In [1]:
```
import pandas as pd
import numpy as np
import json
import matplotlib.pyplot as plt
```

In [201]:
```
train_1 = pd.read_csv('dota2Train.csv', header = None)
test = pd.read_csv('dota2Test.csv', header = None)
```

In [202]:
```
train_1
```

Out[202]:

	0	1	2	3	4	5	6	7	8	9	...	107	108	109	110	111	112	113	114	115	11
0	-1	223	2	2	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
1	1	152	2	2	0	0	0	1	0	-1	...	0	0	0	0	0	0	0	0	0	
2	1	131	2	2	0	0	0	1	0	-1	...	0	0	0	0	0	0	0	0	0	
3	1	154	2	2	0	0	0	0	0	0	...	-1	0	0	0	0	0	0	0	0	
4	-1	171	2	3	0	0	0	0	0	-1	...	0	0	0	0	0	0	0	0	0	
5	1	122	2	3	0	1	0	0	0	0	...	1	0	0	0	0	0	0	0	0	
6	1	224	8	3	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
7	-1	227	8	3	0	-1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
8	-1	111	2	3	0	0	1	0	0	0	...	0	1	0	0	0	0	0	0	0	
9	-1	151	2	2	0	0	0	1	0	0	...	0	0	1	0	0	0	0	0	0	
10	1	145	2	3	0	1	0	0	1	1	...	1	0	0	0	0	0	0	0	0	

In [3]:
```
col_1 = pd.DataFrame(json.load(open('regions.json'))['regions'])
col_1 = col_1.set_index('id').iloc[:,0]
col_2 = pd.DataFrame(json.load(open('mods.json'))['mods'])
col_2 = col_2.set_index('id').iloc[:, 0]
col_3 = pd.DataFrame(json.load(open('lobbies.json'))['lobbies'])
col_3 = col_3.set_index('id').iloc[:, 0]
col_4 = pd.DataFrame(json.load(open('heroes.json'))['heroes'])
```

In [200]:
```
train
```

Out[200]:

	result	regions	mods	lobbies	4	5	6	7	8	9	...	107	108	109	110
0	loss	China	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
1	win	Southeast Asia	Captains Mode	Tournament	0	0	0	1	0	-1	...	0	0	0	0
2	win	Europe West	Captains Mode	Tournament	0	0	0	1	0	-1	...	0	0	0	0
3	win	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	0	...	-1	0	0	0
4	loss	Australia	Captains Mode	Tutorial	0	0	0	0	0	-1	...	0	0	0	0
5	win	US East	Captains Mode	Tutorial	0	1	0	0	0	0	...	1	0	0	0
6	win	China	Reverse Captains Mode	Tutorial	0	0	0	0	0	0	...	0	0	0	0
7	loss	China	Reverse Captains Mode	Tutorial	0	-1	0	0	0	0	...	0	0	0	0
8	loss	US West	Captains Mode	Tutorial	0	0	1	0	0	0	...	0	1	0	0
9	loss	Southeast Asia	Captains Mode	Tournament	0	0	0	1	0	0	...	0	0	1	0
10	win	South Korea	Captains Mode	Tutorial	0	1	0	0	1	1	...	1	0	0	0
11	loss	China	Captains Mode	Tournament	-1	0	1	0	0	0	...	-1	0	0	0
12	loss	Russia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
13	loss	Southeast Asia	Captains Mode	Tutorial	0	1	0	0	-1	-1	...	0	0	0	0
14	loss	Russia	Captains Mode	Tournament	0	0	0	1	0	0	...	0	0	0	0
15	loss	South Korea	Captains Mode	Tutorial	0	0	0	-1	0	1	...	-1	0	0	0
16	win	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
17	win	China	Captains Mode	Tutorial	0	0	0	0	0	-1	...	0	0	0	0
18	win	Southeast Asia	Captains Mode	Tournament	0	0	0	1	0	0	...	1	0	0	0
19	loss	China	Reverse Captains Mode	Tutorial	1	0	0	-1	0	0	...	0	0	0	0
20	loss	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	-1	...	-1	1	0	0

	result	regions	mods	lobbies	4	5	6	7	8	9	...	107	108	109	110
21	win	Russia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
22	loss	Russia	Captains Mode	Tutorial	0	0	0	0	0	0	...	0	0	0	0
23	win	Australia	Captains Mode	Tournament	0	0	0	0	-1	0	...	0	0	0	0
24	win	Russia	Reverse Captains Mode	Tutorial	0	0	0	0	-1	0	...	0	0	0	0
25	loss	South Korea	Captains Mode	Tournament	0	0	0	0	0	1	...	0	0	0	0
26	loss	South Korea	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
27	win	Southeast Asia	Captains Mode	Tutorial	0	0	0	0	0	0	...	0	0	0	0
28	win	Southeast Asia	Captains Mode	Tournament	1	0	0	1	0	0	...	0	-1	0	0
29	loss	US East	?? INTRO/DEATH ??	Tournament	0	0	0	0	0	0	...	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
92620	loss	Europe West	Captains Mode	Tutorial	0	0	0	0	0	0	...	-1	0	0	0
92621	loss	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	-1	0
92622	win	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	0	...	-1	0	0	0
92623	loss	China	Captains Mode	Tournament	1	0	0	0	0	0	...	0	0	0	0
92624	win	Southeast Asia	Captains Mode	Tournament	0	1	0	0	0	0	...	0	0	0	0
92625	win	US East	Captains Mode	Tutorial	1	0	0	0	-1	0	...	0	0	0	0
92626	win	China	Greeviling	Tournament	0	0	0	0	0	1	...	0	1	0	0
92627	loss	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	1	...	-1	0	0	0
92628	loss	South America	Captains Mode	Tutorial	-1	1	0	0	0	0	...	0	0	0	1
92629	loss	Southeast Asia	Captains Mode	Tournament	0	1	0	0	0	-1	...	0	0	0	0
92630	loss	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	1	0	0
92631	loss	Russia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	0	0
92632	win	Southeast Asia	Captains Mode	Tournament	0	-1	0	0	0	-1	...	0	0	0	0

	result	regions	mods	lobbies	4	5	6	7	8	9	...	107	108	109	110
92633	win	Southeast Asia	Captains Mode	Tournament	0	0	0	1	0	0	...	1	0	0	0
92634	loss	Southeast Asia	Captains Mode	Tutorial	0	0	0	0	0	0	...	0	0	0	0
92635	win	Australia	Captains Mode	Tournament	1	-1	0	0	0	0	...	1	0	0	0
92636	win	Southeast Asia	Greeviling	Tournament	0	0	0	0	0	0	...	0	0	1	0
92637	win	Southeast Asia	Captains Mode	Tutorial	-1	0	0	0	0	0	...	0	0	0	0
92638	loss	India	Captains Mode	Tournament	1	0	0	-1	0	0	...	0	0	0	0
92639	win	US East	Captains Mode	Tutorial	0	0	0	-1	0	1	...	0	0	0	0
92640	win	Russia	Captains Mode	Tournament	0	0	0	0	0	0	...	0	0	1	0
92641	win	Southeast Asia	Greeviling	Tournament	0	0	0	0	0	0	...	1	1	-1	0
92642	win	China	Captains Mode	Tutorial	-1	0	0	1	0	0	...	0	0	0	0
92643	loss	Southeast Asia	Captains Mode	Tutorial	0	-1	0	0	0	1	...	1	0	0	0
92644	win	Southeast Asia	Captains Mode	Tournament	0	0	0	0	0	1	...	0	0	0	0
92645	loss	Southeast Asia	Captains Mode	Tutorial	1	0	0	-1	0	0	...	0	0	0	0
92646	win	Southeast Asia	Captains Mode	Tournament	0	0	0	0	-1	0	...	1	0	0	0
92647	win	US West	Captains Mode	Tutorial	0	0	0	0	0	0	...	0	0	0	0
92648	loss	Russia	Captains Mode	Tournament	0	0	0	0	0	1	...	0	0	0	0
92649	loss	South America	Captains Mode	Tournament	0	-1	0	0	1	0	...	0	0	0	0

92650 rows × 117 columns

In [4]:
```
#drop unused hero
#train = train.drop(27, axis = 1)
```

In [5]:
```
#rename first four columns
train = train.rename(columns={0: "result", 1: "regions", 2: "mods", 3: 'lob
```

In [199]:
```
train.shape
```

Out[199]:
```
(92650, 117)
```

In [208]:
```
train_1[0].value_counts()
```

Out[208]:
```
 1    48782
-1    43868
Name: 0, dtype: int64
```

In [6]:
```
heroes = col_4.drop('localized_name', axis = 1)
heroes.id = heroes.id + 3
hero_dict = heroes.set_index('id').to_dict()['name']
```

In [7]:
```
#rename hero name
#train = train.rename(columns = hero_dict)
```

In [8]:
```
train.result = train.result.apply(lambda x: 'win' if x == 1 else 'loss')
```

