In [1]:

```
import pandas as pd
import numpy as np
from functools import reduce
from missingpy import MissForest
import sklearn
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
import random
from sklearn.multioutput import MultiOutputRegressor
from sklearn.ensemble import GradientBoostingRegressor
import numpy as np
import matplotlib.pyplot as plt
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.n
eighbors.base module is  deprecated in version 0.22 and will be removed in version 0.24. The correspo
nding classes / functions should instead be imported from sklearn.neighbors. Anything that cannot be
imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
```

In [2]:

```
traindata=pd.read_csv('/Users/swaggy/Downloads/ContestSubmissions/train.csv')
testdata=pd.read_csv('/Users/swaggy/Downloads/ContestSubmissions/test.csv')
```

Adding next , previous data for all variables

In [3]:

```
trainprev=traindata.shift(1)
trainprev.columns=['ID', 'Time', 'prevLatitude', 'prevLongitude', 'prevAltB', 'prevGndSpd', 'prevVSpd']
trainprev=trainprev.drop(['ID', 'Time'], axis=1)
trainnext=traindata.shift(-1)
trainnext.columns=['ID', 'Time', 'nextLatitude', 'nextLongitude', 'nextAltB', 'nextGndSpd', 'nextVSpd']
trainnext=trainnext.drop(['ID', 'Time'], axis=1)
fulltraindata=pd.concat([traindata,trainprev,trainnext],axis=1)
# Observation distances
TimeOffsets=traindata.shift(1)
TimeOffsets['DistFromLastObs']=TimeOffsets['Time']%16
TimeOffsets['DistFromNextObs']=16-TimeOffsets['Time']%16
TimeOffsets['DistFromNextObs'] = np.where((TimeOffsets.DistFromNextObs == 16),0,TimeOffsets.DistFromNextO
TimeOffsets['DistFromLastObs']=TimeOffsets['DistFromLastObs'].fillna(0)
TimeOffsets['DistFromNextObs']=TimeOffsets['DistFromNextObs'].fillna(0)
TimeOffsets['Time']=TimeOffsets['Time']+1
TimeOffsets=TimeOffsets[['ID','Time','DistFromLastObs','DistFromNextObs']]

fulltraindata=pd.merge(fulltraindata,TimeOffsets, on=['ID','Time'], how='left')
fulltraindata['DistFromLastObs']=fulltraindata['DistFromLastObs'].fillna(0)
fulltraindata['DistFromNextObs']=fulltraindata['DistFromNextObs'].fillna(0)
fulltraindata['prevLatitude']=fulltraindata['prevLatitude'].fillna(0)
fulltraindata['prevLongitude']=fulltraindata['prevLongitude'].fillna(0)
fulltraindata['prevAltB']=fulltraindata['prevAltB'].fillna(0)
fulltraindata['prevGndSpd']=fulltraindata['prevGndSpd'].fillna(0)
fulltraindata['prevVSpd']=fulltraindata['prevVSpd'].fillna(0)
fulltraindata['nextLatitude']=fulltraindata['nextLatitude'].fillna(0)
fulltraindata['nextLongitude']=fulltraindata['nextLongitude'].fillna(0)
fulltraindata['nextAltB']=fulltraindata['nextAltB'].fillna(0)
fulltraindata['nextGndSpd']=fulltraindata['nextGndSpd'].fillna(0)
fulltraindata['nextVSpd']=fulltraindata['nextVSpd'].fillna(0)

# prev and curr differences
#prev diff
fulltraindata['DiffPrevLati']=fulltraindata['Latitude']-fulltraindata['prevLatitude']
fulltraindata['DiffPrevLongi']=fulltraindata['Longitude']-fulltraindata['prevLongitude']
fulltraindata['DiffPrevAltB']=fulltraindata['AltB']-fulltraindata['prevAltB']
fulltraindata['DiffPrevGndSpd']=fulltraindata['GndSpd']-fulltraindata['prevGndSpd']
fulltraindata['DiffPrevVSpd']=fulltraindata['VSpd']-fulltraindata['prevVSpd']


#prev next
fulltraindata['DiffNextLati']=fulltraindata['Latitude']-fulltraindata['nextLatitude']
```

```
fulltraindata['DiffNextLongi']=fulltraindata['Longitude']-fulltraindata['nextLongitude']
fulltraindata['DiffNextAltB']=fulltraindata['AltB']-fulltraindata['nextAltB']
fulltraindata['DiffNextGndSpd']=fulltraindata['GndSpd']-fulltraindata['nextGndSpd']
fulltraindata['DiffNextVSpd']=fulltraindata['VSpd']-fulltraindata['nextVSpd']

#Differences made 0 where null
fulltraindata['DiffPrevLati']=fulltraindata['DiffPrevLati'].fillna(0)
fulltraindata['DiffPrevLongi']=fulltraindata['DiffPrevLongi'].fillna(0)
fulltraindata['DiffPrevAltB']=fulltraindata['DiffPrevAltB'].fillna(0)
fulltraindata['DiffPrevGndSpd']=fulltraindata['DiffPrevGndSpd'].fillna(0)
fulltraindata['DiffPrevVSpd']=fulltraindata['DiffPrevVSpd'].fillna(0)

fulltraindata['DiffNextLati']=fulltraindata['DiffNextLati'].fillna(0)
fulltraindata['DiffNextLongi']=fulltraindata['DiffNextLongi'].fillna(0)
fulltraindata['DiffPrevAltB']=fulltraindata['DiffPrevAltB'].fillna(0)
fulltraindata['DiffNextGndSpd']=fulltraindata['DiffNextGndSpd'].fillna(0)
fulltraindata['DiffNextVSpd']=fulltraindata['DiffNextVSpd'].fillna(0)
```

