Cherry Blossom Prediction 2023

REDACTED and REDACTED

2023-02-28

Narrative

When we heard about the cherry blossom prediction competition, our thoughts lept to prediction markets and "wisdom-of-the-crowds" methods. According to the work of Philip Tetlock and his collaborators, trained human forecasters combined in a weighted average with mass crowd predictions consistently outperform even the best machine learning and statistical models in similar forecasting efforts.

The core of the idea is quite simple. Each person in the crowd has some amount of true and useful information about the desired prediction, and some amount of incorrect information (bias and random error). By aggregating the predictions of each person, randomness and random bias cancel each other out, and the true and useful begins to dominate the prediction. Trained, practiced forecasters contribute a larger amount of true information on average, so in a weighted average against an untrained crowd they receive a higher weighting. Tetlock's organization, the Good Judgment Project, has empirically discovered these weightings over years of study.

Time, resource, and rule constraints precluded our ability to garner a prediction in such a way. However, we drew inspiration from the core of the idea to create our own "wisdom of the crowds" method. We decided to collect as many previously-submitted cherry blossom bloom prediction models as possible, update them with the latest data, and re-run them to obtain new predictions. We would run the models on past data to obtain their past predictions, then use these past predictions to create a weighted average of the models. Finally, we would use the weightings and the future predictions of the models to generate our own future predictions.

The core of the idea remains the same as the wisdom-of-the-crowds model: each model is based on similar (but usually not identical) data. Each model uses different methods to achieve its predictions. We therefore assume that each model contains some true and useful information, some bias, and some random error. By creating a weighted average of their predictions, we hope to reduce some bias inherent in the models without an increase in variance, thus improving the bias-variance tradeoff.

Finding the models proved to be a challenge. We were not able to look at submitted competition models directly. Instead, we searched through forked github repositories from the main GMU-CherryBlossomCompetition branch. Parsing through nearly a hundred of these, we found those which warranted further investigation. We then explored these models one-by-one to determine which might be viable candidates for our final model.

Candidates were evaluated for their working code, substantive modeling efforts, variety in data sources, and types of models. Due to the enormous time required to parse and update each model, and the limited variability in available models, we chose five models for our final submission:

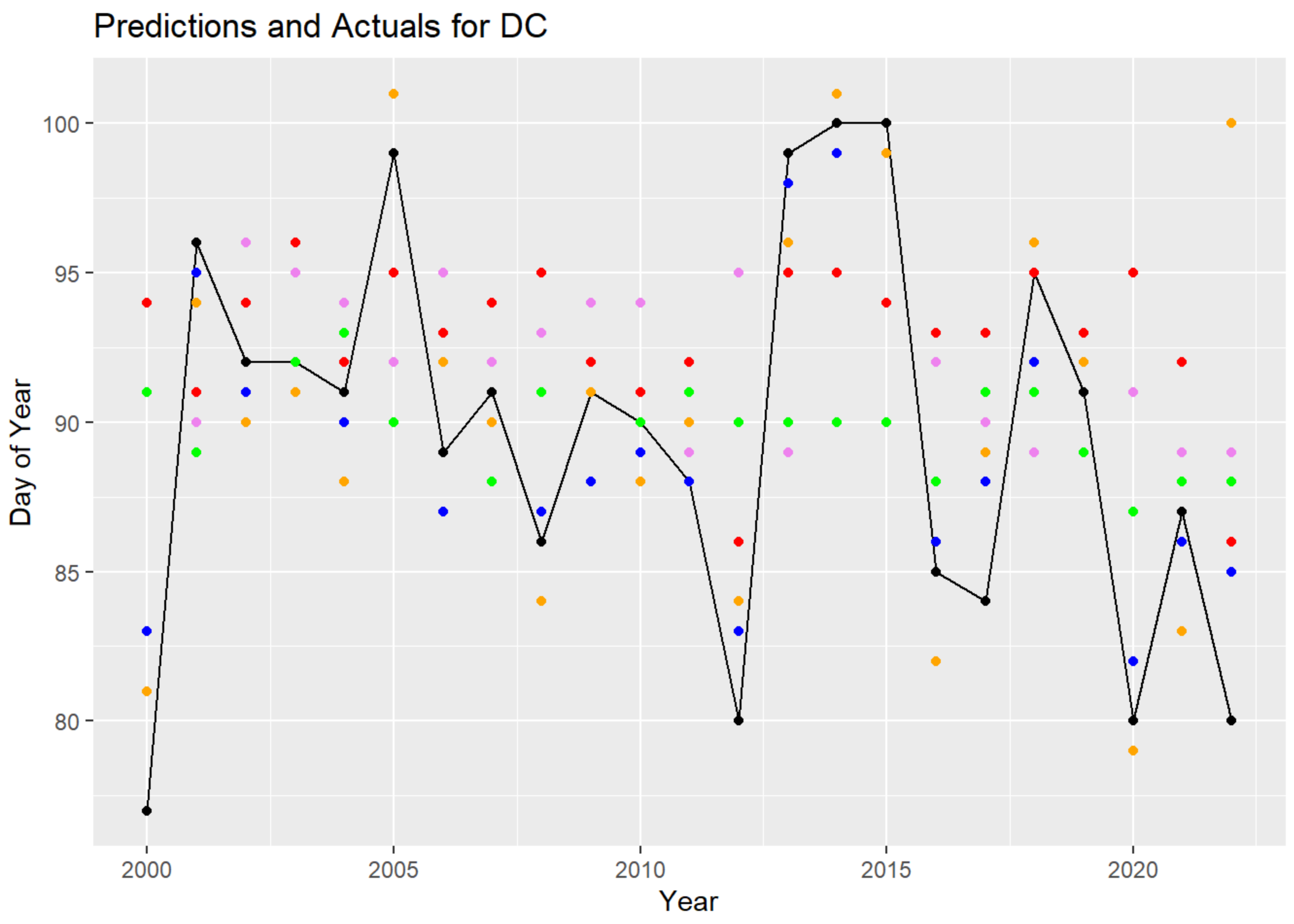
- stevenlio88
- siyueyang (previous award winner)
- mattharding23 (previous award winner)
- kenkoonwong (previous award winner)

