In [2]:

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot
from sklearn.model_selection import KFold
```

In [81]:

```python
df1 = pd.read_csv("Randomedi100.csv")
pd.set_option("display.max_columns", None)

df1
```

Out[81]:

	GaugeID	AnnuaL Mean Temperature	Precipitation Seasonality	Drainage Area (km^2)	Basin Length	Compa Coef
0	IWM-gauge-0100	22.307884	93.547569	192.502640	21644.85156	1.
1	IWM-gauge-0248	17.263815	67.755791	213.932037	22196.08398	1.
2	IWM-gauge-0387	25.653601	112.179329	18450.912110	404836.18750	1.
3	IWM-gauge-0636	23.128416	158.140106	420.320160	128530.13280	1.
4	IWM-gauge-0763	24.650948	116.591377	5066.837402	353065.90630	1.
5	IWM-gauge-0877	26.049732	141.446732	350.433624	124719.17190	1.
6	IWM-gauge-0880	24.212206	131.164093	2306.040039	280623.65630	2.
7	IWM-gauge-0908	23.686493	138.930984	3779.519775	341352.90630	2.
8	IWM-gauge-1061	25.385981	118.453384	35702.257810	123911.21090	1.
9	IWM-gauge-1089	24.259298	115.949387	14968.109380	213105.40630	1.
10	IWM-gauge-1169	24.121849	131.046005	1942.562744	107232.51560	1.
11	IWM-gauge-1442	25.424381	156.476242	791.283936	66069.01563	1.
12	IWM-gauge-1553	25.237150	148.410629	541.287476	101152.00000	1.
13	IWM-gauge-1602	24.768211	88.936455	69253.421880	97097.92969	1.
14	IWM-gauge-1642	24.924210	98.838333	7870.645508	121632.57030	1.
15	IWM-gauge-1784	25.415009	146.423004	300.548340	39819.76172	1.
	IWM-					

	GaugeID	AnnuaL Mean Temperature	Precipitation Seasonality	Drainage Area (km^2)	Basin Length	Compa Coef
16	gauge-2037	25.485306	113.687416	15810.197270	179379.31250	1.5
17	IWM-gauge-2039	25.589821	134.401764	35479.253910	98009.97656	1.7
18	IWM-gauge-2104	24.263506	112.860062	12061.231450	217899.56250	1.7
19	IWM-gauge-2113	26.207943	79.652954	8584.285156	171659.20310	1.3
20	IWM-gauge-2205	25.652811	100.314293	4394.541504	362027.06250	1.8
21	IWM-gauge-2257	24.182590	154.482971	993.997376	389372.43750	1.8
22	IWM-gauge-2293	27.012070	126.736610	45774.484380	104476.42190	1.4
23	IWM-gauge-2353	24.751366	86.637672	2481.376465	154991.81250	1.4
24	IWM-gauge-2459	26.607559	124.961136	55306.183590	114205.16410	1.4
25	IWM-gauge-2509	24.400270	116.571381	2209.014648	25594.19336	1.2
26	IWM-gauge-2553	23.448601	136.135284	2821.115967	87605.39063	1.5
27	IWM-gauge-2784	24.699221	120.672516	6936.809570	90587.85938	1.3
28	IWM-gauge-2914	24.798716	96.240807	9891.224609	36318.58984	1.3
29	IWM-gauge-2984	25.451561	97.130249	9015.836914	32048.45313	1.6
30	IWM-gauge-3060	24.077030	91.857086	52122.167970	35678.53906	1.2
31	IWM-gauge-3088	25.490677	97.686272	66454.703130	75754.85938	1.3
32	IWM-gauge-3181	26.961117	124.130837	15843.124020	87628.03906	1.6
33	IWM-gauge-3273	25.921089	97.874069	6941.307617	161853.26560	1.7
34	IWM-gauge-3289	21.879065	141.480133	574.843384	39220.07031	1.4
35	IWM-gauge-3333	24.137991	107.703888	14475.828130	193411.75000	1.5
36	IWM-gauge-3369	24.777304	99.339409	6891.486816	82635.13281	1.5
37	IWM-gauge-3643	25.242662	137.147522	3266.582031	392668.06250	1.5
38	IWM-gauge-3744	25.401436	97.490074	64073.714840	38557.53516	1.3
39	IWM-gauge-3812	22.698811	80.948708	5579.921387	170700.29690	1.7
40	IWM-gauge-	24.740263	109.545830	8494.139648	424543.28130	1.6

3825
AnnuaL Precipitation Drainage Compa

In [82]:

```
#Preparing data for training
X = df1.iloc[:, 1:11].values
y = df1.iloc[:, 11].values
```

In [83]:

```
y
```

Out[83]:

```
array([ 133.27  ,   20.07  , 1372.3735, 1650.    ,  579.765 ,  319.5185,
        949.25  ,  803.77  , 3099.5   , 4086.48  , 1181.79  ,  292.83  ,
        326.225 ,  860.395 ,  792.377 ,  275.641 , 2087.024 , 3555.25  ,
       2937.5265,   95.94  ,  640.    ,  932.892 , 3974.11  ,   65.5165,
       2915.    ,  222.645 , 1330.991 ,  870.4019,  207.06  ,  449.098 ,
        841.59  , 1335.645 ,  707.64  ,  475.36  ,  232.4355, 1642.55  ,
        425.4   , 2360.6   , 1488.335 ,   48.705 ,  899.74  ])
```

In [84]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size = 0.20, rando
m_state = 0)
```

In [85]:

```
kf = KFold(n_splits=2, random_state = 0)
kf.get_n_splits(X)
```

Out[85]:

```
2
```

In [86]:

```
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

```
TRAIN: [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40] TES
: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20] TEST: [
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40]
```

In [87]:

```
#Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

In [88]:

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators =16, random_state = 0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

In [89]:

```
y_pred
```

Out[89]:

```
array([ 789.9259375 , 3121.732125  ,  737.548375  , 2463.822     ,
        431.084125  ,  545.1193125 ,  608.0681875 ,  616.49025   ,
        494.933875  , 1408.37025   , 1981.847125  ,  854.72878125,
        715.37934375,  374.5624375 ,  673.39375   ,  741.9255625 ,
        631.10365625, 2164.30765625,  945.3386875 ,  696.2985625 ])
```

In [90]:

```
df5 = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df5.to_csv("highflow1.csv")
df5
```

Out[90]:

	Real Values	Predicted Values
0	932.8920	789.925937
1	3974.1100	3121.732125
2	65.5165	737.548375
3	2915.0000	2463.822000
4	222.6450	431.084125
5	1330.9910	545.119312
6	870.4019	608.068188
7	207.0600	616.490250
8	449.0980	494.933875
9	841.5900	1408.370250
10	1335.6450	1981.847125
11	707.6400	854.728781
12	475.3600	715.379344
13	232.4355	374.562437
14	1642.5500	673.393750
15	425.4000	741.925562
16	2360.6000	631.103656
17	1488.3350	2164.307656
18	48.7050	945.338687
19	899.7400	696.298563

In [91]:

```
regressor.score(X_test,y_test)
```

Out[91]:

0.5717923683570294

In [92]:

```
from sklearn import metrics
a=metrics.mean_absolute_error(y_test,y_pred)
b=metrics.mean_squared_error(y_test,y_pred)
c=np.sqrt(metrics.mean_absolute_error(y_test,y_pred))
a,b,c
```

Out[92]:

(518.1953918749999, 423646.03842514876, 22.76390546182706)

In [93]:

```
y_pred1 = regressor.predict(X_train)
```

In [94]:

```
y_pred1
```

Out[94]:

```
array([ 166.87925   ,  103.20425   , 1892.73225   ,  994.64834375,
        618.35771875,  576.74759375,  923.3484375 ,  751.785     ,
       2599.8165    , 4086.48      ,  858.8659375 ,  439.4746875 ,
        480.381875  , 1315.4870625 ,  828.314125  ,  302.81553125,
       2315.04159375, 3098.144625  , 2682.003875  ,  698.05184375,
        691.746375  ])
```

In [95]:

```
df6 = pd.DataFrame({'Real Values':y_train, 'Predicted Values':y_pred1})
df6.to_csv("highflow2.csv")
df6
```

Out[95]:

	Real Values	Predicted Values
0	133.2700	166.879250
1	20.0700	103.204250
2	1372.3735	1892.732250
3	1650.0000	994.648344
4	579.7650	618.357719
5	319.5185	576.747594
6	949.2500	923.348438
7	803.7700	751.785000
8	3099.5000	2599.816500
9	4086.4800	4086.480000
10	1181.7900	858.865937
11	292.8300	439.474687
12	326.2250	480.381875
13	860.3950	1315.487063
14	792.3770	828.314125
15	275.6410	302.815531
16	2087.0240	2315.041594
17	3555.2500	3098.144625
18	2937.5265	2682.003875
19	95.9400	698.051844
20	640.0000	691.746375

In [96]:

```
from sklearn import metrics
a1=metrics.mean_absolute_error(y_train,y_pred1)
b1=metrics.mean_squared_error(y_train,y_pred1)
c1=np.sqrt(metrics.mean_absolute_error(y_train,y_pred1))
a1,b1,c1
```

Out[96]:

```
(233.44185416666681, 99006.96360926746, 15.278804081689994)
```

In [97]:

```
regressor.score(X_train,y_train)
```

Out[97]:

0.9298892344590857

In [98]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
```

In [99]:

```
importance=regressor.feature_importances_
```

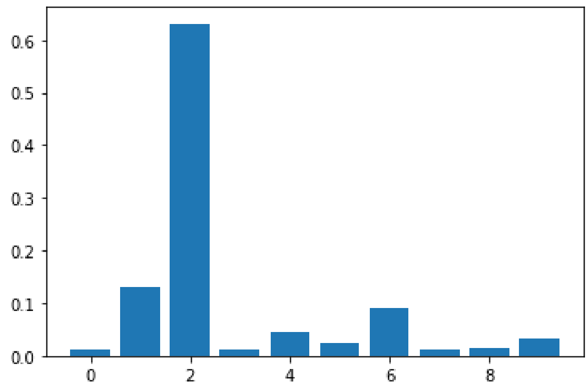
In [100]:

```
from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
```

In [101]:

```
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

```
Feature: 0, Score: 0.01093
Feature: 1, Score: 0.12887
Feature: 2, Score: 0.63044
Feature: 3, Score: 0.01211
Feature: 4, Score: 0.04503
Feature: 5, Score: 0.02354
Feature: 6, Score: 0.09021
Feature: 7, Score: 0.01214
Feature: 8, Score: 0.01437
Feature: 9, Score: 0.03236
```



In [102]:

```
#Correlation Matrix

corr = df1.corr()
corr.style.background_gradient(cmap='RdYlGn')
```

Out[102]:

	AnnuaL Mean Temperature	Precipitation Seasonality	Drainage Area (km^2)	Basin Length	Compactness Coefficient
AnnuaL Mean	1	0.216688	0.225187	0.144021	0.235283

	AnnuaL Mean Temperature	Precipitation Seasonality	Drainage Area (km^2)	Basin Length	Compactness Coefficient
Temperature	1	0.216688	0.325187	0.144921	0.235283
Precipitation Seasonality	0.216688	1	-0.262082	0.161334	0.399939
Drainage Area (km^2)	0.325187	-0.262082	1	-0.223323	-0.241212
Basin Length	0.144921	0.161334	-0.223323	1	0.627992
Compactness Coefficient	0.235283	0.399939	-0.241212	0.627992	1
Drainage Texture	0.178263	0.0888714	-0.138631	0.949853	0.466337
Max Temperature of Warmest Month	0.430429	0.175863	0.0174227	0.519557	0.47626
Maximal Flow Length	0.143433	0.13759	-0.2223	0.985324	0.631661
Mean Temperature of Warmest Quarter	0.438797	0.176151	0.0276366	0.464998	0.504139
Mean Temperature of Wettest Quarter	0.38262	0.206808	0.0196023	0.372912	0.533527
High Flow (m3/s)	0.290149	0.23458	0.444208	0.155113	0.0908691

In [83]:

```
pyplot.savefig("corr.png")
```

<Figure size 432x288 with 0 Axes>

In [88]:

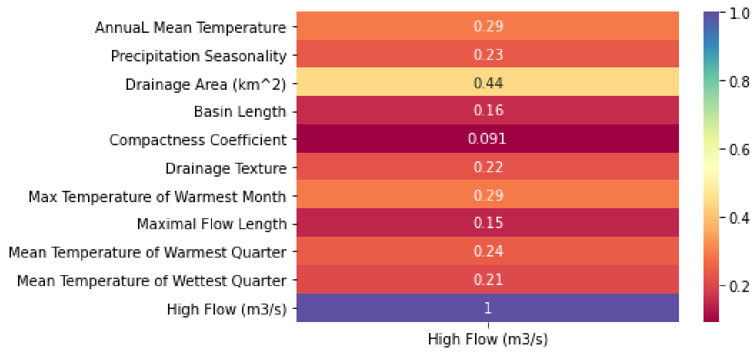
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:60% !important; }</style>"))
```

In [80]:

```
import seaborn as sns
```

In [104]:

```
x = corr[['High Flow (m3/s)']]
sns.heatmap(x,annot=True,cmap="Spectral")
pyplot.savefig("aa3.png", dpi=400, bbox_inches='tight')
```



In [ ]: