Task 3: Application of Machine Learning

To retrieve the data from https://pds-geosciences.wustl.edu/missions/lunarp/reduced_special.html and develop an ML model to derive elemental data directly from the gamma ray spectra.

In [9]:
```
# Importing essential libraries

import pandas as pd
import urllib.request
```

In [2]:
```
# Elemental Data collected

data_collected = ['oxygen5d.txt',
                  'silicon5d.txt',
                  'titanium5d.txt',
                  'aluminum5d.txt',
                  'iron5d.txt',
                  'magnesium5d.txt',
                  'calcium5d.txt',
                  'uranium5d.txt',
                  'potassium5d.txt',
                  'thorium5d.txt']

url = "https://pds-geosciences.wustl.edu/missions/lunarp/reduced/"

for i in data_collected:
    urllib.request.urlretrieve(url + i, f"./Task3/{i}")
```

In [3]:
```
# Removing comments.

for name in data_collected:
    with open(f'./Task3/{name}', 'r+') as f:
        lines = f.readlines()
        f.seek(0)
        for line in lines:
            if not line.startswith('#'):
                f.write(line)
        f.truncate()
```

In [4]:
```
# Converting to .txt file to a datframe and saving it as a .csv file.

for name in data_collected:
    with open(f'./Task3/{name}') as f:
        data = f.readlines()
    cleaned_data = []
    for i in data:
        temp = i.split()
        temp = [float(j.strip(",")) for j in temp]
        # adding new column specifying the element name, used while training the model.
        temp.append(name[:-4])
        cleaned_data.append(temp)
    df = pd.DataFrame(cleaned_data, columns=['lat(min)', 'lat(max)', 'lon(min)', 'lon(max)', 'weight_percent', 'element'])
    df.to_csv(f'./Task3/{name[:-4]}.csv', index = False)
```

In [8]:
```
# Combining all elemental dataframes into one dataframe called cumulative_data.csv
# this dataframe will be used for training the ML model

all_frames = []

for name in data_collected:
    df = pd.read_csv(f"./Task3/{name[:-4]}.csv")
    all_frames.append(df)

result = pd.concat(all_frames)
result.to_csv("./Task3/cumulative_data.csv", index = False)
```

Using KNN to predict Elemental Composition

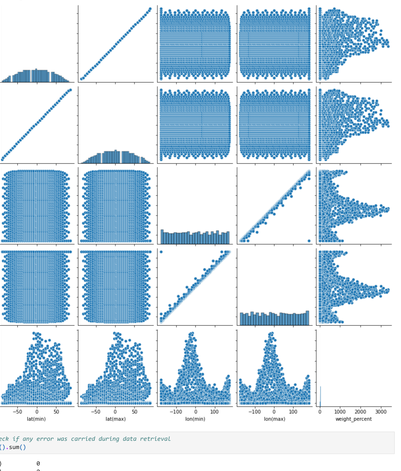
In [45]:
```
# Importing essential libraries

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import MinMaxScaler
```

In [36]:
```
df = pd.read_csv("./Task3/cumulative_data.csv")
print(f'Number of rows: {df.shape[0]} and Columns: {df.shape[1]}')
df.info()
```

```
Number of rows: 17900 and Columns: 6
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17900 entries, 0 to 17899
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   lat(min)        17900 non-null  float64
 1   lat(max)        17900 non-null  float64
 2   lon(min)        17900 non-null  float64
 3   lon(max)        17900 non-null  float64
 4   weight_percent  17900 non-null  float64
 5   element         17900 non-null  object 
dtypes: float64(5), object(1)
memory usage: 839.2+ KB
```

In [17]:
```
# Visualising data
sns.pairplot(df)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x1ddff8c7fa0>
```



In [18]:
```
# To check if any error was carried during data retrieval
df.isna().sum()
```

Out[18]:
```
lat(min)          0
lat(max)          0
lon(min)          0
lon(max)          0
weight_percent    0
element           0
dtype: int64
```

In [38]:
```
# Encoding the categorical data, element
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['encoded_element'] = labelencoder.fit_transform(df['element'])
df
```

