Task1_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
```

[3]:
```
%matplotlib inline
```

0.2 Reading all the csv files of albedo and chemical composition as a Dataframe

[4]:
```
albedo = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Moon/
 ↪Albedo_Map.csv',header=None)
```

[5]:
```
fe = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Moon/LPFe_Map.
 ↪csv',header=None)
```

[6]:
```
k = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Moon/LPK_Map.
 ↪csv',header=None)
```

[7]:
```
th = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Moon/LPTh_Map.
 ↪csv',header=None)
```

[8]:
```
ti = pd.read_csv('/content/drive/MyDrive/GSOC22/ML4SCI/Messenger/Moon/LPTi_Map.
 ↪csv',header=None)
```

0.3 Now we will split the image into two halves and reshape them to column vector.

[9]:
```
def splitAndReshape(data):
  half = 360
  train,test = data.iloc[:,:half] , data.iloc[:,half:]
  trainColumnVector , testColumnVector = train.values.reshape(-1,1), test.
 ↪values.reshape(-1,1)
```

```

    return trainColumnVector , testColumnVector
```

[10]:
```
albedoTrain, albedoTest = splitAndReshape(albedo)
```

[11]:
```
feTrain , feTest = splitAndReshape(fe)
```

[12]:
```
kTrain , kTest = splitAndReshape(k)
```

[13]:
```
thTrain , thTest = splitAndReshape(th)
```

[14]:
```
tiTrain , tiTest = splitAndReshape(ti)
```

0.4 Now we will concate column vector of each dataframw where each row represents the pixel values of the image accordingly

[15]:
```
dataTrain = np.concatenate([feTrain,kTrain,thTrain,tiTrain,albedoTrain], axis=1)
```

[16]:
```
dataTest = np.concatenate([feTest,kTest,thTest,tiTest,albedoTest],axis=1)
```

0.5 The features will be the pixel values of the chemical composition and the target variable will be the brigtness of each pixel in the albedo map

[17]:
```
Xtrain,Xtest = dataTrain[:,:-1],dataTest[:,:-1]
```

[18]:
```
print(Xtrain.shape)
```

```
(129600, 4)
```

[19]:
```
ytrain,ytest = dataTrain[:,-1].reshape(-1,1) , dataTest[:,-1].reshape(-1,1)
```

[20]:
```
print(ytrain.shape)
```

```
(129600, 1)
```

[21]:
```
from sklearn.linear_model import LinearRegression
```

[22]:
```
from sklearn.tree import DecisionTreeRegressor
```

[23]:
```
from xgboost import XGBRegressor
```

[24]:
```
from sklearn.ensemble import RandomForestRegressor
```

0.6 Now we will train various regression models on the chemical composition data to predict the pixel values of albedo. We will then reshape the column vector to 360x360 so that we can plot the predicted image

[25]:
```
def results(model):
  model.fit(Xtrain,ytrain)
  prediction = model.predict(Xtest)
  image = prediction.reshape(360,360)

  return prediction,image
```

[26]:
```
lr  =  LinearRegression()
dt  =  DecisionTreeRegressor()
xgb =  XGBRegressor()
rf  =  RandomForestRegressor()
```

[27]:
```
lrPrediction , lrImage = results(lr)
```

[28]:
```
dtPrediction, dtImage = results(dt)
```

[29]:
```
xgbPrediction, xgbImage = results(xgb)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/core.py:613: UserWarning: Use
subset (sliced data) of np.ndarray is not recommended because it will generate
extra copies and increase memory consumption
  warnings.warn("Use subset (sliced data) of np.ndarray is not recommended " +

[13:13:27] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
is now deprecated in favor of reg:squarederror.
```

[30]:
```
rfPrediction, rfImage = results(rf)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
```

[31]:
```
import sklearn.metrics as skmetrics
```

0.7 We are choosing Root Mean Squared Error to quantify the measurement of performance of each model

[32]:
```
def rmse(prediction,actual):
  error = skmetrics.mean_squared_error(prediction, actual, squared = False)

  return error
```

[33]:
```
lrError = rmse(lrPrediction,albedoTest)
```

[34]:
```
print(lrError)
```

```
0.03201506414975387
```

[35]:
```
dtError = rmse(dtPrediction,albedoTest)
```

[36]:
```
print(dtError)
```

```
0.040546507871480265
```

[37]:
```
xgbError = rmse(xgbPrediction,albedoTest)
```

[38]:
```
print(xgbError)
```

