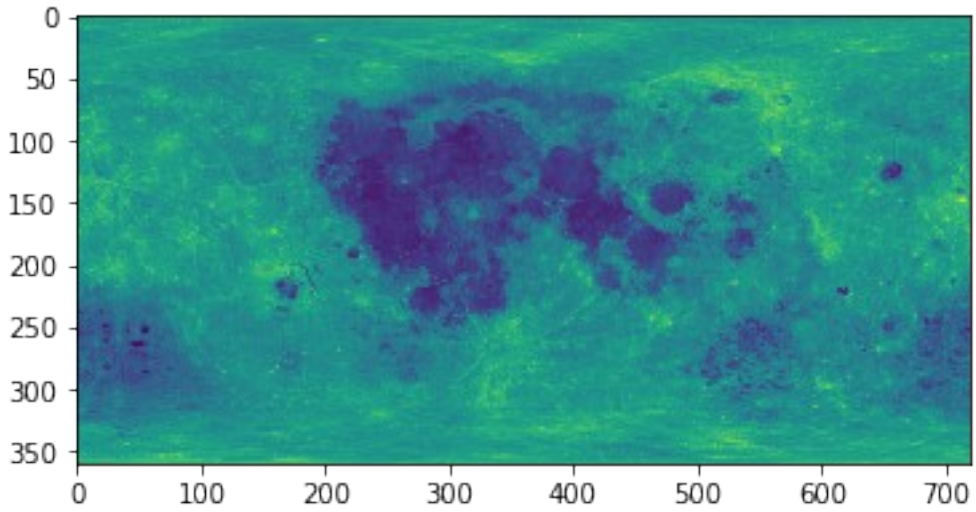
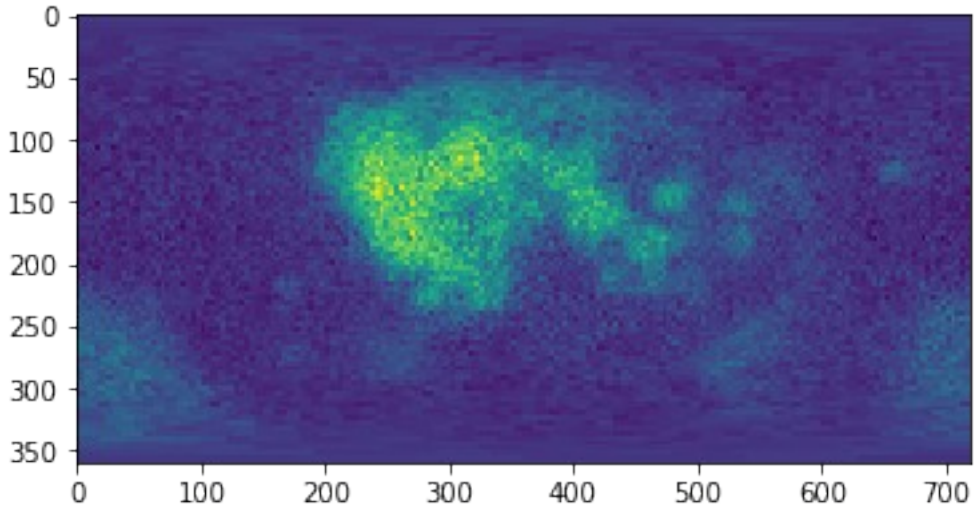
```
import pandas as pd
import numpy as np

albedo_map = pd.read_csv('Dataset/Moon/Albedo_Map.csv')
fe = pd.read_csv('Dataset/Moon/LPFe_Map.csv')
k = pd.read_csv('Dataset/Moon/LPK_Map.csv')
th = pd.read_csv('Dataset/Moon/LPTh_Map.csv')
ti = pd.read_csv('Dataset/Moon/LPTi_Map.csv')

%matplotlib inline
from matplotlib import pyplot as plt
```

Interpolation is done to reduce noise in images

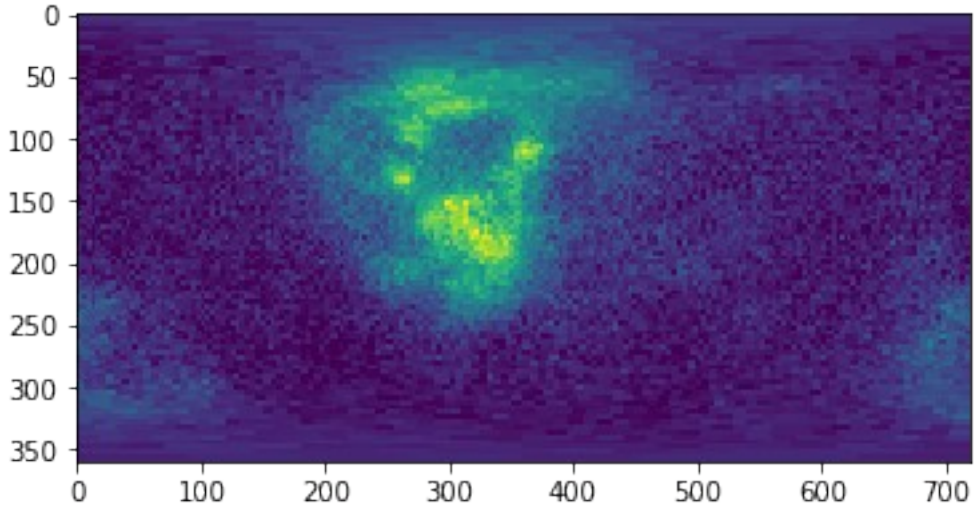
```
albedo_map_arr = albedo_map.to_numpy()
plt.imshow(albedo_map_arr, interpolation='none')
plt.show()
```



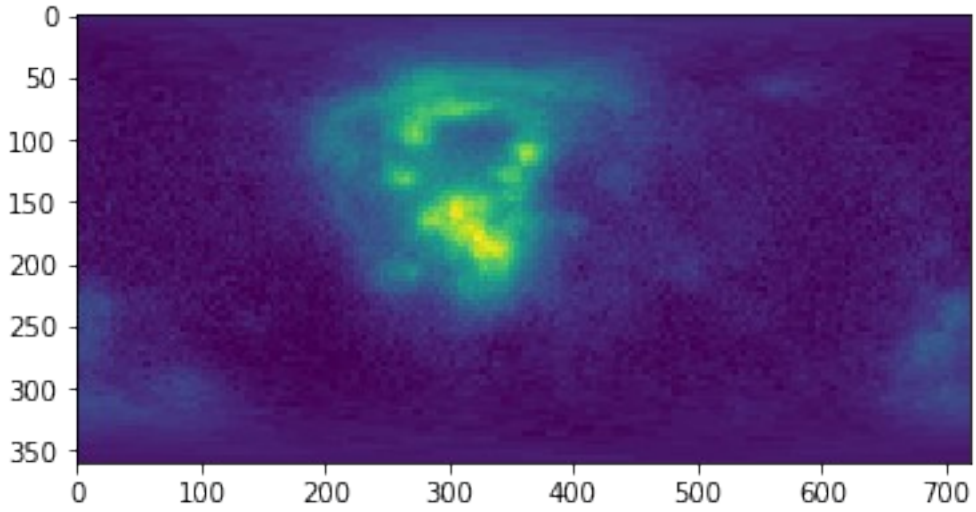

```
fe_arr = fe.to_numpy()
plt.imshow(fe_arr, interpolation='nearest')
plt.show()
```



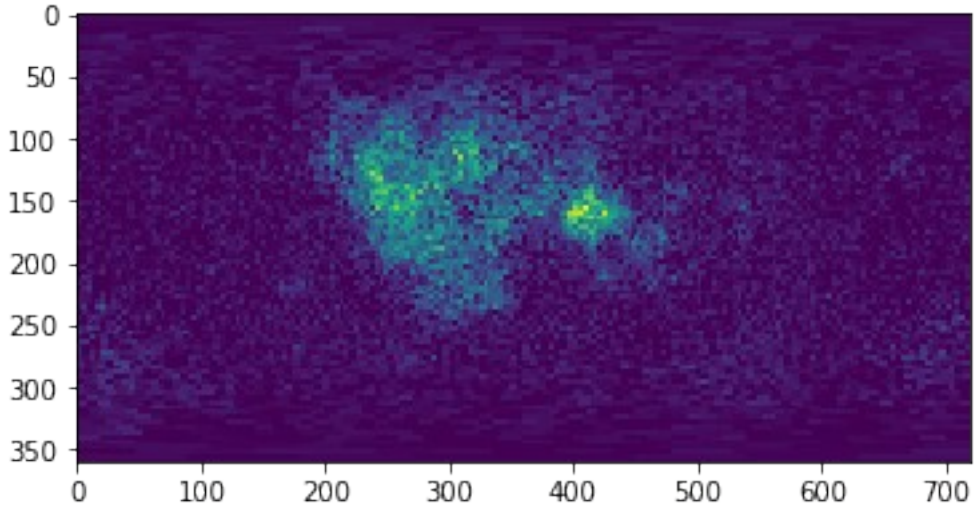

```
k_arr = k.to_numpy()
plt.imshow(k_arr, interpolation='nearest')
plt.show()
```




```
th_arr = th.to_numpy()
plt.imshow(th_arr, interpolation='nearest')
plt.show()
```




```
ti_arr = ti.to_numpy()
plt.imshow(ti_arr, interpolation='nearest')
plt.show()
```



