Notebook

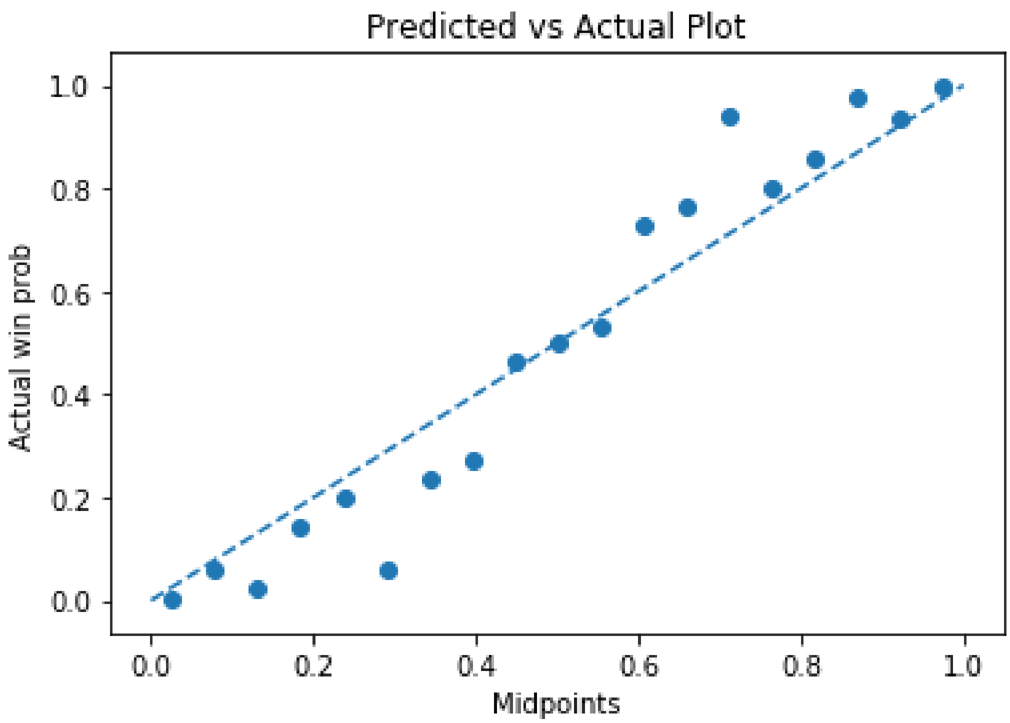
April 20, 2019

Now make a scatterplot using `midpoints` as the x variable and `fraction_outcome` as the y variable. Draw a dashed line from `[0,0]` to `[1,1]` to mark the line y=x.

```
In [53]: %matplotlib inline
         import matplotlib.pyplot as plt

         plt.scatter(midpoints, fraction_outcome.values)
         plt.title('Predicted vs Actual Plot')
         plt.xlabel('Midpoints')
         plt.ylabel('Actual win prob')
         plt.plot([0,1], [0,1], '--')
```

```
Out[53]: [<matplotlib.lines.Line2D at 0x7f3a61d09c18>]
```



0.0.1 Question 5: adding error bars

If you did things correctly, it should look like fivethirtyeight has done "pretty" well with their forecasts: the actual fraction of wins tracks closely with the predicted number. But how do we decide what's "good enough"? Consider this example: I correctly predict that a coin is fair (e.g. that it has a 50% chance of heads, 50% chance of tails). But if I flip it 100 times, I can be pretty sure it won't come up heads exactly 50 times. The fact that it didn't come up heads exactly 50 times doesn't make my prediction incorrect.

To assess how reasonable the predictions are, I need to quantify the uncertainty in my estimate. It's reasonable to assume that within each bin, k, the observed number of wins, $Y_k \sim Bin(n_k, p_k)$, where n_k is the number of elections and p_k is the predicted win probability in bin k.

Classical results tell us that the obseved fraction of wins in bin k, $\hat{p} = \frac{Y_k}{n_k}$ has variance $\text{Var}(\hat{p}_k) = \frac{p_k(1-p_k)}{n_k} \approx \frac{\hat{p}_k(1-\hat{p}_k)}{n_k}$. The standard deviation of the Binomial proportion then is $\hat{\sigma}_k \approx \sqrt{\frac{\hat{p}_k(1-\hat{p}_k)}{n_k}}$.

If we use the normal approximation to generate a confidence interval, then the 95% interval has the form $\hat{p}_k \pm 1.96\hat{\sigma}_k$.