Introduce missing variables in traindata to interpolate

Introduce missing data every 15 secs

In [4]:

```
imptraindata=traindata.copy()
conditions = [
    (imptraindata['Time'] == 1)|(fulltraindata['Time'] % 16-1 == 0)
    ]
#choices = [imptraindata['Time']]

imptraindata['Latitude'] = np.select(conditions, [imptraindata['Latitude']], default=np.NaN)
imptraindata['Longitude'] = np.select(conditions, [imptraindata['Longitude']], default=np.NaN)
imptraindata['AltB'] = np.select(conditions, [imptraindata['AltB']], default=np.NaN)
imptraindata['GndSpd'] = np.select(conditions, [imptraindata['GndSpd']], default=np.NaN)
imptraindata['VSpd'] = np.select(conditions, [imptraindata['VSpd']], default=np.NaN)

#interpolate training data
imptraindata=imptraindata.interpolate(method='linear')
#imptraindata=imptraindata.drop(['ID', 'Time'], axis=1)
imptraindata.columns=['ID', 'Time','impLatitude','impLongitude','impAltB','impGndSpd','impVSpd']
```

Merge imputed data to full training set

In [5]:

```
fulltraindata=pd.merge(fulltraindata,imptraindata,on=['ID','Time'])
```

Prepare testdata for model prediction

In [6]:

```
testprev=testdata.shift(1)
testprev.columns=['ID', 'Time', 'prevLatitude', 'prevLongitude', 'prevAltB', 'prevGndSpd', 'prevVSpd']
testprev=testprev.drop(['ID', 'Time'], axis=1)
testnext=testdata.shift(-1)
testnext.columns=['ID', 'Time', 'nextLatitude', 'nextLongitude', 'nextAltB', 'nextGndSpd', 'nextVSpd']
testnext=testnext.drop(['ID', 'Time'], axis=1)
#Interpolation test data
imptestdata=testdata.interpolate(method='linear')
imptestdata.columns=['ID', 'Time','impLatitude','impLongitude','impAltB','impGndSpd','impVSpd']
imptestdata=imptestdata.drop(['ID', 'Time'], axis=1)
fulltestdata=pd.concat([testdata,testprev,testnext,imptestdata],axis=1)

# Observation distances
TimeOffsets=testdata.shift(1)
TimeOffsets['DistFromLastObs']=TimeOffsets['Time']%16
TimeOffsets['DistFromNextObs']=16-TimeOffsets['Time']%16
TimeOffsets['DistFromNextObs'] = np.where((TimeOffsets.DistFromNextObs == 16),0,TimeOffsets.DistFromNext(
TimeOffsets['DistFromLastObs']=TimeOffsets['DistFromLastObs'].fillna(0)
TimeOffsets['DistFromNextObs']=TimeOffsets['DistFromNextObs'].fillna(0)
TimeOffsets['Time']=TimeOffsets['Time']+1
TimeOffsets=TimeOffsets[['ID','Time','DistFromLastObs','DistFromNextObs']]

fulltestdata=pd.merge(fulltestdata,TimeOffsets, on=['ID','Time'], how='left')
fulltestdata['DistFromLastObs']=fulltestdata['DistFromLastObs'].fillna(0)
fulltestdata['DistFromNextObs']=fulltestdata['DistFromNextObs'].fillna(0)

#prev diff
fulltestdata['DiffPrevLati']=fulltestdata['Latitude']-fulltestdata['prevLatitude']
fulltestdata['DiffPrevLongi']=fulltestdata['Longitude']-fulltestdata['prevLongitude']
fulltestdata['DiffPrevAltB']=fulltestdata['AltB']-fulltestdata['prevAltB']
fulltestdata['DiffPrevGndSpd']=fulltestdata['GndSpd']-fulltestdata['prevGndSpd']
fulltestdata['DiffPrevVSpd']=fulltestdata['VSpd']-fulltestdata['prevVSpd']


#prev next
fulltestdata['DiffNextLati']=fulltestdata['Latitude']-fulltestdata['nextLatitude']
fulltestdata['DiffNextLongi']=fulltestdata['Longitude']-fulltestdata['nextLongitude']
fulltestdata['DiffNextAltB']=fulltestdata['AltB']-fulltestdata['nextAltB']
