ML ASSIGNMENT VOLCANO ON DATASET

SE22MAID002

PARAM SHIV ASHISH

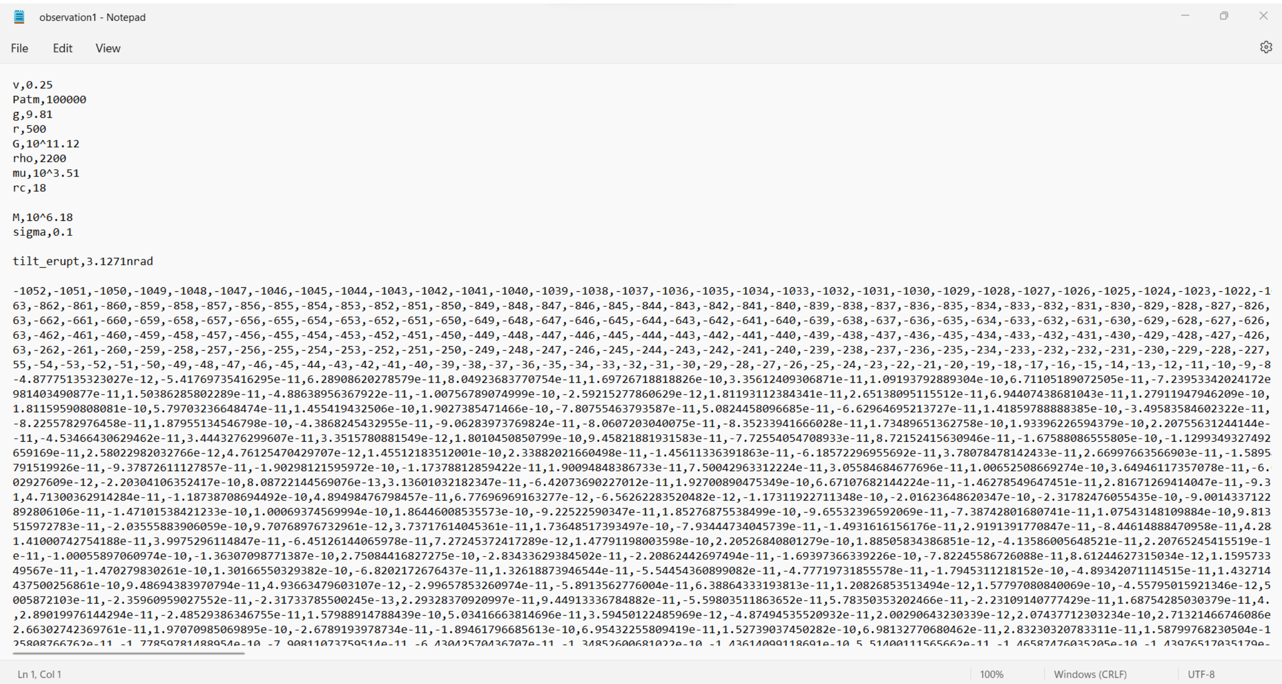
AI AND DS

INITIAL DATA THAT WAS GIVEN:

CONTAINS 10 VOLCANO FOLDERS ASSUMING EACH FOLDER TO CORRESPOND WITH A VOLCANO IN REAL LIFE. CERTAIN NUMBER OF OBSERVATIONS IN EACH FOLDER

```
Volcano_Dataset/
├─ Volcano1/
│   ├─ Observation1.txt
│   ├─ Observation2.txt
│   ├─ Observation3.txt
│   ├─ Observation4.txt
│   ├─ Observation5.txt
├─ Volcano2/
├─ Volcano3/
├─ Volcano4/
├─ Volcano5/
├─ Volcano6/
├─ Volcano7/
├─ Volcano8/
├─ Volcano9/
├─ Volcano10/
```

AN OBSERVATION TEXT FILE FORMAT INSIDE VOLCANO SUB-FOLDERS



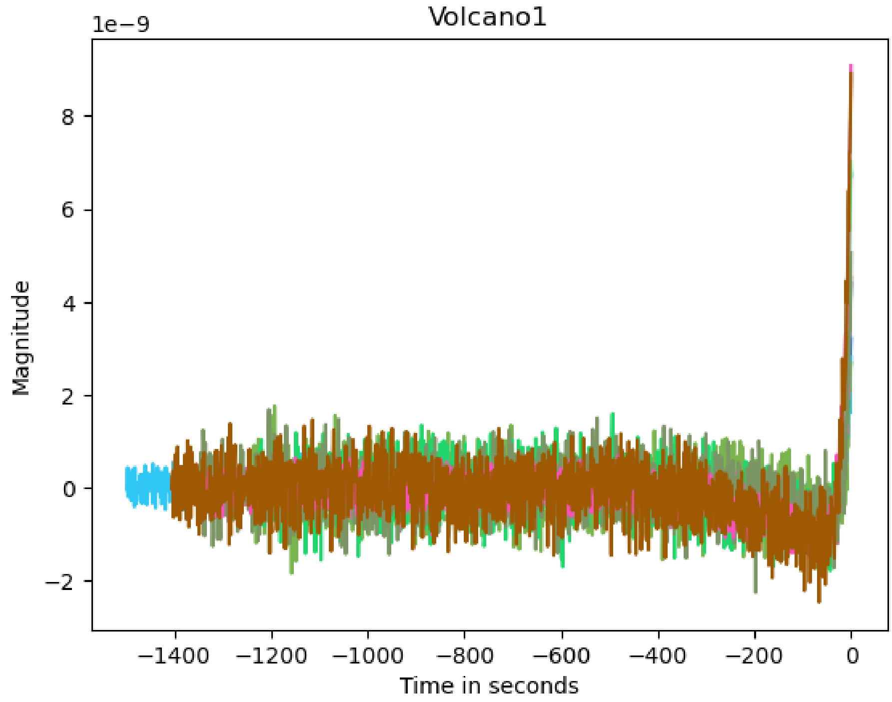
THE FIRST SERIES OF VALUES IN SECONDS STARTING IN THE PAST, INDICATED WITH THE NEGATIVE SIGN UP TO 0 WHICH CAN BE CONSIDERED AS CURRENT TIME

