In [1]:
```python
%matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
```

In [2]:
```python
# plotting the maps of all dataset.
```

In [3]:
```python
albedo_map_data=pd.read_csv("..\ML4SCI_GSoC\Messenger\Moon\Albedo_Map.csv",header=None)
```

In [4]:
```python
albedo_map_data.head(5)
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	710	711
0	0.331936	0.332611	0.332240	0.331028	0.331094	0.332614	0.331964	0.329994	0.327853	0.326532	...	0.330389	0.32908
1	0.338990	0.340417	0.334623	0.333716	0.331404	0.331733	0.335648	0.335849	0.333166	0.332413	...	0.350386	0.34650
2	0.324930	0.325832	0.328177	0.325871	0.321231	0.321791	0.322595	0.325254	0.329132	0.325335	...	0.329577	0.33220
3	0.327572	0.327171	0.333880	0.326805	0.328176	0.327048	0.326413	0.332305	0.325137	0.325540	...	0.317473	0.31428
4	0.347444	0.340715	0.330832	0.335570	0.340129	0.332496	0.335220	0.336632	0.334052	0.328328	...	0.319792	0.33253

5 rows × 720 columns

In [5]:
```python
albedo_map_data.describe()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	
count	360.000000	360.000000	360.000000	360.000000	360.000000	360.000000	360.000000	360.000000	360.000000	360.00000
mean	0.302457	0.300313	0.305411	0.302052	0.305815	0.302412	0.305667	0.301788	0.304058	0.30175
std	0.046314	0.050141	0.046161	0.052069	0.046651	0.048005	0.042653	0.046145	0.043310	0.04964
min	0.141172	0.163120	0.169143	0.155732	0.157913	0.179285	0.212120	0.182382	0.202475	0.18670
25%	0.270387	0.261086	0.274612	0.270031	0.274149	0.265750	0.271608	0.265047	0.267298	0.26081
50%	0.314536	0.316147	0.319692	0.318961	0.319510	0.315784	0.317110	0.316191	0.315347	0.31807
75%	0.336231	0.337058	0.338903	0.337865	0.336247	0.333630	0.336033	0.333705	0.336063	0.33759
max	0.408014	0.394116	0.394901	0.443510	0.466874	0.435901	0.415883	0.389032	0.391424	0.40255

8 rows × 720 columns

Visualizing all dataset.