In [9]:
```
region_dict = col_1.to_dict()
region_dict[232] = 'unknown'
```

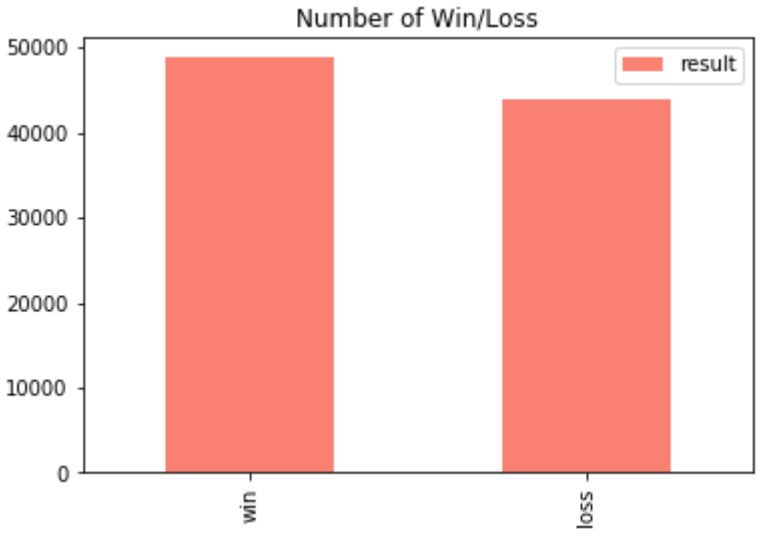
In [10]:
```
train.regions = train.regions.apply(lambda x: region_dict[x])
```

In [11]:
```
train.mods = train.mods.apply(lambda x: col_2.to_dict()[x])
```

In [12]:
```
train.lobbies = train.lobbies.apply(lambda x: col_3.to_dict()[x])
```

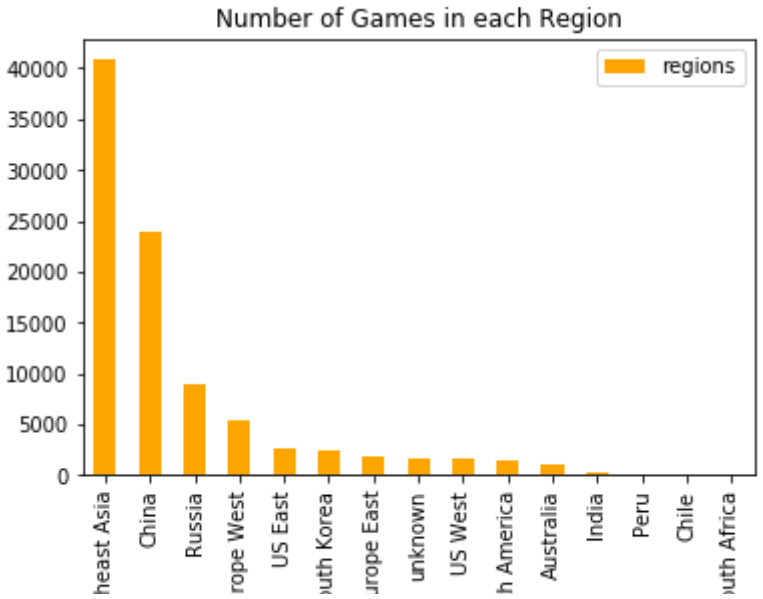
In [13]:
```
train.result.value_counts().to_frame().plot(kind='bar', title = 'Number of
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1103ada90>
```



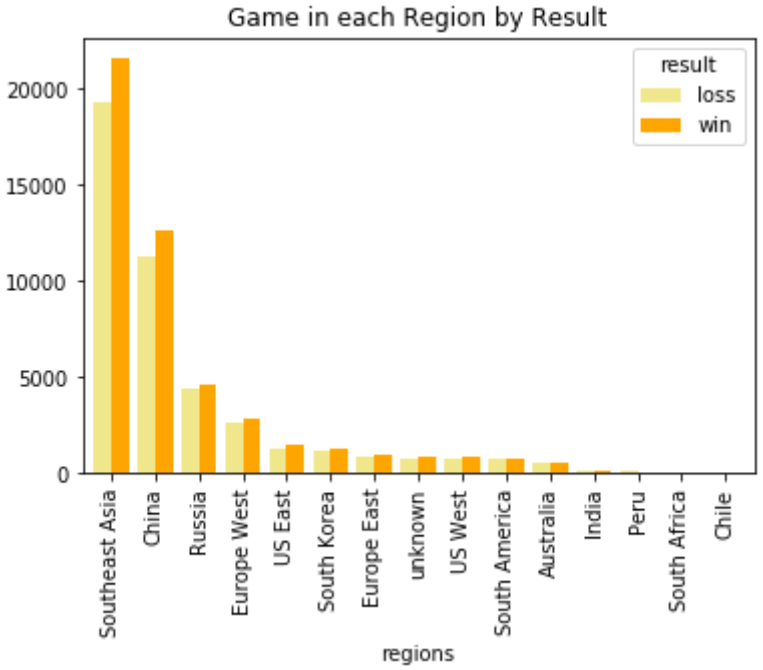
In [51]:
```
train.regions.value_counts().to_frame().plot(kind='bar', title = 'Number of
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1167facc0>
```



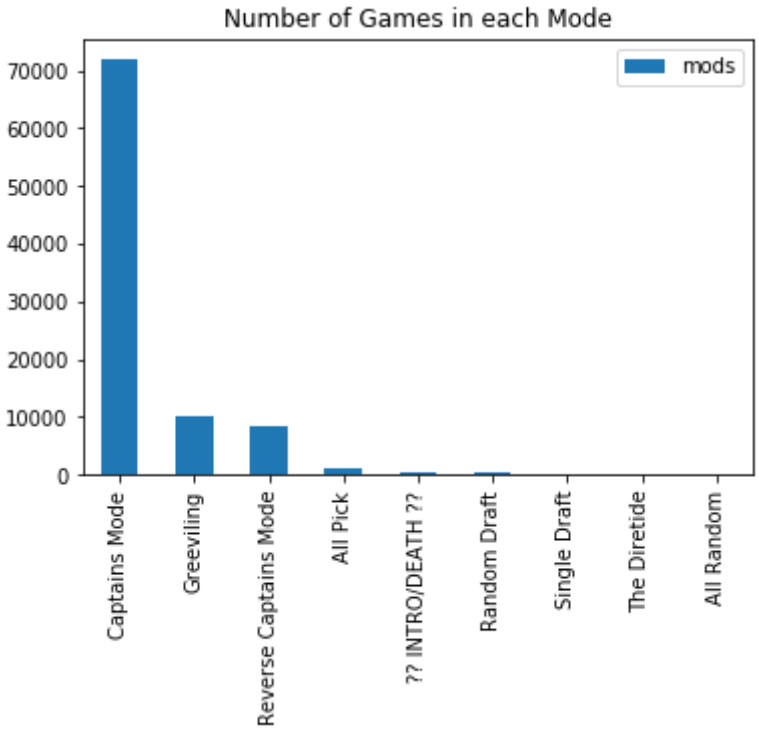
In [217]:
```
cnts = train.pivot_table(index='regions', columns='result', aggfunc='size')

cnts.sort_values('win', ascending = False).plot(kind='bar', width=0.8, colo
```

Out[217]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ad063390>
```



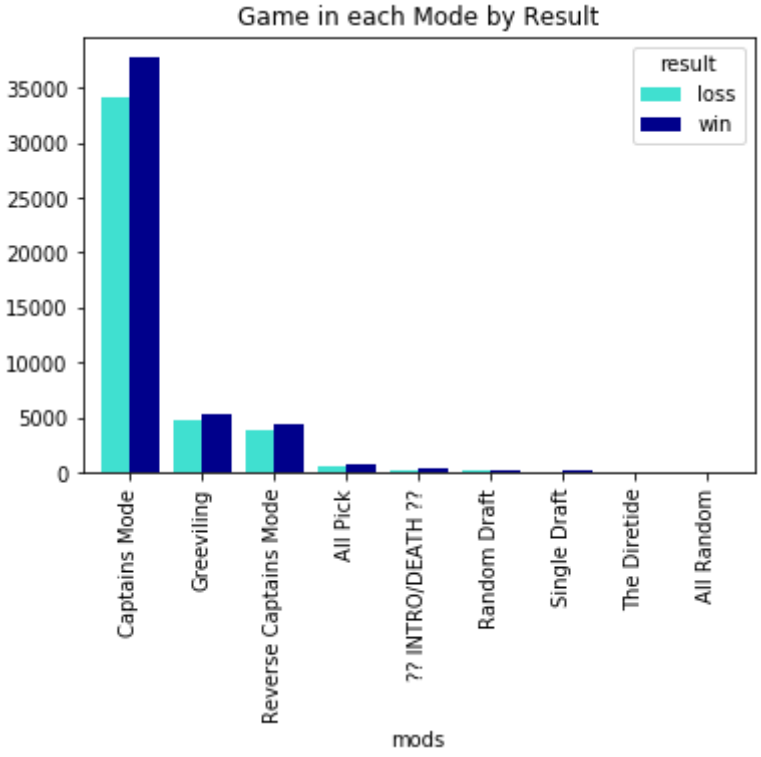
In [15]:
```
train.mods.value_counts().to_frame().plot(kind='bar', title = 'Number of Ga
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1011d53c8>
```



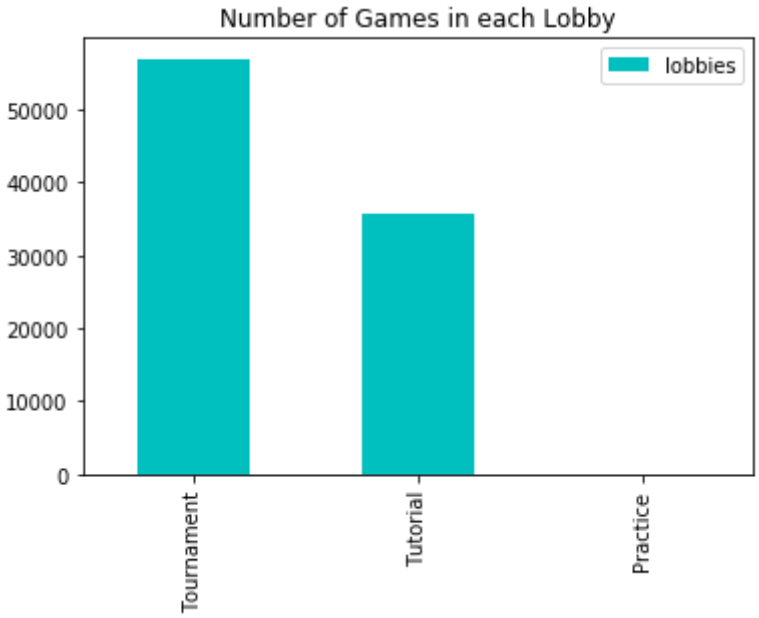
In [218]:
```
cnts = train.pivot_table(index='mods', columns='result', aggfunc='size')

cnts.sort_values('win', ascending = False).plot(kind='bar', width=0.8, colo
```

Out[218]: <matplotlib.axes._subplots.AxesSubplot at 0x1abe58b00>



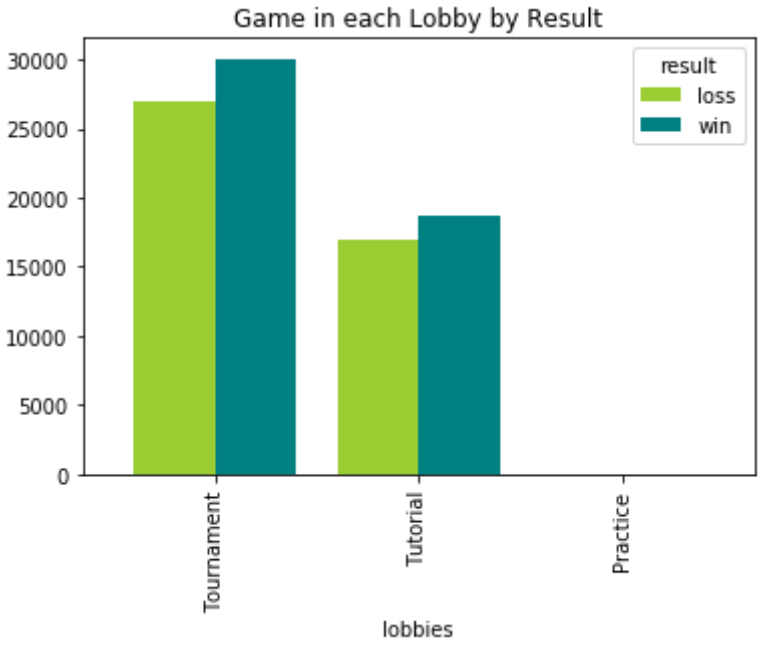
In [16]:
```
train.lobbies.value_counts().to_frame().plot(kind='bar', title = 'Number of
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x110603d30>
```



In [219]:
```
cnts = train.pivot_table(index='lobbies', columns='result', aggfunc='size')

cnts.sort_values('win', ascending = False).plot(kind='bar', width=0.8, colo
```

Out[219]: <matplotlib.axes._subplots.AxesSubplot at 0x1b70c0588>



Heroes

In [73]:
```
hero_df = train.drop(['regions', 'mods', 'lobbies'], axis = 1)#.apply(lambd
```

In [75]:
```
hero_this = hero_df.replace(-1, 0)
hero_other = hero_df.replace(1, 0)
hero_other = hero_other.replace(-1, 1)
hero_other.result = hero_other.result.apply(lambda x: 'win' if x == 'loss'
```

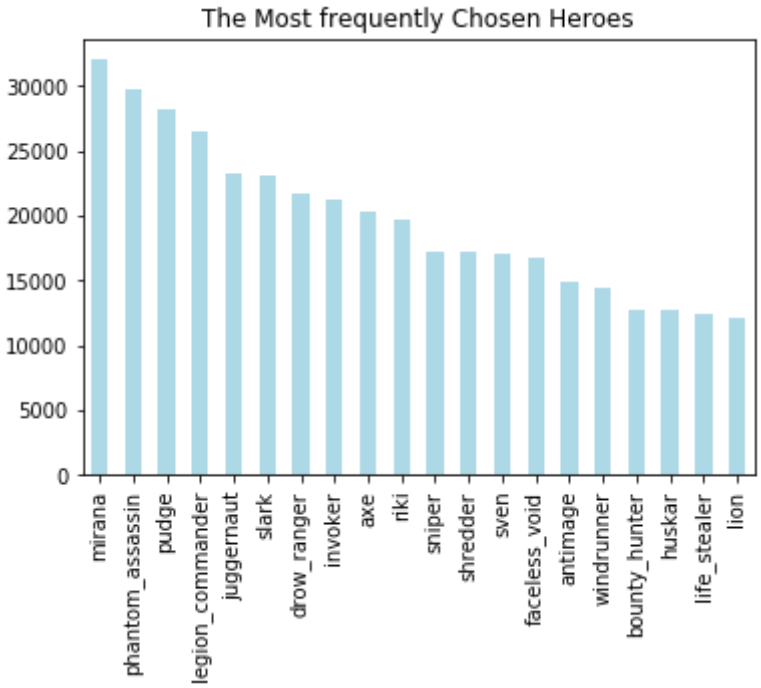
In [76]:
```
comps = pd.concat([hero_this, hero_other])
```

In [33]:
```
hero_this['result'] = train.result
hero_other['result'] = train.result
comps_win = pd.concat([hero_this[hero_this['result']=='win'], hero_other[he
```

In [35]:
```
comps_win = comps_win.drop('result', axis = 1)
```

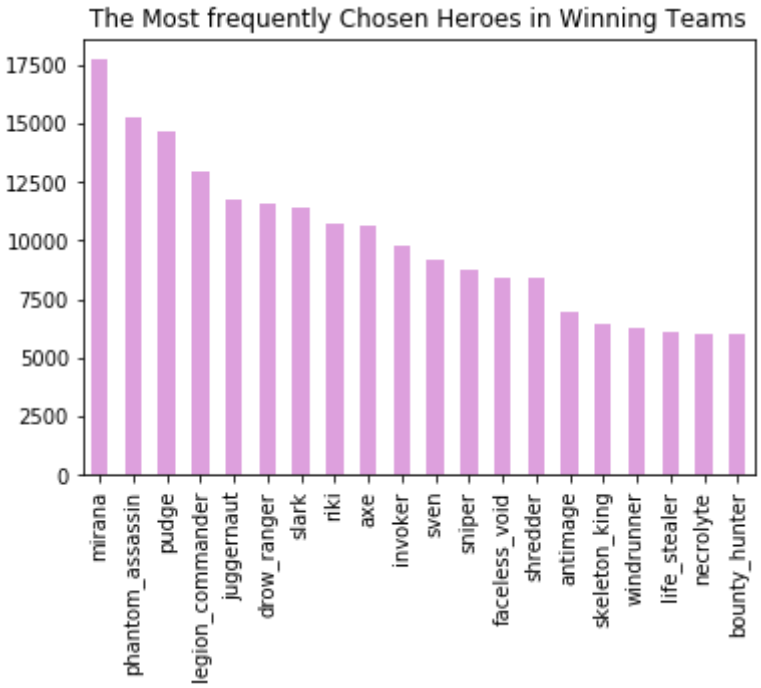
In [20]:
```
comps.rename(columns = hero_dict).apply(sum).sort_values(ascending = False)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x110a38630>
```



In [230]:
```
comps_win.rename(columns = hero_dict).apply(sum).sort_values(ascending = Fa
```

Out[230]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b2a02b00>
```

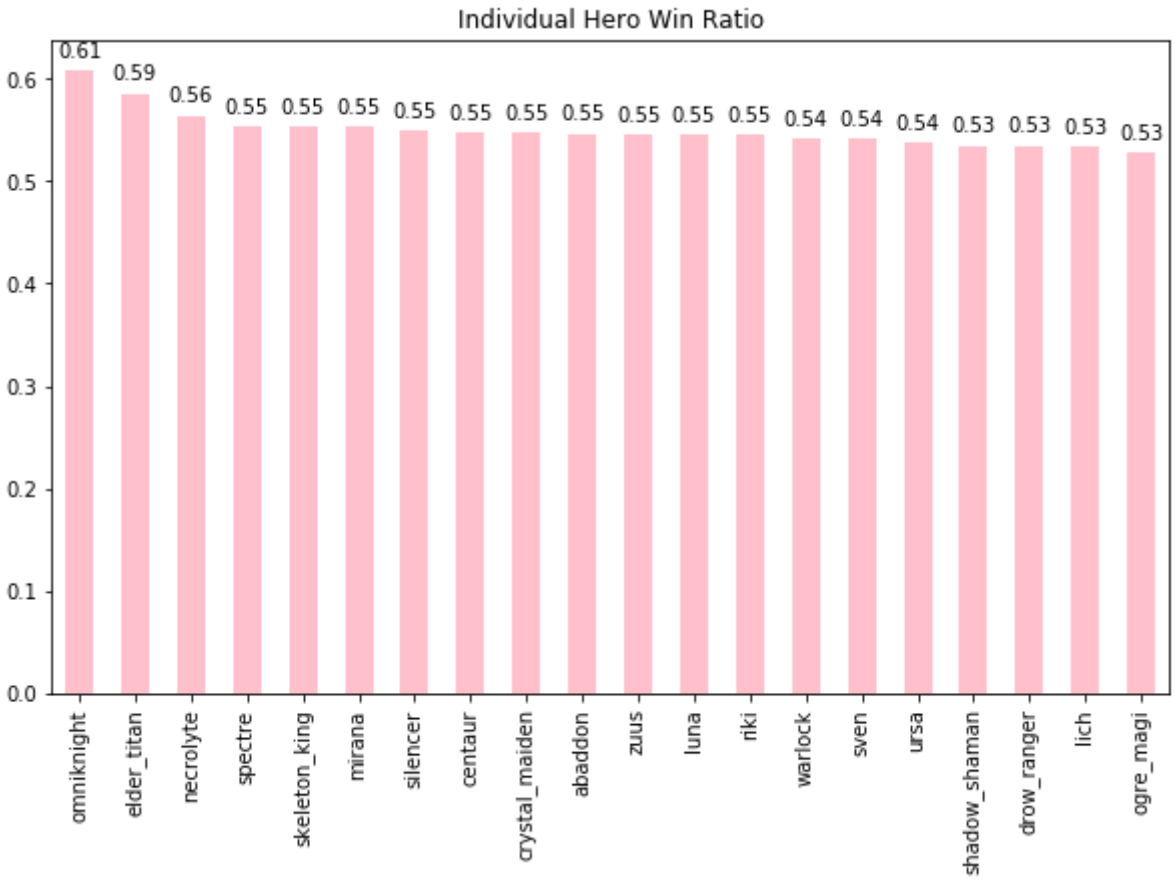


In [79]:
```
comps.result = comps.result == 'win'
```

In [94]:
```
ind_win_ratio = comps.apply(lambda x: (comps.result*x).sum()/x.sum()).drop(
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/computation/expression
s.py:180: UserWarning: evaluating in Python space because the '*' operato
r is not supported by numexpr for the bool dtype, use '&' instead
  .format(op=op_str, alt_op=unsupported[op_str]))
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: RuntimeWa
rning: invalid value encountered in long_scalars
  """Entry point for launching an IPython kernel.
```

In [106]:
```
ax = ind_win_ratio.rename(hero_dict).sort_values(ascending = False)[:20].pl
for p in ax.patches:
    ax.annotate("%.2f" % p.get_height(), (p.get_x() + p.get_width() / 2., p
```



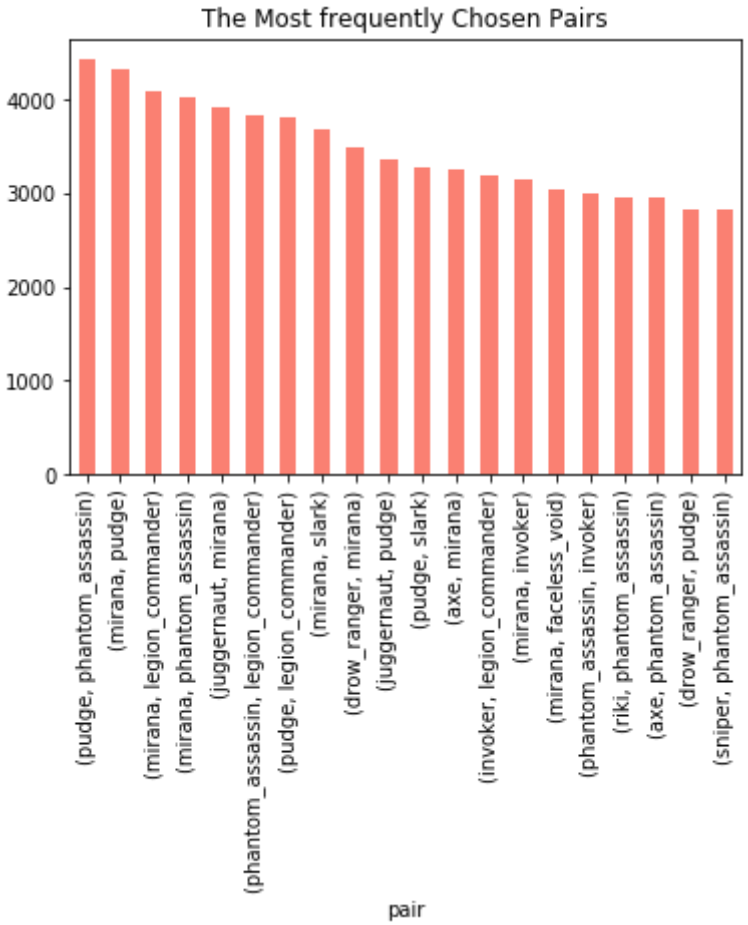
In [137]:
```
def find_pairs(x):
    lst = []
    for i in range(5):
        for j in range(i+1, 5):
            lst.append((x[i], x[j]))
    return pd.Series(lst)
```

In [148]:
```
pair_result = comps_lst.apply(find_pairs)
pair_result['result'] = comps.result
```

In [180]:
```
keys = range(10)
each_pair_result = pd.melt(pair_result, id_vars='result', value_vars=keys,
each_pair_result = each_pair_result.groupby(['pair','result'])['result'].co
each_pair_result = each_pair_result.rename({'result':'count'},axis = 1).res
```

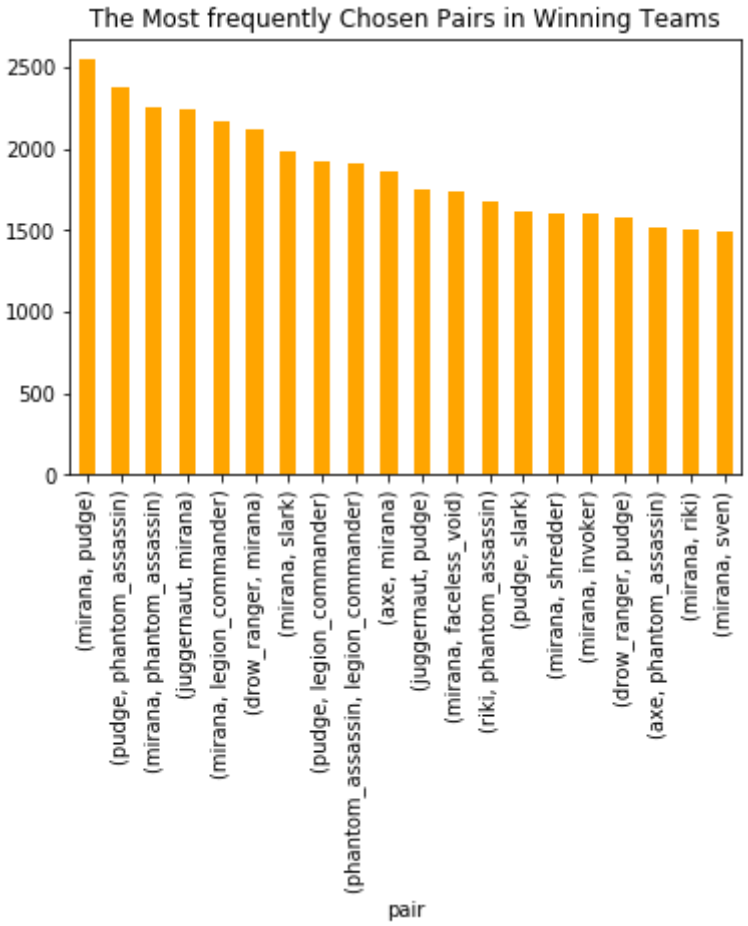
In [231]:
```
each_pair_result.groupby('pair')['count'].sum().sort_values(ascending = Fal
```

Out[231]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b4822908>
```



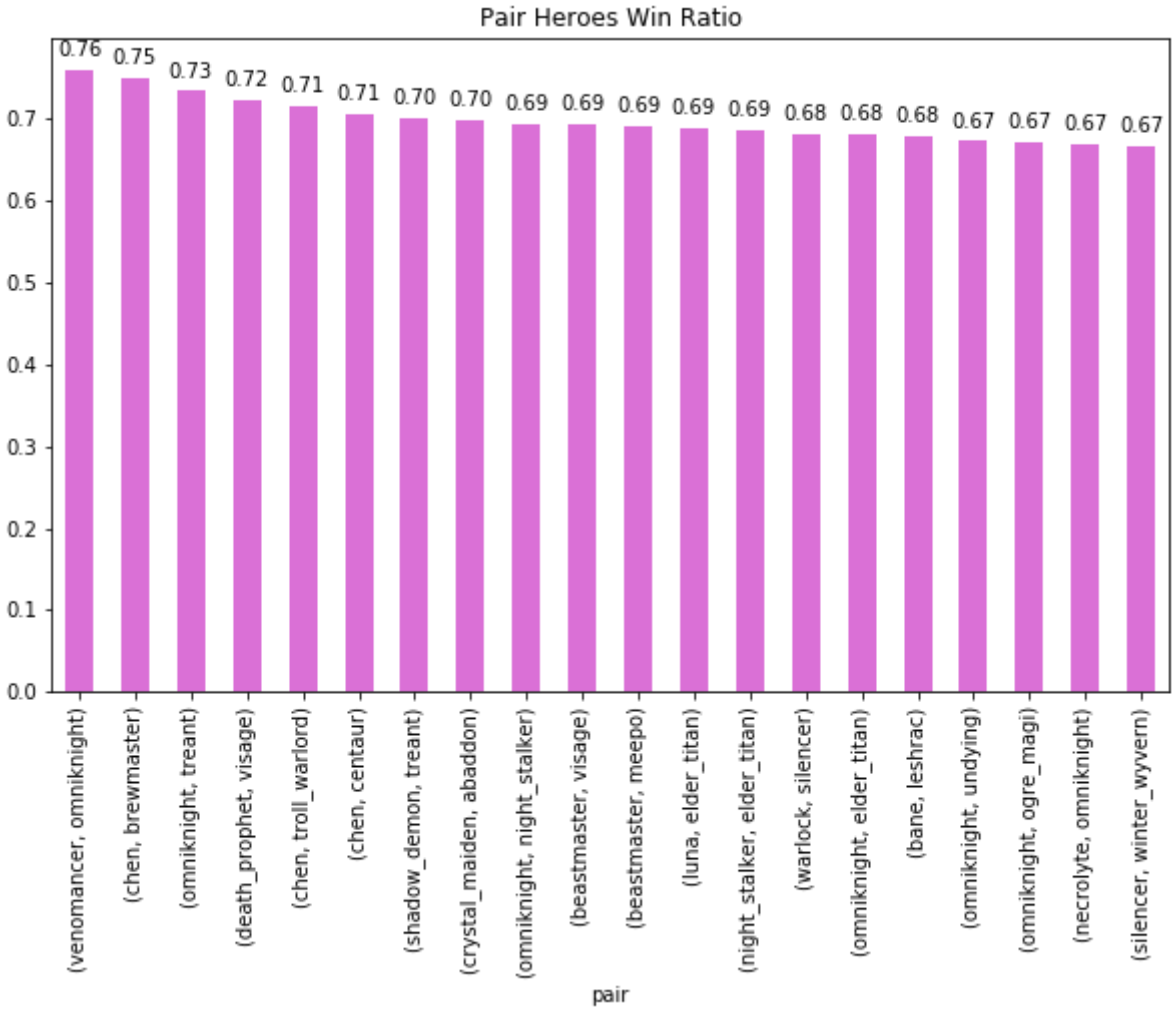
In [232]:
```
each_pair_result[each_pair_result['result']==True].groupby('pair')['count']
```

Out[232]: <matplotlib.axes._subplots.AxesSubplot at 0x1b53b7f98>



In [196]:
```
win_most_pairs = each_pair_result.groupby('pair').apply(lambda x: x[x['resu
```

In [197]:
```
ax = win_most_pairs.sort_values(ascending = False)[:20].plot(kind = 'bar',
for p in ax.patches:
    ax.annotate("%.2f" % p.get_height(), (p.get_x() + p.get_width() / 2., p
```



In [21]:
```
def comp_hero(lst):
    comp = []
    for i in range(len(lst)):
        if lst[i] == 1:
            comp.append(hero_dict[i+4])
    return comp
```

In [22]:
```
comps_lst = comps.drop('result', axis = 1).apply(lambda x: x.tolist(), axis
```

In [37]:
```
comps_win_lst = comps_win.drop('result', axis = 1).apply(lambda x: x.tolist
```

In [40]:
```
comps_cnts = comps_lst.apply(tuple).value_counts()
```

In [44]:
```
comps_cnts[:10]
```

Out[44]:
```
(vengefulspirit, beastmaster, luna, lycan, abaddon)              10
(dazzle, batrider, ancient_apparition, invoker, techies)         10
(antimage, mirana, pudge, zuus, legion_commander)                 5
(axe, juggernaut, mirana, pudge, invoker)                         5
(juggernaut, pudge, slardar, bounty_hunter, invoker)              4
(juggernaut, mirana, riki, invoker, legion_commander)             4
(batrider, ancient_apparition, invoker, rubick, techies)          4
(axe, mirana, sven, lion, invoker)                                4
(riki, tinker, ogre_magi, slark, legion_commander)                4
(drow_ranger, mirana, vengefulspirit, invoker, shadow_demon)      4
dtype: int64
```

In [39]:
```
comps_win_cnts = comps_win_lst.apply(tuple).value_counts()
```

In [45]:
```
comps_win_cnts[:10]
```

Out[45]:
```
(vengefulspirit, beastmaster, luna, lycan, abaddon)                  10
(axe, mirana, sven, lion, invoker)                                    4
(juggernaut, mirana, tidehunter, riki, legion_commander)              3
(antimage, mirana, pudge, zuus, legion_commander)                     3
(axe, juggernaut, mirana, bounty_hunter, shredder)                    3
(riki, tinker, ogre_magi, slark, legion_commander)                    3
(tinker, necrolyte, phantom_assassin, shredder, legion_commander)     3
(juggernaut, pudge, riki, faceless_void, invoker)                     3
(antimage, mirana, phantom_assassin, omniknight, legion_commander)    3
(mirana, riki, ogre_magi, nyx_assassin, legion_commander)             3
dtype: int64
```

In [229]:
```
win_ratio = (comps_win_cnts/comps_cnts)
win_ratio.fillna(0).sort_values(ascending = False)
```

Out[229]:
```
(juggernaut, mirana, tiny, zuus, sniper)
1.0
(juggernaut, life_stealer, spirit_breaker, invoker, techies)
1.0
(drow_ranger, pudge, tiny, enigma, silencer)
1.0
(morphling, pudge, tidehunter, queenofpain, bounty_hunter)
1.0
(mirana, tinker, furion, chaos_knight, treant)
1.0
(bloodseeker, mirana, invoker, disruptor, legion_commander)
1.0
(pudge, sniper, faceless_void, slark, tusk)
1.0
(bloodseeker, drow_ranger, mirana, tidehunter, shredder)
1.0
(mirana, tidehunter, riki, dark_seer, slark)
1.0
(storm_spirit, zuus, tidehunter, phantom_assassin, pugna)
1.0
(slardar, enigma, phantom_assassin, nyx_assassin, slark)
1.0
(axe, huskar, bounty_hunter, alchemist, skywrath_mage)
1.0
(drow_ranger, mirana, nevermore, silencer, rubick)
1.0
(bloodseeker, nevermore, pudge, necrolyte, skeleton_king)
1.0
(drow_ranger, life_stealer, huskar, bounty_hunter, alchemist)
1.0
(juggernaut, windrunner, huskar, shredder, oracle)
1.0
(antimage, skeleton_king, night_stalker, ursa, nyx_assassin)
1.0
(crystal_maiden, viper, life_stealer, huskar, slark)
1.0
(antimage, crystal_maiden, pudge, gyrocopter, chaos_knight)
1.0
(axe, crystal_maiden, zuus, invoker, bristleback)
1.0
(juggernaut, mirana, windrunner, templar_assassin, lycan)
1.0
(tinker, warlock, faceless_void, shredder, legion_commander)
1.0
(drow_ranger, zuus, night_stalker, shredder, abaddon)
1.0
(phantom_lancer, pudge, sniper, undying, ember_spirit)
1.0
(zuus, pugna, rattletrap, ursa, alchemist)
1.0
(drow_ranger, earthshaker, alchemist, disruptor, nyx_assassin)
1.0
(earthshaker, morphling, sniper, gyrocopter, bristleback)
1.0
```

```
(puck, sniper, phantom_assassin, ursa, slark)
1.0
(juggernaut, kunkka, tidehunter, phantom_assassin, spirit_breaker)
1.0
(bloodseeker, juggernaut, pudge, dazzle, medusa)
1.0

...
(pudge, phantom_assassin, invoker, centaur, legion_commander)
0.0
(drow_ranger, faceless_void, death_prophet, luna, medusa)
0.0
(juggernaut, phantom_lancer, windrunner, luna, troll_warlord)
0.0
(sand_king, shadow_shaman, riki, huskar, chaos_knight)
0.0
(tiny, tidehunter, ursa, naga_siren, arc_warden)
0.0
(faceless_void, omniknight, night_stalker, invoker, keeper_of_the_light)
0.0
(juggernaut, tiny, zuus, weaver, elder_titan)
0.0
(tiny, lich, faceless_void, omniknight, magnataur)
0.0
(phantom_lancer, kunkka, sniper, life_stealer, invoker)
0.0
(sand_king, windrunner, phantom_assassin, keeper_of_the_light, legion_com
mander)    0.0
(drow_ranger, vengefulspirit, life_stealer, invoker, terrorblade)
0.0
(bloodseeker, pudge, sniper, life_stealer, medusa)
0.0
(axe, crystal_maiden, tiny, obsidian_destroyer, ember_spirit)
0.0
(windrunner, luna, night_stalker, bounty_hunter, spirit_breaker)
0.0
(drow_ranger, lion, warlock, magnataur, shredder)
0.0
(shadow_shaman, faceless_void, furion, huskar, troll_warlord)
0.0
(drow_ranger, mirana, tidehunter, riki, luna)
0.0
(nevermore, vengefulspirit, riki, queenofpain, shredder)
0.0
(antimage, juggernaut, pudge, lion, shadow_demon)
0.0
(vengefulspirit, death_prophet, omniknight, doom_bringer, terrorblade)
0.0
(drow_ranger, nevermore, pudge, night_stalker, legion_commander)
0.0
(axe, slardar, sniper, huskar, troll_warlord)
0.0
(antimage, mirana, kunkka, lina, furion)
0.0
(phantom_lancer, phantom_assassin, lycan, troll_warlord, ember_spirit)
0.0
(earthshaker, faceless_void, shadow_demon, nyx_assassin, shredder)
```

```
0.0
(antimage, sand_king, zuus, lina, viper)
0.0
(crystal_maiden, mirana, omniknight, alchemist, slark)
0.0
(zuus, gyrocopter, invoker, keeper_of_the_light, slark)
0.0
(earthshaker, juggernaut, tinker, life_stealer, ember_spirit)
0.0
(lich, tinker, beastmaster, wisp, slark)
0.0
Length: 183091, dtype: float64
```

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import json
```

In [2]:
```
train = pd.read_csv('new_train.csv', header = None)
test = pd.read_csv('new_test.csv', header = None)
```

Introduction and Background

w/ literature

In [3]:
```
### TODO
```

Identify the data

Each row of the dataset is a single game with the following features (in the order in the vector):

1. Team won the game (1 or -1)
2. Cluster ID (related to location)
3. Game mode (eg All Pick)
4. Game type (eg. Ranked)
5. to end : Each element is an indicator for a hero. Value of 1 indicates that a player from team '1' played as that hero and '-1' for the other team. Hero can be selected by only one player each game. This means that each row has five '1' and five '-1' values.

In [4]:
```
### TODO basic EDA
```

Predictive task

In [5]:
```
### col_0
```

In [6]:
```
X_train = train.iloc[:, 4:]
y_train = train.iloc[:, 0].values
X_test = test.iloc[:, 4:]
y_test = test.iloc[:, 0].values
```

Data Preprocessing

In [7]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import FunctionTransformer
from sklearn.decomposition import PCA
```

In [8]:
```
left = X_train[X_train > 0].fillna(0).values
right = (-X_train[X_train < 0]).fillna(0).values
X_train = pd.DataFrame(np.hstack((left, right)))

left = X_test[X_test > 0].fillna(0).values
right = (-X_test[X_test < 0]).fillna(0).values
X_test = pd.DataFrame(np.hstack((left, right)))
```

In [34]:
```
pca = PCA(n_components=X_train.shape[1])
pca.fit(X_train)
X_train_pca = np.matmul(X_train, pca.components_.T).iloc[:, :40]
```

In [35]:
```
pca = PCA(n_components=X_test.shape[1])
pca.fit(X_test)
X_test_pca = np.matmul(X_test, pca.components_.T).iloc[:, :40]
```

Comparing Models_unsupervised_approach

KNN

In [43]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [86]:
```
knn = KNeighborsClassifier(n_neighbors = 2, weights="distance")
knn.fit(X_train_pca, y_train)
knn.score(X_test_pca, y_test)
```

Out[86]: 0.49429306911457577

In [50]:
```
kmean.score(X_train_pca, y_train)
```

Out[50]: 0.7558618177402707

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
clf = LogisticRegression(solver='lbfgs').fit(X_train_pca, y_train)
clf.score(X_test_pca, y_test)
```

Out[45]: 0.511195298460343

In [49]:
```
clf.score(X_train_pca, y_train)
```

Out[49]:
```
0.5567360815979601
```

Random Forest

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [47]:
```
rf = RandomForestClassifier(max_depth=2).fit(X_train_pca, y_train)
rf.score(X_test_pca, y_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWa
rning: The default value of n_estimators will change from 10 in version 0.20 to
100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[47]:
```
0.5267861479430764
```

In [48]:
```
rf.score(X_train_pca, y_train)
```

Out[48]:
```
0.527873231740635
```

Deep Learning

In [51]:
```
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.layers import Dense, Input, Activation, Concatenate, Embed
from tensorflow.keras.models import Model
```

In [62]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [65]:
```
onehot = OneHotEncoder(categories= 'auto')
y_train_t = onehot.fit_transform(y_train.reshape(-1, 1))
y_test_t = onehot.transform(y_test.reshape(-1, 1))
```

In [79]:
```
X_train_pca.shape
```

Out[79]:
```
(82355, 40)
```

In [80]:
```
def pred_model():

    x_input = Input(shape=[40,])
    x_dense = Dense(10)(x_input)
    x_acti = Activation("relu")(x_dense)

    final = Dense(2)(x_acti)
    final_acti = Activation("sigmoid")(final)

    model = Model([x_input], final_acti)

    model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accura

    return model
```

In [71]:
```
team1 = X_train.iloc[:, :113]
team2 = X_train.iloc[:, 113:]
```

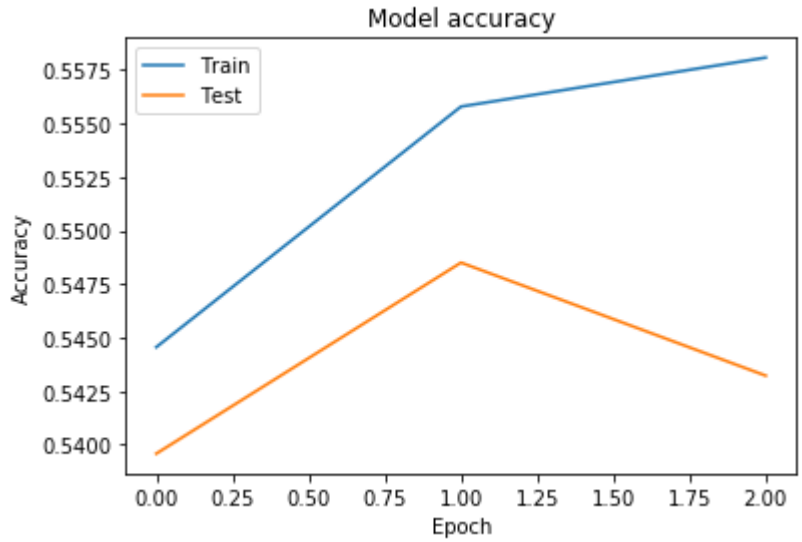
In [83]:
```
model = pred_model()
batch_size = 8
epoch = 3



history = model.fit(x = [X_train_pca], y=y_train_t, batch_size=batch_size, epochs
```

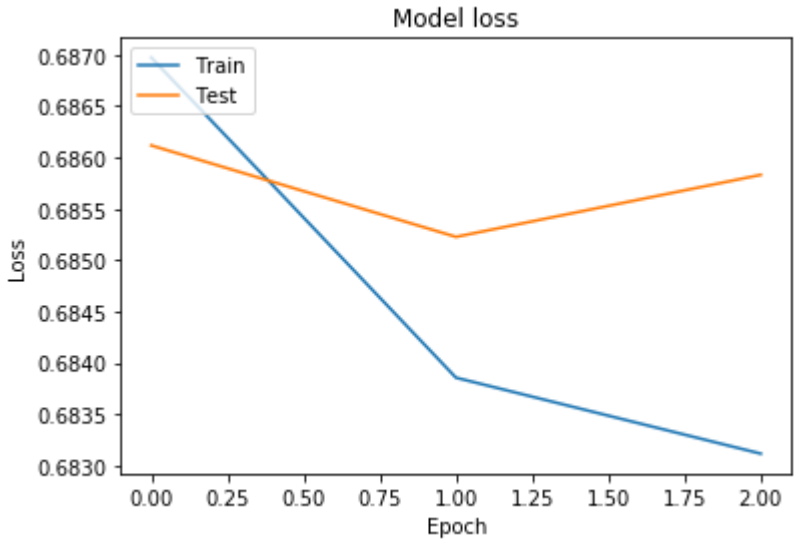
```
Train on 74119 samples, validate on 8236 samples
Epoch 1/3
74119/74119 [==============================] - 51s 688us/sample - loss: 0.6870
- acc: 0.5446 - val_loss: 0.6861 - val_acc: 0.5396
Epoch 2/3
74119/74119 [==============================] - 50s 676us/sample - loss: 0.6839
- acc: 0.5558 - val_loss: 0.6852 - val_acc: 0.5485
Epoch 3/3
74119/74119 [==============================] - 52s 708us/sample - loss: 0.6831
- acc: 0.5581 - val_loss: 0.6858 - val_acc: 0.5432
```

In [84]:
```
import matplotlib.pyplot as plt
```

In [85]:

```
# Plot training & validation accuracy values
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





Comparing Models_supervised_approach

Results and Conclusion

In [ ]:

In [17]:
```
import pandas as pd
import numpy as np
import json
import matplotlib.pyplot as plt
```

Load data

In [2]:
```
train = pd.read_csv('new_train.csv', header = None)
test = pd.read_csv('new_test.csv', header = None)
```

In [3]:
```
#col_0 ->  target
col_1 = pd.DataFrame(json.load(open('regions.json'))['regions'])
col_1 = col_1.set_index('id').iloc[:,0]
col_2 = pd.DataFrame(json.load(open('mods.json'))['mods'])
col_2 = col_2.set_index('id').iloc[:, 0]
col_3 = pd.DataFrame(json.load(open('lobbies.json'))['lobbies'])
col_3 = col_3.set_index('id').iloc[:, 0]
col_4 = pd.DataFrame(json.load(open('heroes.json'))['heroes'])
```

In [4]:
```
X_train = train.iloc[:, 1:]
y_train = train.iloc[:, 0].values
X_test = test.iloc[:, 1:]
y_test = test.iloc[:, 0].values
```

Preprocessing data

In [5]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import FunctionTransformer
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings("ignore")
```

In [6]:
```
X_train[1] = X_train[1].apply(lambda x: col_1[x] if x in col_1.index els
e str(x))
X_train[2] = X_train[2].apply(lambda x: col_2[x] if x in col_2.index els
e str(x))
X_train[3] = X_train[3].apply(lambda x: col_3[x] if x in col_3.index els
e str(x))

X_test[1] = X_test[1].apply(lambda x: col_1[x] if x in col_1.index else
str(x))
X_test[2] = X_test[2].apply(lambda x: col_2[x] if x in col_2.index else
str(x))
X_test[3] = X_test[3].apply(lambda x: col_3[x] if x in col_3.index else
str(x))
```

In [7]:
```
onehot = OneHotEncoder(categories='auto')
y_train = onehot.fit_transform(y_train.reshape(-1,1)).toarray()
y_test = onehot.fit_transform(y_test.reshape(-1,1)).toarray()
```

Models

Try with only heroes

naive

In [8]:
```
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.layers import Dense, Input, Activation, Concatenat
e, Embedding, LSTM
from tensorflow.keras.models import Model
```

In [9]:
```
onehot = OneHotEncoder(categories='auto', handle_unknown = 'ignore')

region_train = onehot.fit_transform(X_train.iloc[:,0].values.reshape(-1,
1)).toarray()
region_test = onehot.transform(X_test.iloc[:,0].values.reshape(-1, 1)).t
oarray()

onehot = OneHotEncoder(categories='auto', handle_unknown = 'ignore')
mode_train = onehot.fit_transform(X_train.iloc[:,1].values.reshape(-1, 1
)).toarray()
mode_test = onehot.transform(X_test.iloc[:,1].values.reshape(-1, 1)).toa
rray()

onehot = OneHotEncoder(categories='auto', handle_unknown = 'ignore')
type_train = onehot.fit_transform(X_train.iloc[:,2].values.reshape(-1, 1
)).toarray()
type_test = onehot.transform(X_test.iloc[:,2].values.reshape(-1, 1)).toa
rray()
```

In [10]:
```
hero_train = X_train.iloc[:, 3:].values
hero_test = X_test.iloc[:, 3:].values
```

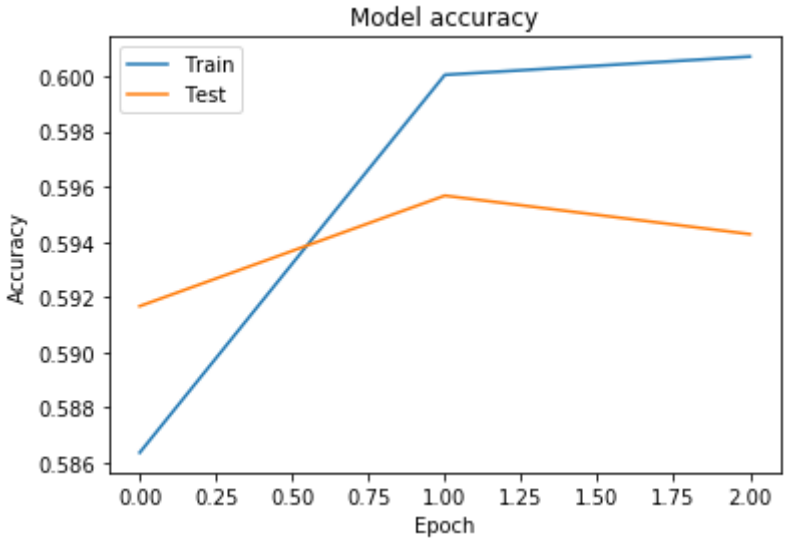
In [21]:
```
def pred_model():
    #hero_layer
    hero_input = Input(shape = [113,])
    hero_dense = Dense(10)(hero_input)
    hero_acti = Activation('relu')(hero_dense)
    #region_layer

    final = Dense(2, activation = 'sigmoid')(hero_acti)

    model = Model([hero_input], final)

    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model

model = pred_model()
batch_size = 8
epoch = 3

history= model.fit(x = [hero_train]\
          , y = y_train, batch_size = batch_size, epochs = epoch, valida
tion_split = .1)
```

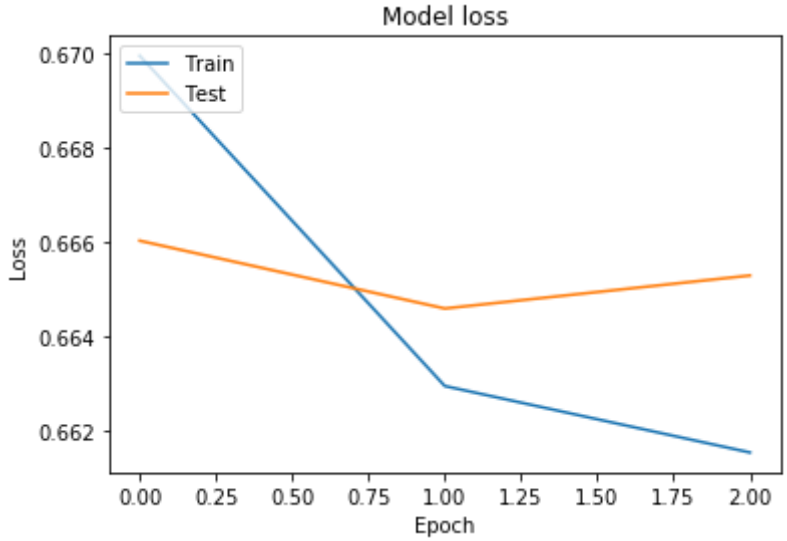
```
Train on 74119 samples, validate on 8236 samples
Epoch 1/3
74119/74119 [==============================] - 36s 486us/sample - loss:
0.6699 - acc: 0.5864 - val_loss: 0.6660 - val_acc: 0.5917
Epoch 2/3
74119/74119 [==============================] - 35s 476us/sample - loss:
0.6629 - acc: 0.6001 - val_loss: 0.6646 - val_acc: 0.5957
Epoch 3/3
74119/74119 [==============================] - 35s 469us/sample - loss:
0.6615 - acc: 0.6007 - val_loss: 0.6653 - val_acc: 0.5943
```

In [20]:
```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model.png')
```

```
Failed to import pydot. You must install pydot and graphviz for `pydotp
rint` to work.
```

In [22]:
```
# Plot training & validation accuracy values
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





In [23]:
```
y_prob = model.predict([ hero_test])
y_classes = y_prob.argmax(axis=-1)
((test.iloc[:, 0] == 1).astype(int) == y_classes).mean()
```

Out[23]:
```
0.5948807615717131
```

team splits

In [13]:
```
train_team_1  = (hero_train > 0).astype(int)
train_team_2 = (hero_train < 0).astype(int)

test_team_1  = (hero_test > 0).astype(int)
test_team_2 = (hero_test < 0).astype(int)
```

In [14]:
```
np.hstack((train_team_1, train_team_2))
```

Out[14]:
```
array([[0, 0, 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, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

In [15]:
```
def pred_model():
    #team1_layer
    team1_input = Input(shape = [113,])
    team1_dense = Dense(32, activation='relu')(team1_input)
    team1_dense = Dense(10, activation='relu')(team1_dense)
    #team2_layer
    team2_input = Input(shape = [113,])
    team2_dense = Dense(32, activation='relu')(team2_input)
    team2_dense = Dense(10, activation='relu')(team2_dense)


    conc = Concatenate()([team1_dense, team2_dense])
    conc = Dense(32, activation='relu')(conc)
    conc = Dense(10, activation='relu')(conc)


    final = Dense(2, activation = 'sigmoid')(conc)

    model = Model([team1_input, team2_input], final)

    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model

model = pred_model()
batch_size = 128
epoch = 3

model.fit(x = [train_team_1, train_team_2]\
          , y = y_train, batch_size = batch_size, epochs = epoch, valida
tion_split = .1)
```

```
Train on 74119 samples, validate on 8236 samples
Epoch 1/3
74119/74119 [==============================] - 6s 74us/sample - loss:
0.6766 - accuracy: 0.5708 - val_loss: 0.6688 - val_accuracy: 0.5871
Epoch 2/3
74119/74119 [==============================] - 5s 66us/sample - loss:
0.6619 - accuracy: 0.6018 - val_loss: 0.6676 - val_accuracy: 0.5912
Epoch 3/3
74119/74119 [==============================] - 5s 68us/sample - loss:
0.6571 - accuracy: 0.6094 - val_loss: 0.6676 - val_accuracy: 0.5910
```

Out[15]:
```
<tensorflow.python.keras.callbacks.History at 0x219fc3f29c8>
```

In [18]:
```
y_prob = model.predict([ test_team_1, test_team_2])
y_classes = y_prob.argmax(axis=-1)
((test.iloc[:, 0] == 1).astype(int) == y_classes).mean()
```

Out[18]:
```
0.5914808878527369
```

In [ ]:
```

```

noetbook-118b

December 3, 2019

[2]:
```
import pandas as pd
import numpy as np
import json
```

[3]:
```
train = pd.read_csv('new_train.csv', header = None)
test = pd.read_csv('new_test.csv', header = None)
```

0.0.1 Introduction and Background

w/ literature

[5]:
```
### TODO
```

0.0.2 Identify the data

Each row of the dataset is a single game with the following features (in the order in the vector): 1. Team won the game (1 or -1) 2. Cluster ID (related to location) 3. Game mode (eg All Pick) 4. Game type (eg. Ranked) 5. to end : Each element is an indicator for a hero. Value of 1 indicates that a player from team '1' played as that hero and '-1' for the other team. Hero can be selected by only one player each game. This means that each row has five '1' and five '-1' values.

[10]:
```
### TODO basic EDA
```

0.0.3 Predictive task

[11]:
```
### col_0
```

[4]:
```
X_train = train.iloc[:, 4:]
y_train = train.iloc[:, 0].values
X_test = test.iloc[:, 4:]
y_test = test.iloc[:, 0].values
```

0.0.4 Data Preprocessing

[5]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import FunctionTransformer
from sklearn.decomposition import PCA
```

[6]:
```
left = X_train[X_train > 0].fillna(0).values
right = (-X_train[X_train < 0]).fillna(0).values
X_train = pd.DataFrame(np.hstack((left, right)))

left = X_test[X_test > 0].fillna(0).values
right = (-X_test[X_test < 0]).fillna(0).values
X_test = pd.DataFrame(np.hstack((left, right)))
```

[7]:
```
pca = PCA(n_components=X_train.shape[1])
pca.fit(X_train)
X_train_pca = np.matmul(X_train, pca.components_.T).iloc[:,:40]
```

[8]:
```
pca = PCA(n_components=X_test.shape[1])
pca.fit(X_test)
X_test_pca = np.matmul(X_test, pca.components_.T).iloc[:,:40]
```

0.0.5 Comparing Models_unsupervised_approach

[16]:
```
from sklearn.mixture import GaussianMixture
```

[17]:
```
gm = GaussianMixture(n_components = 2, verbose = 2)
```

[19]:
```
gm.fit(X_train)
```

```
Initialization 0
  Iteration 10   time lapse 37.89073s     ll change 4.13150
Initialization converged: True   time lapse 41.78649s     ll 117.60433
```

[19]:
```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
                means_init=None, n_components=2, n_init=1, precisions_init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose=2,
                verbose_interval=10, warm_start=False, weights_init=None)
```

[22]:
```
y_pred = gm.predict(X_test)
```

[26]:
```
y_pred
```

[26]:
```
array([0, 0, 0, …, 1, 0, 0])
```

[33]:
```
tval = (y_test == 1).astype(int)
```

[34]:
```
pred = (y_pred == tval)
```

[35]:
```
pred.sum() / len(y_test)
```

[35]:
```
0.5086696779833891
```

0.0.6 Comparing Models_supervised_approach

[11]:
```
from sklearn.linear_model import Perceptron
clf = Perceptron(tol=1e-3, random_state=0)
clf.fit(X_train, y_train)
```

[11]:
```
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=1.0,
           fit_intercept=True, max_iter=1000, n_iter_no_change=5, n_jobs=None,
           penalty=None, random_state=0, shuffle=True, tol=0.001,
           validation_fraction=0.1, verbose=0, warm_start=False)
```

[12]:
```
clf.score(X_train, y_train)
```

[12]:
```
0.5427478598749317
```

[13]:
```
clf.score(X_test, y_test)
```

[13]:
```
0.544076934285298
```

[12]:
```
from sklearn import linear_model
br = linear_model.BayesianRidge()
```

[36]:
```
trainval = (y_train == 1).astype(int)
```

[37]:
```
br.fit(X_train, trainval)
```

[37]:
```
BayesianRidge(alpha_1=1e-06, alpha_2=1e-06, compute_score=False, copy_X=True,
              fit_intercept=True, lambda_1=1e-06, lambda_2=1e-06, n_iter=300,
              normalize=False, tol=0.001, verbose=False)
```

[38]:
```
br.score(X_train, trainval)
```

[38]:
```
0.06056253220440455
```

[39]:
```
br.score(X_test, tval)
```

[39]:
```
0.05397690680954936
```

[9]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

[9]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[11]:
```
gnb.score(X_test, y_test)
```

[11]:
```
0.5850211277866822
```

[47]:
```
prediction = gnb.predict(X_test)
```

[44]:
```
gnb.score(X_train, y_train)
```

[44]:
```
0.5878695889745613
```

[46]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
↪recall_score, classification_report, confusion_matrix
```

[53]:
```
precision_score(y_test, prediction), recall_score(y_test, prediction)
```

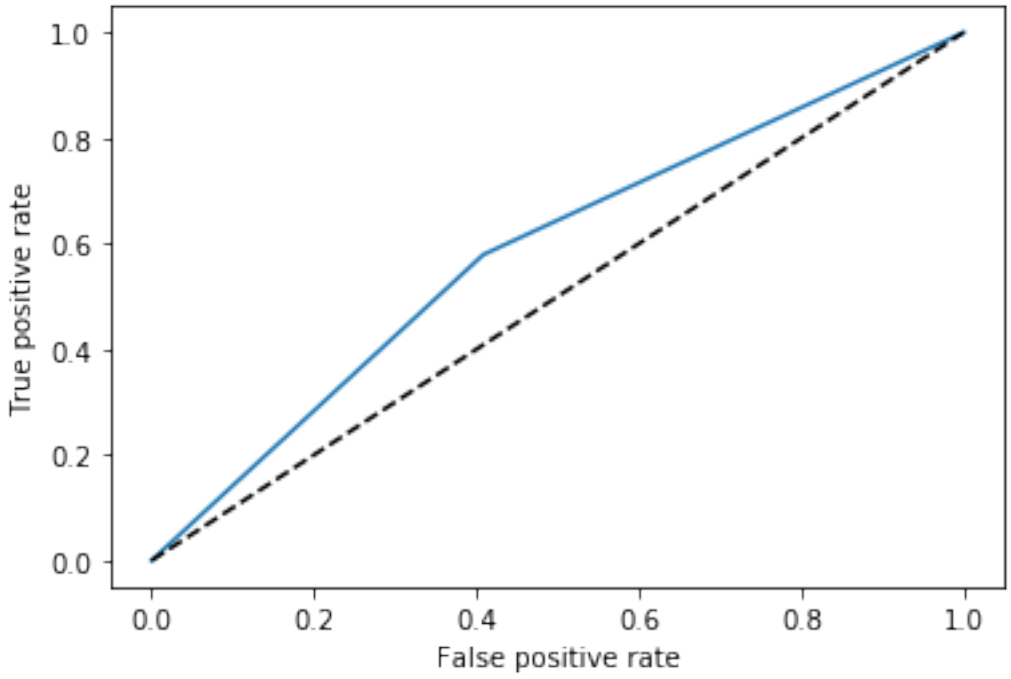
[53]:
```
(0.6121235747003216, 0.5791609036422315)
```

[56]:
```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
fpr, tpr, _ = roc_curve(y_test, prediction)
```

[56]:
```
[<matplotlib.lines.Line2D at 0x7f8c4c3ffbe0>]
```

[59]:
```
plt.plot(fpr, tpr, label='RT + LR')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
```

[59]:
```
Text(0, 0.5, 'True positive rate')
```



[41]:
```
from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB()
bnb.fit(X_train, y_train)
```

[41]:
```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

[43]:
```
bnb.score(X_test, y_test)
```

[43]:
```
0.5993005974063821
```

[45]:
```
bnb.score(X_train, y_train)
```

[45]:
```
0.6024042256086455
```

0.0.7 Results and Conclusion

[ ]:
```

```