

Predicting Rainfall

Mark Woods

1/25/16

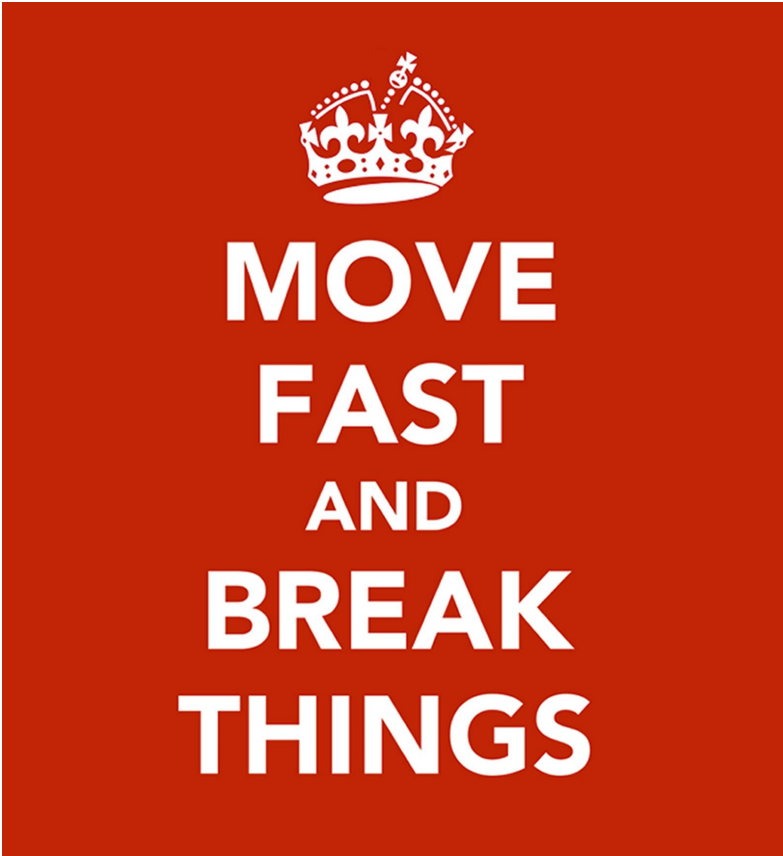
Data Science: Pick two

Simple

Fast

Accurate

Spoiler Alert →



**MOVE
FAST
AND
BREAK
THINGS**

So what are we looking at?

Rainfall data along with associated radar telemetry data for midwestern states over a period of ~20 days.

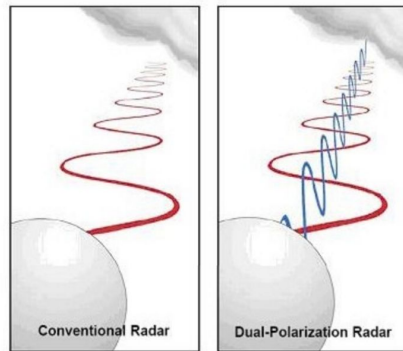
Project Design philosophy:

We will take a completely pragmatic approach, and treat theoretical underpinnings as unknown, and data readings as a black box. In other words...

Competition Description

Rainfall is highly variable across space and time, making it notoriously tricky to measure. Rain gauges can be an effective measurement tool for a specific location, but it is impossible to have them everywhere. In order to have widespread coverage, data from weather radars is used to estimate rainfall nationwide. Unfortunately, these predictions never exactly match the measurements taken using rain gauges.

Recently, in an effort to improve their rainfall predictors, the U.S. National Weather Service upgraded their radar network to be polarimetric. These polarimetric radars are able to provide higher quality data than conventional [Doppler radars](#) because they transmit radio wave pulses with both horizontal and vertical orientations.



Dual pulses make it easier to infer the size and type of precipitation because rain drops become flatter as they increase in size, whereas ice crystals tend to be elongated vertically.

