CSE702 Week 6A: Cross-Validation and Things Out of Whack, and Some Experiments

The fitting process on a training set S (specific to a given rating level R) equates

- The projected T1 match on S to the actual;
- The projected ASD on S to the actual; and when the ev parameter is freed in training,
- The projected EV-match on S to the actual.

This fits T1, EV, and ASD as **unbiased estimators**. They are the components of the regular cheating test. They need to be **validated**. This means ensuring that the z-test is **safe**, which in this case means that the test's z-scores conform to the standard bell curve (at least, the positive portion of it). Safety aligns with avoiding **false positives**, aka. **type 1 errors**. This is done in several ways:

- Extensive randomized resampling trials over the training sets and on fresh data. These involve what I call "Frankenstein Players"---randomly aggregating games by different players---which can be objected as having more independence than a single player. Hence also:
- Field tests of the z-scores in numerous/large tournaments. Those are the tests at the bottom of columns Y and Z in spreadsheets I've shown.

Validation also means assessing that the model is **sensitive**, meaning it avoids **false negatives**, aka. **type 2 errors**. Before the pandemic, there wasn't a lot of data on unambiguous true positives needed to quantify this beyond anecdotal instances---and for in-person chess, there still isn't. Some remarks on common parlance:

- The common "rule of three" partitions the base data into a *training set*, a *validation set*, and a *test set*.
- Resampling is often called **cross-validation** because it is separate from the validation process during the original model construction.
- Because of how prediction and assessment are separated and my model being severely underfitted, I have validation separate from model construction. It does have the common meaning of the minimum direct requirement on the assessment tests needed to deploy them.
- Hence I take **cross-validation** to mean further checks of the model's acuity, not necessarily directly related to the main tests. (Can we find a better, less-intrusive word? Maybe just say ``cross-checking''?)

Here are the test entities that I regard as the most important cross-checks for safety:

- The projection accuracy of the *second*, *third*, *fourth*, and *fifth*-listed moves by the engine.
- The projection accuracy of slight errors, small errors, medium errors, and large errors. These are defined as AD (raw difference, not ASD) 1--10, 11--30, 31--70, and 71--150 centipawns, respectively. They are called `Delta...` in the large bottom section of the performance test printout.
- The projection accuracy of errors of a given magnitude and above: at least 50 cp, at least 100, and blunders of at least 200 or at least 400.

- The internal prediction accuracy, using Sir David Spiegelhalter's z-test.
- (The prediction hit rate, in a line somewhat misleadingly called "**`ProjectionHitsW`**" and succeeding rows, works more toward sensitivity.)

The prediction accuracy, illustrated in [this recent GLL post](), and hit rate are gnarly topics, but the first three are readily amenable to experiments. Some points about them:

1. MM_2, the rate of playing the engine's second-listed move, is not expressly fitted. (You can do so by giving a nonzero weight to **`secondLine`** in the loss function configured under menu option [17] **`runFit`**.) To what extent does it behave as an independent variable?
   (a) Because of underfitting, it can be highly biased. In fact, I've believed it to be generally projected too high by my model, as in the final example [here]().
   (b) On first principles, it should be *anti-correlated* with MM_1 (and MM_3, etc.)
   (c) Upon measuring and correcting for systematic bias B_2, what is the nature of the random variable $MM_2 - B_2$?
   (d) Note that using the "Studentised z-scores" of these variables, rather than their native values, puts everything on a common scale.
2. Same issues and questions for MM_3, MM_4, and MM_5. My impression is that the latter two are tangibly less coupled to MM_1.
3. **`ExpectationLossW`** is highly correlated with ASD, but maybe for that reason, behaves almost as if it (that is, its z-score) were expressly fitted and validated---?
4. The **`Delta...`** and **`Error...`** tests can also be tested for systematic biases in sign as well as magnitude.
5. Experiments on these quantities can be conducted and interpreted in two settings:
   (a) When the performance tests are executed from a rating estimate of a player or set of games---the **`perfTest`** workflow.
   (b) When the performance test is of an expressly computed best-fit, in the **`runFit`**/**`runIPR`** workflow.

Here is where we can craft and employ [Pearson correlation]() tests and tests of conformance to the standard normal distribution.

- Are they biased in performance tests by rating? Here I have large data textfiles to hand in subdirectories of **`/projects/regan/Chess/`** ...
- Are they biased after fitting? Here we'd need to generate results from scripted runs of my program---because I save time by not outputting the **`perfTest`** of individual player fits.
- After subtracting out any bias, how far is the resulting variable from normal?
- How strong are correlations between variables? (Note that by the linear invariance of Pearson correlation, one does not need to correct bias to work on this.)

Is there a good notation for a vector x minus its mean? Try $C(x)$ for that. Then the Pearson formula for the correlation between sample vectors x and y, using $\bullet$ for the dot product, is

$$\frac{C(x) \bullet C(y)}{||C(x)|| \cdot ||C(y)||}$$

Results Files and Their Formats

Here are example outputs from my run of the 2024 Cambridge International Open using Komodo 13.3. I've made a new folder **`/projects/regan/Chess/CSE702/ResultsFiles/`** and have placed a bunch of results files there, in this case **`CambridgeIntlOpenFeb2024t960Kom13UW.txt`** Skipping over the performance tests of all the players at their given ratings (not at their fitted IPR settings), here is the IPR run over all games in the tournament---followed by a performance test *of that fit*.

...

```
IPR: 2190.99 from 0.09695, 2-sigma range [2164.55,2217.44]
IPR if 28176 positions faced were test suite: 2206.68, st. dev. 13.22
AdjIPR: 2190.99 via 0.0955041/0.0955041 = 1: 2164.55--2217.44
Adj. AE/turn: 0.0969532 stdev. 0.00162856, index 9.70206e-06

Line for paper:
CambridgeIntlOpenFeb2024Kom13IPR & & 2190.99 +- 26.44 & 2.2e+03--2.2e+03 & 28176\\  % IPRauto: 2206.68 +-
13.22  / 2190.99
CambridgeIntlOpenFeb2024Kom13IPR(simple): 2190.00 +- 25.00

Final IPR:
IPR-CambridgeIntlOpenFeb2024Kom13IPR & & 2190.99 & 2164.55--2217.44 & 28176, wt = 28176.0000\\  % IPRauto:
2206.68 +- 13.22  / 0.00
IPR of CambridgeIntlOpenFeb2024Kom13IPR(simple): 2190 +- 25
Challenge faced by CambridgeIntlOpenFeb2024Kom13IPR: 0.0961 at ref 2181.00 is 2170.12 with complexity
0.0149; actual ASD 0.0957 and IPR 2191.49
```

Note that the IPR is computed as 2190.99 but rounded to the nearest 05 as 2190, and likewise the error bars. There is a lot of wonky other stuff: "**`IPRauto`**" is the figure that would result if the whole set of 28,176 positions were used as the reference set. Here it is only 17 Elo points different---the games played in Cambridge and the 150 games in the reference set are fairly similar overall. There is an attempt to measure "challenge faced" but in unit-weights mode (**`UW`**) it has little point.

When perfTest goTest is immediately invoked next, the fit that was obtained is shown---along with all the model settings---in the preamble:

```
----------
Test of aggregate /shared/projects/regan/Chess/CC/AA201X/CambridgeIntlOpen*Feb2024*Kom13*aif giving files
CambridgeIntlOpenFeb2024_Kom133d20-30pv64.aif
using PowerShares trial BasicPowerShares: 34418 turns, 28176 filtered by 5 filters
Spec CambridgeIntlOpenFeb2024Kom13IPR: (InvExp:1), Unit weights, error model logErrorC of 1.00*Brier + 0.00*Likely;
by index 1 as f(i) steps from 0.00 to 1.00 at 2; tailMax 0.010 for Kom13 at rating basis 2191.0 with
p = 0.00000, q = 0.00000, r = 0.00000, s = 0.03902, t = 0.00000, u = 0.00000;
```

```
e = 0.00000, f = 0.00000, g = 0.00000, h = 0.00000;
c = 0.35969, a = 1.00000, b = 1.00000;
tz= 0.00000, fz= 0.00000, bz= 0.00000, sf= 0.00000, ne= 0.00000, ev= 2.04131, co= 0.03902;
tc= 0.00000, tp= 0.00000, sp= 1.00000, d = 20.00000, v = 0.03500;
la= 0.05141, lb= 1.11693, lk= 0.05141, lq= 1.00000;
am= 0.09944, ap= 0.08719, bm= 0.20070, bp= 0.09611, cm= 0.82327, cp= 0.62277;
uz=-0.01305, vz= 0.04481, wz= 0.00000, dc= 20.00000, ec= 20.00000, pp= 0.61600, oi= 0.00000, ft= 0.04731;
LogScalerNoPatchLinearWts(6..20)0[6..20,6..20]WEF.SI.UBE.:
(*omodo*,1) (*tockfish*,1) (noSwing:1),simple;carrySwing;mulDiffs;invParams;anchorZero
Filters:
pnew4norm: OrFilter [Prev turn |eval| <= 4, Turn |eval| <= 4, Next turn |eval| <= 4]
numLegalGeq2: # legal moves >= 2
RC0: RepCount == 0
from9: TurnNumber >= 9
to60: TurnNumber <= 60
```

Now the results follow. Here is the move-matching component:

```
From 28176 turns with total weight 28176 and avg. Elo 2047.94 versus 2047.04, move indices first:
Weighted Elo averages: PTM 2047.94, Oppts. 2047.04, White 2048, Black 2046.99
```

i	mDelta	SwNotDD	SwingDD	SwRel	ProjVal	Sigma	Actual	Proj%	Actual%	2sigma range	z-score	BrierSc	LikelySc
1	0.00	0.0000	0.0112	0.0000	13387.01	75.13:	13387.00	47.51%:	47.51%	46.98%--48.05%,	z =-0.00	-5.625	-5.775
2	0.22	-0.0354	0.0059	-0.0053	4985.71	61.11:	5073.00	17.69%:	18.00%	17.26%--18.13%,	z =+1.43	3.579	4.346
3	0.36	-0.0539	0.0035	-0.0077	2576.04	46.87:	2603.00	9.14%:	9.24%	8.81%-- 9.48%,	z =+0.58	1.270	3.981
4	0.45	-0.0655	0.0018	-0.0093	1636.95	38.32:	1647.00	5.81%:	5.85%	5.54%-- 6.08%,	z =+0.26	0.726	5.679
5	0.52	-0.0729	0.0011	-0.0101	1142.28	32.46:	1110.00	4.05%:	3.94%	3.82%-- 4.28%,	z =-0.99	-0.685	2.695
6	0.58	-0.0838	-0.0000	-0.0111	857.06	28.34:	828.00	3.04%:	2.94%	2.84%-- 3.24%,	z =-1.03	-0.727	2.094
7	0.64	-0.0903	-0.0012	-0.0122	669.63	25.17:	675.00	2.38%:	2.40%	2.20%-- 2.56%,	z =+0.21	0.341	2.154
8	0.69	-0.1024	-0.0027	-0.0138	533.66	22.58:	492.00	1.89%:	1.75%	1.73%-- 2.05%,	z =-1.85	-1.524	1.228
9	0.74	-0.1124	-0.0041	-0.0151	434.99	20.44:	426.00	1.54%:	1.51%	1.40%-- 1.69%,	z =-0.44	-0.193	2.318
10	0.78	-0.1214	-0.0047	-0.0157	355.86	18.53:	323.00	1.26%:	1.15%	1.13%-- 1.39%,	z =-1.77	-1.585	0.861
11	0.83	-0.1308	-0.0059	-0.0169	298.15	16.98:	264.00	1.06%:	0.94%	0.94%-- 1.18%,	z =-2.01	-1.749	0.554
12	0.87	-0.1386	-0.0064	-0.0174	243.61	15.39:	191.00	0.86%:	0.68%	0.76%-- 0.97%,	z =-3.42	-3.361	-2.123
13	0.92	-0.1520	-0.0078	-0.0187	201.23	14.01:	202.00	0.71%:	0.72%	0.61%-- 0.81%,	z =+0.05	0.239	1.983
14	0.96	-0.1553	-0.0084	-0.0193	168.35	12.81:	186.00	0.60%:	0.66%	0.51%-- 0.69%,	z =+1.38	1.651	4.345
15	1.01	-0.1577	-0.0086	-0.0194	135.74	11.53:	130.00	0.48%:	0.46%	0.40%-- 0.56%,	z =-0.50	-0.307	1.515

```
Index fits, x10,000: 0.00307, wtd. 0.00307, diff -1.723e-08; mass 0.05795, wtd. 0.05795; diff -1.723e-08
LogSumPlayedMoves: 2.493; LogSumPlayedMovesBinary: 0.8239; PlogpSumPlayedMoves: 0.3518; Entropy sum: 2.303
```

Here the frequency of playing the engine's second-listed move is projected slightly too low. Likewise the third-listed move; even though the difference between 9.14% and 9.24% looks really minor, it's still a "standard score" of +0.58 from over 2,500 hits among 28,000+ data points. The next few ordinal indices are also creditably close---and maybe more important, their signs are mixed. The last two lines start with an overall index-fit score: 0.003 is excellent; anything under 0.01 is good and under 0.02 is decent. The last line gives the loss-function values for maximum-likelihood estimation (MLE) of the played moves and some variants. Assessing why MLE works poorly, and maybe fixing it, is another seminar project idea.

Here are the main z-tests and the predictivity z-tests:

```
Name                 ProjVal   St.Dev     Actual;   Proj% Actual%  2sigma range   z-score            BrierSc LikelySc
AvgScaledDiffW      2690.924  34.042:   2690.924  0.0955: 0.0955 0.0931--0.0979, z = +0.00, adj +0.00    0.038  0.000
ExpectationLossW     863.311  10.211:    876.467  0.0306: 0.0311 0.0299--0.0314, z = -1.29, adj -1.14    0.004  0.000
MoveMatchWtd       13387.014  75.133:  13387.000  47.51%: 47.51% 46.98%--48.05%, z = -0.00, adj -0.00    0.195  0.000
EqValueMatchW[4]   14511.767  74.681:  14465.000  51.50%: 51.34% 50.97%--52.03%, z = -0.63, adj -0.51    0.194  0.000
                                                                                 Combined: adj -0.19

Prediction Tests:
LikelihoodWtd      53021.55 328.17:   55594.60  0.0000: 1.9731 0.0000--0.0000, z = +7.84    0.000  0.000
BrierDefectiveWtd  11281.06  60.52:   11283.83  0.0000: 0.4005 0.0000--0.0000, z = +0.05    0.000  0.000
CombinedScoreWtd   11281.06  60.52:   11283.83  0.0000: 0.4005 0.0000--0.0000, z = +0.05    0.000  0.000
LikelihoodMultiWtd 72704.10 325.58:   75408.51  0.0000: 2.6763 0.0000--0.0000, z = +8.31    0.000  0.000
BrierMultiWtd      12829.85  60.00:   12856.58  0.0000: 0.4563 0.0000--0.0000, z = +0.45    0.000  0.000
CombMultiWtd       12829.85  60.00:   12856.58  0.0000: 0.4563 0.0000--0.0000, z = +0.45, adj +0.35    0.000  0.000
```

Only `AvgScaledDiffW` and `MoveMatchWtd` are expressly fitted. The expectation loss and EV match are annoyingly off, and (only) the latter contributes to the overall combined z-score being -0.19.

Here are the sections with projection hits and the main *uncalibrated* tests:

```
PlayedMoveMatchW    9819.18  64.65:   28176.00  34.85%:100.00% 34.39%--35.31%, z = +283.94    0.503  0.000
ProjectionHitsW    13927.79  76.74:   14030.00  49.43%: 49.79% 48.89%--49.98%, z = +1.33, adj +1.09    0.206  0.000
Proj1 (23050.00)   12347.78  69.73:   12374.00  53.57%: 53.68% 52.96%--54.17%, z = +0.38, adj +0.33    0.204  0.000
Proj2 (3318.00)     1141.49  26.60:    1145.00  34.40%: 34.51% 32.80%--36.01%, z = +0.13, adj +0.11    0.216  0.000
Proj3 (1004.00)      273.08  13.86:     288.00  27.20%: 28.69% 24.44%--29.96%, z = +1.08, adj +0.94    0.204  0.000
Proj4 (385.00)        90.62   8.19:     102.00  23.54%: 26.49% 19.28%--27.79%, z = +1.39, adj +1.21    0.191  0.000
Proj5 (152.00)        32.01   4.95:      31.00  21.06%: 20.39% 14.55%--27.58%, z = -0.20, adj -0.18    0.164  0.000
Proj6 (90.00)         16.96   3.66:      17.00  18.84%: 18.89% 10.72%--26.96%, z = +0.01, adj +0.01    0.152  0.000
Proj7+(177.00)        25.85   4.59:      73.00  14.60%: 41.24% 9.41%--19.79%, z = +10.26, adj +8.93    0.313  0.000
```

```
Name                ProjVal  St.Dev    Actual;  Proj%  Actual% 2sigma range    z-score
Top2Wtd            18372.72  71.18:   18442.00  65.21%: 65.45% 64.70%--65.71%, z = +0.97, adj +0.80
Top3Wtd            20948.76  65.80:   21051.00  74.35%: 74.71% 73.88%--74.82%, z = +1.55, adj +1.27
Top3thr0.50Wtd     19847.88  70.52:   20030.00  70.44%: 71.09% 69.94%--70.94%, z = +2.58, adj +2.12
Match-T2Wtd         8401.31 117.02:    8317.00  29.82%: 29.52% 28.99%--30.65%, z = -0.72, adj -0.59
Match-T3Wtd         5825.27 129.98:    5716.00  20.67%: 20.29% 19.75%--21.60%, z = -0.84, adj -0.69
```

The z-scores of the T2 and T3 tests are almost always positive, which means those tests are biased toward false positives in this fit.

Here are the error tests:

Selection Test	ProjVal	St.Dev	Actual;	Proj%	Actual%	2sigma range	z-score		BrierSc	LikelySc
Delta01-10	1745.99	31.65:	1728.00	32.76%:	32.42%	31.57%--33.95%,	z = +0.57,	engm% = 0.00	1.547	1.810
Delta11-30	2223.05	37.44:	2222.00	27.84%:	27.82%	26.90%--28.77%,	z = +0.03,	engm% = 0.00	2.170	2.671
Delta31-70	1655.54	34.83:	1674.00	16.45%:	16.63%	15.76%--17.14%,	z = -0.53,	engm% = 0.00	3.371	6.083
Delta71-150	754.40	24.48:	760.00	6.89%:	6.94%	6.44%-- 7.34%,	z = -0.23,	engm% = 0.00	2.410	6.674
Error025	3326.69	46.20:	3395.00	23.54%:	24.03%	22.89%--24.20%,	z = -1.48,	engm% = 0.00	4.114	8.710
Error050	1768.41	35.79:	1789.00	12.54%:	12.69%	12.04%--13.05%,	z = -0.58,	engm% = 0.00	3.335	9.248
Error100	743.54	24.27:	762.00	5.30%:	5.43%	4.95%-- 5.64%,	z = -0.76,	engm% = 0.00	3.727	11.869
Error200	274.19	15.22:	278.00	1.97%:	1.99%	1.75%-- 2.18%,	z = -0.25,	engm% = 0.00	2.532	14.319
Error400	106.61	9.68:	79.00	0.79%:	0.58%	0.64%-- 0.93%,	z = +2.85,	engm% = 0.00	-1.788	5.008
EvalGoesToZero	3338.01	32.96:	3236.00	26.62%:	25.81%	26.10%--27.15%,	z = -3.09,	engm% = 25.75	6.985	12.659

Here are miscellaneous other selection tests:

PawnMove	6445.34	55.43:	6487.00	23.92%:	24.07%	23.50%--24.33%,	z = +0.75,	engm% = 25.63	1.633	4.565
KnightMove	4108.08	43.15:	4740.00	21.39%:	24.68%	20.94%--21.84%,	z = +14.64,	engm% = 23.76	14.548	16.885
BishopMove	4230.98	43.98:	4421.00	20.67%:	21.60%	20.24%--21.10%,	z = +4.32,	engm% = 20.70	3.614	5.403
RookMove	6095.77	51.15:	5595.00	24.69%:	22.66%	24.28%--25.11%,	z = -9.79,	engm% = 22.97	-6.084	-4.582
QueenMove	4112.25	40.86:	3905.00	22.86%:	21.71%	22.41%--23.32%,	z = -5.07,	engm% = 21.48	-1.417	0.964
KingMove	2745.59	38.52:	2590.00	10.10%:	9.52%	9.81%--10.38%,	z = -4.04,	engm% = 9.17	-3.914	-2.790
Castling	302.11	13.80:	435.00	14.97%:	21.56%	13.60%--16.34%,	z = +9.63,	engm% = 20.27	9.561	9.227
Capture	5323.93	38.21:	6673.00	22.03%:	27.61%	21.71%--22.34%,	z = +35.31,	engm% = 26.54	14.638	23.120
NonCapture	18847.07	38.21:	17498.00	77.97%:	72.39%	77.66%--78.29%,	z = -35.31,	engm% = 73.46	14.638	23.177
Promotion	18.74	2.40:	12.00	23.42%:	15.00%	17.42%--29.42%,	z = -2.81,	engm% = 15.00	-1.794	-1.670
AdvancingMove	16844.82	63.70:	18241.00	60.28%:	65.27%	59.82%--60.73%,	z = +21.92,	engm% = 64.86	-3.438	-2.302
RetreatingMove	5351.92	50.60:	4542.00	19.51%:	16.55%	19.14%--19.88%,	z = -16.01,	engm% = 16.80	-7.556	-5.440
SidewaysMove	5865.26	54.68:	5279.00	21.44%:	19.30%	21.04%--21.84%,	z = -10.72,	engm% = 19.48	-6.063	-4.640
CheckingMove	1094.41	21.93:	1305.00	8.84%:	10.54%	8.49%-- 9.20%,	z = +9.61,	engm% = 10.42	8.433	12.806
EngineMove	13387.01	75.13:	13387.00	47.51%:	47.51%	46.98%--48.05%,	z = -0.00,	engm% = 100.00	5.625	-5.775
PlayedMove	9819.18	64.65:	28176.00	34.85%:	100.00%	34.39%--35.31%,	z = +283.94,	engm% = 47.51	321.835	440.82
SamePieceAsPrevMov	666.41	15.57:	819.00	12.03%:	14.78%	11.47%--12.59%,	z = +9.80,	engm% = 14.17	2.312	4.655
EqualTopMove	14698.62	73.88:	14573.00	52.31%:	51.86%	51.78%--52.84%,	z = -1.70,	engm% = 100.00	-4.115	-0.851

What can go wrong?

- Knight moves
- Capturing moves
- Advancing moves
- Castling---maybe to a lesser extent.

There z-scores are **invariably astronomically positive**. Are these genuine human psychological tendencies, or is something amiss with their projections in the model? Let's look further...