Create a new "aggregated" dataframe. This time, group `election_sub` by the `bin` and compute both the average of the `probwin_outcome` (`mean`) and the number of observations in each bin (`count`) using the `agg` function. Call this new data frame, `election_agg`.

```
In [55]: election_agg = election_sub.groupby('bin')['probwin_outcome'].agg(['mean','count'])
         election_agg
```

```
Out[55]:                        mean  count
         bin
         (0.0, 0.0526]      0.001715    583
         ... Omitting 14 lines ...
         (0.842, 0.895]     0.976744     43
         (0.895, 0.947]     0.937500     32
         (0.947, 1.0]       0.998478    657
```

Use the mean and count columns of election_agg to create a new column of election_agg titled err, which stores $1.96 \times \hat{\sigma}_k$ in each bin k.

In [56]:
```
#election_agg['count']
var = (election_agg['mean']*(1-election_agg['mean']))/election_agg['count']
error = (var**0.5)*1.96
election_agg['err'] = error
election_agg
```

Out[56]:
```
                        mean  count       err
bin
(0.0, 0.0526]       0.001715    583  0.003359
... Omitting 14 lines ...
(0.842, 0.895]      0.976744     43  0.045048
(0.895, 0.947]      0.937500     32  0.083870
(0.947, 1.0]        0.998478    657  0.002981
```

0.0.2 Question 7: understanding confidence intervals

Are the 95% confidence intervals generally larger or smaller for more confident predictions (e.g. the predictions closer to 0 or 1). What are the factors that determine the length of the confidence intervals?

SOLUTION HERE Higher confidence indicates a larger interval, since you are more confident that the true value will fall between a larger range. Sample size is one big factor that can determine the length of the confidence intervals as well as overall standard error, i.e. the variance. A smaller sample size would give you a larger interval.

In [60]:
```
# Input: a pandas dataframe with a numeric column named `probwin`
# Output: a pandas dataframe with the same columns, with an additional column named `absdiff`
def abs_diff(x):
    x['absdiff'] = max(x['probwin']) - min(x['probwin'])
    return x
```

We can use this function to compute the difference between the maximum and minimum predicted with probabilities for every candidate. To do so, group `election_sub` by `candidate` and `apply` the function `abs_diff`. Find the index of the largest difference in `diff_dataframe` and store it in `max_idx`. Do this using `np.nanargmax` function. This function finds the *index* of the largest value, ignoring any missing values (nans).

In [66]:
```
diff_dataframe =election_sub.groupby('candidate').apply(abs_diff)
diff_dataframe
max_idx = np.nanargmax(diff_dataframe['absdiff'])
#diff_dataframe.iloc[max_idx]
```

Out[66]:
```
year                                2018
office                             House
state                                 KS
... Omitting 10 lines ...
bin                       (0.842, 0.895]
absdiff                          0.65428
Name: 1890, dtype: object
```

Did the candidate win or lose the election?

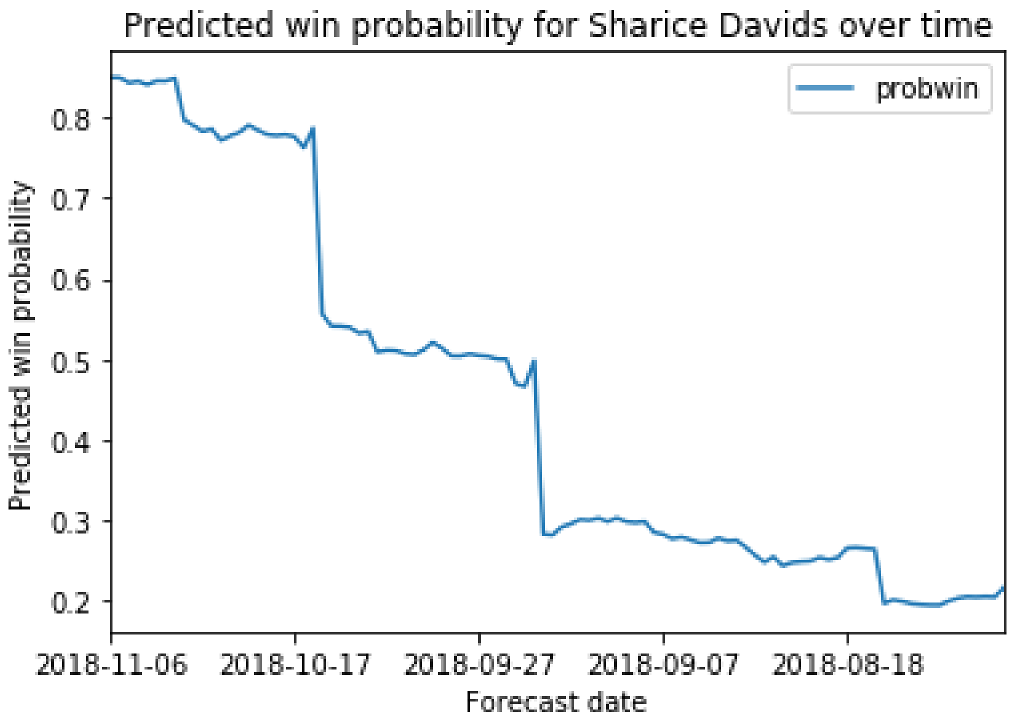
```
In [71]: election_sub[(election_sub.candidate == 'Sharice Davids')]

Out[71]:         year office state  district special election_date forecast_date  \
        1890    2018  House    KS       3.0   False    2018-11-06    2018-11-06
        252855  2018  House    KS       3.0   False    2018-11-06    2018-08-11
        ... Omitting 4 lines ...
                actual_voteshare  probwin  probwin_outcome             bin
        1890                 NaN  0.84994                1  (0.842, 0.895]
        252855               NaN  0.19566                1  (0.158, 0.211]
```

