leagueanalysis

June 19, 2021

1 Loading the data

[1]:
```
import warnings
warnings.filterwarnings('ignore')
'''since we have multiple datasets we use this function to load them all'''
def importAll(path):
    all_files = glob.glob(path + "/*.csv")
    res = {}

    for filename in all_files:
        print('Importing:', filename)
        df = pd.read_csv(filename, index_col=None, header=0)
        res[filename] = df
    return res

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 40)
```

[2]:
```
competitive = importAll('dataset')
'''These are taken from the kaggle competiive LoL dataset, linked here:
https://www.kaggle.com/chuckephron/leagueoflegends?select=LeagueofLegends.csv'''
```

```
Importing: 1076/gold.csv
Importing: 1076/matchinfo.csv
Importing: 1076/bans.csv
Importing: 1076/structures.csv
Importing: 1076/kills.csv
Importing: 1076/monsters.csv
Importing: 1076/LeagueofLegends.csv
Importing: 1076/_columns.csv
```

[2]:
```
'These are taken from the kaggle competiive LoL dataset, linked here:\nhttps://w
ww.kaggle.com/chuckephron/leagueoflegends?select=LeagueofLegends.csv'
```

2 Taking an inital look at the data

[3]:
```
competitive_datasets = list(competitive.keys())
competitive_datasets
#These are the datasets we have to work with...but the gold, kills, monsters,␣
 ↪and structures datasets are all just subsets of LeagueofLegends dataset.
```

[3]:
```
['1076/gold.csv',
 '1076/matchinfo.csv',
 '1076/bans.csv',
 '1076/structures.csv',
 '1076/kills.csv',
 '1076/monsters.csv',
 '1076/LeagueofLegends.csv',
 '1076/_columns.csv']
```

[4]:
```
matchdata = competitive['1076/LeagueofLegends.csv']
print(matchdata.columns)
# thse are our indendepent variables, which we'll use for prediction later
matchdata.describe()
#Something interesting right away....blue side wins 5% more than red side!
```

```
Index(['League', 'Year', 'Season', 'Type', 'blueTeamTag', 'bResult', 'rResult',
       'redTeamTag', 'gamelength', 'golddiff', 'goldblue', 'bKills', 'bTowers',
       'bInhibs', 'bDragons', 'bBarons', 'bHeralds', 'goldred', 'rKills',
       'rTowers', 'rInhibs', 'rDragons', 'rBarons', 'rHeralds', 'blueTop',
       'blueTopChamp', 'goldblueTop', 'blueJungle', 'blueJungleChamp',
       'goldblueJungle', 'blueMiddle', 'blueMiddleChamp', 'goldblueMiddle',
       'blueADC', 'blueADCChamp', 'goldblueADC', 'blueSupport',
       'blueSupportChamp', 'goldblueSupport', 'blueBans', 'redTop',
       'redTopChamp', 'goldredTop', 'redJungle', 'redJungleChamp',
       'goldredJungle', 'redMiddle', 'redMiddleChamp', 'goldredMiddle',
       'redADC', 'redADCChamp', 'goldredADC', 'redSupport', 'redSupportChamp',
       'goldredSupport', 'redBans', 'Address'],
      dtype='object')
```

[4]:

	Year	bResult	rResult	gamelength
count	7620.000000	7620.000000	7620.000000	7620.000000
mean	2016.280971	0.544094	0.455906	37.012598
std	0.848200	0.498085	0.498085	7.983238
min	2014.000000	0.000000	0.000000	17.000000
25%	2016.000000	0.000000	0.000000	31.000000
50%	2016.000000	1.000000	0.000000	36.000000
75%	2017.000000	1.000000	1.000000	41.000000
max	2018.000000	1.000000	1.000000	95.000000

[5]:
```
wins = competitive['1076/matchinfo.csv']
#these are our dependent variables: namely
#League, Year, Season, and Type, and Win/Lose. We disregard all  other columns␣
 ↪for our analysis.
wins.head()
```

[5]:

	League	Year	Season	Type	blueTeamTag	bResult	rResult	redTeamTag
0	NALCS	2015	Spring	Season	TSM	1	0	C9
1	NALCS	2015	Spring	Season	CST	0	1	DIG
2	NALCS	2015	Spring	Season	WFX	1	0	GV
3	NALCS	2015	Spring	Season	TIP	0	1	TL
4	NALCS	2015	Spring	Season	CLG	1	0	T8

	gamelength	blueTop	blueTopChamp	blueJungle	blueJungleChamp	blueMiddle
0	40	Dyrus	Irelia	Santorin	RekSai	Bjergsen
1	38	Cris	Gnar	Impaler	Rengar	Jesiz
2	40	Flaresz	Renekton	ShorterACE	Rengar	Pobelter
3	41	Rhux	Irelia	Rush	JarvanIV	XiaoWeiXiao
4	35	Benny	Gnar	Xmithie	JarvanIV	Link

	blueMiddleChamp	blueADC	blueADCChamp	blueSupport	blueSupportChamp
0	Ahri	WildTurtle	Jinx	Lustboy	Janna
1	Ahri	Mash	Caitlyn	Sheep	Leona
2	Fizz	Altec	Sivir	Gleeb	Annie
3	Leblanc	Apollo	Sivir	Adrian	Thresh
4	Lissandra	Doublelift	Tristana	aphromoo	Janna

	redTop	redTopChamp	redJungle	redJungleChamp	redMiddle
0	Balls	Gnar	Meteos	Elise	Hai
1	Gamsu	Irelia	Crumbzz	JarvanIV	Shiphtur
2	Hauntzer	Sion	Saintvicious	LeeSin	Keane
3	Quas	Gnar	IWDominate	Nunu	Fenix
4	CaliTrlolz8	Sion	Porpoise8	RekSai	Slooshi8

	redMiddleChamp	redADC	redADCChamp	redSupport	redSupportChamp
0	Fizz	Sneaky	Sivir	LemonNation	Thresh
1	Azir	CoreJJ	Corki	KiWiKiD	Annie
2	Azir	Cop	Corki	BunnyFuFuu	Janna
3	Lulu	KEITH	KogMaw	Xpecial	Janna
4	Lulu	Maplestreet8	Corki	Dodo8	Annie

	Address
0	http://matchhistory.na.leagueoflegends.com/en/…
1	http://matchhistory.na.leagueoflegends.com/en/…
2	http://matchhistory.na.leagueoflegends.com/en/…
3	http://matchhistory.na.leagueoflegends.com/en/…
4	http://matchhistory.na.leagueoflegends.com/en/…

[6]:
```
#we select only the columns we are interested in. We select gamelength as well␣
 ↪for use as an independent variable
wins = wins[['League', 'Year', 'Season', 'Type', 'bResult', 'rResult',␣
 ↪'gamelength']]
matchdata = matchdata[[ 'golddiff', 'goldblue', 'bKills', 'bTowers', 'bInhibs',␣
 ↪'bDragons',
       'bBarons', 'bHeralds', 'goldred', 'rKills', 'rTowers', 'rInhibs',
       'rDragons', 'rBarons', 'rHeralds', 'blueTop', 'blueTopChamp',
       'goldblueTop', 'blueJungle', 'blueJungleChamp', 'goldblueJungle',
       'blueMiddle', 'blueMiddleChamp', 'goldblueMiddle', 'blueADC',
       'blueADCChamp', 'goldblueADC', 'blueSupport', 'blueSupportChamp',
       'goldblueSupport', 'blueBans', 'redTop', 'redTopChamp', 'goldredTop',
       'redJungle', 'redJungleChamp', 'goldredJungle', 'redMiddle',
       'redMiddleChamp', 'goldredMiddle', 'redADC', 'redADCChamp',
       'goldredADC', 'redSupport', 'redSupportChamp', 'goldredSupport',
       'redBans']]

wins.shape[0] == matchdata.shape[0]
#weach row/sample in the wins dataset correponds to a row/sample in the␣
 ↪matchdata:
# this makes it extremely easy to combine thme,
# since we would like to combine them first for preprocessing thisataset has␣
 ↪one to one correspondence to matchdata
league = pd.concat([wins, matchdata], axis=1)
league.columns
```

[6]:
```
Index(['League', 'Year', 'Season', 'Type', 'bResult', 'rResult', 'gamelength',
       'golddiff', 'goldblue', 'bKills', 'bTowers', 'bInhibs', 'bDragons',
       'bBarons', 'bHeralds', 'goldred', 'rKills', 'rTowers', 'rInhibs',
       'rDragons', 'rBarons', 'rHeralds', 'blueTop', 'blueTopChamp',
       'goldblueTop', 'blueJungle', 'blueJungleChamp', 'goldblueJungle',
       'blueMiddle', 'blueMiddleChamp', 'goldblueMiddle', 'blueADC',
       'blueADCChamp', 'goldblueADC', 'blueSupport', 'blueSupportChamp',
       'goldblueSupport', 'blueBans', 'redTop', 'redTopChamp', 'goldredTop',
       'redJungle', 'redJungleChamp', 'goldredJungle', 'redMiddle',
       'redMiddleChamp', 'goldredMiddle', 'redADC', 'redADCChamp',
       'goldredADC', 'redSupport', 'redSupportChamp', 'goldredSupport',
       'redBans'],
      dtype='object')
```

3 Transforming and preprocessing the data

3.1 (no cleaning necessary, data was preprocessed beforehand by Kaggle contributor)

[7]:
```
league.head()
```

[7]:

	League	Year	Season	Type	bResult	rResult	gamelength	golddiff	goldblue	bKills	bTowers	bInhibs
0	NALCS	2015	Spring	Season	1	0	40	[0, 0, -14, -65, -268, -431, -488, -789, -494,…	[2415, 2415, 2711, 3887, 5068, 6171, 7412, 866…	[[10.82, 'C9 Hai', 'TSM Bjergsen', [], 9229, 8…	[[27.542, 'MID_LANE', 'BASE_TURRET'], [39.269,…	[[36.686, 'MID_LANE'], [29.274, 'MID_LANE']]
1	NALCS	2015	Spring	Season	0	1	38	[0, 0, -26, -18, 147, 237, -152, 18, 88, -242,…	[2415, 2415, 2705, 4108, 5511, 6797, 7637, 895…	[[11.104, 'DIG Shiphtur', 'CST Jesiz', ['CST I…	[[23.239, 'BOT_LANE', 'OUTER_TURRET'], [33.018…	[]
2	NALCS	2015	Spring	Season	1	0	40	[0, 0, 10, -60, 34, 37, 589, 1064, 1258, 913, …	[2415, 2415, 2726, 3794, 4933, 6236, 8109, 965…	[[5.255, 'GV Keane', 'WFX Pobelter', ['WFX Sho…	[[15.045, 'BOT_LANE', 'OUTER_TURRET'], [39.566…	[[37.511, 'TOP_LANE'], [37.38, 'BOT_LANE'], [3…
3	NALCS	2015	Spring	Season	0	1	41	[0, 0, -15, 25, 228, -6, -243, 175, -346, 16, …	[2415, 2415, 2705, 3847, 5398, 6473, 7720, 930…	[[8.274, 'TL Quas', 'TIP Rhux', ['TIP Apollo']…	[[19.941, 'BOT_LANE', 'OUTER_TURRET'], [38.77,…	[]
4	NALCS	2015	Spring	Season	1	0	35	[40, 40, 44, -36, 113, 158, -121, -191, 23, 20…	[2415, 2415, 2710, 3950, 5404, 6666, 7887, 913…	[[11.438, 'T8 Dodo8', 'CLG Doublelift', ['CLG …	[[22.594, 'MID_LANE', 'OUTER_TURRET'], [34.213…	

```
4                              [[34.069, 'BOT_LANE']]

                                                bDragons     bBarons bHeralds  \
0                                     [[37.267, None]]           []       []
1   [[32.545, None], [26.177, None], [19.119, None]]  [[29.255]]       []
2  [[24.577, None], [37.867, None], [30.87, None]…  [[35.144]]       []
3                                                    []  [[37.513]]       []
4                  [[14.589, None], [30.307, None]]  [[32.556]]       []

                                               goldred  \
0  [2415, 2415, 2725, 3952, 5336, 6602, 7900, 945…
1  [2415, 2415, 2731, 4126, 5364, 6560, 7789, 893…
2  [2415, 2415, 2716, 3854, 4899, 6199, 7520, 859…
3  [2415, 2415, 2720, 3822, 5170, 6479, 7963, 913…
4  [2375, 2375, 2666, 3986, 5291, 6508, 8008, 932…

                                                rKills  \
0  [[16.529, 'TSM Lustboy', 'C9 Balls', ['C9 Mete…
1  [[12.387, 'CST Jesiz', 'DIG Gamsu', ['DIG Ship…
2  [[8.449, 'WFX Altec', 'GV Cop', ['GV BunnyFuFu…
3  [[7.768, 'TIP Rush', 'TL IWDominate', ['TL Fen…
4  [[11.988, 'CLG Doublelift', 'T8 Porpoise8', ['…

                                               rTowers                 rInhibs  \
0  [[39.23, 'TOP_LANE', 'INNER_TURRET'], [20.681,…                      []
1  [[19.257, 'MID_LANE', 'OUTER_TURRET'], [15.206…  [[36.813, 'MID_LANE']]
2  [[24.62, 'MID_LANE', 'OUTER_TURRET'], [30.493,…                      []
3  [[36.384, 'MID_LANE', 'NEXUS_TURRET'], [31.665…  [[35.867, 'MID_LANE']]
4  [[11.644, 'MID_LANE', 'OUTER_TURRET'], [12.438…                      []

                                               rDragons     rBarons rHeralds  \
0    [[17.14, None], [30.934, None], [24.641, None]]  [[29.954]]       []
1                                  [[12.264, None]]           []       []
2                                                []           []       []
3  [[26.274, None], [10.153, None], [18.515, None…           []       []
4                                  [[21.901, None]]           []       []

   blueTop blueTopChamp                                        goldblueTop  \
0    Dyrus       Irelia  [475, 475, 532, 687, 893, 1058, 1172, 1471, 18…
1     Cris         Gnar  [475, 475, 532, 791, 1127, 1509, 1674, 1875, 2…
2  Flaresz     Renekton  [475, 475, 533, 673, 828, 1075, 1428, 1775, 21…
3     Rhux       Irelia  [475, 475, 532, 646, 992, 1253, 1408, 1752, 21…
4    Benny         Gnar  [475, 475, 532, 733, 1038, 1258, 1546, 1850, 2…

   blueJungle blueJungleChamp  \
0    Santorin          RekSai
1     Impaler          Rengar
```

```
2  ShorterACE          Rengar
3        Rush        JarvanIV
4     Xmithie        JarvanIV

                                      goldblueJungle   blueMiddle  \
0  [475, 475, 532, 870, 1049, 1276, 1596, 1815, 2…     Bjergsen
1  [475, 475, 532, 895, 1176, 1334, 1447, 1859, 2…        Jesiz
2  [475, 475, 543, 836, 1041, 1261, 1568, 2002, 2…     Pobelter
3  [475, 475, 532, 909, 1272, 1387, 1705, 2009, 2…  XiaoWeiXiao
4  [475, 475, 532, 827, 1174, 1401, 1515, 1729, 2…         Link

  blueMiddleChamp                                    goldblueMiddle  \
0            Ahri  [475, 475, 532, 807, 1102, 1307, 1651, 1950, 2…
1            Ahri  [475, 475, 532, 816, 1102, 1413, 1624, 1937, 2…
2            Fizz  [475, 475, 533, 756, 1065, 1368, 2056, 2237, 2…
3         Leblanc  [475, 475, 532, 801, 1066, 1409, 1660, 2077, 2…
4       Lissandra  [475, 475, 532, 802, 1112, 1409, 1737, 1939, 2…

      blueADC blueADCChamp                                        goldblueADC  \
0  WildTurtle         Jinx  [475, 475, 532, 797, 1127, 1453, 1766, 2044, 2…
1        Mash      Caitlyn  [475, 475, 532, 856, 1182, 1468, 1653, 1890, 2…
2       Altec        Sivir  [475, 475, 533, 811, 1130, 1524, 1786, 2168, 2…
3      Apollo        Sivir  [475, 475, 532, 791, 1202, 1383, 1752, 2121, 2…
4  Doublelift     Tristana  [475, 475, 532, 857, 1203, 1554, 1883, 2254, 2…

  blueSupport blueSupportChamp  \
0     Lustboy            Janna
1       Sheep            Leona
2       Gleeb            Annie
3      Adrian           Thresh
4    aphromoo            Janna

                                    goldblueSupport  \
0  [515, 515, 583, 726, 897, 1077, 1227, 1381, 15…
1  [515, 515, 577, 750, 924, 1073, 1239, 1392, 15…
2  [515, 515, 584, 718, 869, 1008, 1271, 1474, 16…
3  [515, 515, 577, 700, 866, 1041, 1195, 1349, 14…
4  [515, 515, 582, 731, 877, 1044, 1206, 1359, 15…

                               blueBans       redTop redTopChamp  \
0    ['Rumble', 'Kassadin', 'Lissandra']        Balls        Gnar
1     ['Kassadin', 'Sivir', 'Lissandra']        Gamsu      Irelia
2  ['JarvanIV', 'Lissandra', 'Kassadin']     Hauntzer        Sion
3     ['Annie', 'Lissandra', 'Kassadin']         Quas        Gnar
4     ['Irelia', 'Pantheon', 'Kassadin']  CaliTrlolz8        Sion

                                        goldredTop     redJungle  \
```

```
0  [475, 475, 532, 728, 958, 1284, 1526, 1912, 21…        Meteos
1  [475, 475, 532, 811, 1042, 1237, 1625, 1852, 2…       Crumbzz
2  [475, 475, 533, 706, 861, 1123, 1491, 1698, 18…  Saintvicious
3  [475, 475, 532, 732, 922, 1309, 1678, 1992, 22…    IWDominate
4  [475, 475, 532, 820, 1025, 1290, 1598, 1953, 2…     Porpoise8

  redJungleChamp                                      goldredJungle redMiddle  \
0          Elise  [475, 475, 532, 898, 1192, 1429, 1819, 2107, 2…       Hai
1       JarvanIV  [475, 475, 532, 909, 1105, 1423, 1536, 1732, 1…  Shiphtur
2         LeeSin  [475, 475, 533, 845, 1089, 1443, 1694, 1914, 2…     Keane
3           Nunu  [475, 475, 541, 832, 1171, 1385, 1770, 2087, 2…     Fenix
4         RekSai  [475, 475, 532, 896, 1220, 1444, 1828, 2042, 2…  Slooshi8

  redMiddleChamp                                      goldredMiddle  \
0           Fizz  [475, 475, 552, 842, 1178, 1378, 1635, 1949, 2…
1           Azir  [475, 475, 552, 786, 1097, 1389, 1660, 1955, 2…
2           Azir  [475, 475, 533, 801, 1006, 1233, 1385, 1720, 1…
3           Lulu  [475, 475, 532, 771, 1046, 1288, 1534, 1776, 2…
4           Lulu  [475, 475, 532, 807, 1042, 1338, 1646, 1951, 2…

         redADC redADCChamp  \
0        Sneaky       Sivir
1        CoreJJ       Corki
2           Cop       Corki
3         KEITH      KogMaw
4  Maplestreet8       Corki

                                          goldredADC   redSupport  \
0  [475, 475, 532, 762, 1097, 1469, 1726, 2112, 2…  LemonNation
1  [475, 475, 532, 868, 1220, 1445, 1732, 1979, 2…      KiWiKiD
2  [475, 475, 533, 781, 1085, 1398, 1782, 1957, 2…   BunnyFuFuu
3  [475, 475, 532, 766, 1161, 1438, 1776, 1936, 2…      Xpecial
4  [475, 475, 532, 792, 1187, 1488, 1832, 2136, 2…        Dodo8

  redSupportChamp                                      goldredSupport  \
0          Thresh  [515, 515, 577, 722, 911, 1042, 1194, 1370, 14…
1           Annie  [515, 515, 583, 752, 900, 1066, 1236, 1417, 15…
2           Janna  [515, 515, 584, 721, 858, 1002, 1168, 1303, 14…
3           Janna  [515, 515, 583, 721, 870, 1059, 1205, 1342, 15…
4           Annie  [475, 475, 538, 671, 817, 948, 1104, 1240, 136…

                            redBans
0  ['Tristana', 'Leblanc', 'Nidalee']
1      ['RekSai', 'Janna', 'Leblanc']
2        ['Leblanc', 'Zed', 'RekSai']
3      ['RekSai', 'Rumble', 'LeeSin']
4       ['Rumble', 'Sivir', 'Rengar']
```

This next cell is very long, so let's explain what's going on. If you take a look at the league dataset above, you'll notice that many of the variables are actually JSON string 'max(goldblue)' 'max(goldred)' 'len(bKills)' 'len(bDragons)' 'len(bBarons)' 'len(rDragons)' 'germats. Since we aren't doing any time based analysis, we need to convert these lists into quantitative variables that would be t last ofby our logistic regression model. As a result, we'll perform the following transformation to create a new dataframe: quantleague. alllumns sta[-1]rt -> the total amount of gold blue team earned (gold amount at the last time step)ing h gold' [-1]'g -> the total amount of gold red team earned et gold amt at e -> the total amount of kills by blueteam * 'len(rKills)' -> total kills by red teamar ly, mid, and la -> total amt of dragons captured by blue teamt* role' 'if game -> total amt of dragons captured by red team * 'len(bBarons)' -> total amt of barons captured by blue team * 'len(rBarons)' -> total amt of barons captured by red team * 'len(bHeralds)' -> total amt of heralds captured by blue team * 'len(rHeralds)' -> total amt of heralds captured by red team * was comeback, even, or one sided ((bluegold - redgold) looking at -> this will give us the total gold. * shifts in gold diff)'. We do this by finding the golgamet -> this will give us an idea of how each team is doing throughout the game.*

[8]:
```
TIMES = [(0, 'early'), (1, 'mid'), (2, 'late')]
OBJECTIVES = ['Kills', 'Towers', 'Inhibs', 'Dragons', 'Barons', 'Heralds']
def getGold(name):
    totalgoldblue = []
    for x in league['gold'+name]:
        x = json.loads(x)
        totalgoldblue.append(int(x[-1]))
    return totalgoldblue

def getTimedGold(name, t):
    totalgoldblue = []
    for x in league['gold'+name]:
        x = json.loads(x)
        total = len(x) // 3
        step = total // 2 + t * total #break at 16%, 49%, 83%
        avg = round((int(x[step-1]) + int(x[step]) + int(x[step+1])) / 3)
        totalgoldblue.append(avg)
    return totalgoldblue

def getNum(name):
    totalgoldblue = []
    for i,x in enumerate(league[name]):
        numOccurences = list(x).count('[') -1
        # if name == 'bHeralds' and numOccurences > 2:
        #     print(league.iloc[i, :])

        totalgoldblue.append(numOccurences)
    return totalgoldblue
```

```
def buildQuantitative():
    quantleague = wins[['gamelength']]
    #independent
    for c in ['blue', 'red']:
        quantleague['totalgold'+c] = getGold(c)
        for t in TIMES:
            quantleague[t[1]+'gold'+c] = getTimedGold(c, t[0])
    for t in TIMES:
        quantleague[t[1] + 'gold' + 'diff'] = getTimedGold('diff', t[0])
    for c in ['b', 'r']:
        for o in OBJECTIVES:
            quantleague[c+o] = getNum(c+o)
    quantleague['bluewin'] = wins[['bResult']]
    return quantleague

def addCategoricalNumerical(df):
    #dependent variables, if dummy variables are needed
    categorical = wins[['League', 'Year', 'Season', 'Type']]
    return pd.concat([df, pd.get_dummies(categorical)], axis=1)


def addCategorical(df):
    #if the analysis methods support using strings directly!
    return pd.concat([df, wins[['League', 'Year', 'Season', 'Type']]], axis=1)

quantleague = addCategoricalNumerical(addCategorical(buildQuantitative()))
#this is what we'll use for doing predictions on later!

quantleague.to_csv('processed/quantleague.csv', index=None, sep=',', mode='w')
#save dataframe  for analysis.ipynb
quantleague.head()
```

[8]:

	gamelength	totalgoldblue	earlygoldblue	midgoldblue	lategoldblue	\
0	40	62729	7415	29141	50672	
1	38	57702	7796	26277	45917	
2	40	70270	8000	31424	55072	
3	41	58612	7834	26344	45168	
4	35	60269	6652	22728	42269	

	totalgoldred	earlygoldred	midgoldred	lategoldred	earlygolddiff	\
0	56672	7984	28073	47619	-569	
1	56537	7761	24931	44373	34	
2	56355	7437	26249	46878	563	
3	63119	7858	25226	48523	-25	
4	48947	6602	23054	38690	50	

	midgolddiff	lategolddiff	bKills	bTowers	bInhibs	bDragons	bBarons	\

```
0         1069          3053      32        9        2         1        0
1         1345          1544      20        7        0         3        1
2         5175          8193      44       15        4         4        1
3         1119         -3355      20        4        0         0        1
4         -326          3578      44        8        1         2        1

   bHeralds  rKills  rTowers  rInhibs  rDragons  rBarons  rHeralds  bluewin  \
0         0      18        4        0         3        1         0        1
1         0      18        7        1         1        0         0        0
2         0      16        2        0         0        0         0        1
3         0      42        7        1         4        0         0        0
4         0      20        4        0         1        0         0        1

  League  Year  Season    Type  Year  League_CBLoL  League_CLS  League_EULCS  \
0  NALCS  2015  Spring  Season  2015             0           0             0
1  NALCS  2015  Spring  Season  2015             0           0             0
2  NALCS  2015  Spring  Season  2015             0           0             0
3  NALCS  2015  Spring  Season  2015             0           0             0
4  NALCS  2015  Spring  Season  2015             0           0             0

   League_IEM  League_LCK  League_LCL  League_LJL  League_LLN  League_LMS  \
0           0           0           0           0           0           0
1           0           0           0           0           0           0
2           0           0           0           0           0           0
3           0           0           0           0           0           0
4           0           0           0           0           0           0

   League_MSI  League_NALCS  League_OPL  League_RR  League_TCL  League_WC  \
0           0             1           0          0           0          0
1           0             1           0          0           0          0
2           0             1           0          0           0          0
3           0             1           0          0           0          0
4           0             1           0          0           0          0

   Season_Spring  Season_Summer  Type_International  Type_Playoffs  \
0              1              0                   0              0
1              1              0                   0              0
2              1              0                   0              0
3              1              0                   0              0
4              1              0                   0              0

   Type_Promotion  Type_Regional  Type_Season
0               0              0            1
1               0              0            1
2               0              0            1
3               0              0            1
4               0              0            1
```

4 Exploratory analysis

[9]:
```
# We do a variety of groupbys to find any interesting characteristics across␣
 ↪our dependent variables
tidyleague = pd.concat([ wins[['League', 'Year', 'Season', 'Type']],␣
 ↪quantleague.drop(['League', 'Year', 'Season', 'Type'], axis=1)], axis=1)
# tidyleague is a combination of both our independenct and dependent variables.
grouped = {}
def groupby(column):
    return tidyleague.groupby(column).agg(np.mean).reset_index()

# For exploratory analysis, we grouby each of our dependent variables, to see␣
 ↪how how the independent variables change when the dependent are constant.
for col in ['League', 'Year', 'Season', 'Type']:
    grouped[col] = groupby(col)
# Grouped by type of match
grouped['Type']
```

[9]:

	Type	Year	gamelength	totalgoldblue	earlygoldblue	\
0	International	2015.916413	36.545593	62180.089666	8106.699088	
1	Playoffs	2016.179355	36.832258	62193.574194	8239.470968	
2	Promotion	2016.329923	37.677749	62465.322251	8270.524297	
3	Regional	2015.979021	37.132867	63052.979021	8246.041958	
4	Season	2016.341589	37.042632	63050.672740	8333.506811	

	midgoldblue	lategoldblue	totalgoldred	earlygoldred	midgoldred	\
0	27771.764438	49546.451368	61149.803951	8094.392097	27503.606383	
1	27909.236129	49815.166452	61515.854194	8220.291613	27710.910968	
2	28065.751918	50082.355499	62370.025575	8285.974425	28042.941176	
3	28113.391608	50357.000000	61832.531469	8180.552448	27839.790210	
4	28209.281797	50326.224306	62160.186450	8306.659296	27992.508403	

	lategoldred	earlygolddiff	midgolddiff	lategolddiff	bKills	\
0	48943.344985	12.287234	268.182371	603.098784	26.990881	
1	49392.130323	19.163871	198.350968	423.067097	26.376774	
2	49999.790281	-15.460358	22.805627	82.552430	24.936061	
3	49770.412587	65.468531	273.622378	586.566434	24.349650	
4	49843.873872	26.843446	216.776225	482.342119	25.680170	

	bTowers	bInhibs	bDragons	bBarons	bHeralds	rKills	rTowers	\
0	6.723404	1.258359	1.873860	0.732523	0.249240	25.294833	5.765957	
1	6.683871	1.227097	1.821935	0.725161	0.330323	25.091613	5.889032	
2	6.534527	1.104859	1.774936	0.687980	0.286445	24.976982	6.107417	
3	6.629371	1.181818	1.867133	0.692308	0.188811	23.426573	5.720280	
4	6.774279	1.239165	1.861136	0.727401	0.339996	24.076066	5.960021	

	rInhibs	rDragons	rBarons	rHeralds	bluewin	League_CBLoL	League_CLS	\

```
0  0.969605  1.876900  0.697568  0.218845  0.556231       0.000000    0.000000
1  0.922581  1.874839  0.686452  0.242581  0.539355       0.074839    0.029677
2  1.074169  2.089514  0.767263  0.176471  0.514066       0.000000    0.000000
3  0.937063  1.664336  0.783217  0.139860  0.538462       0.000000    0.000000
4  1.031311  1.930833  0.737308  0.259331  0.545551       0.042986    0.026888

   League_EULCS  League_IEM  League_LCK  League_LCL  League_LJL  League_LLN  \
0      0.000000    0.209726    0.000000    0.000000    0.000000    0.000000
1      0.187097    0.000000    0.098065    0.068387    0.014194    0.025806
2      0.352941    0.000000    0.186701    0.000000    0.000000    0.000000
3      0.230769    0.000000    0.272727    0.000000    0.000000    0.000000
4      0.138511    0.000000    0.222360    0.040333    0.043694    0.039271

   League_LMS  League_MSI  League_NALCS  League_OPL  League_RR  League_TCL  \
0    0.000000    0.168693      0.000000    0.000000   0.153495    0.000000
1    0.085161    0.000000      0.181935    0.070968   0.000000    0.163871
2    0.109974    0.000000      0.350384    0.000000   0.000000    0.000000
3    0.251748    0.000000      0.244755    0.000000   0.000000    0.000000
4    0.111976    0.000000      0.169644    0.071290   0.000000    0.093048

   League_WC  Season_Spring  Season_Summer  Type_International  Type_Playoffs  \
0   0.468085       0.275076       0.724924                1.0            0.0
1   0.000000       0.512258       0.487742                0.0            1.0
2   0.000000       0.634271       0.365729                0.0            0.0
3   0.000000       0.000000       1.000000                0.0            0.0
4   0.000000       0.475146       0.524854                0.0            0.0

   Type_Promotion  Type_Regional  Type_Season
0             0.0            0.0          0.0
1             0.0            0.0          0.0
2             1.0            0.0          0.0
3             0.0            1.0          0.0
4             0.0            0.0          1.0
```

[10]:
```
# Grouped by season
grouped['Season']
```

[10]:
```
   Season         Year  gamelength  totalgoldblue  earlygoldblue  \
0  Spring  2016.387813   37.118736   63190.918850    8330.932232
1  Summer  2016.189630   36.921860   62573.997566    8272.599075

    midgoldblue  lategoldblue  totalgoldred  earlygoldred    midgoldred  \
0  28279.965547  50481.394077  62211.538155   8316.504271  28052.120729
1  28005.168939  49950.112220  61841.456183   8241.590312  27809.593720

    lategoldred  earlygolddiff  midgolddiff  lategolddiff     bKills  \
0  49930.296982      14.424544   227.851082    551.086845  26.137813
```

```
1  49552.805501      30.998296   195.582765    397.306719  25.513145

    bTowers  bInhibs  bDragons   bBarons  bHeralds     rKills   rTowers  \
0  6.764806  1.23918  1.860194  0.709852  0.318907  24.535308  5.907460
1  6.729309  1.22517  1.848588  0.737829  0.331305  24.133398  5.966164

    rInhibs  rDragons   rBarons  rHeralds   bluewin  League_CBLoL  League_CLS  \
0  1.019932  1.974089  0.698178  0.248007  0.548690      0.045843    0.022210
1  1.011441  1.880477  0.759250  0.247322  0.540166      0.034080    0.023612

   League_EULCS  League_IEM  League_LCK  League_LCL  League_LJL  League_LLN  \
0      0.150057    0.019932    0.194476    0.040148    0.042141    0.033884
1      0.139241    0.016553    0.185492    0.034080    0.026777    0.029942

   League_LMS  League_MSI  League_NALCS  League_OPL  League_RR  League_TCL  \
0    0.096241    0.031606      0.167426    0.066913   0.000000    0.089123
1    0.107108    0.000000      0.166504    0.054284   0.024586    0.082765

   League_WC  Season_Spring  Season_Summer  Type_International  Type_Playoffs  \
0   0.000000            1.0            0.0            0.051538       0.113041
1   0.074976            0.0            1.0            0.116115       0.092016

   Type_Promotion  Type_Regional  Type_Season
0        0.070615        0.00000     0.764806
1        0.034810        0.03481     0.722249
```

[11]:
```
#Grouped by Year
grouped['Year']
```

[11]:
```
   Year  gamelength  totalgoldblue  earlygoldblue   midgoldblue  lategoldblue  \
0  2014   36.653846   57765.423077    7329.269231  25790.346154  45832.000000
1  2015   38.331551   60451.748663    8085.742647  27363.474599  48429.720588
2  2016   37.040096   63541.856055    8476.791901  28455.668003  50807.599840
3  2017   36.358502   63178.429477    8237.482936  28137.199034  50332.497735
4  2018   37.643154   67974.278008    8957.228216  30233.883817  54335.705394

   totalgoldred  earlygoldred    midgoldred   lategoldred  earlygolddiff  \
0  56518.653846   7305.102564  25354.179487  45043.358974      24.166667
1  59688.325535   8079.409091  27202.130348  48090.004679       6.320187
2  63058.649559   8475.852045  28379.006415  50555.714515       0.933039
3  62074.948656   8187.744186  27802.768952  49676.757475      49.734219
4  66518.701245   8958.614108  30110.522822  53512.045643      -1.398340

   midgolddiff  lategolddiff     bKills   bTowers   bInhibs  bDragons  \
0   436.179487    788.653846  29.871795  6.833333  1.371795  2.153846
1   161.341578    339.702540  28.665775  6.772727  1.179813  2.176471
2    76.660385    251.882518  24.182839  6.657177  1.161989  1.855253
```

```
3   334.445485    655.733615  26.015705  6.797342  1.298097  1.715494
4   123.394191    823.709544  20.497925  6.755187  1.315353  1.643154

    bBarons  bHeralds     rKills   rTowers   rInhibs  rDragons   rBarons  \
0  0.628205  0.000000  27.410256  5.564103  1.064103  1.833333  0.679487
1  0.709893  0.002674  27.033422  5.979947  0.955214  2.143717  0.710561
2  0.672815  0.502406  23.587009  6.136327  1.022855  1.990377  0.693665
3  0.761401  0.335246  24.021142  5.792208  1.025068  1.786771  0.764422
4  0.887967  0.473029  18.124481  5.784232  1.161826  1.775934  0.804979

   rHeralds   bluewin  League_CBLoL  League_CLS  League_EULCS  League_IEM  \
0  0.000000  0.589744      0.000000    0.000000      0.000000    0.000000
1  0.003342  0.546791      0.000000    0.000000      0.199866    0.031417
2  0.426624  0.528067      0.058540    0.000000      0.143545    0.026063
3  0.225008  0.553307      0.041075    0.050438      0.121413    0.007853
4  0.302905  0.551867      0.078838    0.033195      0.165975    0.000000

   League_LCK  League_LCL  League_LJL  League_LLN  League_LMS  League_MSI  \
0    0.000000    0.000000    0.000000     0.00000    0.000000    0.000000
1    0.262701    0.000000    0.000000     0.00000    0.131684    0.018717
2    0.193264    0.056135    0.043705     0.00000    0.104250    0.016439
3    0.152522    0.042585    0.045002     0.07309    0.086681    0.012685
4    0.269710    0.000000    0.000000     0.00000    0.141079    0.000000

   League_NALCS  League_OPL  League_RR  League_TCL  League_WC  Season_Spring  \
0      0.000000    0.000000   0.000000    0.000000   1.000000       0.000000
1      0.205214    0.000000   0.000000    0.101604   0.048797       0.440508
2      0.154370    0.087410   0.000000    0.085405   0.030874       0.430233
3      0.163395    0.067351   0.030504    0.081244   0.024162       0.464814
4      0.161826    0.070539   0.000000    0.078838   0.000000       1.000000

   Season_Summer  Type_International  Type_Playoffs  Type_Promotion  \
0       1.000000           1.000000       0.000000        0.000000
1       0.559492           0.098930       0.122326        0.088235
2       0.569767           0.073376       0.108260        0.027265
3       0.535186           0.075204       0.097252        0.036545
4       0.000000           0.000000       0.000000        0.290456

   Type_Regional  Type_Season
0       0.000000     0.000000
1       0.033422     0.657086
2       0.018444     0.772654
3       0.014195     0.776805
4       0.000000     0.709544
```

Hmm…Heralds captured seem like a dead giveaway to predicting which year as they weren't yet introduced in 2014.(The rift herald is easier to reach by redside)

It seems like bluewin percentage steadily decreased from 2014-2016

Baron kills definitely increase over the years!

[12]:
```
grouped["League"]
```

[12]:

	League	Year	gamelength	totalgoldblue	earlygoldblue	\
0	CBLoL	2016.578073	39.777409	68018.568106	8972.368771	
1	CLS	2017.045714	35.480000	61399.434286	8057.845714	
2	EULCS	2016.166515	36.941765	62394.008189	8330.469518	
3	IEM	2015.847826	35.702899	60974.101449	7950.753623	
4	LCK	2016.167474	38.811765	65798.914187	8737.622145	
5	LCL	2016.501779	36.491103	62455.604982	8186.530249	
6	LJL	2016.577519	36.670543	61590.554264	8203.019380	
7	LLN	2017.000000	35.462810	61067.892562	7983.214876	
8	LMS	2016.203085	36.696658	61945.326478	8187.208226	
9	MSI	2016.126126	35.720721	61768.297297	8055.315315	
10	NALCS	2016.245283	36.956761	62571.672170	8266.640723	
11	OPL	2016.561135	34.375546	59493.683406	7730.899563	
12	RR	2017.000000	36.801980	63960.900990	8422.009901	
13	TCL	2016.237366	36.026034	61172.278714	8027.117917	
14	WC	2015.516234	37.136364	62284.873377	8091.691558	

	midgoldblue	lategoldblue	totalgoldred	earlygoldred	midgoldred	\
0	30424.285714	54401.388704	67405.372093	9024.674419	30424.807309	
1	27281.462857	48966.217143	60728.331429	7941.320000	26786.200000	
2	28151.124659	49840.092812	61719.245678	8282.355778	27920.483167	
3	26994.934783	48530.340580	60157.956522	7987.557971	26940.528986	
4	29493.244291	52608.040138	64726.935640	8716.918339	29325.875433	
5	27818.967972	49879.096085	61571.131673	8164.572954	27582.249110	
6	27585.895349	49241.007752	61806.748062	8158.120155	27568.344961	
7	26923.185950	48552.838843	60056.900826	7944.958678	26921.904959	
8	27662.365039	49467.937018	60964.715938	8147.961440	27320.719794	
9	27616.369369	49027.459459	59278.810811	7975.972973	26846.990991	
10	28108.951258	49994.463836	61630.358491	8233.161164	27850.632075	
11	26325.194323	47366.248908	59300.218341	7735.969432	26202.585153	
12	28664.138614	51058.059406	64684.178218	8447.376238	28863.148515	
13	27289.915773	48858.526799	60112.467075	8050.540582	27100.886677	
14	27883.198052	49693.071429	61109.490260	8069.185065	27546.707792	

	lategoldred	earlygolddiff	midgolddiff	lategolddiff	bKills	\
0	54166.073090	-52.305648	-0.564784	235.305648	24.192691	
1	48173.045714	116.485714	495.291429	793.160000	27.942857	
2	49565.073703	48.091902	230.648772	275.000000	25.497725	
3	48079.789855	-36.826087	54.449275	450.579710	29.333333	
4	52167.652595	20.703806	167.370934	440.386851	23.278893	
5	49351.277580	21.925267	236.743772	527.822064	27.316726	
6	49194.674419	44.941860	17.492248	46.360465	23.317829	

```
7   48147.954545      38.231405     1.309917    404.904959  24.958678
8   48741.800771      39.235219   341.663239    726.114396  25.262211
9   47698.810811      79.315315   769.423423   1328.612613  28.756757
10  49394.905660      33.463836   258.316038    599.540881  26.415094
11  47240.294760      -5.063319   122.624454    125.986900  27.799127
12  51601.960396     -25.396040  -199.059406   -543.930693  25.584158
13  48278.140888     -23.393568   189.056662    580.396631  29.546708
14  48906.961039      22.493506   336.522727    786.103896  25.766234

     bTowers   bInhibs  bDragons   bBarons  bHeralds     rKills   rTowers  \
0   7.046512  1.468439  2.109635  0.794020  0.448505  22.750831  6.215947
1   6.634286  1.102857  1.794286  0.668571  0.394286  25.942857  6.022857
2   6.821656  1.202002  1.868062  0.758872  0.320291  24.252957  6.189263
3   6.500000  1.181159  1.789855  0.594203  0.246377  27.231884  5.615942
4   6.741176  1.296886  2.016609  0.766090  0.301730  21.645675  5.796540
5   6.697509  1.120996  1.654804  0.626335  0.419929  25.679715  5.836299
6   6.232558  1.058140  1.635659  0.674419  0.399225  23.271318  5.872093
7   6.739669  1.400826  1.578512  0.681818  0.322314  22.818182  6.033058
8   6.493573  1.115681  1.848329  0.706941  0.331620  23.838046  5.787918
9   6.792793  1.108108  2.072072  0.792793  0.360360  24.900901  5.297297
10  6.915881  1.255503  1.880503  0.720126  0.277516  24.803459  6.014151
11  6.478166  1.085153  1.517467  0.620087  0.441048  26.908297  6.061135
12  6.405941  1.168317  1.683168  0.801980  0.415842  28.455446  6.811881
13  6.923430  1.283308  1.813170  0.735069  0.326187  27.843798  5.842266
14  6.902597  1.376623  1.902597  0.750000  0.155844  23.532468  5.659091

     rInhibs  rDragons   rBarons  rHeralds   bluewin  League_CBLoL  \
0   1.239203  2.089701  0.770764  0.415282  0.544850           1.0
1   1.068571  1.811429  0.760000  0.262857  0.525714           0.0
2   1.047316  1.837125  0.744313  0.247498  0.535942           0.0
3   0.920290  1.775362  0.666667  0.195652  0.550725           0.0
4   1.046367  2.013841  0.773702  0.248443  0.550865           0.0
5   0.932384  2.032028  0.679715  0.327402  0.544484           0.0
6   0.968992  2.034884  0.693798  0.372093  0.523256           0.0
7   1.136364  1.900826  0.731405  0.256198  0.541322           0.0
8   0.973008  1.849614  0.697943  0.181234  0.537275           0.0
9   0.747748  1.810811  0.567568  0.297297  0.585586           0.0
10  0.978774  1.982704  0.729560  0.218553  0.547170           0.0
11  1.021834  1.794760  0.716157  0.292576  0.515284           0.0
12  1.435644  2.009901  0.871287  0.465347  0.475248           0.0
13  0.947933  1.851455  0.712098  0.211332  0.566616           0.0
14  0.918831  1.902597  0.701299  0.120130  0.574675           0.0

    League_CLS  League_EULCS  League_IEM  League_LCK  League_LCL  League_LJL  \
0          0.0           0.0         0.0         0.0         0.0         0.0
1          1.0           0.0         0.0         0.0         0.0         0.0
2          0.0           1.0         0.0         0.0         0.0         0.0
```

```
3          0.0           0.0          1.0          0.0          0.0          0.0
4          0.0           0.0          0.0          1.0          0.0          0.0
5          0.0           0.0          0.0          0.0          1.0          0.0
6          0.0           0.0          0.0          0.0          0.0          1.0
7          0.0           0.0          0.0          0.0          0.0          0.0
8          0.0           0.0          0.0          0.0          0.0          0.0
9          0.0           0.0          0.0          0.0          0.0          0.0
10         0.0           0.0          0.0          0.0          0.0          0.0
11         0.0           0.0          0.0          0.0          0.0          0.0
12         0.0           0.0          0.0          0.0          0.0          0.0
13         0.0           0.0          0.0          0.0          0.0          0.0
14         0.0           0.0          0.0          0.0          0.0          0.0

    League_LLN  League_LMS  League_MSI  League_NALCS  League_OPL  League_RR  \
0          0.0         0.0         0.0           0.0         0.0        0.0
1          0.0         0.0         0.0           0.0         0.0        0.0
2          0.0         0.0         0.0           0.0         0.0        0.0
3          0.0         0.0         0.0           0.0         0.0        0.0
4          0.0         0.0         0.0           0.0         0.0        0.0
5          0.0         0.0         0.0           0.0         0.0        0.0
6          0.0         0.0         0.0           0.0         0.0        0.0
7          1.0         0.0         0.0           0.0         0.0        0.0
8          0.0         1.0         0.0           0.0         0.0        0.0
9          0.0         0.0         1.0           0.0         0.0        0.0
10         0.0         0.0         0.0           1.0         0.0        0.0
11         0.0         0.0         0.0           0.0         1.0        0.0
12         0.0         0.0         0.0           0.0         0.0        1.0
13         0.0         0.0         0.0           0.0         0.0        0.0
14         0.0         0.0         0.0           0.0         0.0        0.0

    League_TCL  League_WC  Season_Spring  Season_Summer  Type_International  \
0          0.0         0.0       0.534884       0.465116                0.0
1          0.0         0.0       0.445714       0.554286                0.0
2          0.0         0.0       0.479527       0.520473                0.0
3          0.0         0.0       0.507246       0.492754                1.0
4          0.0         0.0       0.472664       0.527336                0.0
5          0.0         0.0       0.501779       0.498221                0.0
6          0.0         0.0       0.573643       0.426357                0.0
7          0.0         0.0       0.491736       0.508264                0.0
8          0.0         0.0       0.434447       0.565553                0.0
9          0.0         0.0       1.000000       0.000000                1.0
10         0.0         0.0       0.462264       0.537736                0.0
11         0.0         0.0       0.513100       0.486900                0.0
12         0.0         0.0       0.000000       1.000000                1.0
13         1.0         0.0       0.479326       0.520674                0.0
14         0.0         1.0       0.000000       1.000000                1.0
```

	Type_Playoffs	Type_Promotion	Type_Regional	Type_Season
0	0.192691	0.000000	0.000000	0.807309
1	0.131429	0.000000	0.000000	0.868571
2	0.131938	0.125569	0.030027	0.712466
3	0.000000	0.000000	0.000000	0.000000
4	0.052595	0.050519	0.026990	0.869896
5	0.188612	0.000000	0.000000	0.811388
6	0.042636	0.000000	0.000000	0.957364
7	0.082645	0.000000	0.000000	0.917355
8	0.084833	0.055270	0.046272	0.813625
9	0.000000	0.000000	0.000000	0.000000
10	0.110849	0.107704	0.027516	0.753931
11	0.120087	0.000000	0.000000	0.879913
12	0.000000	0.000000	0.000000	0.000000
13	0.194487	0.000000	0.000000	0.805513
14	0.000000	0.000000	0.000000	0.000000

This one seems the most promising for prediction! All the values seem slighly different, especially LCK's

4.1 Dataframes and groupbys are nice..but if we really want to gain some insights we need to some visualizations.

4.1.1 First, let's look at the relationship between early game gold difference and late game gold difference...we would expect to see that games who start off with a lead will tend to keep that lead, but is that really true?

[13]:
```
#We randomly sample 5000 points because this is the maximum altair supports.

gold_heat = alt.Chart(tidyleague.sample(5000), title = "Early Game VS Late Game␣
↪Gold Difference ").mark_rect().encode(
    x=alt.X('earlygolddiff:Q', bin=alt.Bin(maxbins=30)),
    y=alt.Y('lategolddiff:Q', bin=alt.Bin(maxbins=30)),
    color=alt.Color('count(lategoldblue):Q')
)
gold_heat
```

[13]:
```
alt.Chart(…)
```

So we can definitely conclude that there IS a relationship, although not a very strong one. If we were to see a strong correlation between early game gold and late game gold, this bimodal distrbuition should be skewed even more to the right: that is, the upper concentration of lategolddiff should be more along to the right, and the lower concentration should be more along the left. It seems that are doing badly in the early game only lose slight ground...this tells us League of Legends is a game full of upsets and surprises!

4.1.2 Next, let's look instead only at blue side. We will compare early game gold vs late game gold, in order to better understand this relationship.

[14]:
```
display(alt.Chart(tidyleague.sample(5000), title = "Early Game Gold VS Late␣
↪Game Gold for Blueside ").mark_rect().encode(
    x=alt.X('earlygoldblue', scale=alt.Scale(zero=False), bin=alt.
↪Bin(maxbins=30)),
    y=alt.Y('lategoldblue', scale=alt.Scale(zero=False), bin=alt.
↪Bin(maxbins=30)),
    color='count():Q'
).interactive())
display(alt.Chart(tidyleague.sample(5000), title = "Early Game Gold VS Late␣
↪Game Gold for Redside ").mark_rect().encode(
    x=alt.X('earlygoldred', scale=alt.Scale(zero=False), bin=alt.
↪Bin(maxbins=30)),
    y=alt.Y('lategoldred', scale=alt.Scale(zero=False), bin=alt.
↪Bin(maxbins=30)),
    color='count():Q'
).interactive())
```

```
alt.Chart(…)
```

```
alt.Chart(…)
```

Hmm, now this is very interesting. Although before we saw that the amount of advantage gained in the early game did not correspond very highly to the amount of advantage later in the game (measured through gold), here we see an almost perfect correlation between the amount of gold earned in the early game vs the amount of gold earned later. From this graph we may be able to infer that if a te able to g gain more as well: since the diffrenence between gold amounts stays relatively sawmall, as we saw gold difference compared to the gold gteams that earn morwe can conclude that e gold probably don't conettest thearen't as agressive on average as teams that earn less, as their is a close match in the amount red earns as well. ### Finally, let's create an interactive correlation heatmap to see if there are any surprising correlations between our variables.

[15]:
```
#the ideas and the compute_2d_histogram() from the following visualization are␣
↪borrowed from Paul Hiemstra, at
# https://towardsdatascience.com/
↪altair-plot-deconstruction-visualizing-the-correlation-structure-of-weather-data-38fb5668c5



independentvars = buildQuantitative().sample(2000)[['earlygolddiff', 'bluewin',␣
↪'midgolddiff', 'lategolddiff', 'gamelength', 'bBarons', 'rBarons',␣
↪'bDragons', 'rDragons', 'bKills', 'rKills']]

# We take a sample of 2000 in order to speed up compute time, looking only at␣
↪our independent variables.
```

[16]:
```
'''Borrowed from Paul, this is used to show the correlations between two␣
↪variables that the user selects'''
def compute_2d_histogram(var1, var2, df, density=True):
    H, xedges, yedges = np.histogram2d(df[var1], df[var2], density=density)
    H[H == 0] = np.nan

    # Create a nice variable that shows the bin boundaries
    xedges = pd.Series(['{0:.4g}'.format(num) for num in xedges])
    xedges = pd.DataFrame({"a": xedges.shift(), "b": xedges}).dropna().agg(' -␣
↪'.join, axis=1)
    yedges = pd.Series(['{0:.4g}'.format(num) for num in yedges])
    yedges = pd.DataFrame({"a": yedges.shift(), "b": yedges}).dropna().agg(' -␣
↪'.join, axis=1)

    # Cast to long format using melt
    res = pd.DataFrame(H,
                       index=yedges,
                       columns=xedges).reset_index().melt(
                            id_vars='index'
                       ).rename(columns={'index': 'value2',
                                         'value': 'count',
                                         'variable': 'value'})


    # Also add the raw left boundary of the bin as a column, will be used to␣
↪sort the axis labels later
    res['raw_left_value'] = res['value'].str.split(' - ').map(lambda x: x[0]).
↪astype(float)
    res['raw_left_value2'] = res['value2'].str.split(' - ').map(lambda x: x[0]).
↪astype(float)
    res['variable'] = var1
    res['variable2'] = var2
    return res.dropna() # Drop all combinations for which no values where found
```

[17]:
```
# Get a heatmap of positive vs negative earlygolddiff, and win vs loss

# Define selector, this is what ties together our two plots
var_sel_cor = alt.selection_single(fields=['variable', 'variable2'],␣
↪clear=False,
                                   init={'variable': 'bkills', 'variable2':␣
↪'earlygolddiff'})
cor_data = (independentvars
              .corr().stack()
              .reset_index()     # The stacking results in an index on the␣
↪correlation values, we need the index as normal columns for Altair
```

```
            .rename(columns={0: 'correlation', 'level_0': 'variable',␣
↪'level_1': 'variable2'}))
cor_data['correlation_label'] = cor_data['correlation'].map('{:.2f}'.format)  #␣
↪Round to 2 decimal
# Define correlation heatmap
base = alt.Chart(cor_data).encode(
    x='variable2:O',
    y='variable:O'
)

#Add the r^2 values
text = base.mark_text().encode(
    text='correlation_label',
    color=alt.condition(
        alt.datum.correlation > 0.5,
        alt.value('white'),
        alt.value('black')
    )
)

#The code that ties together the correlation plot with our binned histogram plot
cor_plot = base.mark_rect().encode(
    color=alt.condition(var_sel_cor, alt.value('pink'), 'correlation:Q')
).add_selection(var_sel_cor)
```

[18]:
```
value_columns = independentvars
binnedQuantLeague = pd.concat([compute_2d_histogram(var1, var2,␣
↪independentvars) for var1 in value_columns for var2 in value_columns])
#Here we create the dataframe that our histogram plot will use, by using Paul's␣
↪function for every combination of our variables
binnedQuantLeague.head()
```

[18]:

	value2	value	count	raw_left_value	raw_left_value2	variable	variable2
0	-3042 - -2516	-3042 - -2516	1.807168e-09	-3042.0	-3042.0	earlygolddiff	earlygolddiff
11	-2516 - -1990	-2516 - -1990	5.421504e-09	-2516.0	-2516.0	earlygolddiff	earlygolddiff
22	-1990 - -1464	-1990 - -1464	3.072186e-08	-1990.0	-1990.0	earlygolddiff	earlygolddiff
33	-1464 - -938	-1464 - -938	1.301161e-07	-1464.0	-1464.0	earlygolddiff	earlygolddiff
44	-938 - -412	-938 - -412	4.807067e-07	-938.0	-938.0	earlygolddiff	earlygolddiff

[19]:
```
# Define 2d binned histogram plot
scat_plot = alt.Chart(binnedQuantLeague.sample(5000)).transform_filter(
    var_sel_cor
).mark_rect().encode(
    alt.X('value:N', sort=alt.EncodingSortField(field='raw_left_value')),
    alt.Y('value2:N', sort=alt.EncodingSortField(field='raw_left_value2',␣
 ↪order='descending')),
    alt.Color('count:Q', scale=alt.Scale(scheme='blues'))
)
# Finally we combine all plotsat plots both side-by-side
alt.hconcat((cor_plot + text).properties(width=350, height=350), scat_plot.
 ↪properties(width=350, height=350)).resolve_scale(color='independent')
```

[19]:
```
alt.HConcatChart(…)
```

5 Predictive Analysis

[20]:
```
import json
import matplotlib as mpp
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

filename = "processed/quantleague.csv"
#This was is our dataset after preprocessing to transform the columns to␣
 ↪quantitative

quantleague = pd.read_csv(filename, index_col=None)
quantleague.drop(labels='Year.1', axis=1, inplace=True)
defaultxcols = ['gamelength', 'earlygoldblue', 'midgoldblue', 'earlygoldred',␣
 ↪'midgoldred',
                'earlygolddiff', 'midgolddiff',
                 'bDragons', 'bBarons', 'bHeralds',
                 'rDragons', 'rBarons', 'rHeralds', 'Year']
defaultycols = ['bluewin']
```

Since we have so many different dependent variables that we want to do analysis on, it only makes sense to encapsulate our model pipeline in functions, so that we can reuse the same code later by defining the independent and dependent variables. As a result, you'll see all the code below in functions, as we walk through a tutorial analysis first attempting to predict which team will win. To start off with, we define a function to build the training and testing sets, where most of the heavy lifting is done from sklearn's train_test_split

[21]:
```
def createDataset(xcols, ycols, df):
    x = df[xcols]
    # #leaving out late gold columns
    y = df[ycols].to_numpy()
    x = StandardScaler().fit_transform(x)
    return train_test_split(x, y, train_size=.8, random_state=42)
```

[22]:
```
x_train, x_test, y_train, y_test  = createDataset(defaultxcols, defaultycols,␣
 ↪quantleague)
# we attempt to predict who will win with early game gold difference and␣
 ↪neutral objectives.
```

5.1 Let's do a quick PCA analysis to attempt to visualize our data

We use this as a quick way to see if this task will be at all possible, or if the data is so clumped together that there is no hope. Since the original data has 23 dimensions, using PCA we can actually visualize around 60% of the variance with just 2 dimensions.

[23]:
```
def visualizePCA(x_train, y_train, y_name):
    pca = PCA(n_components=2)
    principalcomponents = pca.fit_transform(x_train)
    pcs = pd.DataFrame(data=principalcomponents, columns=['PC1', 'PC2'])
    pcs[y_name] = y_train
    alt.Chart(pcs.sample(5000)).mark_point(size=1).encode(x='PC1', y='PC2',␣
 ↪color=y_name).display()
    return pca
```

[24]:
```
pca = visualizePCA(x_train, y_train, 'Blue win')
```

```
alt.Chart(…)
```

[25]:
```
sum(pca.explained_variance_ratio_)
# This is great news! Around 59% of the variance of our data can be explained␣
 ↪with just two principal components. Let's look at how many components we␣
 ↪would need to capture
# at how many would be needed to explain 95% of the variance:
```

[25]:
```
0.5965303653796767
```

[26]:
```
def visualizePCAComponents(x_train):
    numcomponents = x_train.shape[1]
    pca = PCA(numcomponents).fit(x_train)

    pca_variance = pd.DataFrame({
        'x': list(range(1,numcomponents+1)),
        'y': np.cumsum(pca.explained_variance_ratio_)
    })
```

```
    alt.Chart(pca_variance).mark_line().encode(
        x = alt.X('x', title='Number of components'),
        y = alt.Y('y', title='Cumulative explained variance', scale=alt.
↪Scale(domain=(0.3,1.02)))
    ).interactive().display()
```

[27]:
```
visualizePCAComponents(x_train)
```

```
alt.Chart(…)
```

5.2 Logistic Regression

To explain 95% or more of the variance, we only need to use 11 components, which would be a significant reduction in space complexity. However, we won't actually use this decomposed matrix for prediction, as we would like to still understand which weights correspond to which coefficients, and that our dataset is not large enough to see a signficant time reduction in training.

[28]:
```
def LogReg(x_train, y_train):
    return LogisticRegression().fit(x_train, y_train)

def evaluateModel(model, x_test, y_test):
    return model.score(x_test, y_test)
```

Since all of our variables are categorical, we'll want to use Logistic Regression for predicting the classes. We use scikitlearn's built in model for prediction

[29]:
```
blueWinModel = LogReg(x_train, y_train)
evaluateModel(blueWinModel, x_test, y_test)
```

[29]:
```
0.8976377952755905
```

Our model achieves 89% right out of the box! This is extremely good news, as LoL is a complex game and we weren't entirely sure that this task would be possible.

[30]:
```
def confusion(model, x_test, y_test, labels):
    predictions = model.predict(x_test)
    return metrics.confusion_matrix(y_test, predictions, labels=labels,␣
↪normalize='true')

def confusion_graph(confusion_matrix, labels):
    cmlen = confusion_matrix.shape[0]
    x,y = np.meshgrid(labels,labels)
    source = pd.DataFrame({
        'x': x.ravel(),
        'y': y.ravel(),
        'z':confusion_matrix.ravel()
    })
```

```
    return alt.Chart(source).mark_rect().encode(
        x=alt.X('x:N', title="Predicted"),
        y=alt.Y('y:N', title="Target"),
        color='z:Q'
    ).properties(width=200, height=200)
```

Finally, let's create a confusion matrix to visualize where our model makes mistakes. Since the accuracy is so high for this classification task, the confusion matrix looks more or less ideal. In the graph below, 1 corresponds to Blue winning and 0 corresponds to Red winning.

[31]:
```
confusion_graph(confusion(blueWinModel, x_test, y_test, np.unique(y_test)), np.
↪unique(y_test))
```

[31]:
```
alt.Chart(…)
```

[32]:
```
def createCoeffs(labels, values, classes):
    dat = {}
    for i, k in enumerate(labels):
        dat[k] = []
        for j in values:
            dat[k].append(j[i])
    res= pd.DataFrame(dat, index=classes)
    if len(res) == 2:
        res.iloc[1, :] *= -1
    return res
```

[33]:
```
display(createCoeffs(defaultxcols, blueWinModel.coef_, np.unique(y_test)[::-1]))
```

```
   gamelength  earlygoldblue  midgoldblue  earlygoldred  midgoldred  \
1    0.456919      -0.003758    -0.064918      0.001802   -0.311468
0   -0.456919       0.003758     0.064918     -0.001802    0.311468

   earlygolddiff  midgolddiff  bDragons   bBarons  bHeralds  rDragons  \
1      -0.020671     0.641232  0.649796  1.117982  0.038407 -0.633143
0       0.020671    -0.641232 -0.649796 -1.117982 -0.038407  0.633143

    rBarons  rHeralds     Year
1 -1.254191 -0.037293  0.04978
0  1.254191  0.037293 -0.04978
```

We use this function to map the coefficients found by logistic regression to our labels - in essence, this function will allow us to see which variables were most important for determining the dependent variable, which is this case is who wins. Here we can see Barons influence the most, with Dragons as next. Early gold has nearly no correlation.

5.3 Logistic Regression on all of our Target Variables, specifically focusing on League

[34]:
```
def pipeline(xcols, ycols, df=quantleague):
    print('Creating data...')
    x_train, x_test, y_train, y_test = createDataset(xcols, ycols, df)
    print('Creating model...')
    model = LogReg(x_train, y_train)
    print('Evaluating model...')
    labels = np.unique(y_test)
    print('Accuracy:', evaluateModel(model, x_test, y_test))
    # print('Coefficients:')
    # display(createCoeffs(xcols, model.coef_, labels))
    print('Displaying confusion matrix')
    confusion_graph(confusion(model, x_test, y_test, labels), labels).display()
#We encapsulate this entire process through pipeline, commenting our the␣
 ↪coefficients because they can get quite large...
```

[35]:
```
pipeline(defaultxcols, defaultycols)
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.8976377952755905
Displaying confusion matrix
```

```
alt.Chart(…)
```

[36]:
```
pipeline(['gamelength', 'totalgoldblue', 'earlygoldblue', 'midgoldblue',
       'lategoldblue', 'totalgoldred', 'earlygoldred', 'midgoldred',
       'lategoldred', 'earlygolddiff', 'midgolddiff', 'lategolddiff', 'bKills',
       'bTowers', 'bInhibs', 'bDragons', 'bBarons', 'bHeralds', 'rKills',
       'rTowers', 'rInhibs', 'rDragons', 'rBarons', 'rHeralds',
       'Year'], ['Season'])
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.5872703412073491
Displaying confusion matrix
```

```
alt.Chart(…)
```

There are nearly no differences between how teams play between Summer and Spring...at least from our independent variables.

[37]:
```
pipeline(['gamelength', 'totalgoldblue', 'earlygoldblue', 'midgoldblue',
       'lategoldblue', 'totalgoldred', 'earlygoldred', 'midgoldred',
       'lategoldred', 'earlygolddiff', 'midgolddiff', 'lategolddiff', 'bKills',
```

```
       'bTowers', 'bInhibs', 'bDragons', 'bBarons', 'bHeralds', 'rKills',
       'rTowers', 'rInhibs', 'rDragons', 'rBarons', 'rHeralds',
       ], ['Year'])
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.7152230971128609
Displaying confusion matrix
```

```
alt.Chart(…)
```

We can see there is a significant different in how the game has changed over time, as an accuracy of 71% is much higher than from the baseline of random guessing, which would be 20%

[38]:
```
pipeline(['gamelength', 'totalgoldblue', 'earlygoldblue', 'midgoldblue',
       'lategoldblue', 'totalgoldred', 'earlygoldred', 'midgoldred',
       'lategoldred', 'earlygolddiff', 'midgolddiff', 'lategolddiff', 'bKills',
       'bTowers', 'bInhibs', 'bDragons', 'bBarons', 'bHeralds', 'rKills',
       'rTowers', 'rInhibs', 'rDragons', 'rBarons', 'rHeralds',
       'Year'], ['Type'])
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.7637795275590551
Displaying confusion matrix
```

```
alt.Chart(…)
```

This analysis was fairly useless, as nearly all of the population is dominated by Season type games, and so by simply guessing Season every time our model achieves a 76% accuracy.

5.4 Taking a closer look at the "League" dependent variable

[39]:
```
def gnL(name): #wrapper to select easier
    return quantleague['League'] != name

withoutTournaments = quantleague.loc[gnL('WC') & gnL('RR') & gnL('MSI') &␣
 ↪gnL('IEM')]
def getNonLeague(l):
    res = []
    for x in l:
        if 'League' not in x:
            res.append(x)
    res.remove('Year')
    res.remove('Season')
    res.remove('Type')
    return res
```

```
#Filter out the Tournaments, which are not Leagues, and strip out our dummpy␣
 ↪dependent variables in our quantleague set

pipeline(getNonLeague(list(withoutTournaments.columns)), ['League'],␣
 ↪df=withoutTournaments)
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.25125628140703515
Displaying confusion matrix
```

```
alt.Chart(…)
```

so it seems like league is very hard for the model to classify, as there are 14 classes and most of them don't have much data. Let's try comparing only the top 4 leagues that we have the most data for, in order to see if we can draw more meaningful conclusions. First we need to filter our dataset to only include these three leagues: `* league samples * LCK  1445 * NALCS  1272 * EULCS  1099`

[40]:
```
def gL(name): #wrapper to select easier
    return quantleague['League'] == name

top3 = quantleague.loc[gL('LCK') | gL('EULCS') | gL('NALCS')]
```

[41]:
```
pipeline(getNonLeague(list(top3.columns)), ['League'], df=top3)
```

```
Creating data…
Creating model…
Evaluating model…
Accuracy: 0.4869109947643979
Displaying confusion matrix
```

```
alt.Chart(…)
```

From this accuracy of 48%, we can see that there most definitely is a difference in how different regions play the game. LCK(Korea) seems to have a very distinctive, signature style, while NALCS(North America) and EULCS(Europe) seems to have a more similar playstyle.

[42]:
```
# The following code is for generating one of the figure in our report and does␣
 ↪not have to do with our analysis.
# x = ['0: Baseline (none removed)', '1: Total Gold', '2: Kills', '3:␣
 ↪Inhibitors and Turrets']
# y = [98.2, 98.4, 97.7, 89.7]
# import pandas as pd
# import altair as alt
# z = pd.DataFrame({'After removal of variables': x, 'Accuracy (%)': y})
```

```
# base = alt.Chart(z).encode( alt.X('Accuracy (%):Q', scale=alt.
 ↪Scale(domain=(80, 100))), y= 'After removal of variables')
# alt.layer(base.mark_line(), base.mark_area()).properties(width=500,␣
 ↪height=300).configure_axis(labelFontSize=12, titleFontSize=16)
```

6 Conclusion

There was a trove of insights and surprises that we learned from studying this dataset: not only was League a much more complex game than initially anticipated, we learned that inital leads have nearly no correlation between winning in pro play, as well the fact that there is most definitely a difference in playstyle between the different regions, with Korea specifically having a very distinctive style. Additionally, we learned that League is a constantly evolving game, with each year being quantitatvely distinct from the last. The ethical considerations of esports and their relentless pursuit of collecting their players' data is something that should not be considered lightly, and may point to ever growing focus on statistics in sports, with any performance that is non measurable as effectively non existent. Capturing human morale and emotion is something that is just as significant as any other variable in determining who will win, and one must consider this as well. Data cannot collect the full picture, and we should not act the assumption that it does. We hope you enjoyed this analysis as much as we did, and learned something along the way. Thank you for an amazing quarter!