Task2_ML4SCI

March 31, 2022

0.1 Importing all the necessary libraries

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
import matplotlib.pyplot as plt
from PIL import Image
```

0.2 Reading the CSV files as a DataFrame

[3]:
```
al_si = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪alsimap_smooth_032015.png.csv',header=None)
```

[4]:
```
ca_si = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪casimap_smooth_032015.png.csv',header=None)
```

[5]:
```
fe_si = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪fesimap_smooth_032015.png.csv',header=None)
```

[6]:
```
bottom = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪mercury-albedo-resized-bottom-half.png.csv',header=None)
```

[7]:
```
top = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪mercury-albedo-top-half.png.csv',header=None)
```

[8]:
```
mg_si = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪mgsimap_smooth_032015.png.csv',header=None)
```

[9]:
```
s_si = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Mercury/
↪ssimap_smooth_032015.png.csv',header=None)
```

[10]:
```
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
```

0.3 We now define a function to convert the 720x1440 matrix to to column vector. This is done to find relationship between pixel values of top half of albedo with the top half of chemical composition images.

bold text

[11]:
```
def reshape(dataFrame):
  data = dataFrame.values.reshape(-1,1)

  return data
```

[12]:
```
Xtrain = reshape(top)
```

[13]:
```
Xtest = reshape(bottom)
```

0.4 We apply simple Linear Regression model to predict the brigtness of each pixel using the pixel values of bottom half of the albedo

[14]:
```
def LinearRegressionModel(chemicalData,model):
  yTrain = reshape(chemicalData)
  model.fit(Xtrain,yTrain)
  yPrediction = model.predict(Xtest)
  yImage = yPrediction.reshape(720,1440)

  return yPrediction , yImage
```

[15]:
```
lr = LinearRegression()
```

[16]:
```
al_si_lr_prediction , al_si_lr_bottom = LinearRegressionModel(al_si,lr)
```

[17]:
```
ca_si_lr_prediction , ca_si_lr_bottom = LinearRegressionModel(ca_si,lr)
```

[18]:
```
fe_si_lr_prediction , fe_si_lr_bottom = LinearRegressionModel(fe_si,lr)
```

[19]:
```
mg_si_lr_prediction , mg_si_lr_bottom = LinearRegressionModel(mg_si,lr)
```

[20]:
```
s_si_lr_prediction , s_si_lr_bottom = LinearRegressionModel(s_si,lr)
```

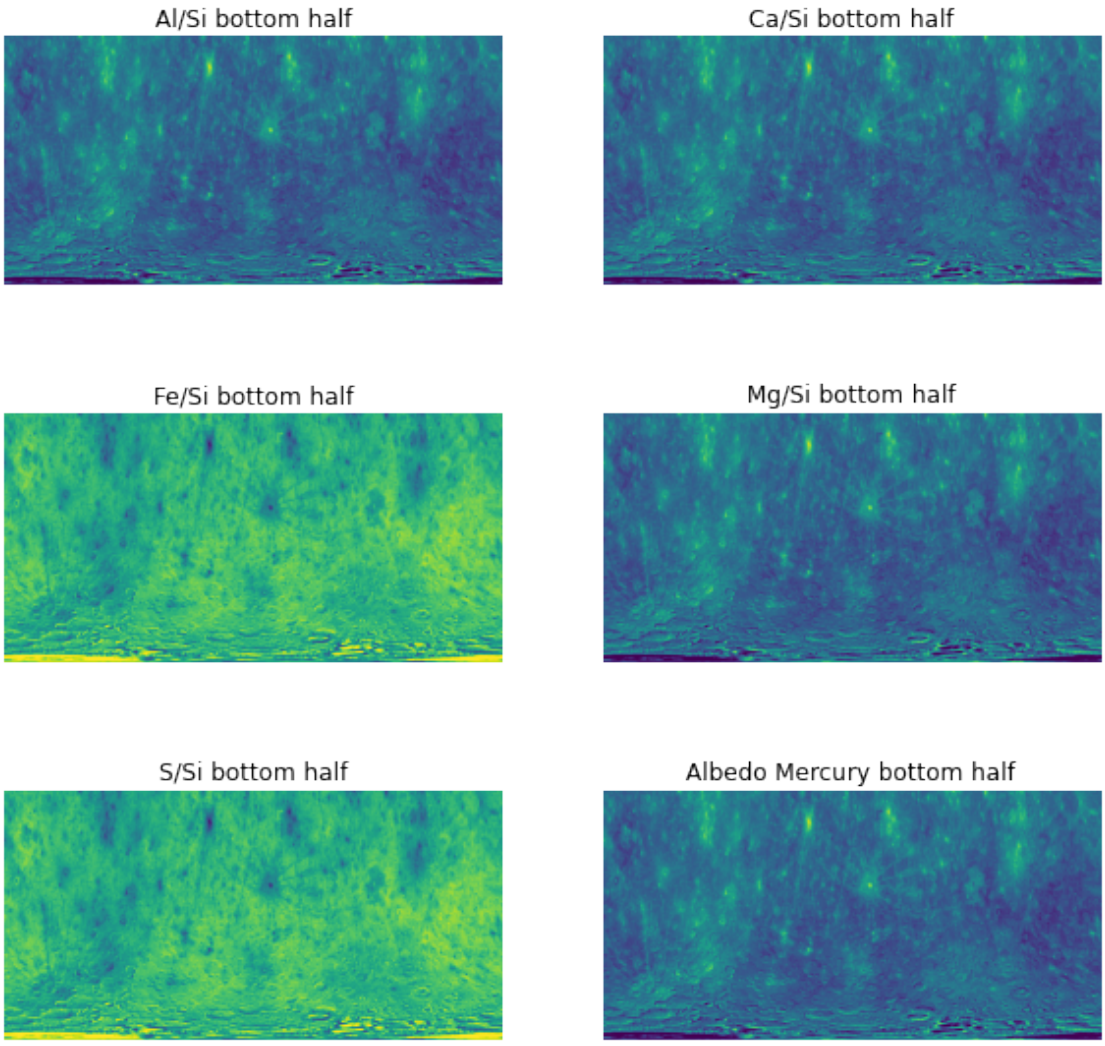
0.5 We now plot the predicted images of each chemical composition and the bottom half of the albedo

[21]:
```
fig = plt.figure(figsize=(10,10))
rows = 3
columns = 2
```

```
Image1 = al_si_lr_bottom
Image2 = ca_si_lr_bottom
Image3 = fe_si_lr_bottom
Image4 = mg_si_lr_bottom
Image5 = s_si_lr_bottom
Image6 = bottom


fig.add_subplot(rows, columns, 1)

plt.imshow(Image1)
plt.axis('off')
plt.title("Al/Si bottom half")


fig.add_subplot(rows, columns, 2)


plt.imshow(Image2)
plt.axis('off')
plt.title("Ca/Si bottom half")


fig.add_subplot(rows, columns, 3)


plt.imshow(Image3)
plt.axis('off')
plt.title("Fe/Si bottom half")


fig.add_subplot(rows, columns, 4)


plt.imshow(Image4)
plt.axis('off')
plt.title("Mg/Si bottom half")

fig.add_subplot(rows, columns, 5)


plt.imshow(Image5)
plt.axis('off')
plt.title("S/Si bottom half")

fig.add_subplot(rows, columns, 6)
```

```
plt.imshow(Image6)
plt.axis('off')
plt.title("Albedo Mercury bottom half")
```

[21]: Text(0.5, 1.0, 'Albedo Mercury bottom half')



0.6 A function is defined to predict brightness of each pixel of chemical composition using XG Boost model and then reshaped to construct the bottom half for each prediction

[22]:
```
def XGBModel(chemicalData,model):
  yTrain = reshape(chemicalData)
  model.fit(Xtrain,yTrain)
  yPrediction = model.predict(Xtest)
  yImage = yPrediction.reshape(720,1440)

  return yPrediction , yImage
```

[23]:
```
xgb =  XGBRegressor()
```

[24]:
```
al_si_xgb_prediction , al_si_xgb_bottom = XGBModel(al_si,xgb)
```

```
[13:17:52] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[25]:
```
ca_si_xgb_prediction , ca_si_xgb_bottom = XGBModel(ca_si,xgb)
```

```
[13:18:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[26]:
```
fe_si_xgb_prediction , fe_si_xgb_bottom = XGBModel(fe_si,xgb)
```

```
[13:18:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[27]:
```
mg_si_xgb_prediction , mg_si_xgb_bottom = XGBModel(mg_si,xgb)
```

```
[13:19:12] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[28]:
```
s_si_xgb_prediction , s_si_xgb_bottom = XGBModel(s_si,xgb)
```