- Daria-Kearny

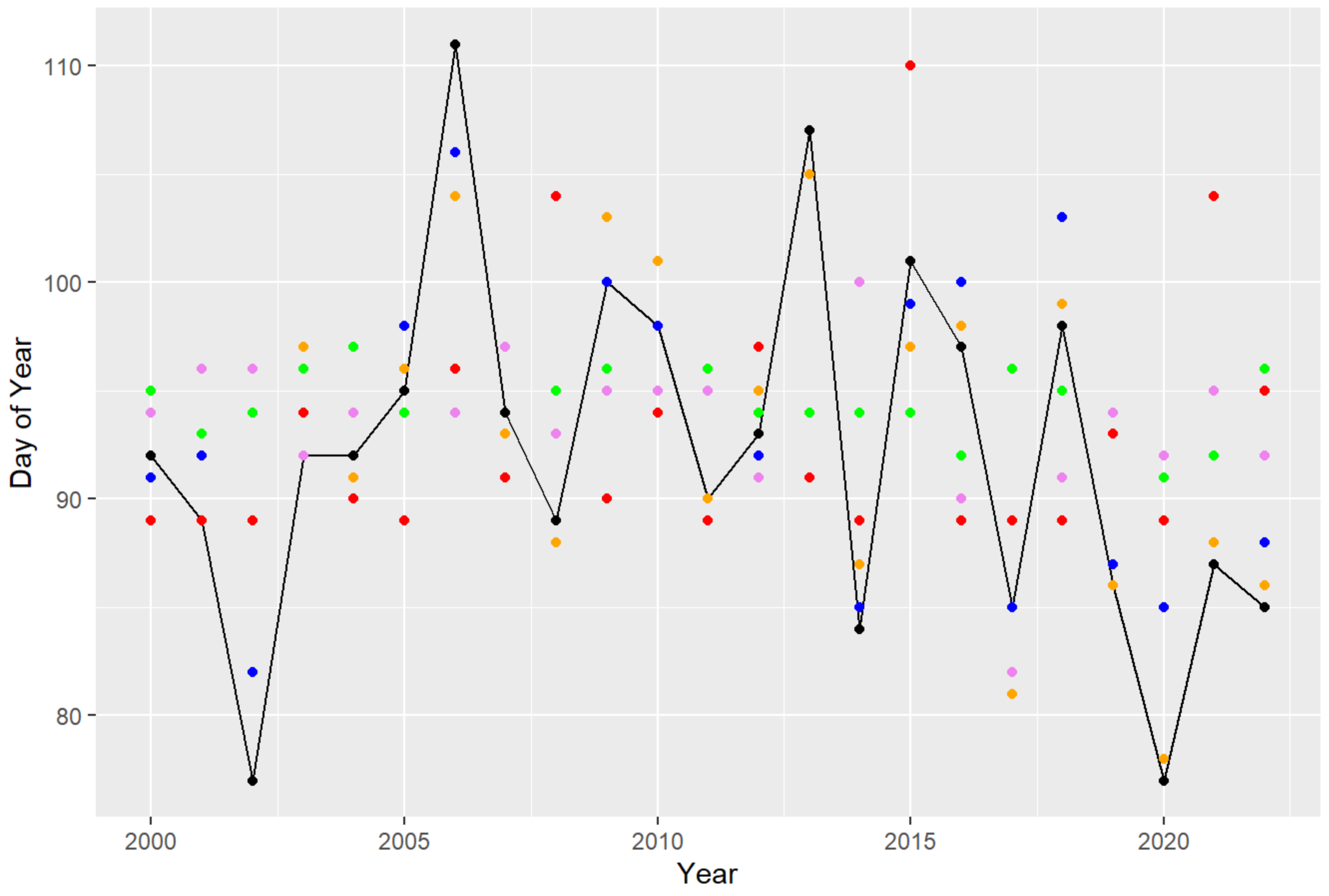
  Our weighted average outperformed its constituent models in terms of mean squared error for every city (except Vancouver, where there are no actuals to compare against). In the future, we feel that a similar effort with more models and possibly with more sophisticated methods for determining model weights should outperform any individual model. Potential methods include recurrent neural networks, LSTM, or various methods of time series regression.

