Ryan Varnell

Discussion of Predictive Models' Performance

The Data.

Before doing anything else, I want to review the data to determine whether or not any 'cleaning up' is necessary. Our models' accuracy will likely suffer otherwise.

```
DatetimeIndex: 1660 entries, 2017-10-21 13:45:00 to 2017-11-10 06:30:00
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   q_cms            1660 non-null   float64
 1   NO3_mgNL         1660 non-null   float64
 2   SRP_mgPL         1660 non-null   float64
 3   DO_mgL           1660 non-null   float64
 4   DO_sat           1660 non-null   float64
 5   fDOM             1660 non-null   float64
 6   pH               1660 non-null   float64
 7   spCond           1660 non-null   float64
 8   temp             1660 non-null   float64
 9   turb             1660 non-null   float64
 10  precip_mm        1660 non-null   float64
 11  temp_C           1660 non-null   float64
 12  atm_mbar         1660 non-null   float64
 13  PAR_uE           1660 non-null   int64
 14  windSpeed        1660 non-null   float64
 15  gustSpeed        1660 non-null   float64
 16  solarRad_wm2     1660 non-null   int64
 17  windDir          1660 non-null   int64
 18  Rh               1660 non-null   float64
 19  dewPoint         1660 non-null   float64
 20  Basic_Threshold  1660 non-null   int64
dtypes: float64(17), int64(4)
memory usage: 285.3 KB
```

A cursory glance at the data tells us that there are no null values in the set—which means we won't have to worry about substituting any values—and all of the data is already in a float/integer data type, which we'll need for our analysis.

Extreme Outliers. We'll also take a look to see if there's any outliers in the data-set that may require attention. We'll do this by calculating the interquartile range and deeming any values over three ranges above or below the 75^{th} or 25^{th} percentile, respectively, as extreme. For brevity's sake, we'll truncate the output here, but the full picture and relevant code is available in the repository.

	count	mean	std	...	75%	max	outliers
SRP_mgPL	1660.0	0.002986	0.002071	...	0.004041	0.012995	True
turb	1660.0	1.126657	3.389062	...	0.752667	62.535333	True
precip_mm	1660.0	0.089277	0.338782	...	0.000000	6.800000	True
PAR_uE	1660.0	65.742169	140.811254	...	69.000000	1024.000000	True
windSpeed	1660.0	0.313855	0.482035	...	0.500000	2.500000	True
solarRad_wm2	1660.0	32.778313	71.155572	...	32.000000	539.000000	True

Luckily, only a handful of features contain extreme outliers. And to their credit, outliers are to be expected for features such as '*precip_mm*', '*windSpeed*', and '*turb*'. The other features' relevance to our purposes is questionable. '*SRP_mgPL*', '*PAR_uE*', and '*solarRad_wm2*' are related to soil phosphorus, photosynthetically active radiation, and solar radiation, respectively. We will keep this in mind as we continue.

Selecting Features. To determine the features we want to use for our model, we'll calculate the Pearson correlation coefficient to determine if there's any linear relationship present between our chosen dependent variable (we'll be focusing on temperature, so *'temp'*) and the other available features.

	temp
DO_mgL	-0.972560
NO3_mgNL	-0.897186
DO_sat	-0.778274
atm_mbar	-0.435633
Basic_Threshold	-0.217459
solarRad_wm2	-0.077421
windDir	-0.073441
PAR_uE	-0.073142
fDOM	-0.011512
q_cms	0.068921
gustSpeed	0.074531
windSpeed	0.106290
precip_mm	0.148349
turb	0.149258
Rh	0.184314
SRP_mgPL	0.480550
pH	0.487443
spCond	0.584501
temp_C	0.807261
dewPoint	0.879371
temp	1.000000

We can immediately see that there are a few features which have a very strong correlation with temperature. To reduce our dimensionality and avoid over-fitting, we'll be

