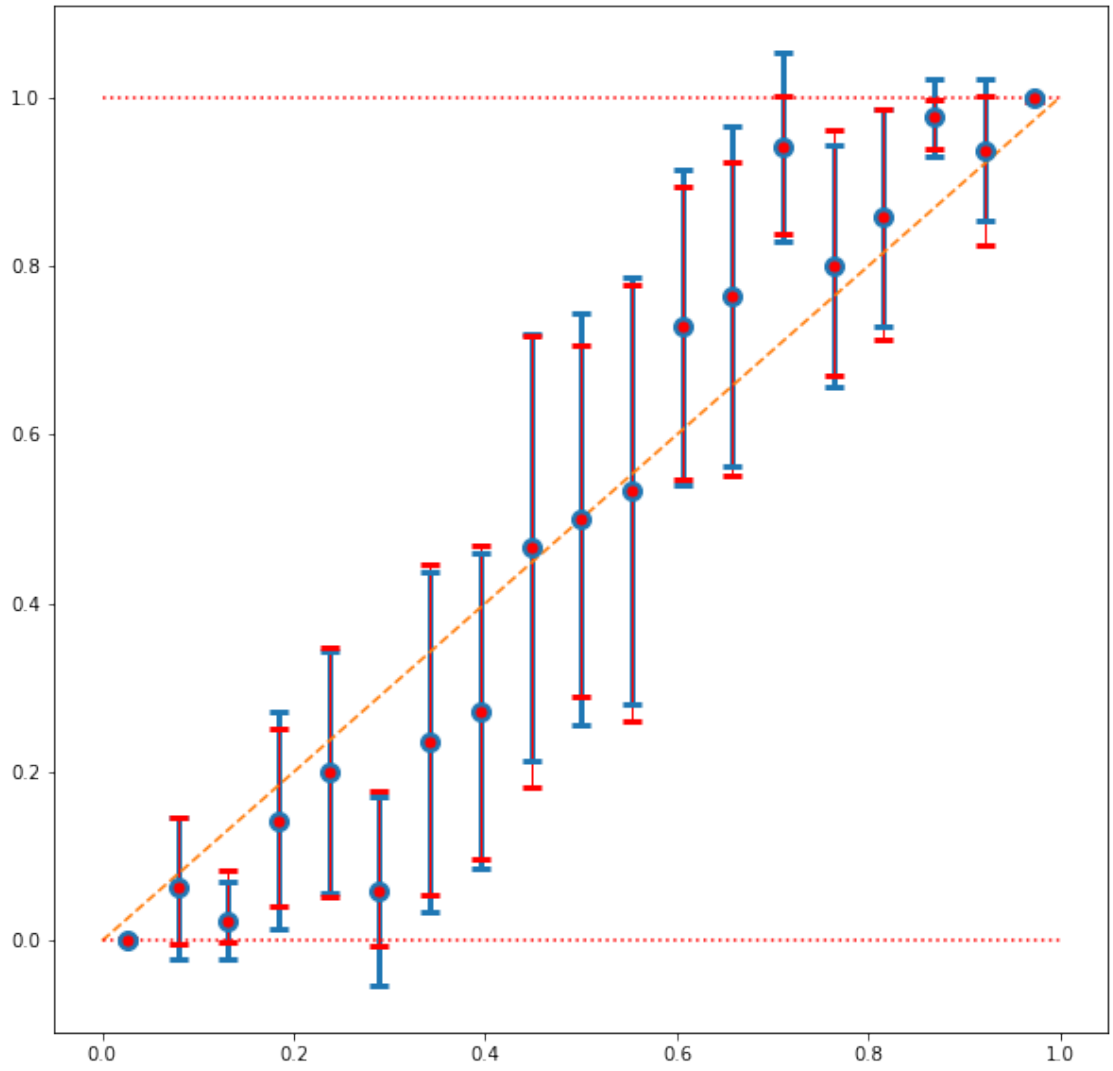
Use pandas.DataFrame.apply and bootstrap_error_bars functions to calculate and visualize the error bars.

In addition, to the figure code used in 5b, add horizontal lines at 0 and 1.

```
In [100]: # calculate error bars
          bootstrap_election_agg = bootstrap_election_100_agg.apply(bootstrap_error_bars,axis=1)

          plt.figure(figsize=(10, 10))
          plt.errorbar(midpoints,
                       election_agg['mean'].values,
                       yerr=election_agg['err'].values,
                       fmt='.', elinewidth=3, ms=20,
                       capsize=5, capthick=3)
          # Overlay empirical error bars on the same plot. Use the following
          # visual attributes:
          # ..., fmt='.r', elinewidth=1, ms=10, ecolor='red', capsize=5, ...
          plt.errorbar(midpoints,
                       election_agg['mean'].values,
                       yerr=bootstrap_election_agg.drop('mean',axis=1).transpose().values,
                       fmt='.r', elinewidth=1, ms=10, ecolor="red",
                       capsize=5, capthick=3)
          plt.plot([0, 1], [0, 1], '--')
          plt.plot([0, 1], [1, 1], ':r')
          plt.plot([0, 1], [0, 0], ':r')
          plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

0.0.1 Question 1c: Interpreting the Results

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 error bars?

Compare and contrast model-based error bars and empirically obtained error bars. What are the advantages and disadvantages of these two approaches?

The 95% confidence intervals are generally smaller for more confident predictions as it can be seen above. The factors that determine the lengh of error bars is how spread out the data is and the number of observations available. Empirical obtained error bars are a better fit for a specific instance of data since you derive it from the data itself. Model-based is better for the overall problem you are trying to solve since it is based on multiple instances of the data and gives a better idea of the model as a whole.

0.1 Part 2: Revisiting Recommender System

0.1.1 Question 2a: Using Linear Algebra for Optimization

In recommender system module, low-rank matrix factorization was used to execute latent factor modeling of movie ratings data.

Specifically, we calculated matrices U and V to solve the following optimization problem (if all ratings were given):

$$\min_{U,V} f(U,V) = \min_{U,V} \|R - VU^T\|_F^2 = \min_{U,V} \left\{ \sum_{m=1}^{M} \sum_{i=1}^{I} (r_{mi} - v_m u_i^T)^2 \right\}.$$

The best U and V were calculated iteratively by improving on current estimates:

$$u_i^{\text{new}} = u_i + 2\alpha(r_{mi} - v_m u_i^T) \cdot v_m$$
$$v_m^{\text{new}} = v_m + 2\alpha(r_{mi} - v_m u_i^T) \cdot u_i,$$

where α is the step-size that is to be chosen by the user. (We won't discuss the role in this class, but treat it as an arbitrary, but given, parameter)

We can make calculating the updates more efficient by calculating them with matrix operations. For example, instead of calculating each deviation $\gamma_{mi} = r_{mi} - v_m u_i^T$ separately for all $m = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, I$, matrix Γ of all deviations can be computed together using matrix operation *(verify for yourself)*:

$$\Gamma = R - VU^T$$

Similarly, updating U and V can be combined into matrix calculations which makes the optimization procedure more efficient.

First, note that updates for u_i, $i = 1, 2, \ldots, I$ can be rewritten as

$$\begin{aligned} u_1^{\text{new}} &= u_1 + 2\alpha\gamma_{m1} \cdot v_m \\ u_2^{\text{new}} &= u_2 + 2\alpha\gamma_{m2} \cdot v_m \\ \vdots \qquad & \qquad \vdots \\ u_I^{\text{new}} &= u_I + 2\alpha\gamma_{mI} \cdot v_m. \end{aligned}$$

Stacking all I equations into a matrix form,

$$U^{\text{new}} = U + 2\alpha\Gamma_{m-}^T v_m,$$

where Γ_{m-} is the m-th row of Γ (use the notation Γ_{-i} for the i-th column).

Note that there are M such update equations (one for each $m = 1, 2, \ldots, M$) that can also be combined into one matrix update equation involving matrices U, V, Γ and scalars. As stated earlier, since α is assumed to be an arbitrary step-size parameter, we can replace α/M with α.

Complete the following update equations:

$$U^{\text{new}} = U + 2\alpha[\text{some function of } \Gamma][\text{some function of } V]$$
$$V^{\text{new}} = V + 2\alpha[\text{some function of } \Gamma][\text{some function of } U]$$

$$U + 2\alpha\Gamma^T V$$
$$V + 2\alpha\Gamma U$$

By referring back to the function used to calculate the quantities in each figure, describe what each figure is showing and interpret the behavior of the optimization algorithm.

The RMSE graph is showing that as we include more iterations the RMSE will decrease indicating a better fit to the data. The max update graph is keeping track of the number of updates that occur for each iteration. The max residual change graph keeps track of how much the residuals vary from each iteration.

0.1.2 Question 2e: Logistic function

Note the reconstructed ratings can be smaller than 1 and greater than 5. To confine ratings to between the allowed range, we can use the logistic function. Logistic function is defined as

$$h(x) = \frac{1}{1 + e^{-x}}.$$

It is straightforward to show the derivative is

$$h'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = h(x)(1 - h(x)).$$

Therefore, we can rescale the ratings from $r_{mi} \in [1, 5]$ to $r_{mi} \in [0, 1]$. Then, we can find the best U and V to optimize the following:

$$\min_{U,V} \|R - h(VU^T)\|_F^2 = \sum_{m,i} (r_{mi} - h(v_m u_i^T))^2,$$

where function h is applied elementwise.

According to the new objective function, rewrite update functions analytically.

$$R - h(VU^T)$$
$$U - \alpha * (-2 * G * h(VU^T) * (1 - h(VU^T))^T * V$$
$$V - \alpha * (-2 * G * h(VU^T) * (1 - h(VU^T)) * U$$

Consider the above plot. By reading the code, comment on what the plot is illustrating. What happens when you add `counts=True` to `transform_density`? What can you conclude?

The plot is illustrating the kernel density estimation over the data supplied with probability estimates plotted. When you add counts=True the plot changes by plotting values by their counts making it more smooth. It looks to me that a lot of people have rated their movies mostly between 3.5 and 5.0.

0.1.3 Question 2g: Make Recommendation

What movies would you recommend to `user id` 601? Do you see any similarities to movies the user rated high?

I would recommend Richard III and Seven. I noticed that the movies the user rated high were mostly movies that invloved some sort of action in them.

```
In [ ]: Rbig.iloc[:,600].head(60)
```

```
In [ ]: Rhatbig.iloc[:,600].head(60)
```