final

December 11, 2022

[ ]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```python
df=pd.read_excel('PassEventsForwardFootball.xlsx')
```

[ ]:
```python
# -------------------------------------------------------- Observe
# there will be a lot of columns that are not informative( just have one unique␣
 ↪value), then check how many here
def get_columns_with_one_unique_value(df):
    col_counts = df.nunique()
    cols_with_one_unique_value = col_counts[col_counts == 1]
    return list(cols_with_one_unique_value.index)
print("get_columns_with_one_unique_value:")
get_columns_with_one_unique_value(df)

# ********************************************************

# -------------------------------------------------------- Observe
def print_unique_value(df):
    for col in df:
        print("column name:",col)
        print(df[col].unique())
        print("---")
print("print_unique_value:")
print_unique_value(df)

# from the result, we could see that columns
# Type, x_pitchsize, y_pitchsize each have just one unique value->done
# Club has just one unique vale ['Team Forward Football']-> done


# isForward and isSucceeded are either True or False, then convert them to␣
 ↪numerical value 0 and 1 in order to better process-> done
# Team is either ['Team Forward Football_1' 'Team Forward Football_2'], then␣
 ↪convert them to numerical value 0 and 1 -> done
# Pass type ['Forward pass' 'Lateral pass' 'Backward pass']-> 0,1,2->done
```

```
# Pressure level -> ['Full Pressure' 'No Pressure' 'Limited Pressure']->done
# column name: Zone ['Attack' 'Defence' 'Mid field']->0,1,2
# column name: Playing direction_first half ['left' 'right']->done
# column name: Playing direction_second half ['left' 'right']->done

# matchDuration is written in minutes and also just have two values-> don't
↪need to convert

# *******************************************************
```

```
get_columns_with_one_unique_value:
print_unique_value:
column name: Type
['Pass']
---
column name: TimeStamp
['2022-05-05T11:39:15.000000000' '2022-05-05T11:39:59.000000000'
 '2022-05-05T11:41:47.000000000' … '2022-05-12T12:52:37.000000000'
 '2022-05-12T12:53:57.000000000' '2022-05-12T12:55:38.000000000']
---
column name: posX_passer
[ 32  16  92  72  74  24  90  17  82  79  45  43  52  69  30  33  22  20
  46  78  49  28  65  41  14   6  38  37  66  23  84  73  56  70  71  12
  36  67  75  63  10  68  31  55  59  64  61  50   4  88  53  35   2  47
   0  26   8  44  42  77  62  81  48  21  58  29  85  19  34  94  86  76
  39   1  40   3  11  27  18  25   9  98  87  13  96  15  91  95  97 100
  80  51  99  57   5   7  89  93  83  54 101  60  -1]
---
column name: posY_passer
[58 23 56 55 53 48 33 59 44 50 63 61 62 42 40 37 19 31 34 25 12 11 14 18
  0  5  7 13  3 26  9  4  1 35 36 64 52 22 15 10 51 20  6  2 46 32 38 27
 29 47 41 49 39 21 17  8 16 30 28 60 43 24 45 57 54]
---
column name: received_PosX
[ 26  20  68  76  83  67  10  78  86  46  42  54  66  37  25  29  18  45
  93  57  39  15  69  34  19  30  35  63  70  80  73  38  53  56  91  11
   9  62  64  43  59   0  74  40  60  75   1   4  13  17  88  52  14  44
  -1  49  27  48  21  79   2  28  72  24  33  55  85  47  12   7   3  31
  71  97  61  92  32  41  65  22  36  16  89  84  58   6  50  82  98 101
  96  99  77  51  81  94 100  23  90  95   5  87   8 102]
---
column name: received_PosY
[55 29 52 62 50 48 19 58 54 49 64 42 35 47 57 20 32 56 59 18 34 15  9 26
  7 11 23 -1  6 31  3  1 40 41 13 46 39 16 10 51 53 37 17 36  0 22 12 38
 21 30 24 27 45  5 28 33 14 43 25  8 44  4  2 63 65 66 60 61]
---
column name: isForward
```

```
[ True False]
---
column name: isSucceeded
[ True False]
---
column name: receiverId
[95583. 95601. 95597. 95587. 95579.    nan 95594. 95595. 95988. 95602.
 95591. 95592. 95588. 95986. 95586. 95593. 95581. 95580. 95585. 95582.
 95600. 95584. 95589. 95987. 95603. 95617. 95618. 95624.]
---
column name: Player_id
[95582 95585 95584 95597 95601 95600 95583 95987 95603 95586 95588 95580
 95594 95589 95988 95986 95592 95581 95591 95593 95587 95602 95579 95595
 95624 95618 95617]
---
column name: Team
['Team Forward Football_1' 'Team Forward Football_2']
---
column name: startTime
['2022-05-05T11:36:13.000000000' '2022-05-12T12:06:13.000000000']
---
column name: matchDuration
[66.22186667 61.15793333]
---
column name: Club
['Team Forward Football']
---
column name: Time block
[1 2 3 4 5 6]
---
column name: Zone
['Attack' 'Defence' 'Mid field']
---
column name: Area Football Pitch
[15 17  3  6  2 12  9 14 11  5  7 10 13  1  4 16  8 18]
---
column name: Angle Passe
[0.46364761 2.15879893 0.16514868 2.08994244 2.8198421  0.
 0.54678884 1.34156439 1.29249667 2.62244654 1.63736449 1.37340077
 0.78539816 3.14159265 0.70862627 1.05165021 1.57079633 1.50422816
 2.21429744 3.01723766 1.26491746 0.22679885 2.89661399 0.14189705
 2.35619449 0.71883    0.98279372 1.21202566 0.64350111 1.152572
 0.21866895 0.90975316 2.2794226  1.19028995 2.03444394 1.735945
 0.38050638 0.62879629 1.9513027  2.67794504 2.60117315 1.24904577
 0.09065989 1.10714872 1.28700222 0.67474094 2.62714134 0.32175055
 2.11121583 2.14213381 1.13838855 1.44644133 2.73670087 1.68145355
 1.47112767 1.46013911 2.8753406  1.48765509 1.46771472 1.78946527
 2.2318395  1.77829255 1.91956733 1.09345094 2.76108628 0.96525166
```

1.39094283	1.03037683	1.17600521	0.57133748	0.5880026	1.15628945
2.2655346	2.78282198	0.24497866	1.1479424	1.71269338	1.32581766
0.43662716	0.27829966	0.47439988	0.348771	0.35877067	0.11065722
0.44441921	1.36677835	0.29849893	0.56331626	2.44685438	2.12245131
1.89254688	1.96558745	2.22649195	0.55165498	2.49809154	1.49096634
0.88506682	0.29145679	0.19739556	1.30454428	0.92729522	0.5070985
0.37433362	0.5404195	0.27829966	0.51914611	0.98279372	1.16590454
0.95054684	0.64350111	0.72664234	0.70862627	0.21866895	0.86217005
0.40489179	1.8766752	1.66145621	1.04600056	1.35970299	0.51914611
1.8736812	1.31019394	2.55359005	0.17324567	0.60005021	1.17227388
0.96525166	1.18018928	2.96173915	0.12435499	0.5485494	0.89605538
3.07028519	0.29849893	0.87605805	1.23150371	0.19739556	1.83704838
0.07259945	1.81577499	0.69473828	1.32581766	1.69515132	2.94419709
0.14189705	0.348771	1.40101805	1.42889927	2.46685171	0.28605144
0.53172407	0.16514868	0.24497866	0.68572951	0.55859932	2.86329299
0.05875582	0.03844259	0.86217005	0.90975316	0.99003997	2.25652584
1.43526861	2.01063891	1.31347261	2.07789483	1.84909599	0.93804749
3.041924	2.96692045	1.16066899	0.69473828	0.43662716	0.2220819
1.152572	1.64210379	2.72336832	2.24553727	0.11065722	1.37961187
2.52134317	0.02856366	3.00904112	0.52479577	0.29544084	0.85196633
0.32175055	0.7328151	2.94419709	0.5070985	0.5485494	0.27416745
0.68052122	1.52734543	1.97568811	0.14888995	2.98499078	0.14707836
0.49394137	1.47112767	0.56192156	1.35673564	0.89605538	2.85013586
0.03224688	0.41822433	3.08609415	1.62787711	1.929567	1.4204249
0.5880026	0.66104317	0.10487694	1.06369782	0.21109333	2.08318579
3.113029	0.12970254	1.29984948	2.0032041	0.55165498	2.98894333
0.09966865	1.1284221	1.03708814	1.40564765	1.26987609	1.14416883
0.35563588	1.76819189	0.66596924	1.75390714	0.607802	0.49394137
2.92292371	0.62024949	1.53081764	2.99270271	2.9996956	1.93797016
2.43933572	2.53086669	0.67474094	1.31561394	1.65393756	2.45586314
1.0863184	0.9964915	1.64756822	1.07437357	0.48447793	0.84089667
0.39060704	0.81986726	1.98902066	1.13095374	0.36717383	1.99742382
0.15264933	1.48013644	1.72344566	0.09966865	1.67046498	2.41495031
0.92729522	0.90675016	1.08990905	0.82884906	0.70361378	0.52807445
0.0434509	0.56192156	0.81569192	0.61253756	0.62548504	1.11745763
0.43069251	1.78188966	1.18247761	0.72989966	0.08314123	0.12435499
1.19990504	1.35212738	2.08994244	1.71269338	0.06656816	0.83798123
0.42285393	0.56672922	1.63321514	0.0344691	0.84415399	3.03093543
1.3633001	0.66596924	0.4825133	0.61466295	0.0344691	0.82537685
0.27094685	0.05258306	1.27933953	1.86929526	2.80491783	2.2706892
0.49642275	2.64765128	1.01914134	0.07130746	1.91008894	2.84309372
2.30361143	0.9151007	1.39612413	0.61072596	0.62024949	0.73997488
0.05549851	2.38648825	0.1746722	1.01219701	1.79759517	2.64224593
0.4825133	1.38544838	0.13255153	1.2722974	1.22777239	0.22679885
1.39860551	1.47432255	1.75614428	1.63736449	1.48405799	2.63449415
1.28274088	2.06721908	1.06663037	0.17219081	0.76721835	0.39479112
0.53439548	1.26791146	0.5404195	1.120135	1.14103405	2.32590073
1.080839	1.76198079	0.92180077	2.19104581	1.52321322	0.88708702

```
 2.97644398 0.11379201 1.28474489 0.53172407 2.28962633 0.90250691
 2.43296638 0.7140907  2.25972072 1.45368758 0.41241044 2.61351821
 2.50656592 1.02224692 0.87605805 0.23554498 2.18545928 2.96692045
 1.50837752 1.44109379 2.71496516 0.82241828 1.74750518 1.46591939
 0.05875582 0.30805278 0.2234766 ]
---
column name: Pass type
['Forward pass' 'Lateral pass' 'Backward pass']
---
column name: Pass length
[ 6.70820393  7.21110255 24.33105012  8.06225775  9.48683298  4.
 26.92582404 30.8058436  14.56021978 15.03329638  5.09901951  3.60555128
  4.24264069  7.          9.21954446  3.         25.         19.92485885
         nan  4.12310563  7.07106781  5.65685425 13.         10.63014581
  8.54400375  5.          9.8488578  18.43908891 22.8035085   5.38516481
  8.48528137  6.08276253 13.60147051  4.47213595 11.66190379 18.97366596
 22.09072203 11.18033989  6.40312424 26.41968963  3.16227766 17.49285568
  8.         16.64331698 14.31782106  1.          7.61577311  9.05538514
 20.09975124 11.40175425 12.04159458 29.15475947  2.82842712 19.41648784
 11.70469991 32.64965543 10.77032961 15.8113883  33.54101966 27.31300057
  7.81024968 16.4924225  12.72792206 21.9317122  14.14213562 18.
 10.         16.15549442 23.32380758 28.28427125 12.36931688 16.55294536
  7.28010989 41.59326869  8.24621125 17.08800749 23.2594067  29.61418579
 27.20294102 22.47220505 15.26433752  8.94427191  2.23606798 12.64911064
 12.16552506  6.         16.40121947 30.52867504 25.07987241 28.42534081
 10.44030651 10.19803903  2.         20.         10.29563014 30.08321791
 24.18677324  8.60232527  0.         24.08318916  6.32455532 11.04536102
 21.9544984  28.63564213 16.76305461 15.5241747  40.60788101 23.02172887
  9.89949494 20.61552813 15.         18.38477631 12.80624847 32.24903099
 21.09502311 19.20937271 28.0713377  15.62049935 18.02775638  9.
 55.14526272 31.2409987  18.24828759 35.35533906 35.51056181 12.
 17.72004515 19.72308292 42.63801121  9.43398113  5.83095189 16.1245155
 17.02938637 26.01922366 38.27531842 14.2126704  22.20360331 18.78829423
 19.6468827  18.60107524 10.04987562 17.2626765  31.78049716 22.36067977
 28.8444102  14.03566885 31.57530681 30.         35.0142828  15.13274595
 24.04163056 13.45362405 33.24154028 27.01851217 20.22374842 19.23538406
 27.29468813 29.52964612 21.63330765 31.90611227  1.41421356 23.53720459
 22.         31.01612484 19.         35.05709629 33.37663854 34.20526275
 19.10497317 18.35755975 23.19482701 18.68154169 13.15294644 21.02379604
 25.55386468 30.3644529  12.08304597 25.49509757 37.33630941 10.81665383
 17.80449381 27.45906044 28.01785145 14.76482306 34.78505426 25.01999201
 13.92838828 12.20655562 26.87005769 23.76972865 21.47091055 20.24845673
 13.03840481 15.23154621 13.34166406 18.11077028 20.51828453 38.41874542
 32.44996148 15.29705854 40.         19.79898987 29.20616373 25.94224354
 15.55634919 25.61249695 36.76955262 27.65863337 16.2788206  43.27817002
 14.         32.52691193 27.78488798 23.34523506 45.22167622 43.38202393
 40.71854614 38.62641583 37.73592453 16.03121954 34.05877273 29.01723626
 23.70653918 29.12043956 20.1246118  14.4222051  20.80865205 17.69180601
```

```
 37.36308338 19.02629759 21.1896201  24.8394847  11.         18.86796226
 31.30495168 20.39607805 31.144823   28.3019434  15.65247584 12.52996409
 30.2654919  16.         29.73213749 32.984845   23.08679276 28.16025568
 33.12099032 38.89730068 22.627417   26.         48.37354649 56.93856338
 33.52610923 34.43835072 26.40075756 17.         36.40054945 42.04759208
 34.82814953 35.22782991 21.26029163 24.20743687 19.84943324 22.02271555
 17.11724277 30.06659276 17.4642492  30.14962686 13.89244399 13.41640786
 23.60084744 21.21320344 24.59674775 24.16609195 25.70992026 28.44292531
 22.56102835]
---
column name: Playing direction_first half
['left' 'right']
---
column name: Playing direction_second half
['left' 'right']
---
column name: x_pitchsize
[99]
---
column name: y_pitchsize
[63]
---
column name: Pressure level
['Full Pressure' 'No Pressure' 'Limited Pressure']
---
column name: Distance to first opponent
[ 1.61245155  4.70744092  2.05912603  3.86264158  3.4         1.64924225
  4.72016949  4.61735855 11.06345335  2.69072481  5.3141321   4.01995025
  1.07703296  1.81107703  2.40831892  2.03960781  3.84707681  4.77074418
  5.81377674  3.67695526  4.20475921  8.63712915  6.55133574  2.69072481
  1.28062485  1.6         3.13049517  2.6         4.38634244  1.
 13.28006024  4.11825206  3.62215406  2.0880613   4.38634244  5.20384473
  5.49181209  2.86356421  3.05941171  1.16619038  5.8         2.28035085
  3.60555128  3.10483494  1.72046505  3.13049517  4.47213595  1.13137085
  1.4         2.2627417   2.15406592  3.2249031   2.52982213  4.38634244
  2.          3.10483494  3.92937654  1.16619038  2.50599282  3.64965752
  3.          3.04630924  1.88679623  2.03960781  0.72111026  4.12310563
  2.95296461  1.21655251  1.44222051  3.77359245  2.97321375  2.2627417
  5.65685425  6.22253967 13.4714513   0.8         1.07703296  3.
  3.98497177  2.23606798  1.28062485  4.93963561  1.4         2.60768096
  1.          4.66904701  6.21288983  3.68781778  3.49857114  7.72010363
  2.          2.68328157  4.70744092  1.97989899  3.4525353   3.2984845
  3.80525952  4.16173041  2.34093998  3.82099463  4.21900462  2.15406592
  2.16333077  3.23109888  8.35703297  3.25576412  3.82099463  2.47386338
  2.60768096  4.44072066  3.02654919  2.47386338  0.89442719  5.69209979
  3.2984845   5.16139516  2.97321375  2.0880613   4.40454311  2.40831892
  4.49444101  5.01198563  4.20475921  4.93963561  1.26491106  6.26418391
  3.44093011  3.64965752  2.47386338  1.61245155  2.12602916  2.68328157
```

4.68187996	2.56124969	5.26117858	3.2249031	9.93579388	0.89442719
7.81024968	2.43310501	5.6639209	7.	2.43310501	11.28007092
4.39089968	3.25576412	2.63058929	3.54400903	1.41421356	1.84390889
10.6976633	0.4472136	4.88262225	5.83095189	9.3059121	9.1214034
3.98497177	1.69705627	1.41421356	1.97989899	3.82099463	4.10365691
3.4	1.56204994	3.73630834	4.00499688	13.29661611	4.6
1.72046505	3.8	7.53126815	3.67695526	7.15541753	5.21536192
3.49284984	2.95296461	1.0198039	2.03960781	3.25576412	5.09901951
3.11126984	4.51220567	5.73061951	3.44093011	2.66833281	5.12249939
8.68331734	1.0198039	6.05309838	11.62755348	2.23606798	5.2
5.05964426	12.16552506	3.28024389	3.3286634	3.56089876	4.07921561
1.28062485	0.82462113	3.84187454	8.95544527	8.50411665	6.74685112
1.26491106	6.35609943	10.66208235	8.78179936	8.82043083	2.63058929
7.78973684	7.26911274	18.24390309	9.33809402	21.66010157	17.19767426
4.81663783	13.61322886	9.13016977	8.43089556	6.596969	12.52517465
15.6588633	3.82099463	1.8	30.90760424	1.2	3.28024389
3.39411255	6.62721661	1.81107703	6.2	0.4472136	6.51459899
7.24982758	4.56508488	3.4	4.10365691	0.4472136	2.97321375
2.33238076	8.58836422	1.21655251	1.78885438	3.40587727	5.28015151
14.11523999	7.64198927	0.89442719	5.11077294	3.16227766	4.49444101
2.40831892	0.89442719	7.15541753	2.84253408	3.39411255	3.68781778
0.4	5.51724569	3.31058907	2.88444102	6.08276253	5.2497619
1.78885438	10.18626526	2.00997512	2.16333077	4.75394573	2.7784888
5.3141321	6.52993109	8.92860571	4.20475921	1.72046505	15.77846634
2.78567766	4.53431362	16.26898891	1.70880075	7.40270221	4.
2.80713377	0.72111026	4.5254834	1.64924225	2.7784888	2.69072481
10.82589488	1.8	4.75394573	3.23109888	1.61245155	9.33809402
2.12602916	7.60263112	5.21536192	3.44093011	5.81377674	2.69072481
1.61245155	1.	1.88679623	4.87031826	4.60434577	3.35261092
5.5027266	1.64924225	1.70880075	2.82842712	7.13862732	2.2090722
3.4525353	5.11077294	5.28015151	4.5607017	2.40831892	3.49857114
5.81377674	9.8020406	7.16100552	17.36663468	2.28035085	5.33666563
4.81663783	4.21900462	3.62215406	7.76659514	7.15541753	7.24706837
1.34164079	5.12249939	4.30813185	4.93963561	1.70880075	5.81377674
0.89442719	2.23606798	3.0528675	6.11882342	1.13137085	0.56568542
4.04474968	1.88679623	0.63245553	2.2090722	1.8973666	3.20624391
1.84390889	2.50599282	2.05912603	5.72712843	3.60555128	2.88444102
2.7202941	4.75394573	4.53431362	9.48683298	5.9464275	2.41660919
4.44072066	14.40138882	1.44222051	3.00665928	2.47386338	5.54616985
12.36931688	1.26491106	1.26491106	3.3286634	2.15406592	1.84390889
3.	4.24264069	3.25576412	3.13049517	2.7202941	2.28035085
5.09901951	5.49181209	3.67695526	0.82462113	2.7202941	1.78885438
2.34093998	5.6639209	5.68506816	3.72021505	5.63560112	0.6
1.34164079	2.2	0.82462113	6.04648658	15.6115342	15.80126577
13.6	9.24770242	16.84399003	12.31097072	17.12308383	14.49413675
18.08977612	13.72443077	8.34505842	4.29418211	14.64923206	16.68172653
2.12602916	4.04969135	15.72895419	15.77466323	16.14434886	14.01855913
15.9298462	15.	3.60555128	1.44222051	2.40831892	7.66550716

3.5383612	1.64924225	1.44222051	2.88444102	2.86356421	5.43323108
2.86356421	11.40175425	1.4	5.92283716	5.45893763	3.2249031
3.0528675	6.4899923	1.84390889	3.04630924	6.35609943	5.63205114
2.56124969	7.14702735	8.08949937	3.88329757	3.67695526	3.0528675
8.8	2.41660919	2.6	5.77234788	9.67677632	3.49284984
6.40312424	3.2249031	5.01597448	1.41421356	2.78567766	4.90306027
5.85491247	8.	8.02246845	2.05912603	2.12602916	6.80294054
6.99714227	5.40370243	3.93954312	5.90931468	9.	4.42718872
4.75394573	4.96789694	11.56027681	8.48999411	8.2	4.81663783
4.20475921	4.12310563	5.04777179	6.13514466	2.2090722	8.32586332
7.46726188	8.54400375	2.60768096	10.81665383	10.63014581	8.44037914
2.7202941	3.93954312	5.26117858	5.01198563	7.9649231	7.07106781
5.84123275	3.4	8.48528137	1.45602198	4.30813185	2.
2.60768096	25.05992817	11.72006826	3.0528675	5.12249939	3.96988665
5.23450093	9.43398113	2.41660919	11.03086579	8.	4.24264069
2.78567766	2.84253408	5.72712843	3.4525353	24.01416249	3.16227766
5.	3.16227766	9.70978888	3.49857114	2.54558441	3.93954312
8.94427191	11.81524439	19.4	3.54400903	4.11825206	13.40149245
6.51459899	16.2431524	20.61940833	8.26075057	14.31223253	9.62081078
11.21605991	8.2097503	16.37681288	3.44093011	14.04706375	4.66476152
10.88117641	5.93969696	17.55676508	2.7202941	16.1245155	12.48519123
18.91560203	27.60652097	11.81185845	13.52479205	3.93954312	4.61735855
2.68328157	4.28018691	2.33238076	4.65188134	5.73061951	6.20322497
6.7941151	9.05538514	3.75765885	3.2249031	8.88144132	1.70880075
2.8	5.99332963	1.8973666	2.80713377	4.27551167	1.64924225
2.43310501	5.2497619	7.82304289	1.88679623	4.16173041	1.52315462
10.12126474	1.26491106	3.20624391	7.61577311	3.57770876	1.64924225
1.61245155	3.02654919	1.	5.38516481	2.47386338	1.78885438
1.8973666	2.7202941	5.21536192	9.0509668	9.20217366	5.2497619
2.95296461	11.03086579	7.72787163	9.9959992	2.34093998	5.83095189
2.6	2.44131112	4.56508488	3.84707681	16.91626436	4.72016949
9.5036835	5.63205114	7.89176786	9.37443332	3.73630834	2.56124969
3.68781778	3.67695526	4.1761226	2.52982213	3.84187454	5.6639209
3.49284984	5.09901951	1.8973666	9.37443332	3.12409987	3.67151195
4.21900462	3.40587727	0.4472136	3.0528675	1.16619038	3.80525952
1.41421356	1.61245155	1.72046505	5.53172667	13.2242202	4.90306027
9.87927123	5.9464275	2.8	1.16619038	2.86356421	6.01664358
0.63245553	3.67695526	10.78146558	15.27612516	11.98832766	10.56787585
8.43800924	3.62215406	1.78885438	2.86356421	4.36806593	7.96241169
4.68614981	5.90592922	5.54616985	7.04556598	16.2111073	3.05941171
2.28035085	4.40454311	1.97989899	2.	1.96977156	2.60768096
4.66904701	1.70880075	3.49284984	1.	1.2	3.12409987
1.34164079	4.61735855	5.73061951	4.30813185	9.83259884	0.89442719
4.40454311	5.28015151	3.2249031	6.12209115	4.68614981	0.56568542
3.54400903	6.09261848	2.56124969	4.90306027	5.4405882	6.08276253
11.3507709	2.95296461	8.56037382	7.3430239	3.25576412	0.63245553
1.2	3.16227766	3.84187454	3.49284984	11.48564321	16.30828011
4.66904701	16.97291961	16.20987353	15.2	13.8	4.56946386

