7

October 14, 2018

1 Assignment.

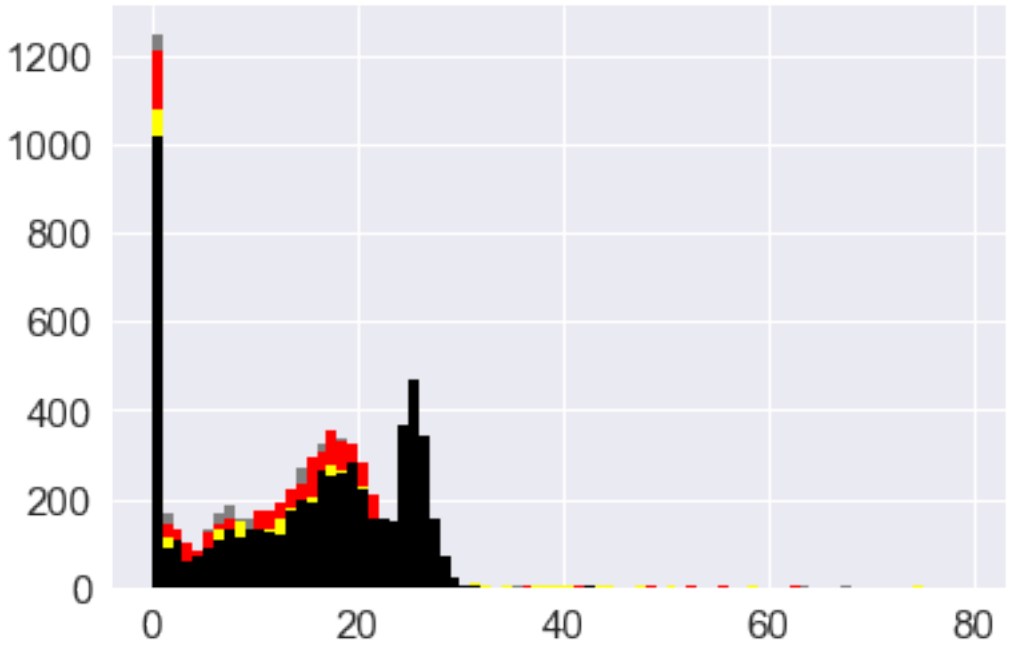
In [1]:
```
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(font_scale=1.5)
import numpy as np

from pylab import rcParams
rcParams['figure.figsize'] = 20, 10

from sklearn.linear_model import LogisticRegression as Model
```

In [2]:
```
kobe = pd.read_csv('../data/kobe.csv')
kobe.dropna(inplace=True)
```

1.0.1 Warmup. Perform some analysis on Kobe's shot selection. Ask and answer (with charts) questions such as: Does Kobe make more shots in the 4th quarter than on average? Does Kobe make more shots from the left more than the right? What was Kobe's best year for shooting percentage? Etc. The more naunced the more you'll have a feel for the data.

In [3]:
```
# First, I'm not a basketball fan, so I presumed periods are quarters.
# Does Kobe make more shots in the 4th quarter than on average?


kobe[(kobe.period==3)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="grey")
kobe[(kobe.period==1)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="red")
kobe[(kobe.period==2)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="yellow
kobe[(kobe.period==4)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="black"
```

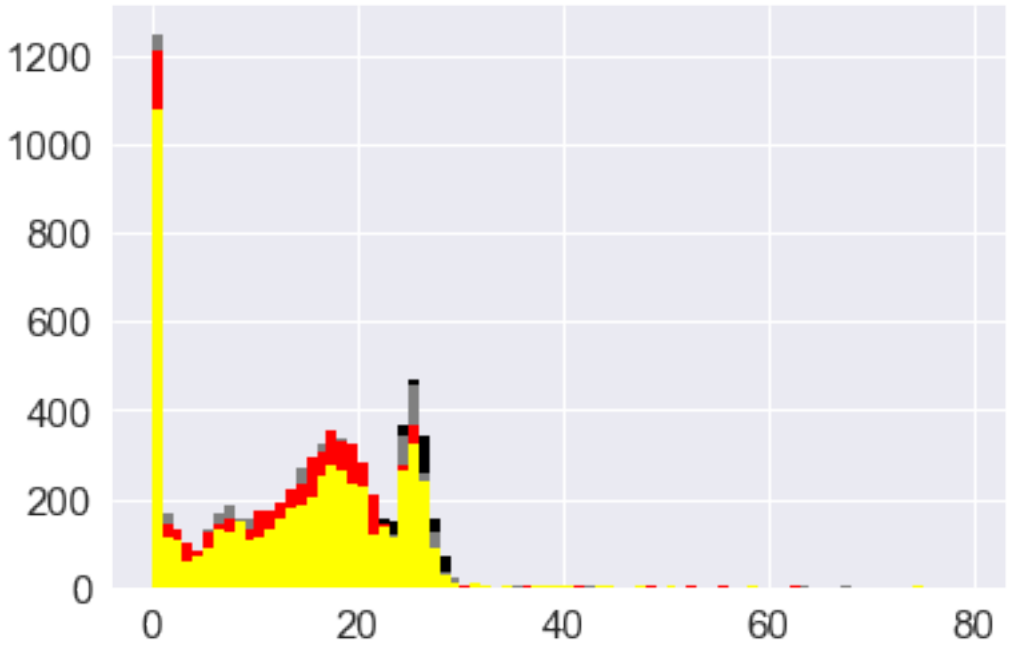
Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x10af95b00>
```



In [4]:
```
# Does Kobe make more shots in the 4th quarter than on average?

print("Kobe tends to shoot less over time. Maybe exhaustion sets in?")
kobe[(kobe.period==4)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="black"
kobe[(kobe.period==3)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="grey")
kobe[(kobe.period==1)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="red")
kobe[(kobe.period==2)].shot_distance.hist(bins=np.arange(0,80,1), alpha=1, color="yellow
```

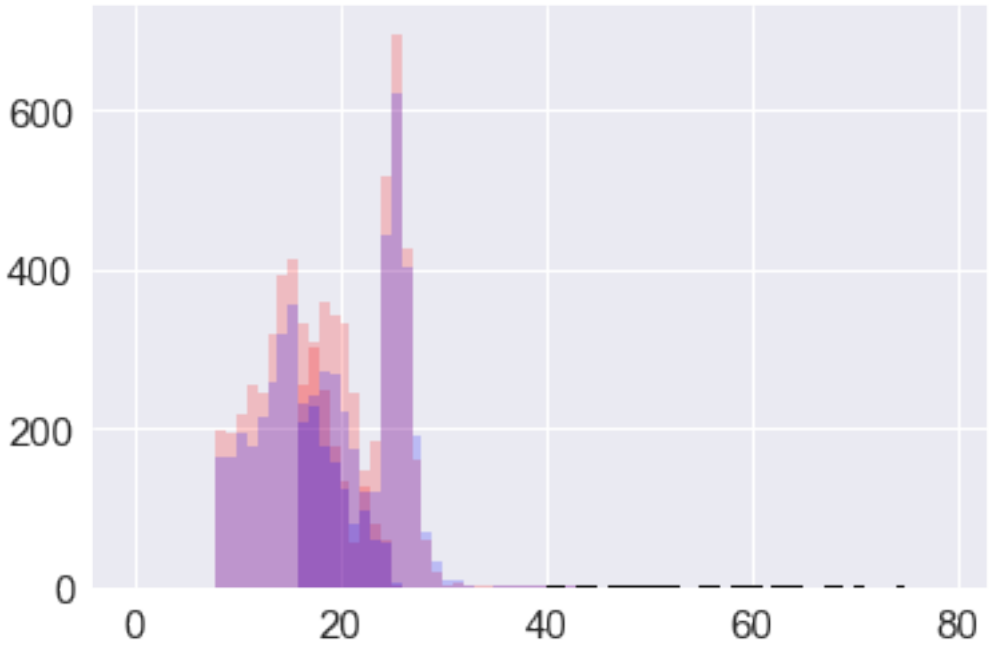
```
Kobe tends to shoot less over time. Maybe exhaustion sets in?
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x107d9c9b0>
```



In [5]:
```
# Does Kobe make more shots from the left more than the right?
# Generally, Kobe shoots slightly more often from the right side of the
# court than the left

kobe[kobe.shot_zone_area == "Right Side(R)"].shot_distance.hist(bins=np.arange(0,70,1),
kobe[kobe.shot_zone_area == "Right Side Center(RC)"].shot_distance.hist(bins=np.arange(0
kobe[kobe.shot_zone_area == "Left Side Center(LC)"].shot_distance.hist(bins=np.arange(0,
kobe[kobe.shot_zone_area == "Left Side(L)"].shot_distance.hist(bins=np.arange(0,70,1), a

# Kobe rarely throws beyond 40 ft, but when he does, he throws as far as
# near 80 ft
kobe[kobe.shot_zone_area == "Back Court(BC)"].shot_distance.hist(bins=np.arange(0,80,1),
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x10abaa390>



1.0.2 1. Create a new column called `abs_x` that is equal to the absolute value of loc_x. Plot a histogram of made shots and missed shots using this variable. Explain in detail (with graphics and evidence) why this could be a better feature/column to use in a Logsitic Regression model instead of `loc_x`.

```
In [6]: kobe['abs_x'] = list(kobe.loc_x)
        list(kobe)
```

```
Out[6]: ['action_type',
         'combined_shot_type',
         'game_event_id',
         'game_id',
         'lat',
         'loc_x',
         'loc_y',
         'lon',
         'minutes_remaining',
         'period',
         'playoffs',
         'season',
         'seconds_remaining',
         'shot_distance',
         'shot_made_flag',
         'shot_type',
         'shot_zone_area',
         'shot_zone_basic',
```

```
 'shot_zone_range',
 'team_id',
 'team_name',
 'game_date',
 'matchup',
 'opponent',
 'shot_id',
 'abs_x']
```

In [7]:
```
kobe.abs_x
```

Out[7]:
```
1       -157
2       -101
3        138
4          0
5       -145
6          0
8        -65
9        -33
10       -94
11       121
12       -67
13       -94
14       -23
15        62
17      -117
18      -132
20         3
21       134
22       -16
23      -109
24       -46
25         0
26       -58
27      -183
28        85
29         3
30       121
31       127
38        91
39       -27
         ...
30661      0
30662     -8
30663    106
30665    -14
30666    -81
30667     40
```

```
30669    171
30670    -74
30671      0
30672     89
30673    117
30674    117
30675   -134
30676   -141
30677   -113
30678     14
30679      0
30681    -18
30683      1
30684    -96
30685     81
30687     40
30688   -126
30689    -12
30690   -113
30691      0
30692      1
30694   -134
30695     31
30696      1
Name: abs_x, Length: 25697, dtype: int64
```

```
In [8]: np.where(kobe.abs_x != kobe.loc_x)
        # Confirmed that both have same values
```

```
Out[8]: (array([], dtype=int64),)
```

```
In [9]: kobe
```

```
Out[9]:               action_type combined_shot_type  game_event_id   game_id  \
1                Jump Shot          Jump Shot             12  20000012
2                Jump Shot          Jump Shot             35  20000012
3                Jump Shot          Jump Shot             43  20000012
4       Driving Dunk Shot               Dunk            155  20000012
5                Jump Shot          Jump Shot            244  20000012
6               Layup Shot              Layup            251  20000012
8                Jump Shot          Jump Shot            265  20000012
9       Running Jump Shot          Jump Shot            294  20000012
10               Jump Shot          Jump Shot            309  20000012
11               Jump Shot          Jump Shot              4  20000019
12      Running Jump Shot          Jump Shot             27  20000019
13               Jump Shot          Jump Shot             66  20000019
14               Jump Shot          Jump Shot             80  20000019
15               Jump Shot          Jump Shot             86  20000019
17               Jump Shot          Jump Shot            138  20000019
```

```
18              Jump Shot          Jump Shot            244  20000019
20              Jump Shot          Jump Shot            255  20000019
21              Jump Shot          Jump Shot            265  20000019
22      Running Jump Shot          Jump Shot            274  20000019
23      Running Jump Shot          Jump Shot            299  20000019
24      Running Jump Shot          Jump Shot            307  20000019
25             Layup Shot              Layup            332  20000019
26              Jump Shot          Jump Shot            345  20000019
27              Jump Shot          Jump Shot            369  20000019
28              Jump Shot          Jump Shot            400  20000019
29              Jump Shot          Jump Shot            429  20000019
30      Running Jump Shot          Jump Shot            488  20000019
31              Jump Shot          Jump Shot            499  20000019
38              Jump Shot          Jump Shot            184  20000047
39              Jump Shot          Jump Shot            202  20000047
...                   ...                ...            ...       ...
30661      Slam Dunk Shot               Dunk            245  49900087
30662           Jump Shot          Jump Shot            259  49900087
30663           Jump Shot          Jump Shot            270  49900087
30665          Layup Shot              Layup            280  49900087
30666           Jump Shot          Jump Shot            295  49900087
30667           Jump Shot          Jump Shot            368  49900087
30669           Jump Shot          Jump Shot            425  49900087
30670   Running Jump Shot          Jump Shot             15  49900088
30671  Driving Layup Shot              Layup             25  49900088
30672           Jump Shot          Jump Shot             29  49900088
30673           Jump Shot          Jump Shot             36  49900088
30674           Jump Shot          Jump Shot             81  49900088
30675           Jump Shot          Jump Shot             84  49900088
30676   Running Jump Shot          Jump Shot             98  49900088
30677           Jump Shot          Jump Shot            101  49900088
30678  Driving Layup Shot              Layup            181  49900088
30679          Layup Shot              Layup            212  49900088
30681           Jump Shot          Jump Shot            218  49900088
30683           Jump Shot          Jump Shot            228  49900088
30684           Jump Shot          Jump Shot            231  49900088
30685           Jump Shot          Jump Shot            249  49900088
30687           Jump Shot          Jump Shot            284  49900088
30688           Jump Shot          Jump Shot            308  49900088
30689           Jump Shot          Jump Shot            326  49900088
30690           Jump Shot          Jump Shot            331  49900088
30691  Driving Layup Shot              Layup            382  49900088
30692           Jump Shot          Jump Shot            397  49900088
30694   Running Jump Shot          Jump Shot            426  49900088
30695           Jump Shot          Jump Shot            448  49900088
30696           Jump Shot          Jump Shot            471  49900088

           lat  loc_x  loc_y       lon  minutes_remaining  period  ...    \
```

1	34.0443	-157	0	-118.4268	10	1	...
2	33.9093	-101	135	-118.3708	7	1	...
3	33.8693	138	175	-118.1318	6	1	...
4	34.0443	0	0	-118.2698	6	2	...
5	34.0553	-145	-11	-118.4148	9	3	...
6	34.0443	0	0	-118.2698	8	3	...
8	33.9363	-65	108	-118.3348	6	3	...
9	33.9193	-33	125	-118.3028	3	3	...
10	33.8063	-94	238	-118.3638	1	3	...
11	33.9173	121	127	-118.1488	11	1	...
12	33.9343	-67	110	-118.3368	7	1	...
13	34.0403	-94	4	-118.3638	2	1	...
14	33.9973	-23	47	-118.2928	1	1	...
15	33.8523	62	192	-118.2078	0	1	...
17	33.8183	-117	226	-118.3868	8	2	...
18	33.9473	-132	97	-118.4018	11	3	...
20	33.9003	3	144	-118.2668	10	3	...
21	33.9173	134	127	-118.1358	9	3	...
22	33.9343	-16	110	-118.2858	7	3	...
23	33.8943	-109	150	-118.3788	5	3	...
24	33.9813	-46	63	-118.3158	5	3	...
25	34.0443	0	0	-118.2698	2	3	...
26	33.8483	-58	196	-118.3278	2	3	...
27	33.8583	-183	186	-118.4528	0	3	...
28	33.8713	85	173	-118.1848	8	4	...
29	33.9573	3	87	-118.2668	6	4	...
30	34.0403	121	4	-118.1488	1	4	...
31	34.0103	127	34	-118.1428	0	4	...
38	33.8603	91	184	-118.1788	3	2	...
39	33.7723	-27	272	-118.2968	0	2	...
...	...	...	...	...	...	...	...
30661	34.0443	0	0	-118.2698	9	3	...
30662	33.9913	-8	53	-118.2778	8	3	...
30663	34.0193	106	25	-118.1638	6	3	...
30665	34.0263	-14	18	-118.2838	5	3	...
30666	33.8733	-81	171	-118.3508	4	3	...
30667	33.7943	40	250	-118.2298	9	4	...
30669	33.9913	171	53	-118.0988	3	4	...
30670	34.0283	-74	16	-118.3438	9	1	...
30671	34.0443	0	0	-118.2698	8	1	...
30672	33.9893	89	55	-118.1808	8	1	...
30673	34.0443	117	0	-118.1528	7	1	...
30674	33.8283	117	216	-118.1528	2	1	...
30675	33.8283	-134	216	-118.4038	2	1	...
30676	34.0443	-141	0	-118.4108	0	1	...
30677	33.9013	-113	143	-118.3828	0	1	...
30678	34.0283	14	16	-118.2558	3	2	...
30679	34.0443	0	0	-118.2698	0	2	...

```
30681  33.7833    -18    261 -118.2878                0       2  ...
30683  33.8283      1    216 -118.2688               10       3  ...
30684  33.9553    -96     89 -118.3658               10       3  ...
30685  33.7943     81    250 -118.1888                7       3  ...
30687  33.9443     40    100 -118.2298                3       3  ...
30688  33.9833   -126     61 -118.3958                1       3  ...
30689  33.3653    -12    679 -118.2818                0       3  ...
30690  33.9443   -113    100 -118.3828               11       4  ...
30691  34.0443      0      0 -118.2698                7       4  ...
30692  33.9963      1     48 -118.2688                6       4  ...
30694  33.8783   -134    166 -118.4038                3       4  ...
30695  33.7773     31    267 -118.2388                2       4  ...
30696  33.9723      1     72 -118.2688                0       4  ...

              shot_zone_area        shot_zone_basic  shot_zone_range  \
1               Left Side(L)              Mid-Range        8-16 ft.
2       Left Side Center(LC)              Mid-Range       16-24 ft.
3      Right Side Center(RC)              Mid-Range       16-24 ft.
4                 Center(C)       Restricted Area  Less Than 8 ft.
5               Left Side(L)              Mid-Range        8-16 ft.
6                 Center(C)       Restricted Area  Less Than 8 ft.
8               Left Side(L)  In The Paint (Non-RA)        8-16 ft.
9                 Center(C)  In The Paint (Non-RA)        8-16 ft.
10      Left Side Center(LC)     Above the Break 3         24+ ft.
11     Right Side Center(RC)              Mid-Range       16-24 ft.
12              Left Side(L)  In The Paint (Non-RA)        8-16 ft.
13              Left Side(L)              Mid-Range        8-16 ft.
14                Center(C)  In The Paint (Non-RA)  Less Than 8 ft.
15                Center(C)              Mid-Range       16-24 ft.
17      Left Side Center(LC)     Above the Break 3         24+ ft.
18      Left Side Center(LC)              Mid-Range       16-24 ft.
20                Center(C)              Mid-Range        8-16 ft.
21     Right Side Center(RC)              Mid-Range       16-24 ft.
22                Center(C)  In The Paint (Non-RA)        8-16 ft.
23      Left Side Center(LC)              Mid-Range       16-24 ft.
24                Center(C)  In The Paint (Non-RA)  Less Than 8 ft.
25                Center(C)       Restricted Area  Less Than 8 ft.
26                Center(C)              Mid-Range       16-24 ft.
27      Left Side Center(LC)     Above the Break 3         24+ ft.
28     Right Side Center(RC)              Mid-Range       16-24 ft.
29                Center(C)  In The Paint (Non-RA)        8-16 ft.
30             Right Side(R)              Mid-Range        8-16 ft.
31             Right Side(R)              Mid-Range        8-16 ft.
38     Right Side Center(RC)              Mid-Range       16-24 ft.
39                Center(C)     Above the Break 3         24+ ft.
...                     ...                   ...              ...
30661             Center(C)       Restricted Area  Less Than 8 ft.
30662             Center(C)  In The Paint (Non-RA)  Less Than 8 ft.
```

```
30663          Right Side(R)              Mid-Range         8-16 ft.
30665              Center(C)        Restricted Area  Less Than 8 ft.
30666   Left Side Center(LC)              Mid-Range        16-24 ft.
30667              Center(C)      Above the Break 3          24+ ft.
30669          Right Side(R)              Mid-Range        16-24 ft.
30670              Center(C)  In The Paint (Non-RA)  Less Than 8 ft.
30671              Center(C)        Restricted Area  Less Than 8 ft.
30672          Right Side(R)              Mid-Range         8-16 ft.
30673          Right Side(R)              Mid-Range         8-16 ft.
30674  Right Side Center(RC)      Above the Break 3          24+ ft.
30675   Left Side Center(LC)      Above the Break 3          24+ ft.
30676           Left Side(L)              Mid-Range         8-16 ft.
30677   Left Side Center(LC)              Mid-Range        16-24 ft.
30678              Center(C)        Restricted Area  Less Than 8 ft.
30679              Center(C)        Restricted Area  Less Than 8 ft.
30681              Center(C)      Above the Break 3          24+ ft.
30683              Center(C)              Mid-Range        16-24 ft.
30684           Left Side(L)              Mid-Range         8-16 ft.
30685              Center(C)      Above the Break 3          24+ ft.
30687              Center(C)  In The Paint (Non-RA)         8-16 ft.
30688           Left Side(L)              Mid-Range         8-16 ft.
30689         Back Court(BC)               Backcourt  Back Court Shot
30690           Left Side(L)              Mid-Range         8-16 ft.
30691              Center(C)        Restricted Area  Less Than 8 ft.
30692              Center(C)  In The Paint (Non-RA)  Less Than 8 ft.
30694   Left Side Center(LC)              Mid-Range        16-24 ft.
30695              Center(C)      Above the Break 3          24+ ft.
30696              Center(C)  In The Paint (Non-RA)  Less Than 8 ft.

          team_id           team_name   game_date      matchup opponent  \
1      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
2      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
3      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
4      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
5      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
6      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
8      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
9      1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
10     1610612747  Los Angeles Lakers  2000-10-31    LAL @ POR      POR
11     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
12     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
13     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
14     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
15     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
17     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
18     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
20     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
21     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
```

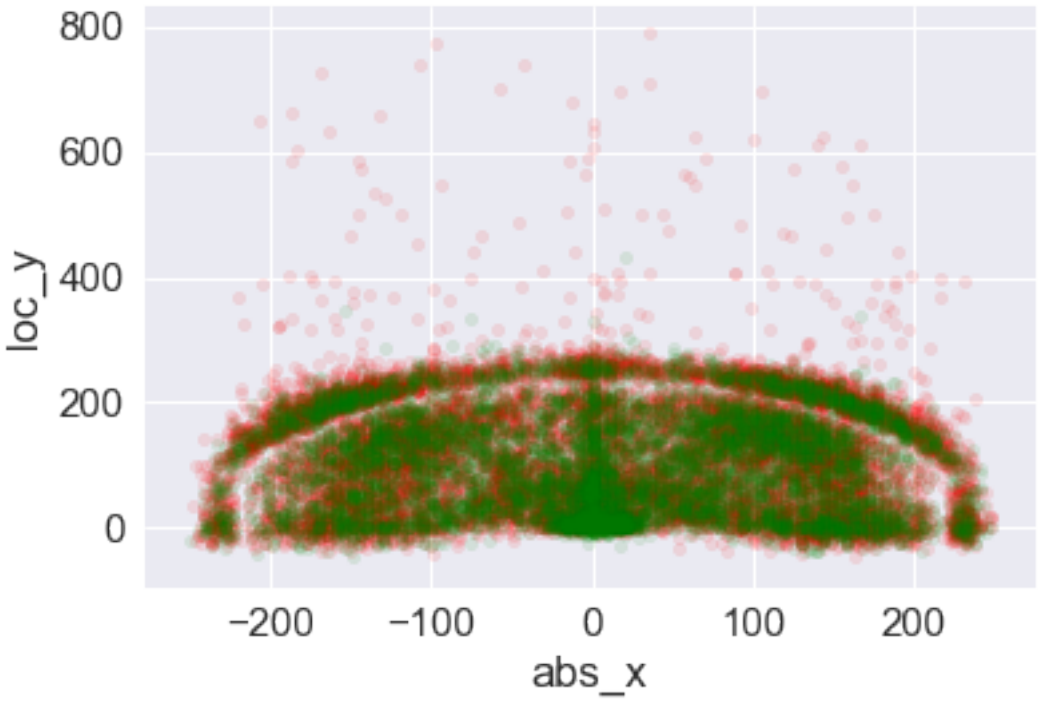
```
22     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
23     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
24     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
25     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
26     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
27     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
28     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
29     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
30     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
31     1610612747  Los Angeles Lakers  2000-11-01  LAL vs. UTA      UTA
38     1610612747  Los Angeles Lakers  2000-11-04    LAL @ VAN      VAN
39     1610612747  Los Angeles Lakers  2000-11-04    LAL @ VAN      VAN
...           ...                 ...         ...          ...      ...
30661  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30662  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30663  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30665  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30666  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30667  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30669  1610612747  Los Angeles Lakers  2000-06-16    LAL @ IND      IND
30670  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30671  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30672  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30673  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30674  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30675  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30676  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30677  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30678  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30679  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30681  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30683  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30684  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30685  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30687  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30688  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30689  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30690  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30691  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30692  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30694  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30695  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND
30696  1610612747  Los Angeles Lakers  2000-06-19  LAL vs. IND      IND

      shot_id  abs_x
1           2   -157
2           3   -101
3           4    138
```

4	5	0
5	6	-145
6	7	0
8	9	-65
9	10	-33
10	11	-94
11	12	121
12	13	-67
13	14	-94
14	15	-23
15	16	62
17	18	-117
18	19	-132
20	21	3
21	22	134
22	23	-16
23	24	-109
24	25	-46
25	26	0
26	27	-58
27	28	-183
28	29	85
29	30	3
30	31	121
31	32	127
38	39	91
39	40	-27
...	...	...
30661	30662	0
30662	30663	-8
30663	30664	106
30665	30666	-14
30666	30667	-81
30667	30668	40
30669	30670	171
30670	30671	-74
30671	30672	0
30672	30673	89
30673	30674	117
30674	30675	117
30675	30676	-134
30676	30677	-141
30677	30678	-113
30678	30679	14
30679	30680	0
30681	30682	-18
30683	30684	1
30684	30685	-96

```
30685   30686     81
30687   30688     40
30688   30689   -126
30689   30690    -12
30690   30691   -113
30691   30692      0
30692   30693      1
30694   30695   -134
30695   30696     31
30696   30697      1

[25697 rows x 26 columns]
```

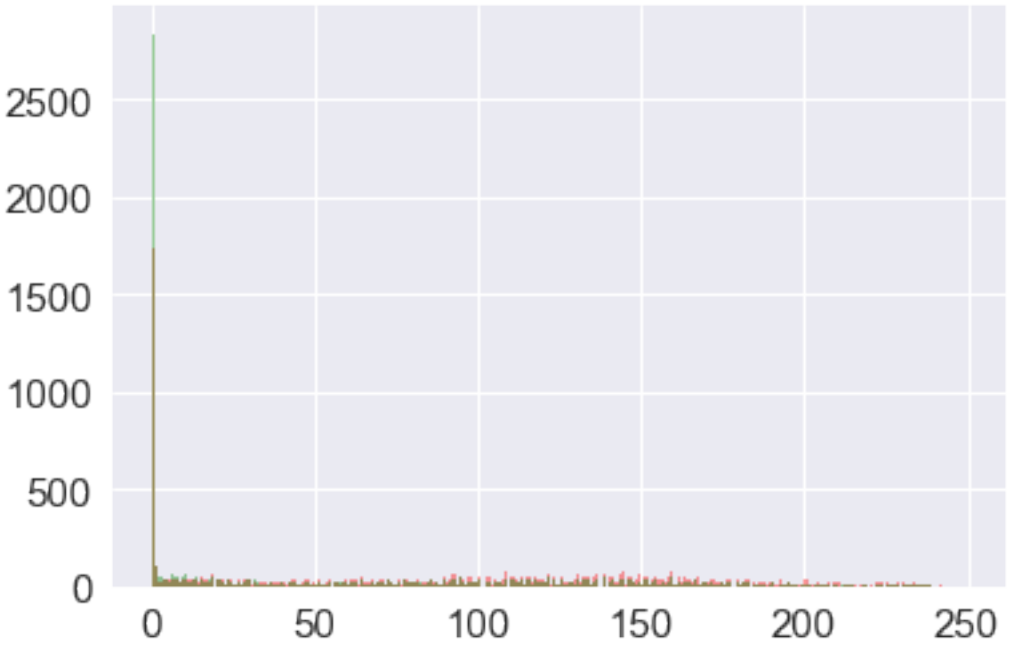
In [10]:
```
# This one is tricky. I don't see the beneift of creating a new
# column with the exact same values of the copying column.

# Let me copy what you've done in the lecture to see if something
# new pops up

fig, ax = plt.subplots()
kobe[kobe.shot_made_flag==0].plot(kind='scatter', x='abs_x', y='loc_y', color='red', al
kobe[kobe.shot_made_flag==1].plot(kind='scatter', x='abs_x', y='loc_y', color='green',
# scatterplot looks the same
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1102032e8>
```



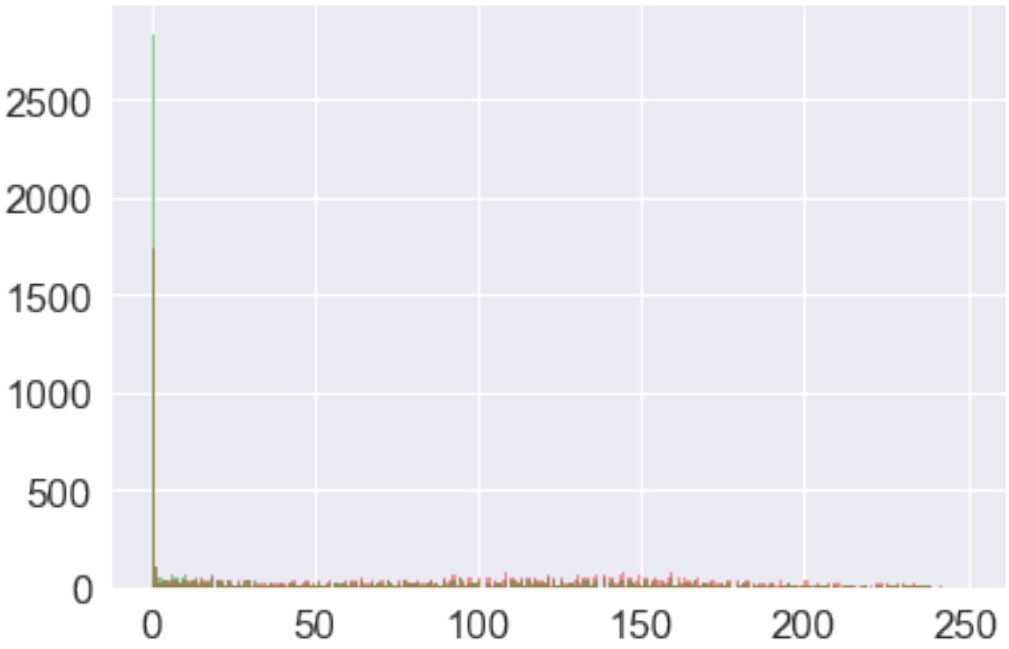
In [11]:
```
kobe[kobe.shot_made_flag==0].abs_x.hist(bins=np.arange(0,250,1), alpha=.4, color="red")
kobe[kobe.shot_made_flag==1].abs_x.hist(bins=np.arange(0,250,1), alpha=.4, color="green
#Kobe likes centered shots.
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1103dc080>
```



In [12]:
```
kobe[kobe.shot_made_flag==0].loc_x.hist(bins=np.arange(0,250,1), alpha=.4, color="red")
kobe[kobe.shot_made_flag==1].loc_x.hist(bins=np.arange(0,250,1), alpha=.4, color="green
# loc_x and abs_x look exactly the same for the obvious reason.
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x110a73e10>
```

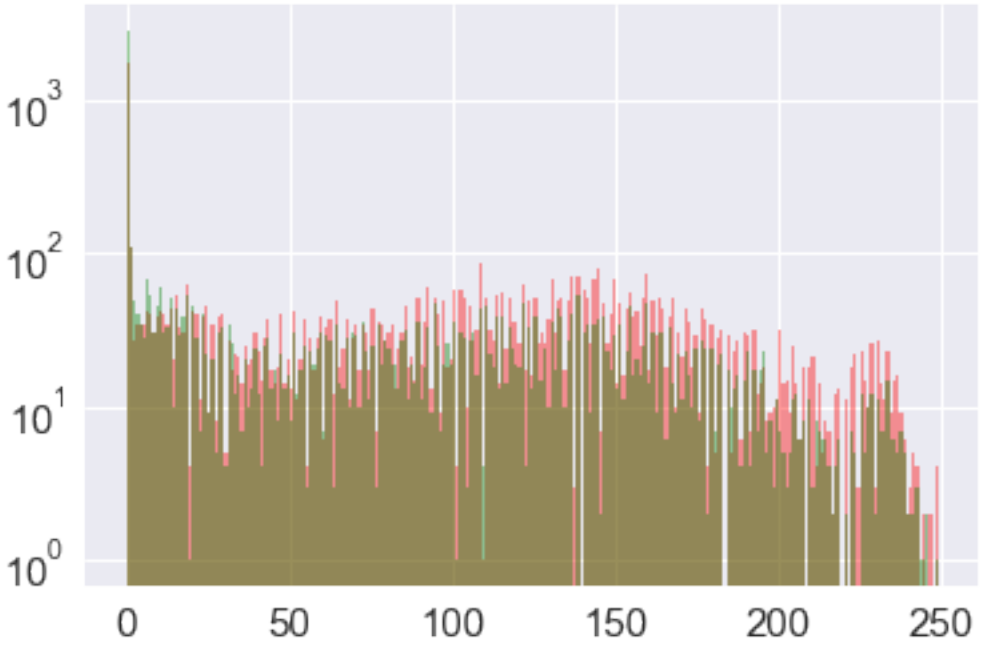


In [13]:
```
# I think log is more appropriate for analyzing success rates because
# raw numbers (number of shot counts) make visual inspection difficult
# for small quantities. But, this applies to both new and original column

kobe[kobe.shot_made_flag==0].abs_x.hist(bins=np.arange(0,250,1), alpha=.4, color="red",
kobe[kobe.shot_made_flag==1].abs_x.hist(bins=np.arange(0,250,1), alpha=.4, color="green
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a17d37c18>
```

In [14]:
```
kobe[kobe.shot_made_flag==0].loc_x.hist(bins=np.arange(0,250,1), alpha=.4, color="red",
kobe[kobe.shot_made_flag==1].loc_x.hist(bins=np.arange(0,250,1), alpha=.4, color="green
# Both the new column and original column looks the same thus far.
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1878acc0>
```

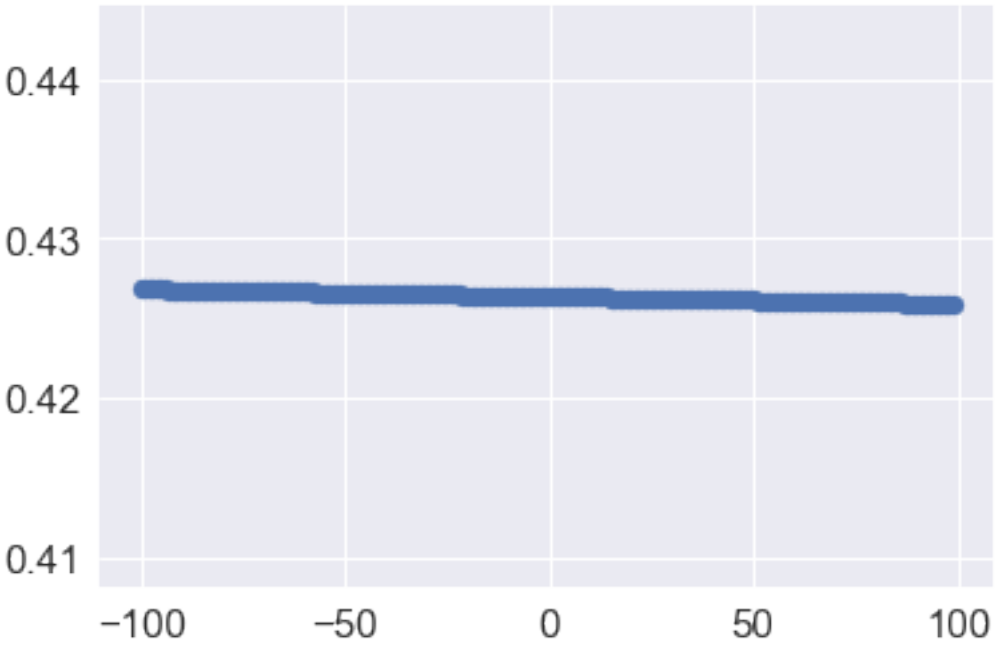


In [15]:
```
# fit a linear regression model and store the predictions
feature_cols = ['loc_x', 'minutes_remaining']
X = kobe[feature_cols]
y = kobe.shot_made_flag

model = Model()
model.fit(X, y)
kobe['pred'] = model.predict(X)


from sklearn.metrics import accuracy_score
accuracy_score(kobe.shot_made_flag, kobe.pred.round())
```

Out[15]:
```
0.5538389695295171
```

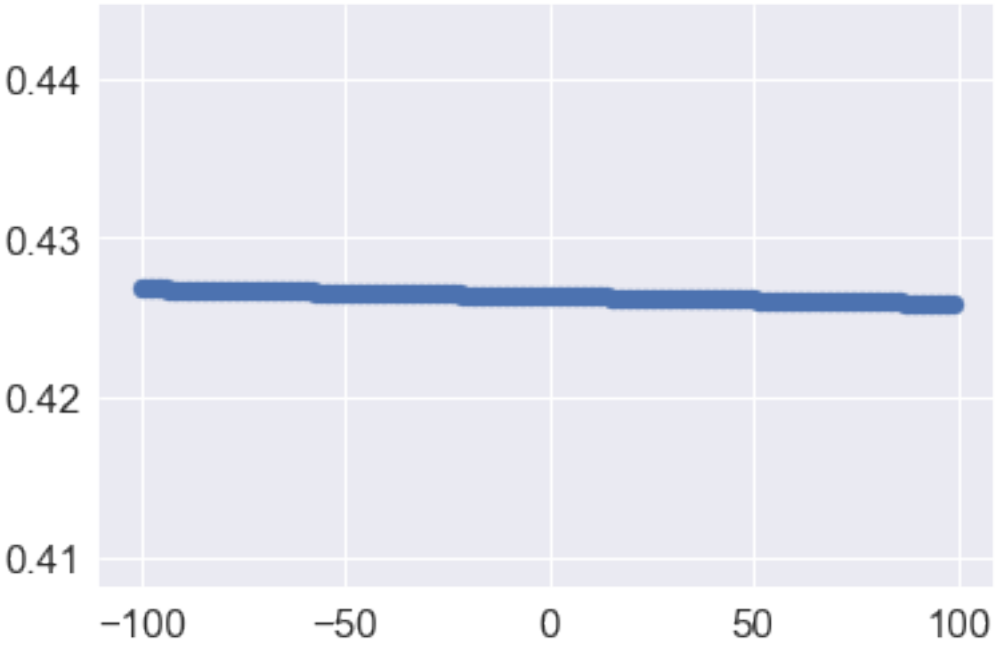
In [16]:
```
location = np.arange(-100, 100)
minutes = np.array([0]*200) # 0 minutes left in quarter
x_trial = np.column_stack((location, minutes))
model.predict_proba(x_trial)
plt.scatter(location, model.predict_proba(x_trial)[:,1])
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x1a18e98198>
```



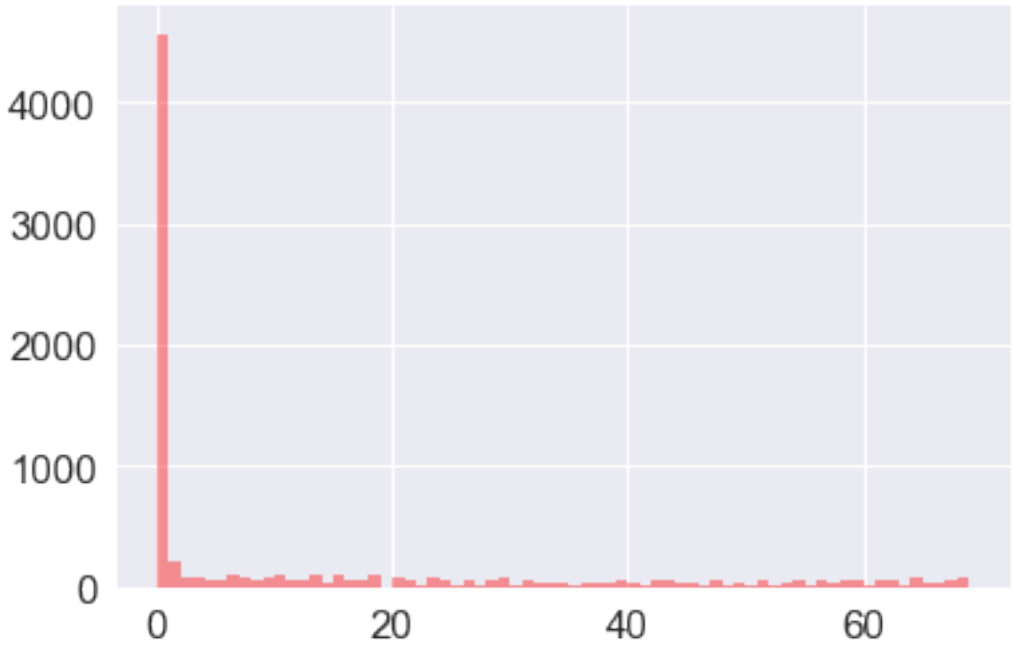
In [17]:
```
# fit a linear regression model and store the predictions
feature_cols = ['abs_x', 'minutes_remaining']
X = kobe[feature_cols]
y = kobe.shot_made_flag

model = Model()
model.fit(X, y)
kobe['pred'] = model.predict(X)

from sklearn.metrics import accuracy_score
accuracy_score(kobe.shot_made_flag, kobe.pred.round())

location = np.arange(-100, 100)
minutes = np.array([0]*200) # 0 minutes left in quarter
x_trial = np.column_stack((location, minutes))
model.predict_proba(x_trial)
plt.scatter(location, model.predict_proba(x_trial)[:,1])

# The abs_x column displays the same graph...
```

Out[17]: <matplotlib.collections.PathCollection at 0x109e76710>



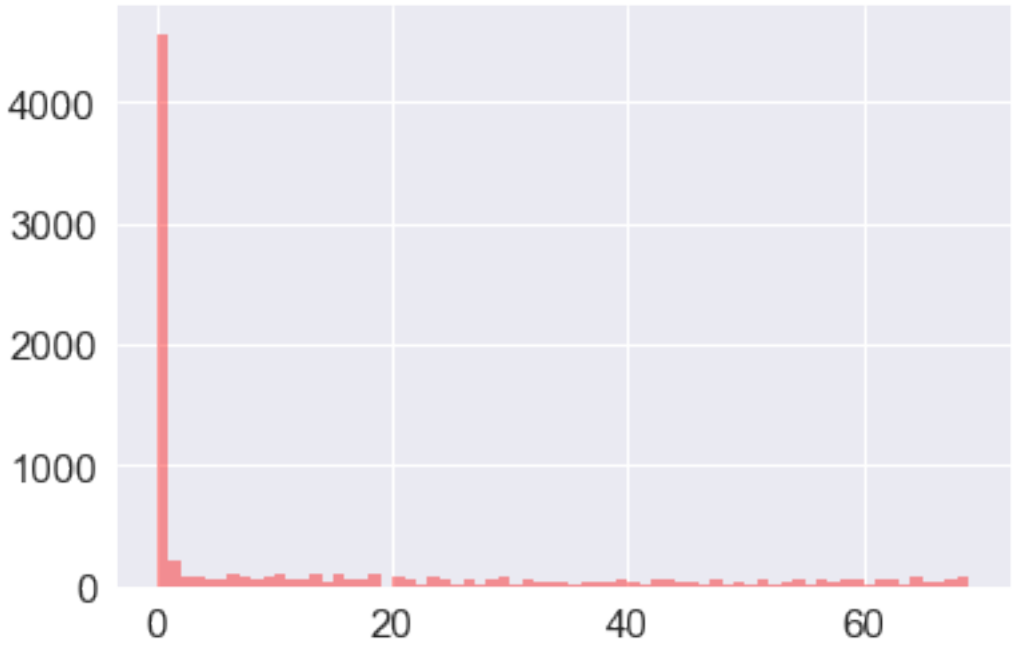
In [18]:
```
kobe[(kobe.pred==0)].loc_x.hist(bins=np.arange(0,70,1), alpha=.4, color="red")
kobe[(kobe.pred==1)].loc_x.hist(bins=np.arange(0,70,1), alpha=.4, color="green")
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x109e82a90>



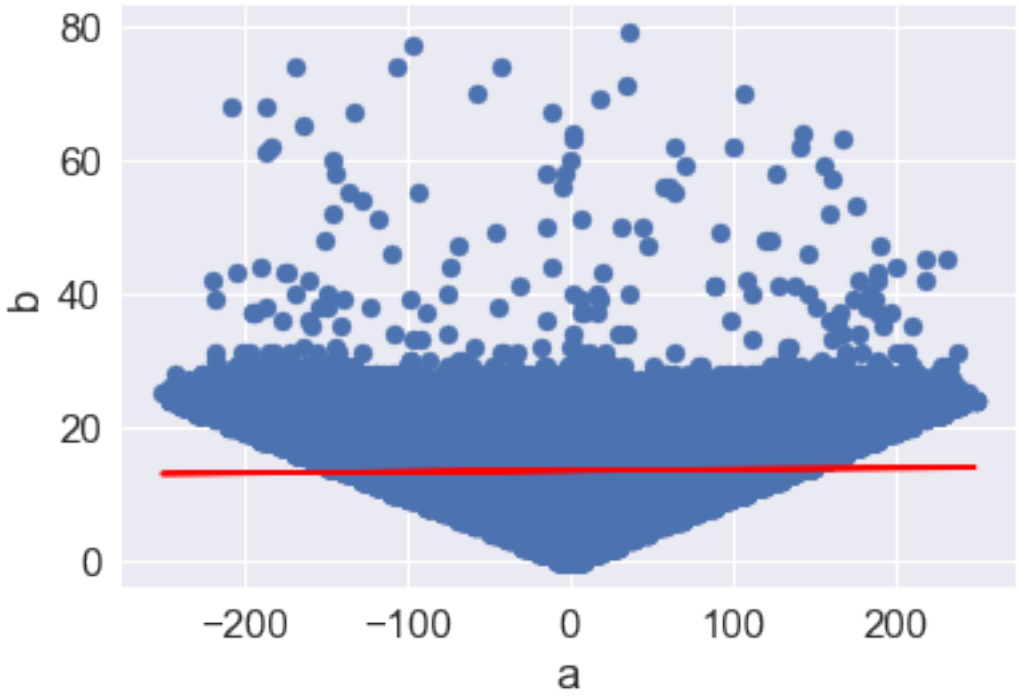
In [19]:
```
kobe[(kobe.pred==0)].abs_x.hist(bins=np.arange(0,70,1), alpha=.4, color="red")
kobe[(kobe.pred==1)].abs_x.hist(bins=np.arange(0,70,1), alpha=.4, color="green")

# I can't figure out why pred==1 is not showing
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x109b612b0>
```



In [20]:
```
# Yeah.. I'm having a hard time with this one.

# fit a linear regression model and store the predictions
example = pd.DataFrame({'a':kobe['abs_x'], 'b':kobe['shot_distance']})
feature_cols = ['a']
X = example[feature_cols]
y = example.b
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
example['pred'] = model.predict(X)
# scatter plot that includes the regression line
plt.scatter(example.a, example.b)
plt.plot(example.a, example.pred, color='red')
plt.xlabel('a')
plt.ylabel('b')

from sklearn.metrics import accuracy_score
accuracy_score(example.b, example.pred.astype(int))
```

Out[20]: 0.030587228081098962



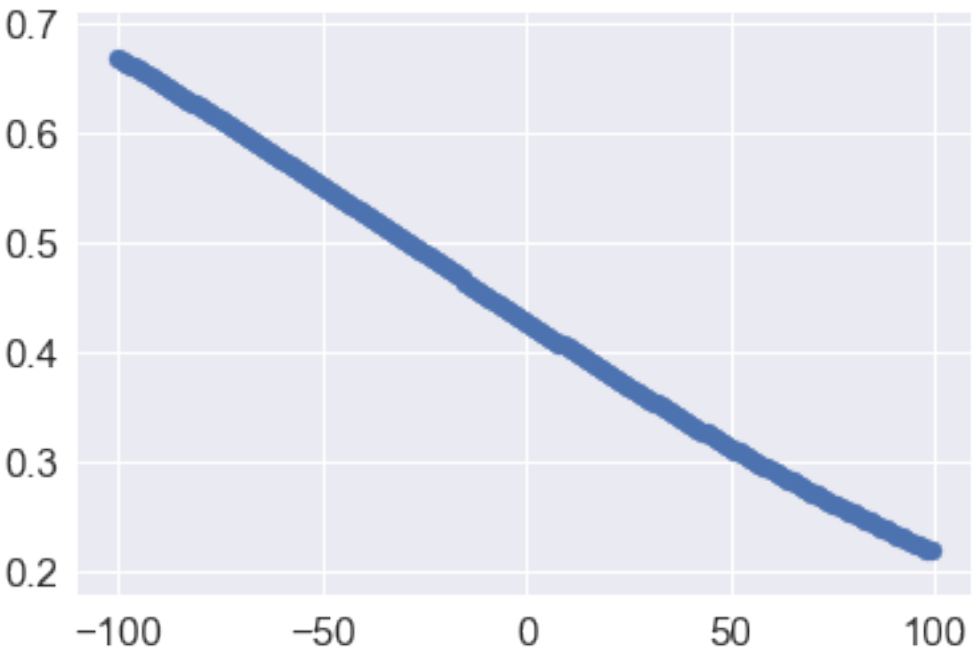
In [21]:
```
# I know you wanted the same values, but the graph outputs just don't make
# sense to me.

# ... So, let me try this instead

from sklearn.preprocessing import MinMaxScaler
kobe['abs_x'] = kobe.loc_x
kobe.abs_x = (kobe.abs_x - min(kobe.abs_x)) / ( max(kobe.abs_x) - min(kobe.abs_x))
kobe.abs_x
```

Out[21]:
```
1        0.186747
2        0.299197
3        0.779116
4        0.502008
5        0.210843
6        0.502008
8        0.371486
9        0.435743
10       0.313253
11       0.744980
12       0.367470
13       0.313253
14       0.455823
15       0.626506
17       0.267068
18       0.236948
20       0.508032
21       0.771084
22       0.469880
23       0.283133
24       0.409639
25       0.502008
26       0.385542
27       0.134538
28       0.672691
29       0.508032
30       0.744980
31       0.757028
38       0.684739
39       0.447791
           ...
30661    0.502008
30662    0.485944
30663    0.714859
30665    0.473896
30666    0.339357
30667    0.582329
30669    0.845382
```

```
30670    0.353414
30671    0.502008
30672    0.680723
30673    0.736948
30674    0.736948
30675    0.232932
30676    0.218876
30677    0.275100
30678    0.530120
30679    0.502008
30681    0.465863
30683    0.504016
30684    0.309237
30685    0.664659
30687    0.582329
30688    0.248996
30689    0.477912
30690    0.275100
30691    0.502008
30692    0.504016
30694    0.232932
30695    0.564257
30696    0.504016
Name: abs_x, Length: 25697, dtype: float64
```

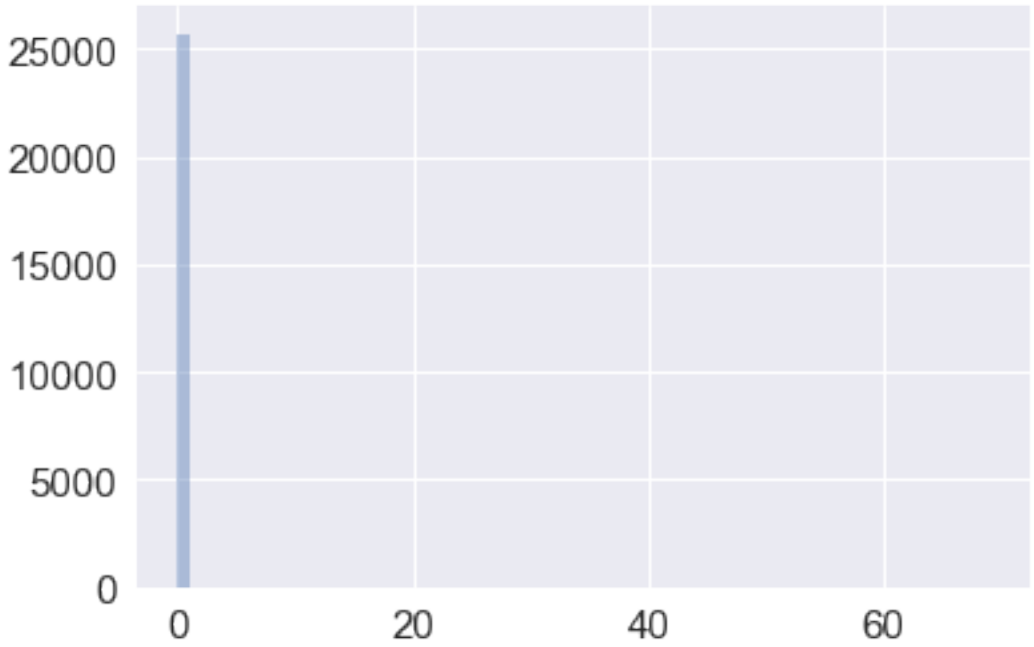
In [22]:
```
# Let's try this again

feature_cols = ['abs_x', 'minutes_remaining']
X = kobe[feature_cols]
y = kobe.shot_made_flag

model = Model()
model.fit(X, y)
kobe['pred'] = model.predict(X)

from sklearn.metrics import accuracy_score
accuracy_score(kobe.shot_made_flag, kobe.pred.round())

location = np.arange(-100, 100)
minutes = np.array([0]*200) # 0 minutes left in quarter
x_trial = np.column_stack((location, minutes))
model.predict_proba(x_trial)
plt.scatter(location, model.predict_proba(x_trial)[:,1])
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x110658390>
```



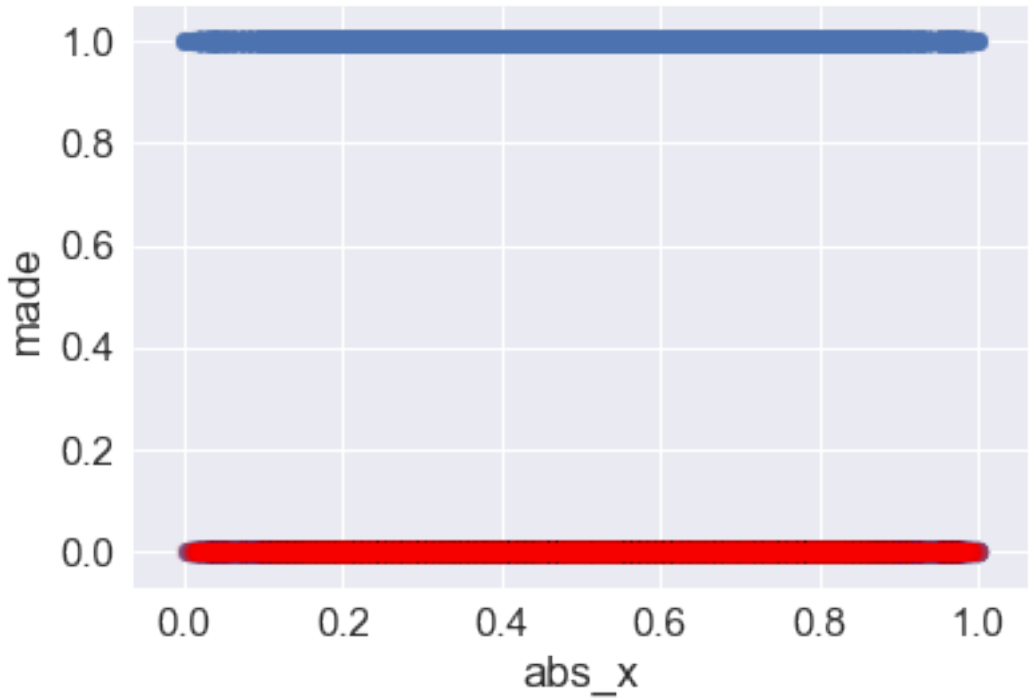
In [23]:
```
# It worked! Chances of scoring increase the closer to abs_x = 0

# If we normalize the data from 0 to 1, then that creates an
# ideal frame for the logistic regression, which is in the form of 0
# to 1.

kobe[(kobe.pred==0)].abs_x.hist(bins=np.arange(0,70,1), alpha=.4)
kobe[(kobe.pred==1)].abs_x.hist(bins=np.arange(0,70,1), alpha=.4)

# A histograph is not going to be helpful here because every value is now
# between 0 to 1. But, I suppose I could figure out a way to break down
# the decimla points.
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x110c80748>
```



In [24]:
```
# scatter plot that includes the regression line
plt.scatter(kobe.abs_x, kobe.shot_made_flag)
plt.scatter(kobe.abs_x, kobe.pred, color='red', alpha=.1)
plt.xlabel('abs_x')
plt.ylabel('made')

# This is showing that for abs_x there are shots made and shots missed
# for ranges in the x-axis.
```

Out[24]:
```
Text(0,0.5,'made')
```



1.0.3 2. Convert several (including ' ') string columns/features into numerical and attempt to use them in fitting a Logistic Regression model. Show histograms (similar to ones above) of made/missed of these new numerical features. Use these histograms to explain and justify why these features could improve the model

```
In [25]: kobe['ZR'] = kobe['shot_zone_range'].replace(
             {'Less Than 8 ft.':0,
              '8-16 ft.':1,
              '16-24 ft.':2,
              '24+ ft.': 3,
              'Back Court Shot': 4
             }
         )
         print(kobe.ZR)
```

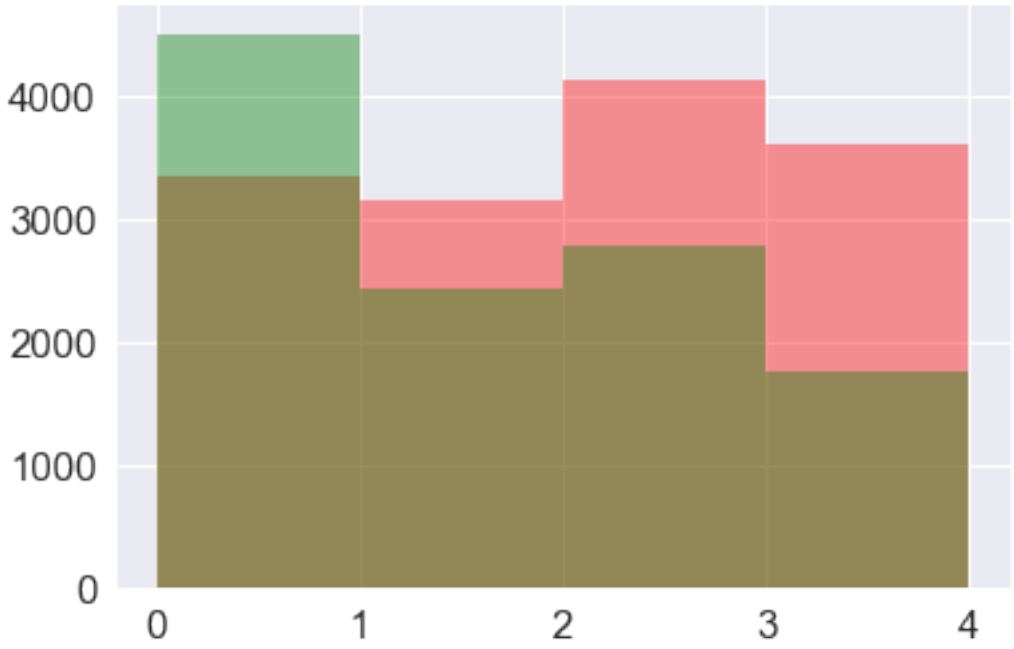
```
1         1
2         2
3         2
4         0
5         1
6         0
8         1
9         1
10        3
```

```
11      2
12      1
13      1
14      0
15      2
17      3
18      2
20      1
21      2
22      1
23      2
24      0
25      0
26      2
27      3
28      2
29      1
30      1
31      1
38      2
39      3
        ..
30661   0
30662   0
30663   1
30665   0
30666   2
30667   3
30669   2
30670   0
30671   0
30672   1
30673   1
30674   3
30675   3
30676   1
30677   2
30678   0
30679   0
30681   3
30683   2
30684   1
30685   3
30687   1
30688   1
30689   4
30690   1
30691   0
```

```
30692    0
30694    2
30695    3
30696    0
Name: ZR, Length: 25697, dtype: int64
```

In [26]:
```
kobe[kobe.shot_made_flag==0].ZR.hist(bins=np.arange(0,5,1), alpha=.4, color="red")
kobe[kobe.shot_made_flag==1].ZR.hist(bins=np.arange(0,5,1), alpha=.4, color="green")

# I can probably create histograms without converting strings to numbers
# So, I would not say this is particularly useful for this purpose
```

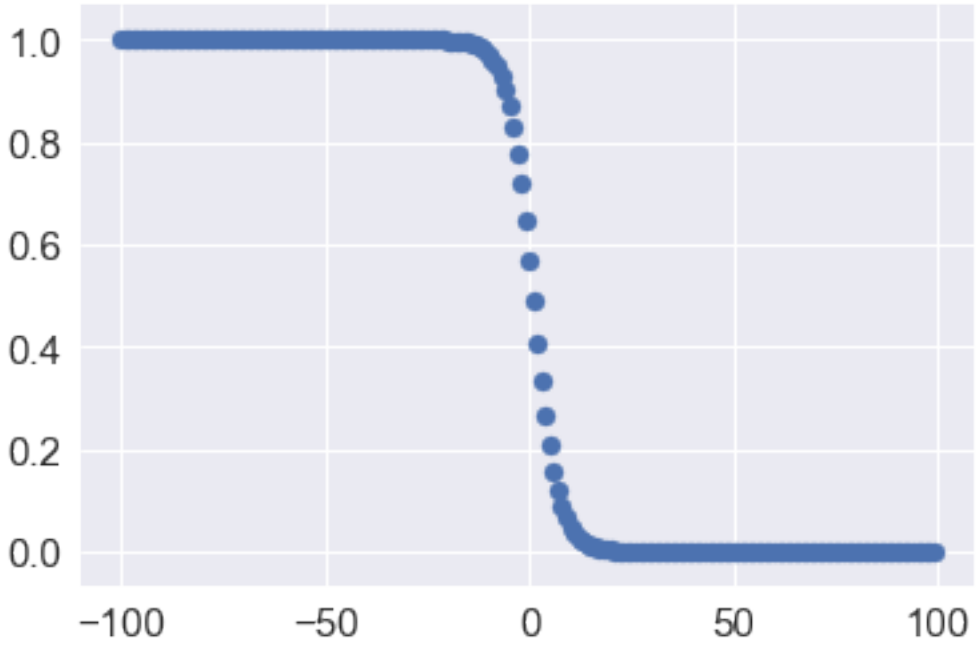
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a19de46d8>



In [27]:
```
# fit a linear regression model and store the predictions
feature_cols = ['ZR', 'minutes_remaining']
X = kobe[feature_cols]
y = kobe.shot_made_flag

model = Model()
model.fit(X, y)
kobe['pred'] = model.predict(X)

from sklearn.metrics import accuracy_score
accuracy_score(kobe.shot_made_flag, kobe.pred.round())
```

```
distances = np.arange(-100, 100)
minutes = np.array([10]*200) # 10 minutes left in quarter
x_trial = np.column_stack((distances, minutes))
model.predict_proba(x_trial)
plt.scatter(distances, model.predict_proba(x_trial)[:,1])
```

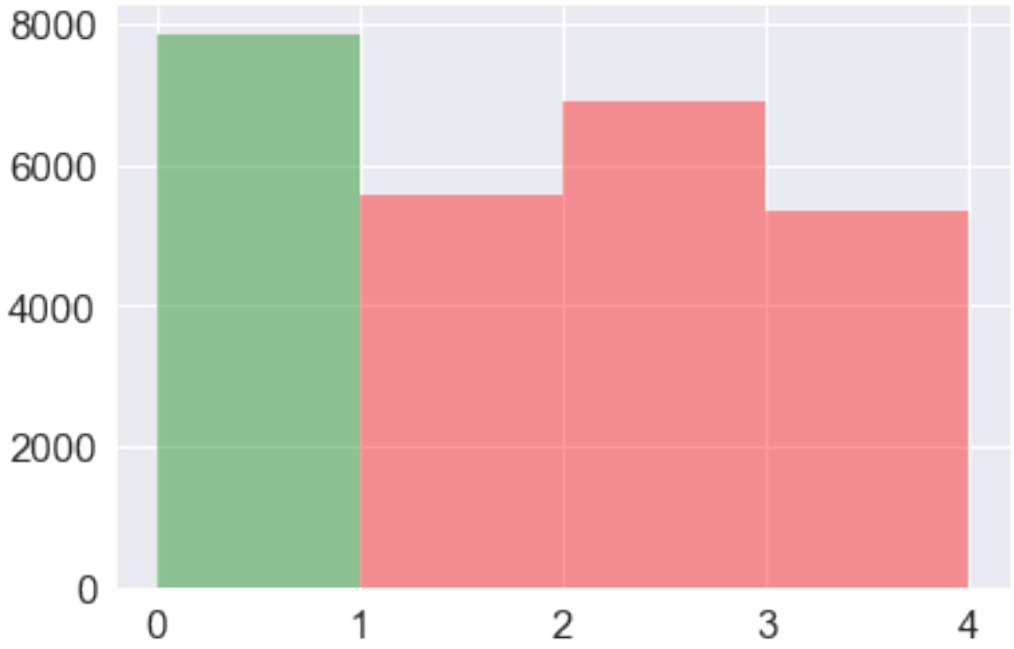
Out[27]: <matplotlib.collections.PathCollection at 0x1a18711978>



In [28]:
```
# Interesting. Replacing the strings to numeric values would allow a
# logarithmic expression. The trick would be figuring out the true
# value for the tipping point near 0
```

In [29]:
```
kobe[(kobe.pred==0)].ZR.hist(bins=np.arange(0,5,1), alpha=.4, color = "red")
kobe[(kobe.pred==1)].ZR.hist(bins=np.arange(0,5,1), alpha=.4, color = "green")
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1a18dc4780>



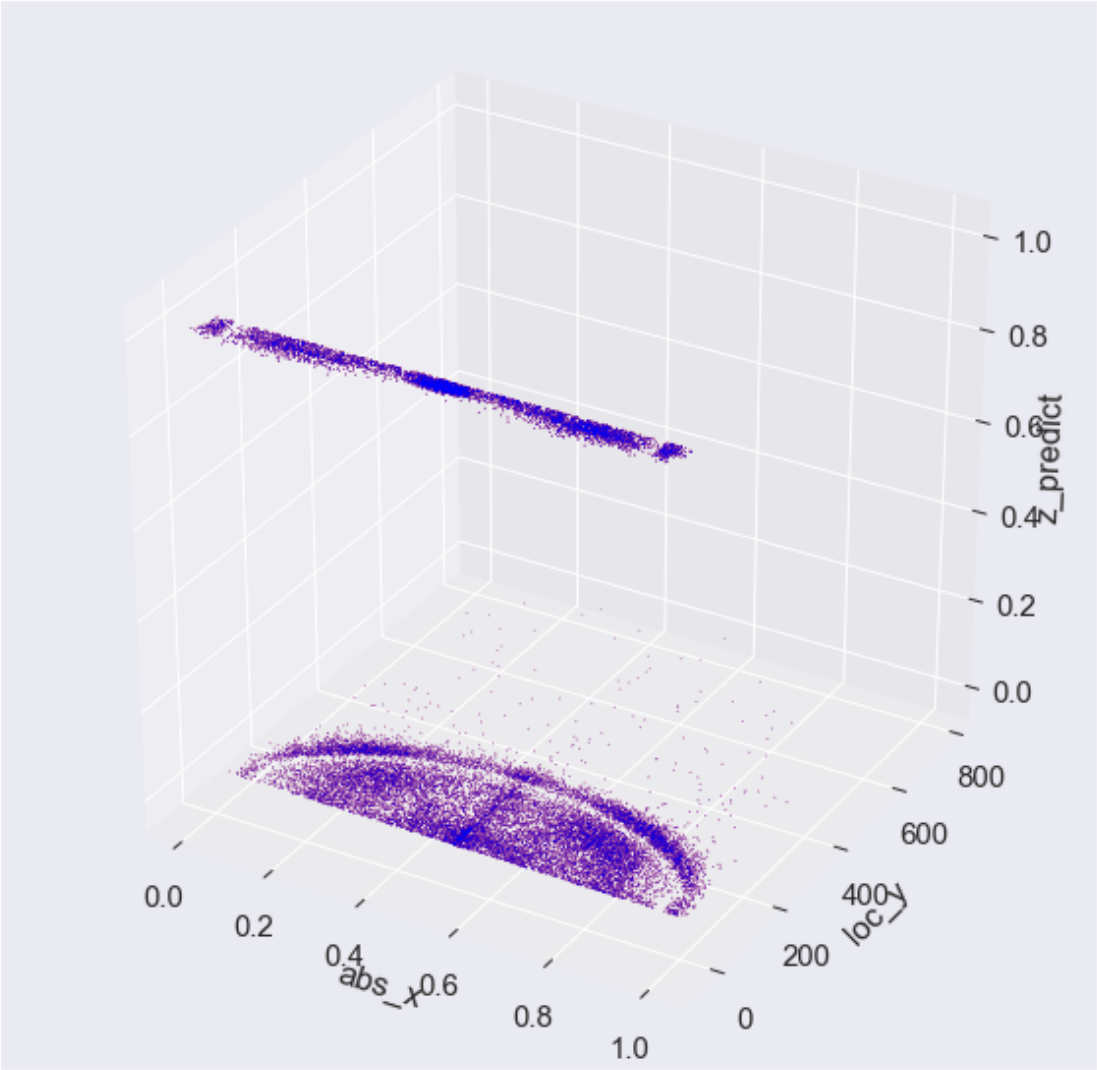
In [30]:
```
# This histogram is meaningful because these are predictive
# values. Looking at the real values, however, they are not that
# accurate.
```

1.0.4 3. Show a 3 dimensional surface plot [https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html#surface plots] of probabilities from a trained Logistic Regression model using only `abs_x` and `loc_y`. The probabilities arise from a distributed grid of `x` values and `y` values as input to the `predict_proba()` function.

In [38]:
```
# fit a linear regression model and store the predictions
feature_cols = ['abs_x', 'loc_y']
X1 = kobe[feature_cols]
y1 = kobe.shot_made_flag

model = Model()
model.fit(X1, y1)
kobe['predict'] = model.predict(X1)

from sklearn.metrics import accuracy_score
accuracy_score(kobe.shot_made_flag, kobe.pred.round())

abs_X1 = np.arange(-100, 100)
loc_y1 = np.arange(-100, 100)
x_trial = np.column_stack((abs_X1, loc_y1))
```

In [35]:
```
x= kobe.abs_x
y= kobe.loc_y
```

```
z= kobe.predict

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np


def randrange(n, vmin, vmax):
    '''
    Helper function to make an array of random numbers having shape (n, )
    with each number distributed Uniform(vmin, vmax).
    '''
    return (vmax - vmin)*np.random.rand(n) + vmin

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')


# For each set of style and range settings, plot n random points in the box
# defined by x in [23, 32], y in [0, 100], z in [zlow, zhigh].
for c, m, zlow, zhigh in [('red', 'o', -50, -25), ('blue', '^', -3, -5)]:
    ax.scatter(x, y, z, c=c, marker=m, s=.2)

ax.set_xlabel('abs_x')
ax.set_ylabel('loc_y')
ax.set_zlabel('z_predict')

plt.show()
```



In [33]: *# This one I don't think I understood the question...*