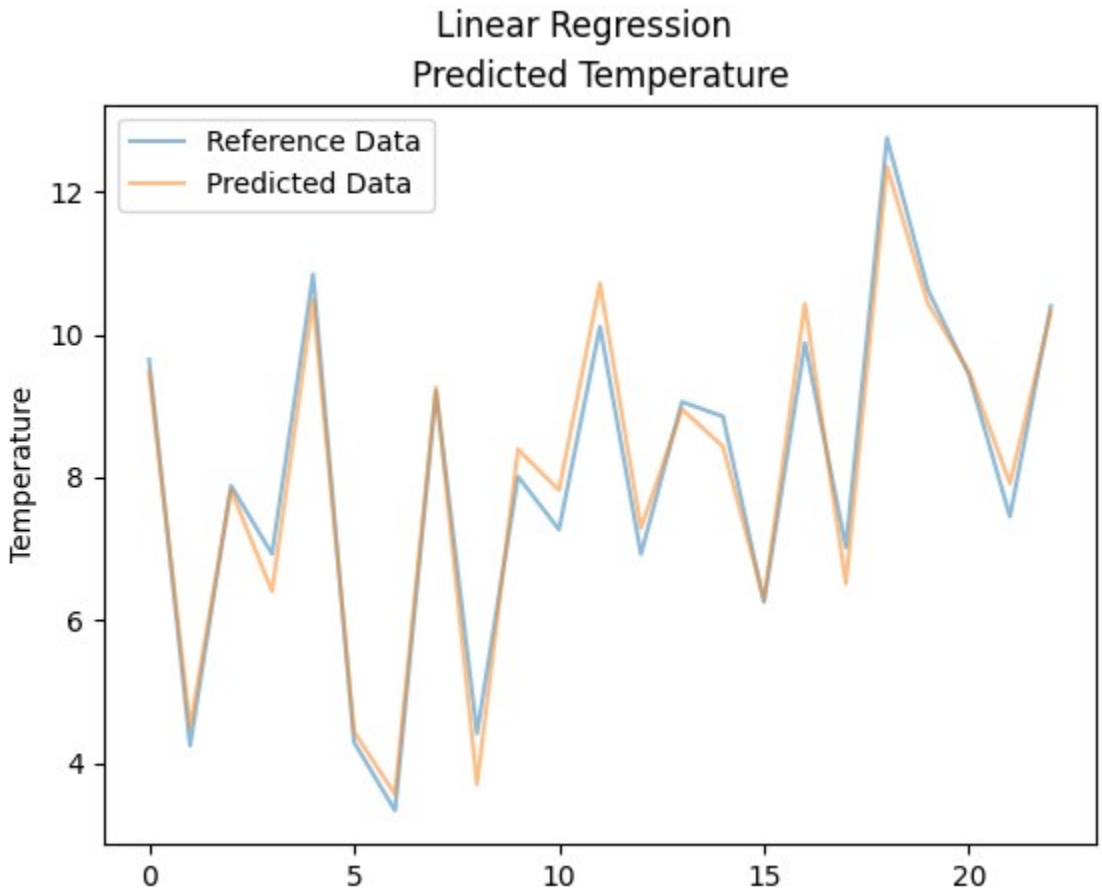
dropping anything with the absolute value of its correlation coefficient less than 0.6, with some leniency. We'll also be ignoring *'temp_C'* for this, because of course it would correlate with the temperature.

Building our First Model.

We'll start by creating and showcasing a basic Linear Regression model, then working to fine-tune it as we see fit. Taking a look at figure 1, we can make the assumption that the model is already performing pretty well. At a glance, each individual predicted data point looks relatively close to the reference point. Our cross validation scores for this model are as such:

```
Linear Regression CV scores: [0.973774   0.97384273 0.9733014  0.9791383
0.97699208 0.97677576
 0.97534507 0.97139934 0.97565612 0.97812507 0.97770152 0.97290375
 0.97579989 0.97488845 0.97643929 0.97675226 0.97295155 0.97607742
 0.97564382 0.97628453 0.97331081 0.98004867 0.97491826 0.97739194
 0.97145104]
Linear Regression mean CV score: 0.9754765230782917
Linear Regression RMSE score: [-0.41915445 -0.42043031 -0.41376851 -0.3896308  -
0.40650833 -0.40534977
 -0.40450367 -0.43200947 -0.41533659 -0.39168104 -0.39296341 -0.42544872
 -0.41542788 -0.4149     -0.39853042 -0.41626476 -0.41237813 -0.40337366
 -0.42721415 -0.38661692 -0.4107594  -0.39263735 -0.41738333 -0.38726028
 -0.43882382]
```

```
Linear Regression mean RMSE score: -0.40953420724025824
```



A 0.97 mean coefficient of determination score is pretty good, and it doesn't seem like we are suffering from over-fitting or anything of the like, but perhaps we can do better. Let's try applying L1 and L2 regularization techniques to our model to see if we can achieve an even higher score.