Now create a lineplot with forecast date on the x-axis and the predicted win probability on the y-axis.

```
In [76]: predicted_probs.plot.line(x='forecast_date', y='probwin')
         plt.title('Predicted win probability for Sharice Davids over time')
         plt.xlabel('Forecast date')
         plt.ylabel('Predicted win probability')
```

```
Out[76]: Text(0, 0.5, 'Predicted win probability')
```

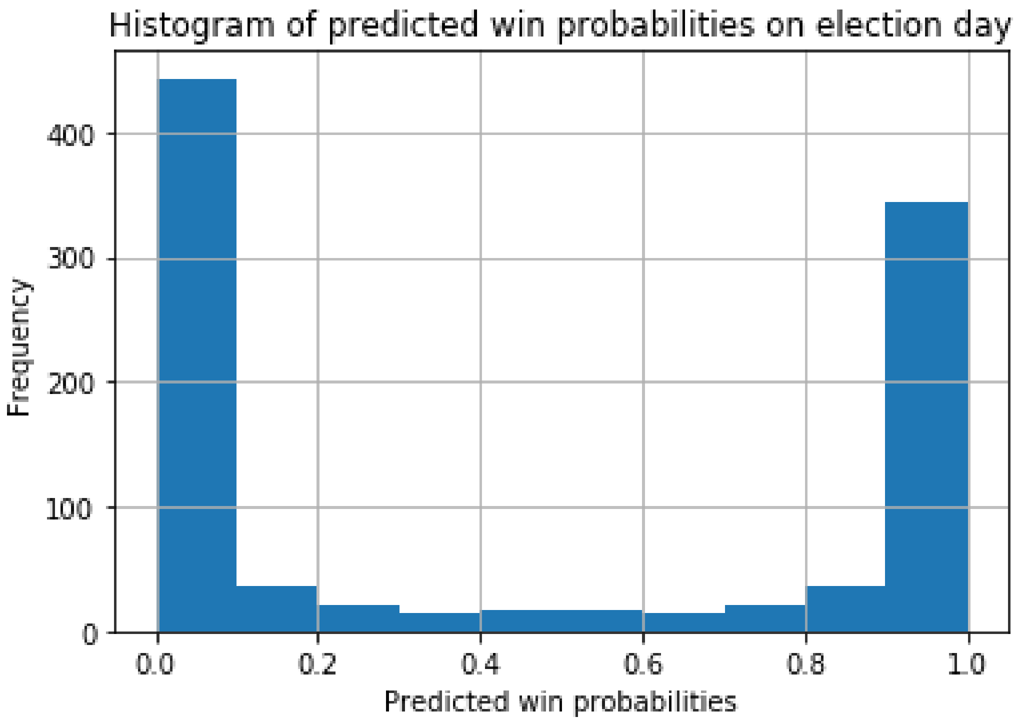


0.0.3 Question 10: prediction histograms

Make a histogram showing the predicted win probabilities on the morning of the election. Again, restrict yourself to only the `classic` predictions.

```
In [77]: electionplot = election_data[(election_data.forecast_type=="classic") & (election_data.forecas
         electionplot['probwin'].hist()
         plt.title('Histogram of predicted win probabilities on election day')
         plt.xlabel('Predicted win probabilities')
         plt.ylabel('Frequency')
```

```
Out[77]: Text(0, 0.5, 'Frequency')
```



Are most house elections easy to forecast or hard to forecast?

SOLUTION HERE House elections are easy to forecast because there is not a lot of randomness associated with predicting them, as it can be said that most incumbents will most likely vote to keep their seat. Given other situations, predicted house elections does not possess as much randomness as would be thought, as there is a better sense of what affects outcomes.

Create a pandas dataframe from the csv and print the first 10 rows.

