Predict Chl-a in Turbid Estuarine Water

Data Setup

Import libraries

In [1]:

```
import arcpy, numpy, scipy, sklearn, sklearn.ensemble, pandas, seaborn, matp
```

Define input data variables

In [2]:

```
in_dataset = r'C:\Users\zieglerhm\Documents\Files\Portfolio\Estimate_Chla\Pr
in_test = r'C:\Users\zieglerhm\Documents\Files\Portfolio\Estimate_Chla\Predi
in_train = r'C:\Users\zieglerhm\Documents\Files\Portfolio\Estimate_Chla\Pred
in_columns = ['SHAPE@XY', 'OBJECTID', 'Station', 'Cnt_Station', 'Ave_Value_C
              'b1_Band', 'b2_Band', 'b3_Band', 'b4_Band', 'b5_Band', 'b6_Ban
              'b10_Band', 'b11_Band', 'b12_Band', 'b13_Band', 'b14_Band', 'b
              'b18_Band', 'b19_Band', 'b20_Band', 'b21_Band', 'b22_Band', 'b
              'b27_Band', 'b28_Band', 'b29_Band', 'b30_Band', 'b31_Band', 'b
              'b36_Band', 'b37_Band', 'b38_Band', 'b39_Band', 'b40_Band', 'b
              'b45_Band', 'b46_Band', 'b47_Band', 'b48_Band', 'b49_Band', 'b
              'b54_Band', 'b55_Band', 'b56_Band', 'b57_Band', 'b58_Band', 'b
              'b63_Band', 'b64_Band', 'b65_Band', 'b66_Band', 'b67_Band', 'b
              'b72_Band', 'b73_Band', 'b74_Band', 'b75_Band', 'b76_Band', 'b
              'b81_Band', 'b82_Band', 'b83_Band', 'b84_Band', 'b85_Band', 'b
```

Import prepared sample data from ArcGIS as numpy array

In [3]:

```
in_dataset_array = arcpy.da.FeatureClassToNumPyArray(in_dataset, in_columns)
in_test_array = arcpy.da.FeatureClassToNumPyArray(in_test, in_columns)
in_train_array = arcpy.da.FeatureClassToNumPyArray(in_train, in_columns)
in_train_spref = arcpy.Describe(in_train).SpatialReference
in_train_array
```

Out[3]:

```
array([([-80.812425  ,  28.68695833], 1, '27010875', 1, 4.00564275, 28.6869583
3, -80.812425  , 27, 36, 34, 38, 47, 55, 48, 48, 48, 51, 53, 54, 56, 60, 68, 6
8, 65, 66, 63, 68, 74, 73, 74, 77, 76, 76, 79, 78, 75, 70, 69, 68, 67, 64, 57,
51, 47, 46, 45, 44, 43, 41, 41, 40, 38, 35, 32, 31, 33, 36, 37, 32, 27, 23, 19,
17, 15, 12,  6,  6,  7,  8, 10, 12, 10,  7,  6,  6,  7,  8,  8, 10, 12, 15, 15,
14, 14, 15, 18, 21, 21, 20, 20, 21, 24, 29, 34),
       ([-80.80200694,  28.63580083], 2, 'IRLI06', 1, 4.74509997, 28.63580083,
-80.80200694, 26, 32, 29, 32, 41, 46, 41, 47, 46, 48, 50, 50, 54, 54, 61, 63, 6
0, 62, 58, 62, 71, 69, 66, 68, 70, 71, 71, 71, 70, 68, 66, 65, 67, 65, 57, 50,
46, 45, 45, 45, 45, 44, 44, 44, 42, 39, 37, 36, 36, 41, 46, 41, 35, 28, 28, 29,
26, 21, 16, 14, 12, 14, 16, 16, 14, 12, 11, 11, 12, 13, 14, 16, 18, 19, 19, 19,
20, 23, 26, 27, 28, 27, 26, 27, 29, 34, 41),
       ([-80.798395  ,  28.60347   ], 3, 'IRLI07', 1, 5.4798699 , 28.60347   ,
-80.798395  , 22, 27, 23, 30, 40, 42, 38, 40, 38, 41, 41, 44, 46, 46, 56, 57, 5
2, 54, 52, 55, 60, 58, 60, 60, 60, 60, 62, 62, 62, 60, 57, 55, 56, 55, 49, 46,
46, 44, 40, 38, 37, 39, 38, 37, 36, 34, 32, 30, 31, 36, 40, 34, 29, 25, 22, 21,
18, 15, 13, 14, 13, 14, 17, 19, 17, 16, 15, 15, 15, 16, 17, 20, 23, 25, 26, 27,
27, 28, 29, 29, 28, 28, 29, 32, 36, 40, 44),
       ([-80.74158333,  28.55636111], 4, 'IRLI09E', 2, 4.17642997, 28.55636111,
-80.74158333, 14, 25, 24, 25, 34, 39, 35, 37, 38, 37, 39, 43, 46, 46, 55, 56, 5
6, 60, 57, 60, 65, 66, 66, 66, 68, 71, 73, 73, 72, 70, 69, 69, 69, 68, 62, 57,
55, 53, 50, 51, 47, 45, 47, 48, 46, 41, 39, 38, 39, 45, 52, 47, 41, 36, 33, 32,
28, 22, 17, 13, 12, 14, 17, 17, 14, 11, 10, 11, 13, 14, 17, 18, 19, 20, 21, 23,
24, 24, 25, 27, 28, 27, 28, 30, 36, 42, 49),
       ([-80.76859389,  28.50121   ], 5, 'IRLI10', 1, 3.92627022, 28.50121   ,
-80.76859389,  0,  3,  7, 18, 27, 27, 24, 28, 27, 26, 29, 30, 30, 30, 39, 39, 3
7, 40, 38, 40, 44, 43, 44, 47, 46, 45, 45, 46, 44, 42, 42, 41, 40, 36, 31, 26,
24, 24, 21, 21, 20, 20, 21, 21, 19, 18, 17, 17, 16, 21, 24, 21, 17, 15, 12, 12,
11, 10,  6,  5,  5,  6,  7,  7,  5,  4,  4,  4,  5,  6,  7,  9, 11, 13, 15, 16,
17, 18, 20, 21, 22, 23, 24, 26, 26, 28, 33),
       ([-80.71723528,  28.73191722], 6, 'IRLML02', 2, 2.50479301, 28.73191722,
-80.71723528, 25, 35, 38, 37, 38, 46, 47, 48, 48, 51, 52, 54, 58, 60, 69, 71, 6
8, 70, 68, 72, 80, 80, 81, 82, 83, 81, 82, 83, 81, 80, 78, 77, 80, 75, 65, 59,
56, 55, 53, 52, 50, 48, 48, 49, 48, 44, 42, 41, 40, 43, 45, 38, 36, 31, 27, 27,
25, 22, 15, 15, 16, 18, 21, 21, 19, 18, 18, 18, 19, 21, 23, 24, 25, 26, 27, 28,
28, 27, 28, 28, 29, 30, 32, 33, 34, 38, 45)],
      dtype=[('SHAPE@XY', '<f8', (2,)), ('OBJECTID', '<i4'), ('Station', '<U25
5'), ('Cnt_Station', '<i4'), ('Ave_Value_Chla', '<f8'), ('Latitude_DD', '<f8'),
('Longitude_DD', '<f8'), ('b1_Band', '<i4'), ('b2_Band', '<i4'), ('b3_Band', '<
i4'), ('b4_Band', '<i4'), ('b5_Band', '<i4'), ('b6_Band', '<i4'), ('b7_Band',
'<i4'), ('b8_Band', '<i4'), ('b9_Band', '<i4'), ('b10_Band', '<i4'), ('b11_Ban
d', '<i4'), ('b12_Band', '<i4'), ('b13_Band', '<i4'), ('b14_Band', '<i4'), ('b1
5_Band', '<i4'), ('b16_Band', '<i4'), ('b17_Band', '<i4'), ('b18_Band', '<i4'),
('b19_Band', '<i4'), ('b20_Band', '<i4'), ('b21_Band', '<i4'), ('b22_Band', '<i
4'), ('b23_Band', '<i4'), ('b24_Band', '<i4'), ('b25_Band', '<i4'), ('b26_Ban
d', '<i4'), ('b27_Band', '<i4'), ('b28_Band', '<i4'), ('b29_Band', '<i4'), ('b3
0_Band', '<i4'), ('b31_Band', '<i4'), ('b32_Band', '<i4'), ('b33_Band', '<i4'),
('b34_Band', '<i4'), ('b35_Band', '<i4'), ('b36_Band', '<i4'), ('b37_Band', '<i
4'), ('b38_Band', '<i4'), ('b39_Band', '<i4'), ('b40_Band', '<i4'), ('b41_Ban
d', '<i4'), ('b42_Band', '<i4'), ('b43_Band', '<i4'), ('b44_Band', '<i4'), ('b4
5_Band', '<i4'), ('b46_Band', '<i4'), ('b47_Band', '<i4'), ('b48_Band', '<i4'),
```

```
('b49_Band', '<i4'), ('b50_Band', '<i4'), ('b51_Band', '<i4'), ('b52_Band', '<i
4'), ('b53_Band', '<i4'), ('b54_Band', '<i4'), ('b55_Band', '<i4'), ('b56_Ban
d', '<i4'), ('b57_Band', '<i4'), ('b58_Band', '<i4'), ('b59_Band', '<i4'), ('b6
0_Band', '<i4'), ('b61_Band', '<i4'), ('b62_Band', '<i4'), ('b63_Band', '<i4'),
('b64_Band', '<i4'), ('b65_Band', '<i4'), ('b66_Band', '<i4'), ('b67_Band', '<i
4'), ('b68_Band', '<i4'), ('b69_Band', '<i4'), ('b70_Band', '<i4'), ('b71_Ban
d', '<i4'), ('b72_Band', '<i4'), ('b73_Band', '<i4'), ('b74_Band', '<i4'), ('b7
5_Band', '<i4'), ('b76_Band', '<i4'), ('b77_Band', '<i4'), ('b78_Band', '<i4'),
('b79_Band', '<i4'), ('b80_Band', '<i4'), ('b81_Band', '<i4'), ('b82_Band', '<i
4'), ('b83_Band', '<i4'), ('b84_Band', '<i4'), ('b85_Band', '<i4'), ('b86_Ban
d', '<i4'), ('b87_Band', '<i4')])
```

In [4]:

```
in_train_array.shape
```

Out[4]:

```
(6,)
```

Convert the numpy array to a pandas data frame

In [5]:

```
in_dataset_df = pandas.DataFrame(in_dataset_array, columns = in_columns[1:])
in_test_df = pandas.DataFrame(in_test_array, columns = in_columns[1:])
in_train_df = pandas.DataFrame(in_train_array, columns = in_columns[1:])
in_train_df
```

Out[5]:

	OBJECTID	Station	Cnt_Station	Ave_Value_Chla	Latitude_DD	Longitude_DD	b1_Band	b2_B
0	1	27010875	1	4.005643	28.686958	-80.812425	27	
1	2	IRLI06	1	4.745100	28.635801	-80.802007	26	
2	3	IRLI07	1	5.479870	28.603470	-80.798395	22	
3	4	IRLI09E	2	4.176430	28.556361	-80.741583	14	
4	5	IRLI10	1	3.926270	28.501210	-80.768594	0	
5	6	IRLML02	2	2.504793	28.731917	-80.717235	25	

6 rows × 93 columns

To check viability of Random Forest model for the in_train_df dataset, create a correlation matrix for in_dataset (which contains all 10 variables from which in_train is a subset). Cast all variables to data type 'float64' before using pandas.DataFrame.corr() to create a correlation chart.

In [6]:

```
correlation = in_dataset_df[numpy.array(['Ave_Value_Chla','b1_Band', 'b2_Ban
                'b10_Band', 'b11_Band', 'b12_Band', 'b13_Band', 'b14_Band', 'b
                'b18_Band', 'b19_Band', 'b20_Band', 'b21_Band', 'b22_Band', 'b
                'b27_Band', 'b28_Band', 'b29_Band', 'b30_Band', 'b31_Band', 'b
                'b36_Band', 'b37_Band', 'b38_Band', 'b39_Band', 'b40_Band', 'b
                'b45_Band', 'b46_Band', 'b47_Band', 'b48_Band', 'b49_Band', 'b
                'b54_Band', 'b55_Band', 'b56_Band', 'b57_Band', 'b58_Band', 'b
                'b63_Band', 'b64_Band', 'b65_Band', 'b66_Band', 'b67_Band', 'b
                'b72_Band', 'b73_Band', 'b74_Band', 'b75_Band', 'b76_Band', 'b
                'b81_Band', 'b82_Band', 'b83_Band', 'b84_Band', 'b85_Band', 'b
correlation
```

Out[6]:

	Ave_Value_Chla	b1_Band	b2_Band	b3_Band	b4_Band	b5_Band	b6_Band	b7_
Ave_Value_Chla	1.000000	0.110250	-0.005274	-0.160892	0.037720	0.236918	0.163635	0.0
b1_Band	0.110250	1.000000	0.968963	0.929777	0.979154	0.942699	0.905088	0.9
b2_Band	-0.005274	0.968963	1.000000	0.978624	0.964031	0.902758	0.893336	0.8
b3_Band	-0.160892	0.929777	0.978624	1.000000	0.947614	0.840990	0.854585	0.8
b4_Band	0.037720	0.979154	0.964031	0.947614	1.000000	0.957883	0.949677	0.9
b5_Band	0.236918	0.942699	0.902758	0.840990	0.957883	1.000000	0.983738	0.9
b6_Band	0.163635	0.905088	0.893336	0.854585	0.949677	0.983738	1.000000	0.9
b7_Band	0.094093	0.900804	0.893506	0.868659	0.954595	0.965591	0.988521	1.0
b8_Band	0.121171	0.903779	0.890619	0.856527	0.946005	0.965695	0.982495	0.9
b9_Band	0.105461	0.875290	0.862160	0.832292	0.925004	0.953729	0.979889	0.9
b10_Band	0.064338	0.885230	0.881206	0.860711	0.938575	0.953268	0.980369	0.9
b11_Band	0.022212	0.887116	0.888440	0.874131	0.940738	0.947927	0.977557	0.9
b12_Band	0.026483	0.877771	0.877053	0.863308	0.930328	0.943059	0.973966	0.9
b13_Band	0.018889	0.849778	0.860726	0.850735	0.906960	0.920153	0.961117	0.9
b14_Band	-0.015762	0.868342	0.887839	0.887355	0.931676	0.926537	0.970267	0.9
b15_Band	-0.034931	0.882814	0.904499	0.906285	0.942756	0.926948	0.966403	0.9
b16_Band	-0.040056	0.883925	0.902541	0.910006	0.940469	0.917140	0.956344	0.9
b17_Band	-0.062928	0.858932	0.873899	0.888808	0.916548	0.896486	0.942428	0.9
b18_Band	-0.092705	0.868765	0.889998	0.906193	0.917694	0.887494	0.930601	0.9
b19_Band	-0.120247	0.869958	0.899634	0.912112	0.921167	0.889179	0.931304	0.9
b20_Band	-0.128695	0.864870	0.903191	0.917517	0.920411	0.886693	0.933177	0.9
b21_Band	-0.126642	0.852819	0.893079	0.917297	0.911992	0.868631	0.922198	0.9
b22_Band	-0.141436	0.849019	0.884296	0.909430	0.906289	0.864210	0.916966	0.9
b23_Band	-0.149501	0.864273	0.896262	0.921863	0.922690	0.873339	0.920497	0.9
b24_Band	-0.148585	0.858604	0.887022	0.918005	0.925889	0.875293	0.927383	0.9
b25_Band	-0.134566	0.850423	0.881511	0.911319	0.913492	0.865285	0.920009	0.9
b26_Band	-0.124565	0.844505	0.885635	0.908170	0.898580	0.860172	0.915286	0.9

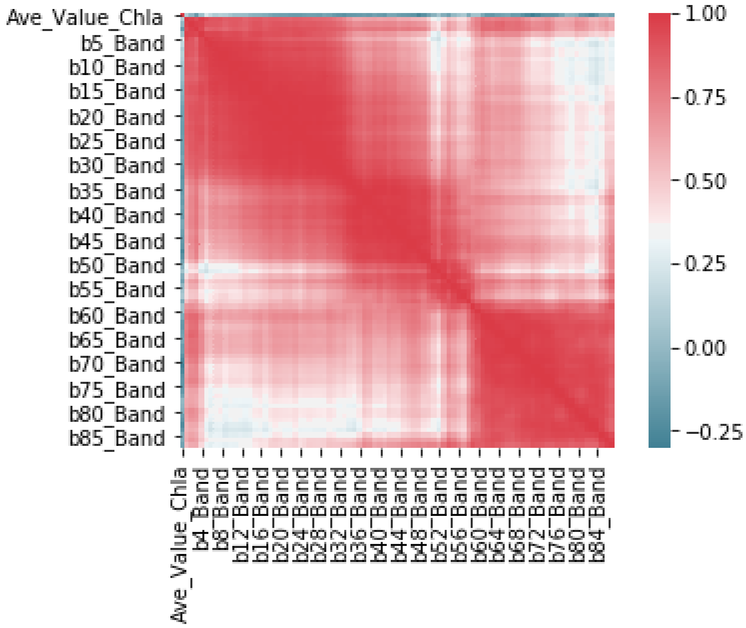
	Ave_Value_Chla	b1_Band	b2_Band	b3_Band	b4_Band	b5_Band	b6_Band	b7_
b27_Band	-0.127280	0.835076	0.883802	0.904310	0.892146	0.858811	0.916064	0.9
b28_Band	-0.138330	0.829176	0.872771	0.894124	0.887296	0.855285	0.912221	0.9
b29_Band	-0.098218	0.825003	0.863346	0.878460	0.879618	0.857928	0.912821	0.9
...	...	...	...	...	...	...	...	
b58_Band	-0.081028	0.522489	0.599633	0.624564	0.479895	0.362061	0.390982	0.4
b59_Band	0.048068	0.671402	0.694041	0.677342	0.582806	0.499464	0.473768	0.5
b60_Band	0.047788	0.733758	0.759689	0.742555	0.671287	0.583765	0.558277	0.6
b61_Band	-0.080733	0.791948	0.828825	0.832380	0.746715	0.614781	0.590908	0.6
b62_Band	-0.164225	0.760106	0.815274	0.823474	0.707448	0.563760	0.540461	0.5
b63_Band	-0.172591	0.737108	0.803195	0.811603	0.680202	0.528940	0.503239	0.5
b64_Band	-0.131267	0.759627	0.832942	0.829549	0.708669	0.571405	0.543547	0.5
b65_Band	-0.114578	0.772829	0.843240	0.835763	0.726540	0.592664	0.563793	0.6
b66_Band	-0.093438	0.758598	0.810234	0.805015	0.716512	0.581021	0.547791	0.6
b67_Band	-0.147001	0.758301	0.802911	0.805289	0.719442	0.577269	0.543403	0.6
b68_Band	-0.145668	0.764902	0.812464	0.807728	0.717381	0.577541	0.538906	0.5
b69_Band	-0.201964	0.726878	0.791086	0.791990	0.673947	0.521470	0.485921	0.5
b70_Band	-0.248653	0.683413	0.755431	0.770269	0.632085	0.459031	0.426750	0.4
b71_Band	-0.303984	0.619463	0.701152	0.723744	0.563573	0.386273	0.359405	0.4
b72_Band	-0.254041	0.637352	0.711071	0.724180	0.575782	0.405751	0.369211	0.4
b73_Band	-0.201580	0.651170	0.716291	0.719678	0.584251	0.422764	0.376649	0.4
b74_Band	-0.186691	0.681729	0.742028	0.742060	0.617304	0.458271	0.407242	0.4
b75_Band	-0.189006	0.642497	0.696043	0.694451	0.576012	0.416054	0.360371	0.4
b76_Band	-0.173308	0.583014	0.631139	0.628498	0.512992	0.359969	0.305695	0.3
b77_Band	-0.105387	0.582114	0.620058	0.607120	0.506914	0.364849	0.308258	0.3
b78_Band	-0.002702	0.600014	0.621949	0.599218	0.503758	0.366230	0.294897	0.3
b79_Band	-0.023808	0.603803	0.637524	0.614692	0.501986	0.355148	0.280799	0.3
b80_Band	-0.045213	0.664197	0.705098	0.680265	0.559362	0.415330	0.341535	0.3
b81_Band	-0.152372	0.641333	0.690464	0.675484	0.535255	0.381405	0.312290	0.3
b82_Band	-0.182189	0.625930	0.664016	0.651430	0.526473	0.365122	0.291938	0.3
b83_Band	-0.245032	0.564884	0.610275	0.607505	0.476030	0.304619	0.237801	0.2
b84_Band	-0.183113	0.553193	0.584948	0.570265	0.454165	0.295986	0.217109	0.2
b85_Band	-0.089060	0.582838	0.625140	0.597783	0.474753	0.345862	0.272869	0.3
b86_Band	-0.077450	0.602068	0.660381	0.637395	0.501800	0.389674	0.336765	0.3
b87_Band	-0.124082	0.533827	0.600292	0.605319	0.439362	0.308465	0.277470	0.3

88 rows × 88 columns

Plot the result as a correlation matrix

In [7]:

```
ax = seaborn.heatmap(correlation, cmap=seaborn.diverging_palette(220, 10, as
matplotlib.pyplot.show()
```



Many of the predictor variables are positive (bright red), which makes random forest a good choice as it can handle predictor variables that are dependent on each other in a way that minimizes bias.

Use the print command to show the sizes of the training and test datasets. Type labels for both and concatenate them with the string version of the variable.

In [8]:

```
print('Training Data Size = ' + str(in_train_df.shape[0]))
print('Test Data Size = ' + str(in_test_df.shape[0]))
print('Entire Dataset Size = ' + str(in_train_df.shape[0] + in_test_df.shape
```

```
Training Data Size = 6
Test Data Size = 4
Entire Dataset Size = 10
```

Train Random Forest Regressor

Train & Test Data

Train your random forest regressor using the training data you have created.

First, create the variable train_rfr to show the results of running the RandomForestRegressor command to create 500 trees. Then use the .fit argument to apply the forest results to the training data.

In [9]:

```
train_rfr = sklearn.ensemble.RandomForestRegressor(n_estimators = 500, oob_s
independent_col = ['b1_Band', 'b2_Band', 'b3_Band', 'b4_Band', 'b5_Band', 'b
                   'b10_Band', 'b11_Band', 'b12_Band', 'b13_Band', 'b14_Band', 'b
                   'b18_Band', 'b19_Band', 'b20_Band', 'b21_Band', 'b22_Band', 'b
                   'b27_Band', 'b28_Band', 'b29_Band', 'b30_Band', 'b31_Band', 'b
                   'b36_Band', 'b37_Band', 'b38_Band', 'b39_Band', 'b40_Band', 'b
                   'b45_Band', 'b46_Band', 'b47_Band', 'b48_Band', 'b49_Band', 'b
                   'b54_Band', 'b55_Band', 'b56_Band', 'b57_Band', 'b58_Band', 'b
                   'b63_Band', 'b64_Band', 'b65_Band', 'b66_Band', 'b67_Band', 'b
                   'b72_Band', 'b73_Band', 'b74_Band', 'b75_Band', 'b76_Band', 'b
                   'b81_Band', 'b82_Band', 'b83_Band', 'b84_Band', 'b85_Band', 'b
dependent_col = ['Ave_Value_Chla']
train_rfr.fit(in_train_df[numpy.array(independent_col)], numpy.array(in_trai
```

Out[9]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
           oob_score=True, random_state=None, verbose=0, warm_start=False)
```

Run the classification again using the test dataset and place in a new variable, test_pred.

In [10]:

```
test_pred = train_rfr.predict(in_test_df[independent_col])
test_pred
```

Out[10]:

```
array([4.25399619, 4.47988191, 4.35075083, 3.6374174 ])
```

create a variable, test_vars, to store the true values of the in_test_df dependent variable.

In [11]:

```
test_vars = numpy.array(in_test_df[dependent_col]).flatten()
test_vars
```

Out[11]:

```
array([5.62214993, 3.45017987, 3.95554015, 4.31573137])
```

Check the accuracy of the result by calculating both RMSE and R^2.

In [12]:

```
test_pred_rmse = (sum((test_pred - test_vars)**2)/len(test_pred))**0.5
print("{:>16}{:5.2f}".format("RMSE: ", test_pred_rmse))
test_pred_rmse_conf = 1.96*test_pred_rmse
print("{:>15}{:6.2f}".format("RMSE 95% Conf.:", test_pred_rmse_conf))
test_pred_rsq = train_rfr.score(in_test_df[numpy.array(independent_col)], te
print("{:>15}{:6.2f}".format("R^2:",test_pred_rsq))
```

```
          RMSE:   0.94
RMSE 95% Conf.:   1.85
           R^2:  -0.37
```

The prediction is not very strong with an R^2 of -0.4 and RMSE of 0.95. 95% of data predicted data should fall within +-1.86 of the prediction.

Entire Dataset

We will accept these results and see what happens when we train the Random Forest Regressor on the entire 10-sample dataset.

In [13]:

```
dataset_rfr = sklearn.ensemble.RandomForestRegressor(n_estimators = 500, oob
dataset_rfr.fit(in_dataset_df[numpy.array(independent_col)], numpy.array(in_
dataset_pred = train_rfr.predict(in_dataset_df[independent_col])
print("Predicted: {}".format(dataset_pred))
dataset_vars = numpy.array(in_dataset_df[dependent_col]).flatten()
print("Actual: {}".format(dataset_vars))
```

```
Predicted: [4.0940153  4.29388371 4.51587803 5.01835443 4.16970642 4.13753847
 4.45888863 4.28773443 3.34741148 3.63815415]
Actual: [4.00564275 5.62214993 4.74509997 5.4798699  4.17642997 3.92627022
 3.45017987 3.95554015 2.50479301 4.31573137]
```

In [14]:

```
dataset_pred_rmse = (sum((dataset_pred - dataset_vars)**2)/len(dataset_pred)
print("{:>16}{:5.2f}".format("RMSE: ", dataset_pred_rmse))
dataset_pred_rmse_conf = 1.96*dataset_pred_rmse
print("{:>15}{:6.2f}".format("RMSE 95% Conf.:", dataset_pred_rmse_conf))
dataset_pred_rsq = dataset_rfr.score(in_dataset_df[numpy.array(independent_c
print("{:>15}{:6.2f}".format("R^2:",dataset_pred_rsq))
```

```
          RMSE:   0.66
RMSE 95% Conf.:   1.30
           R^2:   0.81
```

Run Random Forest Regressor on Raster to Estimate Chl-a for Each Cell

Import the raster which contains the dataset to be predicted where the independent variables (band reflectance) are known and the dependent variable is unknown (chl-a concentration), as a numpy array.

First, define input variables.

In [16]:

```
in_rast = arcpy.Raster(r'C:\Users\zieglerhm\Documents\Files\Portfolio\Estima
in_rast_lowerleftpt = arcpy.Point(in_rast.extent.XMin, in_rast.extent.YMin)
in_rast_xsize = in_rast.meanCellWidth
in_rast_ysize = in_rast.meanCellHeight
in_rast_sptref = in_rast.spatialReference
```

Now create the raster array. Notice the output shape of in_rast_array.

In [17]:

```
in_rast_array = arcpy.RasterToNumPyArray(in_rast, nodata_to_value = None)
print("Shape: {}".format(in_rast_array.shape))
print("Data type: {}".format(in_rast_array.dtype))
print(in_rast_array)
```

```
Shape: (87, 201, 80)
Data type: int16
[[[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 ...

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]]
```

In order to use in_rast_array transpose and reshape as input as a 2d array with the number of rows being the individual cells (201 x 70 = 16080) and the number of columns being the number of bands (87).

In [18]:

```
in_rast_array_reshape = in_rast_array.transpose(1, 2, 0).reshape(16080,87)
print("Shape: {}".format(in_rast_array_reshape.shape))
print("Data type: {}".format(in_rast_array_reshape.dtype))
print(in_rast_array_reshape)
```

```
Shape: (16080, 87)
Data type: int16
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Apply the Random Forest Regressor model created from the 10 sample points in dataset.rar. The output is a 1d array with the estimated values of Chl-a.

In [19]:

```
in_rast_pred = dataset_rfr.predict(in_rast_array_reshape)
print("Shape: {}".format(in_rast_pred.shape))
print("Data type: {}".format(in_rast_pred.dtype))
print(in_rast_pred)
```

```
Shape: (16080,)
Data type: float64
[4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
```

Convert the 1d array back into a 2d array of shape 201, 80, where each record in the raster grid only has one value, the estimated value of chl-a for that cell.

In [20]:

```
in_rast_pred_reshape = in_rast_pred.reshape(201, 80)
print("Shape: {}".format(in_rast_pred_reshape.shape))
print("Data type: {}".format(in_rast_pred_reshape.dtype))
print(in_rast_pred_reshape)
```

```
Shape: (201, 80)
Data type: float64
[[4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
 [4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
 [4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
 ...
 [4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
 [4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]
 [4.26550856 4.26550856 4.26550856 ... 4.26550856 4.26550856 4.26550856]]
```

Convert the numpy array back to a raster.

In [22]:

```
arcpy.env.overwriteOutput = True
out_rast = arcpy.NumPyArrayToRaster(in_rast_pred_reshape, in_rast_lowerleftp
out_rast.save(r'C:\Users\zieglerhm\Documents\Files\Portfolio\Estimate_Chla\P
arcpy.management.DefineProjection(out_rast, in_rast_sptref)
```

Out[22]:

```
<Result 'C:\\Users\\zieglerhm\\Documents\\Files\\Portfolio\\Estimate_Chla\\Pred
ict_ChlA.gdb\\IRL_Predicted_ChlA'>
```