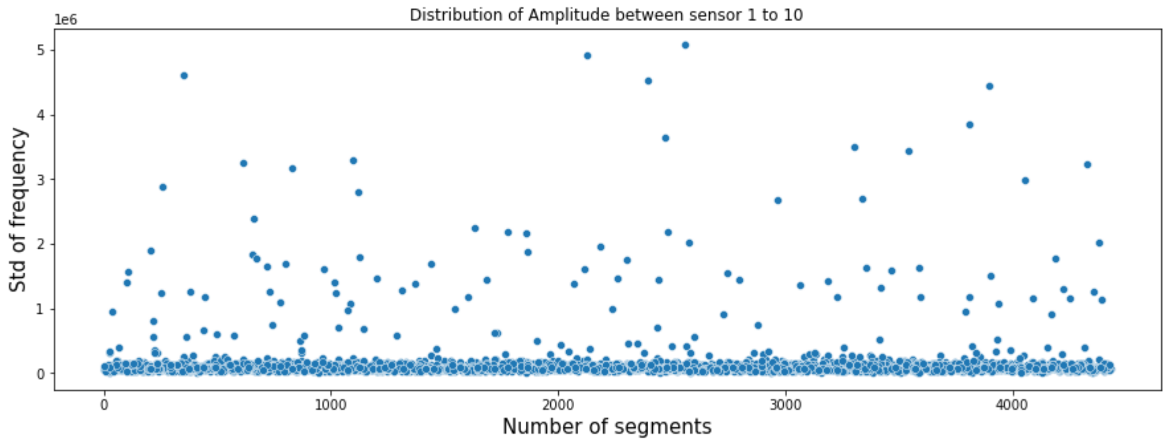
Out[29]:

```
<AxesSubplot:title={'center':'Count of Mean frequency for all segmengets fo
r Sensor 1'}, xlabel='Mean frequency', ylabel='Number of segments'>
```



In [30]:

```
sensor_1 = X_train['sensor_1_std']
plt.figure(figsize=(15,5))
plt.title('Distribution of Amplitude between sensor 1 to 10')
plt.xlabel('Number of segments', size=15)
plt.ylabel('Std of frequency', size=15)
fig = sns.scatterplot(data=sensor_1)

```



3.5 Explain how your analysis relates to the objective of your project, i.e. why are these selected variables important?

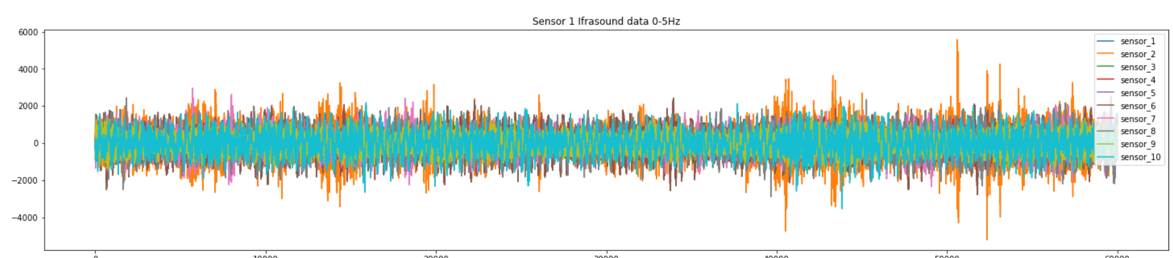
Out dataset contains a set of 10 readings of seismic sensors plotted around a volcano.

In [31]:

```
sensor_1 = df_seg_id
sensor_1.plot(figsize=(25, 5), title='Sensor 1 Ifrasound data 0-5Hz')
```

Out[31]:

```
<AxesSubplot:title={'center':'Sensor 1 Ifrasound data 0-5Hz'}>
```



Our starting point was to find which if any sensor readings similar reading so that we can we can remove redundant data and reduce the feature input. We look at the correlation between sensors and transformed data to a more useful state for the problem, modeling frequency over amplitude rather than amplitude over time.

Using the correlation we estimated that some sensors are more correlated than others, and a low correlation would be more important to use as the sensors may be measuring different elements of LP events around an eruption. We will further examine which sensors to use after hypothesis testing on the standard deviation of frequency.