fulltestdata['DiffNextGndSpd']=fulltestdata['GndSpd']-fulltestdata['nextGndSpd']
fulltestdata['DiffNextVSpd']=fulltestdata['VSpd']-fulltestdata['nextVSpd']
```

```
# Fill NAs with null
fulltestdata['DistFromLastObs']=fulltestdata['DistFromLastObs'].fillna(0)
fulltestdata['DistFromNextObs']=fulltestdata['DistFromNextObs'].fillna(0)
fulltestdata['prevLatitude']=fulltestdata['prevLatitude'].fillna(0)
fulltestdata['prevLongitude']=fulltestdata['prevLongitude'].fillna(0)
fulltestdata['prevAltB']=fulltestdata['prevAltB'].fillna(0)
fulltestdata['prevGndSpd']=fulltestdata['prevGndSpd'].fillna(0)
fulltestdata['prevVSpd']=fulltestdata['prevVSpd'].fillna(0)
fulltestdata['nextLatitude']=fulltestdata['nextLatitude'].fillna(0)
fulltestdata['nextLongitude']=fulltestdata['nextLongitude'].fillna(0)
fulltestdata['nextAltB']=fulltestdata['nextAltB'].fillna(0)
fulltestdata['nextGndSpd']=fulltestdata['nextGndSpd'].fillna(0)
fulltestdata['nextVSpd']=fulltestdata['nextVSpd'].fillna(0)


fulltestdata['DiffPrevLati']=fulltestdata['DiffPrevLati'].fillna(0)
fulltestdata['DiffPrevLongi']=fulltestdata['DiffPrevLongi'].fillna(0)
fulltestdata['DiffPrevAltB']=fulltestdata['DiffPrevAltB'].fillna(0)
fulltestdata['DiffPrevGndSpd']=fulltestdata['DiffPrevGndSpd'].fillna(0)
fulltestdata['DiffPrevVSpd']=fulltestdata['DiffPrevVSpd'].fillna(0)

fulltestdata['DiffNextLati']=fulltestdata['DiffNextLati'].fillna(0)
fulltestdata['DiffNextLongi']=fulltestdata['DiffNextLongi'].fillna(0)
fulltestdata['DiffNextAltB']=fulltestdata['DiffNextAltB'].fillna(0)
fulltestdata['DiffNextGndSpd']=fulltestdata['DiffNextGndSpd'].fillna(0)
fulltestdata['DiffNextVSpd']=fulltestdata['DiffNextVSpd'].fillna(0)
```

In [7]:

```
y_train=fulltraindata[['Latitude','Longitude','AltB','GndSpd','VSpd']]
```

In [8]:

```
X_train=fulltraindata[['Time',
                       'prevLatitude','prevLongitude','prevAltB','prevGndSpd','prevVSpd',
                      'nextLatitude','nextLongitude','nextAltB','nextGndSpd','nextVSpd',
                       'DistFromLastObs','DistFromNextObs',
                       'DiffPrevLati','DiffPrevLongi','DiffPrevAltB','DiffPrevGndSpd','DiffPrevVSpd',
                       'DiffNextLati','DiffNextLongi','DiffPrevAltB','DiffNextGndSpd','DiffNextVSpd',
                       'impLatitude', 'impLongitude', 'impAltB', 'impGndSpd', 'impVSpd'
                      ]]

#X_train=np.ravel(X_train)
```

In [9]:

```
# Scaling data for better ANN performance
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X_train = sc_X.fit_transform(X_train)
y_train = sc_y.fit_transform(y_train)
X_train_cpy=X_train.copy()
```

In [10]:

```
X_test=fulltestdata[['Time',
                       'prevLatitude','prevLongitude','prevAltB','prevGndSpd','prevVSpd',
                      'nextLatitude','nextLongitude','nextAltB','nextGndSpd','nextVSpd',
                       'DistFromLastObs','DistFromNextObs',
                       'DiffPrevLati','DiffPrevLongi','DiffPrevAltB','DiffPrevGndSpd','DiffPrevVSpd',
                       'DiffNextLati','DiffNextLongi','DiffNextAltB','DiffNextGndSpd','DiffNextVSpd',
                       'impLatitude', 'impLongitude', 'impAltB', 'impGndSpd', 'impVSpd'
                    ]]
X_testcpy=X_test.copy()
X_test = sc_X.fit_transform(X_test)
```