THE SECONDS SERIES OF VALUES ARE THE CORRESPONDING MAGNITUDES AT THOSE TIME STAMPS WHICH IS THE HIGHEST AT 0TH SECOND

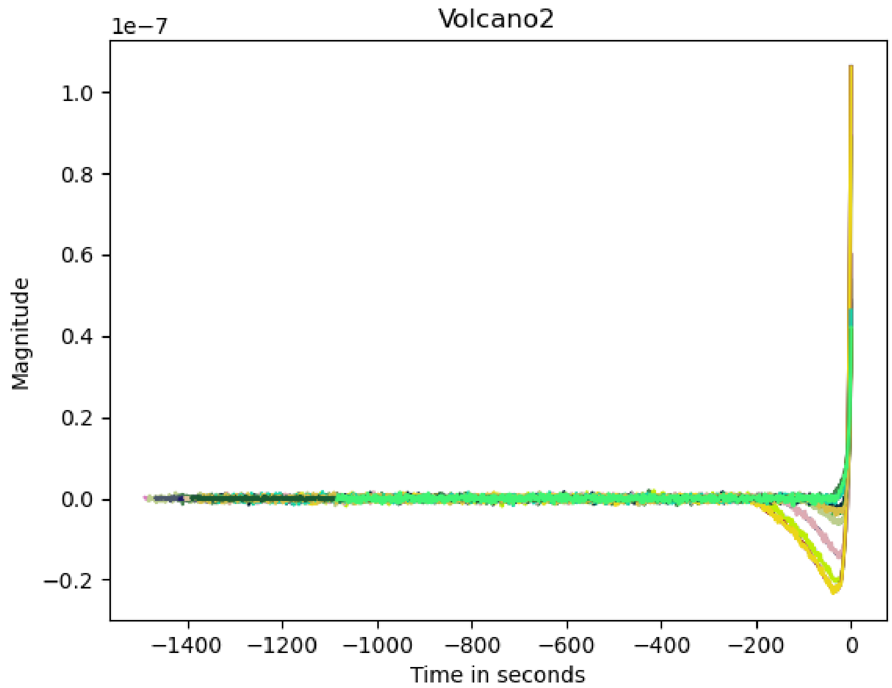
PLOTTING INITIAL DATA THAT WAS GIVEN

VALCANO - 1

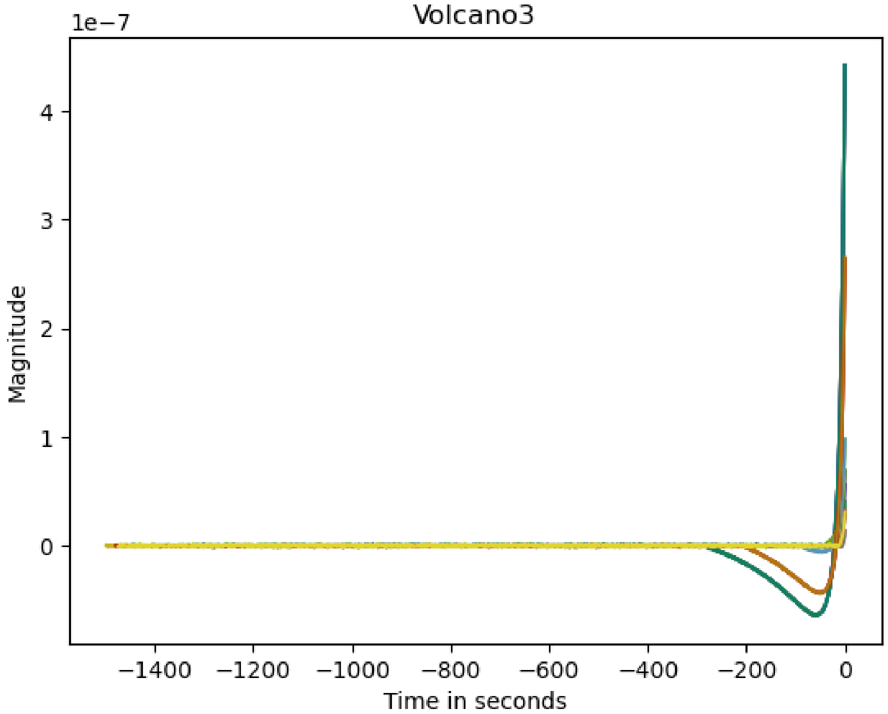
OBERVATIONS - 12



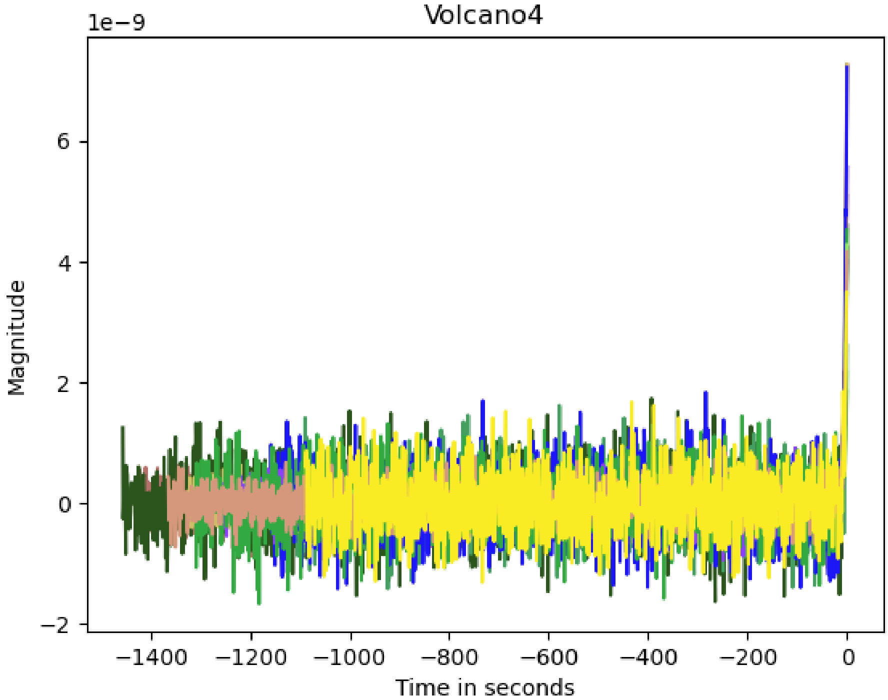
VALCANO - 2
OBERVATIONS - 27



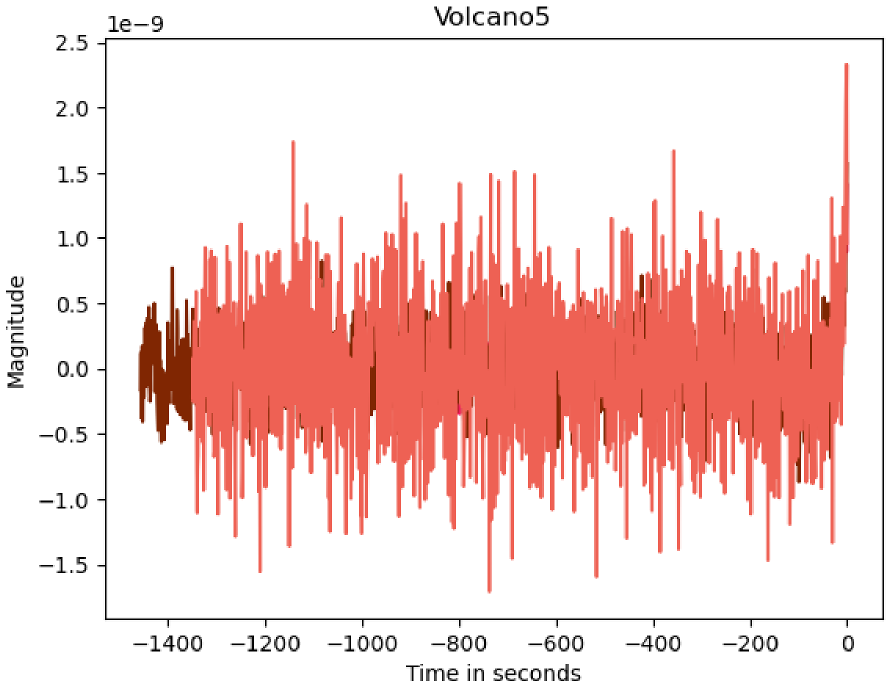
VALCANO - 3
OBERVATIONS - 30



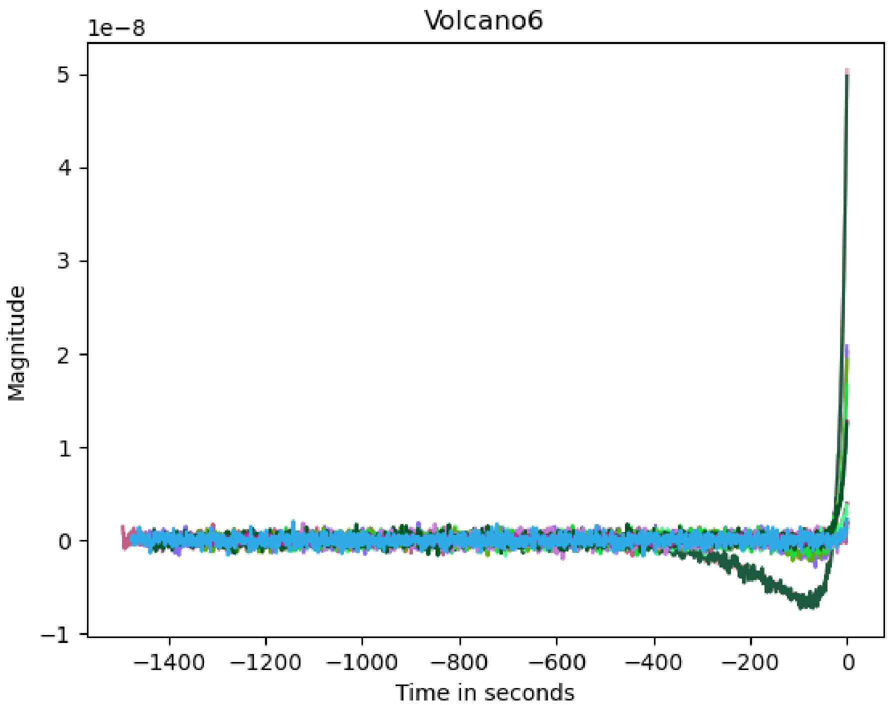
VALCANO - 4
OBERVATIONS - 15



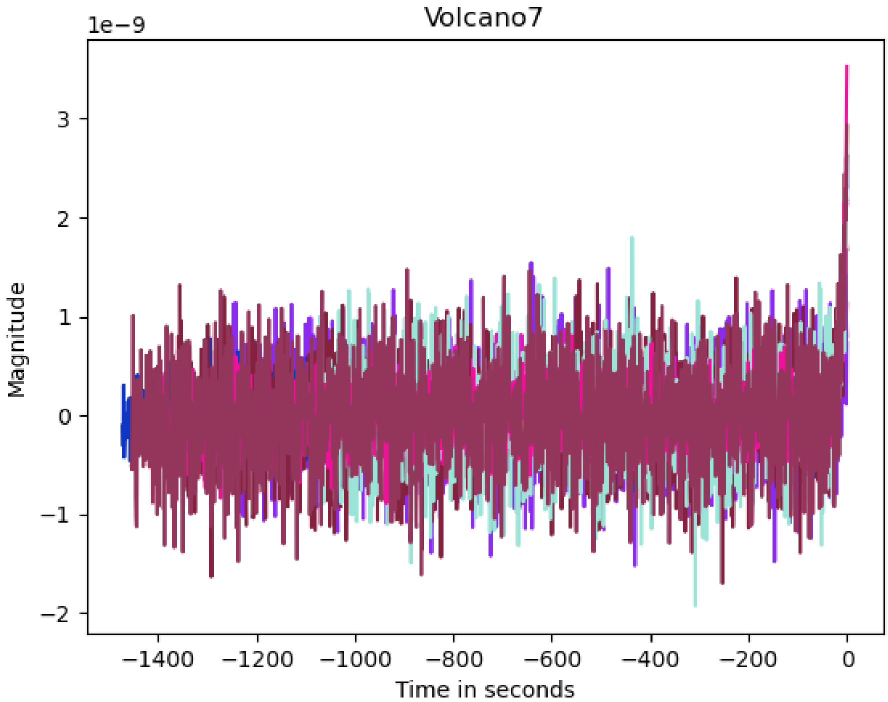
VALCANO - 5
OBERVATIONS - 3



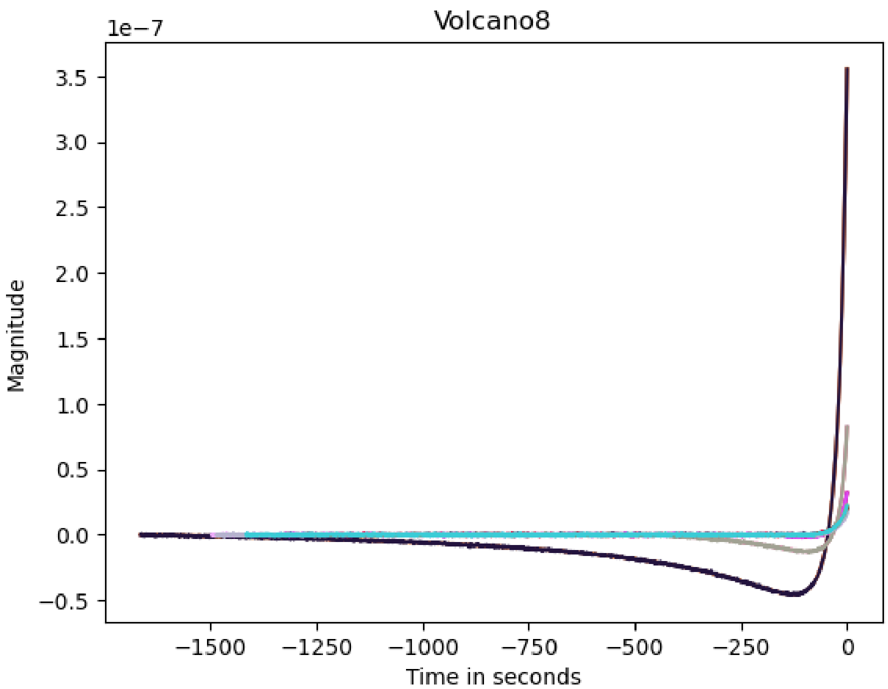
VALCANO - 6
OBERVATIONS - 27



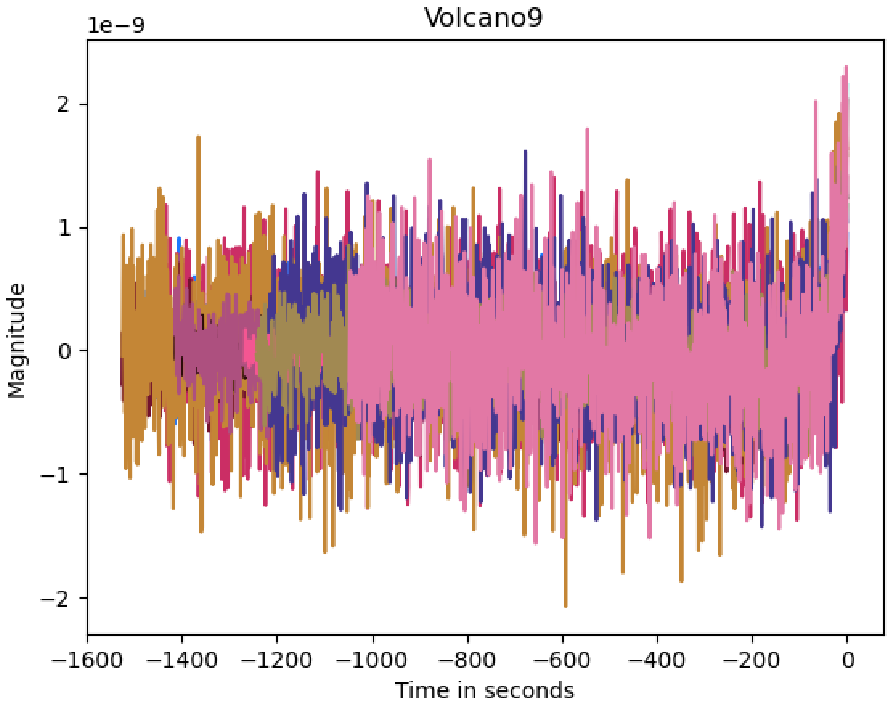