4. Probability Distribution

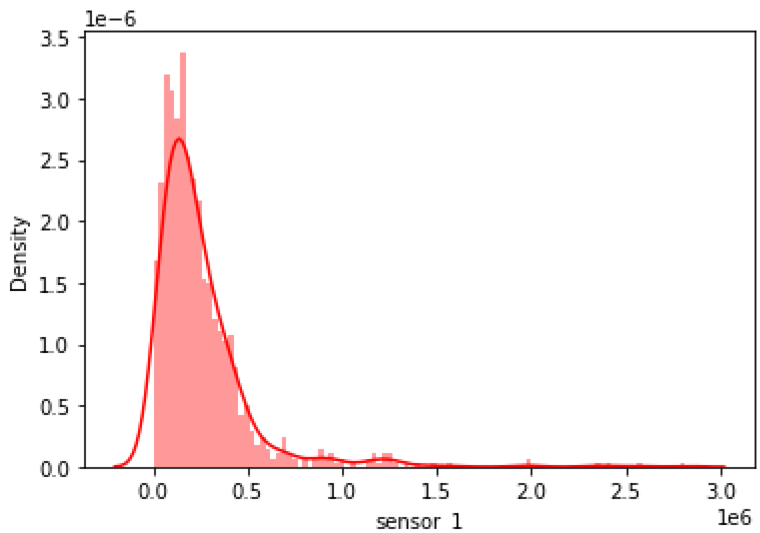
4.1 Use probability distributions to describe some selected variables.

We plot the distribution of frequency of a sensor 'sensor 1' of a random segment. We can see that the area under the curve between 0 and 0.5 is where the highest probabality of frequency occur. Sensor_data looks gaussian and right skewed. The parameters are mean(mu) and standard deviation (sigma). Below plot shows the signal at the median and mean around 0.

In [32]:

```
from scipy.fft import fft, fftshift, rfft, fftfreq

seg_id = train_df['segment_id'][0]
fft_raw_data = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/fft_
signal_transform = fft_raw_data['sensor_1']
freq = 1000

xf = np.linspace(0, 1, freq)
fft_shift = signal_transform[:freq]

pdf = stats.norm.pdf(fft_shift, fft_shift.mean(), fft_shift.std())
ax = sns.distplot(fft_shift, kde=True, color='red', bins=100)
plt.show()

```

```
C:\Users\IJENKINS\Anaconda3\lib\site-packages\seaborn\distributions.py:255
1: FutureWarning: `distplot` is a deprecated function and will be removed i
n a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```



4.3 Hypothesis testing

Our Null Hypothesis is that the sensor readings from two different samples(sensor1 and sensor9) show same readings. The experiment was done to collect the sensor readings from 10 different sensors across a segment. Each segment consists of a time interval of 10 mins. As the sample is taken from normal distribution the parameters of interest are Mean and std. deviation.

Null hypothesis: H0: mean_sensor_1=mean_sensor_9 H0: mu1-mu2=0, where mu1 is the mean of first sample and mu2 is the mean of second sample Alternative hypothesis: Ha: mean_sensor_1 is not equal to mean_sensor_9 Ha: mu-mu2 not equal to 0

We will run another test for the standard deviations: H'0: std1-std2=0, where std1 is the std deviation of first sample and std2 is the std. deviation of second sample H'a: std1-std2 is not equal to 0

We chose significance level as α=0.05 and then used the mean sensor data from consolidated file to run the tests. We have used scipy.stats ttest_ind to calculate the p_value.

If p_value < 0.05 i.e. the test statistic falls in the rejection regions of the null distribution then we reject the null hypothesis. If p_value > 0.05 then we do not reject the null hypothesis.

In [33]:
```
# Testing H0 for sensor 1 and 9
stats.ttest_ind(X_train[['sensor_1_mean','sensor_1_std']],X_train[['sens

# As per the test statistic p_value for both mean and std deviation is >
```

Out[33]:
```
Ttest_indResult(statistic=array([-1.04552834, -0.03595585]), pvalue=array
([0.29582343, 0.97131883]))
```

In [19]:
```
# Testing H0 for sensor 1 and 4
stats.ttest_ind(df_consolidated[['sensor_1_mean','sensor_1_std']],df_con

# As per the test statistic p_value for both mean and std deviation is >
```

Out[19]:
```
Ttest_indResult(statistic=array([-2.52575662,  0.3345197 ]), pvalue=array
([0.01156223, 0.73799535]))
```

In [20]:
```
# Testing H0 for sensor 1 and 5
stats.ttest_ind(df_consolidated[['sensor_1_mean','sensor_1_std']],df_con

# As per the test statistic p_value for both mean and std deviation is <
```

Out[20]:
```
Ttest_indResult(statistic=array([14.1618381 , 16.46582789]), pvalue=array
([4.88185951e-45, 5.00510169e-60]))
```

Conclusion from the test:

As seen from above tests, sensor 1,9 and sensor 1,10 have mean and std deviations in the same range hence we should use likely use one of these two pairs of sensors in our training features.

Alternately, sensor 1 and 5 mean and std deviations are way apart hence both the sensors can be used for modelling purposes.

5. Predictive Analysis

5.1 Apply two predictive machine learning models to solve your problem.

Using SVM technique to classify the class for the time_of_eruption considering three classes:

class A : with time_of_eruption <= 2 days

class B : with time_of_eruption >2 days and <=4 days

class C : with time_of_eruption >4 days

In [21]:

```
# Added new column to display days_to_eruption by converting time in cen
df_consolidated['days_to_eruption']=round(df_consolidated['time_to_erupt
df_consolidated.tail()

```

Out[21]:

	segment_id	time_to_eruption	sensor_1_mean	sensor_1_std	sensor_2_mean	sensor_
4426	873340274	15695097	50223.301755	141617.748867	111255.219243	201801.89
4427	1297437712	35659379	51042.947194	150591.550806	0.000000	0.00
4428	694853998	31206935	10032.737288	25258.098586	72469.377624	100467.32
4429	1886987043	9598270	72848.367371	91930.914799	0.000000	0.00
4430	1100632800	20128938	18884.277615	64064.019529	49003.558585	94161.22

5 rows × 24 columns

In [22]:

```
# Create a new column to identify the classes based on the days of erupt
# 'A' for days_to_eruption <=2
# 'B' for days_to_eruption >=2 and <4
# 'C' for days_to_eruption >4

def classes(x):
    classValue = ""
    if x<=2:
        classValue = "A"
    elif (x>2 and x<=4):
        classValue = "B"
    else:
        classValue = "C"

    return classValue


df_consolidated['class']=df_consolidated['days_to_eruption'].apply(class
df_consolidated.to_csv("../../predict-volcanic-eruptions-ingv-oe/fft_sta
#df_consolidated=df_consolidated.fillna(0)
#df_consolidated.tail(10)

```

In [10]:

```
df_consolidated = pd.read_csv("../../predict-volcanic-eruptions-ingv-oe/
df_consolidated.head()
```

Out[10]:

	segment_id	time_to_eruption	sensor_1_mean	sensor_1_std	sensor_2_mean	sensor_2_st
0	1136037770	12262005	18839.763682	71814.619373	79794.038063	145748.08921
1	1969647810	32739612	30404.978085	102982.692327	68742.438608	146640.43398
2	1895879680	14965999	18382.472061	56249.752906	89613.013241	182899.73015
3	2068207140	26469720	16290.547583	51876.076170	45764.101203	93557.32338
4	192955606	31072429	18969.185026	61231.679644	0.000000	0.00000

5 rows × 24 columns

In [23]:

```
X = df_consolidated[['sensor_1_mean','sensor_2_mean','sensor_10_mean','se
X = df_consolidated[['sensor_1_mean','sensor_5_mean','sensor_2_mean','sen
y = df_consolidated['class']
y.head()
```

Out[23]:

```
0    A
1    B
2    A
3    B
4    B
Name: class, dtype: object
```

In [12]:

```
# Put a testing set aside and do not look at it before you test your mode
# Split the rest of the data into a training set and a validation set.