In [6]:
```python
plt.subplots(figsize=(15,7))
sns.heatmap(albedo_map_data,xticklabels=False,yticklabels=False,cmap='gray')
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x24aeea97b48>

In [7]:
```python
Fe_map_data=pd.read_csv('..\ML4SCI_GSoC\Messenger\Moon\LPFe_Map.csv',header=None)
```

In [8]:
```python
Fe_map_data.head()
```

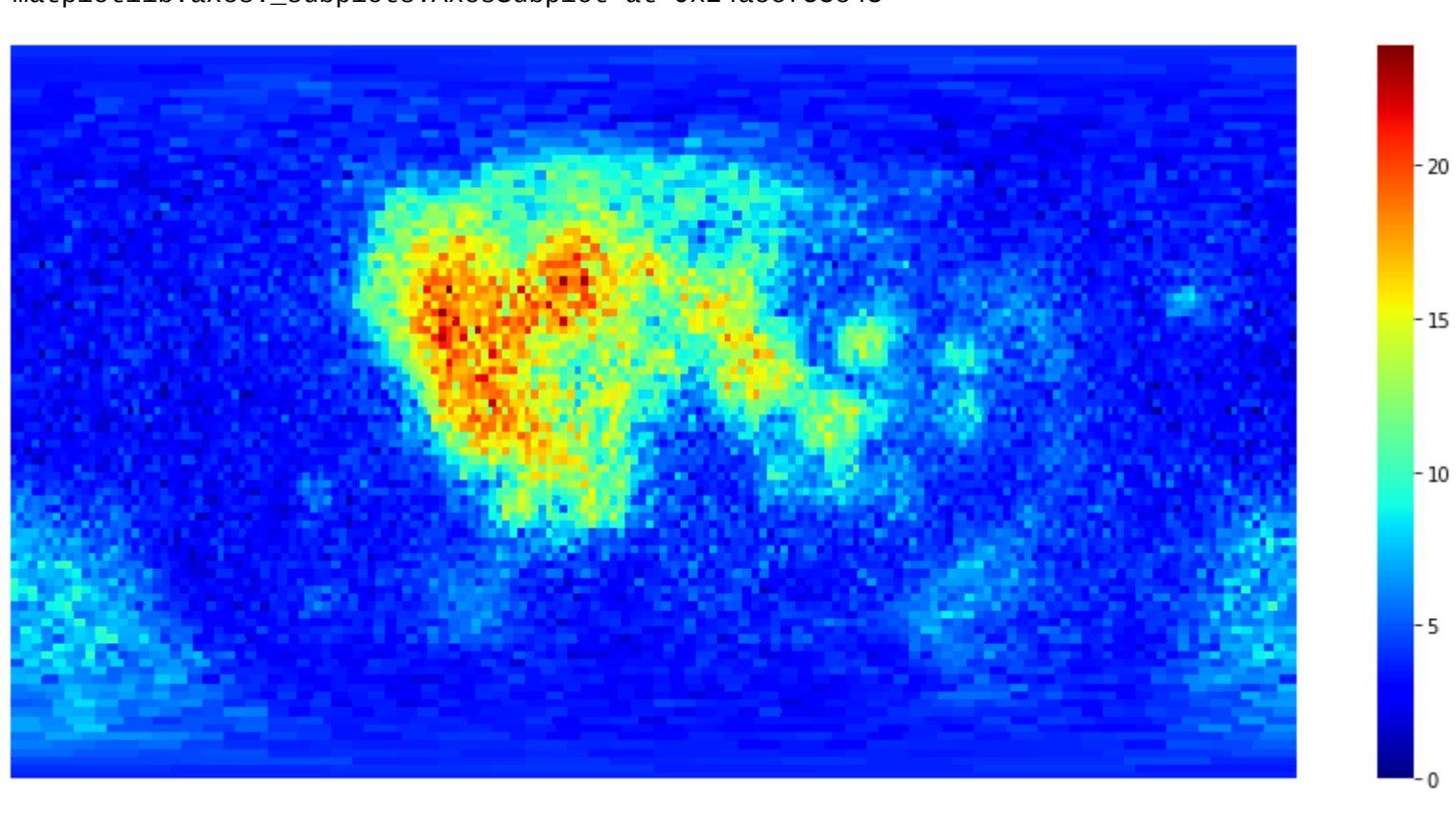
Out[8]:

	0	1	2	3	4	5	6	7	8	9	...	710	711	712	
0	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	...	4.04409	4.04409	4.04409	4.0
1	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	4.04409	...	4.04409	4.04409	4.04409	4.0
2	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	...	3.81408	3.81408	3.81408	3.8
3	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	...	3.81408	3.81408	3.81408	3.8
4	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	3.93285	...	3.81408	3.81408	3.81408	3.8

5 rows × 720 columns

In [9]:
```python
plt.subplots(figsize=(15,7))
sns.heatmap(Fe_map_data,xticklabels=False,yticklabels=False,cmap='jet')
```

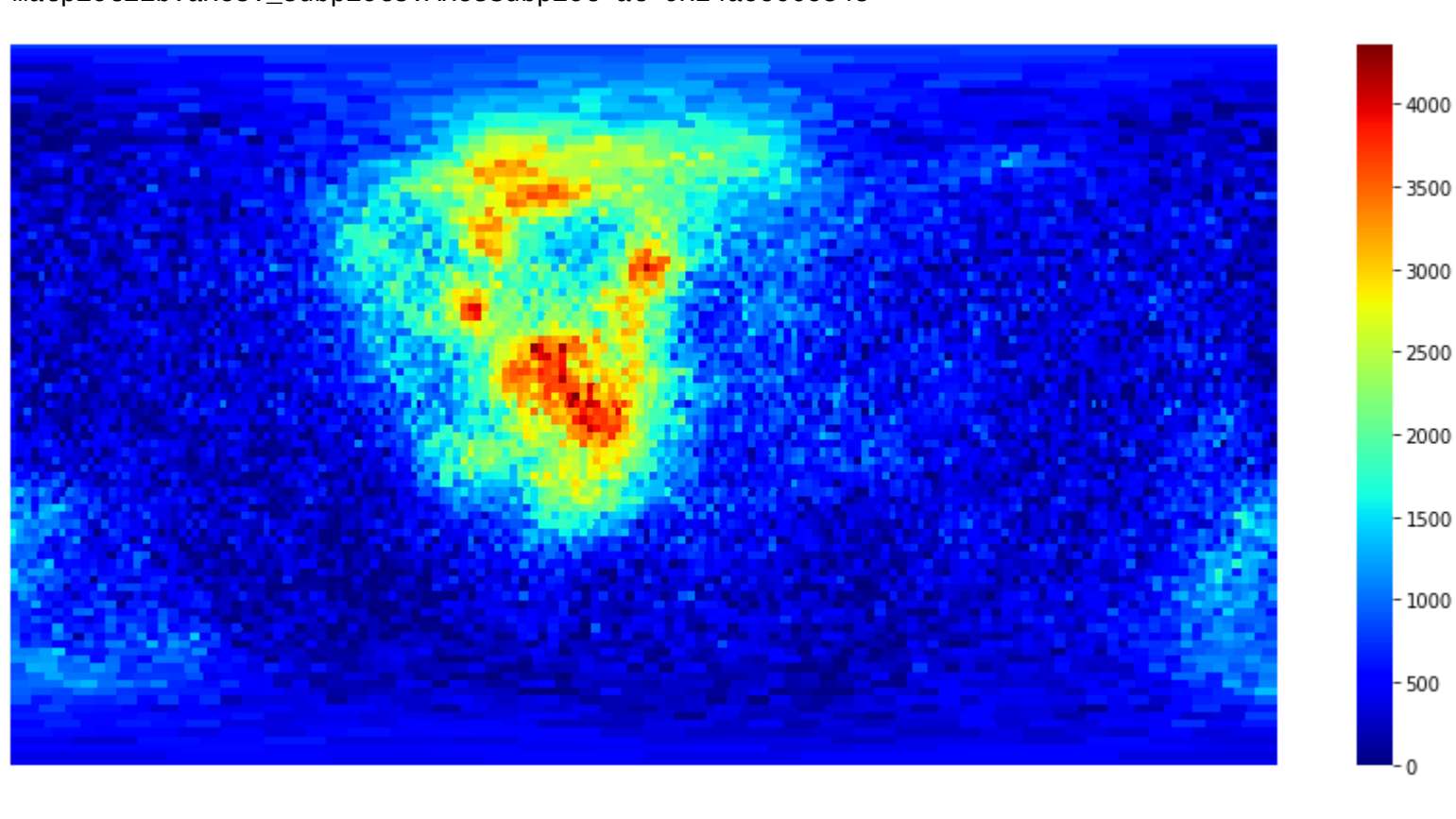