Basic Visualizations

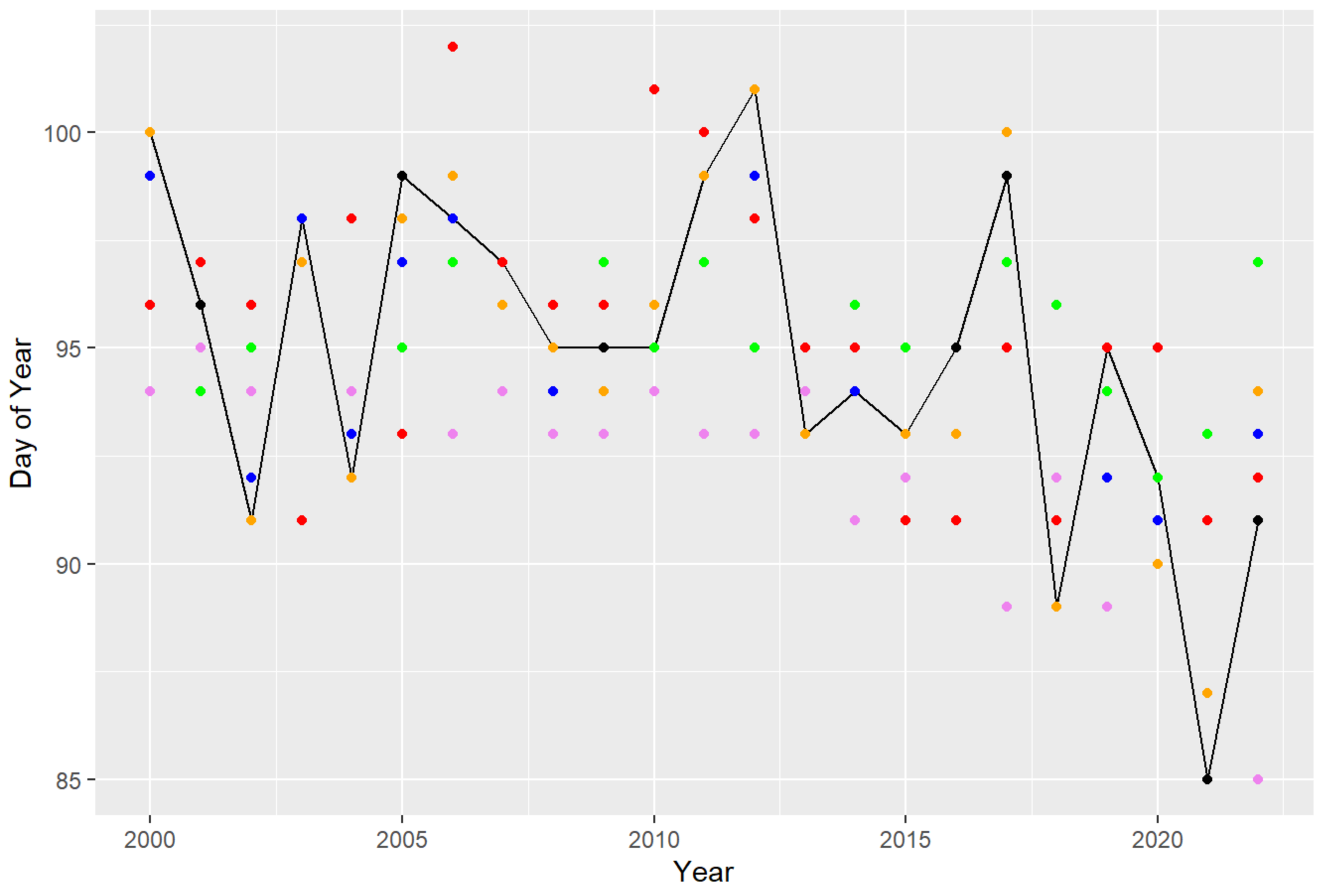
our first explorations gave us hope. DC's actual numbers are close to the predictions made for past years by the models. The same is true for both Liestal and Kyoto. Vancouver's actual bloom dates weren't part of the main data set and uncovering them would have been a misappropriation of team resources.