In this competition, you are given snapshots of polarimetric radar values and asked to predict the hourly rain gauge total. A word of caution: many of the gauge values in the training dataset are implausible (gauges may get clogged, for example). More details are on the [data page](#).

So what are we looking at?

Rainfall data along with associated radar telemetry data for midwestern states over a period of ~20 days.

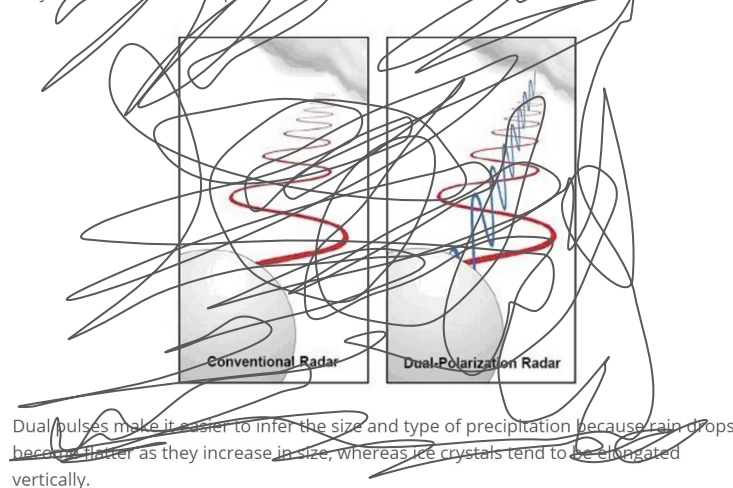
Project Design philosophy:

We will take a completely pragmatic approach, and treat theoretical underpinnings as unknown, and data readings as a black box. In other words...

Competition Description

Rainfall is highly variable across space and time, making it notoriously tricky to measure. Rain gauges can be an effective measurement tool for a specific location, but it is impossible to have them everywhere. In order to have widespread coverage, data from weather radars is used to estimate rainfall nationwide. Unfortunately, these predictions never exactly match the measurements taken using rain gauges.

Recently, in an effort to improve their rainfall predictors, the U.S. National Weather Service upgraded their radar network to be polarimetric. These polarimetric radars are able to provide higher quality data than conventional [Doppler radars](#) because they transmit radio wave pulses with both horizontal and vertical orientations.



Dual pulses make it easier to infer the size and type of precipitation because rain drops become flatter as they increase in size, whereas ice crystals tend to be elongated vertically.

In this competition, you are given snapshots of polarimetric radar values and asked to predict the hourly rain gauge total. A word of caution: many of the gauge values in the training dataset are implausible (gauges may get clogged, for example). More details are on the [data page](#).

Let's look at some attributes of the data

It's pretty sparse.

In reality, only about 20% of the recorded rainfall amounts have complete radar telemetry behind them.

```
df.head()
```

| | Id | minutes_past | radardist_km | Ref | Ref_5x5_10th | Ref_5x5_50th | Ref_5x5_90th | RefComposite | RefComposite_5x5_10th | RefComposite_5x5 |
|---|----|--------------|--------------|-----|--------------|--------------|--------------|--------------|-----------------------|------------------|
| 0 | 1 | 3 | 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | 1 | 16 | 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 1 | 25 | 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | 1 | 35 | 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | 1 | 45 | 10 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