```
  9.8386991  13.93125981 21.09502311 18.80957203 15.84045454 15.4
 13.88092216 13.17269904 15.47384891  3.60555128  4.47213595  4.01995025
  4.4         1.97989899  3.5383612   0.89442719  2.          2.95296461
  2.28035085  3.67695526  3.39411255  3.80525952  3.40587727  4.32666153
  6.20966988  2.80713377  2.7784888   3.31058907  7.60263112  3.75765885
 10.37689742  3.04630924  2.2         5.99332963 13.05986217  9.0354856
  5.81377674  1.52315462  3.64965752  2.84253408  9.81835017  4.25205833
  5.4405882  20.6          4.25205833  5.80344725  2.86356421  1.21655251
  1.78885438 10.19803903  8.98220463  1.56204994  0.82462113  8.1215762
  5.54616985  5.38516481  3.39411255  3.5383612   3.          3.39411255
  6.80294054  1.81107703  6.53911309  3.25576412  4.47213595  7.3593478
  2.52982213  2.12602916  2.68328157  1.28062485  9.96393497  7.37563557
  1.          5.04777179 10.09554357 10.48045801  7.43236167  5.43323108
  3.04630924  2.97321375  3.60555128  2.88444102  1.78885438  4.40454311
  5.68858506  1.44222051  1.07703296 12.82809417  1.26491106  5.16139516
  2.56124969  2.52982213  8.04984472  1.96977156 10.66770828  1.84390889
  4.42718872  2.15406592  0.84852814  1.61245155  3.68781778  1.78885438
  5.33666563  6.01331855  2.16333077  9.04433524  9.26498786  2.47386338
  8.48763807  3.75765885 14.20422472  5.2497619   3.25576412  2.66833281
  4.83321839  3.10483494  3.86264158  4.16173041  7.49666593 10.01798383
  4.80832611 12.04159458  3.42344855  2.          6.17737808  3.94461658
  5.09901951  2.28035085 13.89244399  1.81107703  4.56946386  1.56204994
  4.          6.95701085  3.68781778  2.4         2.63058929  2.66833281
  1.41421356  0.84852814  3.77359245  2.12602916  3.4176015   3.13049517
  5.72712843  1.84390889  5.92283716  3.40587727  2.28035085  2.68328157
  9.87927123  1.84390889  3.20624391  7.37563557  5.          6.27694193
  3.77359245  1.45602198  0.28284271]
---
column name: Outpassed opponents
[0 2 1 3]
---
column name: total_passes
[29 20 13 25 58 34 16 40 32 22 28 19 12 26 23 39 24 30 35 21 14 15  8 11
 41]
---
```

[ ]:
```
# ---------------------------------------------------------- Process features

# convert data type of caregorical columns to int for easier process in the
 ↪future
df.isForward = df.isForward.replace({True: 1, False: 0})
df.isSucceeded = df.isSucceeded.replace({True: 1, False: 0})
df.Team=df.Team.replace({'Team Forward Football_1':0,'Team Forward Football_2':
 ↪1}) # first team 1-> number 1,but seems that it's better to have classified
 ↪value begin from 0
df['Pass type']=df['Pass type'].replace({"Forward pass":0,"Lateral pass":
 ↪1,"Backward pass":2})
```

```
df['Pressure level']=df['Pressure level'].replace({"Full Pressure":2,"Limited␣
↪Pressure":1,"No Pressure":0})
df['Zone']=df['Zone'].replace({'Attack':0,'Defence':1,'Mid field':2})
df['Playing direction_first half']=df['Playing direction_first half'].
↪replace({'left':0,'right':1})
df['Playing direction_second half']=df['Playing direction_second half'].
↪replace({'left':0,'right':1})

# ***********************************************************
```

[ ]:
```
# ----------------------------------------------------------- Add features

df['pass_x']=df["posX_passer"]-df["received_PosX"]
df['pass_y']=df["posY_passer"]-df["received_PosY"]

# ***********************************************************
```

[ ]:
```
# ----------------------------------------------------------- Observe

# evidence for using history of player
set1 = set(df[df.Team==0]['Player_id'])
set2= set(df[df.Team==1]['Player_id'])

print("set1: ",set1)
print("set2: ",set2)
overlap = set1.intersection(set2)
print("intersection:",overlap)

# ***********************************************************
```

```
set1:  {95579, 95580, 95581, 95582, 95583, 95584, 95585, 95586, 95587, 95588,
95589, 95591, 95592, 95593, 95594, 95595, 95597, 95600, 95601, 95986, 95987,
95603, 95988, 95602}
set2:  {95617, 95618, 95624, 95581, 95582, 95583, 95584, 95585, 95586, 95587,
95588, 95589, 95591, 95592, 95593, 95594, 95597, 95600, 95601, 95602, 95987,
95988, 95986}
intersection: {95581, 95582, 95583, 95584, 95585, 95586, 95587, 95588, 95589,
95591, 95592, 95593, 95594, 95597, 95600, 95601, 95602, 95987, 95988, 95986}
```

[ ]:
```
# ----------------------------------------------------------- Observe

# conclusion: from the sorted TimeStamp, I found that there are two matches␣
↪rather than two teams in one match
# -> more obvious, day is different
# ->done
# Also, information is not so much to be periodic-> split method should not use␣
↪timeseriessplit
```

```
df_team_one=df[df.Team==0]
df_team_two=df[df.Team==1]

df_team_one.sort_values('TimeStamp') # check the starting and ending time of␣
↪each match
```

[ ]:
```
     Type           TimeStamp  posX_passer  posY_passer  received_PosX  \
526  Pass 2022-05-05 11:36:14           49           33             55
338  Pass 2022-05-05 11:36:16           55           28             36
439  Pass 2022-05-05 11:36:18           39           40             37
87   Pass 2022-05-05 11:36:24           46           44             56
549  Pass 2022-05-05 11:36:27           57           36             46
..    …                   …          …          …             …
243  Pass 2022-05-05 12:41:27           98           45             80
207  Pass 2022-05-05 12:41:35           73           39             65
144  Pass 2022-05-05 12:41:39           63           30             45
86   Pass 2022-05-05 12:41:44           44            9             35
28   Pass 2022-05-05 12:41:47           38           25             30

     received_PosY  isForward  isSucceeded  receiverId  Player_id  …  \
526             29          0            1     95594.0      95602  …
338             39          1            0         NaN      95594  …
439             45          0            1     95589.0      95581  …
87              39          0            1     95579.0      95601  …
549             19          0            1     95597.0      95579  …
..             …        …          …         …        …  …
243             42          1            1     95583.0      95603  …
207             36          1            1     95601.0      95583  …
144              8          0            1     95597.0      95601  …
86              24          0            0         NaN      95597  …
28              32          1            1     95594.0      95582  …

     Playing direction_first half Playing direction_second half  x_pitchsize  \
526                            0                             0           99
338                            0                             0           99
439                            1                             1           99
87                             0                             0           99
549                            0                             0           99
..                           …                           …          …
243                            0                             0           99
207                            0                             0           99
144                            0                             0           99
86                             0                             0           99
28                             0                             0           99

    y_pitchsize  Pressure level  Distance to first opponent  \
```

```
526            63               0                   8.800000
338            63               0                   5.656854
439            63               0                   4.440721
87             63               2                   1.000000
549            63               1                   2.126029
..            ...             ...                   ...
243            63               0                  30.907604
207            63               1                   2.973214
144            63               0                   5.261179
86             63               1                   2.607681
28             63               0                   4.386342

     Outpassed opponents  total_passes  pass_x  pass_y
526                    0            23      -6       4
338                    1            20      19     -11
439                    0            26       2      -5
87                     0            58     -10       5
549                    1            39      11      17
..                   ...           ...     ...     ...
243                    0            16      18       3
207                    1            34       8       3
144                    1            58      18      22
86                     1            25       9     -15
28                     1            29       8      -7

[627 rows x 30 columns]
```

[ ]:
```python
df_team_two.sort_values('TimeStamp')

# **********************************************************
```

[ ]:
```
     Type           TimeStamp  posX_passer  posY_passer  received_PosX  \
811  Pass 2022-05-12 12:06:20           59           20             68
710  Pass 2022-05-12 12:06:22           66           25             61
958  Pass 2022-05-12 12:06:25           59           15             70
998  Pass 2022-05-12 12:06:29           72           40             73
711  Pass 2022-05-12 12:06:35           79           35             88
..    ...                 ...          ...          ...            ...
986  Pass 2022-05-12 13:06:33           42            2             44
924  Pass 2022-05-12 13:06:51           56           44             79
869  Pass 2022-05-12 13:06:54           84           41             81
889  Pass 2022-05-12 13:07:13           11           29             11
855  Pass 2022-05-12 13:07:17           13           25             12

     received_PosY  isForward  isSucceeded  receiverId  Player_id  ...  \
811             27          1            0         NaN      95988  ...
710             15          0            0         NaN      95588  ...
```

```
958              39         0            0        NaN      95592  …
998              31         0            1    95588.0      95583  …
711              34         0            1    95603.0      95588  …
..              …        …          …         …        …   …
986               6         0            1    95586.0      95592  …
924              40         1            1    95587.0      95589  …
869              40         0            0        NaN      95587  …
889              21         0            1    95987.0      95591  …
855              36         0            1    95600.0      95987  …

     Playing direction_first half Playing direction_second half  x_pitchsize  \
811                             1                             1           99
710                             0                             0           99
958                             1                             1           99
998                             0                             0           99
711                             0                             0           99
..                            …                             …          …
986                             1                             1           99
924                             1                             1           99
869                             1                             1           99
889                             1                             1           99
855                             1                             1           99

    y_pitchsize  Pressure level  Distance to first opponent  \
811          63               2                    1.811077
710          63               1                    2.720294
958          63               2                    0.824621
998          63               0                    9.963935
711          63               0                    5.215362
..          …              …                         …
986          63               2                    1.400000
924          63               1                    2.416609
869          63               1                    3.492850
889          63               0                   15.473849
855          63               0                   11.350771

     Outpassed opponents  total_passes  pass_x  pass_y
811                    0            21      -9      -7
710                    0            35       5      10
958                    2            29     -11     -24
998                    0            21      -1       9
711                    0            35      -9       1
..                   …            …     …     …
986                    0            29      -2      -4
924                    0            20     -23       4
869                    0            14       3       1
889                    0            20       0       8
```

```
855                    0            24       1     -11

[515 rows x 30 columns]
```

[ ]:
```python
# ---------------------------------------------------------- Observe

df.groupby(['Player_id']).size()

# calculate this in order to make sure there is no player with just a single␣
↪line data
# -> avoid when splitting dataset, there is no solution to split into training␣
↪and test dataset
# -> the result is positive-> done

# *********************************************************
```

[ ]:
```
Player_id
95579    39
95580    22
95581    56
95582    53
95583    55
95584    39
95585    36
95586    69
95587    36
95588    67
95589    48
95591    46
95592    51
95593    28
95594    36
95595    39
95597    44
95600    53
95601    99
95602    31
95603    16
95617    32
95618    15
95624    25
95986    23
95987    44
95988    40
dtype: int64
```

[ ]:
```python
# ----------------------------------------------------------- Observe

def print_column_name(df):
    for col in df:
        print("column name:",col)
print_column_name(df)


# feature expansion
# 1. this is obvious that the distance between passer and receicver has the␣
 ↪influence on if the pass is successful
# -> found that Pass length has been calculated->done
# -> but angle of start is also informative-> angle is also calculated->done
# 2. the difference between startTime and the start time have an influence on␣
 ↪the physical strength-> done-> not informative, not peroidic->discard
# 3. if the exact position or the interval of position has an influence on the␣
 ↪result-> generate features-> how to use it
#  * feature selection ( based on model, or correlation )
#  * binned feature and then feature selection
#  * alternative column: zone -> selected
# 4. note that former part is more team 1, later part is for team 2, then there␣
 ↪is also the overlap timestamp of records, try to use time to generate␣
 ↪features-> not correct
# time can also use Time block, or calculated time using minus -> select Time␣
 ↪Block
# 5. time related features
# -> how many opponents appear in 5s in a exact scope-> don't have enough data
# -> how many friends appear in 5s or around 5s in the same field-> don't have␣
 ↪enough data
# -> how many rival passes-> don't have enough data
# -> how many friends passes-> done
# -> how many rivals in a certain fields.-> don't have enough data
# -> how many friends in a certain fields.-> don't have enough data

# 5. Angle passe is for ridian, maybe degree is better -> not sure,but don't␣
 ↪think so, just leave this idea here-> select ridian

# consider the success rate of pass of one player-> should done after␣
 ↪split->done

# outlier?-> not suitable in this project->done

# ***********************************************************
```

```
column name: Type
column name: TimeStamp
column name: posX_passer
```

```
column name: posY_passer
column name: received_PosX
column name: received_PosY
column name: isForward
column name: isSucceeded
column name: receiverId
column name: Player_id
column name: Team
column name: startTime
column name: matchDuration
column name: Club
column name: Time block
column name: Zone
column name: Area Football Pitch
column name: Angle Passe
column name: Pass type
column name: Pass length
column name: Playing direction_first half
column name: Playing direction_second half
column name: x_pitchsize
column name: y_pitchsize
column name: Pressure level
column name: Distance to first opponent
column name: Outpassed opponents
column name: total_passes
column name: pass_x
column name: pass_y
```

[ ]:
```
# --------------------------------------------------------- Add features
# How many friends passes in x second

df.set_index('TimeStamp', drop=True, inplace=True)
df = df.sort_index()
window=df['Zone'].rolling('10s')

def count_same_zone(x, current_player):
    return x[x == current_player].count()

df['player_num_in_same_zone'] = window.apply(lambda x: count_same_zone(x,␣
 ↪x[0]), raw=False)
df=df.reset_index()
# *********************************************************
```

[ ]:
```
# --------------------------------------------------------- Observe: check␣
 ↪missing values
def get_none_percent(df):
    return df.isna().sum()/df.shape[0]
```

```python
def get_none_num(df):
    return df.isna().sum()

get_none_percent(df)
# ********************************************************
```

[ ]:
```
TimeStamp                          0.000000
Type                               0.000000
posX_passer                        0.000000
posY_passer                        0.000000
received_PosX                      0.000000
received_PosY                      0.000000
isForward                          0.000000
isSucceeded                        0.000000
receiverId                         0.285464
Player_id                          0.000000
Team                               0.000000
startTime                          0.000000
matchDuration                      0.000000
Club                               0.000000
Time block                         0.000000
Zone                               0.000000
Area Football Pitch                0.000000
Angle Passe                        0.000000
Pass type                          0.000000
Pass length                        0.005254
Playing direction_first half       0.000000
Playing direction_second half      0.000000
x_pitchsize                        0.000000
y_pitchsize                        0.000000
Pressure level                     0.000000
Distance to first opponent         0.000000
Outpassed opponents                0.000000
total_passes                       0.000000
pass_x                             0.000000
pass_y                             0.000000
player_num_in_same_zone            0.000000
dtype: float64
```

[ ]:
```python
# -------------------------------------------------------- Fill missing value␣
↪of Pass length
# even the pass is not successful, then there is a expected sending point and a␣
↪received point, so all the Pass length could be calculated
# -> the missing value of Pass length is solved-> done

def compute_lenght(df):
    import math
```

```
    length=np.
↪sqrt((df['posX_passer']-df['received_PosX'])**2+(df['posY_passer']-df['received_PosY'])**2)
    return length
fill_series=compute_lenght(df)
df["Pass length"]=df["Pass length"].fillna(fill_series)
# *********************************************************
```

[ ]:
```
# --------------------------------------------------------- Observe
# found that there is no receivedId if isSucceed is false, check if they have␣
↪the one-to-one relation
# -> the answer is yes, so receivedId is also the answer!!! we can't use it in␣
↪training a model
# -> just delete it (receivedId)->done

1-df['isSucceeded'].sum()/df.shape[0]

# *********************************************************
```

[ ]: 0.2854640980735552

[ ]:
```
# --------------------------------------------------------- Observe
# considering about the imbalance of isSucceeded

print("successful:",df['isSucceeded'].sum()/df.shape[0])
print("failure:",1-df['isSucceeded'].sum()/df.shape[0])

# -> not so balanced
# -> so try to interpolate the minor class or reduce the major class or using␣
↪another score metric *** different choice
# -> which is suitable? but imbalanced data is not series, so tring score␣
↪metric, for example, f1 score->selected
# *********************************************************
```

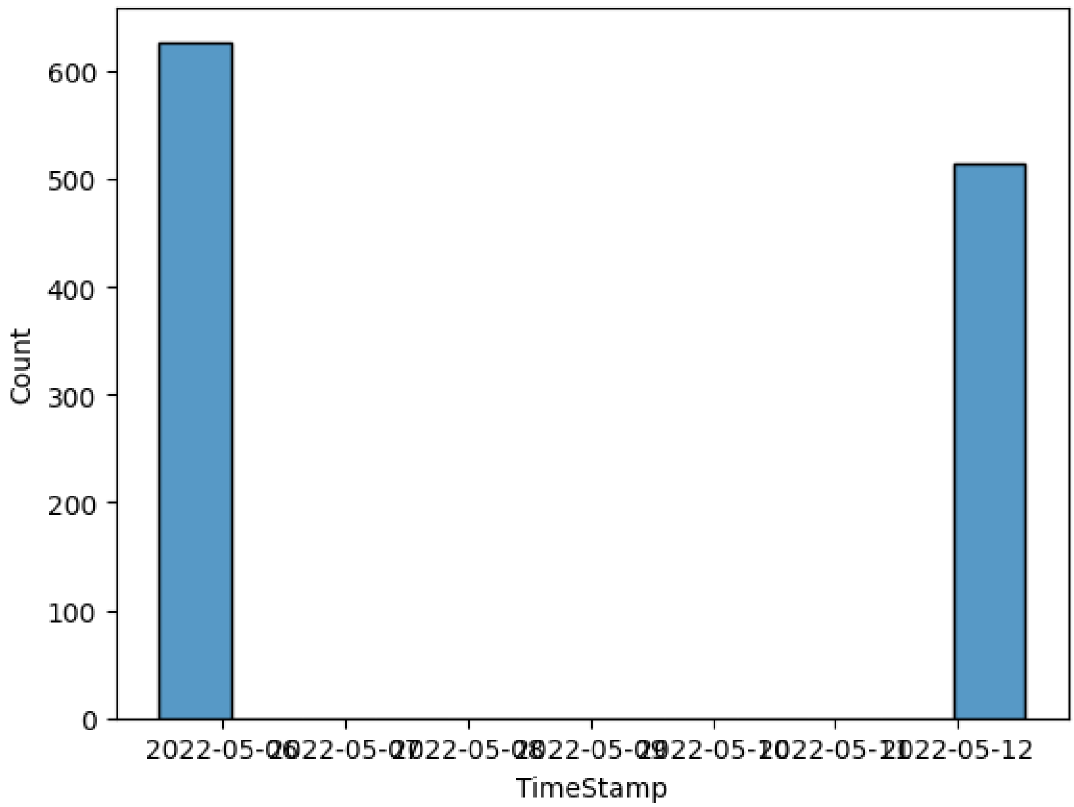
```
successful: 0.7145359019264448
failure: 0.2854640980735552
```

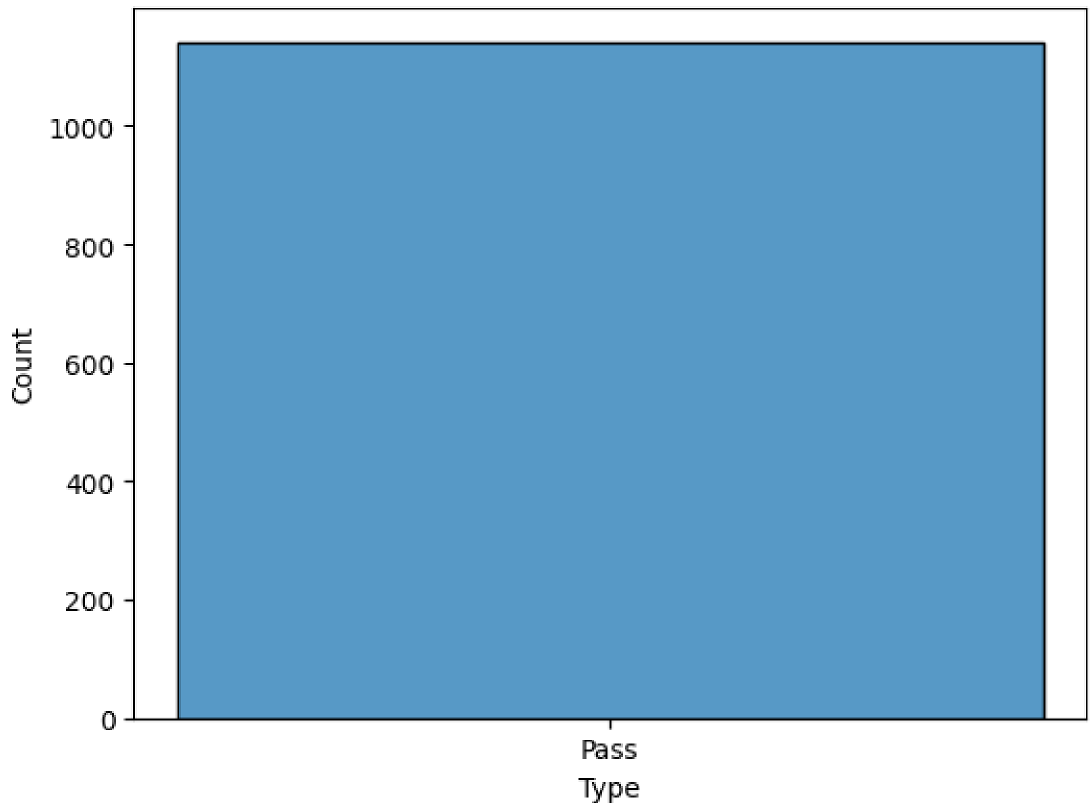
[ ]:
```
# --------------------------------------------------------- Observe
# Define a custom function that returns the data type of a column

def check_dtype(col):
    return col.dtype

# Apply the custom function to each column of the DataFrame
df.apply(check_dtype)

# *********************************************************
```

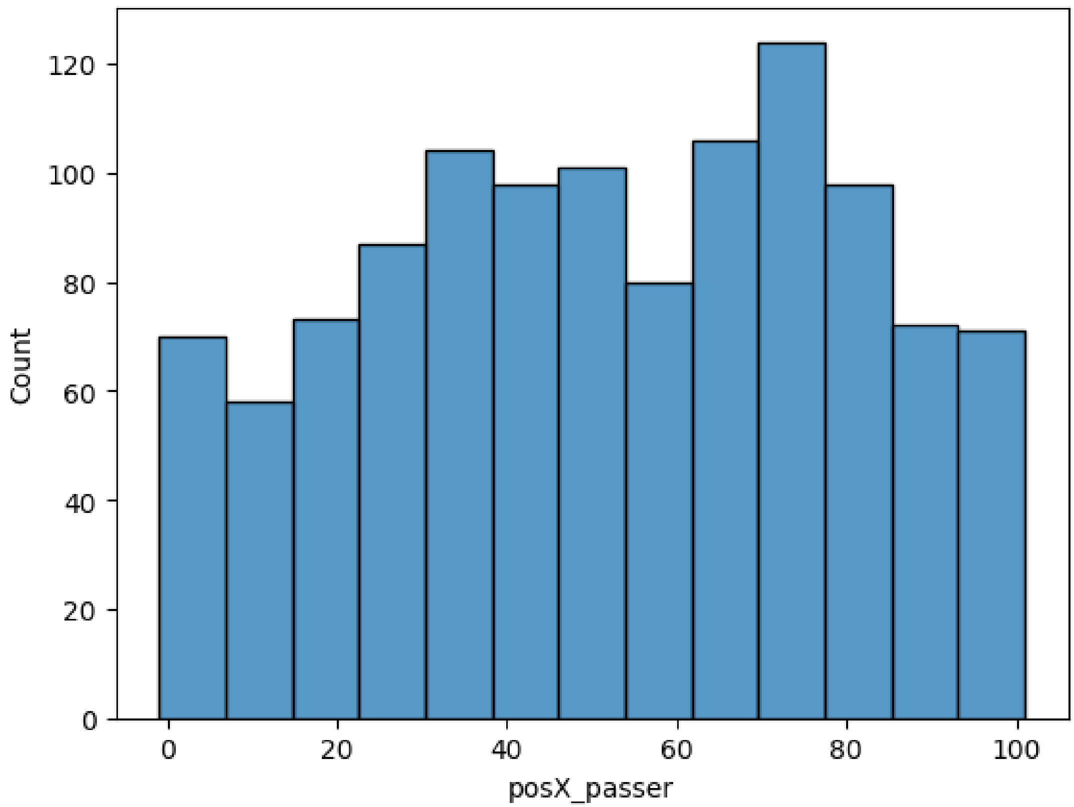
[ ]:
```
TimeStamp                              datetime64[ns]
Type                                           object
posX_passer                                     int64
posY_passer                                     int64
received_PosX                                   int64
received_PosY                                   int64
isForward                                       int64
isSucceeded                                     int64
receiverId                                    float64
Player_id                                       int64
Team                                            int64
startTime                              datetime64[ns]
matchDuration                                 float64
Club                                           object
Time block                                      int64
Zone                                            int64
Area Football Pitch                             int64
Angle Passe                                   float64
Pass type                                       int64
Pass length                                   float64
Playing direction_first half                    int64
Playing direction_second half                   int64
x_pitchsize                                     int64
y_pitchsize                                     int64
Pressure level                                  int64
Distance to first opponent                    float64
Outpassed opponents                             int64
total_passes                                    int64
pass_x                                          int64
pass_y                                          int64
player_num_in_same_zone                       float64
dtype: object
```

[ ]:
```
# --------------------------------------------------------- Add features
# generate bin values
num_bins = 4
df['angle_bins'] = pd.qcut(df['Angle Passe'], num_bins)

lst=df['angle_bins'].unique()
my_dic={}
my_dict = {}
for i, val in enumerate(lst):
    my_dict[val] = i

df.angle_bins = df.angle_bins.replace(my_dict).astype("int64")
# *********************************************************
```

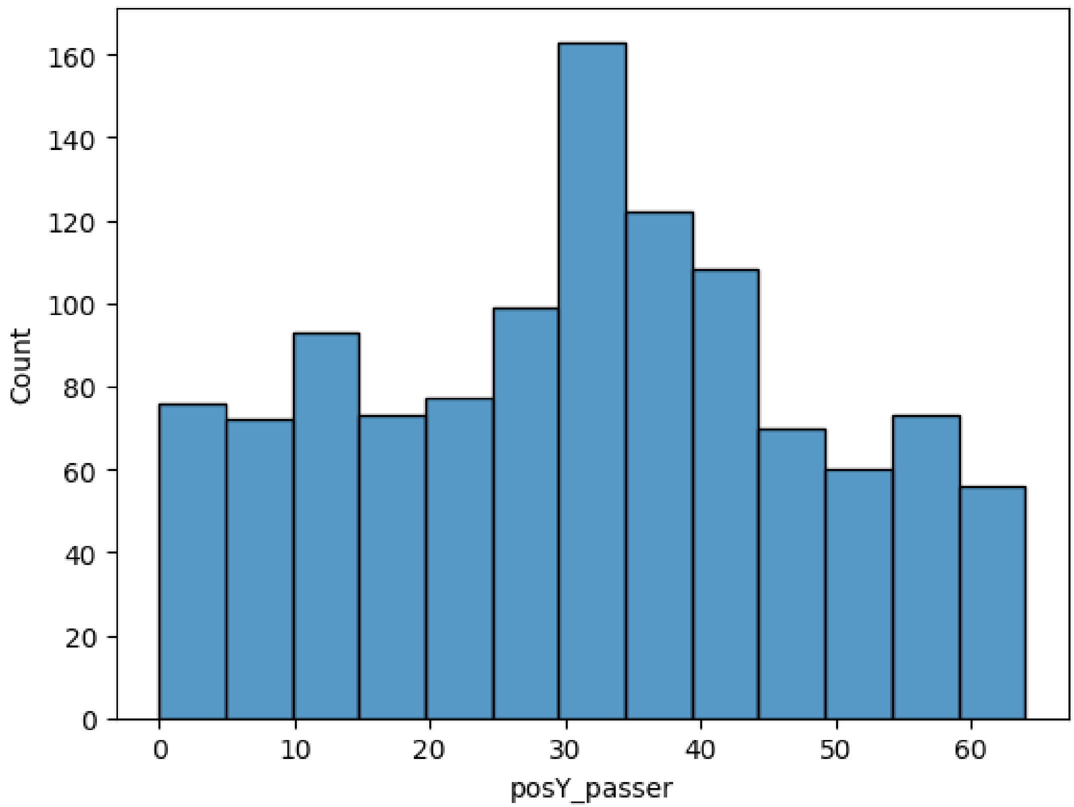
```
[ ]: # ---------------------------------------------------------- Observe
import seaborn as sns
def get_distribution_of_each_column(df):
    # Loop over the features
    for col in df:
        # Select the feature
        feature = df[col]

        # Plot the distribution of the feature
        sns.histplot(feature)
        plt.show()

def get_distribution_columns(df,col_lst):
    # Loop over the features
    for col in col_lst:
        # Select the feature
        feature = df[col]

        # Plot the distribution of the feature
        sns.histplot(feature)
        plt.show()
get_distribution_of_each_column(df)
# found that Pass length and Distance to first oppoment have log distribution,␣
 ↪also pair_count
# pair_count is necesseay for transformation???-> selected not
# **********************************************************

# ---------------------------------------------------------- Turn distribution
# actually, this part should be put after all the features are generated, but I␣
 ↪had done so, the result is the same
# -> considering the interface, it's convenient to put it here
from sklearn.preprocessing import QuantileTransformer
transformer = QuantileTransformer(output_distribution='normal')
df['Pass length'] = transformer.fit_transform(df['Pass length'].values.
 ↪reshape(-1,1))
df['Distance to first opponent'] = transformer.fit_transform(df['Distance to␣
 ↪first opponent'].values.reshape(-1,1))
# df['pair_count'] = transformer.fit_transform(df['pair_count'].values.
 ↪reshape(-1,1))


# for validation
get_distribution_columns(df,['Pass length','Distance to first opponent']) #␣
 ↪found that normal distribution has been transformed successfully
# **********************************************************
```


0
100
200
300
400
500
600
Count
2022-05-06
2022-05-07
2022-05-08
2022-05-09
2022-05-10
2022-05-11
2022-05-12
TimeStamp

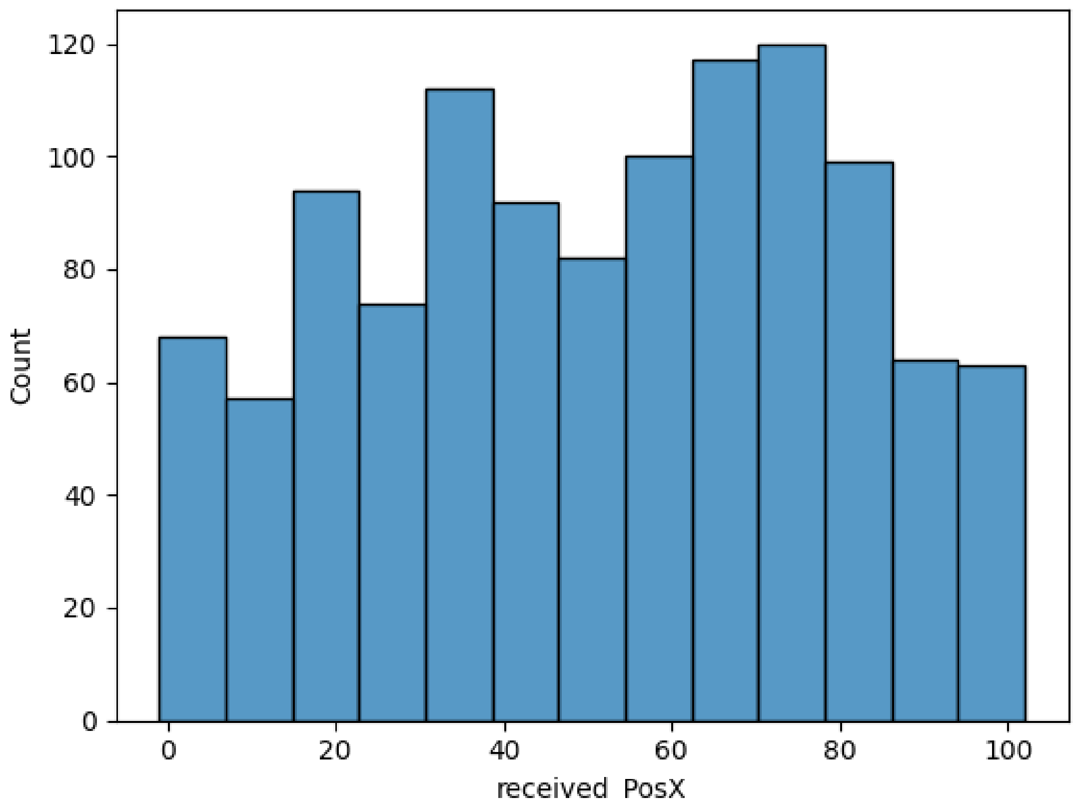

1000
800
600
400
200
0
Count
Pass
Type



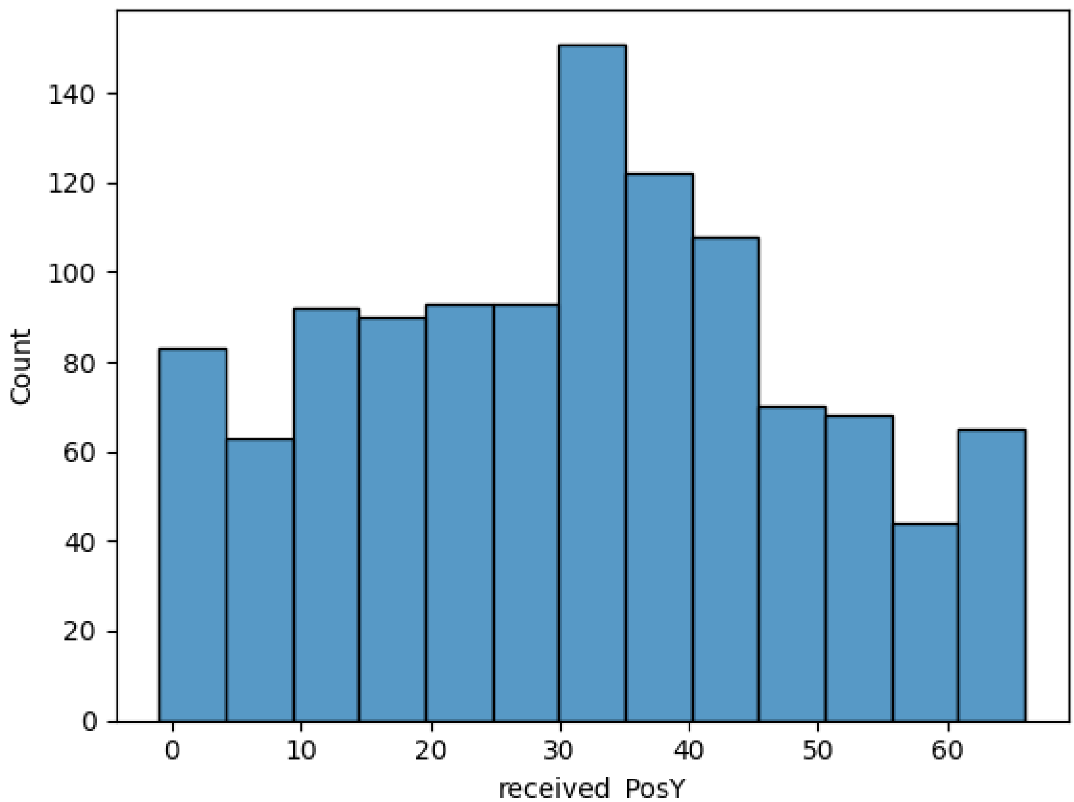
Count
posX_passer
0
20
40
60
80
100
120

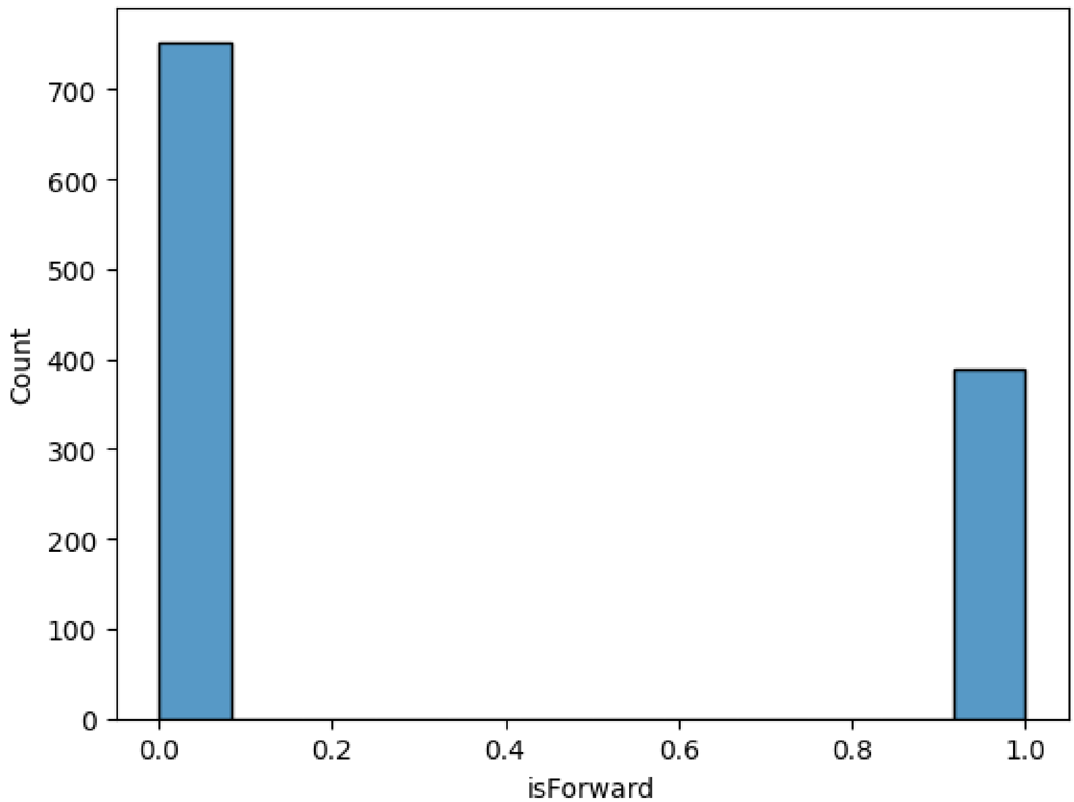


160
140
120
100
80
60
40
20
0
Count
0
10
20
30
40
50
60
posY_passer

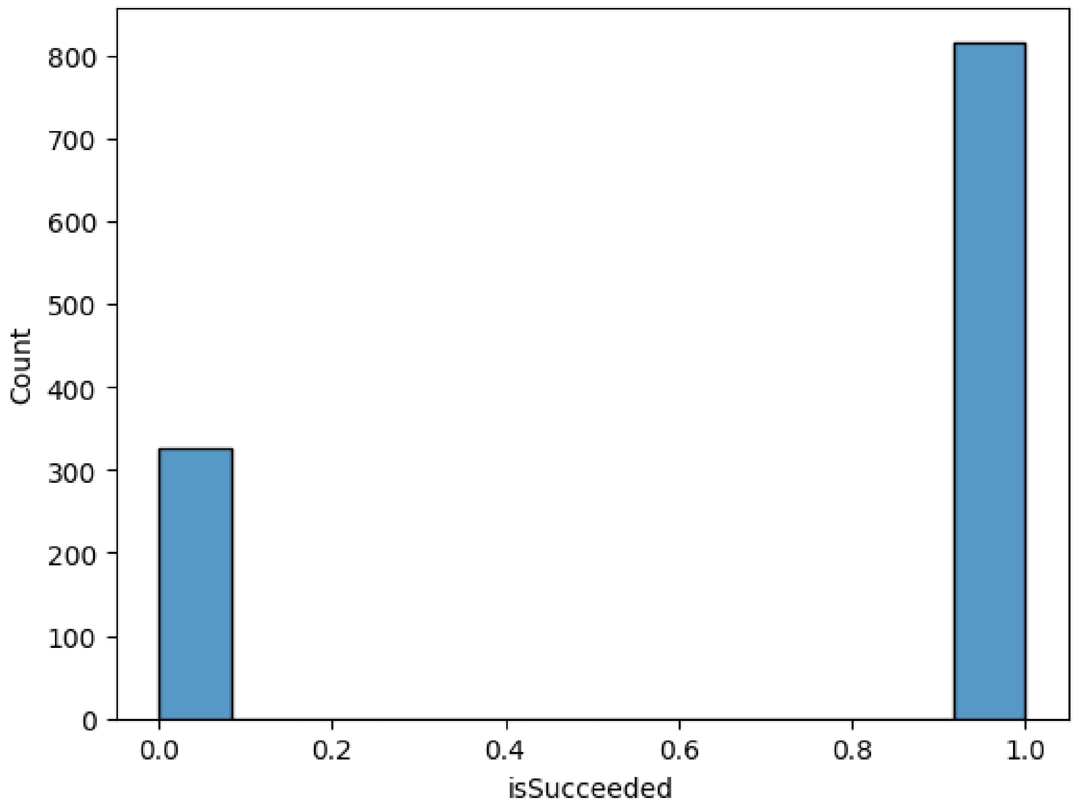


Count
received_PosX
0
20
40
60
80
100
120

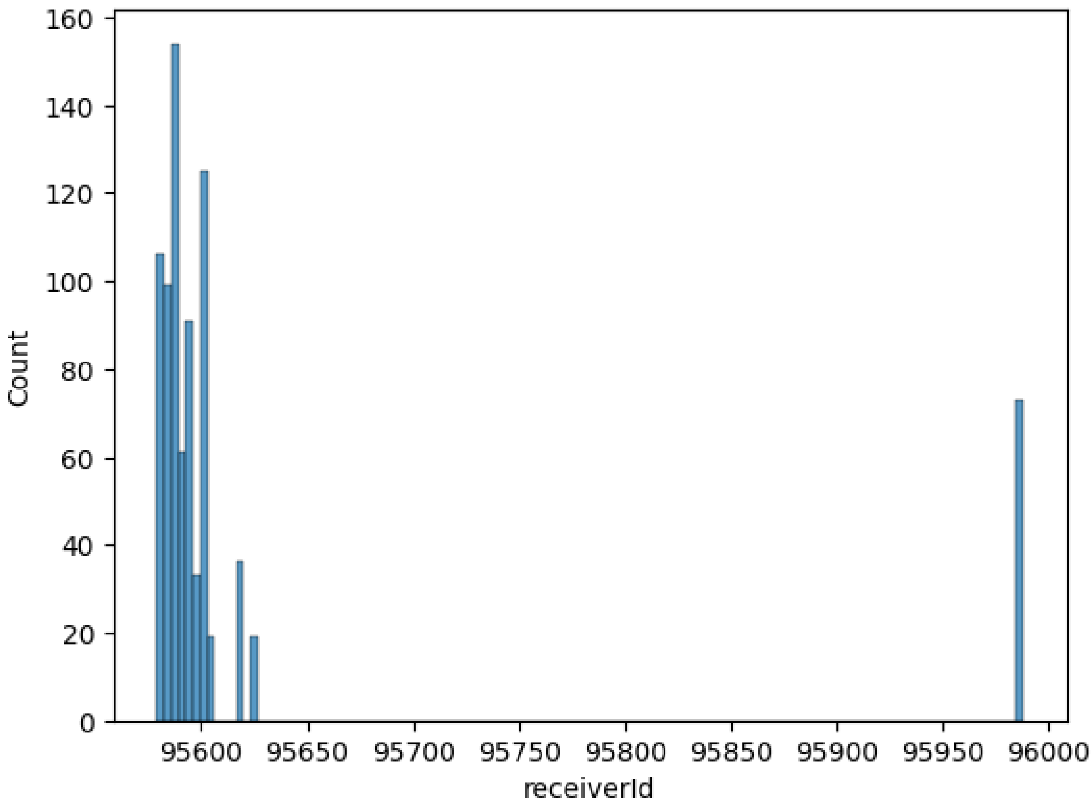

140
120
100
80
60
40
20
0
Count
0
10
20
30
40
50
60
received_PosY

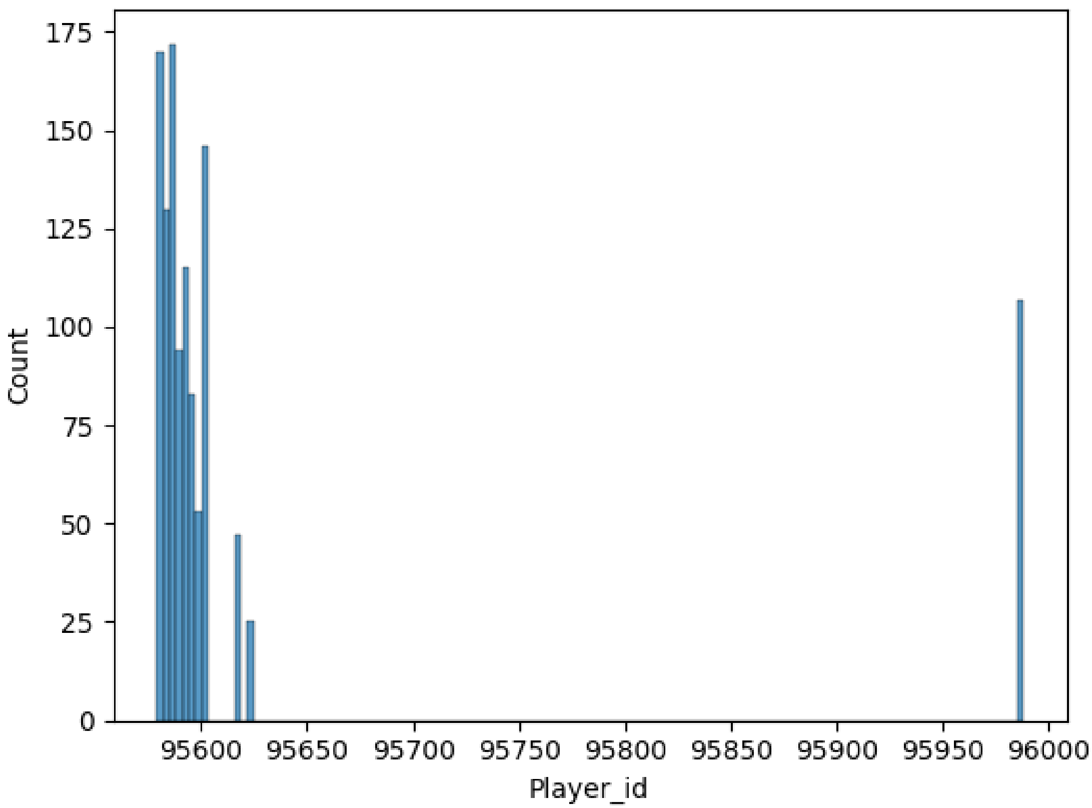


700
600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
isForward

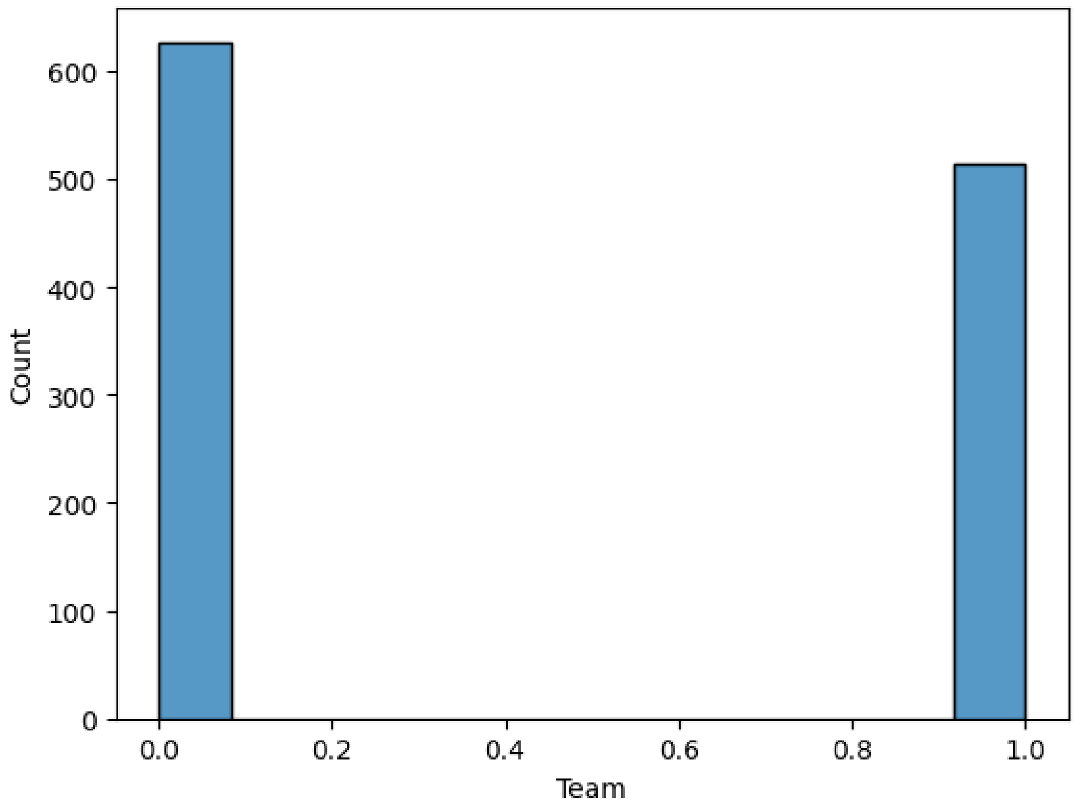


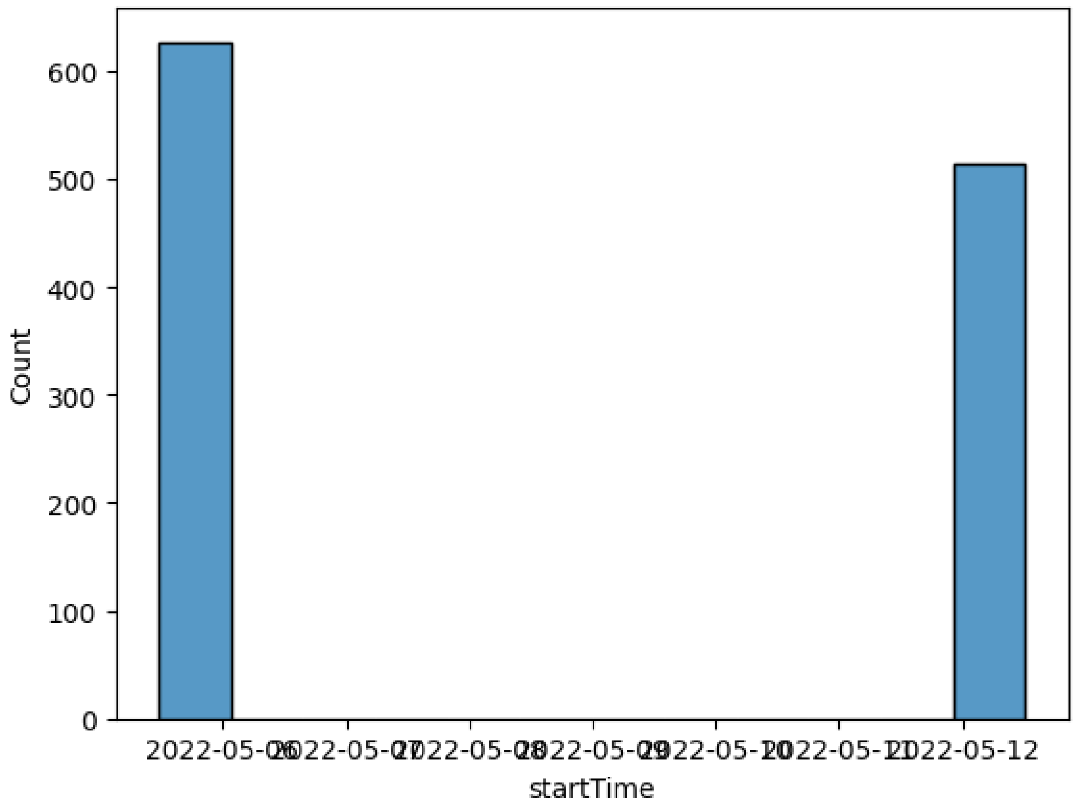
800
700
600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
isSucceeded


Count
receiverId
0
20
40
60
80
100
120
140
160
95600
95650
95700
95750
95800
95850
95900
95950
96000

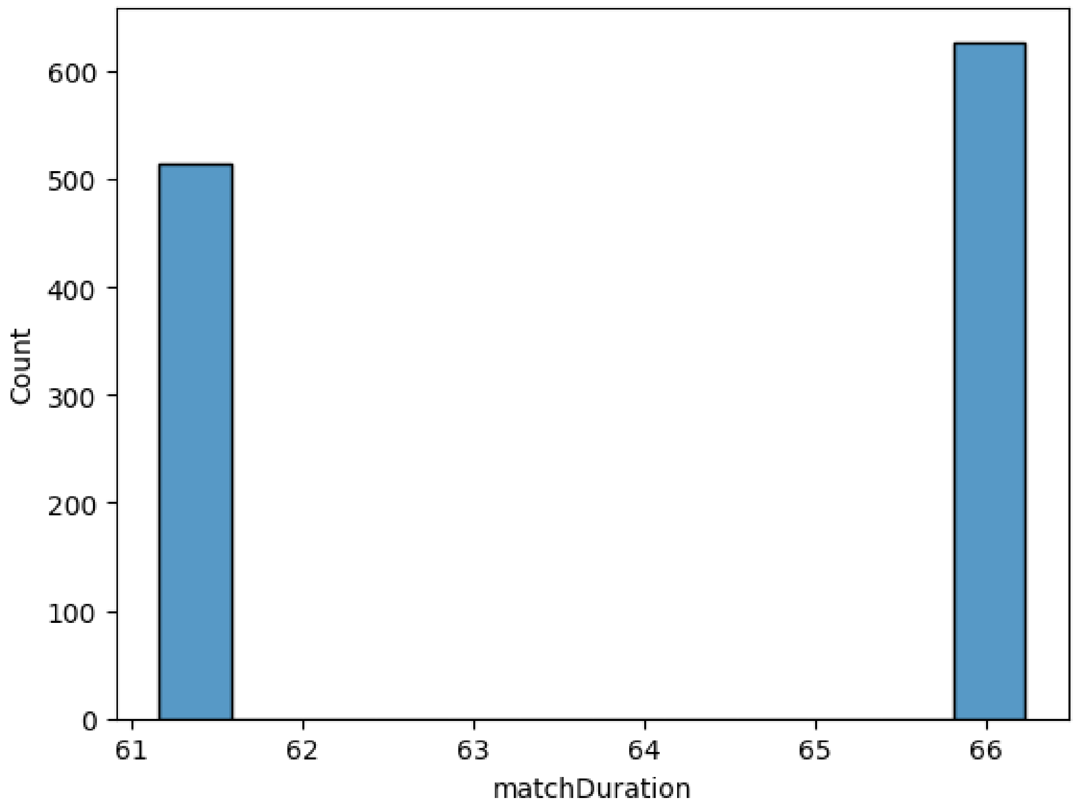


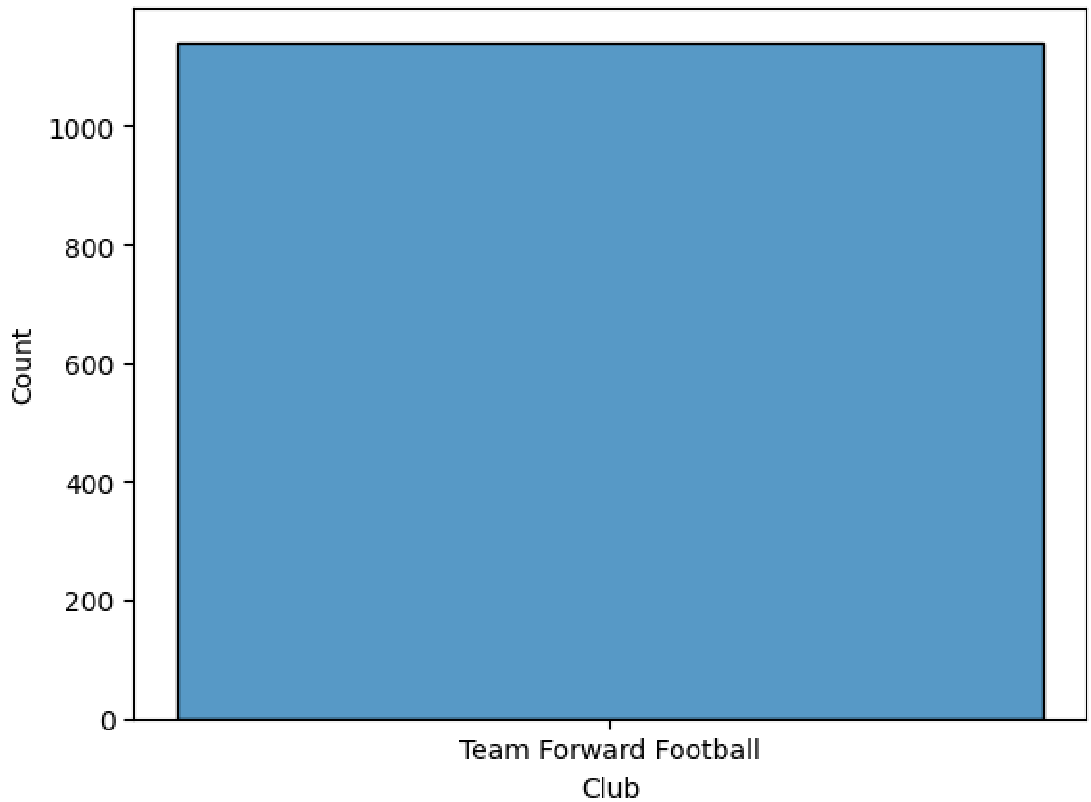
175
150
125
100
75
50
25
0
Count
95600
95650
95700
95750
95800
95850
95900
95950
96000
Player_id

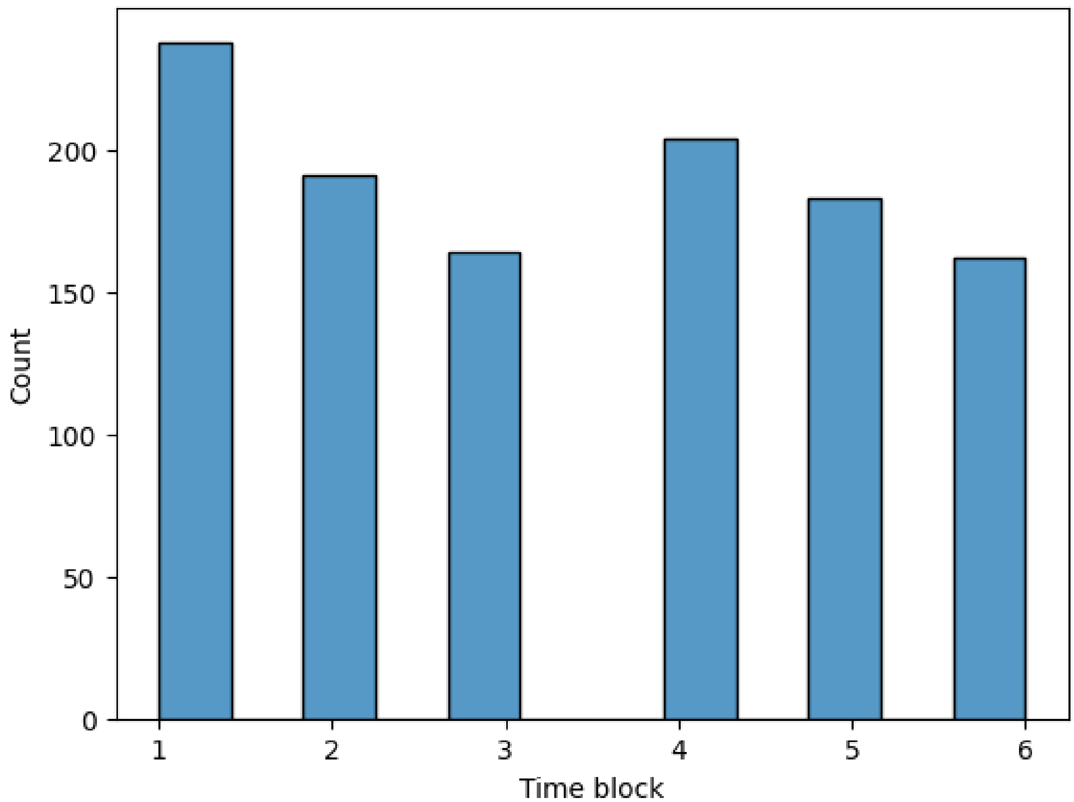

600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Team



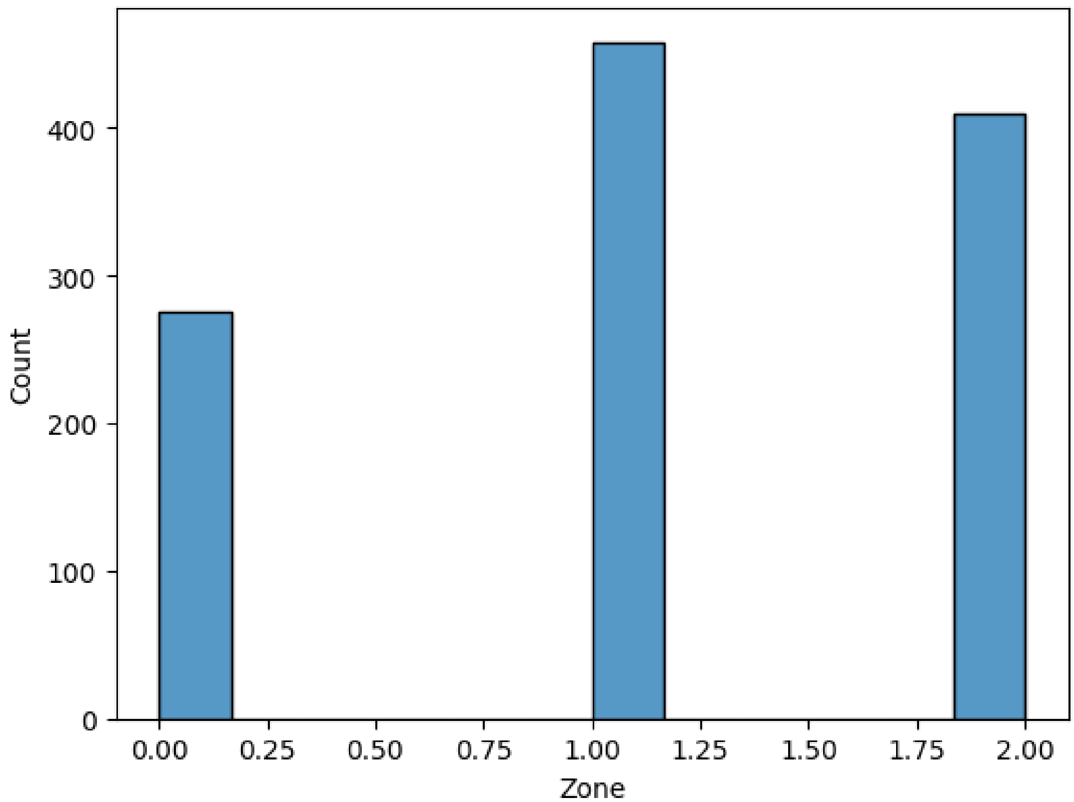
600
500
400
300
200
100
0
Count
2022-05-06
2022-05-07
2022-05-08
2022-05-09
2022-05-10
2022-05-11
2022-05-12
startTime


Count
matchDuration
0
100
200
300
400
500
600
61
62
63
64
65
66

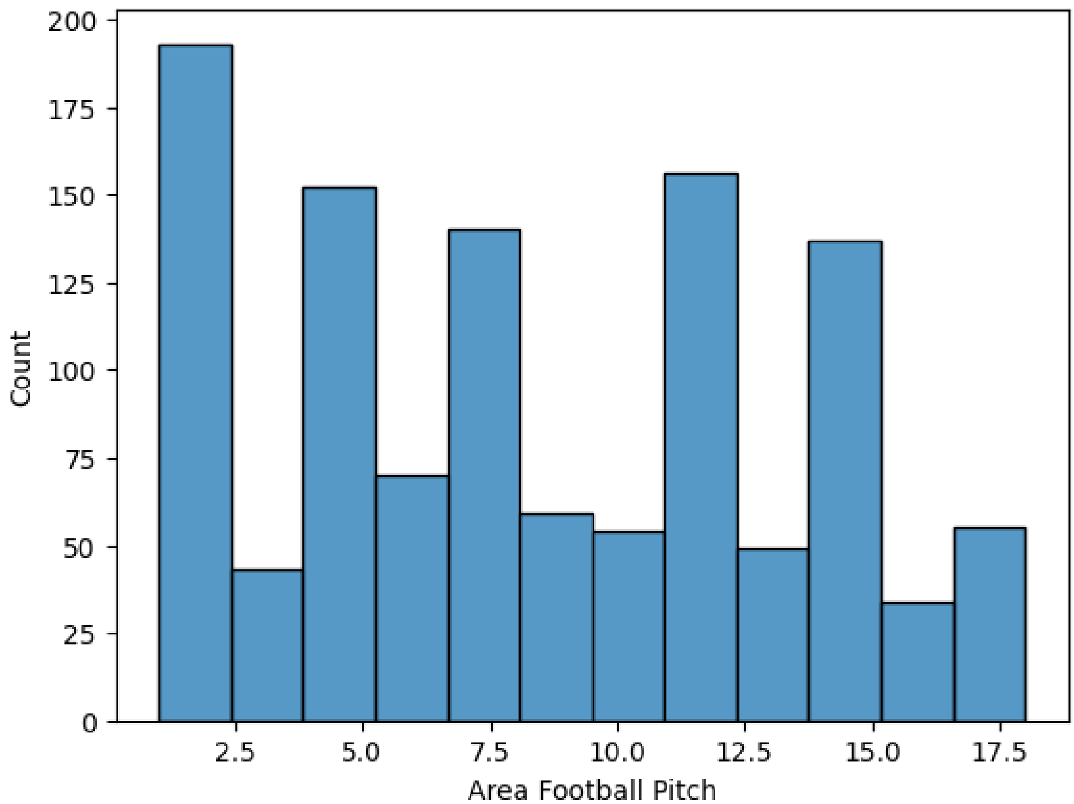

1000
800
600
400
200
0
Count
Team Forward Football
Club



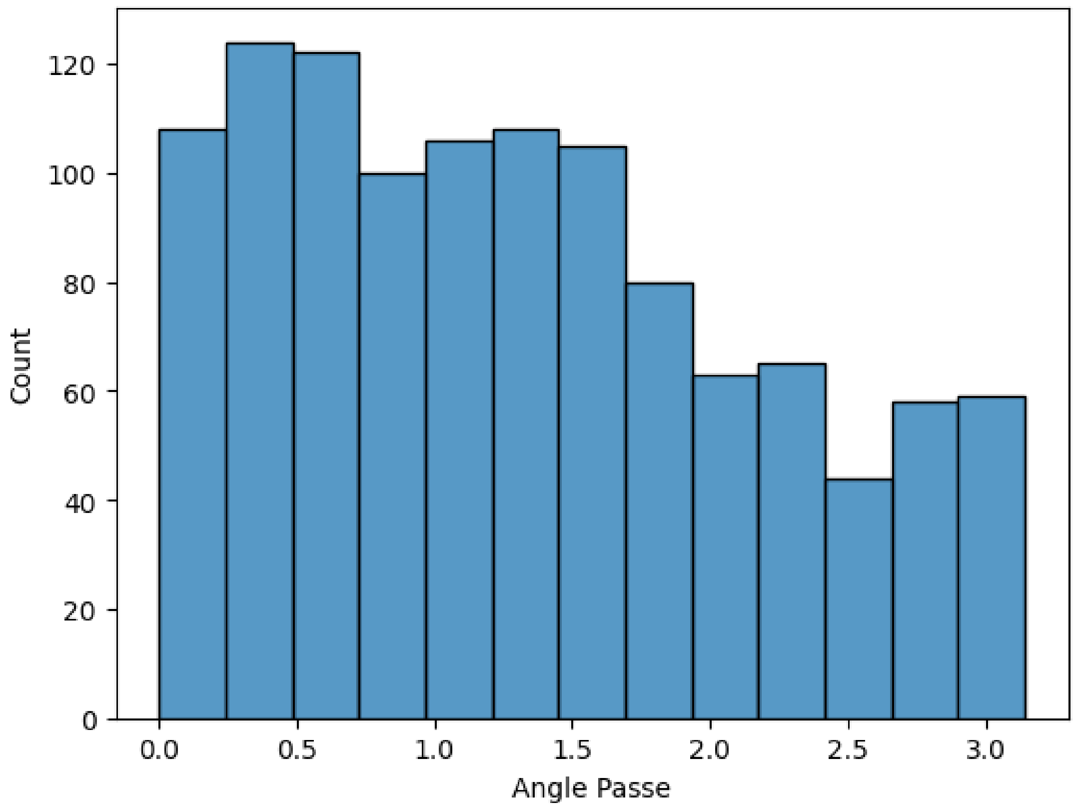
Count
200
150
100
50
0
1
2
3
4
5
6
Time block



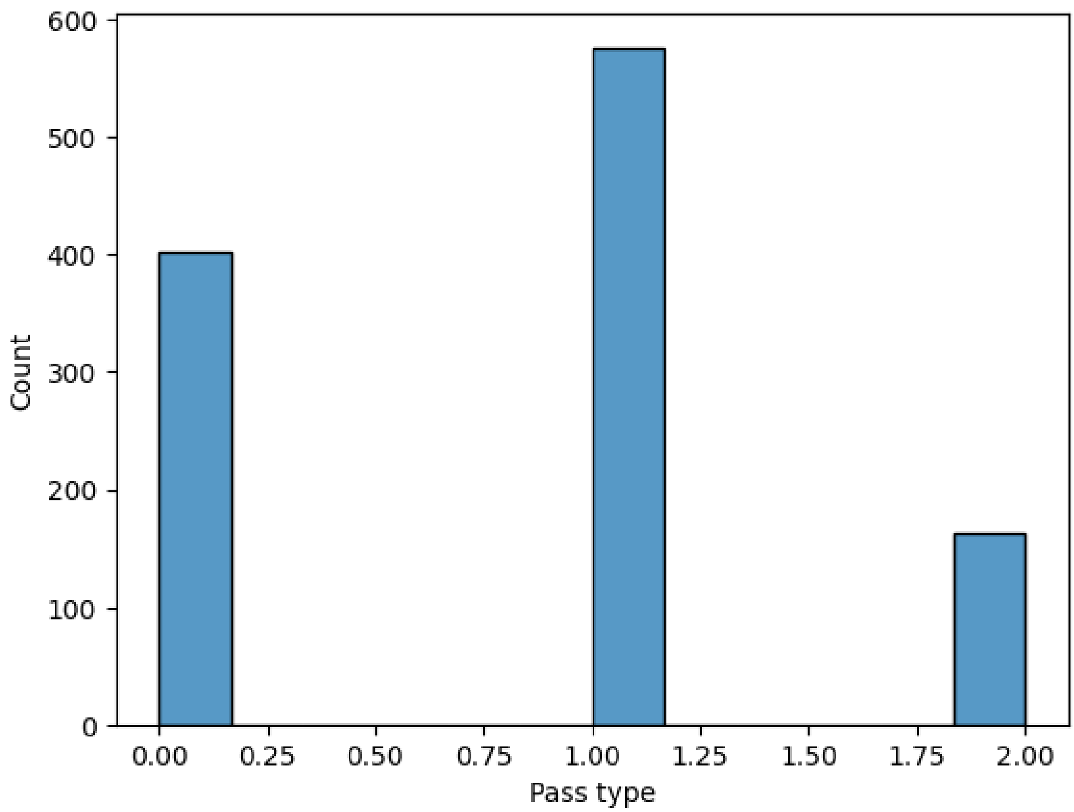
Count
400
300
200
100
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Zone

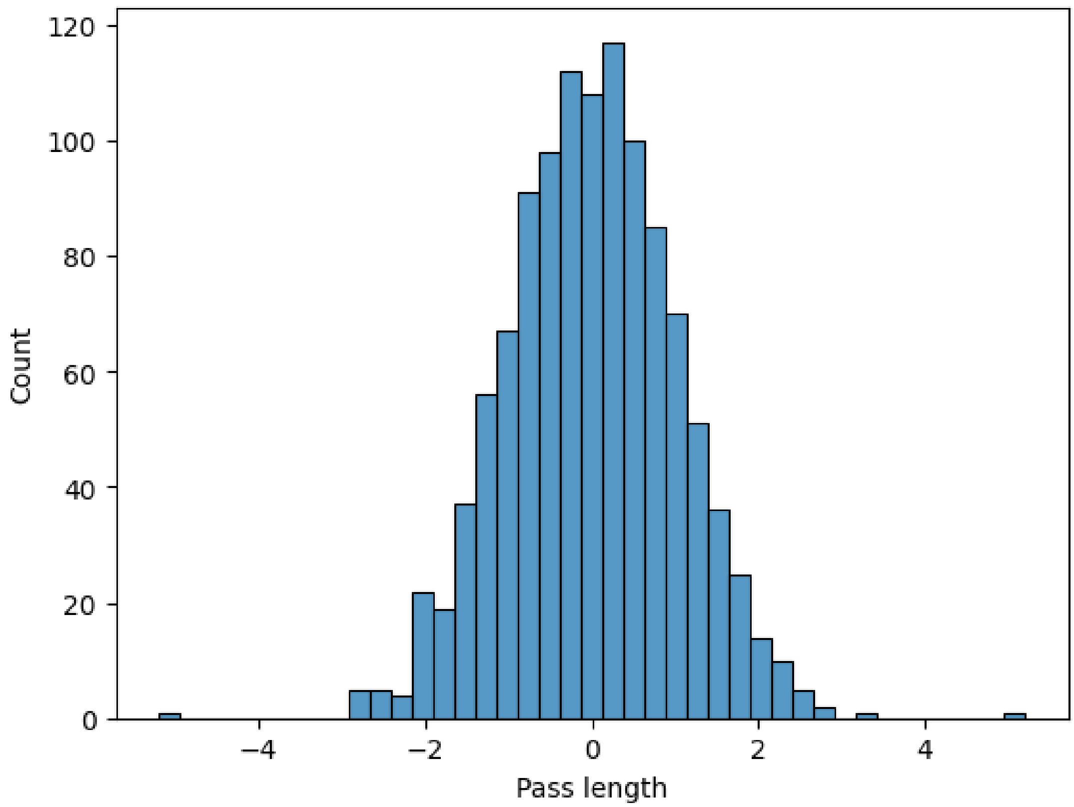


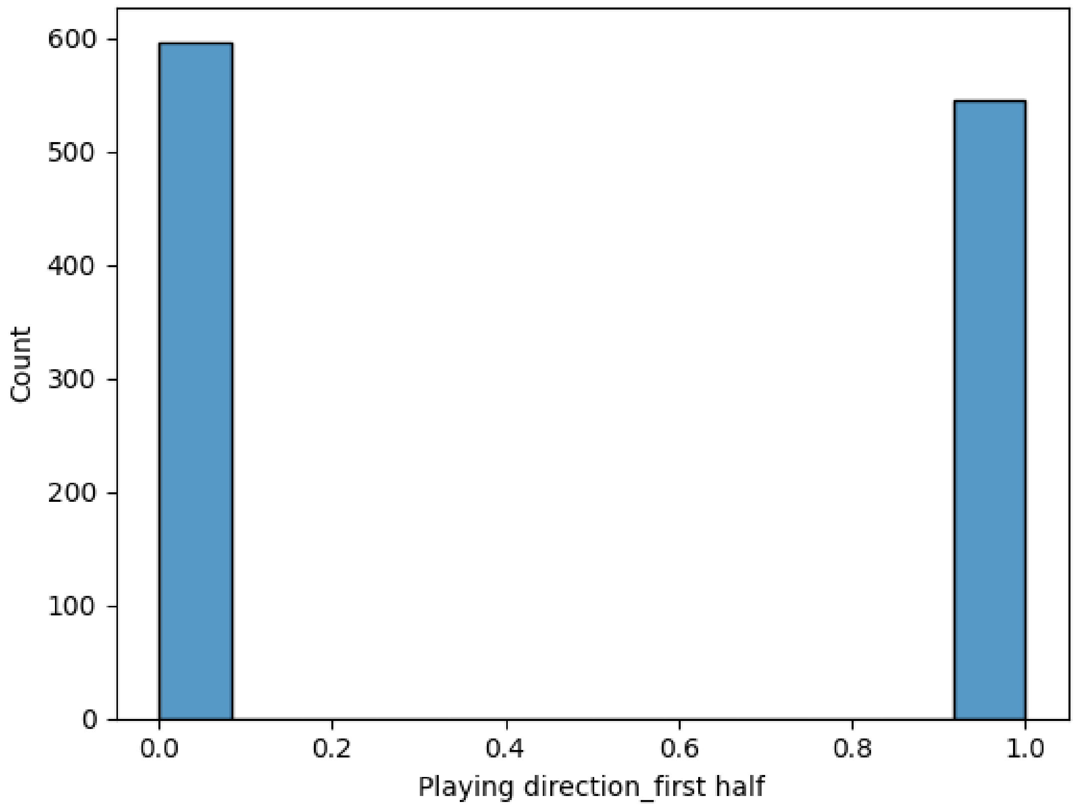
200
175
150
125
100
75
50
25
0
Count
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Area Football Pitch



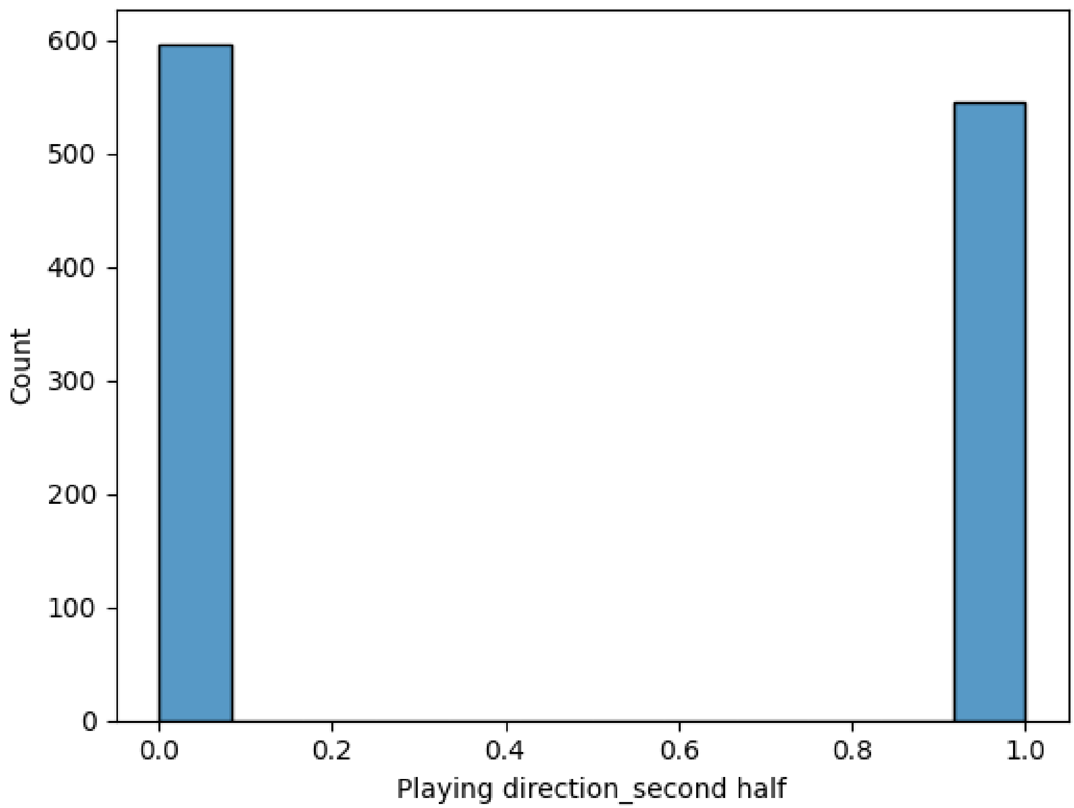
Count
Angle Passe


600
500
400
300
200
100
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Pass type

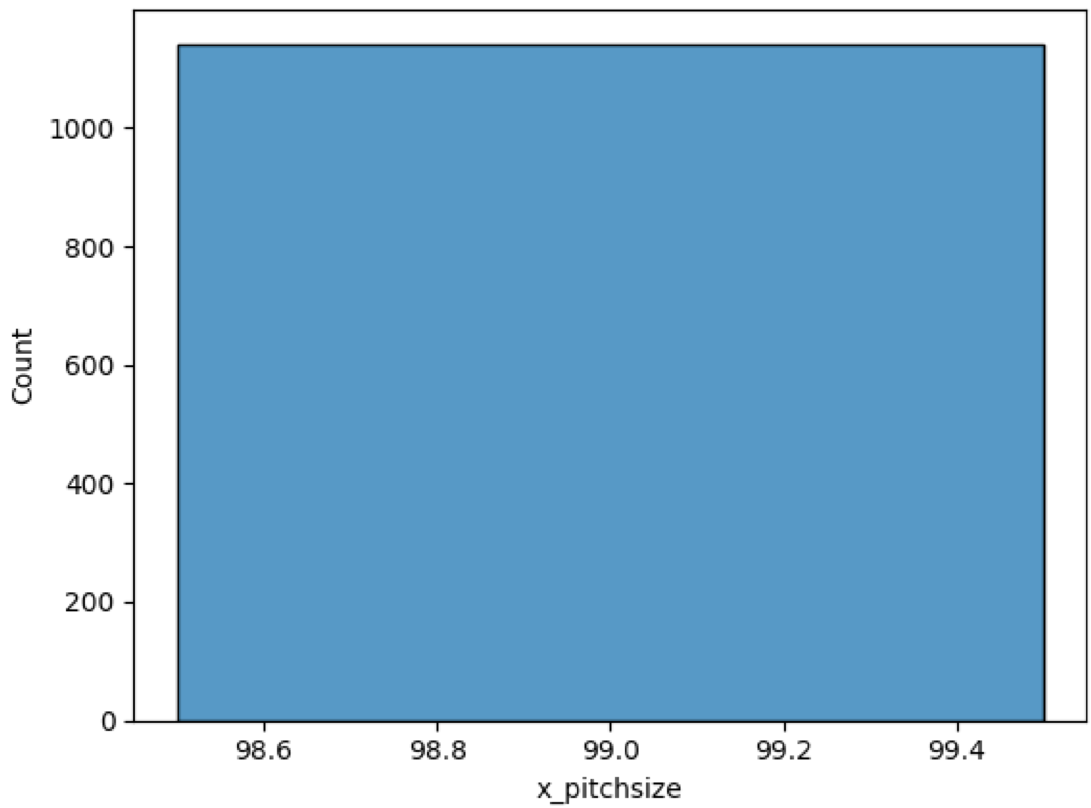

120
100
80
60
40
20
0
Count
−4
−2
0
2
4
Pass length

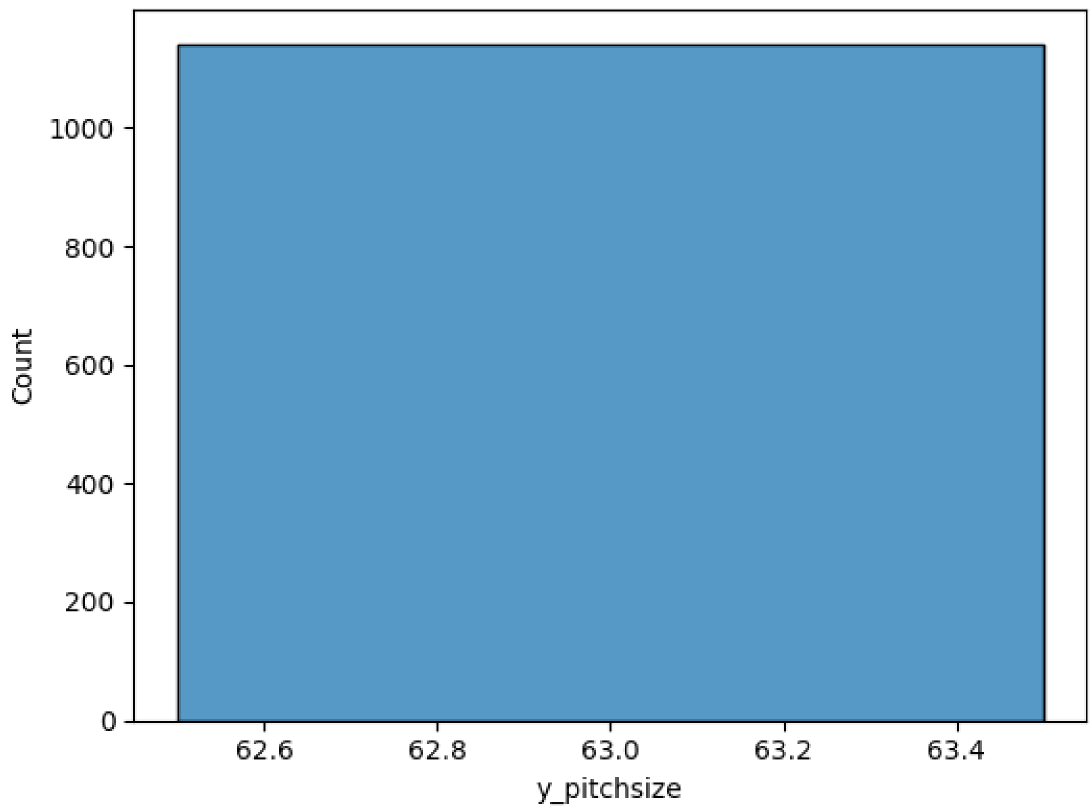


600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Playing direction_first half

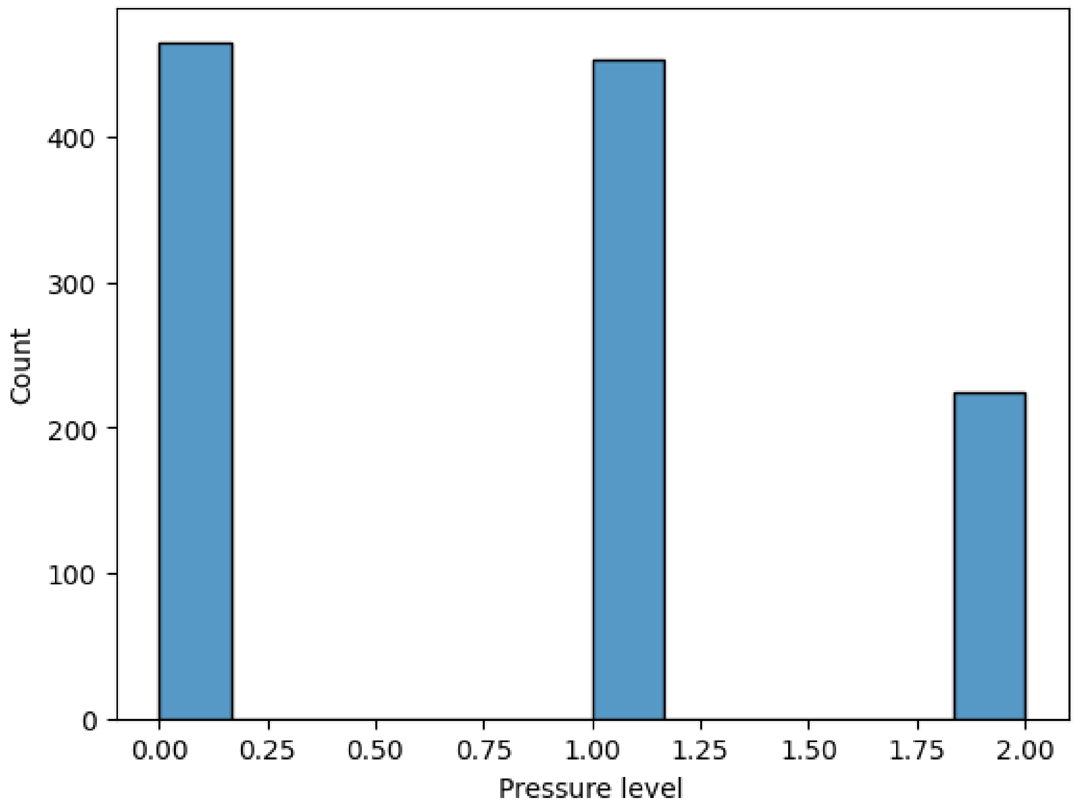


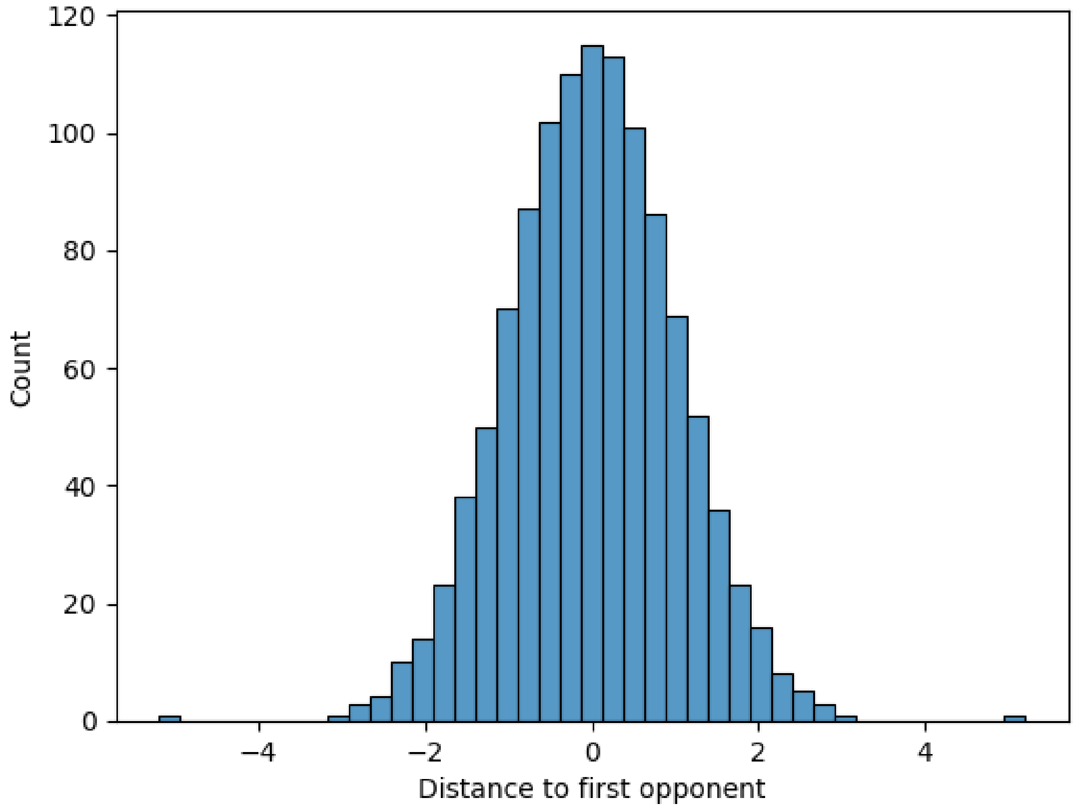
600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Playing direction_second half


Count
1000
800
600
400
200
0
98.6
98.8
99.0
99.2
99.4
x_pitchsize

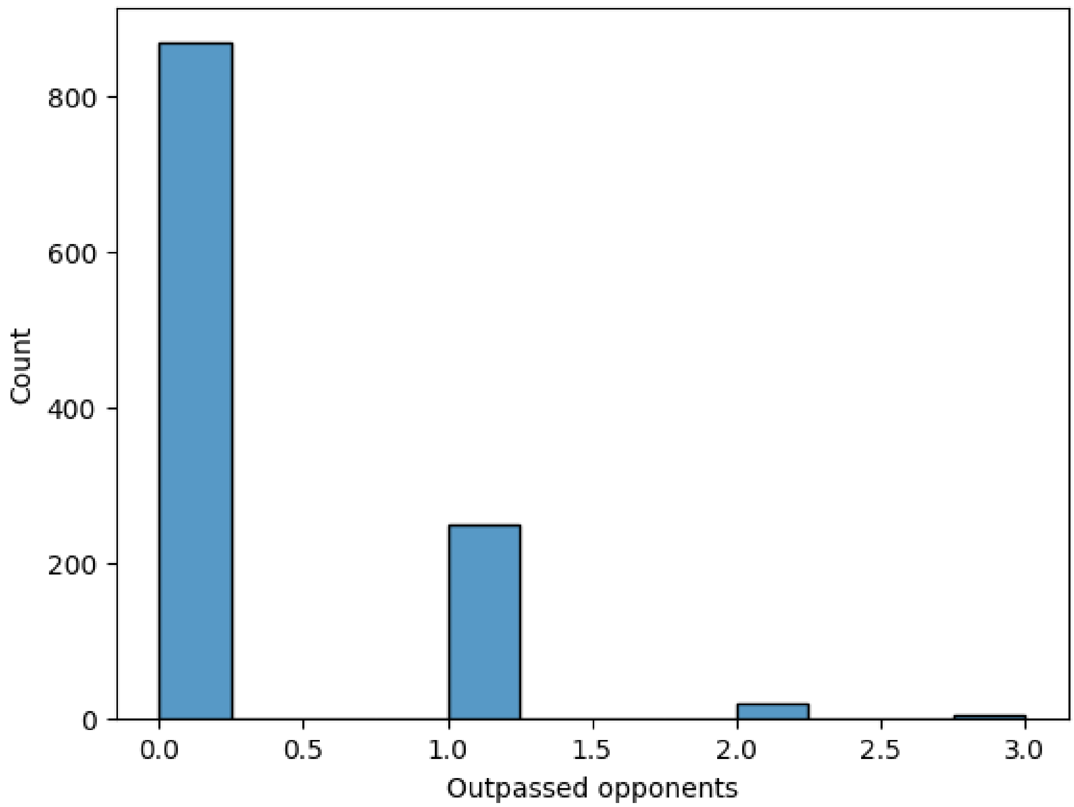


1000
800
600
400
200
0
Count
62.6
62.8
63.0
63.2
63.4
y_pitchsize

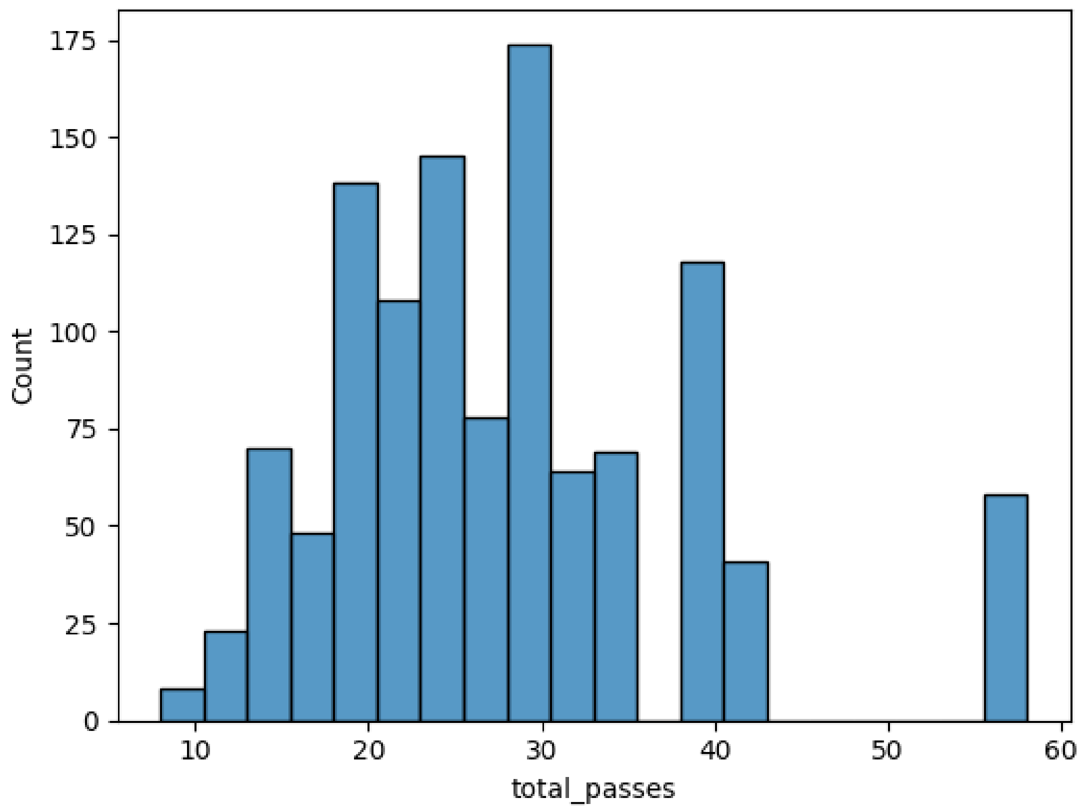

400
300
200
100
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Pressure level



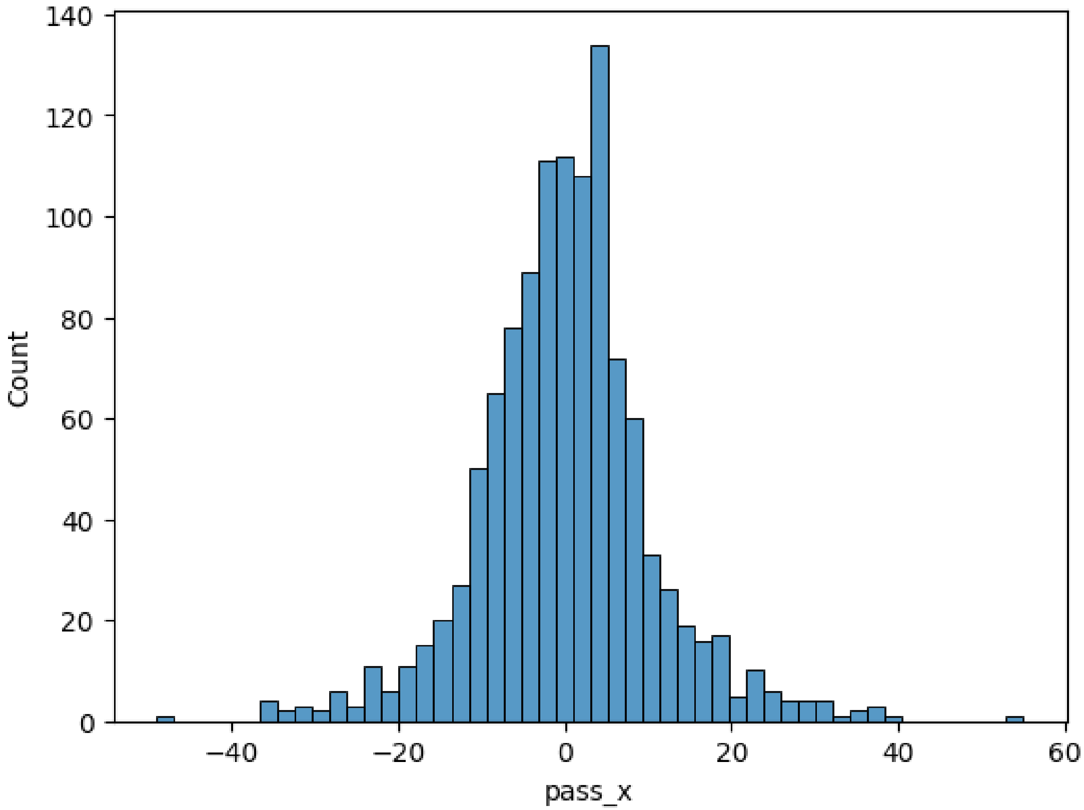
120
100
80
60
40
20
0
Count
−4
−2
0
2
4
Distance to first opponent

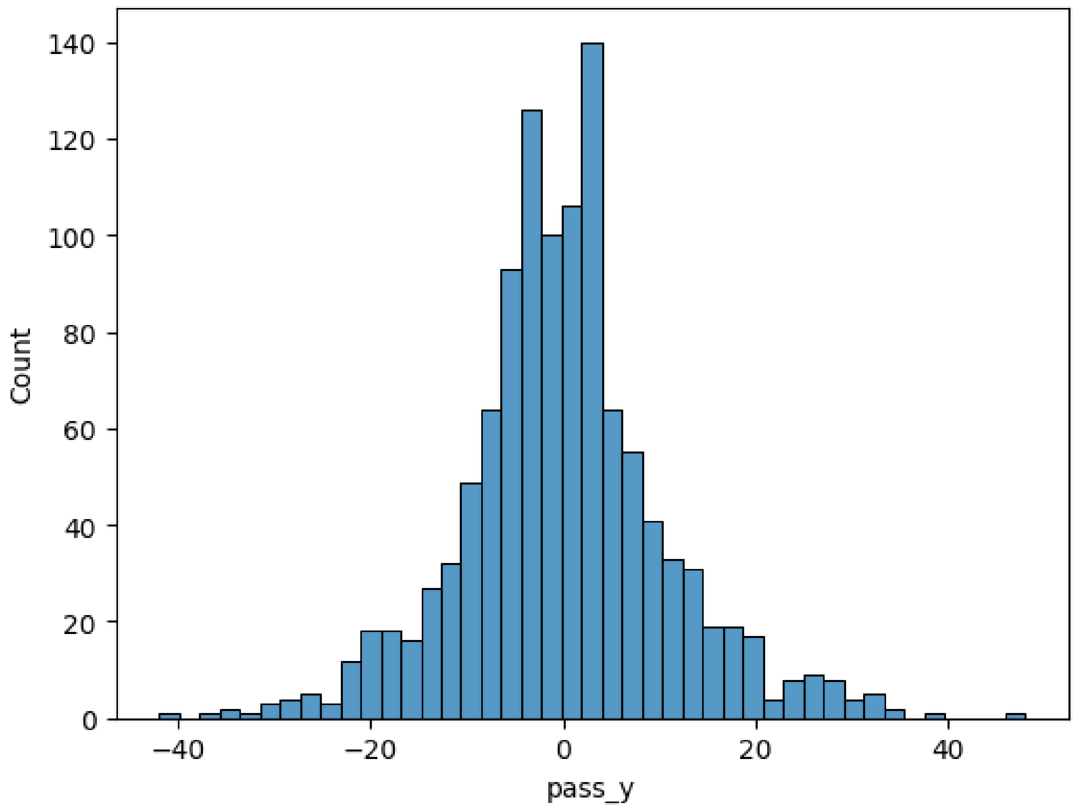


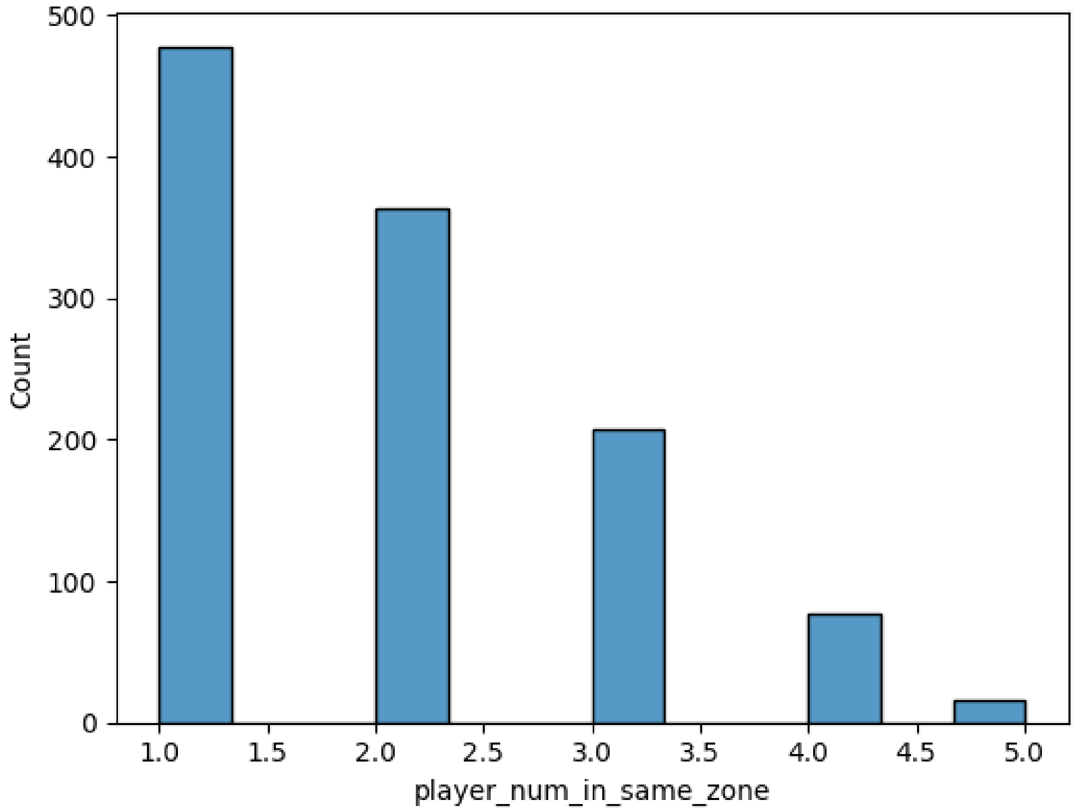
800
600
400
200
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Outpassed opponents



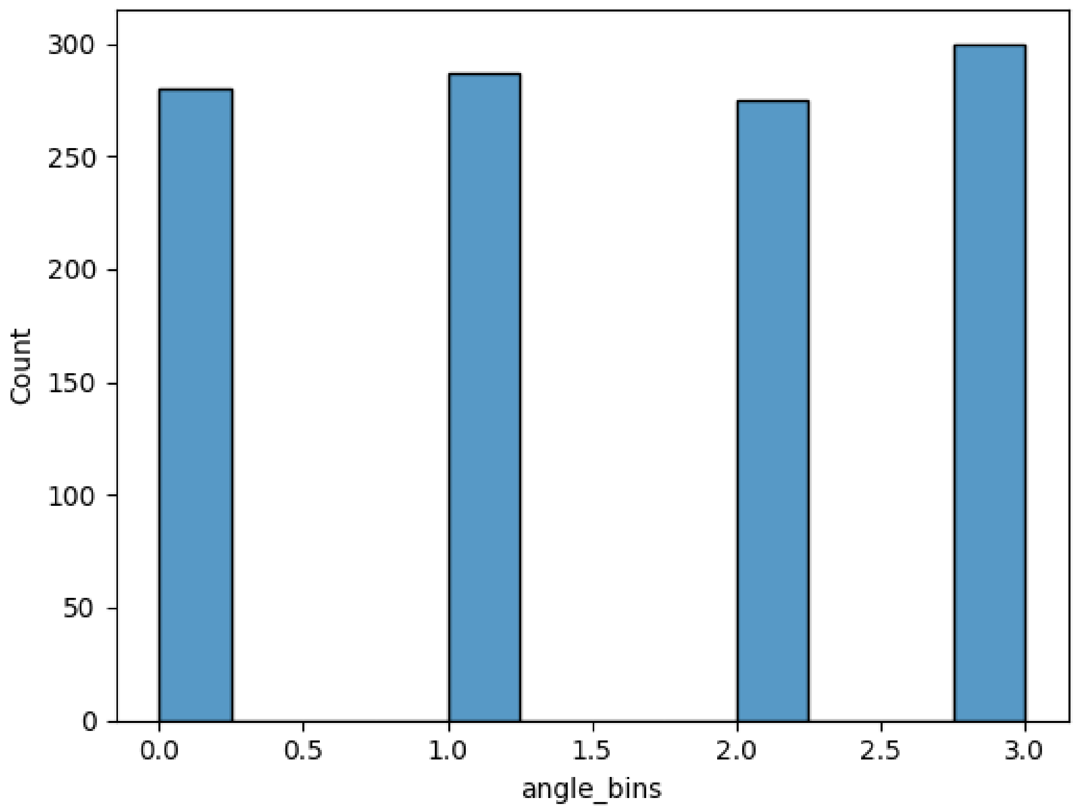
175
150
125
100
75
50
25
0
Count
10
20
30
40
50
60
total_passes


140
120
100
80
60
40
20
0
Count
−40
−20
0
20
40
60
pass_x

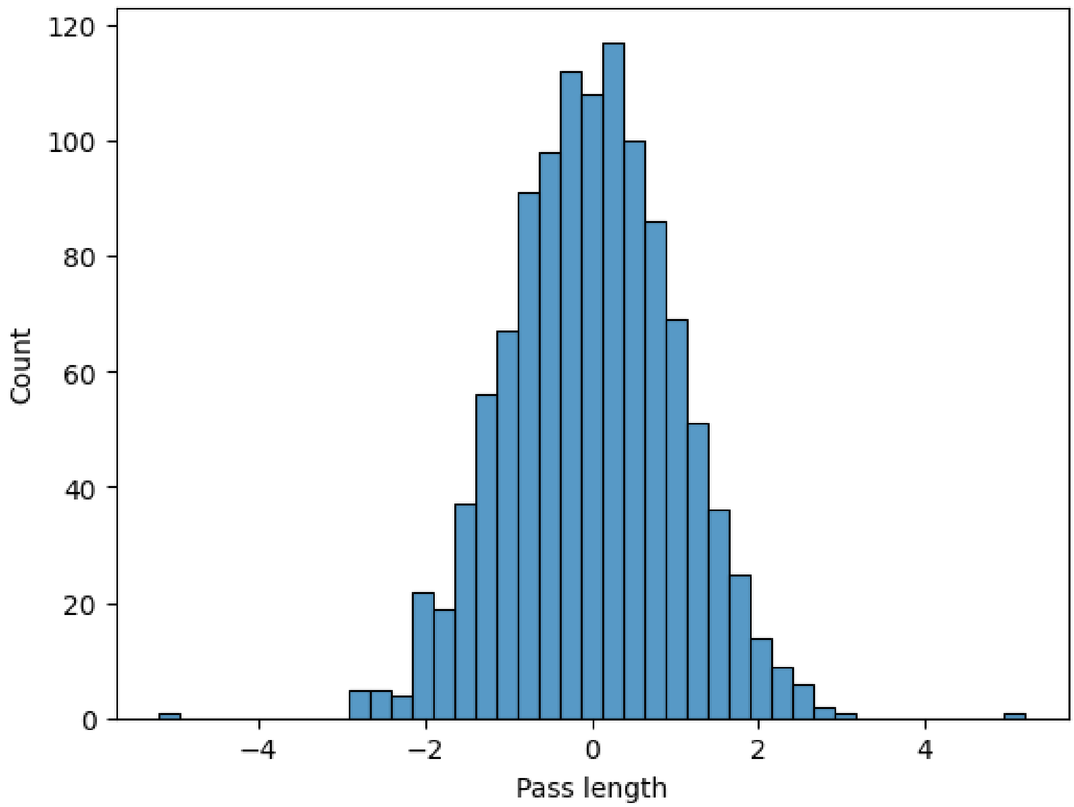

140
120
100
80
60
40
20
0
Count
−40
−20
0
20
40
pass_y

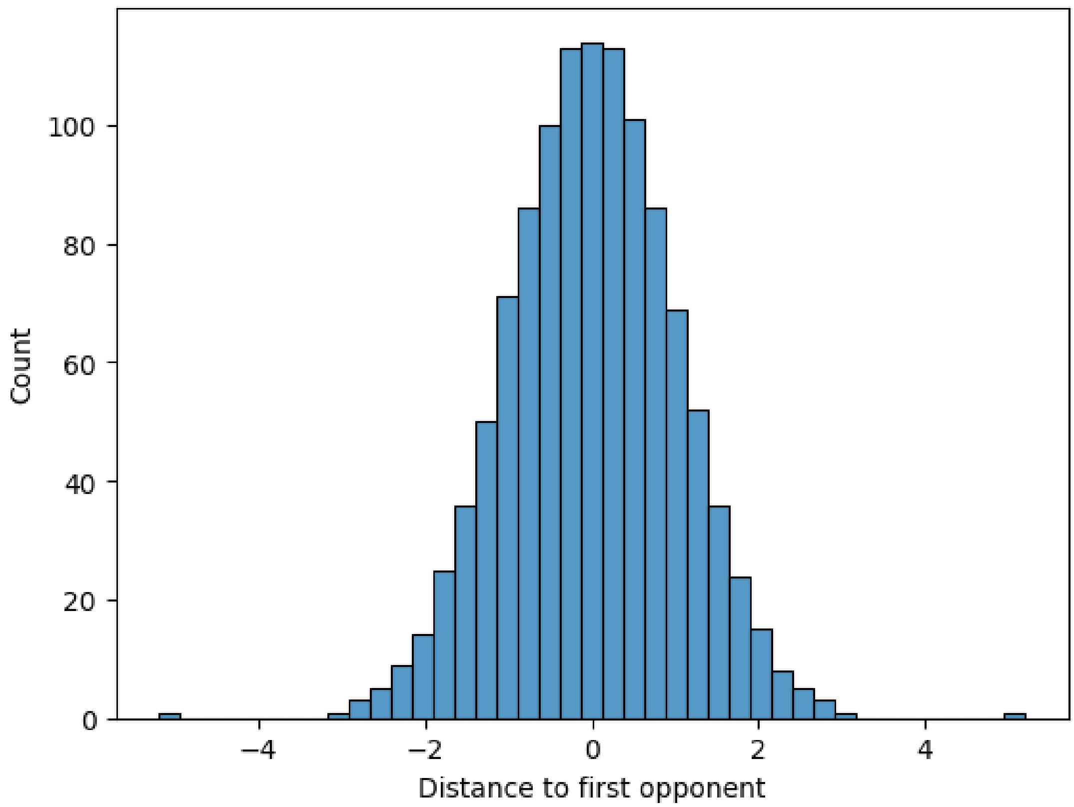


500
400
300
200
100
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
player_num_in_same_zone



Count
300
250
200
150
100
50
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
angle_bins


120
100
80
60
40
20
0
Count
−4
−2
0
2
4
Pass length




```
[ ]: # ---------------------------------------------------------- Data split
from sklearn.model_selection import train_test_split
df["receiverId"]=df["receiverId"].fillna(0) # a sign bit, in order to␣
 ↪distinguish
X = df.loc[:, ~df.columns.isin(["isSucceeded"])]
y = df["isSucceeded"]
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪stratify=df[['Player_id','isSucceeded']]) # former proven history variety,␣
 ↪later proven target variety
df_train=pd.concat([X_train, y_train], axis=1)
df_test= pd.concat([X_test, y_test], axis=1)
# **********************************************************
```

```
[ ]: # ---------------------------------------------------------- Add features
# Note: the code segment is duplicate-> for future improvement !!!
# count pair (receiverId and playId)
def added_X_train_test_pair(df_train,X_train,X_test):
    data_dict=df_train.groupby(['receiverId','Player_id']).
 ↪sum('isSucceeded')['isSucceeded'].to_dict()

    def use_lambda(df):
```

```
        import math
        if (df["receiverId"],df['Player_id']) in data_dict.keys() and␣
↪df["receiverId"]!=0:
            return data_dict[(df["receiverId"],df['Player_id'])]
        return 0

    def add_pair_rate_train(df):
        df['pair_count']=df.apply(use_lambda,axis=1)
        return df

    def add_pair_rate_test(df):
        df['pair_count']=df.apply(use_lambda,axis=1)
        return df

    return add_pair_rate_train(X_train),add_pair_rate_test(X_test)

X_train,X_test=added_X_train_test_pair(df_train,X_train,X_test)
# X_test.pair_count.sum() # result is more than 100, so a good idea at least

def add_count_player_id(X_train,X_test):
    pass_dict=df_train.groupby(['Player_id']).count()['posX_passer'].to_dict()
    def add_success_count_train(df): # note player_id and success_rate is one␣
↪vs one, have no information, but will have information on the test data!!!
        df['succeed_count']=df['Player_id'].map(lambda x:pass_dict[x])
        return df
    def use_lambda(x):
        import math
        if x in pass_dict.keys():
            return pass_dict[x]
        return np.mean(list(pass_dict.values()))
    def add_success_rate_test(df):
        df['succeed_count']=df['Player_id'].map(lambda x:use_lambda(x))
        return df
    return add_success_count_train(X_train),add_success_rate_test(X_test)

X_train,X_test=add_count_player_id(X_train,X_test)

def add_success_rate(df_train,X_train,X_test):
    data_dict=df_train.groupby(['Player_id']).
↪mean('isSucceeded')['isSucceeded'].to_dict()

    def add_success_rate_train(df): # note player_id and success_rate is one vs␣
↪one, have no information, but will have information on the test data!!!
        df['succeed_rate']=df['Player_id'].map(lambda x:data_dict[x])
        return df

    def use_lambda(x):
```

```
        import math
        if x in data_dict.keys():
            return data_dict[x]
        return np.mean(list(data_dict.values()))
    def add_success_rate_test(df):
        df['succeed_rate']=df['Player_id'].map(lambda x:use_lambda(x))
        return df
    return add_success_rate_train(X_train),add_success_rate_test(X_test)
X_train,X_test=add_success_rate(df_train,X_train,X_test)

def add_zone_rate(df_train,X_train,X_test):
    # generate feature based on received_id and also zone
    zone_dict=df_train.groupby(['Player_id','Zone']).
 ↪mean('isSucceeded')['isSucceeded'].to_dict()
    # df['zone_rate']=df[['Player_id','Zone']].map(lambda x:
 ↪zone_dict[(x[0],x[1])]) this can't work, map can only used in series, use␣
 ↪apply instead
    def use_zone_lambda_train(df):
        return zone_dict[(df["Player_id"],df['Zone'])]
    def add_zone_rate_train(df): # note player_id and success_rate is one vs␣
 ↪one, have no information, but will have information on the test data!!!
        df['zone_rate']=df[['Player_id','Zone']].
 ↪apply(use_zone_lambda_train,axis=1)
        return df

    def use_zone_lambda_test(df):
        import math
        if (df["Player_id"],df['Zone']) in zone_dict.keys():
            return zone_dict[(df["Player_id"],df['Zone'])]
        else:
            sub_dict = {key: value for key, value in zone_dict.items() if␣
 ↪key[1]== df['Zone']}
            return np.mean(list(sub_dict.values()))
    def add_zone_rate_test(df):
        df['zone_rate']=df[['Player_id','Zone']].
 ↪apply(use_zone_lambda_test,axis=1)
        return df
    return add_zone_rate_train(X_train),add_zone_rate_test(X_test)
X_train,X_test=add_zone_rate(df_train,X_train,X_test)

def add_pass_type_rate(df_train,X_train,X_test):
    pass_type_dict=df_train.groupby(['Player_id','Pass type']).
 ↪mean('isSucceeded')['isSucceeded'].to_dict()

    def use_pass_type_lambda_train(df):
        return pass_type_dict[(df["Player_id"],df['Pass type'])]
```

```
    def add_pass_type_rate_train(df):
        df['pass_type_rate']=df[['Player_id','Pass type']].
 ↪apply(use_pass_type_lambda_train,axis=1)
        return df

    def use_pass_type_lambda_test(df):
        import math
        if (df["Player_id"],df['Pass type']) in pass_type_dict.keys():
            return pass_type_dict[(df["Player_id"],df['Pass type'])]
        else:
            sub_dict = {key: value for key, value in pass_type_dict.items() if␣
 ↪key[1]== df['Pass type']}
            return np.mean(list(sub_dict.values()))

    def add_pass_type_rate_test(df):
        df['pass_type_rate']=df[['Player_id','Pass type']].
 ↪apply(use_pass_type_lambda_test,axis=1)
        return df
    return add_pass_type_rate_train(X_train),add_pass_type_rate_test(X_test)

X_train,X_test=add_pass_type_rate(df_train,X_train,X_test)

def add_pressure_level_rate(df_train,X_train,X_test):
    pressure_level_dict=df_train.groupby(['Player_id','Pressure level']).
 ↪mean('isSucceeded')['isSucceeded'].to_dict()

    def use_pressure_level_lambda_train(df):
        return pressure_level_dict[(df["Player_id"],df['Pressure level'])]

    def add_pressure_level_rate_train(df):
        df['pressure_level_rate']=df[['Player_id','Pressure level']].
 ↪apply(use_pressure_level_lambda_train,axis=1)
        return df

    def use_pressure_level_lambda_test(df):
        import math
        if (df["Player_id"],df['Pressure level']) in pressure_level_dict.keys():
            return pressure_level_dict[(df["Player_id"],df['Pressure level'])]
        else:
            sub_dict = {key: value for key, value in pressure_level_dict.
 ↪items() if key[1]== df['Pressure level']}
            return np.mean(list(sub_dict.values()))

    def add_pressure_level_rate_test(df):
```

```
        df['pressure_level_rate']=df[['Player_id','Pressure level']].
 ↪apply(use_pressure_level_lambda_test,axis=1)
        return df
    return␣
 ↪add_pressure_level_rate_train(X_train),add_pressure_level_rate_test(X_test)

X_train,X_test=add_pressure_level_rate(df_train,X_train,X_test)

# ********************************************************
```

```
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:1
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pair_count']=df.apply(use_lambda,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:1
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pair_count']=df.apply(use_lambda,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:2
9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['succeed_count']=df['Player_id'].map(lambda x:pass_dict[x])
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:3
7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['succeed_count']=df['Player_id'].map(lambda x:use_lambda(x))
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:4
7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['succeed_rate']=df['Player_id'].map(lambda x:data_dict[x])
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:5
6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['succeed_rate']=df['Player_id'].map(lambda x:use_lambda(x))
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:6
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['zone_rate']=df[['Player_id','Zone']].apply(use_zone_lambda_train,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:7
9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['zone_rate']=df[['Player_id','Zone']].apply(use_zone_lambda_test,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:9
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pass_type_rate']=df[['Player_id','Pass
type']].apply(use_pass_type_lambda_train,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:1
03: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pass_type_rate']=df[['Player_id','Pass
type']].apply(use_pass_type_lambda_test,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:1
16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pressure_level_rate']=df[['Player_id','Pressure
level']].apply(use_pressure_level_lambda_train,axis=1)
/var/folders/ws/p_92kz8j46lgdkjkzzrm9wm40000gn/T/ipykernel_23298/1032941593.py:1
28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['pressure_level_rate']=df[['Player_id','Pressure
level']].apply(use_pressure_level_lambda_test,axis=1)
```

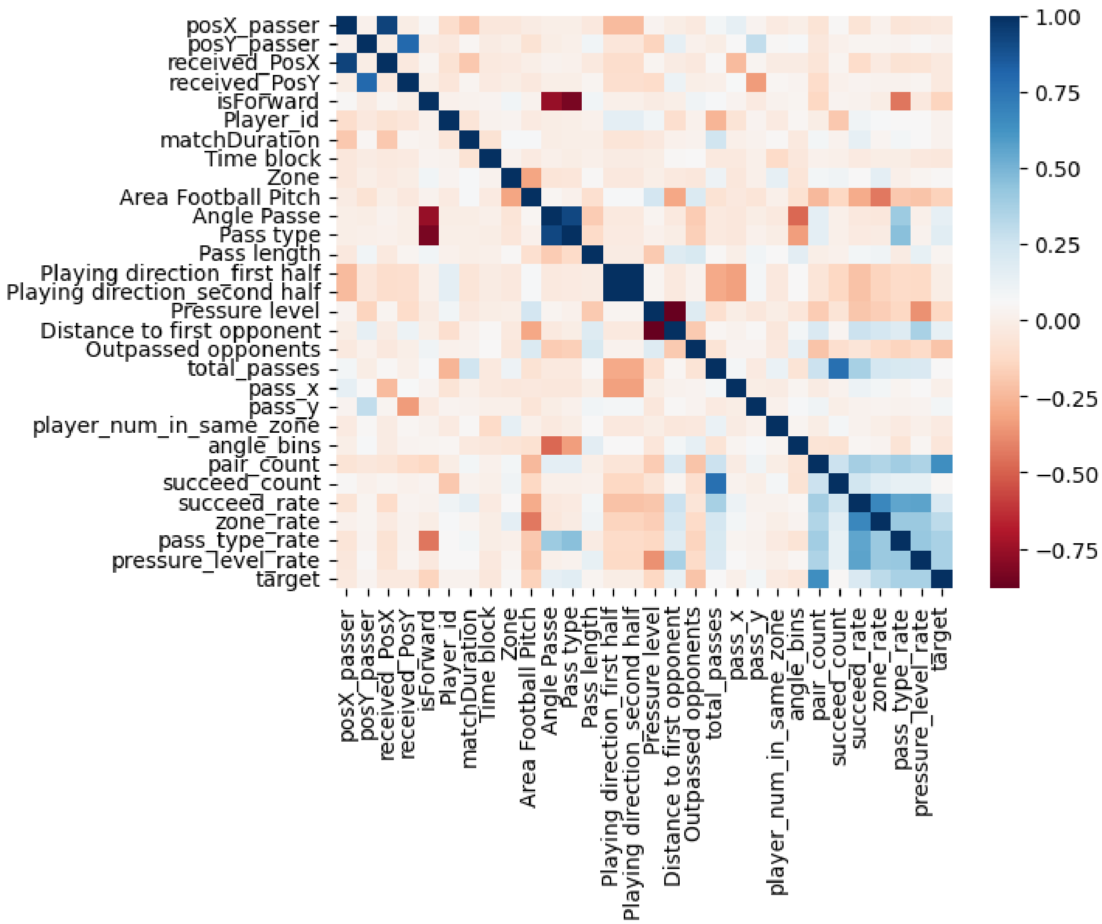
[ ]:
```
import copy
X_train_b=copy.deepcopy(X_train)
X_test_b=copy.deepcopy(X_test)
```

[ ]:
```
X_train=copy.deepcopy(X_train_b)
X_test=copy.deepcopy(X_test_b)
```

[ ]:
```
# --------------------------------------------------------- Drop uninformative␣
↪columns
def drop_columns(df,lst_columns):
    return df.drop(columns=lst_columns,axis=1)
def drop_columns_with_one_unique_value(df):
    return drop_columns(df,get_columns_with_one_unique_value(df))

def get_columns_with_one_unique_value(df):
    col_counts = df.nunique()
    cols_with_one_unique_value = col_counts[col_counts == 1]
    return list(cols_with_one_unique_value.index)

lst_delete=['TimeStamp','startTime','Team','receiverId']
lst_one_value=get_columns_with_one_unique_value(df)
X_train=drop_columns(X_train,lst_delete+lst_one_value)
X_test=drop_columns(X_test,lst_delete+lst_one_value)
print("lst_one_value:")
print(lst_one_value)

# *********************************************************
```

```
lst_one_value:
['Type', 'Club', 'x_pitchsize', 'y_pitchsize']
```

```
[ ]: # ---------------------------------------------------------- Show added features are informative
def plot_corr(X_train,y_train):
    import seaborn as sns
    import copy
    df_plot=copy.deepcopy(X_train)
    df_plot['target'] = copy.deepcopy(y_train)
    corr = df_plot.corr()
    sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap="RdBu")
plot_corr(X_train,y_train)

# conclusion: pair_count is informative
# **********************************************************
```




```
[ ]: # ---------------------------------------------------------- model training
from sklearn import metrics
def get_dummy_score(X_train, X_test, y_train, y_test):
```

```
    from sklearn.dummy import DummyClassifier


    # Create an instance of the DummyClassifier class
    dummy_classifier = DummyClassifier(strategy='most_frequent')

    # Fit the DummyClassifier instance to the training data
    dummy_classifier.fit(X_train, y_train)

    y_pred=dummy_classifier.predict(X_test)
    f1 = metrics.f1_score(y_test, y_pred)
    acc = metrics.accuracy_score(y_test, y_pred)
    prec = metrics.precision_score(y_test, y_pred)
    print("F1 Score:", f1)
    print("Accuracy:", acc)
    print("Precision:", prec)
get_dummy_score(X_train, X_test, y_train, y_test)

# *********************************************************
```

```
F1 Score: 0.8397565922920892
Accuracy: 0.7237762237762237
Precision: 0.7237762237762237
```

[ ]:
```
# Import the necessary libraries and models
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
↪GradientBoostingClassifier, BaggingClassifier, ExtraTreesClassifier
from xgboost import XGBClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

# insight: classifiers are just above the dummy model or even above, try to␣
↪extract more features

# Define a dictionary of classification models
models = {
    "logistic_regression": LogisticRegression(),
    "support_vector_machine": SVC(),
    "k_nearest_neighbors": KNeighborsClassifier(),
    "decision_tree": DecisionTreeClassifier(),
    "random_forest": RandomForestClassifier(),
    "ada_boost": AdaBoostClassifier(),
```

```
    "gradient_boosting": GradientBoostingClassifier(),
    "xg_boost": XGBClassifier(),
    "bagging": BaggingClassifier(),
    "extra_trees": ExtraTreesClassifier(),
    "mlp": MLPClassifier(),
    "gaussian_process": GaussianProcessClassifier(),
    "quadratic_discriminant_analysis": QuadraticDiscriminantAnalysis()
}
```

[ ]:
```
from sklearn import metrics
def evaluate_models(model,name, X_train, X_test, y_train, y_test ):
  # Loop through each model
  # Fit the model and make predictions
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)

  # Calculate and print the metrics
  f1 = metrics.f1_score(y_test, y_pred)
  acc = metrics.accuracy_score(y_test, y_pred)
  prec = metrics.precision_score(y_test, y_pred)
  print(name)
  print("F1 Score:", f1)
  print("Accuracy:", acc)
  print("Precision:", prec)
```

[ ]:
```
# Loop through the models and evaluate each one
for name, model in models.items():
    evaluate_models(model,name,X_train, X_test, y_train, y_test)

# note: majority of models, the score is bad than dummy model, which means␣
 ↪there are so many noise.
```

```
logistic_regression
F1 Score: 0.8397565922920892
Accuracy: 0.7237762237762237
Precision: 0.7237762237762237
support_vector_machine
F1 Score: 0.8397565922920892
Accuracy: 0.7237762237762237
Precision: 0.7237762237762237
k_nearest_neighbors
F1 Score: 0.8565121412803531
Accuracy: 0.7727272727272727
Precision: 0.7886178861788617
decision_tree
F1 Score: 0.896
Accuracy: 0.8636363636363636
```

```
Precision: 1.0
random_forest
F1 Score: 0.896
Accuracy: 0.8636363636363636
Precision: 1.0
ada_boost
F1 Score: 0.896
Accuracy: 0.8636363636363636
Precision: 1.0
gradient_boosting
F1 Score: 0.896
Accuracy: 0.8636363636363636
Precision: 1.0
xg_boost
F1 Score: 0.896
Accuracy: 0.8636363636363636
Precision: 1.0
bagging
F1 Score: 0.896
Accuracy: 0.8636363636363636
Precision: 1.0
extra_trees
F1 Score: 0.9028871391076115
Accuracy: 0.8706293706293706
Precision: 0.9885057471264368
mlp
F1 Score: 0.8397565922920892
Accuracy: 0.7237762237762237
Precision: 0.7237762237762237
gaussian_process
F1 Score: 0.7951219512195122
Accuracy: 0.7062937062937062
Precision: 0.8029556650246306
quadratic_discriminant_analysis
F1 Score: 0.8877284595300261
Accuracy: 0.8496503496503497
Precision: 0.9659090909090909
```

```
/Users/horus_liang/opt/anaconda3/envs/intro_to_ds/lib/python3.8/site-
packages/sklearn/discriminant_analysis.py:808: UserWarning: Variables are
collinear
  warnings.warn("Variables are collinear")
```

[ ]:
```
# Define the parameter grid
param_grid = {
    'n_estimators': [600,610,620,590],
    'max_depth': [40,50],
}
```

```python
from sklearn.model_selection import RandomizedSearchCV

# Create an instance of the Random Forest classifier
classifier = ExtraTreesClassifier()

# Create an instance of the RandomizedSearchCV class
random_search = RandomizedSearchCV(classifier, param_grid,cv=5,scoring='f1')

# Fit the RandomizedSearchCV to the data
random_search.fit(X_train, y_train)

print(random_search.best_params_)

classifier = random_search.best_estimator_
y_pred=classifier.predict(X_test)
# Calculate and print the metrics
f1 = metrics.f1_score(y_test, y_pred)
acc = metrics.accuracy_score(y_test, y_pred)
prec = metrics.precision_score(y_test, y_pred)
print("F1 Score:", f1)
print("Accuracy:", acc)
print("Precision:", prec)

# !!! conclusion: this kind of method is not correct, cv=ts_split, the best␣
↪estimator would be the one tuning the parameter based on validation␣
↪dataset,leading to overfitting, not generalization.
# ->wrong
# so note: maybe the model is too complex ?->discard this model->done
# ->wrong
# even through cv has test dataset, but still can overfit, becase overfit is␣
↪the situation when strong model and less test score, less test data
```

```
/Users/horus_liang/opt/anaconda3/envs/intro_to_ds/lib/python3.8/site-
packages/sklearn/model_selection/_search.py:285: UserWarning: The total space of
parameters 8 is smaller than n_iter=10. Running 8 iterations. For exhaustive
searches, use GridSearchCV.
  warnings.warn(
```

```
{'n_estimators': 620, 'max_depth': 50}
F1 Score: 0.8970976253298152
Accuracy: 0.8636363636363636
Precision: 0.9883720930232558
```

[ ]:

```python

```