Lasso Regression. Without tuning any hyper-parameters, our L1 regularized (Lasso) model performs pretty poorly. Our cross-validation coefficient of determination scores land at about 0.34, which is greatly reduced from our previous score. However, after tuning the alpha of our model using a grid search cross-validation method, we achieve a much improved performance:

```
Lasso alpha: 0.01
Lasso CV scores: [0.97105222 0.9744032  0.97238348 0.97561909 0.97606406 0.97448862
 0.9756176  0.97135871 0.97203314 0.9767279  0.97538476 0.96977016
 0.97558316 0.97346465 0.97637744 0.97648632 0.9721784  0.97432618
 0.97331603 0.9743595  0.97284687 0.98019524 0.97354809 0.97606219
```

```
 0.96740902]
Lasso mean CV score: 0.9740422408723972
Lasso RMSE score: [-0.44036794 -0.4159017  -0.42082131 -0.42121455 -0.41462549 -
0.42484074
 -0.4022618  -0.43231621 -0.44517127 -0.40399597 -0.4128731  -0.44937672
 -0.41728399 -0.42650003 -0.39905318 -0.41863892 -0.41823025 -0.41787734
 -0.44716341 -0.40200198 -0.4143141  -0.39119241 -0.42863223 -0.39848646
 -0.46886063]
Lasso mean RMSE score: -0.4212800689605406
```

Of course, both our coefficient of determination and RMSE scores are slightly worse than our original Linear Regression model. Let's try L2 regularization.

Ridge Regression. Before tuning, our L2, or Ridge, regularized model is already performing very well with a mean coefficient of determination score of 0.97—identical to our original Linear Regression model. After tuning, however, the model actually performs slightly *worse* than it did before tuning.

```
Ridge alpha: 0.16
Ridge CV scores: [0.94804887 0.95928663 0.95787035 0.9519628  0.95879109 0.95085835
 0.95671135 0.95503086 0.95271006 0.96010627 0.95754687 0.94941844
 0.95725592 0.95488764 0.95636269 0.95882116 0.95265286 0.9541259
 0.95583881 0.95412692 0.9577411  0.96287171 0.95000596 0.95869283
 0.94609312]
Ridge mean CV score: 0.9551127425063535
Ridge RMSE score: [-0.58993685 -0.52452518 -0.51976514 -0.59124473 -0.5440344  -
0.58963607
 -0.5359911  -0.54170453 -0.57888152 -0.5289463  -0.54221232 -0.58128451
 -0.55210917 -0.55610197 -0.54237078 -0.55400841 -0.54559612 -0.55858247
 -0.57525636 -0.53770482 -0.51686708 -0.53562165 -0.58927149 -0.52346068
 -0.60299979]
Ridge mean RMSE score: -0.5543245374946888
```

Conclusion of Model 1. Though the basic Linear Regression model and the Lasso regularized model performed very similarly, the basic Linear Regression model consistently pulled ever-so-slightly ahead of the Lasso model, while the Ridge model performed worse in every test. This is likely due to the features being carefully chosen before-hand to ensure strong linear relationships between the independent and dependent variables. As there was no over-fitting or extreme outliers present initially, the regularization seems to have only harmed our model's performance.