VALCANO - 7
OBERVATIONS - 12



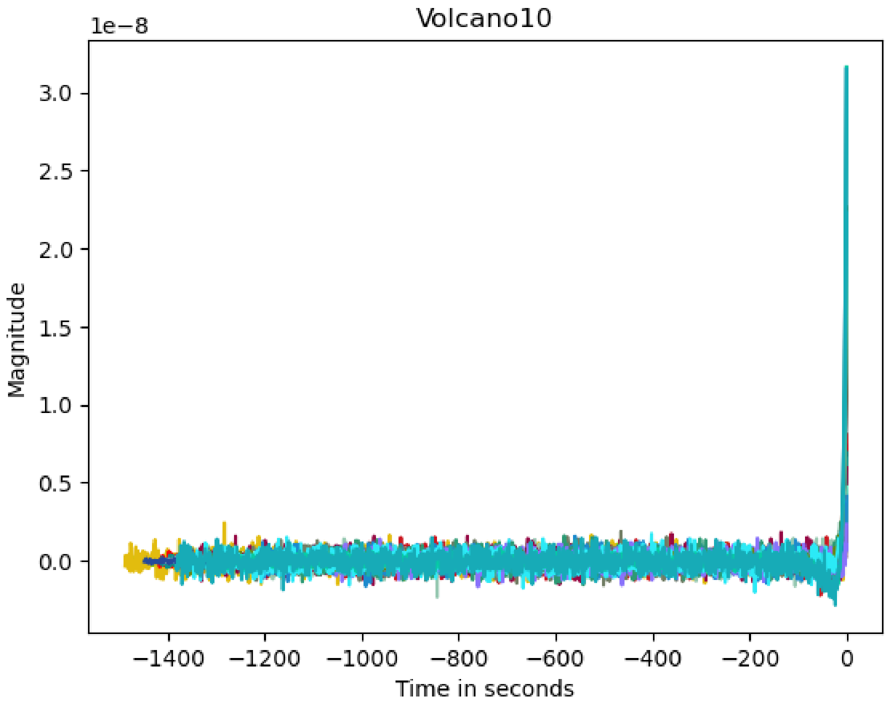
VALCANO - 8
OBERVATIONS - 21



VALCANO - 9
OBERVATIONS - 12



VALCANO - 10
OBERVATIONS - 30



FEATURE ENGINEERING FOR DIRECT FORECASTING

The direct forecasting strategy uses an ML model for each forecasting step. More specifically, *each model is trained using as target the time series shifted of the desired number of time periods into the future*. Now we need to predict the magnitude that sensor reads at the time of eruption which is corresponding to the 0 seconds in our data. In this case, each timestamp in the target time series is chosen as magnitude at 0^{th} second. The feature values are then