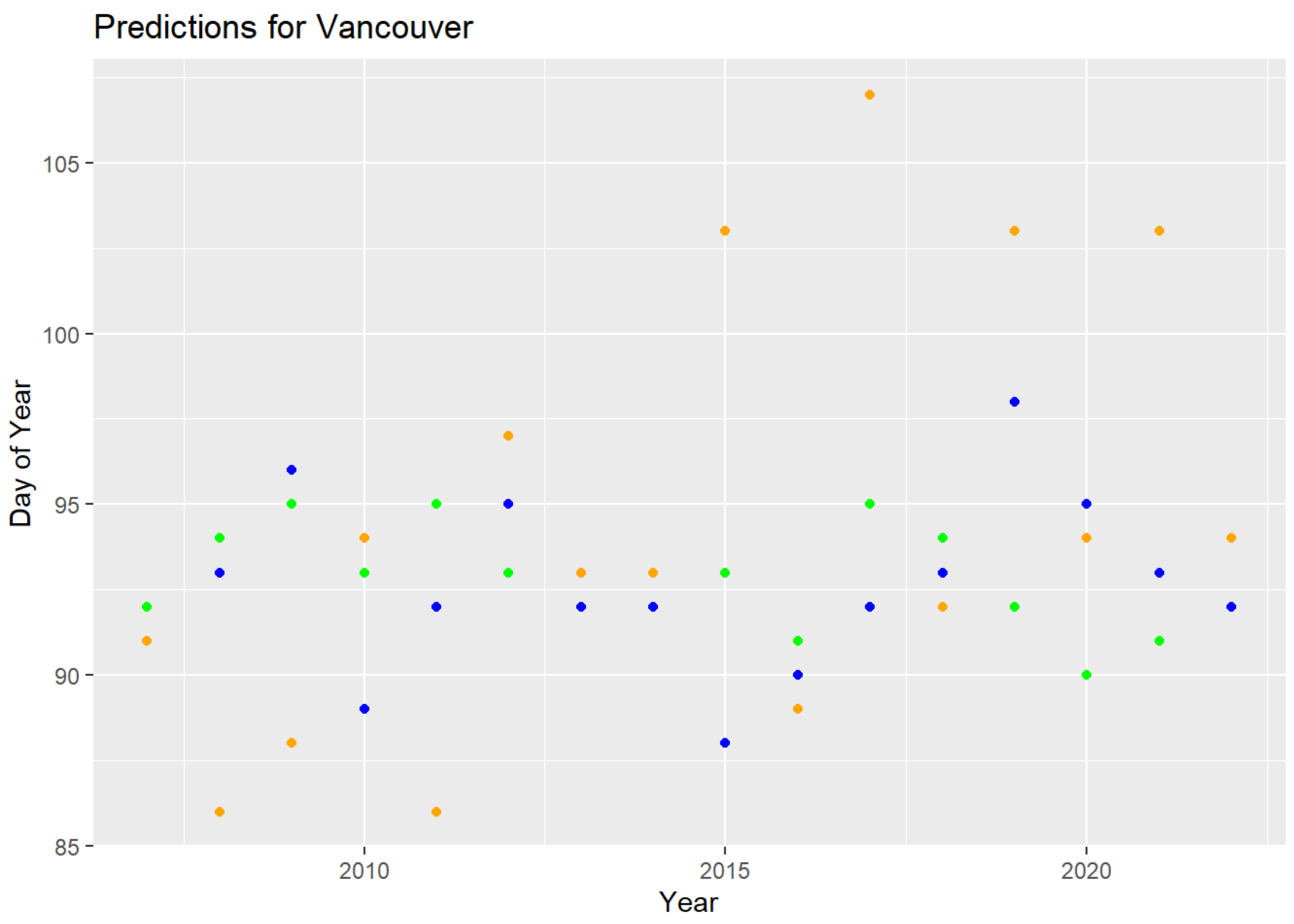


Predictions and Actuals for Liestal



Predictions and Actuals for Kyoto





Linear Modeling

Since we were looking only for a weighted average of the models, we forced the intercept to 0. This also reduced the danger of overfitting. Between this and the low amount of data available acrosss all 5 models, we chose not to split into training and test sets.

```
##
## Call:
## lm(formula = dc_day ~ 0 + dc_dk + dc_kkw + dc_mh + dc_sl + dc_syy,
##     data = dplyr::filter(models_and_actuals, year < 2023))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0557 -0.9663 -0.2409  1.5290  3.9896
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## dc_dk   1.246536   0.208730   5.972 1.19e-05 ***
## dc_kkw -0.143256   0.325797  -0.440    0.665
## dc_mh  -0.095088   0.240039  -0.396    0.697
## dc_sl  -0.011156   0.148041  -0.075    0.941
## dc_syy  0.003314   0.244047   0.014    0.989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.475 on 18 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9992
## F-statistic:  6069 on 5 and 18 DF,  p-value: < 2.2e-16
```

```
##           mse_names    dc.mse
## 1 DC Regression MSE  4.795099
## 2         DC DK MSE  6.217391
## 3        DC KKW MSE 41.695652
## 4         DC MH MSE 53.652174
## 5         DC SL MSE 23.478261
## 6        DC SYY MSE 44.478261
```

```
##
## Call:
## lm(formula = liestal_day ~ 0 + liestal_dk + liestal_kkw + liestal_mh +
##     liestal_sl + liestal_syy, data = dplyr::filter(models_and_actuals,
##     year < 2023))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1279 -1.4636 -0.5736  1.3429  6.1546
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## liestal_dk   0.36865    0.19229   1.917  0.07586 .
## liestal_kkw  0.12826    0.17673   0.726  0.47995
## liestal_mh  -0.19816    0.15198  -1.304  0.21331
## liestal_sl   0.71400    0.17332   4.120  0.00104 **
## liestal_syy -0.02265    0.12596  -0.180  0.85988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.545 on 14 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9993
## F-statistic:  5067 on 5 and 14 DF,  p-value: < 2.2e-16
```

```
##                mse_names liestal.mse
## 1 liestal Regression MSE     4.77408
## 2         liestal DK MSE    13.13043
## 3        liestal KKW MSE    67.43478
## 4         liestal MH MSE          NA
## 5         liestal SL MSE          NA
## 6        liestal SYY MSE    82.34783
```

```
##
## Call:
## lm(formula = kyoto_day ~ 0 + kyoto_dk + kyoto_kkw + kyoto_mh +
##     kyoto_sl + kyoto_syy, data = dplyr::filter(models_and_actuals,
##     year < 2023))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8853 -0.5375 -0.1215  0.5782  2.1530
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## kyoto_dk   0.02673    0.22289   0.120   0.9059
## kyoto_kkw -0.18744    0.11892  -1.576   0.1334
## kyoto_mh   0.29894    0.10422   2.868   0.0107 *
## kyoto_sl   1.01358    0.15284   6.632 4.24e-06 ***
## kyoto_syy -0.14176    0.09712  -1.460   0.1626
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.039 on 17 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 3.657e+04 on 5 and 17 DF,  p-value: < 2.2e-16
```

```
##              mse_names  kyoto.mse
## 1 kyoto Regression MSE  0.8347397
## 2         kyoto DK MSE  4.1304348
## 3        kyoto KKW MSE 13.6521739
## 4         kyoto MH MSE         NA
## 5         kyoto SL MSE  1.3043478
## 6        kyoto SYY MSE 13.8260870
```

The regression has the lowest MSE among all of the models, consistently. There is nothing unexpected about this, nor is this a particular sign of model effectiveness; regression models are built to find reduced MSE among a set of points.

```
##    year washingtondc  liestal    kyoto vancouver
## 1  2023     89.53342 84.35243 83.43323      94.6
## 2  2024     89.79939 84.80990 90.22933      93.2
## 3  2025     89.75300 86.36616 86.17502      94.0
## 4  2026     89.91857 92.92042 81.10713      95.0
## 5  2027     89.94849 85.78043 88.20218      93.6
## 6  2028     89.95964 83.63843 82.30815      95.6
## 7  2029     89.90387 92.07816 90.22933      95.0
## 8  2030     89.85924 90.13431 88.57706      94.6
## 9  2031     89.95964 81.69459 86.06353      96.8
## 10 2032     89.94849 83.12259 82.16992      95.8
```

Finally, we can see our predictions for 2023 and forward. As actual values for Vancouver are not generally available, values for the prediction were found by taking an unweighted average among the models.