MLP Regressor

In [11]:

```
random.seed(200)
from sklearn.neural_network import MLPRegressor

mlp = MLPRegressor(hidden_layer_sizes=(8,8,8), activation='relu', solver='adam', max_iter=500)
mlp.fit(X_train,y_train)
```

Out[11]:

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(8, 8, 8), learning_rate='constant',
             learning_rate_init=0.001, max_fun=15000, max_iter=500,
             momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
             power_t=0.5, random_state=None, shuffle=True, solver='adam',
             tol=0.0001, validation_fraction=0.1, verbose=False,
             warm_start=False)
```

```
In [12]: mlppred = mlp.predict(X_test)

         # Rescaling
         mlppred=sc_y.inverse_transform(mlppred)
         mlppred=pd.DataFrame(mlppred)
         mlppred.columns=['Latitude','Longitude','AltB','GndSpd','VSpd']
         mlppred=pd.concat([testdata[['ID','Time']],mlppred],axis=1)
         mlppred
```

```
Out[12]:
             ID  Time   Latitude   Longitude         AltB      GndSpd         VSpd
       0     31    57  41.615178  -95.516682  2488.325141  104.060882    41.392749
       1     31    58  42.041837  -97.810908  1922.043641   88.278243  -175.668192
       2     31    59  41.631526  -95.515271  2489.491788  104.699739    38.139944
       3     31    60  41.636602  -95.517574  2486.570818  104.964237    38.837281
       4     31    61  41.641678  -95.519878  2483.649847  105.228736    39.534619
     ...    ...   ...        ...         ...          ...         ...          ...
  208283    110   209  41.832504  -95.602966  2731.698593  113.576388   -15.580593
  208284    110   210  41.836685  -95.597991  2737.301238  113.459048   -14.349995
  208285    110   211  41.835329  -95.597942  2735.885879  113.406498   -13.726727
  208286    110   212  42.137431  -94.158298  6747.158500  178.986532 -1920.706133
  208287    110   213  41.832617  -95.597844  2733.055161  113.301397   -12.480192

208288 rows × 7 columns
```

```
In [13]: parameter_space = {
             'hidden_layer_sizes': [(8,8,8), (8,8,8), (500,)],
             'activation': ['relu'],
             'solver': [ 'adam'],
             'alpha': [0.0001, 0.05],
             'learning_rate': ['constant','adaptive'],
         }
```

```
In [14]: from sklearn.model_selection import GridSearchCV

         clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=3)
         clf.fit(X_train, y_train)
```

```
Out[14]: GridSearchCV(cv=3, error_score=nan,
                      estimator=MLPRegressor(activation='relu', alpha=0.0001,
                                             batch_size='auto', beta_1=0.9, beta_2=0.999,
                                             early_stopping=False, epsilon=1e-08,
                                             hidden_layer_sizes=(8, 8, 8),
                                             learning_rate='constant',
                                             learning_rate_init=0.001, max_fun=15000,
                                             max_iter=500, momentum=0.9,
                                             n_iter_no_change=10,
                                             nesterovs_momentum=True, power_t=0.5,
                                             random_state...
                                             solver='adam', tol=0.0001,
                                             validation_fraction=0.1, verbose=False,
                                             warm_start=False),
                      iid='deprecated', n_jobs=-1,
                      param_grid={'activation': ['relu'], 'alpha': [0.0001, 0.05],
                                  'hidden_layer_sizes': [(8, 8, 8), (8, 8, 8), (500,)],
                                  'learning_rate': ['constant', 'adaptive'],
                                  'solver': ['adam']},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

Plot of predicted vs imputed values

In [15]:

```python
plotdata=fulltestdata[['ID','Time','impLatitude','impLongitude','impAltB','impGndSpd','impVSpd']]
plotdata=plotdata[plotdata['ID']==35]

preddata=mlppred[['ID','Time','Latitude','Longitude','AltB','GndSpd','VSpd']]
preddata=preddata[preddata['ID']==35]
preddata=preddata.drop(['ID', 'Time'], axis=1)
plotdata=pd.concat([plotdata,preddata],axis=1)
```