```
0.03089871895457188
```

[39]:
```
rfError = rmse(rfPrediction,albedoTest)
```

[40]:
```
print(rfError)
```

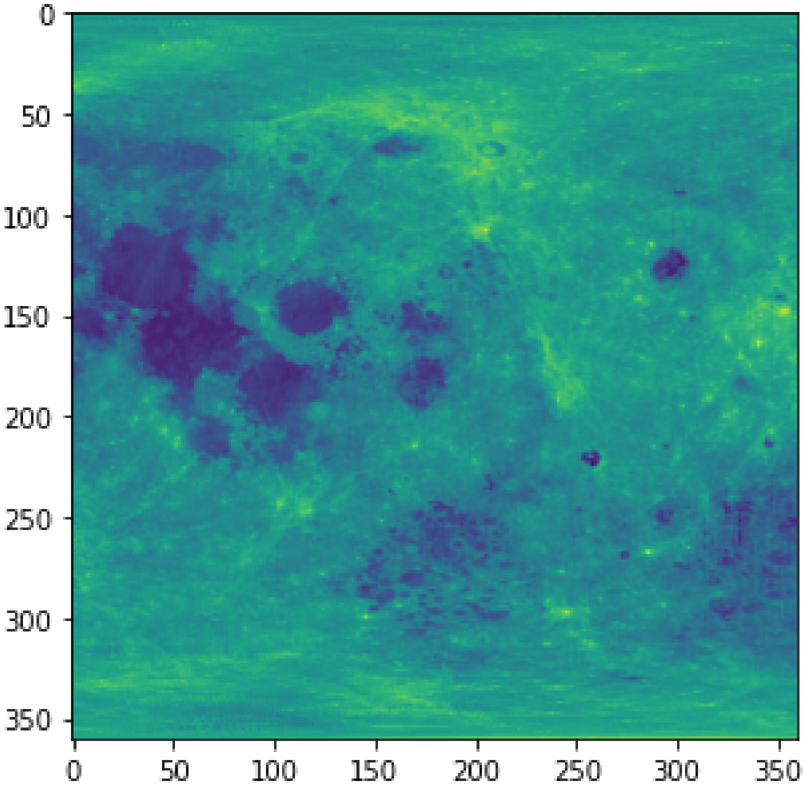
```
0.03427114178738713
```

[41]:
```
testImage = albedo.iloc[:,360:]
```

0.8 Now we will plot the right half of the albedo and also the predicted right half of albedo from each model

[42]:
```
fig = plt.figure(figsize=(5,5))
plt.imshow(testImage)
```

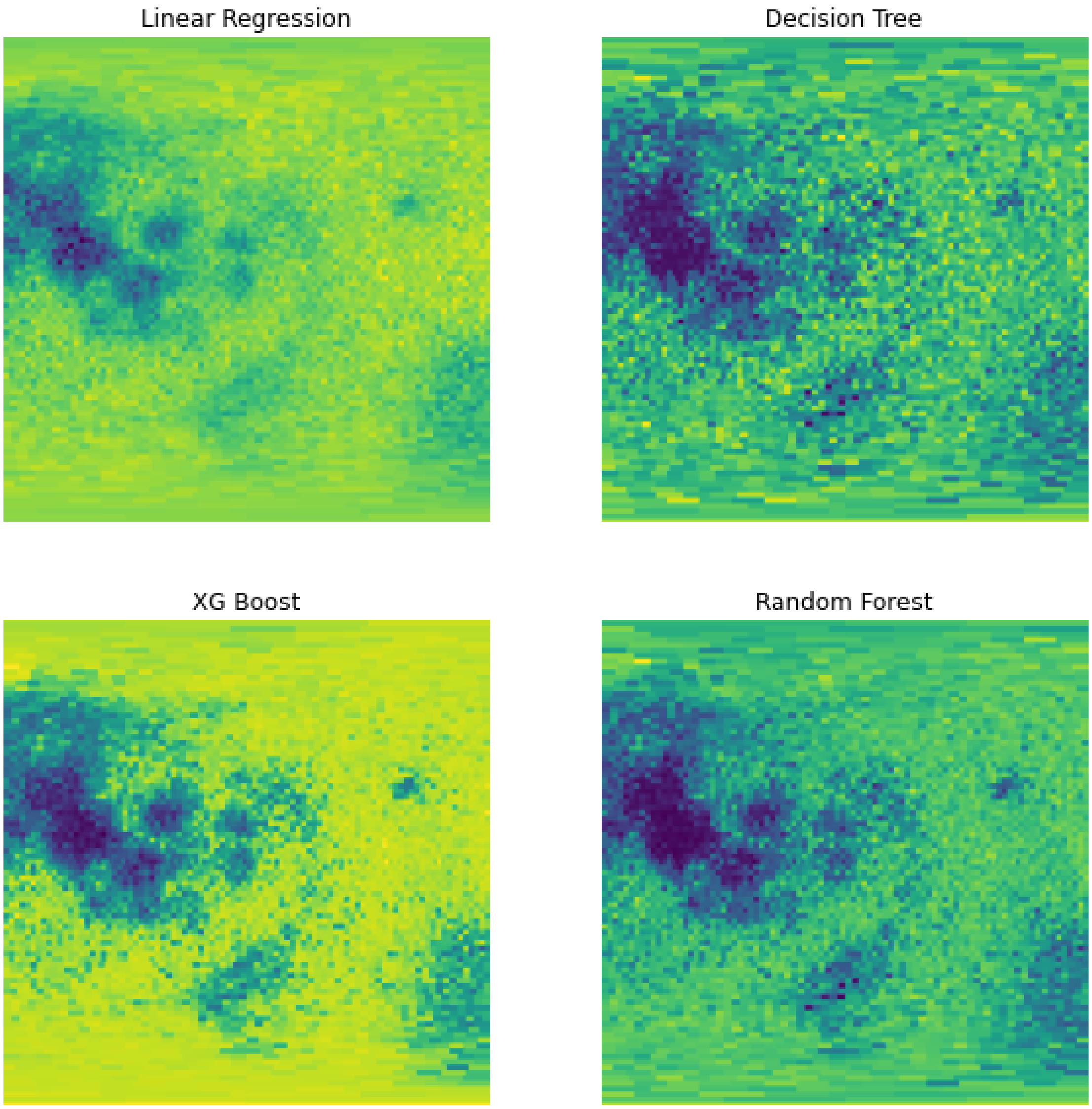
[42]:
```
<matplotlib.image.AxesImage at 0x7fb1d5aaeb90>
```



[43]:
```
fig = plt.figure(figsize=(10,10))
rows = 2
columns = 2


Image1 = lrImage
Image2 = dtImage
Image3 = xgbImage
Image4 = rfImage


fig.add_subplot(rows, columns, 1)

plt.imshow(Image1)
plt.axis('off')
plt.title("Linear Regression")


fig.add_subplot(rows, columns, 2)
```

```
plt.imshow(Image2)
plt.axis('off')
plt.title("Decision Tree")


fig.add_subplot(rows, columns, 3)


plt.imshow(Image3)
plt.axis('off')
plt.title("XG Boost")


fig.add_subplot(rows, columns, 4)


plt.imshow(Image4)
plt.axis('off')
plt.title("Random Forest")
```

```
[43]: Text(0.5, 1.0, 'Random Forest')
```



0.9 We will define a function to find the difference in the pixel values between the predicted image and the actual image

[44]:

```
def difference(predictedImage,actualImage):
  diff = predictedImage - actualImage

  return diff
```

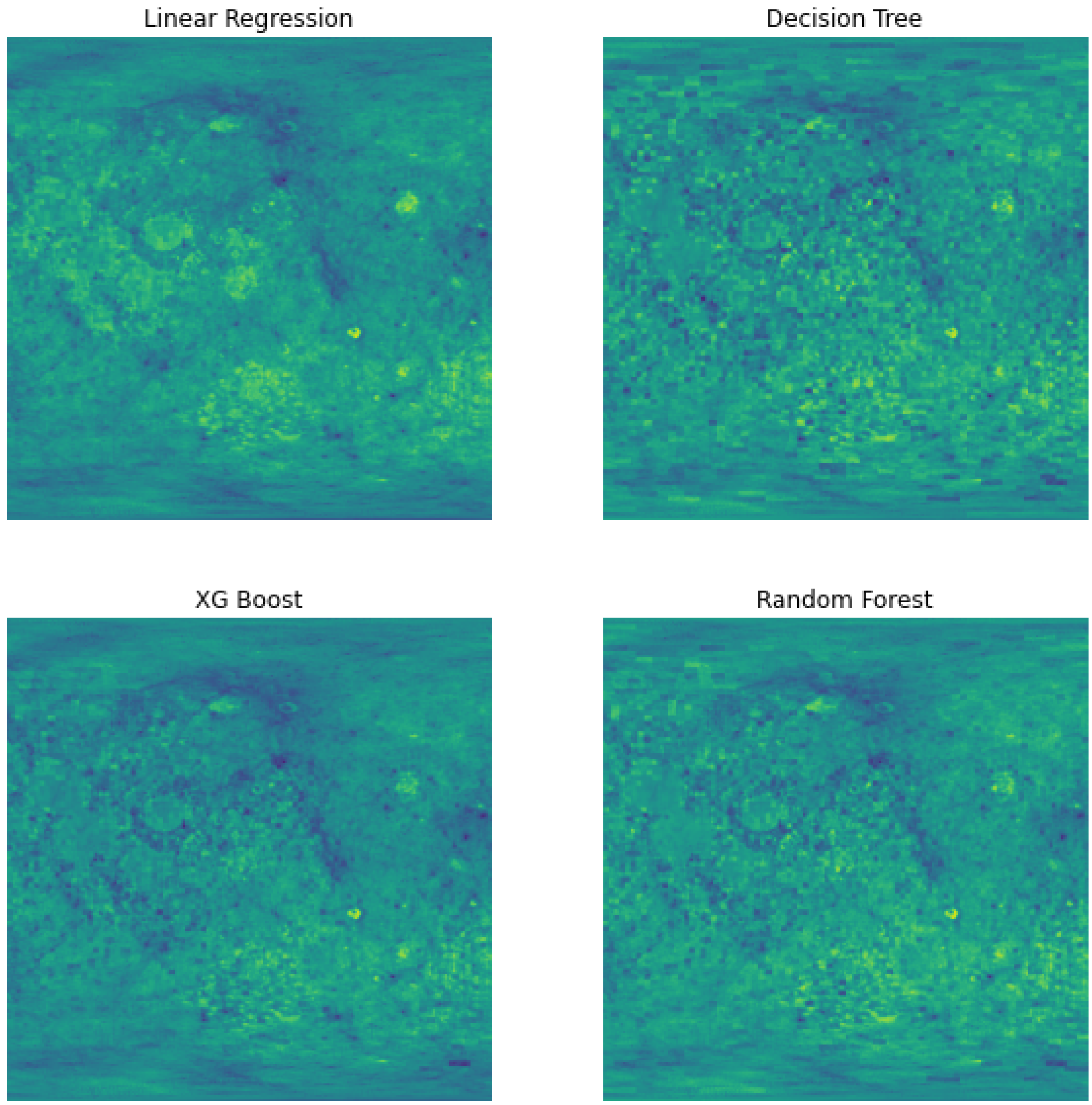
[45]:

```
lrDifference = difference(lrImage,testImage)
dtDifference = difference(dtImage,testImage)
xgbDifference = difference(xgbImage,testImage)
```

```
rfDifference = difference(rfImage,testImage)
```

[46]:
```
fig = plt.figure(figsize=(10,10))
rows = 2
columns = 2


Image1 = lrDifference
Image2 = dtDifference
Image3 = xgbDifference
Image4 = rfDifference


fig.add_subplot(rows, columns, 1)


plt.imshow(Image1)
plt.axis('off')
plt.title("Linear Regression")


fig.add_subplot(rows, columns, 2)


plt.imshow(Image2)
plt.axis('off')
plt.title("Decision Tree")


fig.add_subplot(rows, columns, 3)


plt.imshow(Image3)
plt.axis('off')
plt.title("XG Boost")


fig.add_subplot(rows, columns, 4)