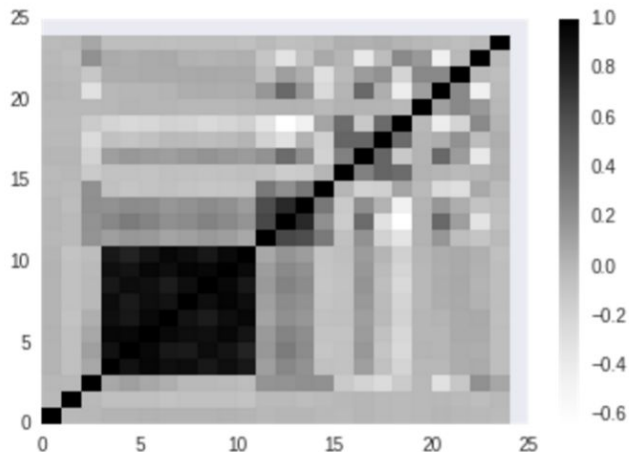
For simplicity, we will just impute these with the mean values.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100000 entries, 0 to 99999
Data columns (total 24 columns):
Id                100000 non-null int64
minutes_past      100000 non-null int64
radardist_km      100000 non-null float64
Ref               45076 non-null float64
Ref_5x5_10th      38462 non-null float64
Ref_5x5_50th      45211 non-null float64
Ref_5x5_90th      53336 non-null float64
RefComposite      48109 non-null float64
RefComposite_5x5_10th  42296 non-null float64
RefComposite_5x5_50th  48195 non-null float64
RefComposite_5x5_90th  55457 non-null float64
RhoHV             36125 non-null float64
RhoHV_5x5_10th    30947 non-null float64
RhoHV_5x5_50th    36098 non-null float64
RhoHV_5x5_90th    42333 non-null float64
Zdr               36125 non-null float64
Zdr_5x5_10th      30947 non-null float64
Zdr_5x5_50th      36098 non-null float64
Zdr_5x5_90th      42333 non-null float64
Kdp               31356 non-null float64
Kdp_5x5_10th      26480 non-null float64
Kdp_5x5_50th      31402 non-null float64
Kdp_5x5_90th      36688 non-null float64
Expected          100000 non-null float64
dtypes: float64(22), int64(2)
```

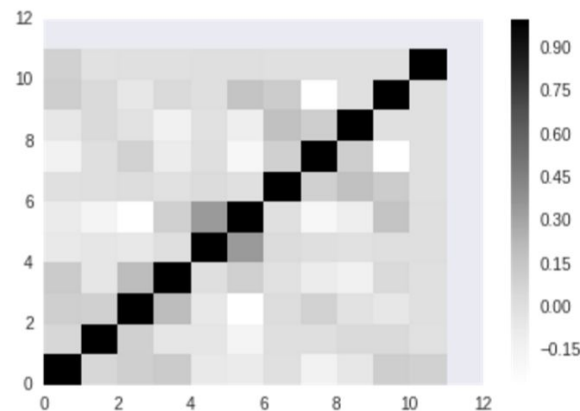
Removing correlated variables

Here we're looking at the correlation matrix of the feature space

The first few are just labels, the last one is our rainfall values. So let's clean it.



Ew. Large blocks of highly correlated features.

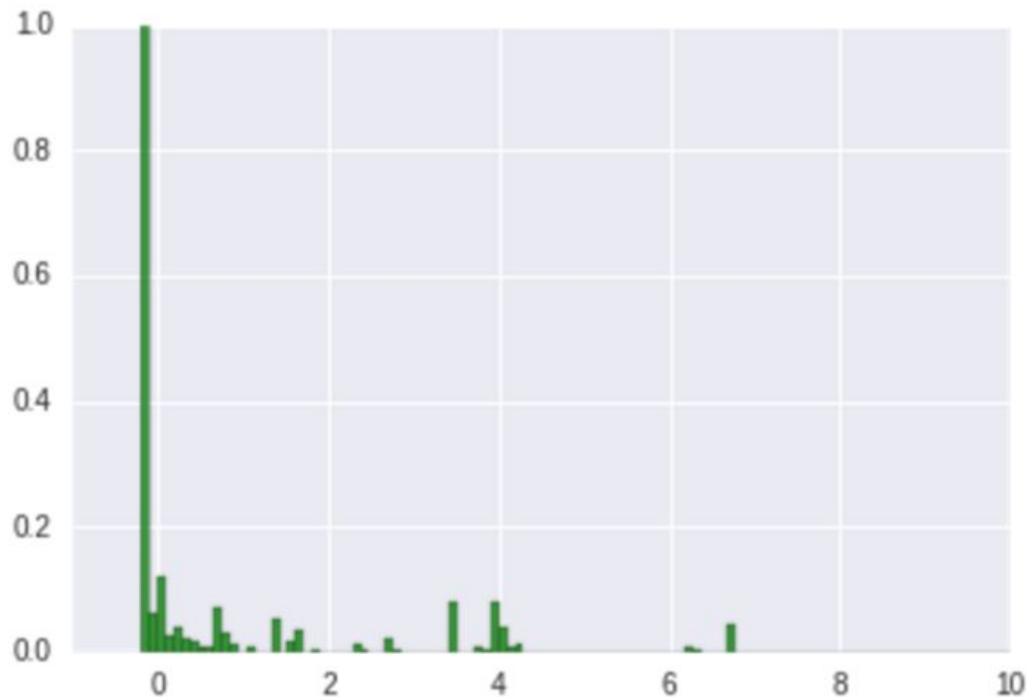


Horray! Cleaner!

Let's look at our standardized rainfall values

Obviously not quite Gaussian as would be the ideal starting point, but for now we will run with it and hope for the best.

This is with 0 mean and 1 std.



That was quite a tail. Let's look at outliers.

Looking at outliers in the data, we're using a simplified metric, where we count the number of values outside 3 std and compare it to the expected amount.