taken as a consecutive chain of time series magnitudes, here I have considered consecutive chain of length 300. In this way, we create a model trained to predict future value of magnitude. The same procedure is repeated for all forecasting steps.

The direct method outlined above does not generate error accumulation at each forecast, but it has larger computational cost making it not suitable for large forecasting horizons.

Moreover, it cannot model statistical relationships among predictions since the models used for each time step are independent.

IMPLEMENTING A SIMPLE DIRECT FORECASTING FOR VOLCANO DATASET

TIME	**MAGNITUDE**
-1000	1.23E-20
-999	1.45E-20
-998	1.60E-20
-997	1.87E-20
.	.
.	.
.	.
.	.
.	.
.	.
4	4.12E-20
3	4.22E-20
2	4.45E-20
1	4.56E-20
0	4.79E-20

THE ABOVE TIME SERIES CAN BE TRANSFORMED TO X FEATURES AND Y OUTPUTS AS SHOWN BELOW

X1	**X2**	**X3**	**.**	**.**	**.**	**X298**	**X299**	**X300**	**Y1**	**Y2**
1.23E-20	**1.45E-20**	**1.60E-20**	**.**	**.**	**.**	**2.12E-20**	**2.33E-20**	**2.60E-20**	**4.79E-20**	**700**
1.45E-20	**1.60E-20**	**1.87E-20**	**.**	**.**	**.**	**2.33E-20**	**2.60E-20**	**2.95E-20**	**4.79E-20**	**701**

THE SAMPLES IN THIS STRUCTURE CAN THEN BE USED TO MAKE LINEAR REGRESSION MODELS.

TO ACHIEVE THIS IN PYTHON I HAVE WRITTEN A FUNCTION WHICH EXTRACTS ALL THE SAMPLES FROM THE CONSIDERED OBSERVATIONS.