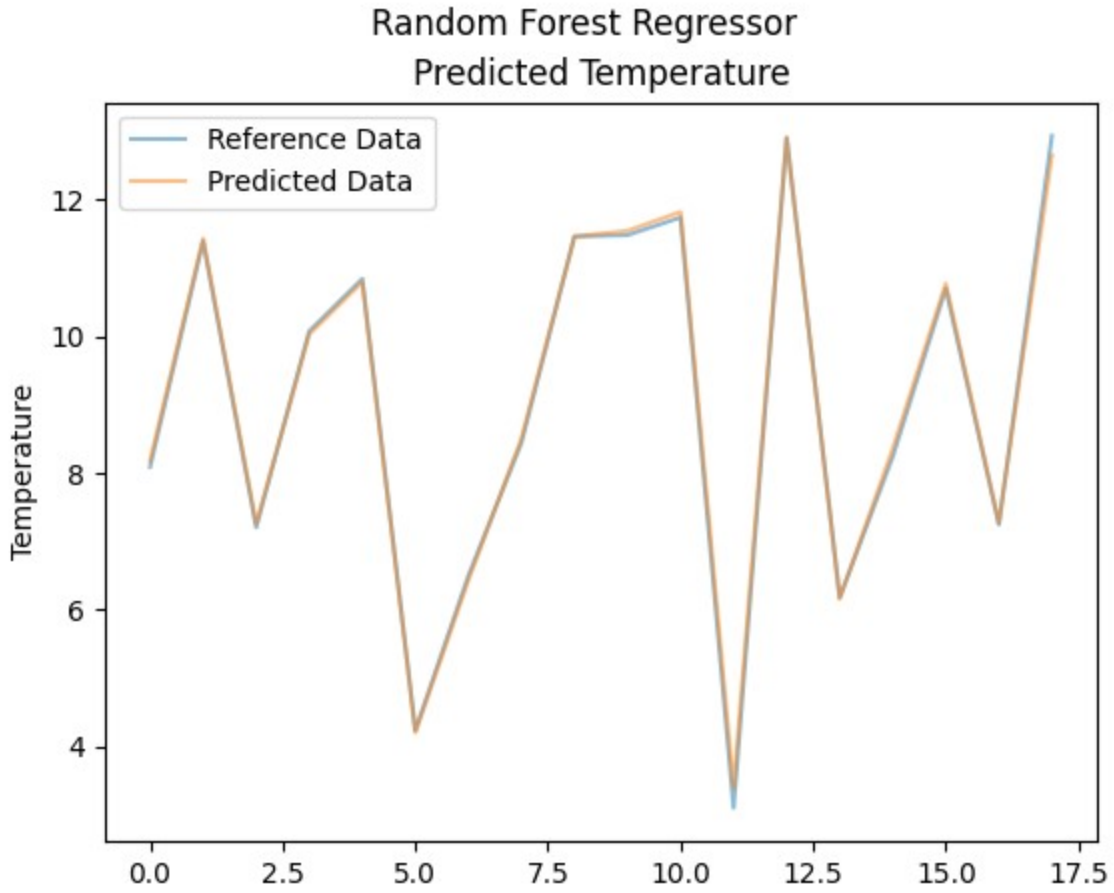
Building our Second Model.

For our second model, we'll be building and training a Random Forest regression model. We won't need to do anything special with the data this time around, as we'll just use the cleaned-up data we gathered for the Linear Regression model.

There's not a whole lot to talk about in the way of building the model, as Sci-Kit Learn's RandomForestRegressor class doesn't require a lot of setup outside of the initial data cleanup we've already done. I will say, however, that I've chosen 10 trees for this model.

The Results. Our Random Forest regression model performs amazingly well, with a mean coefficient of determination score of 0.99 and a mean RMSE score of 0.11 without any further tuning or modifications to the model.

```
Random Forest Regressor CV scores: [0.99795757 0.99852792 0.9978382  0.99834483
0.99794952 0.99614701
 0.99817261 0.99855333 0.99817593 0.99714033 0.99874002 0.99648785
 0.99882956 0.9981917  0.99810364 0.99838899 0.99776892 0.99838035
 0.99696107 0.99721153 0.99778069 0.99843841 0.99820378 0.99842965
 0.99736115]
Random Forest Regressor mean CV score: 0.9979233818870747
Random Forest Regressor RMSE score: [-0.11697186 -0.09973854 -0.11773918 -
0.10974864 -0.12135511 -0.16510426
 -0.11012512 -0.09716063 -0.11369107 -0.14161748 -0.09341069 -0.15317188
 -0.0913613  -0.11133776 -0.113065   -0.10957914 -0.11843543 -0.10495772
 -0.15090427 -0.13257075 -0.1184484  -0.1098474  -0.11169571 -0.10206344
 -0.1334143 ]
Random Forest Regressor mean RMSE score: -0.11790060316698799
```



Conclusion. There's not much to say here. The random forest regression model vastly outperformed all three of our linear regression models with a fraction of the effort. That's not to say that will always be the case, as it greatly depends on the data and the user's expected results. In this case, however, our random forest is the obvious choice.