Fantasy Football AI

BY: Mohammed Khan, Robert Parsons, Ramon Moreno

Why Fantasy Football?

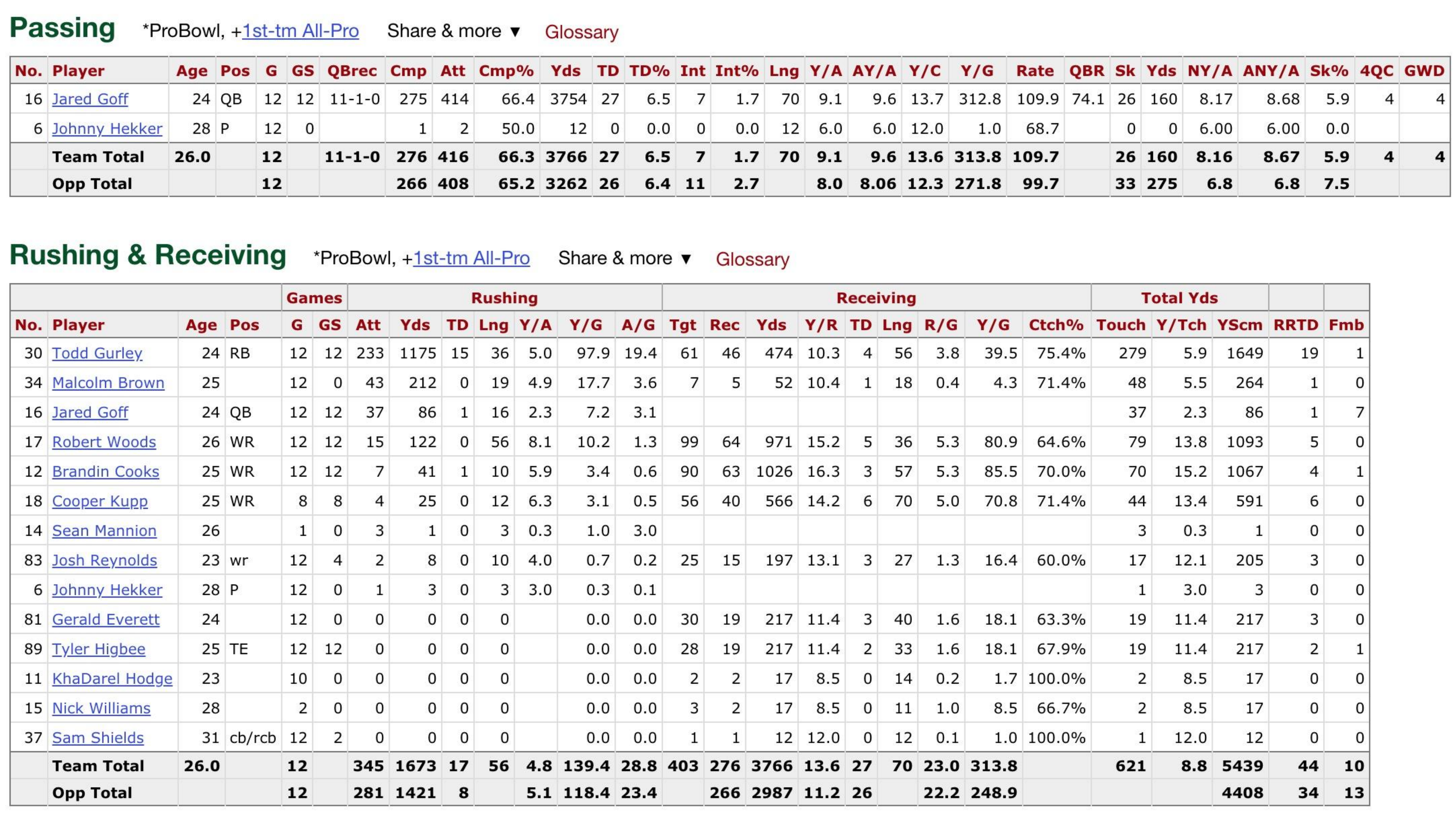
- Many people partake in Fantasy Football, thus making it interesting to see if we can make an AI that predicts the outcome of points
- Fantasy Football entails that statistics, conditions, team formations, and other data sets can lend itself to Artificial Intelligence.

Initial Issues

We needed to figure out which method of Artificial Intelligence is best suited for our problem

Another issue was data related:

- **We were we going to get the Data (those that cost money was not an option)**
- **In which method is the data going to be gathered and stored**
- **Which data is relevant to are question in hand**



Passing

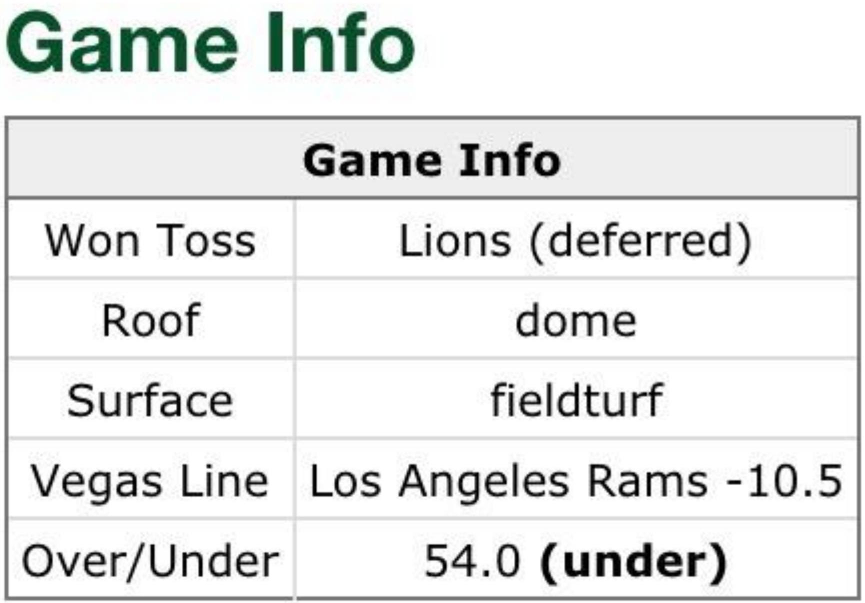
*ProBowl, +1st-tm All-Pro Share & more ▼ Glossary

No.	Player	Age	Pos	G	GS	QBrec	Cmp	Att	Cmp%	Yds	TD	TD%	Int	Int%	Lng	Y/A	AY/A	Y/C	Y/G	Rate	QBR	Sk	Yds	NY/A	ANY/A	Sk%	4QC	GWD
16	Jared Goff	24	QB	12	12	11-1-0	275	414	66.4	3754	27	6.5	7	1.7	70	9.1	9.6	13.7	312.8	109.9	74.1	26	160	8.17	8.68	5.9	4	4
6	Johnny Hekker	28	P	12	0		1	2	50.0	12	0	0.0	0	0.0	12	6.0	6.0	12.0	1.0	68.7		0	0	6.00	6.00	0.0		
	Team Total	**26.0**		**12**		**11-1-0**	**276**	**416**	**66.3**	**3766**	**27**	**6.5**	**7**	**1.7**	**70**	**9.1**	**9.6**	**13.6**	**313.8**	**109.7**		**26**	**160**	**8.16**	**8.67**	**5.9**	**4**	**4**
	Opp Total			**12**			**266**	**408**	**65.2**	**3262**	**26**	**6.4**	**11**	**2.7**		**8.0**	**8.06**	**12.3**	**271.8**	**99.7**		**33**	**275**	**6.8**	**6.8**	**7.5**		