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x24aee758e48>



In [10]:
```python
K_map_data=pd.read_csv('..\ML4SCI_GSoC\Messenger\Moon\LPK_Map.csv',header=None)
```

In [11]:
```python
plt.subplots(figsize=(15,7))
sns.heatmap(K_map_data,xticklabels=False,yticklabels=False,cmap='jet')
```

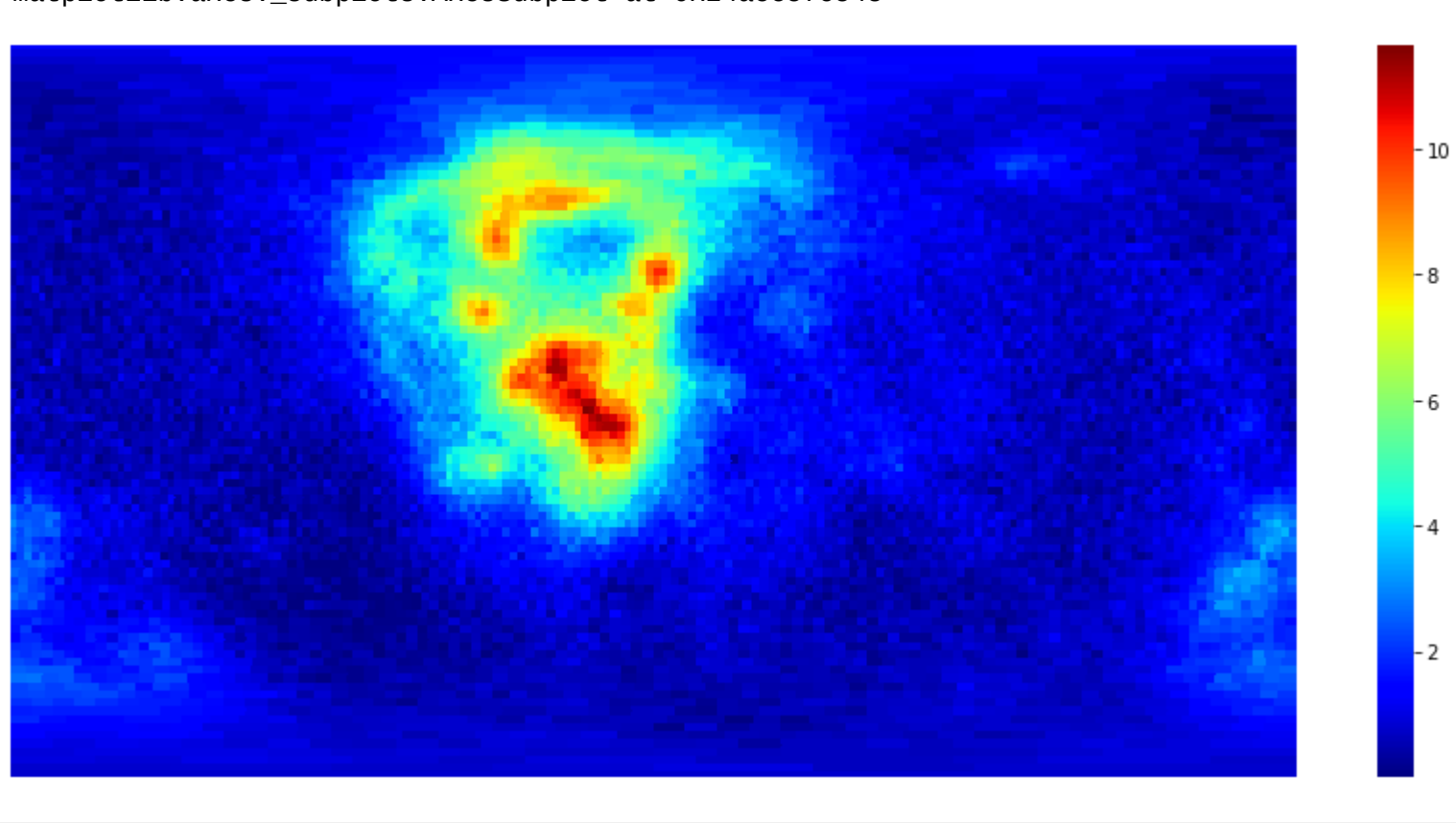
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x24aee66c848>



In [12]:
```python
Th_map_data=pd.read_csv('..\ML4SCI_GSoC\Messenger\Moon\LPTh_Map.csv',header=None)
```

In [13]:
```python
plt.subplots(figsize=(15,7))
sns.heatmap(Th_map_data,xticklabels=False,yticklabels=False,cmap='jet')
```

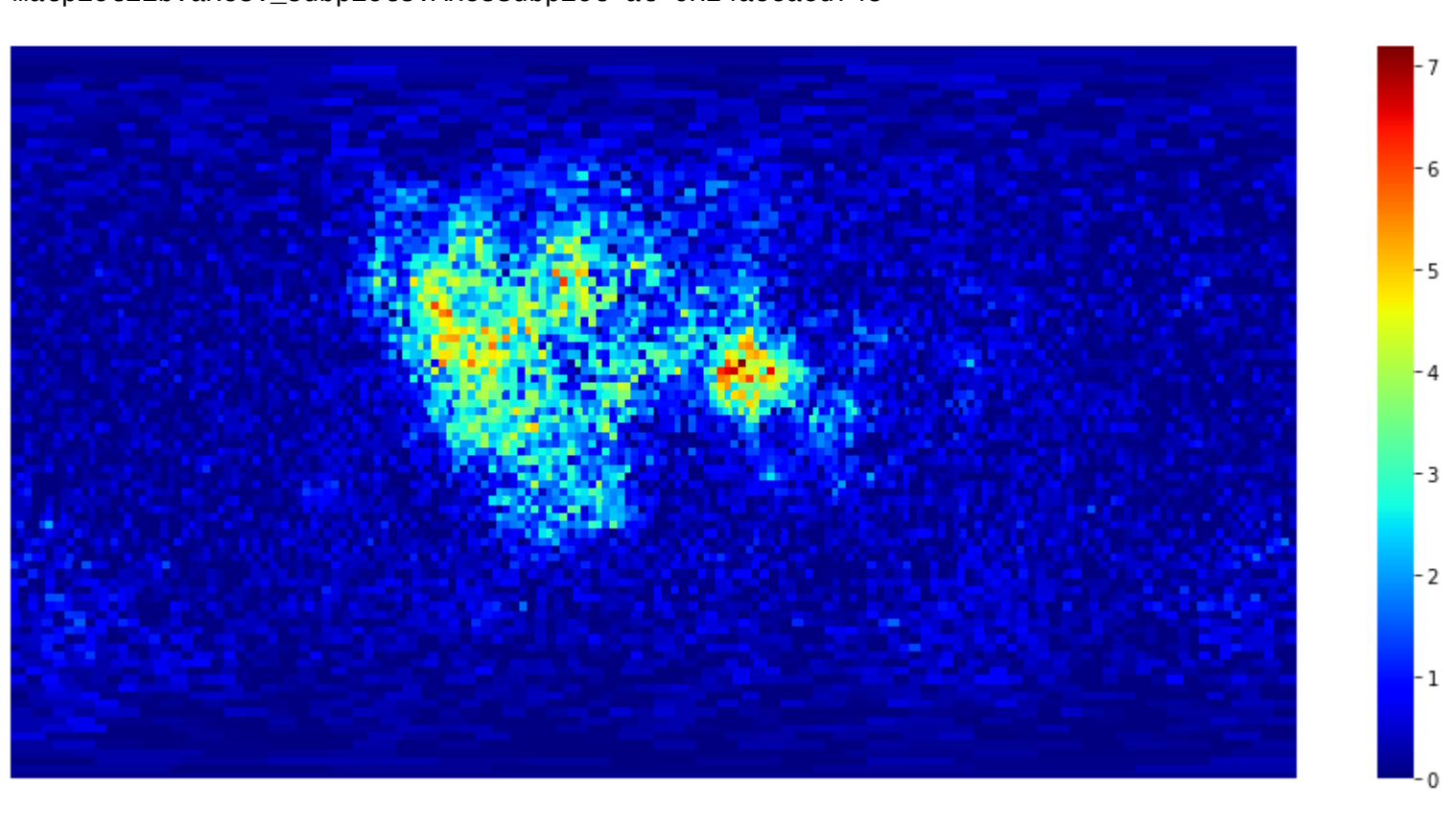
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x24aee57c548>



In [14]:
```python
Ti_map_data=pd.read_csv('..\ML4SCI_GSoC\Messenger\Moon\LPTi_Map.csv',header=None)
```

In [15]:
```python
plt.subplots(figsize=(15,7))
sns.heatmap(Ti_map_data,xticklabels=False,yticklabels=False,cmap='jet')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x24aeeaed748>