Out[38]:

	lat(min)	lat(max)	lon(min)	lon(max)	weight_percent	element	encoded_element
0	-90.0	-87.5	-180.0	180.0	45.28900	oxygen	4
1	-87.5	-82.5	-180.0	-135.0	45.52900	oxygen	4
2	-87.5	-82.5	-135.0	-90.0	44.66000	oxygen	4
3	-87.5	-82.5	-90.0	-45.0	44.70600	oxygen	4
4	-87.5	-82.5	-45.0	0.0	44.77700	oxygen	4
...	...	...	...	...	...	...	...
17895	82.5	87.5	0.0	45.0	1.65770	thorium	7
17896	82.5	87.5	45.0	90.0	1.47520	thorium	7
17897	82.5	87.5	90.0	135.0	1.13880	thorium	7
17898	82.5	87.5	135.0	180.0	0.92528	thorium	7
17899	87.5	90.0	-180.0	180.0	1.31050	thorium	7

17900 rows × 7 columns

In [92]:
```
scaler = MinMaxScaler()
# np_data = np.array(data_0['sum_accepted_spectrum'])
# np_data_4 = np_data_4.reshape(-1,1)
df["norm_lat(min)"] = scaler.fit_transform(np.array(df["lat(min)"]).reshape(-1, 1))
df["norm_lat(max)"] = scaler.fit_transform(np.array(df["lat(max)"]).reshape(-1, 1))
df["norm_lon(min)"] = scaler.fit_transform(np.array(df["lon(min)"]).reshape(-1, 1))
df["norm_lon(max)"] = scaler.fit_transform(np.array(df["lon(max)"]).reshape(-1, 1))
df["norm_weight_percent"] = scaler.fit_transform(np.array(df["weight_percent"]).reshape(-1, 1))
df["norm_element"] = scaler.fit_transform(np.array(df["encoded_element"]).reshape(-1, 1))
df
```

Out[92]:
```
array([[0.        ],
       [0.12676056],
       [0.12676056],
       ...,
       [0.74647887],
       [0.74647887],
       [0.        ]])
```

In [51]:
```
X = df[['norm_lat(min)','norm_lat(max)','norm_lon(min)','norm_lon(max)','norm_element']]
Y = df['norm_weight_percent']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.2, random_state=42)
print(f'Train: {X_train.shape, Y_train.shape} \nTest: {X_test.shape, Y_test.shape}')
```

```
Train: ((14320, 5), (14320,))
Test: ((3580, 5), (3580,))
```

In [97]:
```
knn_model = KNeighborsRegressor(n_neighbors=6,
                                weights='uniform',
                                algorithm='auto',
                                leaf_size=30,
                                p=2).fit(X_train, Y_train)
score_knn = knn_model.score(X_test, Y_test)
print(f'KNN Score: {score_knn}')
```

```
KNN Score: 0.9760128705004325
```

Model Performance

NOTE: The true values and predicted values have been normalised.

In [99]:
```
predictions = knn_model.predict(X_test)
predict_dataframe = pd.DataFrame({ 'True Value': Y_test,
                                   'Prediction': predictions,
                                   'Error': Y_test - preds})
predict_dataframe
```

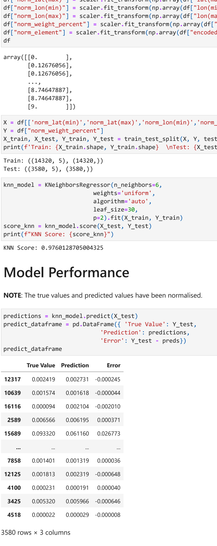
Out[99]:

	True Value	Prediction	Error
12317	0.002419	0.002731	-0.000245
10639	0.001574	0.001618	-0.000044
16116	0.000094	0.002104	-0.002010
2589	0.006566	0.006195	0.000371
15689	0.093320	0.061160	0.026773
...	...	...	...
7858	0.001401	0.001319	0.000036
12125	0.001811	0.002319	-0.000648
4100	0.000231	0.000191	0.000040
3425	0.005320	0.005966	-0.000646
4518	0.000022	0.000029	-0.000008

3580 rows × 3 columns

Task Complete

The ML Model was succesfully trained using KNN. I used a ML tool, **Dataiku** to determine the best possible ML algorithm for the given dataset which I constructed from the specified website. Based on the preliminary results, KNN was chosen and the accuracy was improved.



Kindly note that certain outputs while fetching the data have been cleared. Only necessary outputs have been displayed