Fe

```
x_train = fe.iloc[:,:int(fe.shape[1]/2)]
x_test = fe.iloc[:,int(fe.shape[1]/2):]
y_train = albedo_map.iloc[:,:int(albedo_map.shape[1]/2)]
y_test = albedo_map.iloc[:,int(albedo_map.shape[1]/2):]

from xgboost import XGBRegressor
from sklearn.multioutput import MultiOutputRegressor
regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

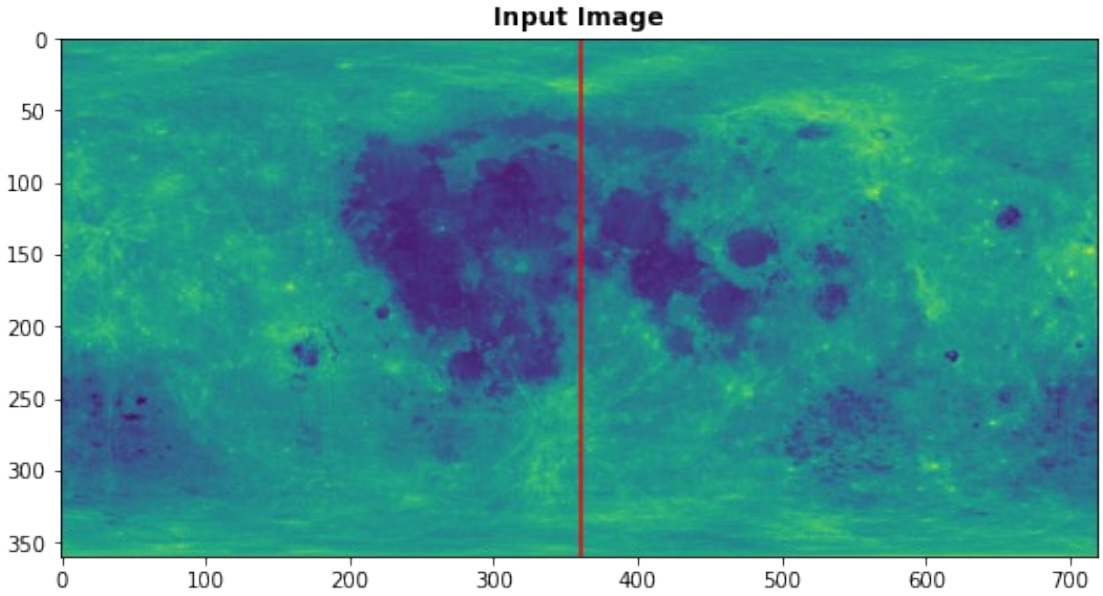
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

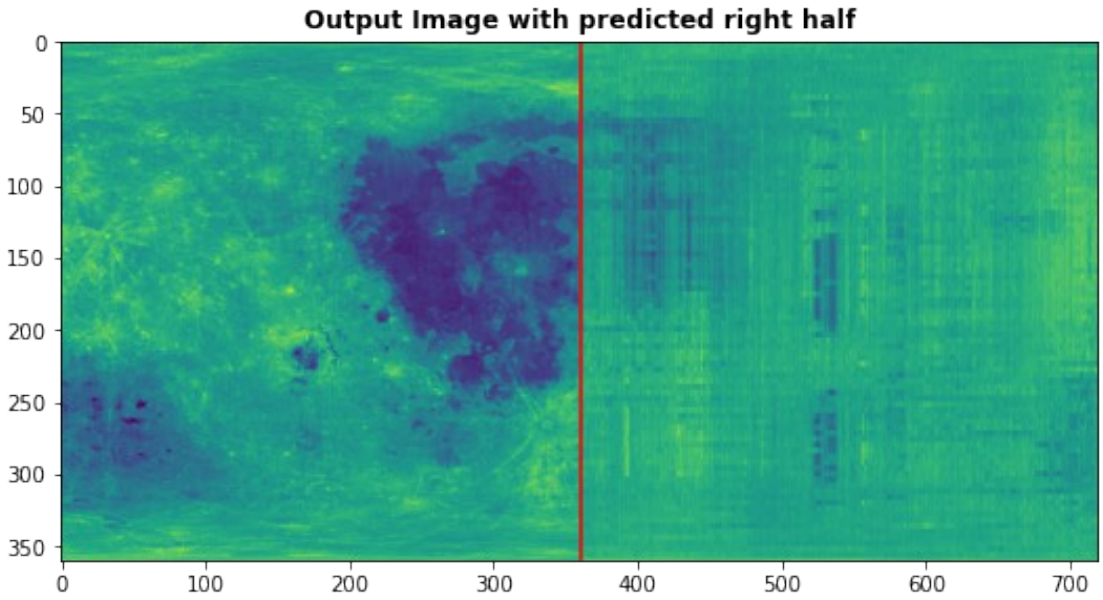
```
0.0022739482971186227
```

Output predicted comparisions

```
column = []
for i in range(361,721):
    column.append(i)

y_pred = pd.DataFrame(y_pred, columns=column)
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 1)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axvline(x=360, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axvline(x=360, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted right half', fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted right half')
```

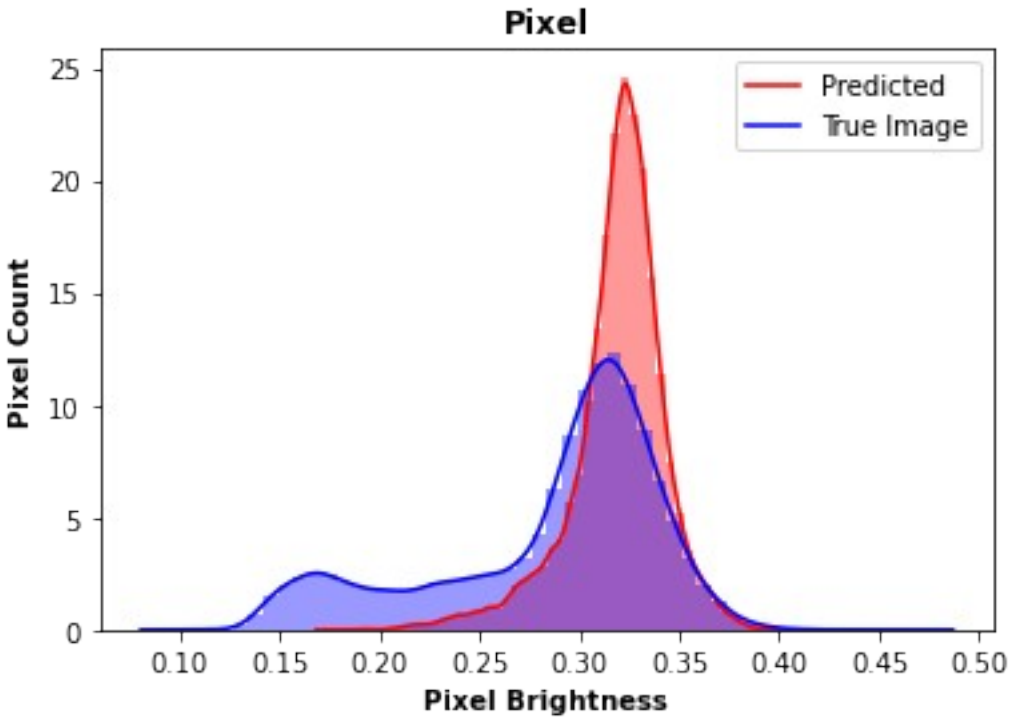




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x12fe18640>
```



K

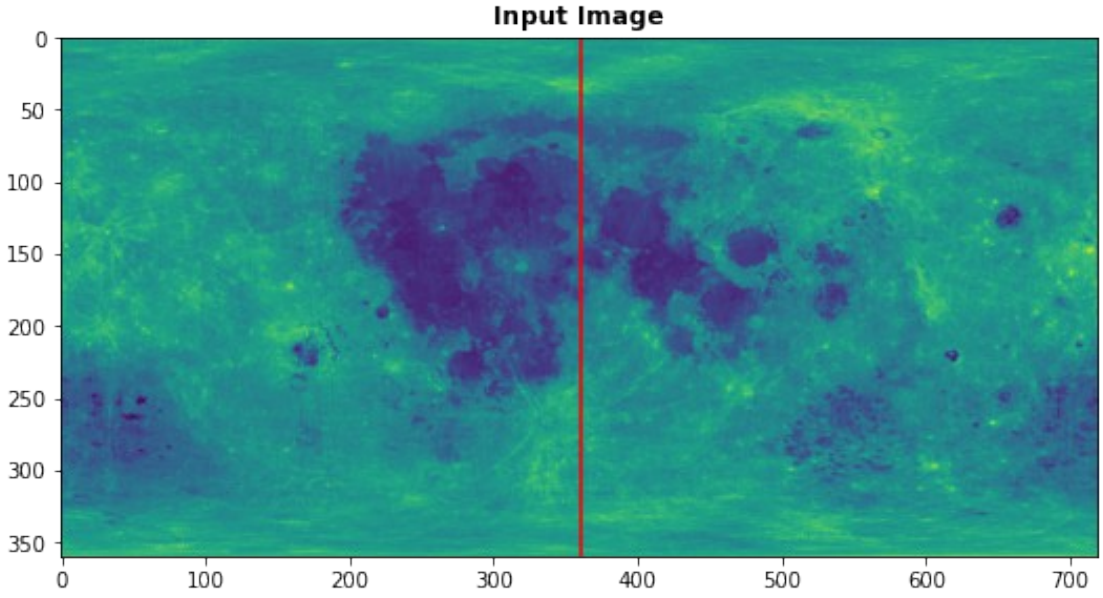
```
x_train = k.iloc[:,:int(k.shape[1]/2)]
x_test = k.iloc[:,int(k.shape[1]/2):]
y_train = albedo_map.iloc[:,:int(albedo_map.shape[1]/2)]
y_test = albedo_map.iloc[:,int(albedo_map.shape[1]/2):]

regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)

0.002342920007568432
```