# First split data to train, test and then split train again into valida

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print(len(X_train),len(X_test))

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_
print(len(X_train),len(X_val),len(X_test))
```

```
3544 887
2835 709 887
```

In [13]:

```
1
2 X_train.head()
```

Out[13]:

	sensor_1_mean	sensor_5_mean	sensor_2_mean	sensor_9_mean	sensor_1_std	sensoı
4088	37653.529221	0.000000	113445.646871	60939.370320	173166.214154	0.
4108	32557.534683	2044.087463	169902.257275	53322.555272	136458.961542	3250.
657	41070.118702	35340.785952	151634.146372	29083.367517	82287.886830	52165.
4139	18533.488897	19559.239313	41398.899271	18476.618548	75144.228865	38651.
1726	71759.451145	20632.048058	0.000000	134064.233209	127122.239133	41496.

5.2 For each model, state the following:

What is the name of the model?

SVM - Support Vector Machine is a set of supervised learning method used for classification, Regression and outlier detection. It is effective in high dimension spaces. It looks at the data and divides it into 2 or more categories.

What is the mathematical expression of the model?

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2. \qquad (2)$$

C - It controls the trade off between smooth decision boundary and classifying training points correctly. A large value of C means we will get more training points correctly. If C is small, the penalty for misclassified points is low so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. By default its 1.

gamma - Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class).

For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and gamma parameter need to optimized simultaneously.

The parametrs are what will be learned, which are the boundaries for data.

Reference: [https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167 (https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167)](https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167)

In [14]:
```
# Fitting the training data into the SVC model
model=SVC(kernel="rbf")
model.fit(X_train,y_train)
# Calculating the score for test data
model.score(X_val, y_val)
```

Out[14]: 0.5557122708039492

In [15]:
```
# Fitting the training data into the SVC model
from sklearn import tree
from sklearn.neighbors import NearestCentroid
model= tree.DecisionTreeClassifier()
model.fit(X_train,y_train)
# Calculating the score for test data
model.score(X_val, y_val)
```

Out[15]: 0.7122708039492243

In [16]:
```
# Fitting the training data into the SVC model
from sklearn import tree
from sklearn.neighbors import NearestCentroid
model= NearestCentroid()
model.fit(X_train,y_train)
# Calculating the score for test data
model.score(X_val, y_val)
```

Out[16]: 0.3723554301833568

What is the name of the model?

Guassian Naive Bayes classifier

What are the hyperparameters?

It does not have any parameters to tune.

In [17]:
```
model= GaussianNB()
model.fit(X_train, y_train)
model.score(X_test,y_test)
```

Out[17]: 0.2649379932356257

Evaluate their performance.

From our test, our base test for Gaussian performed poorly, SVC was only slighly better, experimening we found Desicion Tree improved the result with about 70 % score on out validation set.

Conclusion

Our goal was to build an early warning system by creating a predictive model to infer on seismic LP event data. Such data is continually monitored around a volcanos and could potentially be passed to a one such model for analysis. We had access to data from 10 sensors around a volcano, each of 60,000 timesteps at sampling frequency of 100Hz, 10 min of data per time segment. Each we a total of 4431 segment files.

We decided to process the raw data using a fast-Fourier transform and reduce the dimensionality of data using the mean and standard deviation of the frequency and amplitude for each segment. After analysis of the sensor data we further eliminated the number of features, by reducing the sensor input to 3 sensors. These three sensors were chosen that proves the our null hypothesis wrong, such that the sensor data was statically different form each other.

We choose to predict several classes instead of a continuous value to reduce the complexity of the problem but hindsight we could chosen a regression model entirely as we already had a time to eruption data. We hoped however that predict a class, i,e, 0-2 days, 2-4 days, 4+ days should be a easier task.

We choose two models, SVC and Gaussian as both are appropriate classification models. Gaussian was chosen as a base model and SVC due to being able to handle higher dimensionality well. The input data of our dataset is already known to a complex problem proposed by the INGV, that is to interpret the LP volcanic-earthquake events. With our current approach, our models have not yielded interpretable results, with a model score around 50 percent. This maybe that the dimension is now too low, using just the standard deviation and one could include the min and max quantiles. Further we propose to use an LSTM model that is well known to model time-series data.

References

[1] https://www.usgs.gov/natural-hazards/volcano-hazards/monitoring-volcano-seismicity-provides-insight-volcanic-structure (https://www.usgs.gov/natural-hazards/volcano-hazards/monitoring-volcano-seismicity-provides-insight-volcanic-structure) [2] Multidisciplinary investigation on a lava fountain preceding a flank eruption: The 10 May 2008 Etna case, A. Bonaccorso, 2008 [4] https://www.pbs.org/wgbh/nova/volcano/seis_lpe.html#:~:text=LP%20events%20are%20volcano%2I
(https://www.pbs.org/wgbh/nova/volcano/seis_lpe.html#:~:text=LP%20events%20are%20volcano%2

In [ ]:

```
1
```