In [16]:

```python
plotdata
```

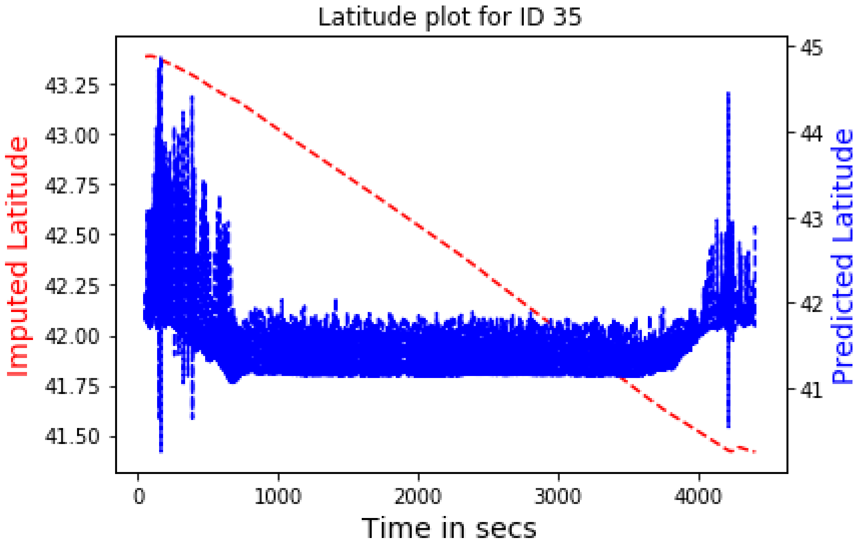
Out[16]:

	ID	Time	impLatitude	impLongitude	impAltB	impGndSpd	impVSpd	Latitude	Longitude	AltB	GndSpd	V
9380	35	53	43.388554	-95.139824	1445.0000	26.670000	-12.820000	41.805622	-98.114373	-1195.292220	15.973965	774.18
9381	35	54	43.388672	-95.139859	1445.1875	26.519375	-11.661875	42.135099	-97.727701	1904.841036	90.949964	-133.46
9382	35	55	43.388791	-95.139894	1445.3750	26.368750	-10.503750	41.767968	-95.560289	2589.657104	104.094029	35.11
9383	35	56	43.388909	-95.139930	1445.5625	26.218125	-9.345625	41.767680	-95.560236	2589.907794	104.079398	35.07
9384	35	57	43.389028	-95.139965	1445.7500	26.067500	-8.187500	41.767391	-95.560182	2590.158484	104.064768	35.03
...	...	...	...	...	...	...	...	...	...	...	...	
13728	35	4401	41.417159	-96.111849	1328.5500	38.805000	-87.190000	41.730310	-95.569580	2592.232134	105.103977	24.97
13729	35	4402	41.417013	-96.111669	1326.8625	37.501250	-60.640000	41.733390	-95.564147	2599.787585	104.827586	27.33
13730	35	4403	41.416868	-96.111489	1325.1750	36.197500	-34.090000	41.731106	-95.563021	2600.565284	104.654002	27.45
13731	35	4404	41.416722	-96.111310	1323.4875	34.893750	-7.540000	42.882213	-94.802210	5330.524756	128.324479	24.59
13732	35	4405	41.416576	-96.111130	1321.8000	33.590000	19.010000	42.848568	-94.816477	5517.976758	129.812284	-19.68

4353 rows × 12 columns

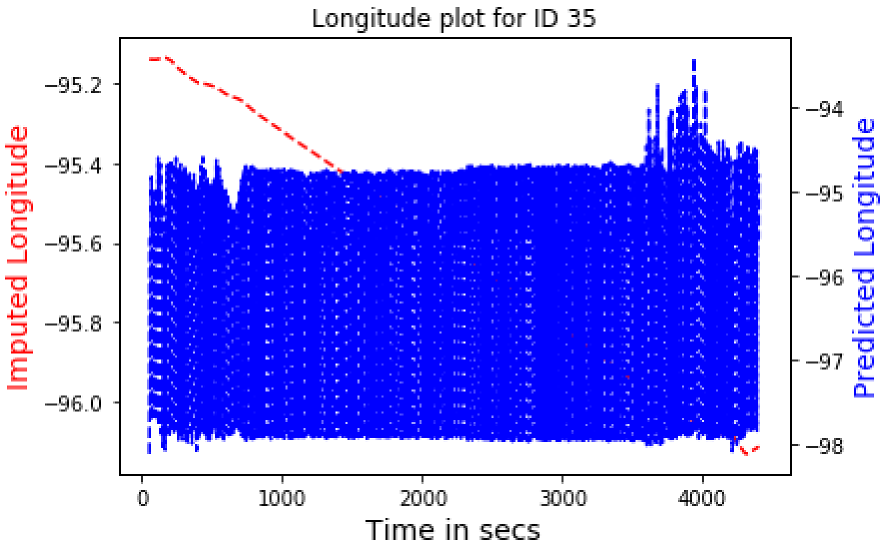
In [17]:

```python
fig,ax = plt.subplots()
ax.plot(plotdata.Time, plotdata.impLatitude, color="red",linestyle='--')
# set x-axis label
ax.set_xlabel("Time in secs",fontsize=14)
# set y-axis label
ax.set_ylabel("Imputed Latitude",color="red",fontsize=14)
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(plotdata.Time, plotdata.Latitude,color="blue",linestyle='--')
ax2.set_ylabel("Predicted Latitude",color="blue",fontsize=14)
ax.set_title('Latitude plot for ID 35')
plt.show()
```