```
#outlier detection time

outliers=0.
for i in df['Expected']:
    if i>3 or i<-3:
        outliers+=1

print outliers
print outliers/100000*100, "%"
print 'for a normal distrubtion, this should be .2%'
```

2780.0

2.78 %

for a normal distrubtion, this should be .2%

How shall we handle outliers?

We have a few options:

1. Eliminate those observations
 - a. Pros: No more outliers.
 - b. Cons: No more statistically guaranteed data points
2. Account for them in a statistically precise way
 - a. Pros: Probably the most elegant and accurate way to deal with them
 - b. Cons: Complicated, time-consuming
3. See no evil, Hear no evil, Speak no evil
 - a. Pros: Quick, cheap, and dirty. Like having McDonalds for dinner, it'll fill your stomach, but you won't necessarily feel good afterwards.
 - b. Cons: The outliers will remain and will skew the results

How to deal with outliers?

We have

1. Eliminate

a.

b.

2. Accept

a.

b.

3. See no evil, Hear no evil, Speak no evil

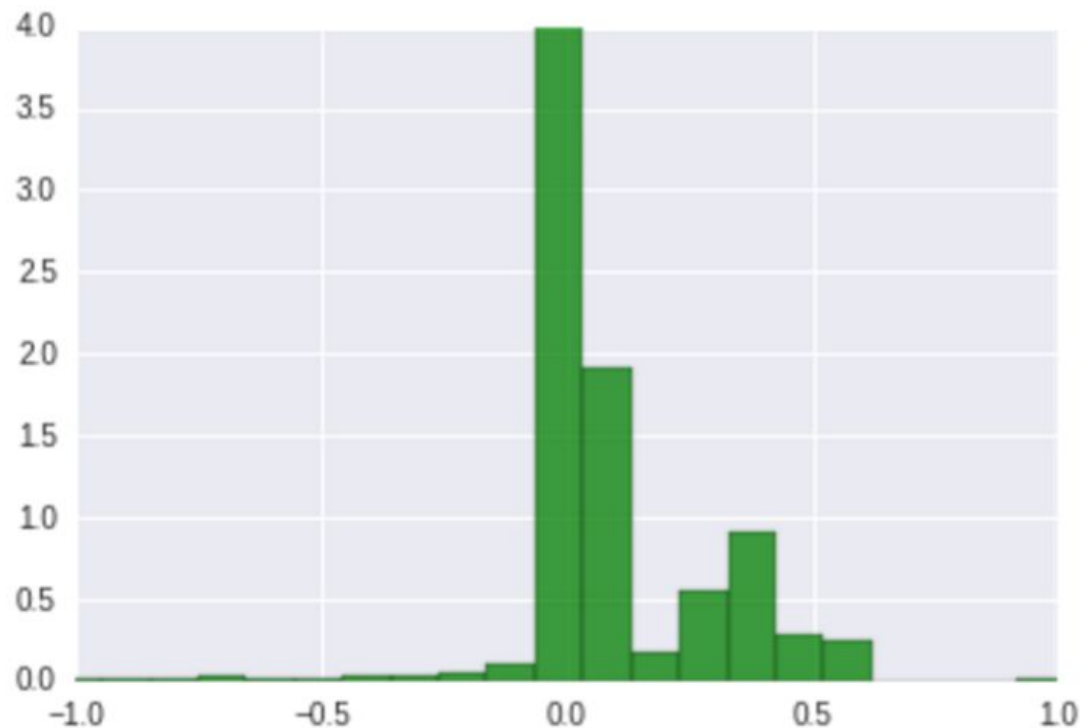
a. Pros: Quick, cheap, and dirty. Like having McDonalds for dinner, it'll fill your stomach, but you won't necessarily feel good afterwards.

b. Cons: The outliers will remain and will skew the results

i. *But we can try to reduce this disadvantage by helping ensure that outliers make a statistically insignificant difference in our predictions. Perhaps by.....*



K-Nearest Neighbors Regression with large K (300)

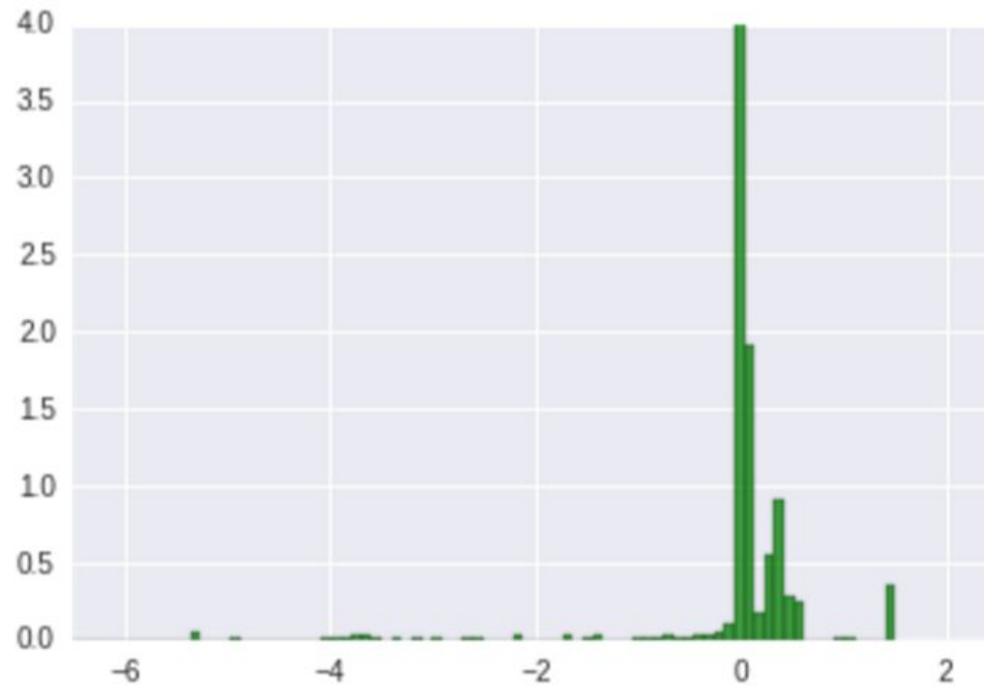