plt.imshow(Image4)
plt.axis('off')
plt.title("Random Forest")
```

[46]:
```
Text(0.5, 1.0, 'Random Forest')
```


Linear Regression

Decision Tree

XG Boost

Random Forest

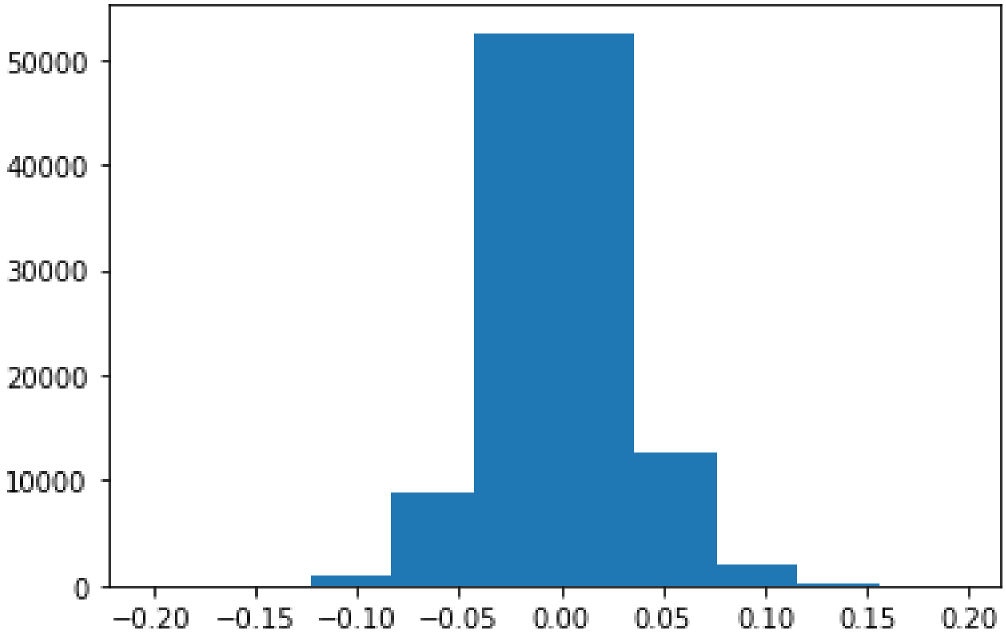
0.10 Plotting histogram of difference values between preidcted image and actual image

[47]:

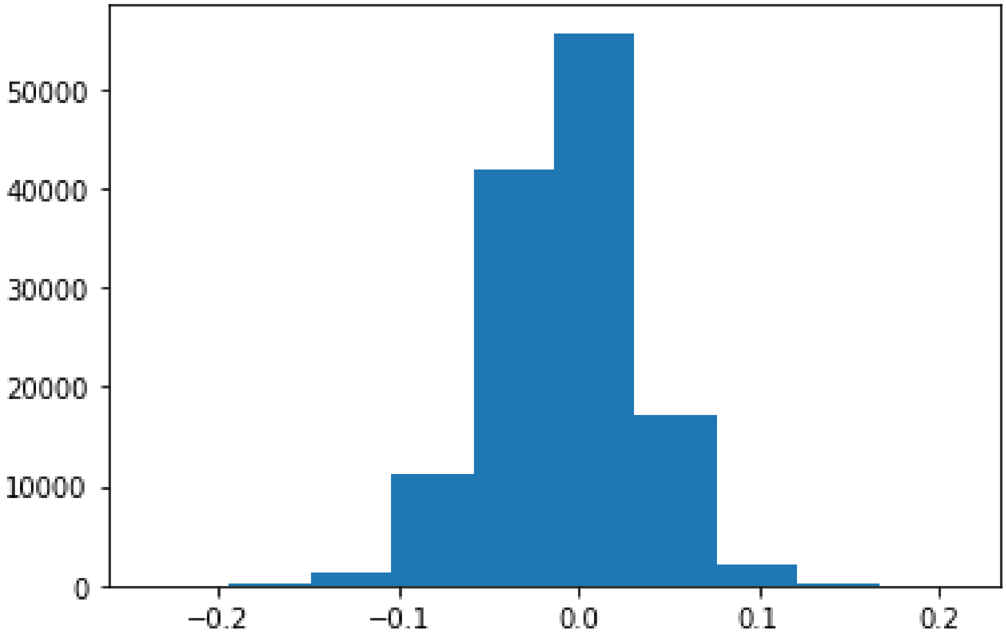
```
def histogram(difference):
  data = difference.values.reshape(129600,1)