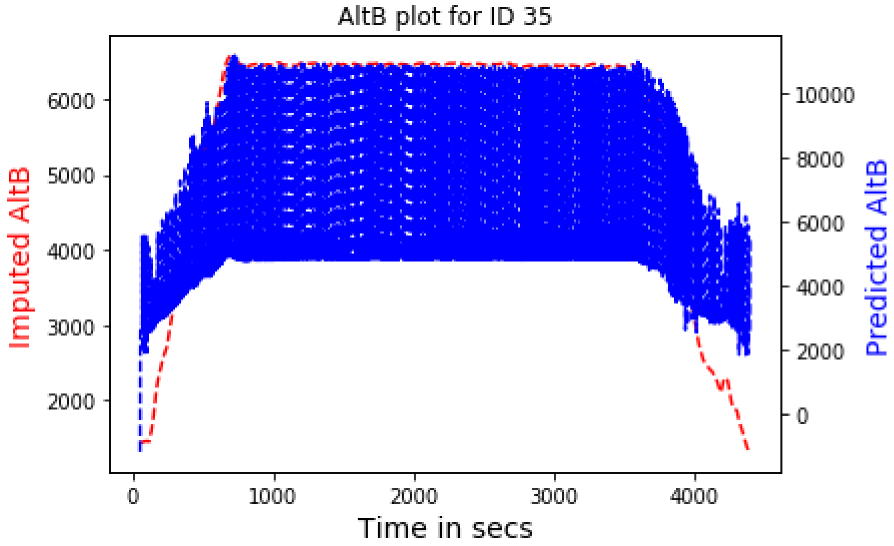
In [18]:

```
fig,ax = plt.subplots()
ax.plot(plotdata.Time, plotdata.impLongitude, color="red",linestyle='--')
# set x-axis label
ax.set_xlabel("Time in secs",fontsize=14)
# set y-axis label
ax.set_ylabel("Imputed Longitude",color="red",fontsize=14)
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(plotdata.Time, plotdata.Longitude,color="blue",linestyle='--')
ax2.set_ylabel("Predicted Longitude",color="blue",fontsize=14)
ax.set_title('Longitude plot for ID 35')
plt.show()
```



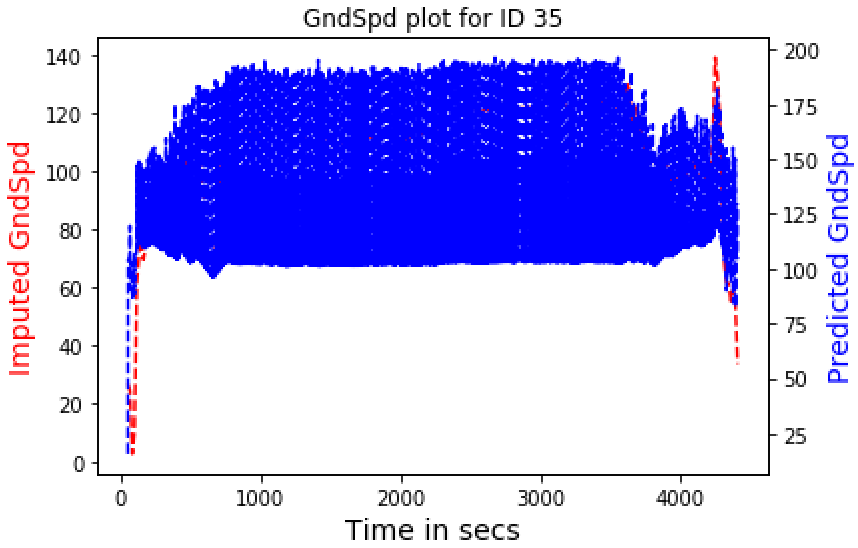
In [19]:

```
fig,ax = plt.subplots()
ax.plot(plotdata.Time, plotdata.impAltB, color="red",linestyle='--')
# set x-axis label
ax.set_xlabel("Time in secs",fontsize=14)
# set y-axis label
ax.set_ylabel("Imputed AltB",color="red",fontsize=14)
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(plotdata.Time, plotdata.AltB,color="blue",linestyle='--')
ax2.set_ylabel("Predicted AltB",color="blue",fontsize=14)
ax.set_title('AltB plot for ID 35')
plt.show()
```



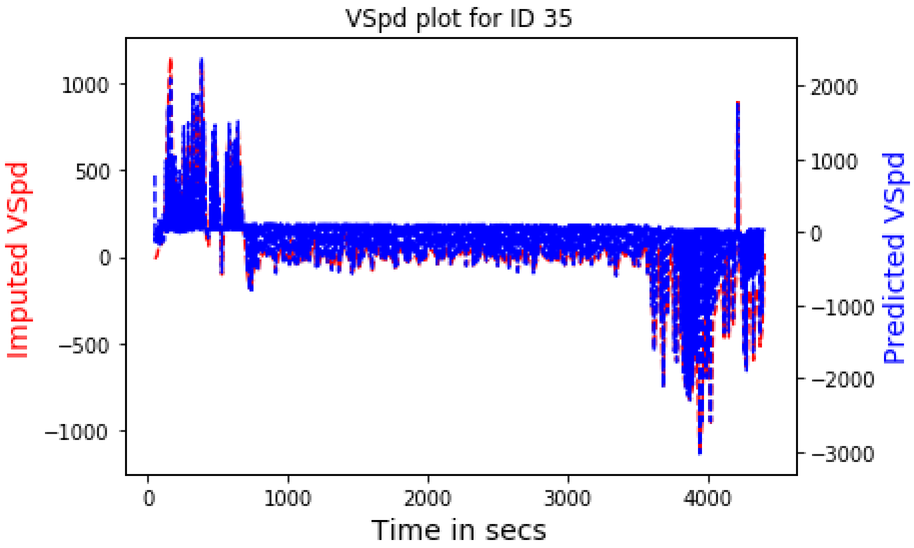
In [20]:

```
fig,ax = plt.subplots()
ax.plot(plotdata.Time, plotdata.impGndSpd, color="red",linestyle='--')
# set x-axis label
ax.set_xlabel("Time in secs",fontsize=14)
# set y-axis label
ax.set_ylabel("Imputed GndSpd",color="red",fontsize=14)
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(plotdata.Time, plotdata.GndSpd,color="blue",linestyle='--')
ax2.set_ylabel("Predicted GndSpd",color="blue",fontsize=14)
ax.set_title('GndSpd plot for ID 35')
plt.show()
```



In [21]:

```
fig,ax = plt.subplots()
ax.plot(plotdata.Time, plotdata.impVSpd, color="red",linestyle='--')
# set x-axis label
ax.set_xlabel("Time in secs",fontsize=14)
# set y-axis label
ax.set_ylabel("Imputed VSpd",color="red",fontsize=14)
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(plotdata.Time, plotdata.VSpd,color="blue",linestyle='--')
ax2.set_ylabel("Predicted VSpd",color="blue",fontsize=14)
ax.set_title('VSpd plot for ID 35')
plt.show()
```



Gradient Boost Regressor

In [22]:

```
gbr=MultiOutputRegressor(GradientBoostingRegressor(random_state=0)).fit(X_train, y_train)
```

In [23]:

```
gbrpred = gbr.predict(X_test)
gbrpred=sc_y.inverse_transform(gbrpred)
gbrpred=pd.DataFrame(gbrpred)
gbrpred.columns=['Latitude','Longitude','AltB','GndSpd','VSpd']
gbrpred=pd.concat([testdata[['ID','Time']],gbrpred],axis=1)
gbrpred
```

Out[23]:

	ID	Time	Latitude	Longitude	AltB	GndSpd	VSpd
0	31	57	41.430434	-95.553079	3279.288140	108.771589	-4.011854
1	31	58	42.170391	-96.673713	3671.877163	131.004618	-131.066154
2	31	59	41.430434	-95.553079	3279.288140	108.771589	-4.011854
3	31	60	41.430434	-95.553079	3279.288140	108.771589	-4.011854
4	31	61	41.430434	-95.553079	3279.288140	108.771589	-4.011854
...	...	...	...	...	...	...	...
208283	110	209	41.479647	-95.500282	3365.769370	109.647235	-4.011854
208284	110	210	41.479647	-95.500282	3365.769370	109.647235	-4.011854
208285	110	211	41.479647	-95.500282	3365.769370	109.647235	-4.011854
208286	110	212	42.309541	-95.835209	4166.066326	137.564198	-528.086236
208287	110	213	41.479647	-95.500282	3350.774314	109.647235	-4.011854

208288 rows × 7 columns

ANN using difference of imputed and actual as response variable

In this approach, the response variable was chosen to be the difference between the actual and imputed variables from the train data, was predicted using the variables mentioned above.

After prediction, the predicted differences were added to the imputed variables to get the predicted actual values.

This approach was motivated by the research published at :-
https://www.researchgate.net/publication/42428947_Filling_gaps_in_wave_records_with_artificial_neural_networks/link/56d56b9808aefd177
(https://www.researchgate.net/publication/42428947_Filling_gaps_in_wave_records_with_artificial_neural_networks/link/56d56b9808aefd177

In [24]:

```
# Generating differences between actual and imputed values of training data - this will be used as a res

fulltraindata['DiffImpLati'] = fulltraindata['Latitude'] - fulltraindata['impLatitude']
fulltraindata['DiffImpLongi'] = fulltraindata['Longitude'] - fulltraindata['impLongitude']
fulltraindata['DiffImpAltB'] =   fulltraindata['AltB'] - fulltraindata['impAltB']
fulltraindata['DiffImpGndSpd'] = fulltraindata['GndSpd'] - fulltraindata['impGndSpd']
fulltraindata['DiffImpVSpd'] =   fulltraindata['VSpd']  - fulltraindata['impVSpd']
y_train=fulltraindata[['DiffImpLati','DiffImpLongi','DiffImpAltB','DiffImpGndSpd','DiffImpVSpd']]
y_train = sc_y.fit_transform(y_train)
```

In [25]:

```
random.seed(200)

mlp = MLPRegressor(hidden_layer_sizes=(8,8,8), activation='relu', solver='adam', max_iter=500)
mlp.fit(X_train_cpy,y_train)
mlpdiff = mlp.predict(X_test)

# Rescaling
mlpdiff=sc_y.inverse_transform(mlpdiff)
mlpdiff=pd.DataFrame(mlpdiff)
mlpdiff.columns=['DiffImpLati','DiffImpLongi','DiffImpAltB','DiffImpGndSpd','DiffImpVSpd']
mlpdiff=pd.concat([testdata[['ID','Time']],mlpdiff],axis=1)
mlpdiff
```

Out[25]:

	ID	Time	DiffImpLati	DiffImpLongi	DiffImpAltB	DiffImpGndSpd	DiffImpVSpd
0	31	57	0.030986	-0.080012	143.224495	15.857604	456.337302
1	31	58	2.515202	-1.149156	1200.313552	18.930492	-78.510460
2	31	59	0.365552	0.158765	641.018100	9.426789	475.201409
3	31	60	0.394957	0.158731	634.562088	8.925772	464.583254
4	31	61	0.424363	0.158696	628.106077	8.424754	453.965099
...	...	...	...	...	...	...	...
208283	110	209	-0.565436	-0.182967	381.332759	3.101163	426.048452
208284	110	210	-0.605059	-0.525647	614.697938	2.642420	1063.733385
208285	110	211	-0.627885	-0.531315	645.888991	2.795457	1098.162362
208286	110	212	2.044134	-1.310498	-6.108681	70.656152	1038.595384
208287	110	213	-0.673538	-0.542651	708.271097	3.101530	1167.020316

208288 rows × 7 columns

In [26]:

```
# Adding residuals back to Imputed variables
mlpdiff['Latitude']=mlpdiff['DiffImpLati']+X_testcpy['impLatitude']
mlpdiff['Longitude']=mlpdiff['DiffImpLongi']+X_testcpy['impLongitude']
mlpdiff['AltB']=mlpdiff['DiffImpAltB']+X_testcpy['impAltB']
mlpdiff['GndSpd']=mlpdiff['DiffImpGndSpd']+X_testcpy['impGndSpd']
mlpdiff['VSpd']=mlpdiff['DiffImpVSpd']+X_testcpy['impVSpd']
pred=mlpdiff[['ID','Time','Latitude','Longitude','AltB','GndSpd','VSpd']]
pred
```

Out[26]:

```
         ID  Time   Latitude  Longitude         AltB      GndSpd        VSpd
      0  31    57  41.297622 -95.837504  1383.324495   60.317604  384.897302
      1  31    58  43.781543 -96.906658  2442.913552   65.192992 -123.799210
      2  31    59  41.631599 -95.598748  1886.118100   57.491789  456.063909
      3  31    60  41.660710 -95.598793  1882.162088   58.793272  471.597004
      4  31    61  41.689822 -95.598838  1878.206077   60.094754  487.130099
    ...  ...   ...        ...        ...          ...         ...         ...
 208283 110   209  42.802724 -95.323805  2368.532759   97.213663  -45.506548
 208284 110   210  42.763246 -95.665930  2595.897938   96.534295  541.262135
 208285 110   211  42.740565 -95.671043  2621.088991   96.466707  524.774862
 208286 110   212  45.412729 -96.449672  1963.091319  164.106777  414.291634
 208287 110   213  42.695202 -95.681271  2671.471097   96.331530  491.800316

208288 rows × 7 columns
```

In [ ]: