10/24: new data, baseline

daily plan

alright, so. to start out with, i got new data from peterx. it's a zipped giant csv that contains a bunch of de_dust2 matches. the first thing we'll do is start using that data; then i have to make a few changes based on feedback from peterx, and finally i can set up a baseline model/predictor. i also have to put thought into what metrics to use for this.

- [x] unzip and start using new, expanded dataset
- [x] train/test split per-match, not per-round (since correlated rounds: equipment, money carries over, momentum)
- [x] remove rounds that have fewer than 10 players (parser artifacts)
- [x] baseline model:
    - map-based? very simple heuristic
    - count of players alive?
    - basic logistic regression?
    - something from peterx's paper? he mentioned he wanted to see what effect the distance itself had

new data

looks like the playerframes csv (2gb) doesn't have headers. asking peterx...

also looks like the csgo_frames_dust2 csv is what i was working with? but it doesn't include steamid. the playerframes does though. i wonder if it's the same headers.

actually it looks like the new format for frames/playerframes is completely different. guess we're rewriting the transform function again... hopefully it mostly works as long as the correct headers are passed in, since i wrote using column names instead of indices.

i guess while waiting on this, i'll get started with some code for the baseline stuff or data cleanup.

here are the headers for the playerframes data:

```
['MatchId', 'MapName', 'RoundNum', 'Tick', 'Second', 'PlayerId',
'PlayerSteamId', 'TeamId', 'Side', 'X', 'Y', 'Z', 'ViewX',
'ViewY', 'AreaId', 'Hp', 'Armor', 'IsAlive', 'IsFlashed',
'IsAirborne', 'IsDucking', 'IsScoped', 'IsWalking', 'EqValue',
'HasHelmet', 'HasDefuse', 'DistToBombsiteA', 'DistToBombsiteB',
'Created', 'Updated']
```

also looks like the frames csv is not what i want at all, i do need the playerframes. let me make sure everything i use is in this csv

`SteamId` -> `PlayerSteamId`, `IsAlive` instead of `Hp == 0`?

looks like pretty simple changes. IsAlive instead of checking for hp=0 is a nice addition.

actually i also want to make sure i check the rounds csv to ensure it still has the cols i want.

`RoundWinnerSide` -> `WinningSide`, changed.

looks like this is running fine. time to write baseline code while it runs i guess.

data cleanup

first step i guess is re-breaking the code so that it doesn't work with fewer than 10 players, as a litmus test for whether we're filtering stuff right. this means going back and removing the dynamic view stuff

ok now for actually removing the rounds. has to be done when writing the individual rounds csv's.

this continue should do the trick (commit `c559bae`). have to test later. for now, switch back to per-game splitting instead of per-round

ok looks like per-game splitting is working again. i also am still splitting by round in order to make the whole process (hopefully) not have as many changes. i am not sure how this affects IO (probably better when shuffled right? but worse when not?).

time to go back to the new dataset and see what changes need to be made to adapt to that.

ok testing this it looks like it's working fine. i'm not gonna bother running the entire thing because i need to re-run for the dust2 dataset anyway. so i'm gonna go ahead and do that instead

oh well that was cool. a solid hour and a half after starting this code i realized i was copying the wrong thing to file... so it was taking forever, and was giving me the wrong results. -_- starting over...

didn't find any rounds with less than 10 players. so hopefully it actually works?

looks like still having issues with the transform and different input sizes... :/

ah. looks like some games have 10 players but specific rounds have 8 players. so this is probably what's throwing stuff off. gonna move the check to per-round instead of per-game

STILL running into issues. match id 1071, de_dust2, round 1 has 4 T's and 5 CT's each tick - but overall 10 PlayerSteamId values??? asking peterx

```
In [12]: import pandas as pd
         df = pd.read_csv('G:/datasets/csgo/match-map-unique/train/match-1071-de_dust2-1-278.csv')
```

```
In [16]: df.groupby(['MatchId']).agg({'PlayerSteamId': 'nunique'})
```

Out[16]:

	PlayerSteamId
MatchId	
1071	10

```
In [22]: df.groupby(['Tick']).agg({'PlayerSteamId': 'nunique'})
```

Out[22]:

	PlayerSteamId
Tick	
28	9
61	9
94	9
127	9
160	9
...	...
9039	9
9072	9
9105	9
9138	9
9171	9

```
In [20]: grouped = df.groupby(['Tick', 'Side'], as_index=False).agg({'PlayerSteamId': 'nunique'})
         grouped[grouped['Side'] == 'T']['PlayerSteamId'].unique()

Out[20]: array([4], dtype=int64)
```

```
In [21]: df[df['Side'] == 'T']['PlayerSteamId'].unique()

Out[21]: array([76561198047402862, 76561197997981170, 76561197994395491,
                76561197978321481, 76561198012987839], dtype=int64)
```

it does in fact refer to 5 people... but i don't understand why there's always one player missing from each tick. it looks like two of the players "switched out" about halfway through the match? could this be a d/c issue? how would i even VERIFY this?

```
In [26]: for x in df[df['Side'] == 'T']['PlayerSteamId'].unique():
             print(f'https://steamcommunity.com/profiles/{x}/')
             print(f"Showed up in {df[df['PlayerSteamId'] == x]['Tick'].nunique()} ticks")

         https://steamcommunity.com/profiles/76561198047402862/
         Showed up in 140 ticks
         https://steamcommunity.com/profiles/76561197997981170/
         Showed up in 278 ticks
         https://steamcommunity.com/profiles/76561197994395491/
         Showed up in 278 ticks
         https://steamcommunity.com/profiles/76561197978321481/
         Showed up in 278 ticks
         https://steamcommunity.com/profiles/76561198012987839/
         Showed up in 138 ticks
```

apparently it might be a dc issue. so i'll filter out based on the total row count being less than player count times tick count...

looks like this is finally working. however, it's also *very* slow. taking forever to load 240 samples and i have about 3500. probably going to have to rewrite this so that the dataset is in RAM instead of on disk.

Dataset rewrite

so we need to keep it in RAM. let's try and make this efficient i guess.

rewrote a huge chunk of code and am now transforming the dataset in memory to make sure it works. gonna output the transformed stuff too since this is taking forever.

looks like the entire dataset needs to be *kept* in memory. writing to disk takes up 30gb and an enormous amount of RAM (12gb+) plus swap space. i am just going to have to do the transforms every time.

baseline code

let's try implementing a couple of baselines. first the map based one, then the player count based one, and then a logreg. i need to look into peterx's idea of using his model as a baseline too, but i don't know if i want to do that since it certainly sounds more complex (i would have to incorporate all of the data that he used, such as damage events, in order to accurately assess what the distance metric's influence on win probability is)

map based baseline

since i only have one map now (de_dust2), just count how many rounds each side won from the rounds csv. sounds easy.

writing that code was indeed pretty simple, let's see if it gives me the expected output once the train/test splitting is done.

looks like de_dust2 is: 47.44% CT win, 52.56% T win. so predicting T every time would yield an accuracy of 52.56%.

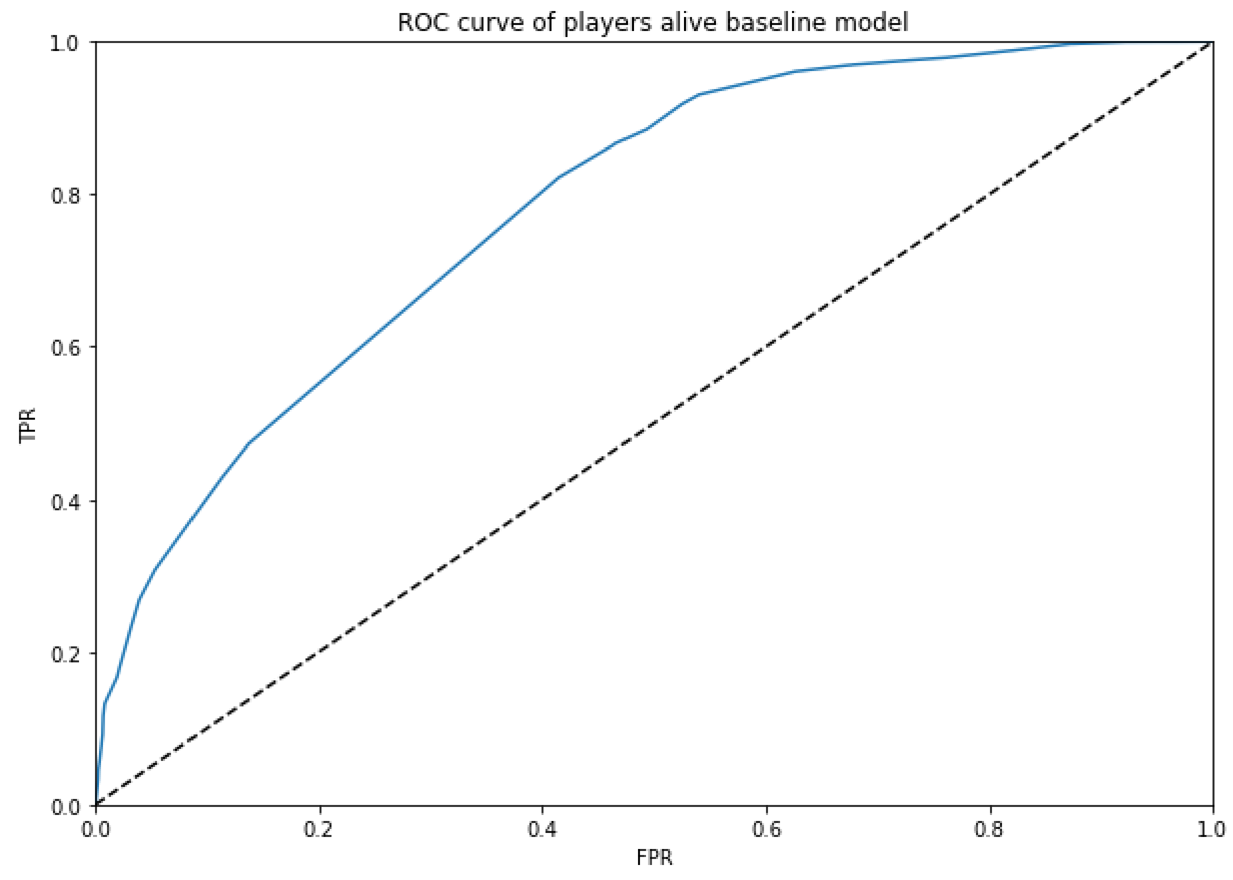
map baseline accuracy: 0.5256
map baseline auc score: 0.5

--- break for me to rewrite dataset code ---

player count baseline

do i need a test set for this? i don't think i do, since it's a nonparametric approach. how would i overfit?

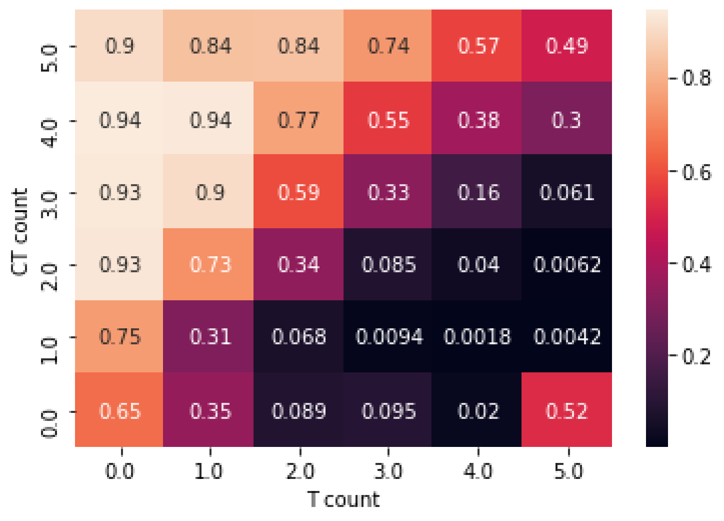
ok actually i do need a test set so that i can measure accuracy/brier score/etc. but i don't need a val set. i'm doing a different split than the other stuff because i don't think it matters much. still splitting along games though



player count accuracy: 0.6922
player count auc score: 0.7838

dude. this is an insane baseline. i have no idea how i can beat this. i hope i can.

there's also something weird going on because with 0 ct's and 5 t's there's still a *HUGE* chance of CT winning?



CT count \ T count	0.0	1.0	2.0	3.0	4.0	5.0
5.0	0.9	0.84	0.84	0.74	0.57	0.49
4.0	0.94	0.94	0.77	0.55	0.38	0.3
3.0	0.93	0.9	0.59	0.33	0.16	0.061
2.0	0.93	0.73	0.34	0.085	0.04	0.0062
1.0	0.75	0.31	0.068	0.0094	0.0018	0.0042
0.0	0.65	0.35	0.089	0.095	0.02	0.52

looks like there are a few instances of all CT's being at 0 hp even though in reality they probably weren't. this might be a separate round? or it might be a dc issue? but that's what's causing the weird numbers in the heatmap above. i asked peterx about it but i guess he'll just say "how often does this happen" and tell me to throw that data out, lol. i'm not sure how i can filter for something like that without it being super annoying though

alright looks like he says it might be the case if a round is restarted manually for some reason. i guess just ignore and carry on

logistic regression baseline

implementing this from sklearn directly. i'm going to run a small grid search to make sure it's optimized at least somewhat decently, but for now let me just see if it runs in the first place. the dataset is rather large and i don't know how much memory this is going to take up.

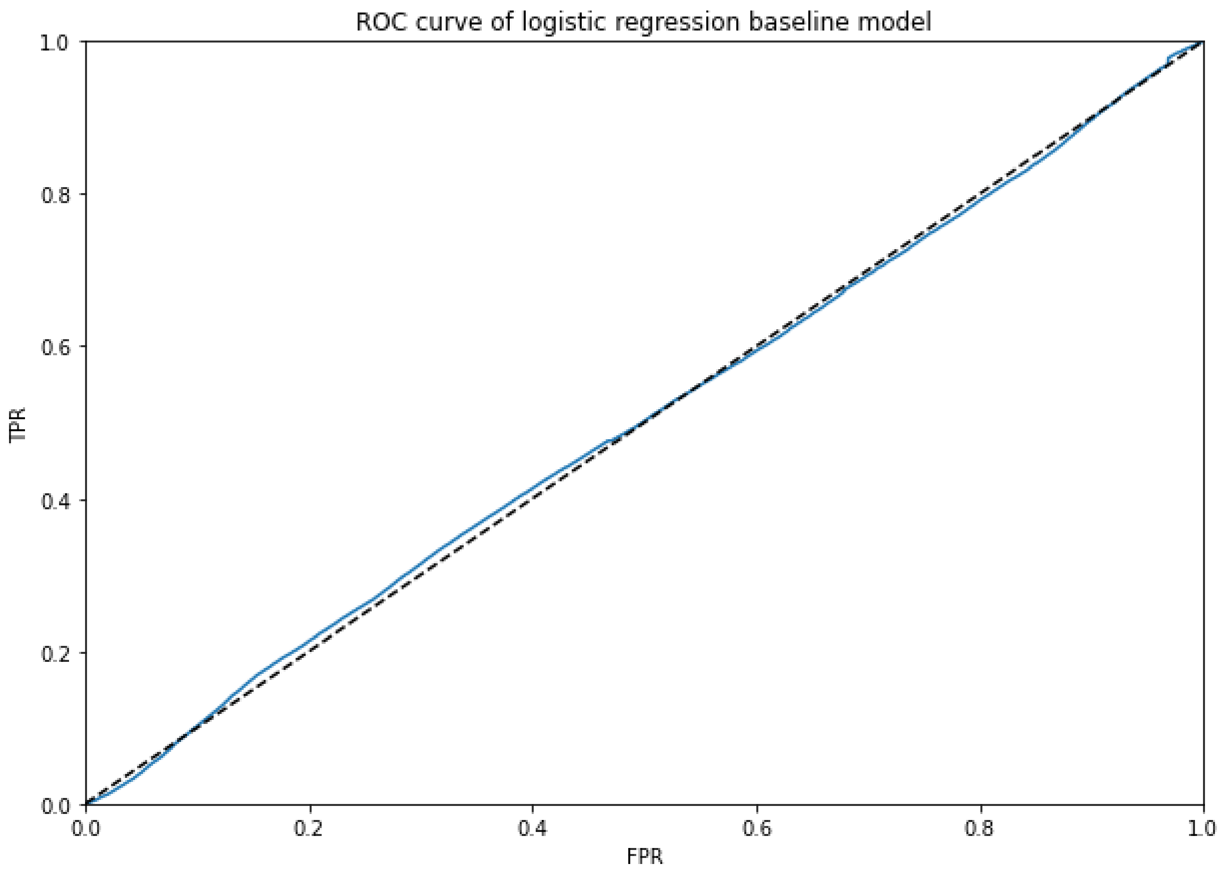
also measuring via auc score.

preliminary results (with only 100 rounds of data) indicate that LR is trash. 0.49 AUCish. gonna run this with the entire dataset and hopefully it will 1. fit in memory 2. run 3. not be complete trash. i also am not running a grid search for best hyperparams yet, so maybe i'm just being silly. but i want to see if this runs in the first place.

full dataset yields... 0.51 AUC. time to CV.

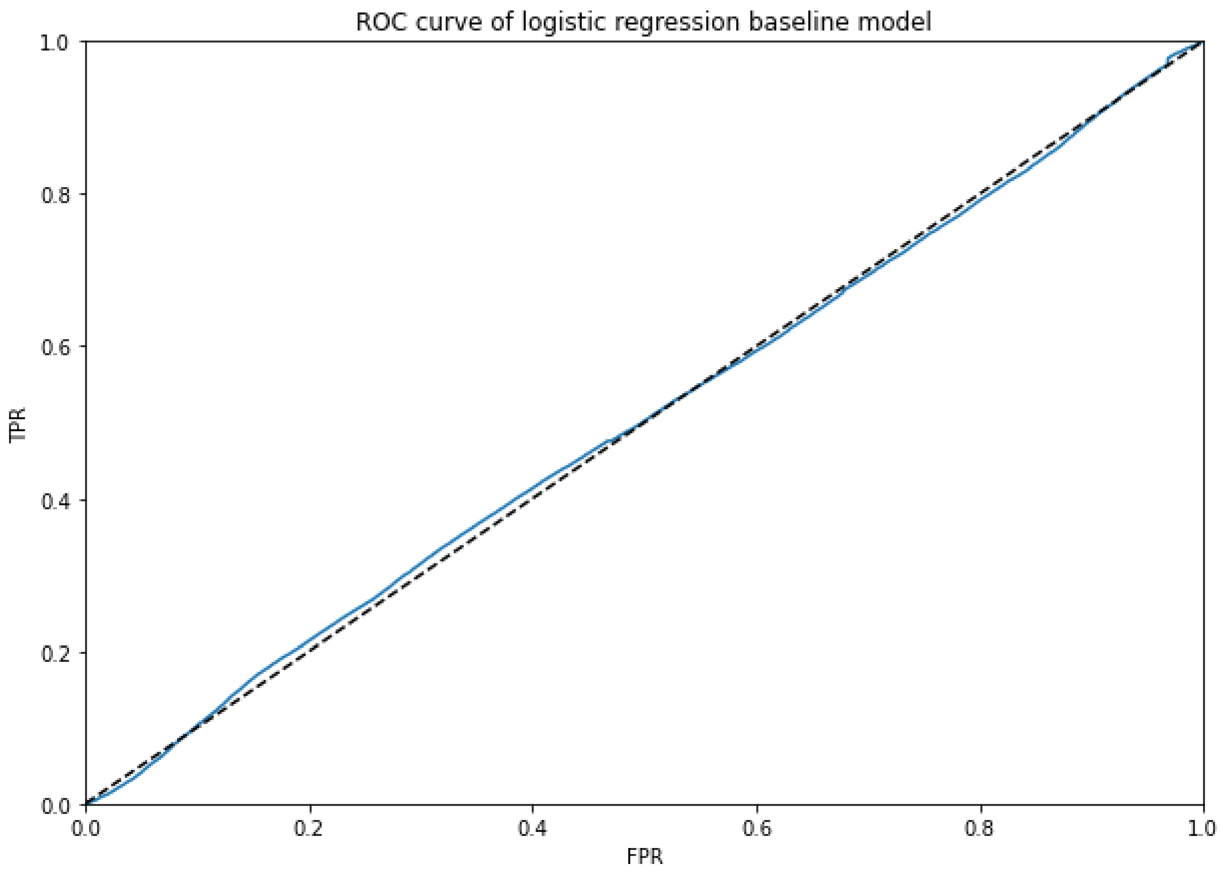
CV results are also really, really crappy. trying with scaling + removal of constant features but it looks like it's also 0.51 AUC. not sure what would lead it to have such a terrible performance when theoretically it has access to the same features as the player count baseline.

```
0.5017980159110792
```



with scaling:

```
0.5018716279161413
```



i'm not sure the graph even changed.

LR baseline AUC: 0.5019
LR baseline acc: 0.5512

---

summary

things done

- finished data cleanup
- started using new data that peterx gave me
- implemented 3 baselines (one of which is pretty terrible)
- train-test splitting done per-match, not per-round anymore

questions for peterx

- none really. maybe - why is LR so terrible? but not sure he can answer that

next steps/remaining action items

- research into what neural net archs might work best for this problem

- implementing three (hopefully) neural net architectures that can ingest the data in the current (n_samples, 10, 12) shape and output a win probability (i.e. last layer is single sigmoid)