data preprocessing

In [16]:
```python
# now splitting the label dataset into training set and test set........
```

In [17]:
```python
#function for data splitting:
def data_splitter(data):
    F1=data.iloc[:,0:360].values
    F2=data.iloc[:,360:720].values
    return F1,F2
```

In [18]:
```python
Y_label_train,Y_label_test=data_splitter(albedo_map_data)
```

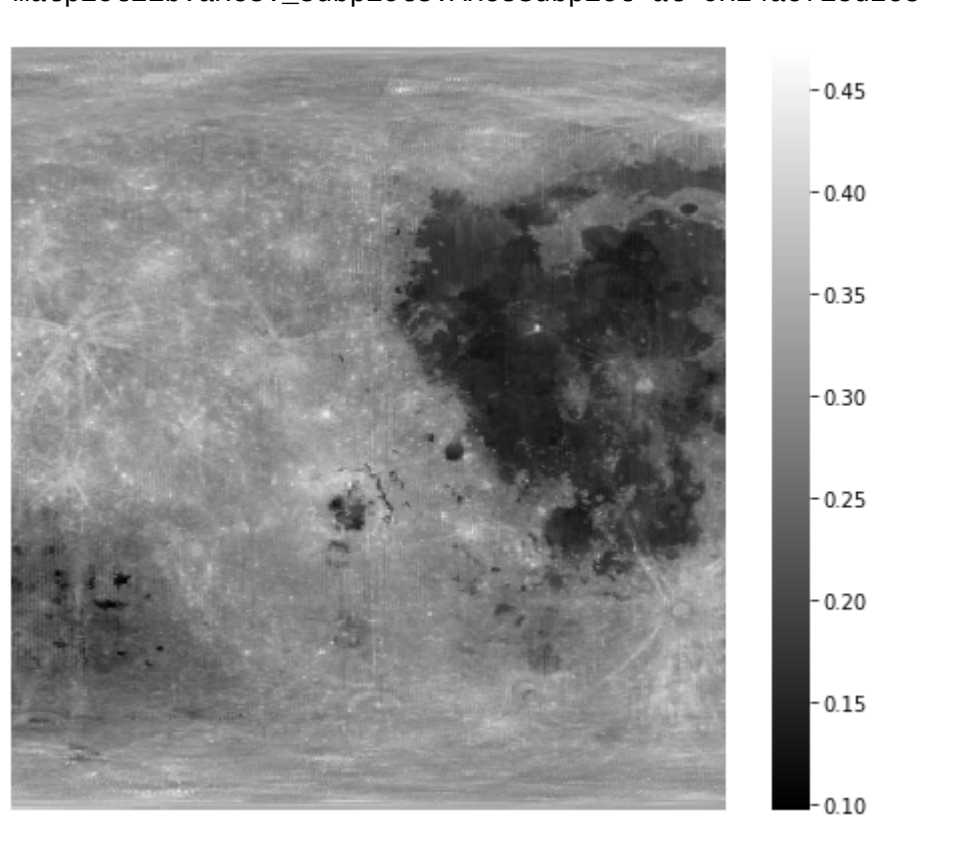
In [19]:
```python
Y_label_train.shape
```