  return plt.hist(data)
```

[48]:

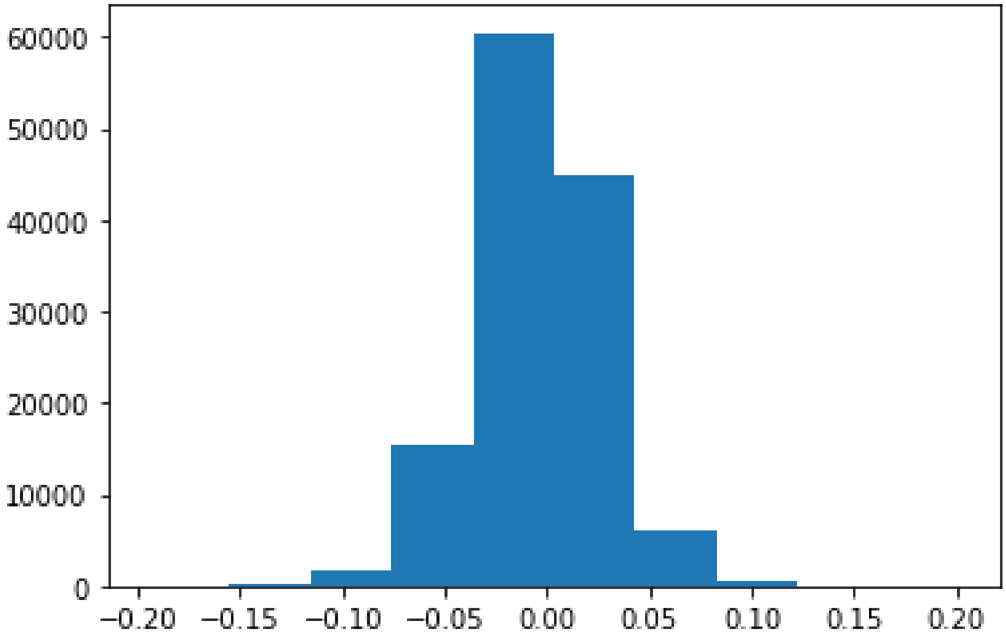
```
lrHistogram = histogram(lrDifference)
plt.show(lrHistogram)
```



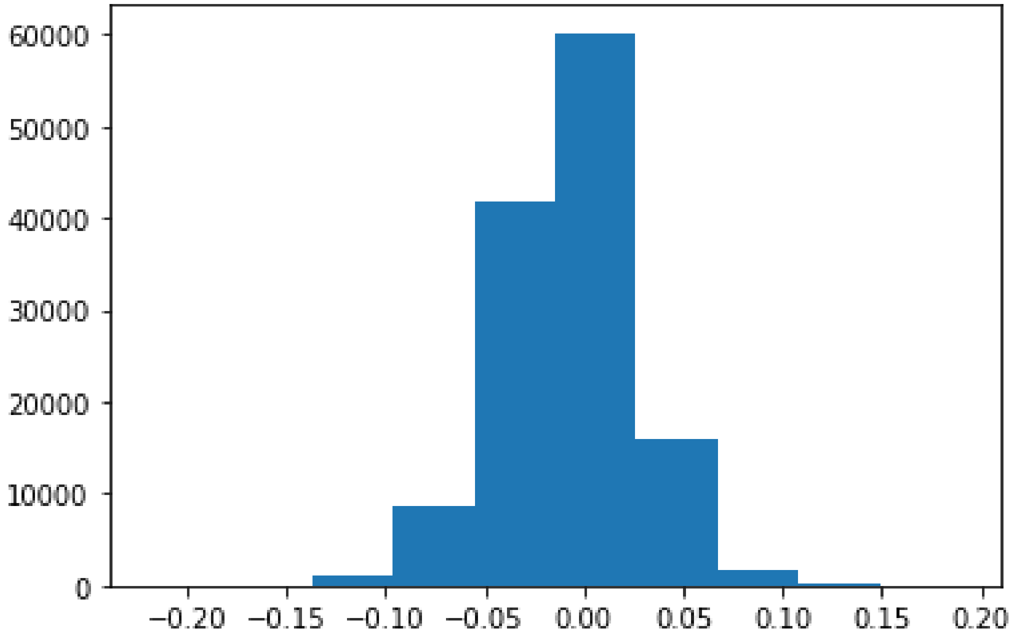
[49]:
```
dtHistogram = histogram(dtDifference)
plt.show(dtHistogram)
```



```
[50]: xgbHistogram = histogram(xgbDifference)
      plt.show(xgbHistogram)
```



```
[51]: rfHistogram = histogram(rfDifference)
      plt.show(rfHistogram)
```



[ ]:

```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Task1_ML4SCI.ipynb')
```

```
--2022-03-31 13:14:53--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.111.133, 185.199.110.133, 185.199.109.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.111.133|: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 13:14:53 (17.5 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%
```