(ONLY CONSIDERING HALF OF THE OBSERVATIONS AS THE DATASET IS TOO LARGE)

```
import os
import math
def FeatureEngineering(folder,mainDF):
    print(f"VALCANO - {folder}")
    initial_count = 0
    dir = (f"C:\\Users\\crims\\Desktop\\ML Assignment\\Volcano_Dataset\\Volcano{folder}")
    for path in os.listdir(dir):
        if os.path.isfile(os.path.join(dir, path)):
            initial_count += 1
# Only considering half of the observations in each volcano folder as this is compute intensive and
# can easily be scaled by considering all the observations of a give volcano
    print(f"Observations Considered - {math.ceil(initial_count/2)}")
    for x in range(1,math.ceil(initial_count/2)):
# change the path as per your folder path where this project folder is located
        f = open(f"C:\\Users\\crims\\Desktop\\ML Assignment\\Volcano_Dataset\\Volcano{folder}\\observation{x}.txt")
        lines = f.read().splitlines()
        main = [l for l in lines if l != " "]
        tempP = main[-1].split(",")
        p = [float(str(e)) for e in tempP]
        for idx,x in enumerate(p):
            if idx + 300 < len(p):
                instance = [p[i] for i in range(idx,idx+300)]
                instance.append(p[-1])
                instance.append(len(p)-(idx+300))
                mainDF.loc[len(mainDF.index)] = instance
    return math.ceil(initial_count/2)
```

THE ABOVE CODE CONVERTS THE TIME SERIES INTO 300 FEATURES AND 2 TARGETS

THIS RETURNS A DATASET WITH SAMPLES CORRESPONDING TO A SINGLE VOLCANO

SIMPLE FUNCTION FOR METRIC ANALYSIS

```
def metric(y_test,y_pred):
    mse = mean_squared_error(y_test.to_numpy(),y_pred,squared=False,multioutput= 'raw_values')
    variance_score = explained_variance_score(y_test.to_numpy(), y_pred,multioutput = 'raw_values')
    r2 = r2_score(y_test.to_numpy(),y_pred, multioutput ="raw_values")
    mae = mean_absolute_error(y_test.to_numpy(),y_pred, multioutput ="raw_values")
    return mse,variance_score,r2,mae
```

TRAINING THE LINEAR REGRESSOR MODEL USING PYTHON

TRAINING A MODEL FOR EVERY VOLCANO WITH 2 TARGETS Y1 AND Y2

Q1 outputs and Q2 outputs are both taken in this model
Data set Format
X1 X2 X3 . . . . X300 // Y1 Y2
where Xs are the consecutive sensor values forming a chain of length 300 sensor values per sample
Y1 is the sensor value at time = 0
Y2 is the time in seconds which is remaining until the Eruption

```python
# Q1 outputs and Q2 outputs are both taken in this model
# Data set Format
# X1 X2 X3 . . . . X300 // Y1 Y2
# where Xs are the consecutive sensor values forming a chain of length 300 sensor values per sample
# Y1 is the sensor value at time = 0
# Y2 is the time in seconds which is remaining until the Eruption
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold,cross_val_score
from IPython.display import display
import pandas as pd
modelPerVolcano = []
for f in range(1,10+1):
    print(f"Building Model for Volcano {f}")
    mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
# Actually building our dataset
    o = FeatureEngineering(f,mainDF)
# just shuffling the sample, not really required
    mainDF = mainDF.sample(frac=1)
    inputDF = mainDF.iloc[:,:-2]
    outputDF = mainDF.iloc[:,300:302]
    display(mainDF.head(5))
    x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
    kf = KFold(n_splits = 5)
# We can make the model better with different combinations of params, I had left it empty for faster model training
    params = {}
    model = LinearRegression()
    gv = GridSearchCV(model,param_grid=params,scoring = 'neg_mean_absolute_error',cv = kf)
    gv.fit(x_train,y_train)
    model = gv.best_estimator_
    y_pred = model.predict(x_test)
# evaluating a few metrics
    mse,variance_score,r2,mae = metric(y_test,y_pred)
    modelPerVolcano.append({"volcano": f,'model':model,'observationsConsidered' : o ,'neg_mean_absolute' :gv.best_score_
for m in modelPerVolcano:
    for key,value in m.items():
        print(f"{key} : {value}")
    print()
#saving models
import joblib
for i,m in enumerate(modelPerVolcano):
    model = m["model"]
    filename = "Q1Q2Models\\volcano"+str(m['volcano'])+".joblib"
    print(f"Saving Model {i}")
    joblib.dump(model, filename)
```

```
volcano : 1
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -97.57778057982007
mse : [1.94814466e-09 2.41239771e+02]
variance_score : [-0.03123205  0.30518383]
r2 : [-0.03335851  0.30409217]
mae : [1.75670110e-09 1.93904039e+02]

volcano : 2
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -122.3613398242079
mse : [1.65646222e-08 2.94893507e+02]
```

```
variance_score : [0.02208243 0.09224182]
r2 : [0.02190775 0.09111543]
mae : [1.30414009e-08 2.44953506e+02]

volcano : 3
model : LinearRegression()
observationsConsidered : 15
neg_mean_absolute : -109.48190136211397
mse : [1.91013411e-07 2.74222673e+02]
variance_score : [0.09854558 0.17476368]
r2 : [0.09832661 0.17456918]
mae : [1.75295522e-07 2.21066750e+02]

volcano : 4
model : LinearRegression()
observationsConsidered : 8
neg_mean_absolute : -126.69547319192353
mse : [1.77968090e-09 2.89567855e+02]
variance_score : [-0.01270171 -0.01879843]
r2 : [-0.01278633 -0.02246905]
mae : [1.56530128e-09 2.44116181e+02]

volcano : 5
model : LinearRegression()
observationsConsidered : 2
neg_mean_absolute : -131.1719472777568
mse : [  0.         299.01479396]
variance_score : [ 1.         -0.27760749]
r2 : [ 1.         -0.27831909]
mae : [  0.         245.13714396]

volcano : 6
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -103.06275144527224
mse : [1.56685145e-08 2.54802790e+02]
variance_score : [0.22492813 0.15519286]
r2 : [0.22491849 0.15518569]
mae : [1.26258082e-08 2.01940545e+02]

volcano : 7
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -131.1936882886096
mse : [8.27542064e-10 3.06893522e+02]
variance_score : [-0.11238432  0.04025424]
r2 : [-0.11290911  0.03821124]
mae : [7.81701636e-10 2.59889492e+02]

volcano : 8
model : LinearRegression()
observationsConsidered : 11
neg_mean_absolute : -122.94225051795902
mse : [1.11655615e-07 3.09627219e+02]
variance_score : [0.49734878 0.21298945]
r2 : [0.49714622 0.21287286]
mae : [9.46410480e-08 2.44541169e+02]

volcano : 9
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -124.77685061692526
mse : [3.94753926e-10 2.97057141e+02]
variance_score : [0.33155095 0.21787659]
```

r2 : [0.33131857 0.21662227]
mae : [3.32538428e-10 2.39427482e+02]

TRAINING A MODEL FOR EVERY VOLCANO WITH 1 TARGETS Y1

Training with Q1 outputs only
given a time series predict the magnitude at 0 seconds
Data set Format
X1 X2 X3 . . . . X300 // Y1
where Xs are the consecutive sensor values forming a chain of length 300 sensor values per sample
Y1 is the sensor value at time = 0

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold,cross_val_score
from IPython.display import display
import pandas as pd
modelPerVolcano = []
for f in range(1,10+1):
    print(f"Building Model for Volcano {f}")
    mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
# Actually building our dataset
    o = FeatureEngineering(f,mainDF)
# just shuffling the sample, not really required
    mainDF = mainDF.sample(frac=1)
    inputDF = mainDF.iloc[:,:-2]
    outputDF = mainDF.iloc[:,300:301]
    display(mainDF.head())
    x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
    kf = KFold(n_splits = 5)
# We can make the model better with different combinations of params, I had left it empty for faster model training
    params = {}
    model = LinearRegression()
    gv = GridSearchCV(model,param_grid=params,scoring = 'neg_mean_absolute_error',cv = kf)
    gv.fit(x_train,y_train)
    model = gv.best_estimator_
    y_pred = model.predict(x_test)
# evaluating a few metrics
    mse,variance_score,r2,mae = metric(y_test,y_pred)
    modelPerVolcano.append({"volcano": f,'model':model,'observationsConsidered' : o ,'neg_mean_absolute' :gv.best_score_
for m in modelPerVolcano:
    for key,value in m.items():
        print(f"{key} : {value}")
    print()
#saving models
import joblib
for i,m in enumerate(modelPerVolcano):
    model = m["model"]
    filename = "Q1Models\\volcano"+str(m['volcano'])+"Q"+str(1)+".joblib"
    print(f"Saving Model {i+1} For Q1")
    joblib.dump(model, filename)
```

volcano : 1
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -1.8028479870665053e-09
mse : [1.94245444e-09]
variance_score : [-0.01232022]
r2 : [-0.01234496]
mae : [1.75011137e-09]

volcano : 2
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -1.3071602514930183e-08
mse : [1.64608213e-08]
variance_score : [0.05216863]
r2 : [0.05213251]
mae : [1.298671e-08]

volcano : 3

```
model : LinearRegression()
observationsConsidered : 15
neg_mean_absolute : -1.7513592680965734e-07
mse : [1.89849988e-07]
variance_score : [0.10932453]
r2 : [0.10918578]
mae : [1.73319429e-07]

volcano : 4
model : LinearRegression()
observationsConsidered : 8
neg_mean_absolute : -1.583499832939313e-09
mse : [1.83162029e-09]
variance_score : [-0.01034224]
r2 : [-0.0112746]
mae : [1.6140701e-09]

volcano : 5
model : LinearRegression()
observationsConsidered : 2
neg_mean_absolute : -1.2407709188295415e-25
mse : [0.]
variance_score : [1.]
r2 : [1.]
mae : [0.]

volcano : 6
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -1.2813019692186025e-08
mse : [1.55221247e-08]
variance_score : [0.23431414]
r2 : [0.23258764]
mae : [1.26316773e-08]

volcano : 7
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -7.87277572674364e-10
mse : [8.38164832e-10]
variance_score : [-0.09451914]
r2 : [-0.09703457]
mae : [7.90873505e-10]

volcano : 8
model : LinearRegression()
observationsConsidered : 11
neg_mean_absolute : -9.535096813979395e-08
mse : [1.12595105e-07]
variance_score : [0.49269383]
r2 : [0.49269058]
mae : [9.51329867e-08]

volcano : 9
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -3.359623036601528e-10
mse : [4.03828146e-10]
variance_score : [0.29691795]
r2 : [0.29691761]
mae : [3.4255245e-10]
```

TRAINING A MODEL FOR EVERY VOLCANO WITH 1 TARGETS Y2

Training with Q2 outputs only
given a time series predict the time remaining for eruption
Data set Format
X1 X2 X3 . . . . X300 // Y2
where Xs are the consecutive sensor values forming a chain of length 300 sensor values per sample
Y2 is the time in seconds which is remaining until the Eruption

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold,cross_val_score
from IPython.display import display
import pandas as pd
modelPerVolcano = []
for f in range(1,10+1):
    print(f"Building Model for Volcano {f}")
    mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
# Actually building our dataset
    o = FeatureEngineering(f,mainDF)
# just shuffling the sample, not really required
    mainDF = mainDF.sample(frac=1)
    inputDF = mainDF.iloc[:,:-2]
    outputDF = mainDF.iloc[:,301:302]
    display(mainDF.head())
    x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
    kf = KFold(n_splits = 5)
# We can make the model better with different combinations of params, I had left it empty for faster model training
    params = {}
    model = LinearRegression()
    gv = GridSearchCV(model,param_grid=params,scoring = 'neg_mean_absolute_error',cv = kf)
    gv.fit(x_train,y_train)
    model = gv.best_estimator_
    y_pred = model.predict(x_test)
# evaluating a few metrics
    mse,variance_score,r2,mae = metric(y_test,y_pred)
    modelPerVolcano.append({"volcano": f,'model':model,'observationsConsidered' : o ,'neg_mean_absolute' :gv.best_score_
for m in modelPerVolcano:
    for key,value in m.items():
        print(f"{key} : {value}")
    print()
#saving models
import joblib
for i,m in enumerate(modelPerVolcano):
    model = m["model"]
    filename = "Q2Models\\volcano"+str(m['volcano'])+"Q"+str(2)+".joblib"
    print(f"Saving Model {i+1} For Q2")
    joblib.dump(model, filename)
```

```
volcano : 1
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -192.13152926381218
mse : [245.29704525]
variance_score : [0.2775991]
r2 : [0.27684691]
mae : [194.51415443]

volcano : 2
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -243.97487348983728
mse : [296.72994315]
variance_score : [0.11189373]
r2 : [0.11171043]
mae : [245.73568167]

volcano : 3
```

```
model : LinearRegression()
observationsConsidered : 15
neg_mean_absolute : -219.83992863526638
mse : [272.50377888]
variance_score : [0.17144118]
r2 : [0.1713098]
mae : [218.87593194]

volcano : 4
model : LinearRegression()
observationsConsidered : 8
neg_mean_absolute : -251.2468354658128
mse : [300.69253749]
variance_score : [-0.07836757]
r2 : [-0.07915243]
mae : [252.13749916]

volcano : 5
model : LinearRegression()
observationsConsidered : 2
neg_mean_absolute : -268.5240725210555
mse : [281.32834511]
variance_score : [-0.19920282]
r2 : [-0.2077473]
mae : [227.66155176]

volcano : 6
model : LinearRegression()
observationsConsidered : 14
neg_mean_absolute : -206.25666672037778
mse : [256.13651777]
variance_score : [0.14015635]
r2 : [0.13751768]
mae : [204.51730837]

volcano : 7
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -262.720835407909
mse : [305.27324033]
variance_score : [0.02341456]
r2 : [0.02100751]
mae : [257.43734246]

volcano : 8
model : LinearRegression()
observationsConsidered : 11
neg_mean_absolute : -247.18729343527562
mse : [302.62960384]
variance_score : [0.20145598]
r2 : [0.20134074]
mae : [239.34060316]

volcano : 9
model : LinearRegression()
observationsConsidered : 6
neg_mean_absolute : -249.80174768663977
mse : [297.88335983]
variance_score : [0.24708372]
r2 : [0.24681474]
mae : [244.54250077]
```

LOAD THE SAVED MODELS SO WE DON'T HAVE TO TRAIN THEM AGAIN
WE CAN JUST CHANGE TO THE VALUES INSIDE LOAD() FUNCTION TO LOAD A MODEL FOR A DIFFERENT VOLCANO

LOAD MODEL FOR Q1 AND Q2 PREDICTIONS

```
#Load Q1Q2 Models
#change Volcano folders to get predictions of different volcano
model1 = joblib.load('Q1Q2Models\\volcano'+str(1)+".joblib")
mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
FeatureEngineering(1,mainDF)
mainDF = mainDF.sample(frac=1)
inputDF = mainDF.iloc[:,:-2]
outputDF = mainDF.iloc[:,300:302]
x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
y_pred = model1.predict(x_test)
for idx,x in enumerate(y_test.to_numpy()):
    print(f"Q1 {x[0]} : {y_pred[idx][0]}  | Q2 {x[1]} : {y_pred[idx][1]}")

```

LOAD MODEL FOR Q1 PREDICTIONS

```
### Load Q1 Models
#change Volcano folders to get predictions of different volcano
model1 = joblib.load('Q1Models\\volcano'+str(1)+"Q"+str(1)+".joblib")
mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
FeatureEngineering(1,mainDF)
mainDF = mainDF.sample(frac=1)
inputDF = mainDF.iloc[:,:-2]
outputDF = mainDF.iloc[:,300:302]
x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
y_pred = model1.predict(x_test)
for idx,x in enumerate(y_test.to_numpy()):
    print(f"{x[0]} : {y_pred[idx][0]}")
```

LOAD MODEL FOR Q1 AND Q2 PREDICTIONS

```
#Load Q2 Models
#change Volcano folders to get predictions of different volcano
model1 = joblib.load('Q2Models\\volcano'+str(1)+"Q"+str(2)+".joblib")
mainDF = pd.DataFrame(columns=[x for x in range(0,302)])
FeatureEngineering(1,mainDF)
mainDF = mainDF.sample(frac=1)
inputDF = mainDF.iloc[:,:-2]
outputDF = mainDF.iloc[:,301:302]
x_train, x_test, y_train, y_test = train_test_split(inputDF,outputDF, test_size=.2, random_state = 0)
y_pred = model1.predict(x_test)
for idx,x in enumerate(y_test.to_numpy()):
    print(f"{x[0]} : {y_pred[idx][0]}")
```