Out[19]: (360, 360)

In [20]:
```python
# checking that dataset splitting correctly works by plotting graphs
```

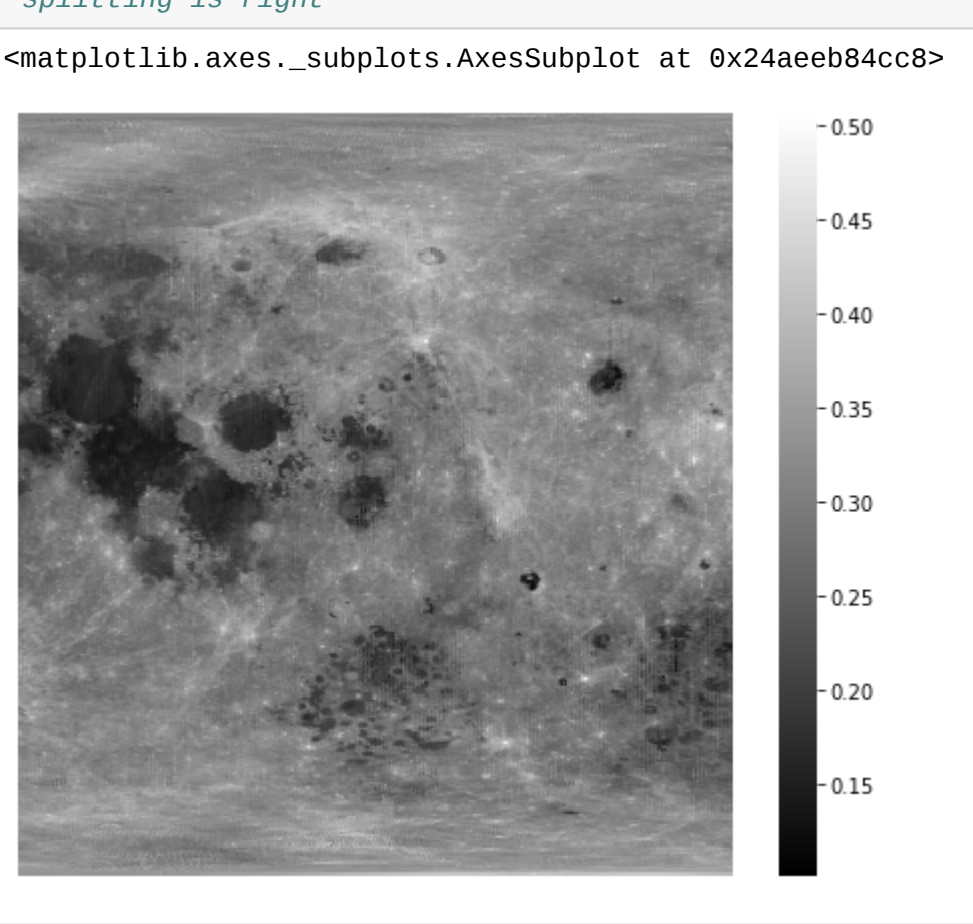
In [21]:
```python
plt.subplots(figsize=(8,7))
sns.heatmap(Y_label_train,xticklabels=False,yticklabels=False,cmap='gray') #just checking, i
f splitting is right.
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24aef15d2c8>



In [22]:
```python
plt.subplots(figsize=(8,7))
sns.heatmap(Y_label_test,xticklabels=False,yticklabels=False,cmap='gray') #just checking if
 splitting is right
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x24aeeb84cc8>



In [23]:
```python
Y_label_train.shape
```

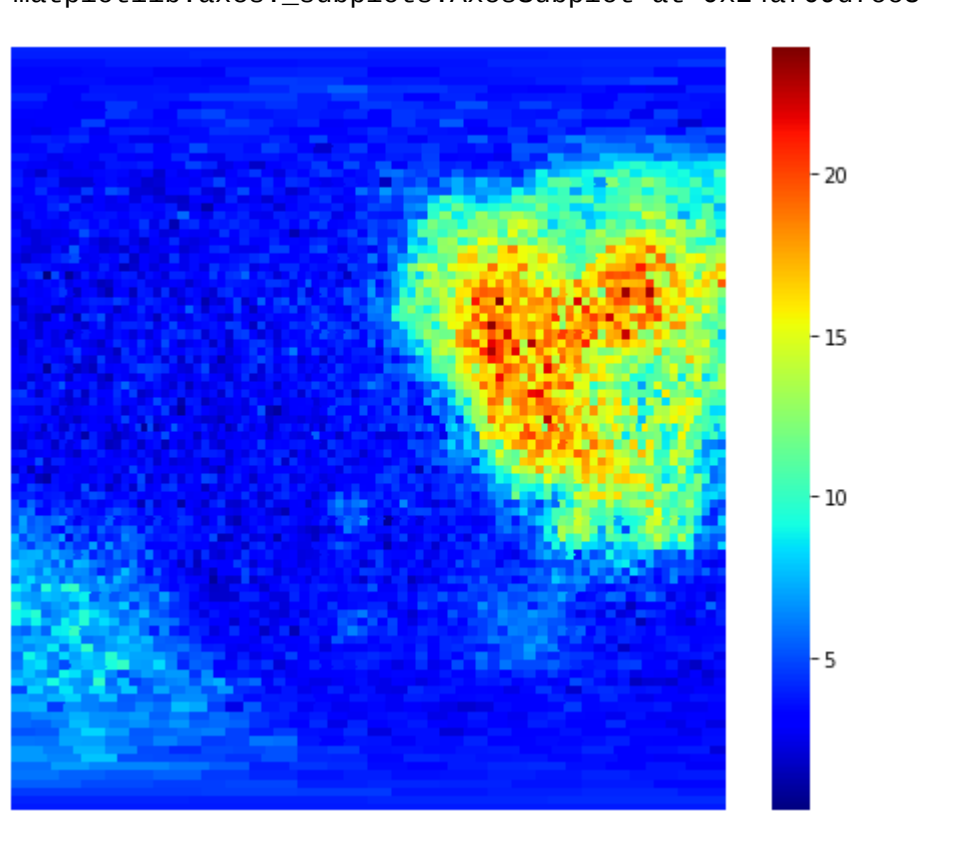
Out[23]: (360, 360)

In [24]:
```python
# now splitting feature training and test sets and converting in numpy arra
y....................
```

In [25]:
```python
Fe_features_train,Fe_features_test=data_splitter(Fe_map_data)
```

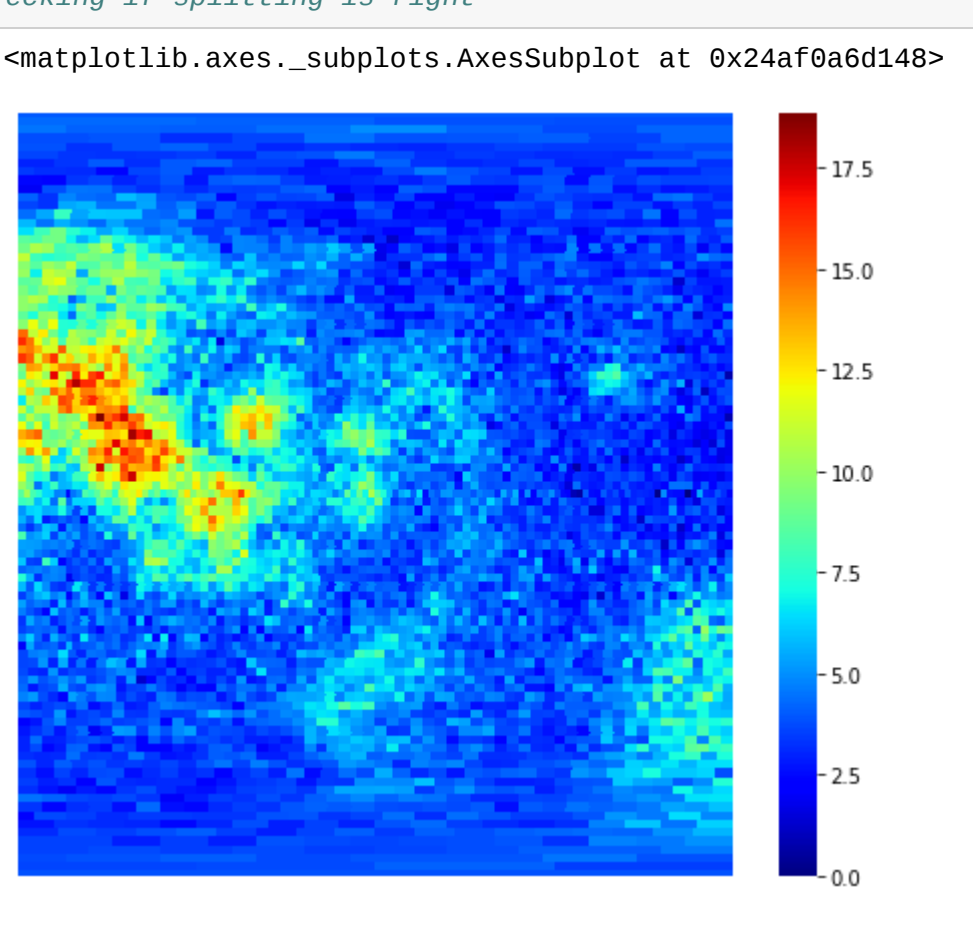
In [26]:
```python
plt.subplots(figsize=(8,7))
sns.heatmap(Fe_features_train,xticklabels=False,yticklabels=False,cmap='jet')   #just checki
ng if splitting is right
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x24af09d7cc8>



In [27]:
```python
plt.subplots(figsize=(8,7))
sns.heatmap(Fe_features_test,xticklabels=False,yticklabels=False,cmap='jet')        #just ch
ecking if splitting is right
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x24af0a6d148>



In [28]:
```python
K_features_train,K_features_test=data_splitter(K_map_data)
```

In [29]:
```python
Th_features_train,Th_features_test=data_splitter(Th_map_data)
```

In [30]:
```python
Ti_features_train,Ti_features_test=data_splitter(Ti_map_data)
```

In [31]:
```python
#  now, in order to make a feature set, we have to merge all the different chemical datasets
into one with proper positon
# so that we can use them to train and predict.
```

In [32]:
```python
# way of merging different chemical datasets: first we will convert the all chemical dataset
into a column vector after that
# we will merge all column vector column-wise.
# for label data set, we will also convert it into column vector.
```

In [33]:
```python
def column_converter(data):
    column_arr=data.reshape(-1,1,order='F') # here order='F' specifies that it will read col
umn vice
    return column_arr
```

In [34]:
```python
Fe_column_train=column_converter(Fe_features_train)
```

In [35]:
```python
Fe_column_train.shape
```

Out[35]: (129600, 1)

In [36]:
```python
K_column_train=column_converter(K_features_train)

Th_column_train=column_converter(Th_features_train)

Ti_column_train=column_converter(Ti_features_train)
```

In [37]:
```python
# now all the data is converted to column vector now we will add them.
```

In [38]:
```python
X_train=np.concatenate((Fe_column_train,K_column_train,Th_column_train,Ti_column_train),axis
=1)
```

In [39]:
```python
X_train.shape
```

Out[39]: (129600, 4)

In [40]:
```python
# now we have our feature set ready,now converting label set.
```

In [41]:
```python
Y_train=column_converter(Y_label_train)
```

In [42]:
```python
Y_train.shape
```

Out[42]: (129600, 1)

In [43]:
```python
# I have applied many feature engineering but none of them giving good result and getting ev
en worse, so no feature
#engineering on dataset.
```

In [44]:
```python
# seeing corelation between target and featureset.
comb_data=np.concatenate((Y_train,X_train),axis=1)
kp=pd.DataFrame(comb_data)
p=kp.corr()
p[0]
```

Out[44]:
```
0    1.000000
1   -0.851169
2   -0.644436
3   -0.669347
4   -0.747507
Name: 0, dtype: float64
```

Model Training.

In [45]:
```python
Lin_model=LinearRegression().fit(X_train,Y_train)
```

In [46]:
```python
DT_model=DecisionTreeRegressor().fit(X_train,Y_train)
```

In [47]:
```python
RF_model=RandomForestRegressor().fit(X_train,Y_train.ravel())
```

In [48]:
```python
Rd_model=Ridge().fit(X_train,Y_train)
```

checking the model credibility via cross velidation.

In [50]:
```python
from sklearn.model_selection import cross_val_score
```

In [58]:
```python
# function for cross validation score.
def crossval_score(model,X,Y):
    score=cross_val_score(model,X,Y,scoring='r2')
    return score.mean(),score.std()
```

In [59]:
```python
#dataset for cross validation.
X_cval=X_train[0:150,:]
Y_cval=Y_train[0:150,:]
```

In [63]:
```python
# Linear reg cross validation score
cv_mean,cv_std=crossval_score(Lin_model,X_cval,Y_cval)
print("mean:linear reg model:",cv_mean)
print("std:Linear reg model:",cv_std)
```

```
mean:linear reg model: -1.3212461763753953
std:Linear reg model: 1.5793644255124564
```

In [64]:
```python
# ridge cross validation score
cv2_mean,cv2_std=crossval_score(Rd_model,X_cval,Y_cval)
print("mean:ridge model:",cv2_mean)
print("std:ridge model:",cv2_std)
```

```
mean:ridge model: -1.281559334800153
std:ridge model: 1.4964201579096719
```

In [65]:
```python
# decision tree cross validation score
cv2_mean,cv2_std=crossval_score(DT_model,X_cval,Y_cval)
print("mean:decision tree model:",cv2_mean)
print("std:decision tree model:",cv2_std)
```

```
mean:decision tree model: -4.172437944110979
std:decision tree model: 1.4253038661884279
```

In [66]:
```python
# random forest cross validation score
cv2_mean,cv2_std=crossval_score(RF_model,X_cval,Y_cval)
print("mean:decision tree model:",cv2_mean)
print("std:decision tree model:",cv2_std)
```

```
C:\Users\NIRBHAY MAURYA\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:51
5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\NIRBHAY MAURYA\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:51
5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\NIRBHAY MAURYA\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:51
5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\NIRBHAY MAURYA\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:51
5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\NIRBHAY MAURYA\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:51
5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please c
hange the shape of y to (n_samples,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

```
mean:decision tree model: -2.5448447059618475
std:decision tree model: 1.6098947907762067
```

In [67]:
```python
 # so, ridge model is performing best out of all algorithms.
```

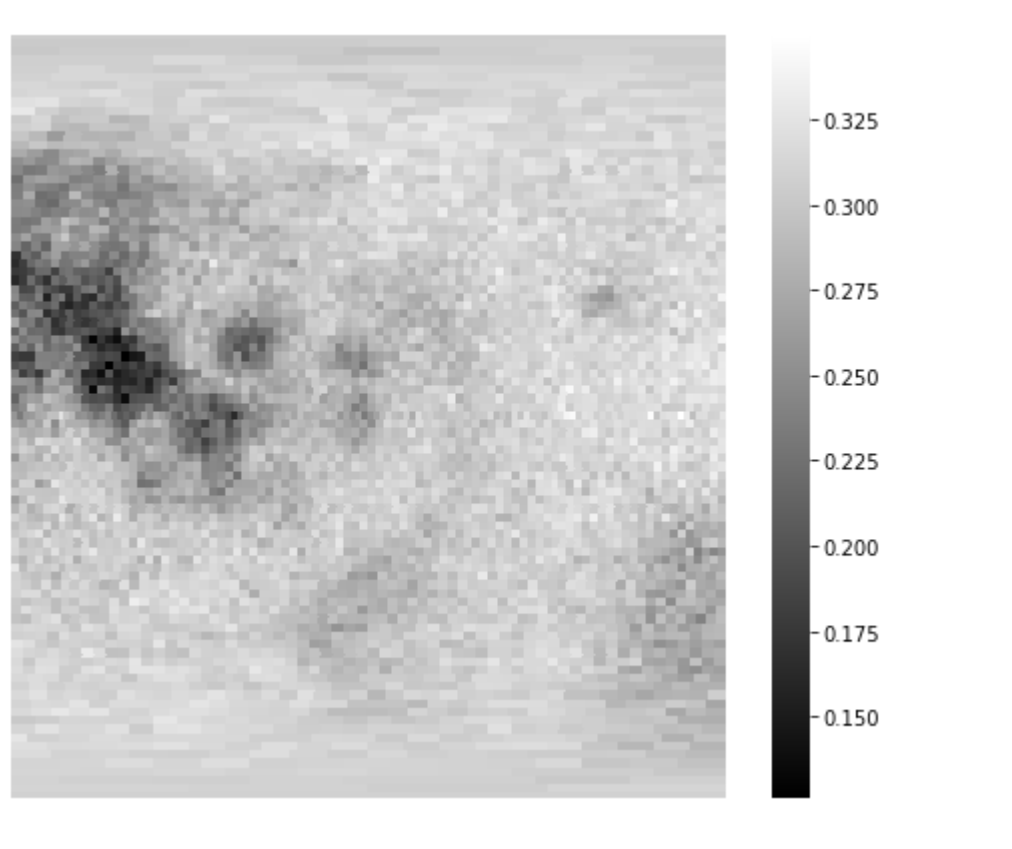
processing testing data.

In [68]:
```python
Fe_column_test=column_converter(Fe_features_test)