Here we're looking at a trained KNN model with a somewhat optimized K value of 300.

This is specifically a probability distribution of the absolute errors of the model in an arbitrary 75/25 train/test split.

Big peak at 0 = 😊

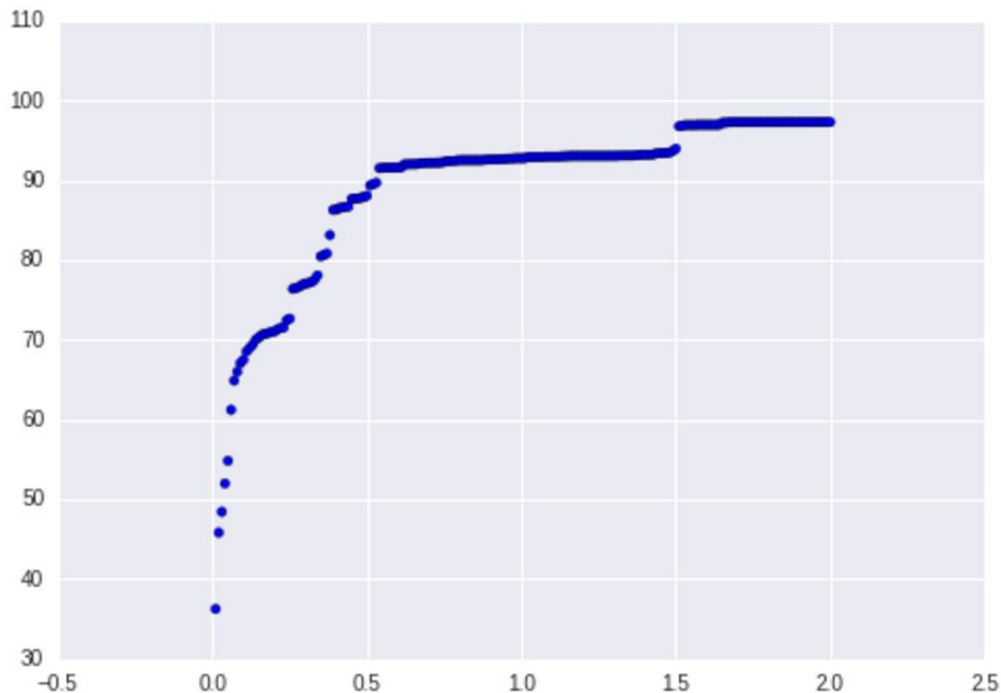
...but it's still not perfect.



What to tell your boss

Y axis shows what percentage of predictions lie within $\pm X$ standard deviations of the observed value.

So, I'll report that ~90% of the predictions lie within $\pm .5$ std of the true values.

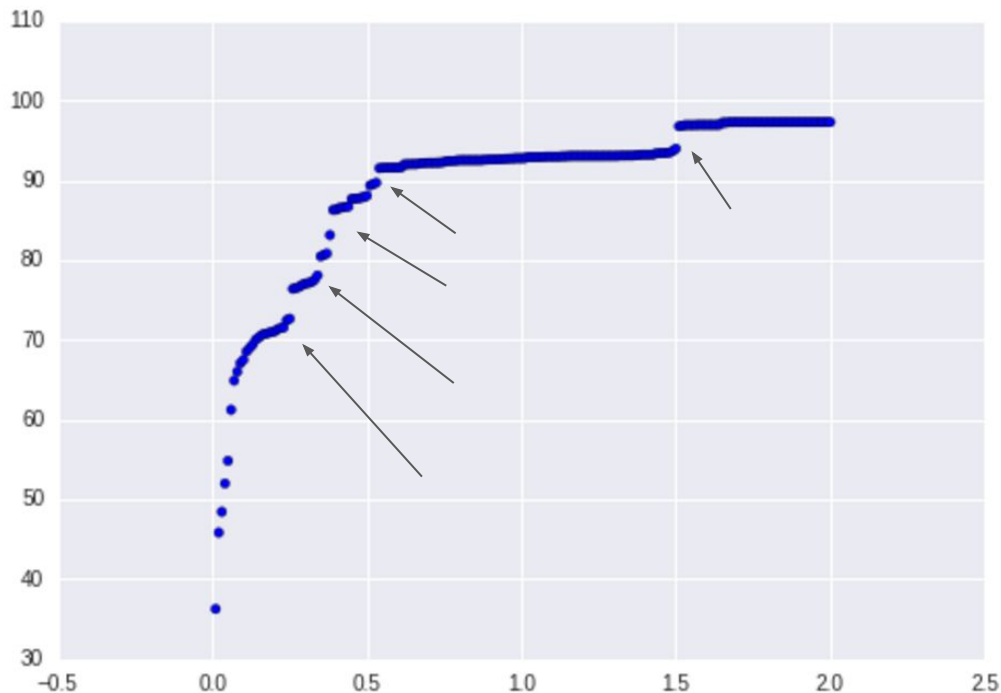


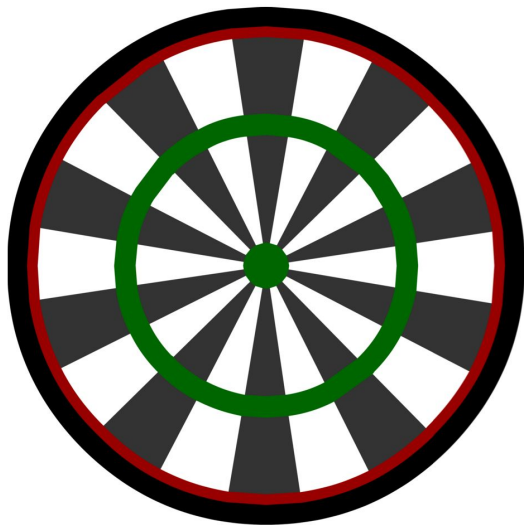
What to tell your boss

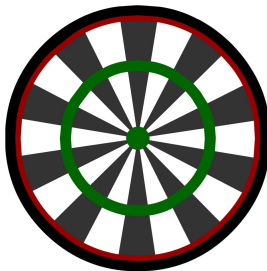
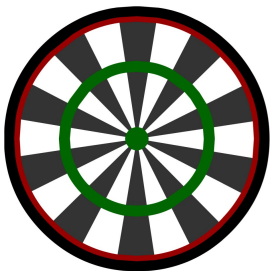
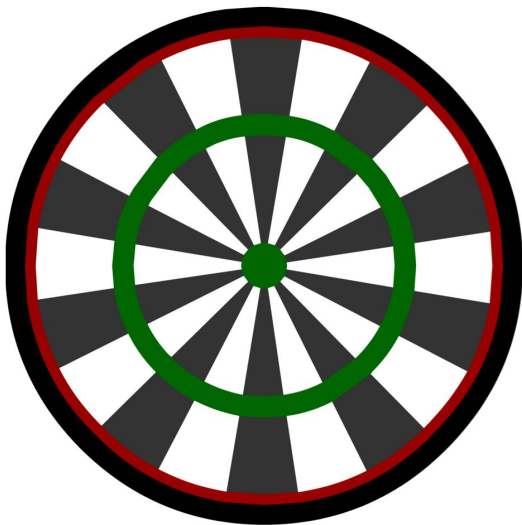
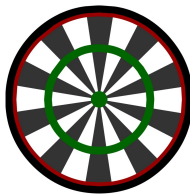
Y axis shows what percentage of predictions lie within $\pm X$ standard deviations of the observed value.

So, I'll report that ~90% of the predictions lie within $\pm .5$ std of the true values.

But we have some interesting cusp features in this chart that suggest unaccounted for subsets present in the data







Where do we go from here?

- Don't take a black box approach, but instead try to glean meaning from the given data features
- Better imputation/handling of missing data
- PCA to remove highly correlated features instead of just cutting them out
- Better handling of outliers
- Extracting subsets of data and allowing them separate handling
- And of course, trying different models also.





That's all Folks!