Rushing & Receiving

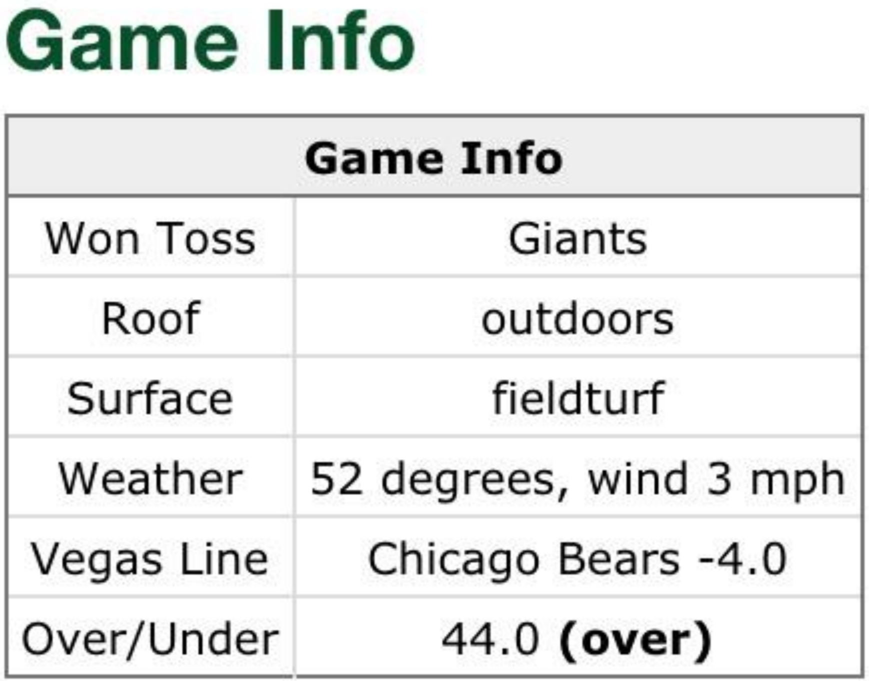
*ProBowl, +1st-tm All-Pro Share & more ▼ Glossary

				Games		Rushing							Receiving									Total Yds				
No.	**Player**	**Age**	**Pos**	**G**	**GS**	**Att**	**Yds**	**TD**	**Lng**	**Y/A**	**Y/G**	**A/G**	**Tgt**	**Rec**	**Yds**	**Y/R**	**TD**	**Lng**	**R/G**	**Y/G**	**Ctch%**	**Touch**	**Y/Tch**	**YScm**	**RRTD**	**Fmb**
30	Todd Gurley	24	RB	12	12	233	1175	15	36	5.0	97.9	19.4	61	46	474	10.3	4	56	3.8	39.5	75.4%	279	5.9	1649	19	1
34	Malcolm Brown	25		12	0	43	212	0	19	4.9	17.7	3.6	7	5	52	10.4	1	18	0.4	4.3	71.4%	48	5.5	264	1	0
16	Jared Goff	24	QB	12	12	37	86	1	16	2.3	7.2	3.1										37	2.3	86	1	7
17	Robert Woods	26	WR	12	12	15	122	0	56	8.1	10.2	1.3	99	64	971	15.2	5	36	5.3	80.9	64.6%	79	13.8	1093	5	0
12	Brandin Cooks	25	WR	12	12	7	41	1	10	5.9	3.4	0.6	90	63	1026	16.3	3	57	5.3	85.5	70.0%	70	15.2	1067	4	1
18	Cooper Kupp	25	WR	8	8	4	25	0	12	6.3	3.1	0.5	56	40	566	14.2	6	70	5.0	70.8	71.4%	44	13.4	591	6	0
14	Sean Mannion	26		1	0	3	1	0	3	0.3	1.0	3.0										3	0.3	1	0	0
83	Josh Reynolds	23	wr	12	4	2	8	0	10	4.0	0.7	0.2	25	15	197	13.1	3	27	1.3	16.4	60.0%	17	12.1	205	3	0
6	Johnny Hekker	28	P	12	0	1	3	0	3	3.0	0.3	0.1										1	3.0	3	0	0
81	Gerald Everett	24		12	0	0	0	0	0		0.0	0.0	30	19	217	11.4	3	40	1.6	18.1	63.3%	19	11.4	217	3	0
89	Tyler Higbee	25	TE	12	12	0	0	0	0		0.0	0.0	28	19	217	11.4	2	33	1.6	18.1	67.9%	19	11.4	217	2	1
11	KhaDarel Hodge	23		10	0	0	0	0	0		0.0	0.0	2	2	17	8.5	0	14	0.2	1.7	100.0%	2	8.5	17	0	0
15	Nick Williams	28		2	0	0	0	0	0		0.0	0.0	3	2	17	8.5	0	11	1.0	8.5	66.7%	2	8.5	17	0	0
37	Sam Shields	31	cb/rcb	12	2	0	0	0	0		0.0	0.0	1	1	12	12.0	0	12	0.1	1.0	100.0%	1	12.0	12	0	0
	Team Total	**26.0**		**12**		**345**	**1673**	**17**	**56**	**4.8**	**139.4**	**28.8**	**403**	**276**	**3766**	**13.6**	**27**	**70**	**23.0**	**313.8**		**621**	**8.8**	**5439**	**44**	**10**
	Opp Total			**12**		**281**	**1421**	**8**		**5.1**	**118.4**	**23.4**		**266**	**2987**	**11.2**	**26**		**22.2**	**248.9**				**4408**	**34**	**13**



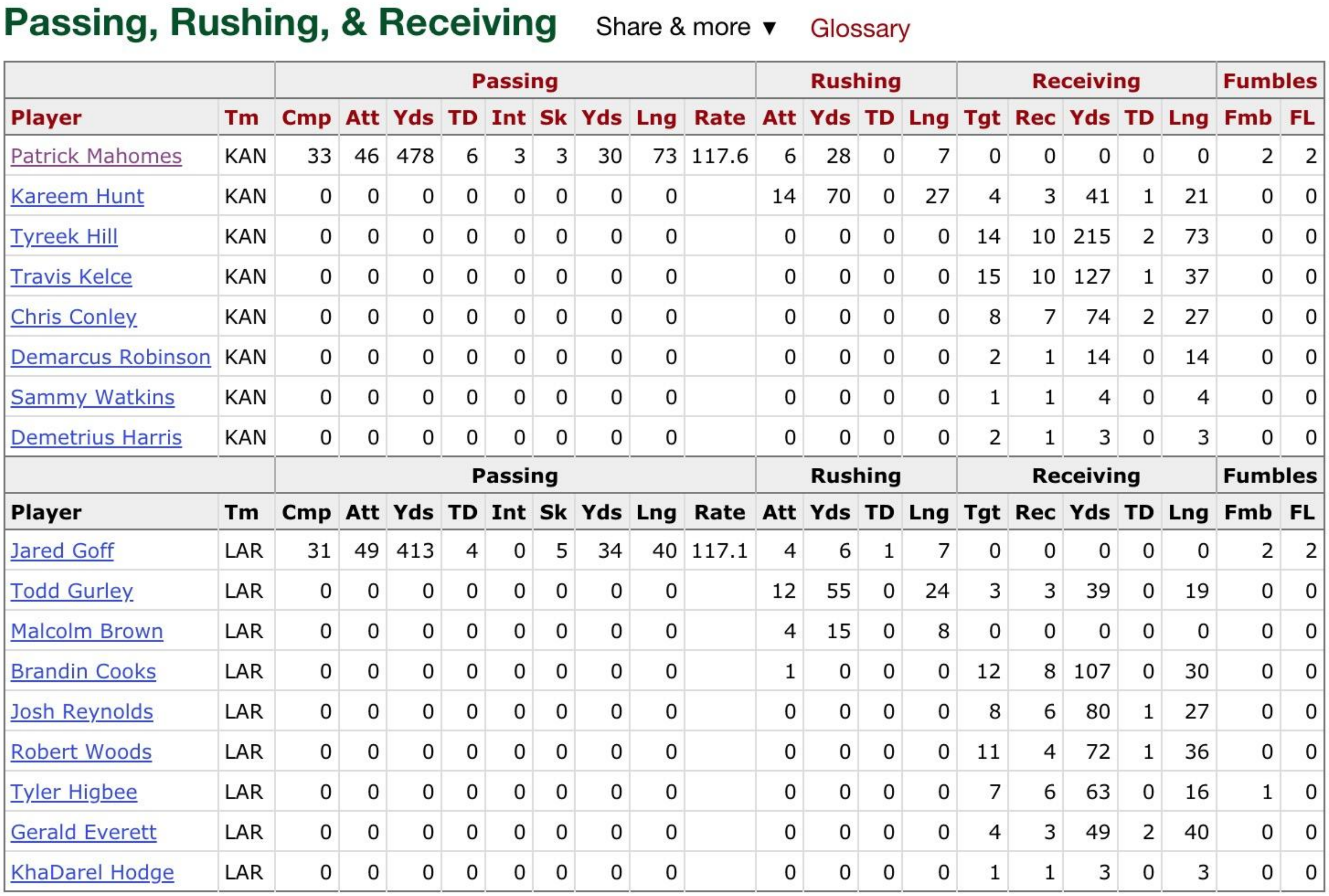
Game Info

Game Info	
Won Toss	Lions (deferred)
Roof	dome
Surface	fieldturf
Vegas Line	Los Angeles Rams -10.5
Over/Under	54.0 **(under)**



Game Info

Game Info	
Won Toss	Giants
Roof	outdoors
Surface	fieldturf
Weather	52 degrees, wind 3 mph
Vegas Line	Chicago Bears -4.0
Over/Under	44.0 **(over)**

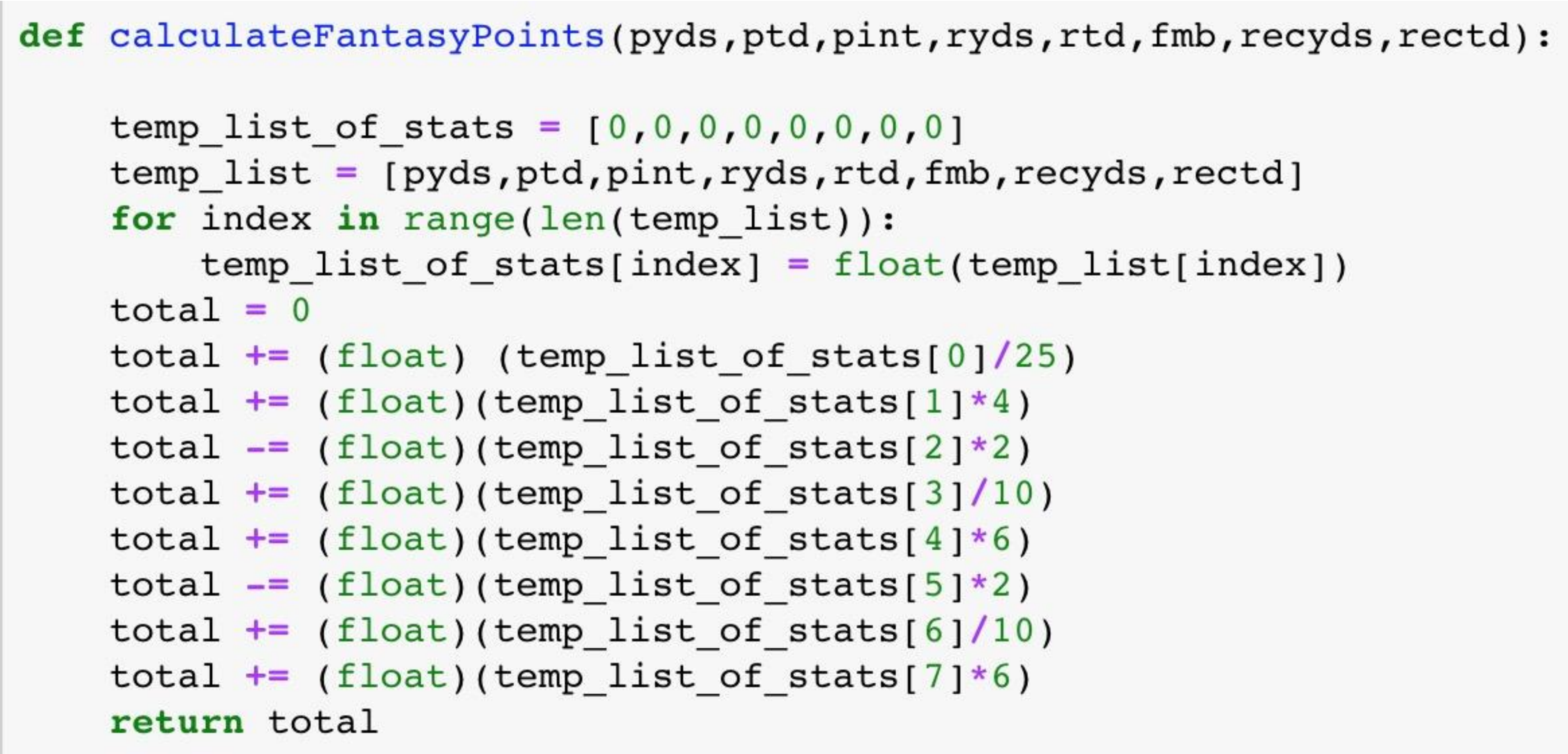


Passing, Rushing, & Receiving

Share & more ▼ Glossary

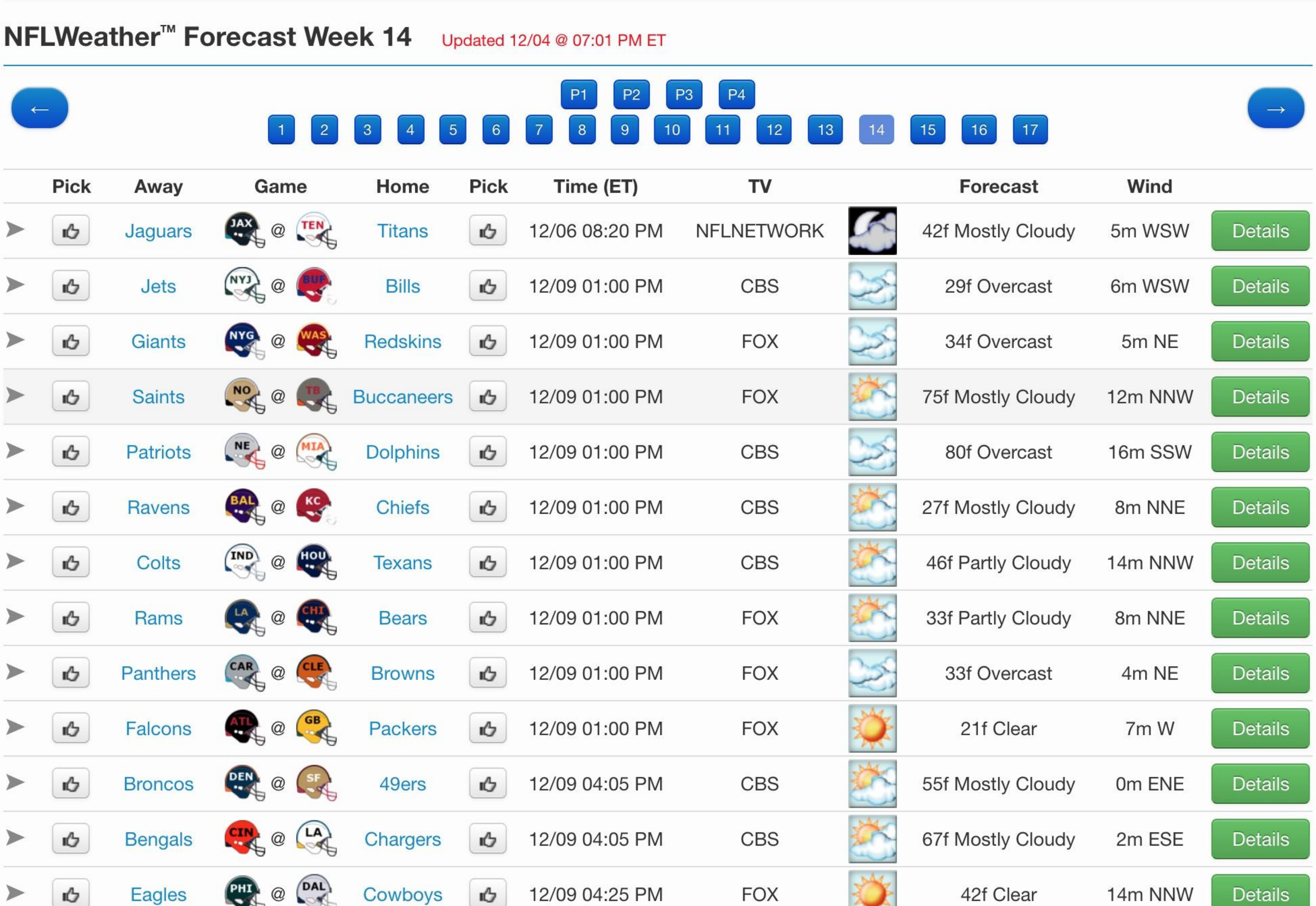
		Passing									Rushing				Receiving					Fumbles	
Player	Tm	Cmp	Att	Yds	TD	Int	Sk	Yds	Lng	Rate	Att	Yds	TD	Lng	Tgt	Rec	Yds	TD	Lng	Fmb	FL
Patrick Mahomes	KAN	33	46	478	6	3	3	30	73	117.6	6	28	0	7	0	0	0	0	0	2	2
Kareem Hunt	KAN	0	0	0	0	0	0	0	0		14	70	0	27	4	3	41	1	21	0	0
Tyreek Hill	KAN	0	0	0	0	0	0	0	0		0	0	0	0	14	10	215	2	73	0	0
Travis Kelce	KAN	0	0	0	0	0	0	0	0		0	0	0	0	15	10	127	1	37	0	0
Chris Conley	KAN	0	0	0	0	0	0	0	0		0	0	0	0	8	7	74	2	27	0	0
Demarcus Robinson	KAN	0	0	0	0	0	0	0	0		0	0	0	0	2	1	14	0	14	0	0
Sammy Watkins	KAN	0	0	0	0	0	0	0	0		0	0	0	0	1	1	4	0	4	0	0
Demetrius Harris	KAN	0	0	0	0	0	0	0	0		0	0	0	0	2	1	3	0	3	0	0

		Passing									Rushing				Receiving					Fumbles	
Player	Tm	Cmp	Att	Yds	TD	Int	Sk	Yds	Lng	Rate	Att	Yds	TD	Lng	Tgt	Rec	Yds	TD	Lng	Fmb	FL
Jared Goff	LAR	31	49	413	4	0	5	34	40	117.1	4	6	1	7	0	0	0	0	0	2	2
Todd Gurley	LAR	0	0	0	0	0	0	0	0		12	55	0	24	3	3	39	0	19	0	0
Malcolm Brown	LAR	0	0	0	0	0	0	0	0		4	15	0	8	0	0	0	0	0	0	0
Brandin Cooks	LAR	0	0	0	0	0	0	0	0		1	0	0	0	12	8	107	0	30	0	0
Josh Reynolds	LAR	0	0	0	0	0	0	0	0		0	0	0	0	8	6	80	1	27	0	0
Robert Woods	LAR	0	0	0	0	0	0	0	0		0	0	0	0	11	4	72	1	36	0	0
Tyler Higbee	LAR	0	0	0	0	0	0	0	0		0	0	0	0	7	6	63	0	16	1	0
Gerald Everett	LAR	0	0	0	0	0	0	0	0		0	0	0	0	4	3	49	2	40	0	0
KhaDarel Hodge	LAR	0	0	0	0	0	0	0	0		0	0	0	0	1	1	3	0	3	0	0



```python
def calculateFantasyPoints(pyds,ptd,pint,ryds,rtd,fmb,recyds,rectd):

    temp_list_of_stats = [0,0,0,0,0,0,0,0]
    temp_list = [pyds,ptd,pint,ryds,rtd,fmb,recyds,rectd]
    for index in range(len(temp_list)):
        temp_list_of_stats[index] = float(temp_list[index])
    total = 0
    total += (float) (temp_list_of_stats[0]/25)
    total += (float)(temp_list_of_stats[1]*4)
    total -= (float)(temp_list_of_stats[2]*2)
    total += (float)(temp_list_of_stats[3]/10)
    total += (float)(temp_list_of_stats[4]*6)
    total -= (float)(temp_list_of_stats[5]*2)
    total += (float)(temp_list_of_stats[6]/10)
    total += (float)(temp_list_of_stats[7]*6)
    return total
```

```
Deshaun Watson: 18.707111111111
Carson Wentz: 19.136666666666667
Philip Rivers: 19.669999999999998
Kareem Hunt: 19.726666666666667
Ben Roethlisberger: 19.89
Aaron Rodgers: 20.012
Jared Goff: 20.062666666666665
Nick Mullens: 20.18
Mitchell Trubisky: 20.274
James Conner: 20.32
Alvin Kamara: 20.47
Andrew Luck: 20.476
Melvin Gordon: 20.685714285714287
Drew Brees: 21.75
Cam Newton: 22.244
Ryan Fitzpatrick: 22.672
Matt Ryan: 24.014
Todd Gurley: 24.104444444444447
Patrick Mahomes: 25.967555555555553
```



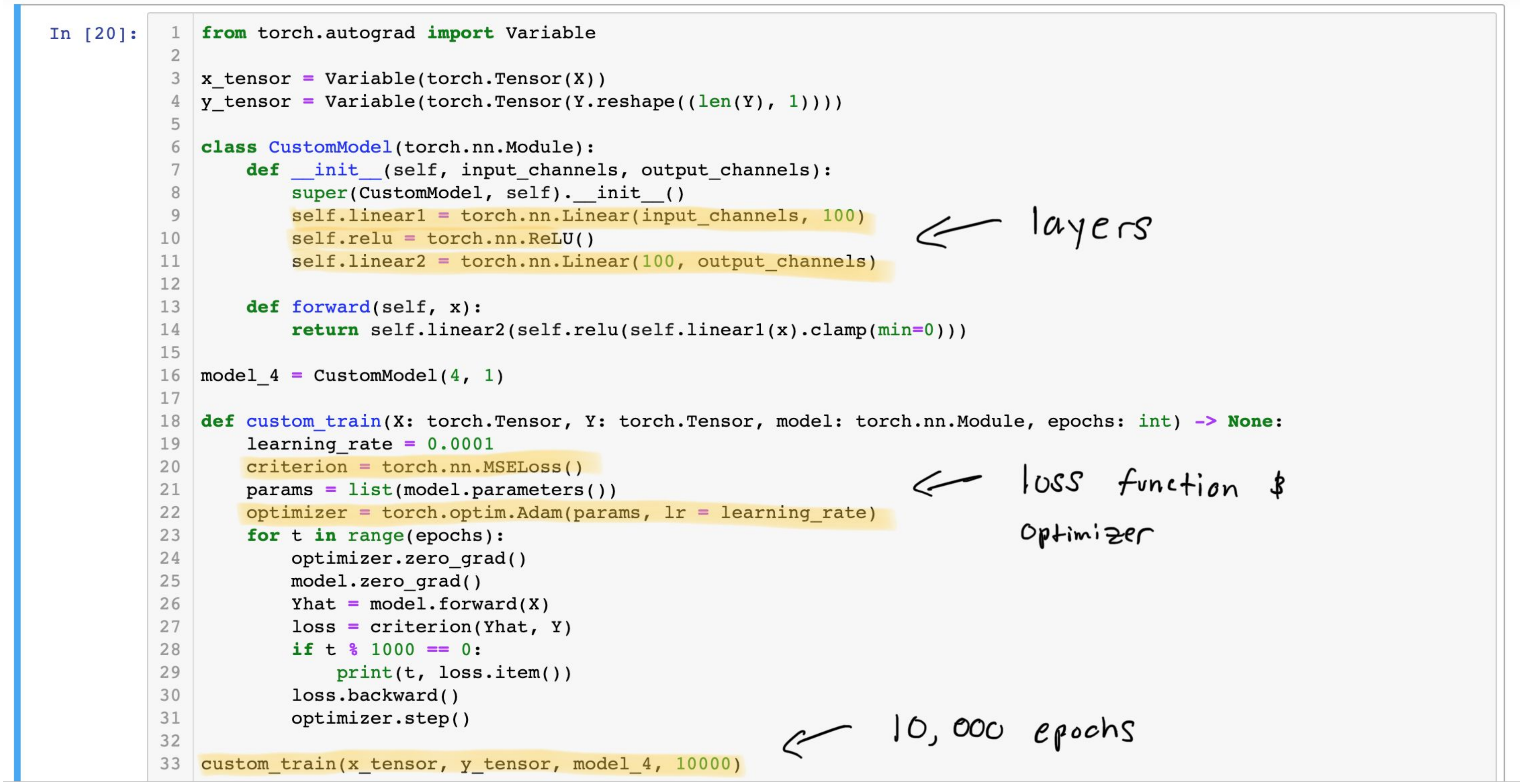
NFLWeather™ Forecast Week 14

Updated 12/04 @ 07:01 PM ET

	Pick	Away	Game	Home	Pick	Time (ET)	TV		Forecast	Wind	
➤	👍	Jaguars	JAX @ TEN	Titans	👍	12/06 08:20 PM	NFLNETWORK		42f Mostly Cloudy	5m WSW	Details
➤	👍	Jets	NYJ @ BUF	Bills	👍	12/09 01:00 PM	CBS		29f Overcast	6m WSW	Details
➤	👍	Giants	NYG @ WAS	Redskins	👍	12/09 01:00 PM	FOX		34f Overcast	5m NE	Details
➤	👍	Saints	NO @ TB	Buccaneers	👍	12/09 01:00 PM	FOX		75f Mostly Cloudy	12m NNW	Details
➤	👍	Patriots	NE @ MIA	Dolphins	👍	12/09 01:00 PM	CBS		80f Overcast	16m SSW	Details
➤	👍	Ravens	BAL @ KC	Chiefs	👍	12/09 01:00 PM	CBS		27f Mostly Cloudy	8m NNE	Details
➤	👍	Colts	IND @ HOU	Texans	👍	12/09 01:00 PM	CBS		46f Partly Cloudy	14m NNW	Details
➤	👍	Rams	LA @ CHI	Bears	👍	12/09 01:00 PM	FOX		33f Partly Cloudy	8m NNE	Details
➤	👍	Panthers	CAR @ CLE	Browns	👍	12/09 01:00 PM	FOX		33f Overcast	4m NE	Details
➤	👍	Falcons	ATL @ GB	Packers	👍	12/09 01:00 PM	FOX		21f Clear	7m W	Details
➤	👍	Broncos	DEN @ SF	49ers	👍	12/09 04:05 PM	CBS		55f Mostly Cloudy	0m ENE	Details
➤	👍	Bengals	CIN @ LA	Chargers	👍	12/09 04:05 PM	CBS		67f Mostly Cloudy	2m ESE	Details
➤	👍	Eagles	PHI @ DAL	Cowboys	👍	12/09 04:25 PM	FOX		42f Clear	14m NNW	Details

Models

- Initially started with three Sequential models with loss in the 30’s and high 20’s.
- Developed a custom model with 4 input layers, 100 hidden layers, an ReLU layer, and one output layer that achieves a loss of around 28.73.



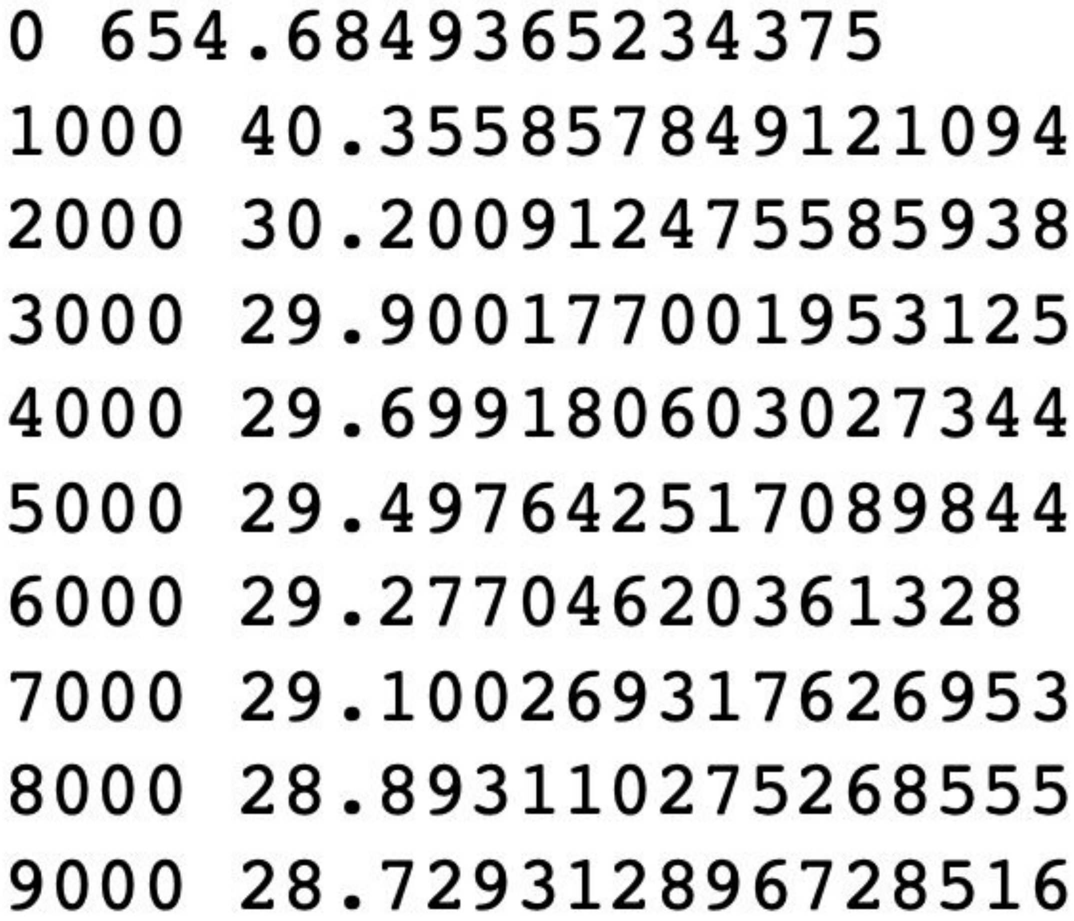
In [20]:

```python
from torch.autograd import Variable

x_tensor = Variable(torch.Tensor(X))
y_tensor = Variable(torch.Tensor(Y.reshape((len(Y), 1))))

class CustomModel(torch.nn.Module):
    def __init__(self, input_channels, output_channels):
        super(CustomModel, self).__init__()
        self.linear1 = torch.nn.Linear(input_channels, 100)
        self.relu = torch.nn.ReLU()
        self.linear2 = torch.nn.Linear(100, output_channels)

    def forward(self, x):
        return self.linear2(self.relu(self.linear1(x).clamp(min=0)))

model_4 = CustomModel(4, 1)

def custom_train(X: torch.Tensor, Y: torch.Tensor, model: torch.nn.Module, epochs: int) -> None:
    learning_rate = 0.0001
    criterion = torch.nn.MSELoss()
    params = list(model.parameters())
    optimizer = torch.optim.Adam(params, lr = learning_rate)
    for t in range(epochs):
        optimizer.zero_grad()
        model.zero_grad()
        Yhat = model.forward(X)
        loss = criterion(Yhat, Y)
        if t % 1000 == 0:
            print(t, loss.item())
        loss.backward()
        optimizer.step()

custom_train(x_tensor, y_tensor, model_4, 10000)
```

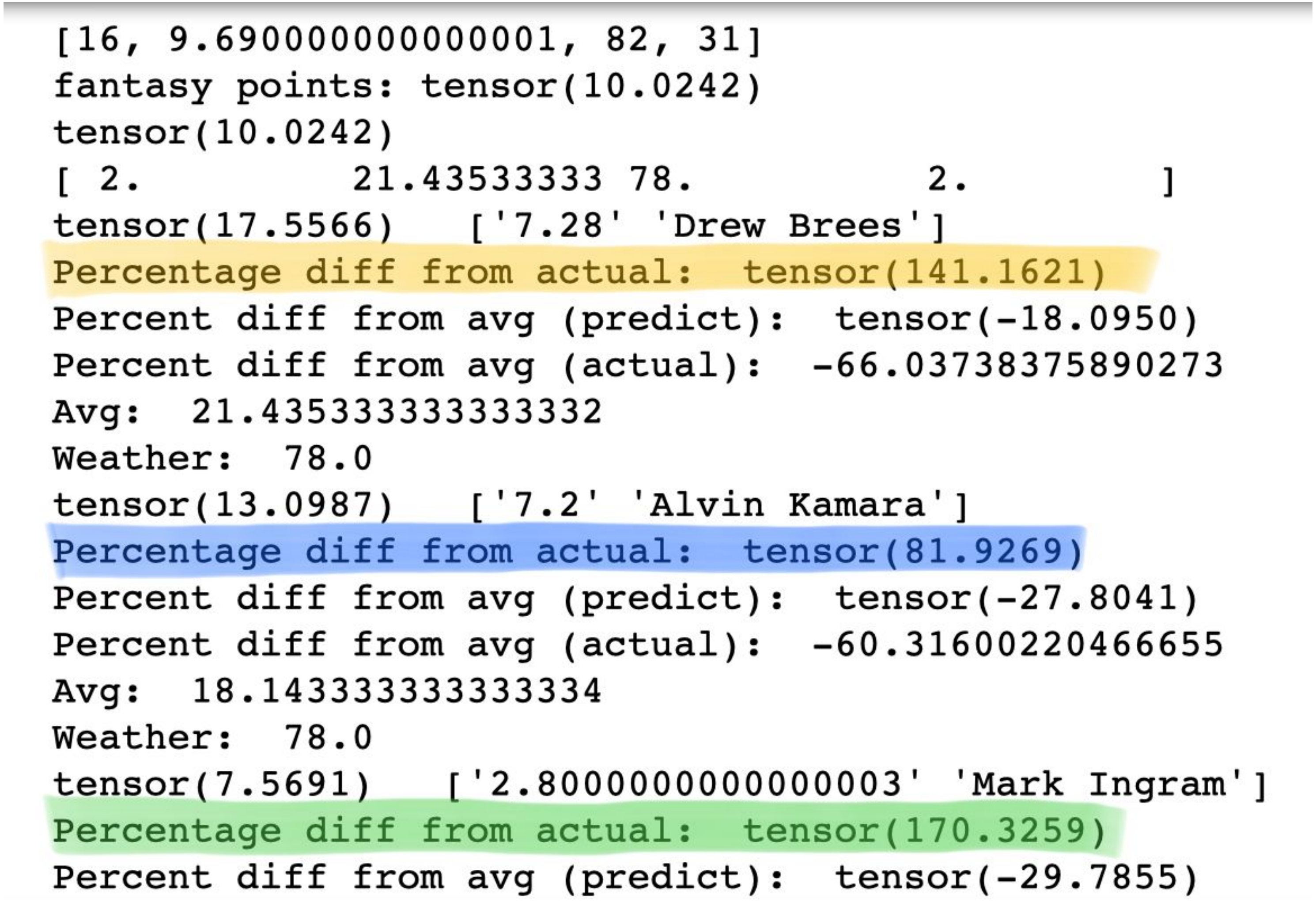
← layers

← loss function & optimizer

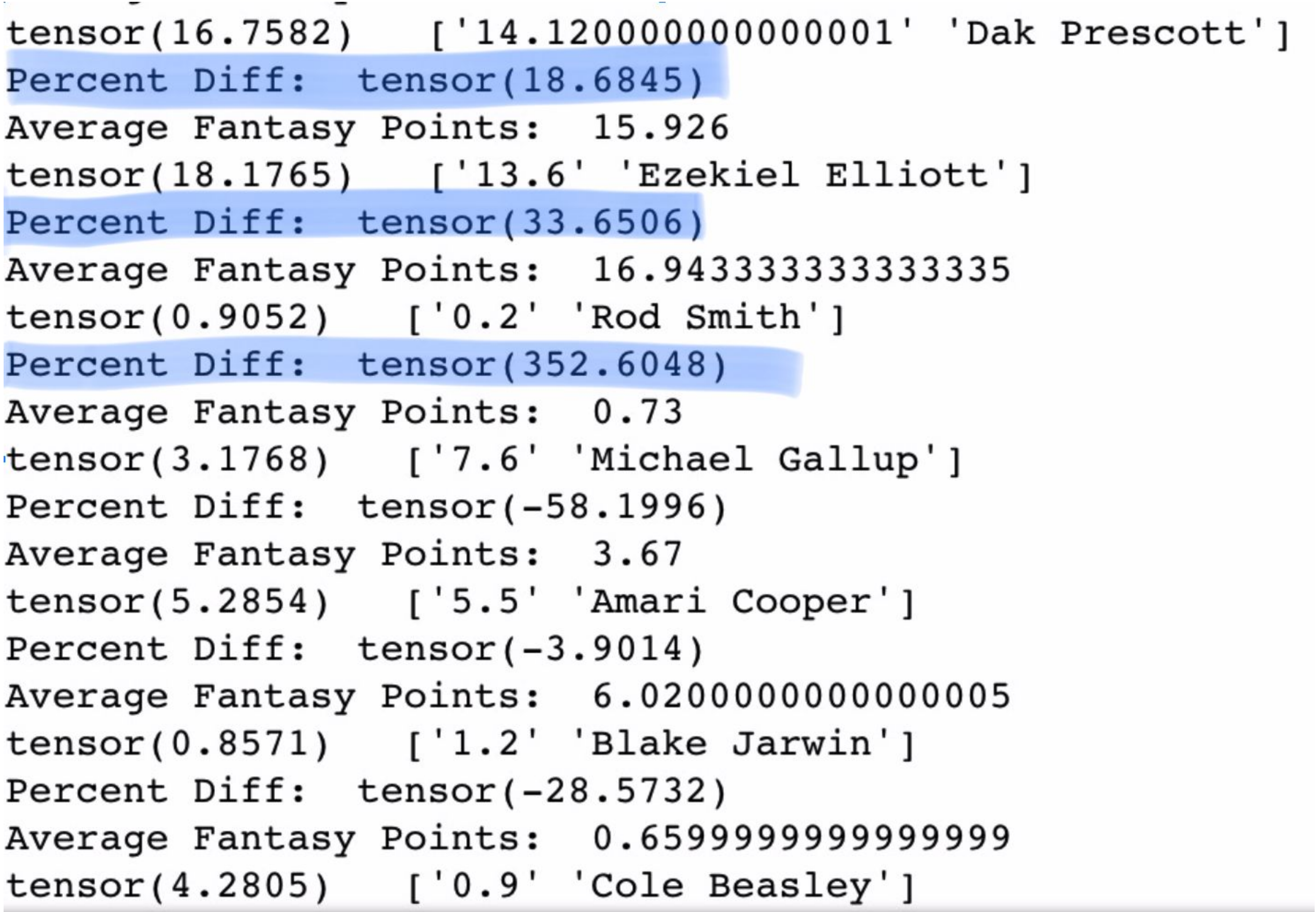
← 10,000 epochs



```
0 654.6849365234375
1000 40.355857849121094
2000 30.200912475585938
3000 29.900177001953125
4000 29.699180603027344
5000 29.497642517089844
6000 29.27704620361328
7000 29.100269317626953
8000 28.893110275268555
9000 28.729312896728516
```



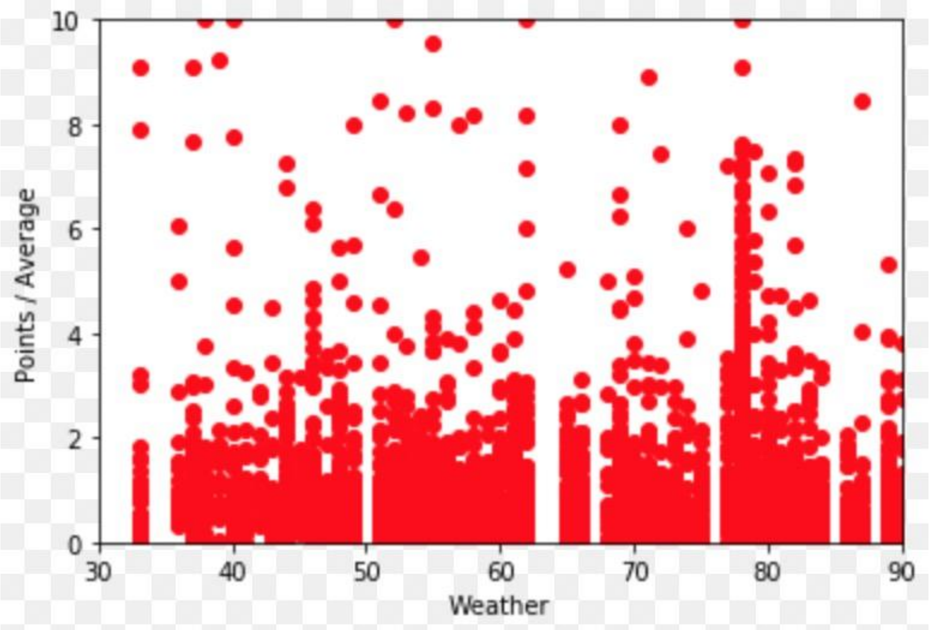
```
[16, 9.690000000000001, 82, 31]
fantasy points: tensor(10.0242)
tensor(10.0242)
[ 2.         21.43533333 78.          2.        ]
tensor(17.5566)   ['7.28' 'Drew Brees']
Percentage diff from actual:  tensor(141.1621)
Percent diff from avg (predict):  tensor(-18.0950)
Percent diff from avg (actual):  -66.03738375890273
Avg:  21.435333333333332
Weather:  78.0
tensor(13.0987)   ['7.2' 'Alvin Kamara']
Percentage diff from actual:  tensor(81.9269)
Percent diff from avg (predict):  tensor(-27.8041)
Percent diff from avg (actual):  -60.31600220466655
Avg:  18.143333333333334
Weather:  78.0
tensor(7.5691)   ['2.8000000000000003' 'Mark Ingram']
Percentage diff from actual:  tensor(170.3259)
Percent diff from avg (predict):  tensor(-29.7855)
```



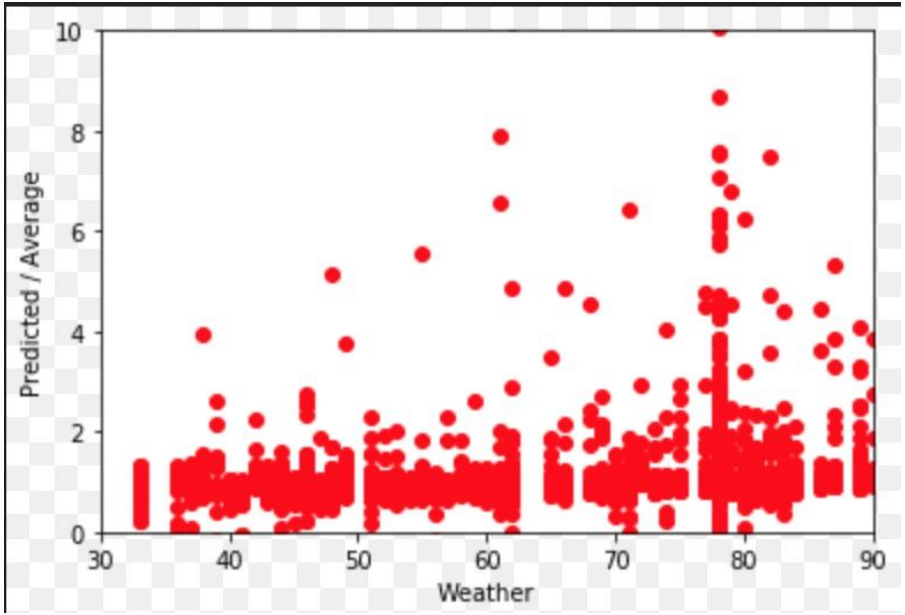
```
tensor(16.7582)   ['14.120000000000001' 'Dak Prescott']
Percent Diff:  tensor(18.6845)
Average Fantasy Points:  15.926
tensor(18.1765)   ['13.6' 'Ezekiel Elliott']
Percent Diff:  tensor(33.6506)
Average Fantasy Points:  16.943333333333335
tensor(0.9052)   ['0.2' 'Rod Smith']
Percent Diff:  tensor(352.6048)
Average Fantasy Points:  0.73
tensor(3.1768)   ['7.6' 'Michael Gallup']
Percent Diff:  tensor(-58.1996)
Average Fantasy Points:  3.67
tensor(5.2854)   ['5.5' 'Amari Cooper']
Percent Diff:  tensor(-3.9014)
Average Fantasy Points:  6.0200000000000005
tensor(0.8571)   ['1.2' 'Blake Jarwin']
Percent Diff:  tensor(-28.5732)
Average Fantasy Points:  0.6599999999999999
tensor(4.2805)   ['0.9' 'Cole Beasley']
```

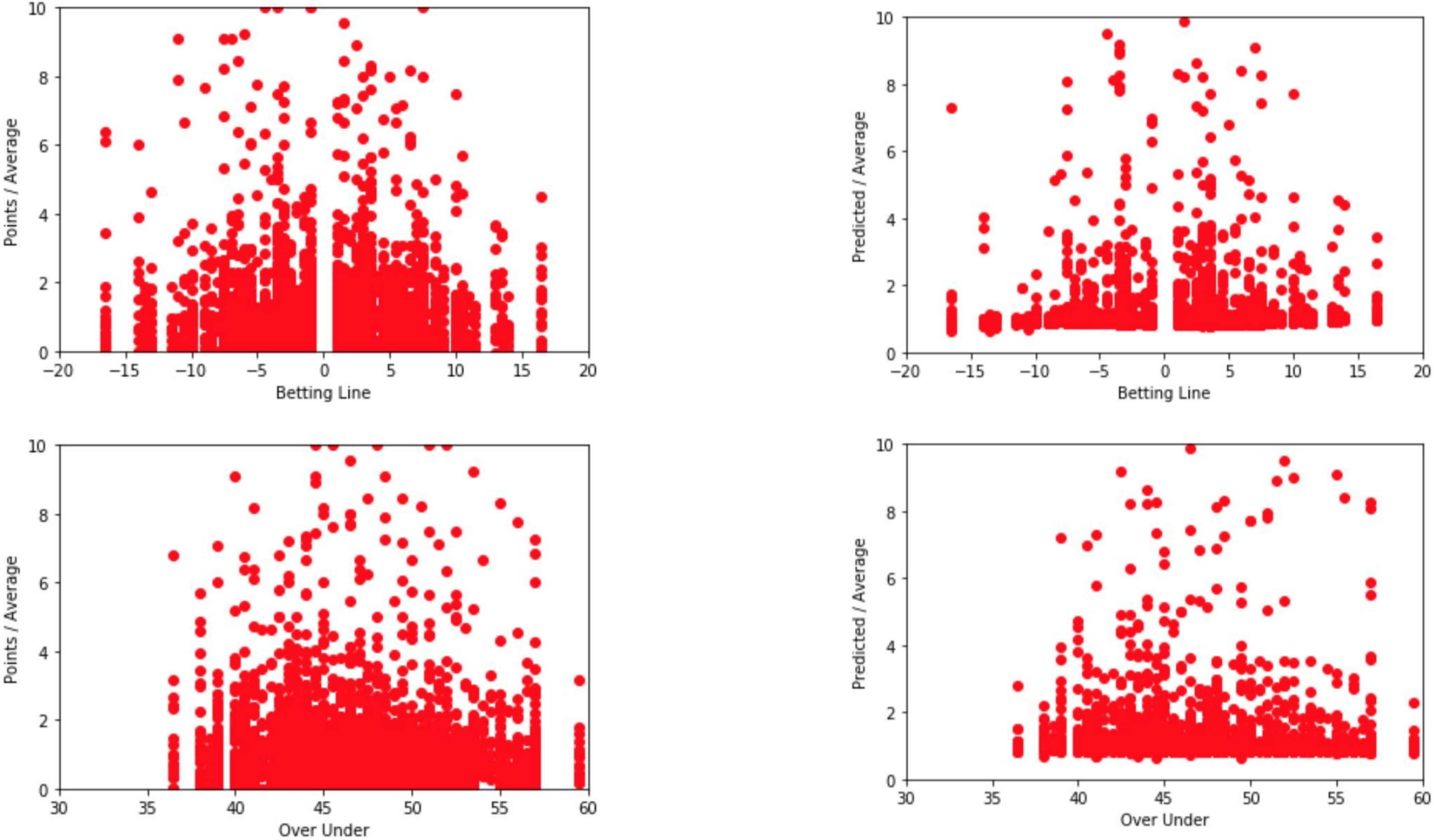
Weather vs Points

Actual



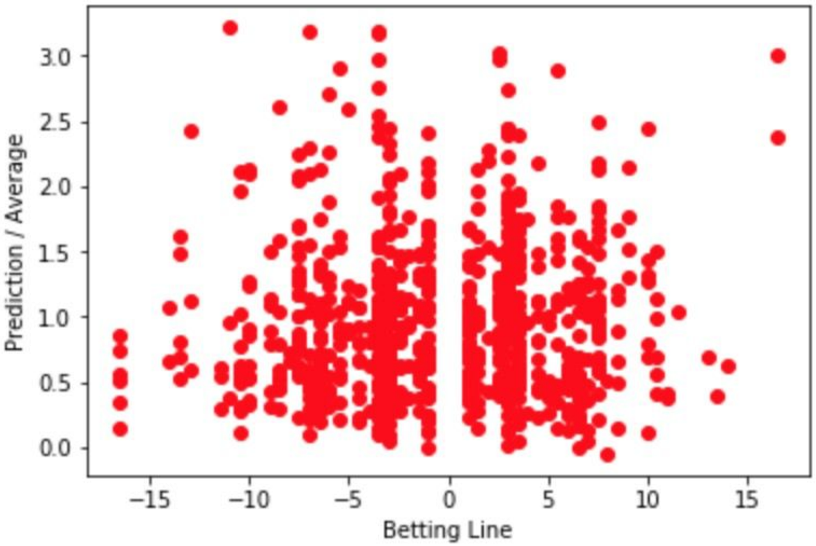
Predicted



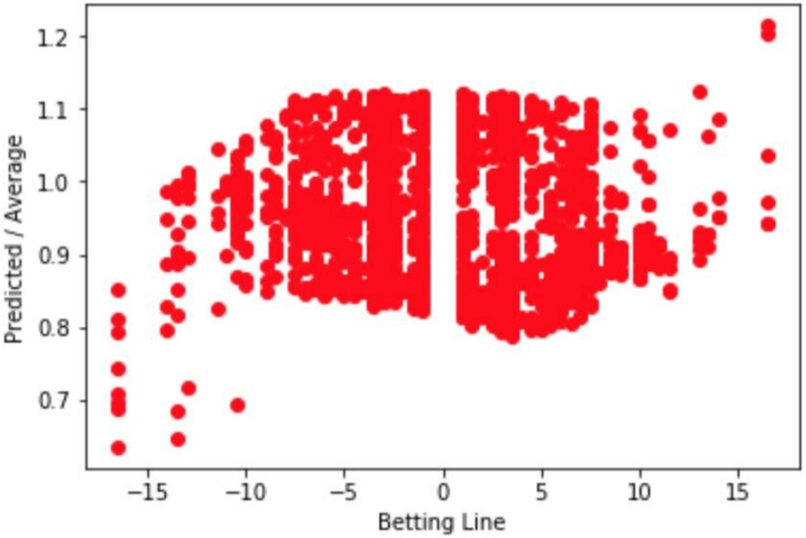

Points / Average
Betting Line
10
8
6
4
2
0
−20
−15
−10
−5
0
5
10
15
20
Predicted / Average
Betting Line
10
8
6
4
2
0
−20
−15
−10
−5
0
5
10
15
20
Points / Average
Over Under
10
8
6
4
2
0
30
35
40
45
50
55
60
Predicted / Average
Over Under
10
8
6
4
2
0
30
35
40
45
50
55
60

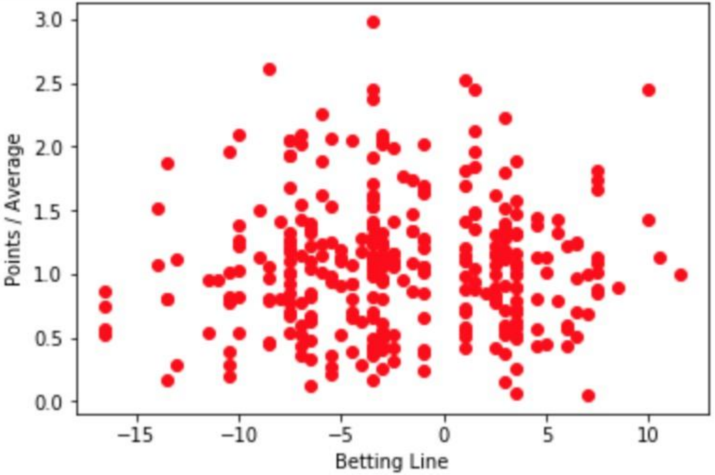
Betting Line vs Fantasy Points (> 5)

Actual

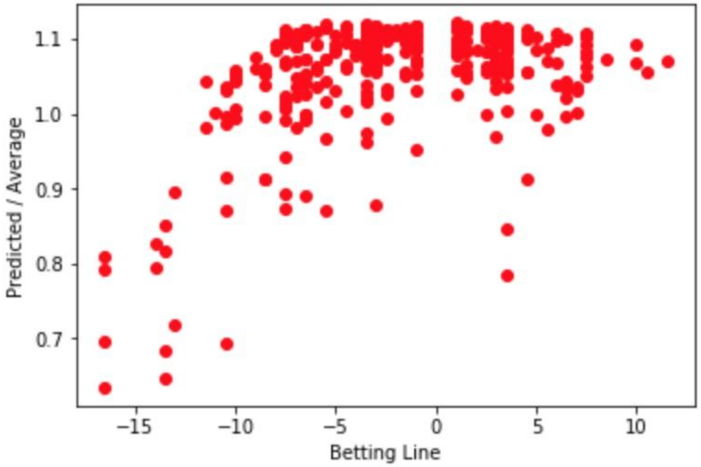


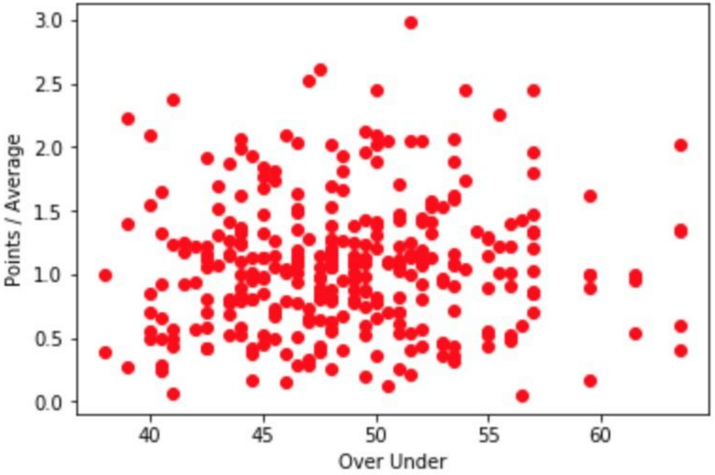
Prediction

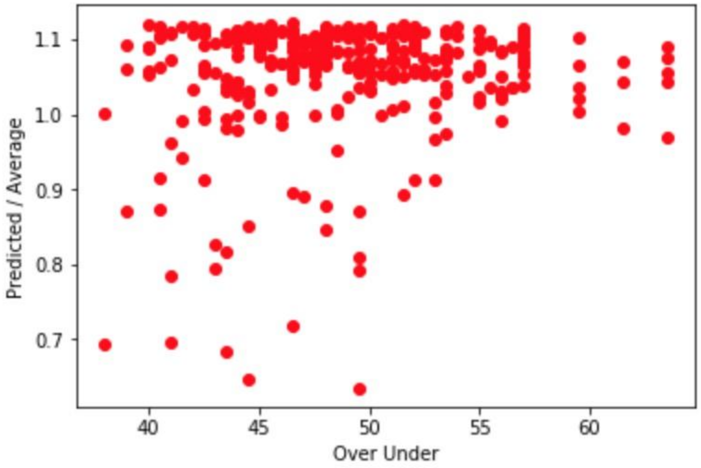




Betting Line vs Fantasy Points (> 15)







Next Steps

- Train on an Nvidia GeForce GTX 1070 to see if that helps improve the loss / speed of the custom model.
- Change and optimize the model (i.e., try different loss functions, different optimizers, etc).
- Create a new model that reduces noise
- Maybe add previous season data for more data
- Maybe create a model to predict something else (betting line / wins)