```
[13:19:37] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

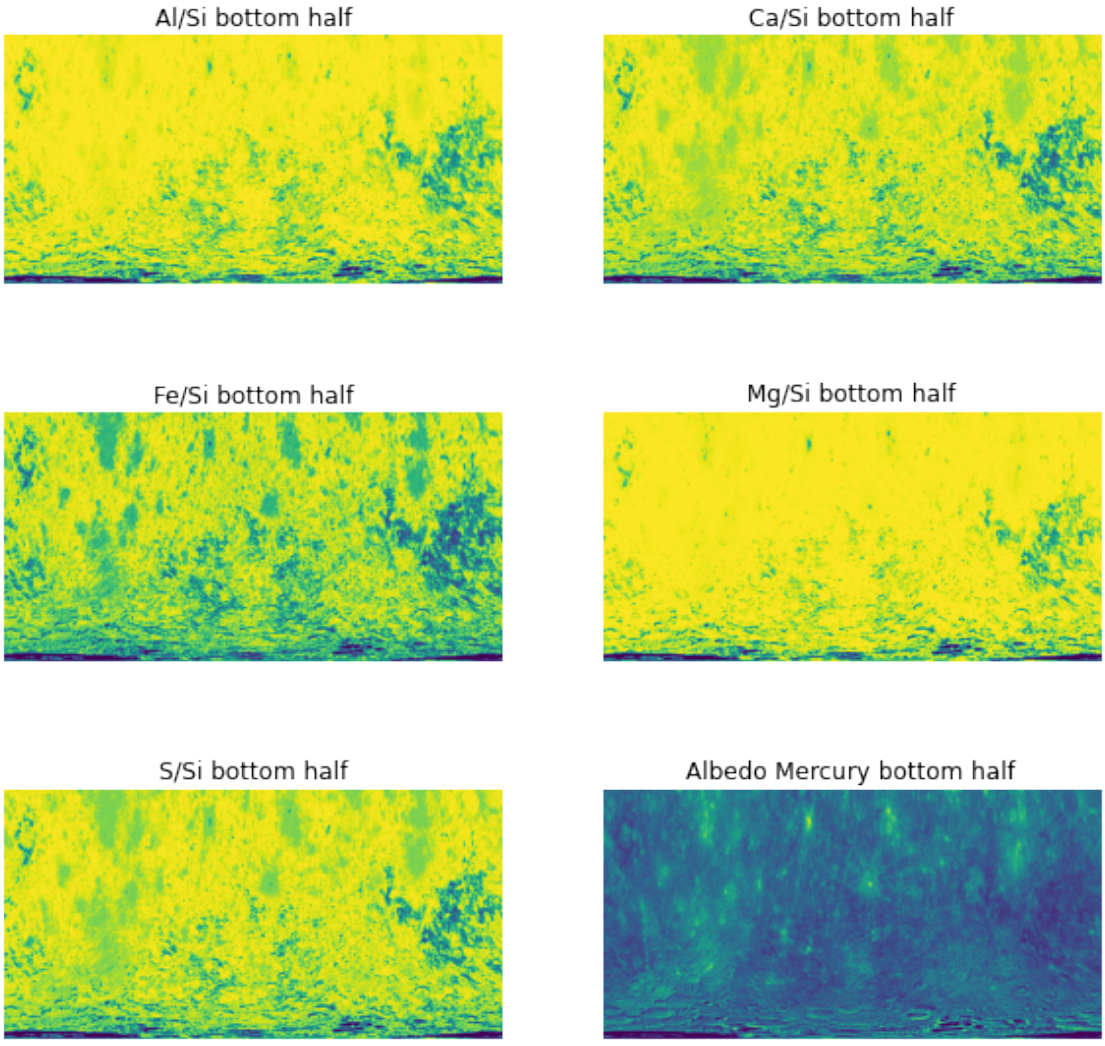
0.7 We now plot the predicted images for each chemical compostion using the bottom half of the albedo

[29]:
```
fig = plt.figure(figsize=(10,10))
rows = 3
columns = 2
```

```
Image1 = al_si_xgb_bottom
Image2 = ca_si_xgb_bottom
Image3 = fe_si_xgb_bottom
Image4 = mg_si_xgb_bottom
Image5 = s_si_xgb_bottom
Image6 = bottom


fig.add_subplot(rows, columns, 1)

plt.imshow(Image1)
plt.axis('off')
plt.title("Al/Si bottom half")


fig.add_subplot(rows, columns, 2)


plt.imshow(Image2)
plt.axis('off')
plt.title("Ca/Si bottom half")


fig.add_subplot(rows, columns, 3)


plt.imshow(Image3)
plt.axis('off')
plt.title("Fe/Si bottom half")


fig.add_subplot(rows, columns, 4)


plt.imshow(Image4)
plt.axis('off')
plt.title("Mg/Si bottom half")

fig.add_subplot(rows, columns, 5)


plt.imshow(Image5)
plt.axis('off')
plt.title("S/Si bottom half")

fig.add_subplot(rows, columns, 6)
```

```
plt.imshow(Image6)
plt.axis('off')
plt.title("Albedo Mercury bottom half")
```

[29]: Text(0.5, 1.0, 'Albedo Mercury bottom half')



[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Task2_ML4SCI.ipynb')
```

```
--2022-03-31 13:20:03--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.109.133, 185.199.108.133, 185.199.111.133, …
```

```
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-03-31 13:20:03 (21.1 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```