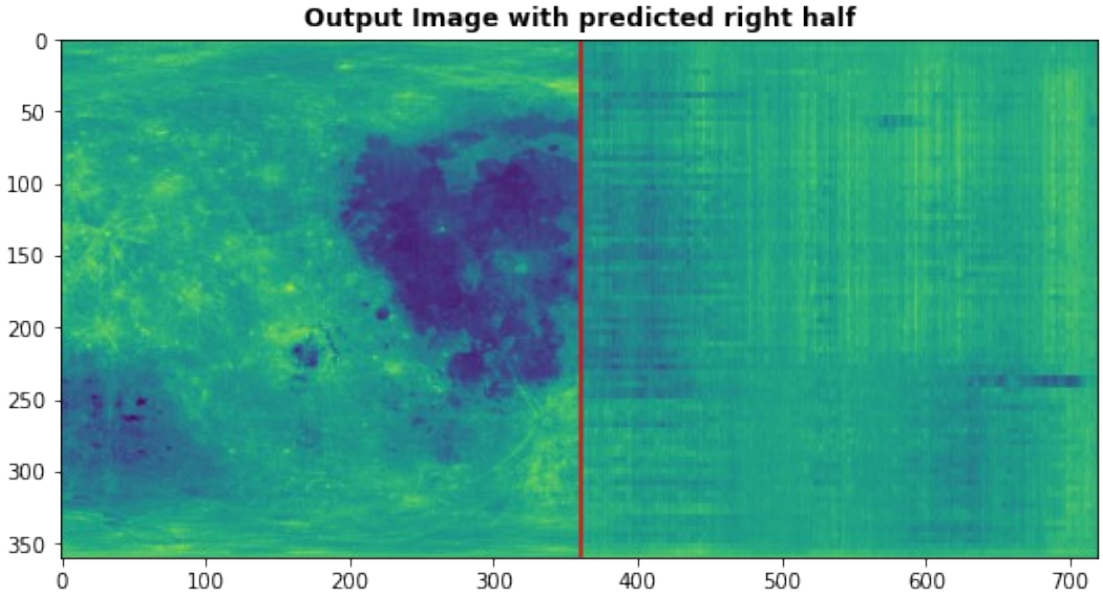
Output predicted comparisions

```
column = []
for i in range(361,721):
    column.append(i)

y_pred = pd.DataFrame(y_pred, columns=column)
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 1)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
```

```
plt.axvline(x=360, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axvline(x=360, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted right half', fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted right half')
```



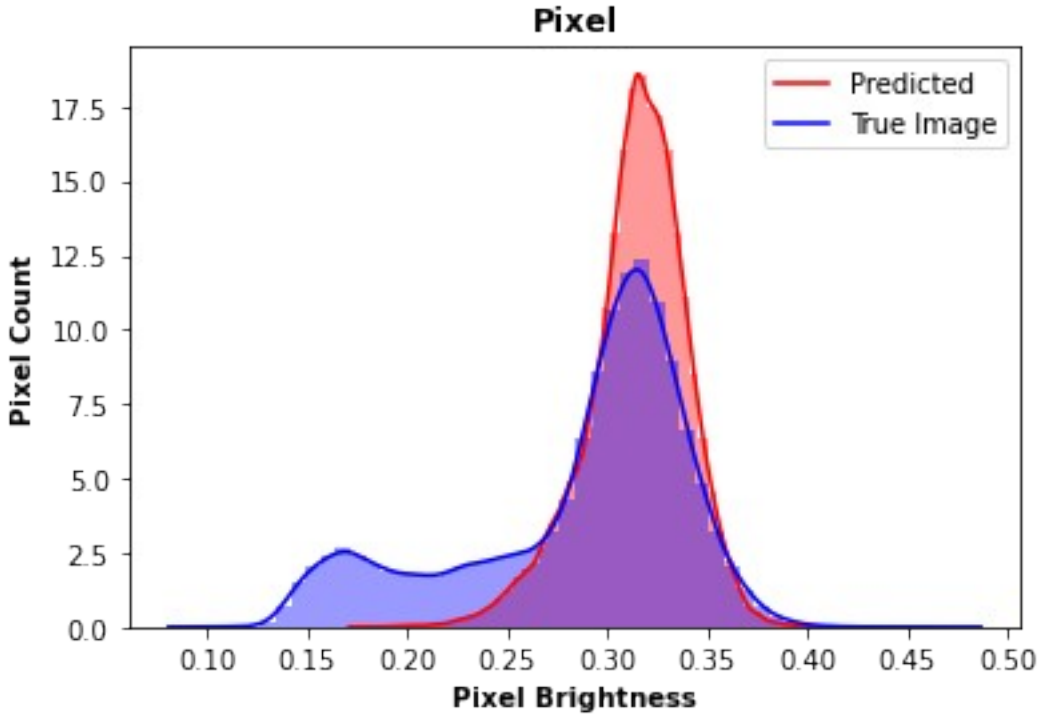


Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.legend.Legend at 0x12e025880>
```



Th

```
x_train = th.iloc[:,:int(th.shape[1]/2)]
x_test = th.iloc[:,int(th.shape[1]/2):]
```

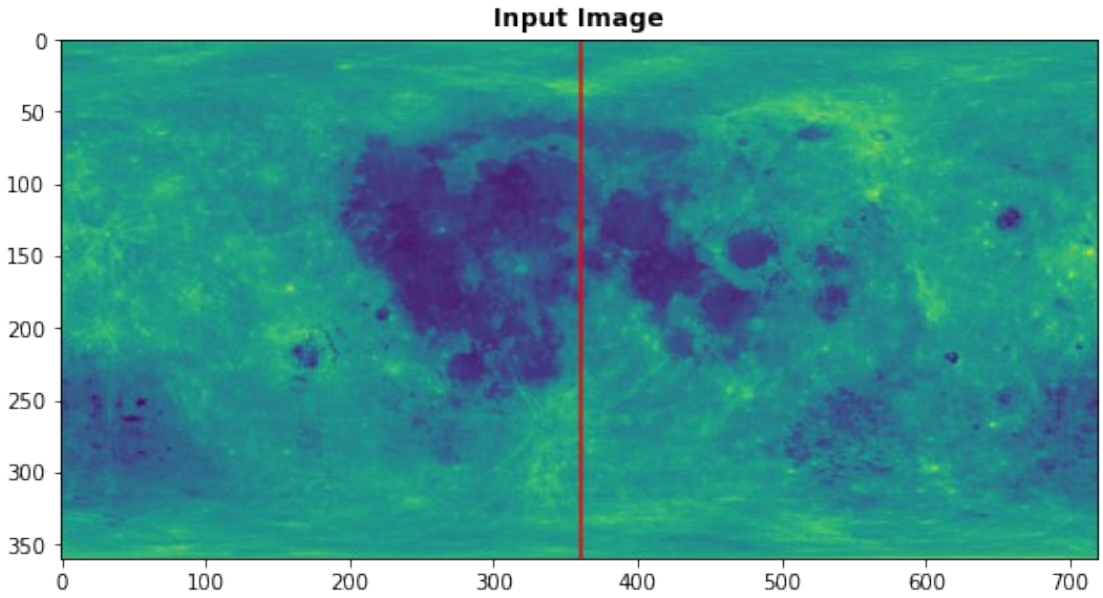
```
y_train = albedo_map.iloc[:,:int(albedo_map.shape[1]/2)]
y_test = albedo_map.iloc[:,int(albedo_map.shape[1]/2):]

regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

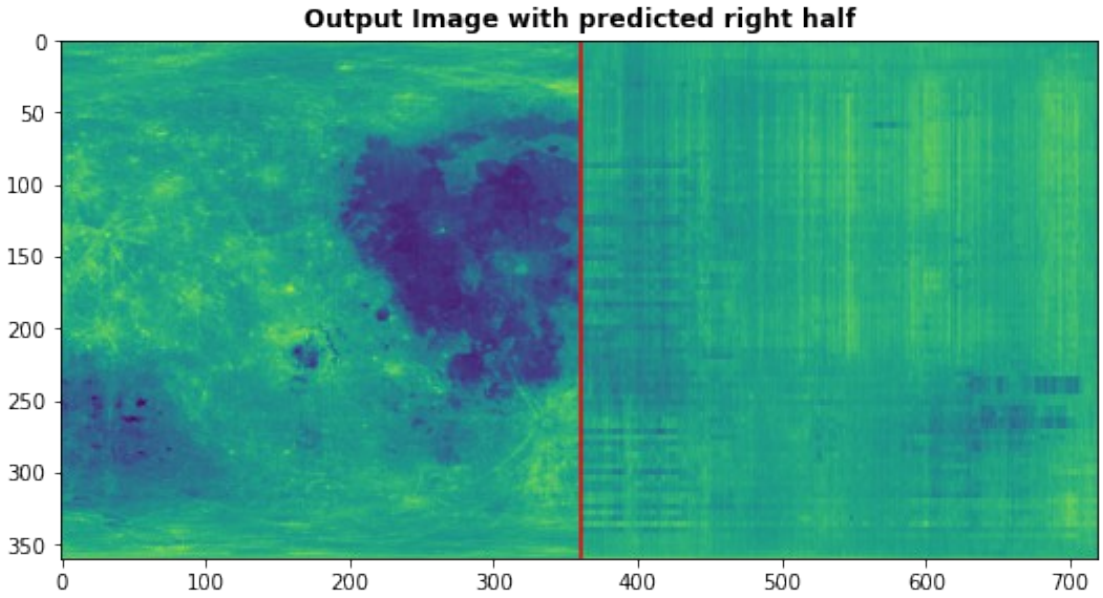
```
0.002606791837194686
```

Output predicted comparisions

```
column = []
for i in range(361,721):
    column.append(i)

y_pred = pd.DataFrame(y_pred, columns=column)
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 1)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axvline(x=360, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axvline(x=360, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted right half', fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted right half')
```



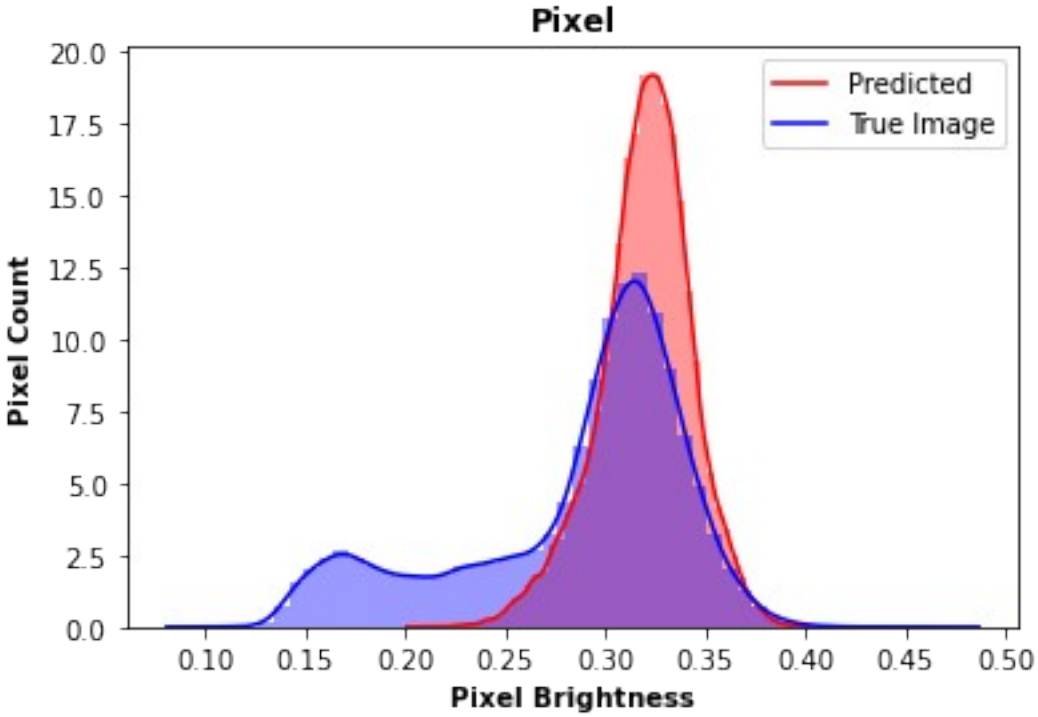


Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
```

```
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x131fd74c0>
```



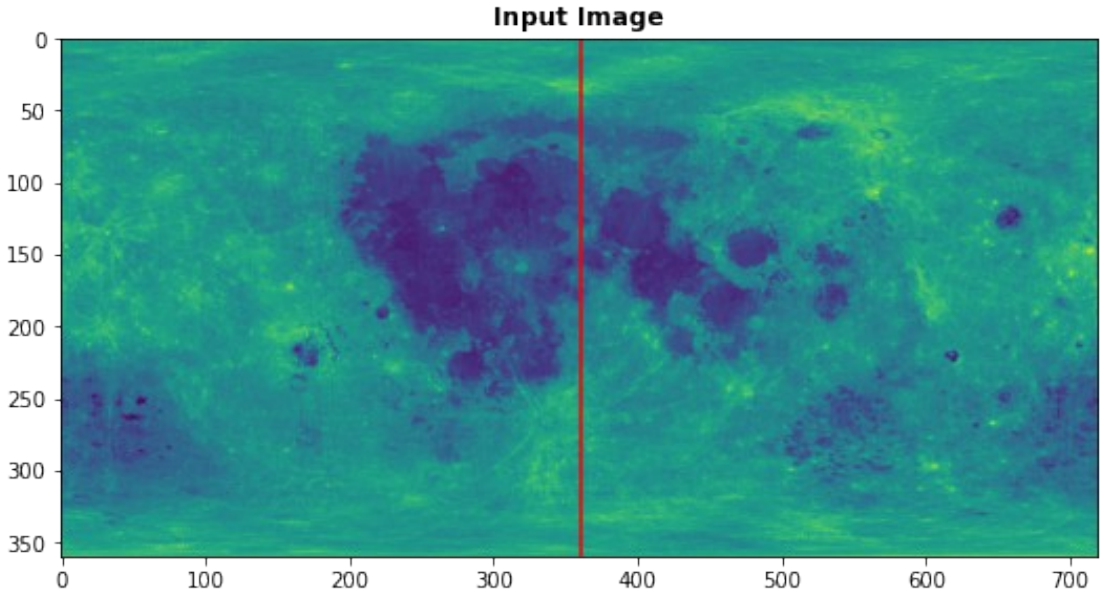
Ti

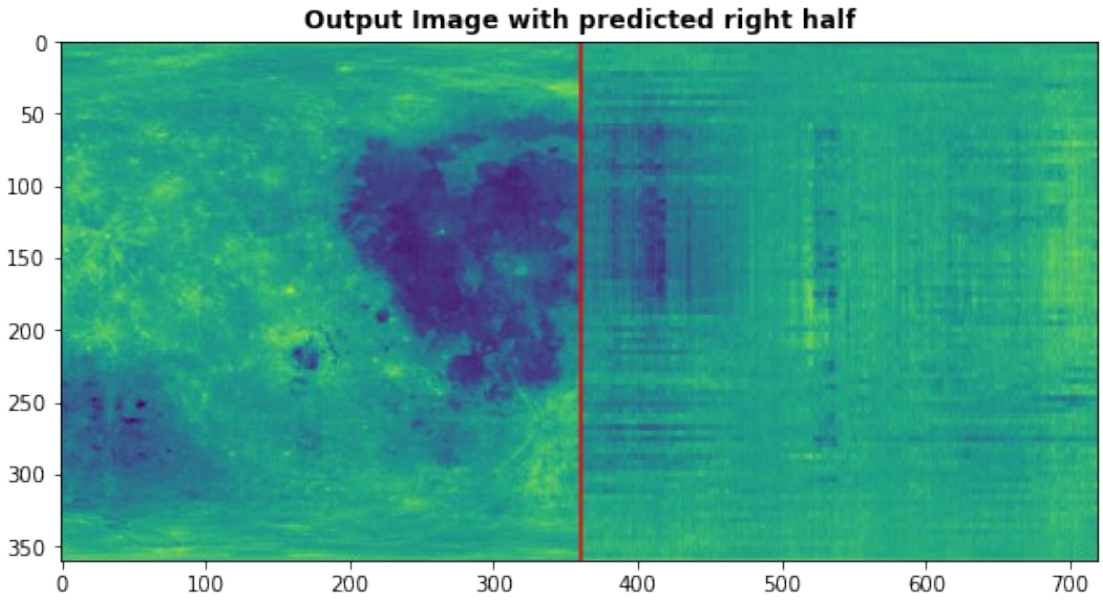
```
x_train = ti.iloc[:,:int(ti.shape[1]/2)]
x_test = ti.iloc[:,int(ti.shape[1]/2):]
y_train = albedo_map.iloc[:,:int(albedo_map.shape[1]/2)]
y_test = albedo_map.iloc[:,int(albedo_map.shape[1]/2):]

regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)

0.0019408721516637713
```

Output predicted comparisions

```
column = []
for i in range(361,721):
    column.append(i)

y_pred = pd.DataFrame(y_pred, columns=column)
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 1)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axvline(x=360, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axvline(x=360, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted right half', fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted right half')
```

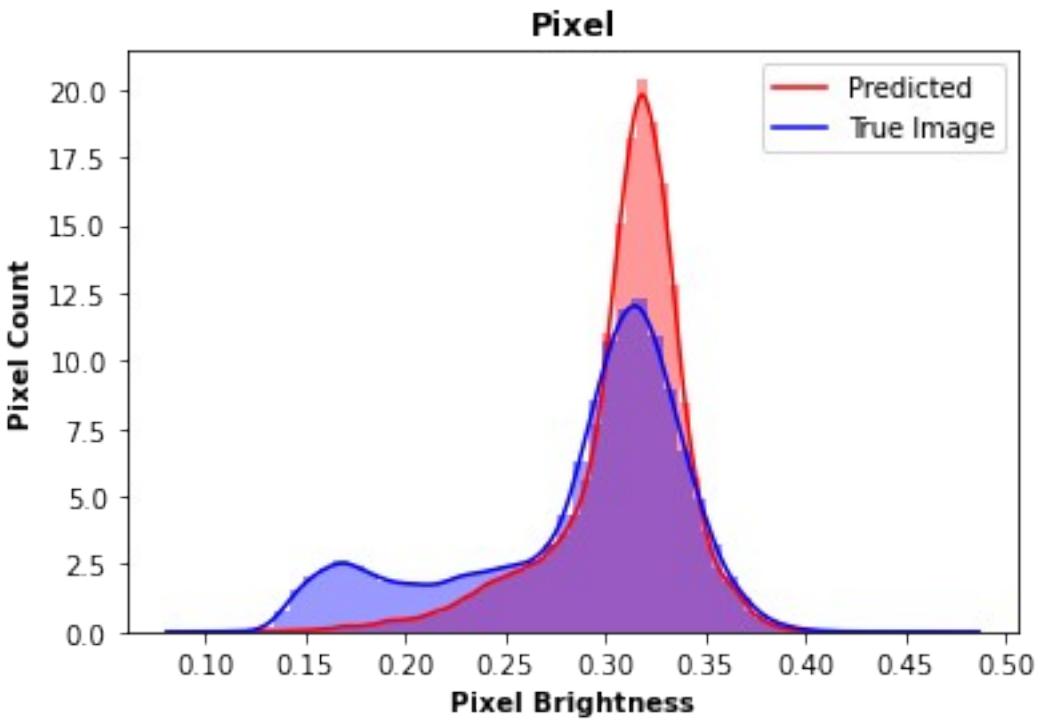




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x131f4ed60>
```



Other Ratios

Dividing data in other ratios for better results

We are dividing for Fe dataset

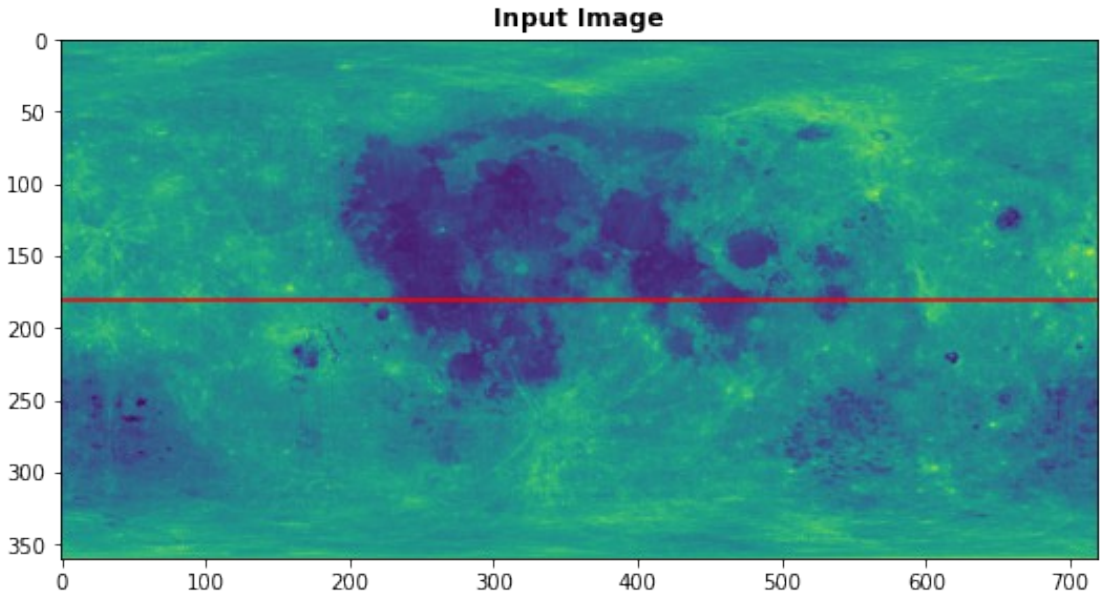
Horizontal

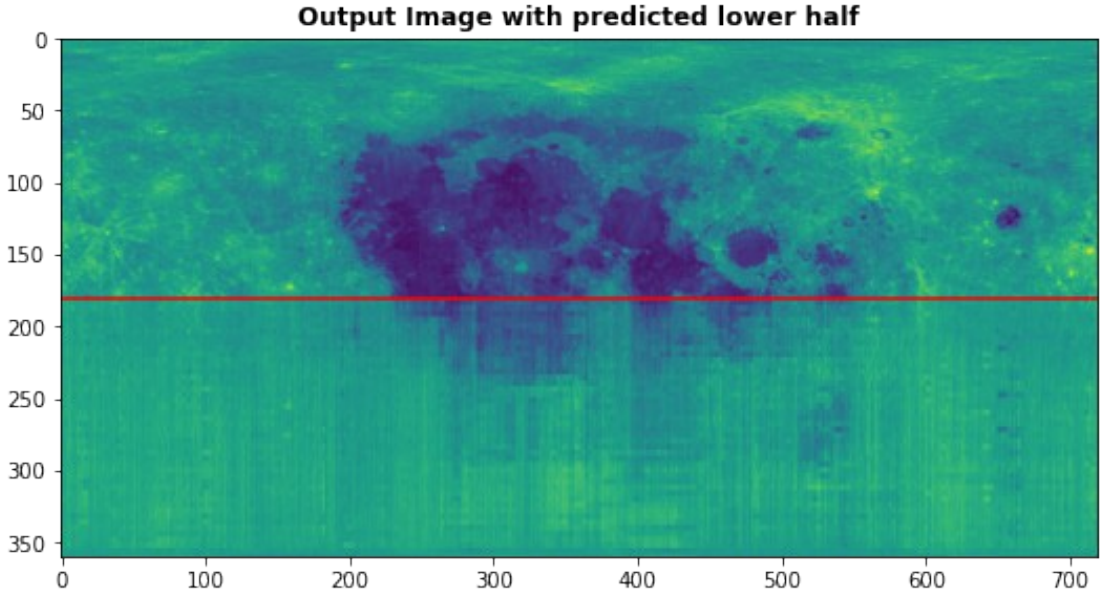
```
x_train = fe.iloc[:int(fe.shape[0]/2),]
x_test = fe.iloc[int(fe.shape[0]/2):,]
y_train = albedo_map.iloc[:int(albedo_map.shape[0]/2),]
y_test = albedo_map.iloc[int(albedo_map.shape[0]/2):,]

from xgboost import XGBRegressor
from sklearn.multioutput import MultiOutputRegressor
regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)

MultiOutputRegressor(estimator=XGBRegressor(base_score=None,
booster=None,
                                            colsample_bylevel=None,
                                            colsample_bynode=None,
                                            colsample_bytree=None,
                                            enable_categorical=False,
eta=0.1,
                                            eval_metric='rmse',
gamma=None,
```

```
                                                gpu_id=None,
importance_type=None,

interaction_constraints=None,
                                                learning_rate=None,
                                                max_delta_step=None,
max_depth=None,
                                                min_child_weight=None,
missing=nan,
                                                monotone_constraints=None,
                                                n_estimators=150,
n_jobs=None,
                                                num_parallel_tree=None,
                                                predictor=None,
random_state=0,
                                                reg_alpha=None,
reg_lambda=None,
                                                scale_pos_weight=None,
                                                subsample=0.6,
tree_method=None,
                                                validate_parameters=None,
                                                verbosity=None))

y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)

0.0028835320951995365

column = []
for i in range(1,721):
    column.append(i)

index = []
for i in range(180,360):
    index.append(i)

y_pred = pd.DataFrame(y_pred, columns=column, index = index)
y_train.columns = column
frames = [y_train,y_pred]
y_output = pd.concat(frames, axis = 0)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axhline(y=180, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
```

```
plt.axhline(y=180, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted lower half', fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted lower half')
```





Histogram

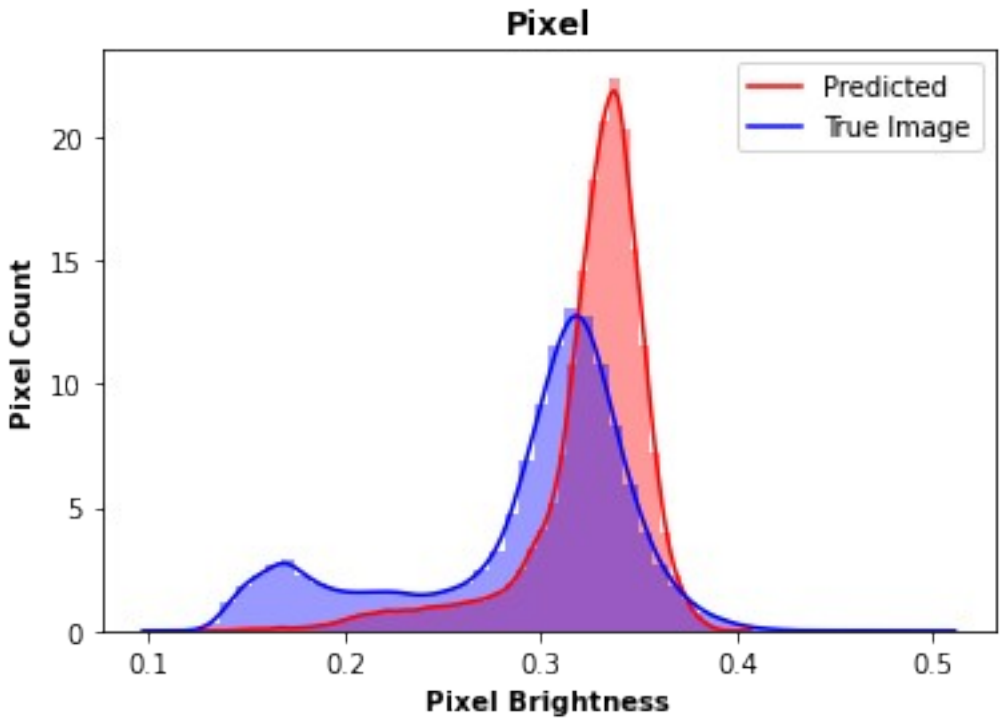
```
import seaborn as sns
```

```
sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
```

```
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.legend.Legend at 0x1324acb80>
```



80-20 Horizontal

```
x_train = fe.iloc[:288,]
x_test = fe.iloc[288:,]
y_train = albedo_map.iloc[:288,]
y_test = albedo_map.iloc[288:,]
```

```
from xgboost import XGBRegressor
from sklearn.multioutput import MultiOutputRegressor
regressor = MultiOutputRegressor(XGBRegressor(objective =
```

```
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
```

```
MultiOutputRegressor(estimator=XGBRegressor(base_score=None,
booster=None,
                                            colsample_bylevel=None,
                                            colsample_bynode=None,
                                            colsample_bytree=None,
                                            enable_categorical=False,
eta=0.1,
                                            eval_metric='rmse',
gamma=None,
                                            gpu_id=None,
importance_type=None,

interaction_constraints=None,
                                            learning_rate=None,
                                            max_delta_step=None,
max_depth=None,
                                            min_child_weight=None,
missing=nan,
                                            monotone_constraints=None,
                                            n_estimators=150,
n_jobs=None,
                                            num_parallel_tree=None,
                                            predictor=None,
random_state=0,
                                            reg_alpha=None,
reg_lambda=None,
                                            scale_pos_weight=None,
                                            subsample=0.6,
tree_method=None,
                                            validate_parameters=None,
                                            verbosity=None))
```

```
y_pred = regressor.predict(x_test)
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

```
0.0009540594969898107
```

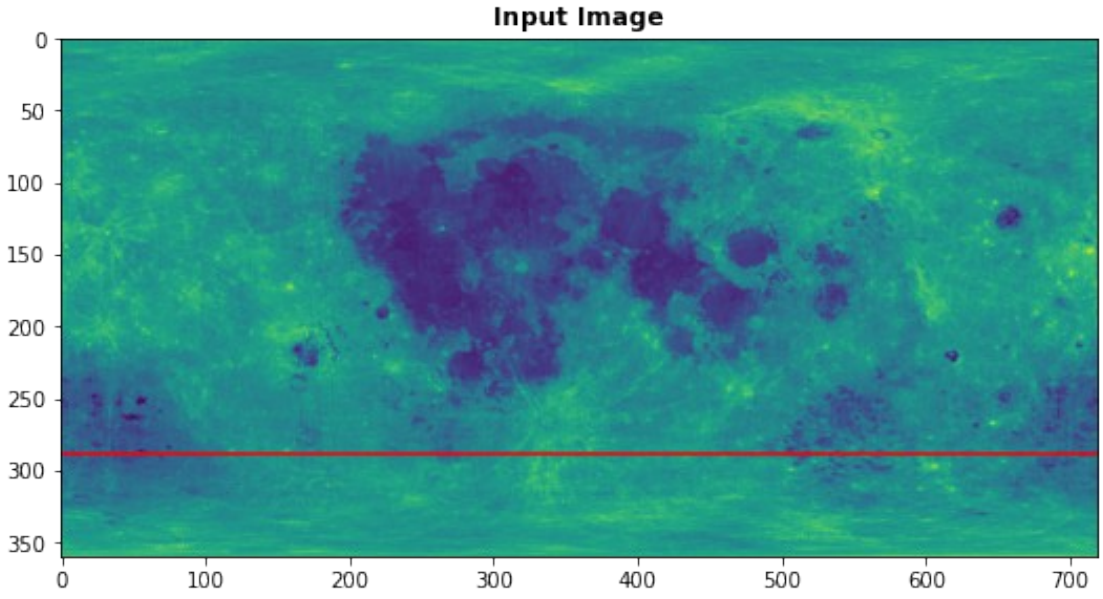
```
column = []
for i in range(1,721):
    column.append(i)

index = []
for i in range(288,360):
    index.append(i)
```

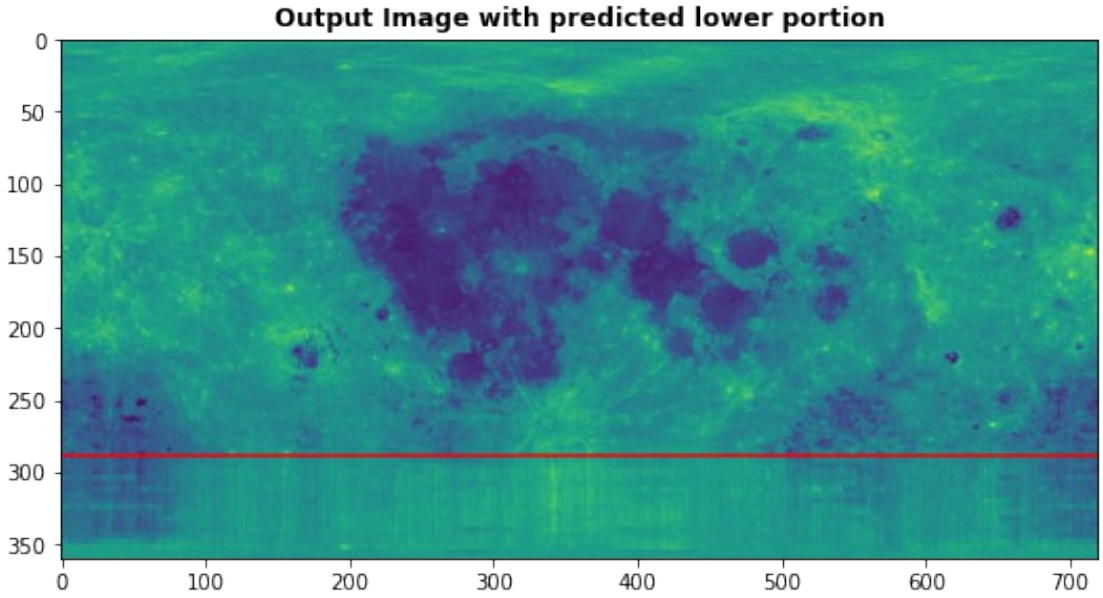
```
y_pred = pd.DataFrame(y_pred, columns=column, index = index)
y_train.columns = column
frames = [y_train,y_pred]
y_output = pd.concat(frames, axis = 0)
```

```
plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axhline(y=288, color='r')
plt.title('Input Image', fontweight="bold")
```

```
plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axhline(y=288, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted lower portion',
fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted lower portion')
```

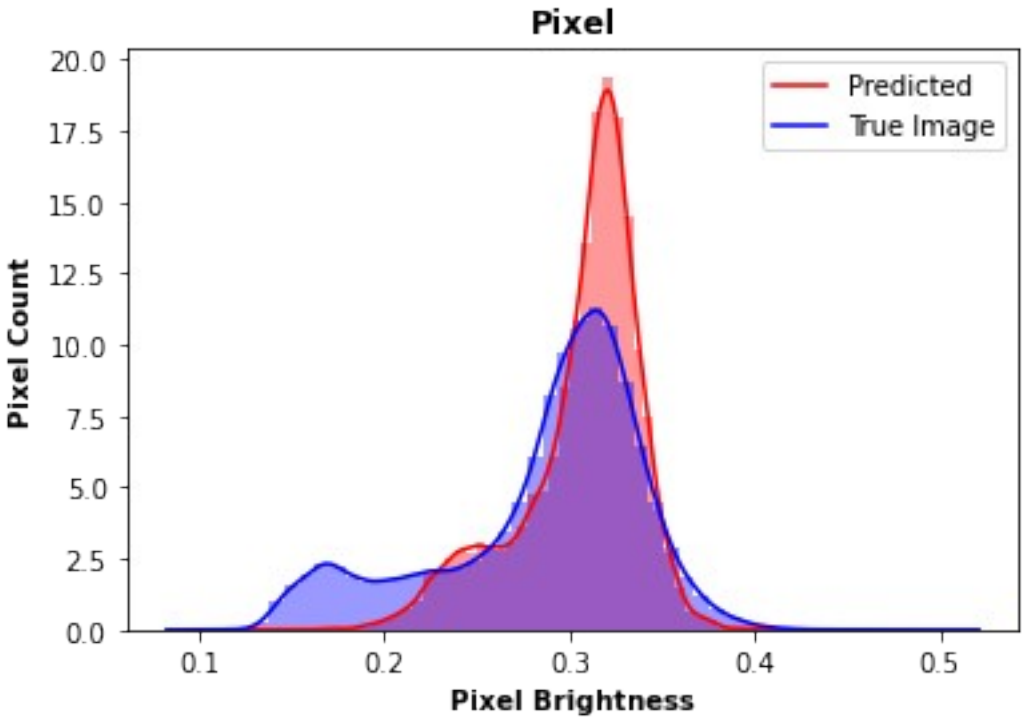




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x1332e81c0>
```



Becuase the error observed was very less for 80-20 ratio lets plot other datasets for 80-20

80-20

K

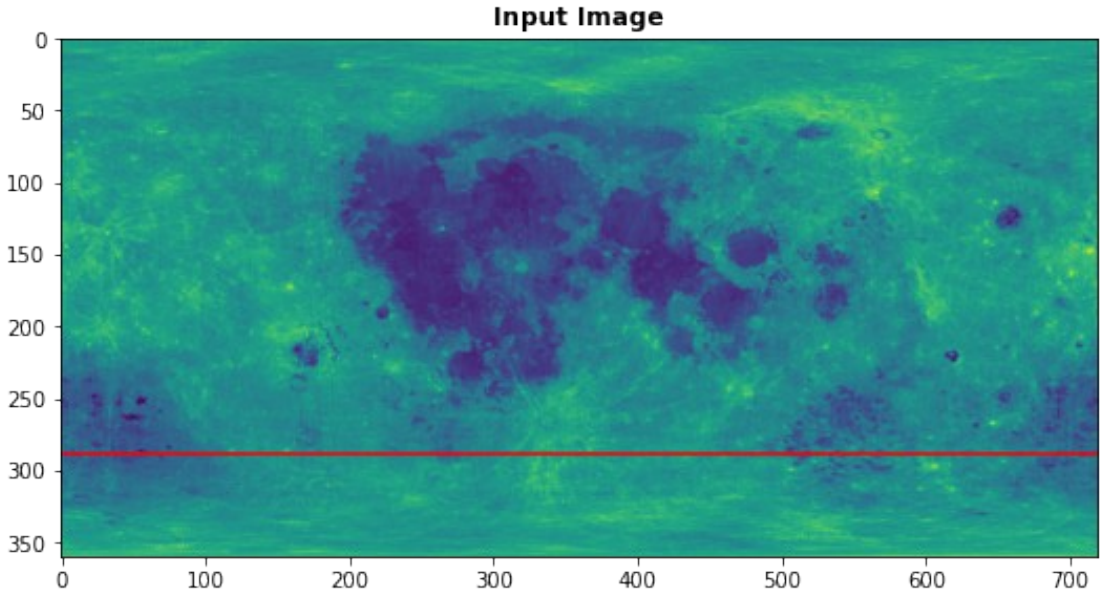
```
x_train = k.iloc[:288,]
x_test = k.iloc[288:,]
y_train = albedo_map.iloc[:288,]
y_test = albedo_map.iloc[288:,]

regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)

0.00184545332557657
```

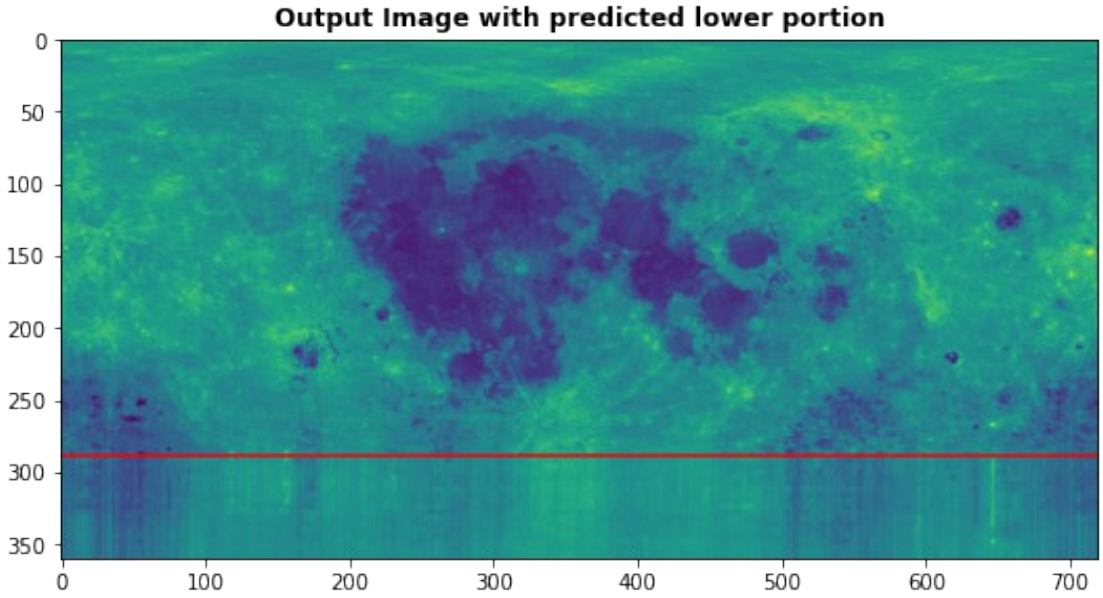
Output predicted comparisions

```
column = []
for i in range(1,721):
    column.append(i)

index = []
```

```
for i in range(288,360):
    index.append(i)

y_pred = pd.DataFrame(y_pred, columns=column, index = index)
y_train.columns = column
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 0)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axhline(y=288, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axhline(y=288, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted lower portion',
fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted lower portion')
```

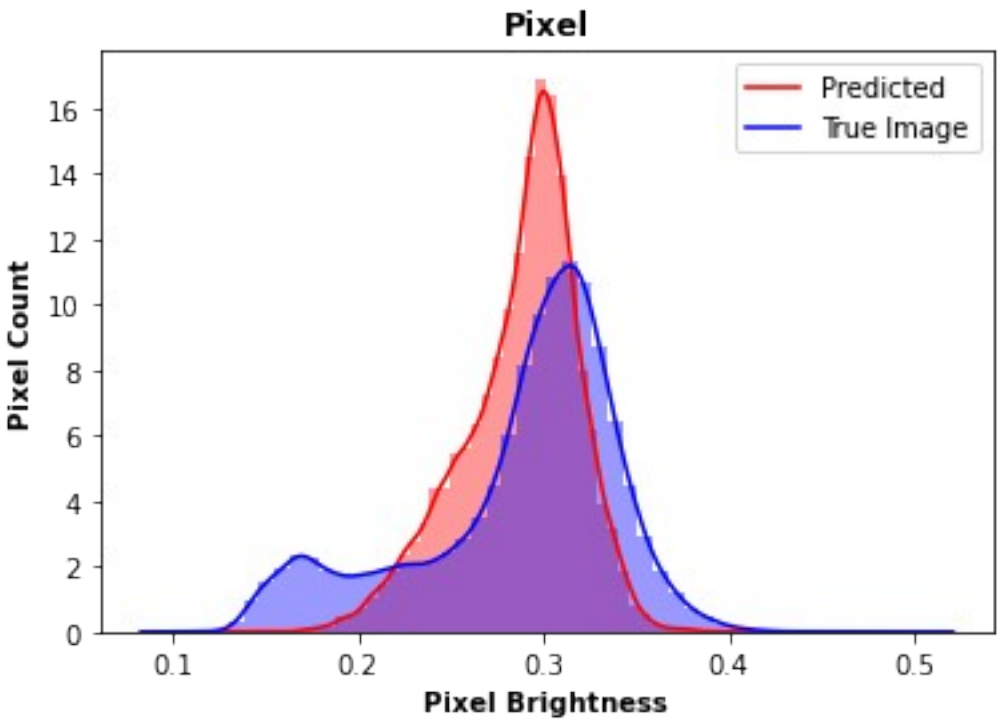




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x1333fddc0>
```



Th

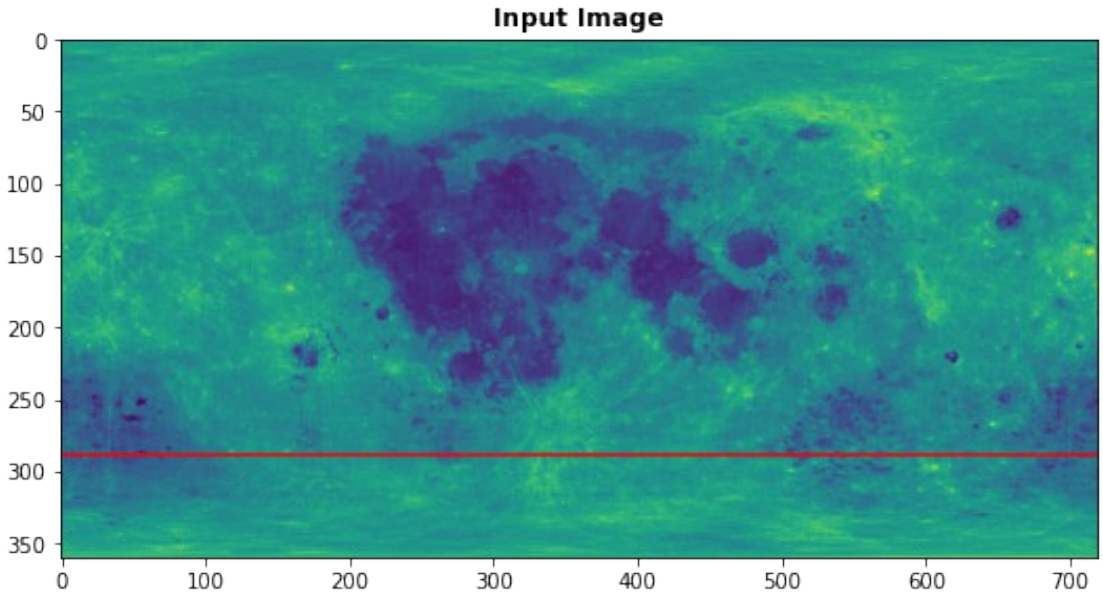
```
x_train = th.iloc[:288,]
x_test = th.iloc[288:,]
y_train = albedo_map.iloc[:288,]
y_test = albedo_map.iloc[288:,]

regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)

0.0015103615614187327
```

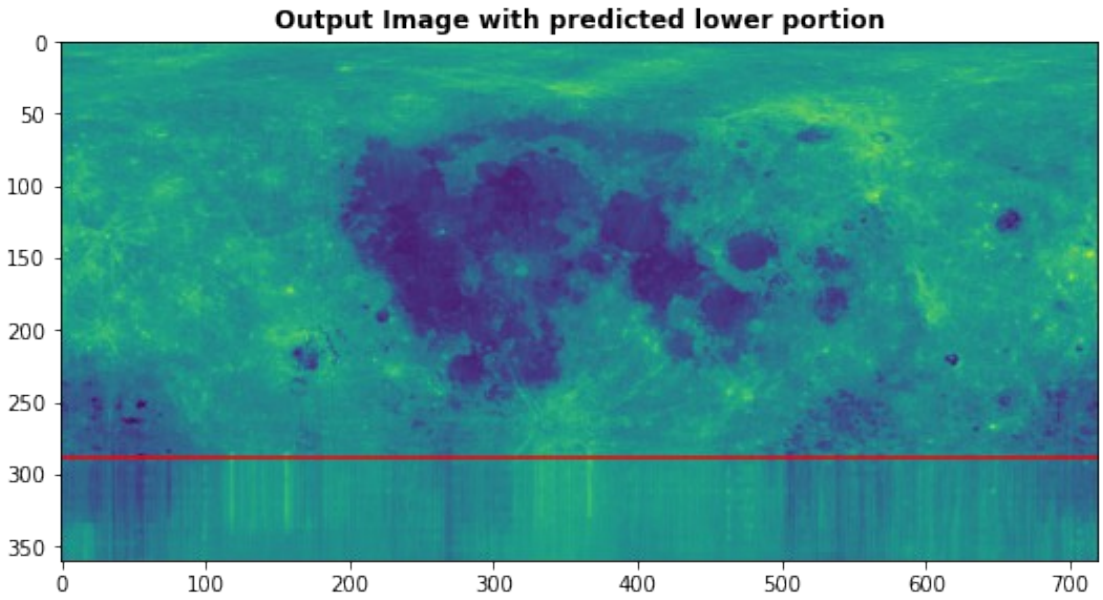
Output predicted comparisions

```
column = []
for i in range(1,721):
    column.append(i)

index = []
for i in range(288,360):
    index.append(i)

y_pred = pd.DataFrame(y_pred, columns=column, index = index)
y_train.columns = column
```

```
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 0)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axhline(y=288, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axhline(y=288, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted lower portion',
fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted lower portion')
```

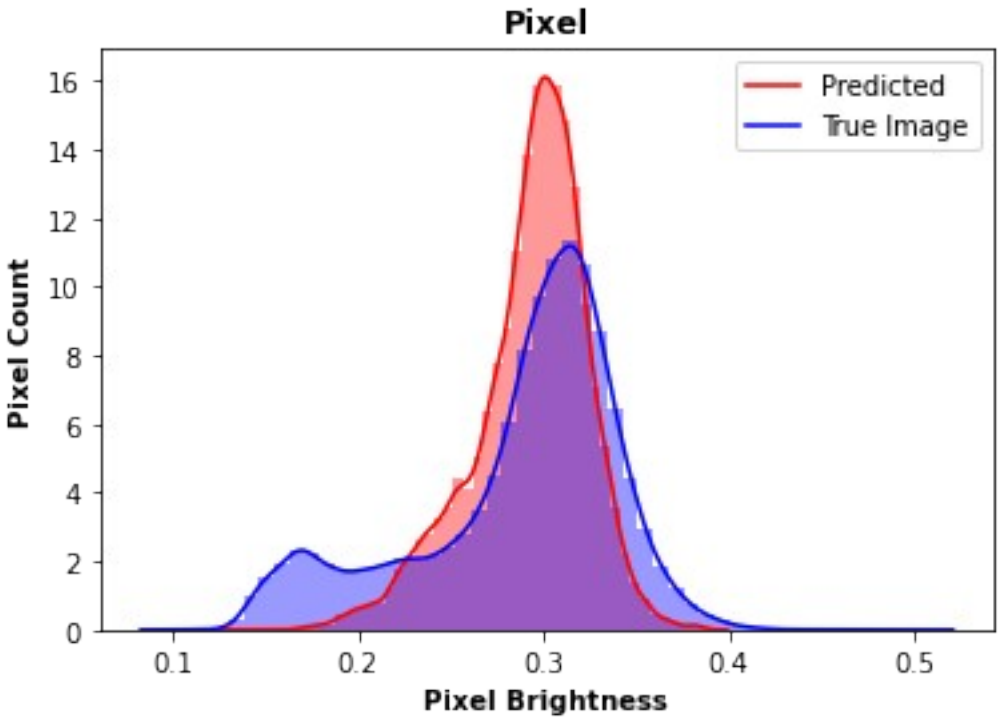




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x13372d0d0>
```



Ti

```
x_train = ti.iloc[:288,]
x_test = ti.iloc[288:,]
y_train = albedo_map.iloc[:288,]
y_test = albedo_map.iloc[288:,]
```

```
regressor = MultiOutputRegressor(XGBRegressor(objective =
'reg:squarederror',n_estimators=150, random_state = 0, eta = 0.1,
subsample = 0.6, eval_metric = 'rmse'))
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

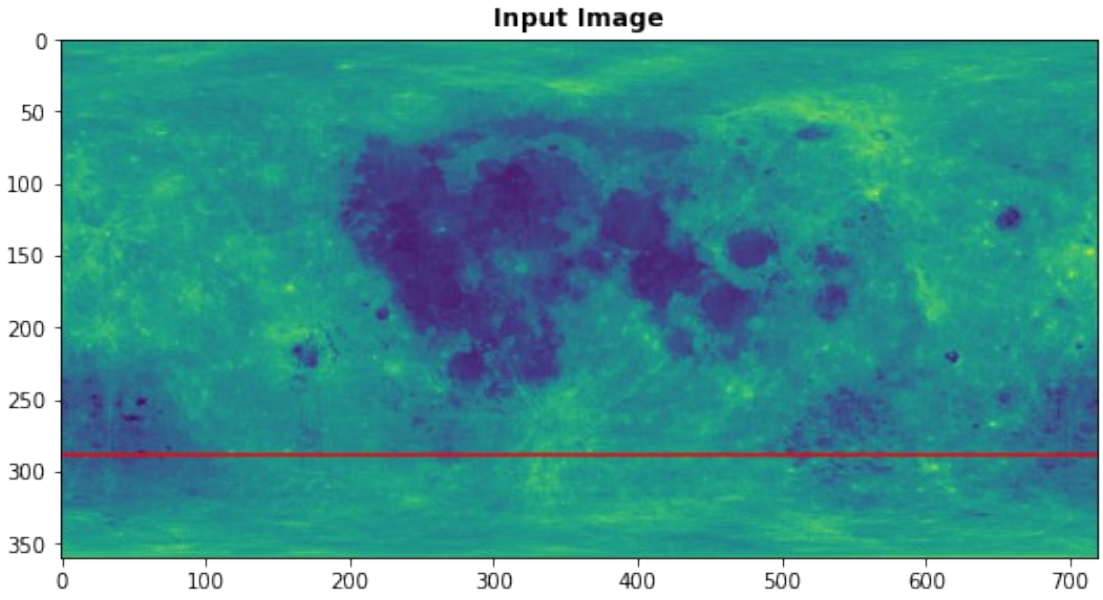
```
0.0008940511064421169
```

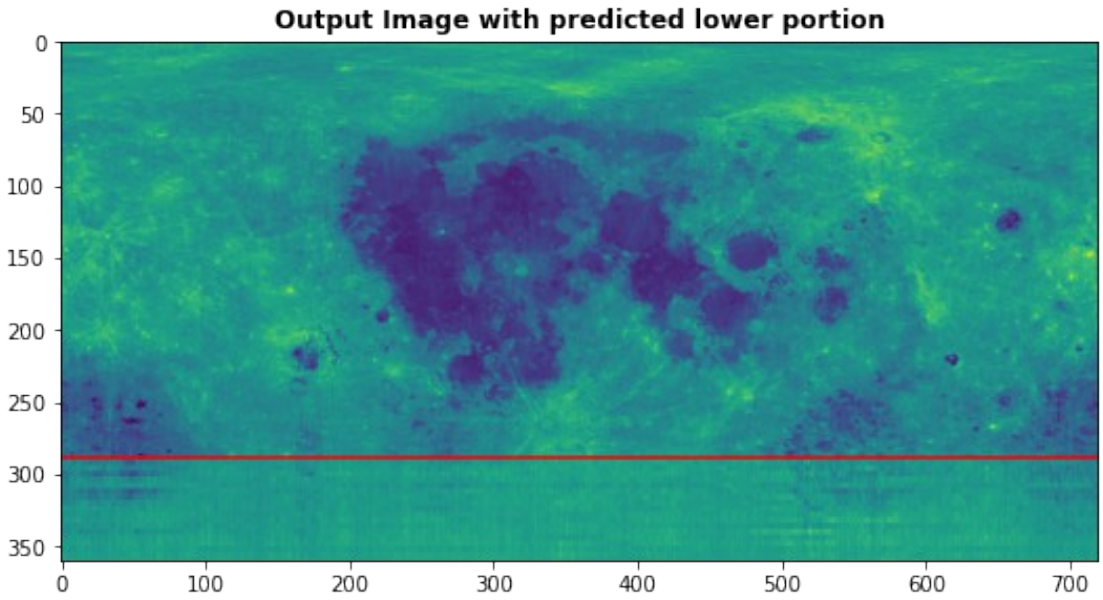
Output predicted comparisions

```
column = []
for i in range(1,721):
    column.append(i)

index = []
for i in range(288,360):
    index.append(i)

y_pred = pd.DataFrame(y_pred, columns=column, index = index)
y_train.columns = column
```

```
frames = [y_train, y_pred]
y_output = pd.concat(frames, axis = 0)

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 1)
plt.imshow(albedo_map_arr)
plt.axhline(y=288, color='r')
plt.title('Input Image', fontweight="bold")

plt.figure(figsize =(10,10))
plt.subplot(2, 1, 2)
plt.axhline(y=288, color='r')
plt.imshow(y_output)
plt.title('Output Image with predicted lower portion',
fontweight="bold")
```

```
Text(0.5, 1.0, 'Output Image with predicted lower portion')
```

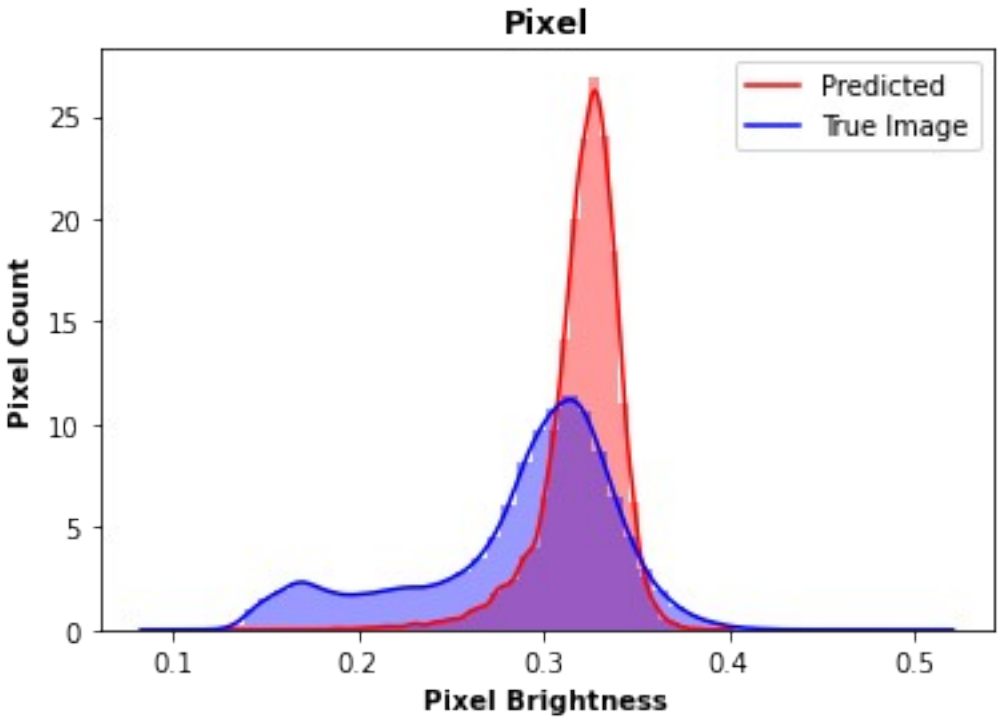




Histogram

```
import seaborn as sns

sns.distplot(y_pred, color= 'red')
sns.distplot(y_train, color = 'blue')
plt.title('Pixel', fontweight="bold")
plt.xlabel('Pixel Brightness', fontweight="bold")
plt.ylabel('Pixel Count', fontweight="bold")
plt.legend(['Predicted', 'True Image'])
```

```
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<matplotlib.legend.Legend at 0x133703b80>
```


Pixel
Predicted
True Image
Pixel Count
0
5
10
15
20
25
0.1
0.2
0.3
0.4
0.5
Pixel Brightness