K_column_test=column_converter(K_features_test)

Th_column_test=column_converter(Th_features_test)

Ti_column_test=column_converter(Ti_features_test)

X_test=np.concatenate((Fe_column_test,K_column_test,Th_column_test,Ti_column_test),axis=1)
```

In [69]:
```python
predicted_albedo_data=Rd_model.predict(X_test)
```

In [70]:
```python
# converting to dimentions and plotting right side predicted albedo.
right_albedo_data=predicted_albedo_data.reshape(360,-1,order='F')
plt.subplots(figsize=(8,7))
sns.heatmap(right_albedo_data,xticklabels=False,yticklabels=False,cmap='gray')
```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x24af15cd208>



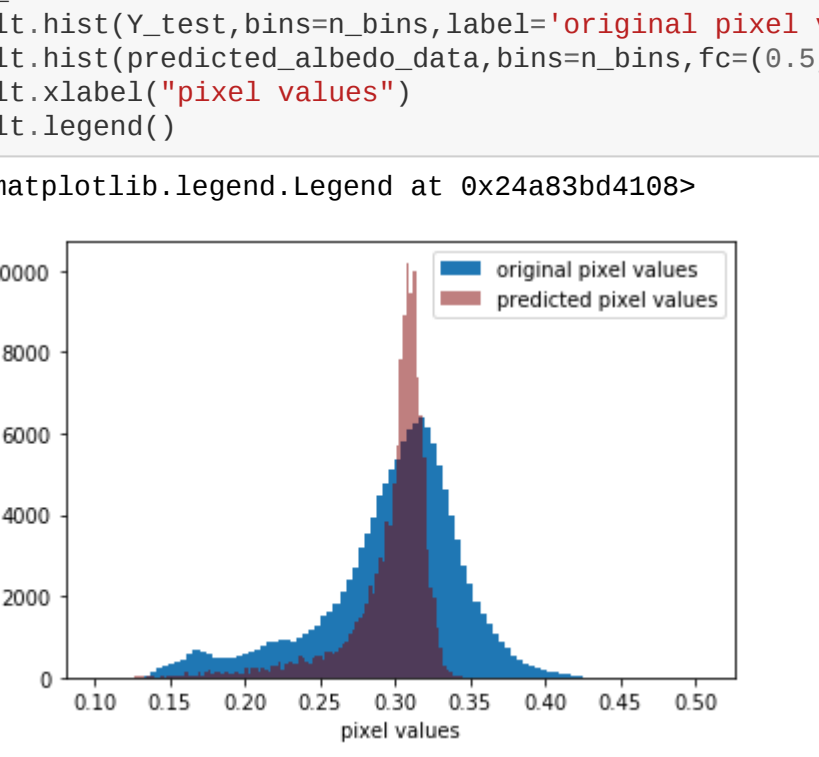
checking performance.

In [93]:
```python
# model accuracy.
original_right_albedo=Y_label_test
Y_test=column_converter(Y_label_test)
r2_score(original_right_albedo,right_albedo_data)
```

Out[93]: 0.38386097405240327

residual plotting

In [110]:
```python
# in 2-d
plt.xlabel('original pixel values')
plt.ylabel('predicted pixel values')
plt.title('residual plot')
plt.scatter(Y_test,predicted_albedo_data)
plt.plot(Y_test,Y_test,color='red')
plt.legend(['expected when accuracy is 100%'],loc="best")
plt.show()
```



In [109]:
```python
# histogram plot
n_bins=100
plt.hist(Y_test,bins=n_bins,label='original pixel values')
plt.hist(predicted_albedo_data,bins=n_bins,fc=(0.5,0,0,0.5),label='predicted pixel values')
plt.xlabel("pixel values")
plt.legend()
```

Out[109]: <matplotlib.legend.Legend at 0x24a83bd4108>

conclusion: basically in maximum of the cases(regression problem),RandomforestRegressor give better result than linear regression and In randomforest, there is high chances of high variance and the variance can be reduced by regularisation. so, as we used ridge(regularized algorithm), performance got better. so, here we can say that this model have high variance problem,so, getting more features in data accuracy of model can be increased....