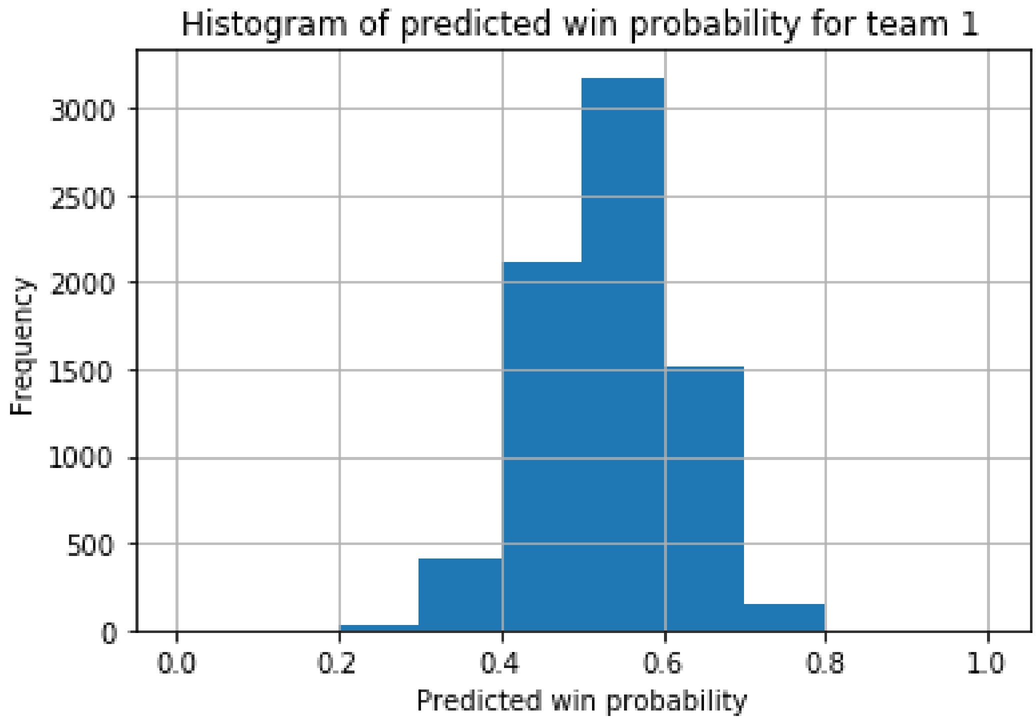
```
In [80]: baseball_data = pd.read_csv("mlb_games.csv")
         baseball_data.head(10)
```

```
Out[80]:    season        date    team1    team2  dh     prob1  prob1_outcome  \
         0    2018  2018-10-28  Dodgers  Red Sox   0  0.483877            0.0
         1    2018  2018-10-27  Dodgers  Red Sox   0  0.508342            0.0
         ... Omitting 16 lines ...
         7  0.396971            1.0
         8  0.431984            1.0
         9  0.399572            0.0
```

In this dataframe `prob1` is the predicted win probability for `team1`. Make a histogram of `prob1`. Set the limits of the x-axis to `[0, 1]`

```
In [79]: baseball_data['prob1'].hist(range = [0,1])
         plt.title('Histogram of predicted win probability for team 1')
         plt.xlabel('Predicted win probability')
         plt.ylabel('Frequency')

Out[79]: Text(0, 0.5, 'Frequency')
```



0.0.4 Question 12

Find the most "surprising" baseball game outcome. To do so, select all of the entries for which `prob1_outcome` is 1 (i.e. `team1` won the game), and then look for the index of the row containing the smallest value of `prob1`. This will correspond to the game that was most suprising according to fivethirtyeights predictions. Find and print the row corresponding to this most surprising outcome.

```
In [82]: base = baseball_data[(baseball_data.prob1_outcome==1)]
         base[(base.prob1 == base['prob1'].min())]
         #Royals had 28.7% chance of winning, and they did win.
```

```
Out[82]:      season        date   team1    team2  dh     prob1  prob1_outcome  \
         521    2018  2018-08-25  Royals  Indians   0  0.287558            1.0

                 prob2  prob2_outcome
         521  0.712442            0.0
```

0.0.5 Question 13

Are the outcomes of baseball games generally easier or harder to predict than the outcomes of political elections? In a few sentences, comment on why this might be the case. What data is available for these predictions? What factors affect the outcomes of elections and baseball games? What makes an event like an election or a baseballgame "random"?

SOLUTION HERE Baseball games outcomes are generally harder to predict than political elections. There is more randomness involved with baseball games, as in, not everything in baseball games are clearly predicted and defined. For example, there are many factors in a baseball game that can affect the outcome, i.e. the players playing in that specific game, stamina, etc. Therefore, in order to better predict baseball games, you would need more data than just probability of that team winning. These are also the events that make an election or a baseball game random, because we cannot thoroughly predict and account for these factors.

Create an analogous plot for empirical error bars with `bootstrap_election_agg`. Also draw a horizontal lines at 0 and 1.

SOLUTION HERE

Compare the two error bar plots and explain.
SOLUTION HERE