In [41]:
```
import glob
import pandas
import json
import time
print('pandas',pandas.__version__)
import matplotlib.pyplot as plt
import numpy
print('numpy',numpy.__version__)
```

```
pandas 0.23.4
numpy 1.13.3
```

In [2]:
```
with open("voting_data/Santa_Fe/Santa Fe/2018/JsonFiles/CvrExport.json", "r") as read_file:
    data = json.load(read_file)
```

In [3]:
```
type(data)
```

Out[3]:
```
dict
```

In [4]:
```
data.keys()
```

Out[4]:
```
dict_keys(['Version', 'ElectionId', 'Sessions'])
```

In [5]:
```
print(data['Version'])
```

```
5.4.17.5
```

In [6]:
```
print(data['ElectionId'])
```

```
Santa Fe 2018 Municipal
```

In [7]:
```
type(data['Sessions'])
```

Out[7]:
```
list
```

In [8]:
```
len(data['Sessions'])
```

Out[8]:
```
20670
```

In [9]:
```
type(data['Sessions'][0])
```

Out[9]:
```
dict
```

In [10]:
```
data['Sessions'][0].keys()
```

Out[10]:
```
dict_keys(['TabulatorId', 'BatchId', 'RecordId', 'CountingGroupId', 'ImageMask', 'SessionType', 'VotingSessionIdentifier', 'UniqueVotingIdentifier', 'Original'])
```

In [11]:
```
data['Sessions'][0]
```

```
Out[11]: {'TabulatorId': 37,
          'BatchId': 0,
          'RecordId': 671489,
          'CountingGroupId': 3,
          'ImageMask': 'C:\\NAS\\Santa Fe 2018 Municipal\\Results\\Tabulator0003
         7\\Batch000\\Images\\00037_00000_671489*.*',
          'SessionType': 'ScannedVote',
          'VotingSessionIdentifier': '',
          'UniqueVotingIdentifier': '',
          'Original': {'PrecinctPortionId': 2,
           'BallotTypeId': 6,
           'IsCurrent': True,
           'Cards': [{'Id': 1006,
             'PaperIndex': 0,
             'Contests': [{'Id': 1,
               'Undervotes': 0,
               'Overvotes': 0,
               'OutstackConditionIds': [],
               'Marks': [{'CandidateId': 4,
                 'PartyId': 0,
                 'Rank': 1,
                 'MarkDensity': 93,
                 'IsAmbiguous': False,
                 'IsVote': True,
                 'OutstackConditionIds': []},
                {'CandidateId': 3,
                 'PartyId': 0,
                 'Rank': 2,
                 'MarkDensity': 98,
                 'IsAmbiguous': False,
                 'IsVote': False,
                 'OutstackConditionIds': []},
                {'CandidateId': 2,
                 'PartyId': 0,
                 'Rank': 3,
                 'MarkDensity': 99,
                 'IsAmbiguous': False,
                 'IsVote': False,
                 'OutstackConditionIds': []},
                {'CandidateId': 1,
                 'PartyId': 0,
                 'Rank': 4,
                 'MarkDensity': 99,
                 'IsAmbiguous': False,
                 'IsVote': False,
                 'OutstackConditionIds': []},
                {'CandidateId': 5,
                 'PartyId': 0,
                 'Rank': 5,
                 'MarkDensity': 99,
                 'IsAmbiguous': False,
                 'IsVote': False,
                 'OutstackConditionIds': []}]},
              {'Id': 3,
               'Undervotes': 0,
               'Overvotes': 0,
               'OutstackConditionIds': [],
```

```
                'Marks': [{'CandidateId': 8,
                  'PartyId': 0,
                  'Rank': 1,
                  'MarkDensity': 98,
                  'IsAmbiguous': False,
                  'IsVote': True,
                  'OutstackConditionIds': []},
                 {'CandidateId': 9,
                  'PartyId': 0,
                  'Rank': 2,
                  'MarkDensity': 98,
                  'IsAmbiguous': False,
                  'IsVote': False,
                  'OutstackConditionIds': []},
                 {'CandidateId': 10,
                  'PartyId': 0,
                  'Rank': 3,
                  'MarkDensity': 100,
                  'IsAmbiguous': False,
                  'IsVote': False,
                  'OutstackConditionIds': []}]}],
               'OutstackConditionIds': []}]}}
```

In [12]:
```
data['Sessions'][0]['Original'].keys()
```

Out[12]:
```
dict_keys(['PrecinctPortionId', 'BallotTypeId', 'IsCurrent', 'Cards'])
```

In [13]:
```
data['Sessions'][0]['Original']
```

```
Out[13]: {'PrecinctPortionId': 2,
          'BallotTypeId': 6,
          'IsCurrent': True,
          'Cards': [{'Id': 1006,
            'PaperIndex': 0,
            'Contests': [{'Id': 1,
              'Undervotes': 0,
              'Overvotes': 0,
              'OutstackConditionIds': [],
              'Marks': [{'CandidateId': 4,
                'PartyId': 0,
                'Rank': 1,
                'MarkDensity': 93,
                'IsAmbiguous': False,
                'IsVote': True,
                'OutstackConditionIds': []},
               {'CandidateId': 3,
                'PartyId': 0,
                'Rank': 2,
                'MarkDensity': 98,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []},
               {'CandidateId': 2,
                'PartyId': 0,
                'Rank': 3,
                'MarkDensity': 99,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []},
               {'CandidateId': 1,
                'PartyId': 0,
                'Rank': 4,
                'MarkDensity': 99,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []},
               {'CandidateId': 5,
                'PartyId': 0,
                'Rank': 5,
                'MarkDensity': 99,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []}]},
             {'Id': 3,
              'Undervotes': 0,
              'Overvotes': 0,
              'OutstackConditionIds': [],
              'Marks': [{'CandidateId': 8,
                'PartyId': 0,
                'Rank': 1,
                'MarkDensity': 98,
                'IsAmbiguous': False,
                'IsVote': True,
                'OutstackConditionIds': []},
               {'CandidateId': 9,
                'PartyId': 0,
```

```
                'Rank': 2,
                'MarkDensity': 98,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []},
               {'CandidateId': 10,
                'PartyId': 0,
                'Rank': 3,
                'MarkDensity': 100,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []}]}],
            'OutstackConditionIds': []}]}
```

In [14]:
```
data['Sessions'][0]['Original']['Cards']
```

```
Out[14]: [{'Id': 1006,
           'PaperIndex': 0,
           'Contests': [{'Id': 1,
             'Undervotes': 0,
             'Overvotes': 0,
             'OutstackConditionIds': [],
             'Marks': [{'CandidateId': 4,
               'PartyId': 0,
               'Rank': 1,
               'MarkDensity': 93,
               'IsAmbiguous': False,
               'IsVote': True,
               'OutstackConditionIds': []},
              {'CandidateId': 3,
               'PartyId': 0,
               'Rank': 2,
               'MarkDensity': 98,
               'IsAmbiguous': False,
               'IsVote': False,
               'OutstackConditionIds': []},
              {'CandidateId': 2,
               'PartyId': 0,
               'Rank': 3,
               'MarkDensity': 99,
               'IsAmbiguous': False,
               'IsVote': False,
               'OutstackConditionIds': []},
              {'CandidateId': 1,
               'PartyId': 0,
               'Rank': 4,
               'MarkDensity': 99,
               'IsAmbiguous': False,
               'IsVote': False,
               'OutstackConditionIds': []},
              {'CandidateId': 5,
               'PartyId': 0,
               'Rank': 5,
               'MarkDensity': 99,
               'IsAmbiguous': False,
               'IsVote': False,
               'OutstackConditionIds': []}]},
            {'Id': 3,
             'Undervotes': 0,
             'Overvotes': 0,
             'OutstackConditionIds': [],
             'Marks': [{'CandidateId': 8,
               'PartyId': 0,
               'Rank': 1,
               'MarkDensity': 98,
               'IsAmbiguous': False,
               'IsVote': True,
               'OutstackConditionIds': []},
              {'CandidateId': 9,
               'PartyId': 0,
               'Rank': 2,
               'MarkDensity': 98,
               'IsAmbiguous': False,
```

```
                'IsVote': False,
                'OutstackConditionIds': []},
               {'CandidateId': 10,
                'PartyId': 0,
                'Rank': 3,
                'MarkDensity': 100,
                'IsAmbiguous': False,
                'IsVote': False,
                'OutstackConditionIds': []}]}],
             'OutstackConditionIds': []}]
```

In [15]:
```
len(data['Sessions'][0]['Original']['Cards'])
```

Out[15]: 1

In [16]:
```
data['Sessions'][0]['Original']['Cards'][0]['Contests']
```

```
Out[16]: [{'Id': 1,
           'Undervotes': 0,
           'Overvotes': 0,
           'OutstackConditionIds': [],
           'Marks': [{'CandidateId': 4,
             'PartyId': 0,
             'Rank': 1,
             'MarkDensity': 93,
             'IsAmbiguous': False,
             'IsVote': True,
             'OutstackConditionIds': []},
            {'CandidateId': 3,
             'PartyId': 0,
             'Rank': 2,
             'MarkDensity': 98,
             'IsAmbiguous': False,
             'IsVote': False,
             'OutstackConditionIds': []},
            {'CandidateId': 2,
             'PartyId': 0,
             'Rank': 3,
             'MarkDensity': 99,
             'IsAmbiguous': False,
             'IsVote': False,
             'OutstackConditionIds': []},
            {'CandidateId': 1,
             'PartyId': 0,
             'Rank': 4,
             'MarkDensity': 99,
             'IsAmbiguous': False,
             'IsVote': False,
             'OutstackConditionIds': []},
            {'CandidateId': 5,
             'PartyId': 0,
             'Rank': 5,
             'MarkDensity': 99,
             'IsAmbiguous': False,
             'IsVote': False,
             'OutstackConditionIds': []}]},
          {'Id': 3,
           'Undervotes': 0,
           'Overvotes': 0,
           'OutstackConditionIds': [],
           'Marks': [{'CandidateId': 8,
             'PartyId': 0,
             'Rank': 1,
             'MarkDensity': 98,
             'IsAmbiguous': False,
             'IsVote': True,
             'OutstackConditionIds': []},
            {'CandidateId': 9,
             'PartyId': 0,
             'Rank': 2,
             'MarkDensity': 98,
             'IsAmbiguous': False,
             'IsVote': False,
             'OutstackConditionIds': []},
```

```
{'CandidateId': 10,
 'PartyId': 0,
 'Rank': 3,
 'MarkDensity': 100,
 'IsAmbiguous': False,
 'IsVote': False,
 'OutstackConditionIds': []}]}]
```

In [17]:
```
len(data['Sessions'][0]['Original']['Cards'][0]['Contests'])
```

Out[17]: 2

summarize the relevant information

In [18]:
```
print('ballot ID:',data['Sessions'][0]['Original']['Cards'][0]['Id'])
```

```
ballot ID: 1006
```

In [19]:
```
print('contest ID:',data['Sessions'][0]['Original']['Cards'][0]['Contest
s'][0]['Id'])
pandas.DataFrame(data['Sessions'][0]['Original']['Cards'][0]['Contests']
[0]['Marks'])
```

```
contest ID: 1
```

Out[19]:

	CandidateId	IsAmbiguous	IsVote	MarkDensity	OutstackConditionIds	PartyId	Rank
0	4	False	True	93	[]	0	1
1	3	False	False	98	[]	0	2
2	2	False	False	99	[]	0	3
3	1	False	False	99	[]	0	4
4	5	False	False	99	[]	0	5

In [20]:
```
print('contest ID:',data['Sessions'][0]['Original']['Cards'][0]['Contest
s'][1]['Id'])
pandas.DataFrame(data['Sessions'][0]['Original']['Cards'][0]['Contests']
[1]['Marks'])
```

```
contest ID: 3
```

Out[20]:

	CandidateId	IsAmbiguous	IsVote	MarkDensity	OutstackConditionIds	PartyId	Rank
0	8	False	True	98	[]	0	1
1	9	False	False	98	[]	0	2
2	10	False	False	100	[]	0	3

loop over all ballots

In [21]:
```
start_time= time.time()
reslts={}
empty_ballot_count={}
invalid_ballot_count={}
invalid_ranking={}
for sess in data['Sessions']:
    if len(sess['Original']['Cards'])>1:
        print('duplicate card: ',len(sess['Original']['Cards']))
    else:
        for contest in sess['Original']['Cards'][0]['Contests']:
            df = pandas.DataFrame(contest['Marks'])
            if (df.shape[0]==0 and df.shape[1]==0): # empty ballot -- no
marks made
                try:
                    empty_ballot_count[contest['Id']]
                except KeyError:
                    empty_ballot_count[contest['Id']]=[]
                empty_ballot_count[contest['Id']].append(sess['ImageMas
k'])
            elif len(set([y for x in list(df['OutstackConditionIds']) fo
r y in x]))>1: # errors in ballot
                #print('invalid entries in',sess['ImageMask'])
                invalid_df = df
                try:
                    invalid_ballot_count[contest['Id']]
                except KeyError:
                    invalid_ballot_count[contest['Id']]=[]
                invalid_ballot_count[contest['Id']].append(sess['ImageMa
sk'])
            elif len(df)>6:
                print('more than 6 entries for',sess['ImageMask'])
                excess_df = df
                try:
                    invalid_ballot_count[contest['Id']]
                except KeyError:
                    invalid_ballot_count[contest['Id']]=[]
                invalid_ballot_count[contest['Id']].append(sess['ImageMa
sk'])
            elif (len(df[df['Rank']==1]['CandidateId'].values)!=1):
                #print('invalid number of first ranked candidates for',s
ess['ImageMask'])
                try:
                    invalid_ranking[contest['Id']]
                except KeyError:
                    invalid_ranking[contest['Id']]=[]
                invalid_ranking[contest['Id']].append(sess['ImageMask'])
            else: # ballot exists and does not have errors
                try:
                    reslts[contest['Id']]
                except KeyError:
                    reslts[contest['Id']]=[]
                reslts[contest['Id']].append(df)

print('elapsed',round(time.time()-start_time,2),'seconds')
```

```
elapsed 148.45 seconds
```

In [22]:
```
invalid_df
```

Out[22]:

	CandidateId	IsAmbiguous	IsVote	MarkDensity	OutstackConditionIds	PartyId	Rank
0	9	False	True	100	[]	0	1
1	8	False	False	98	[9]	0	2
2	9	False	False	91	[12, 9]	0	2

In [23]:
```
# https://spapas.github.io/2016/04/27/python-nested-list-comprehensions/
set([y for x in list(invalid_df['OutstackConditionIds']) for y in x])
```

Out[23]: {9, 12}

In [24]:
```
print('valid results')
for k,v in reslts.items():
    print('election',k,':',len(v))
```

```
valid results
election 1 : 20237
election 3 : 6217
election 2 : 6767
election 4 : 1410
election 5 : 4834
```

In [25]:
```
print('empty votes')
for k,v in empty_ballot_count.items():
    print('election',k,':',len(v))
```

```
empty votes
election 4 : 530
election 3 : 173
election 2 : 447
election 5 : 173
election 1 : 40
```

In [26]:
```
print('invalid results')
for k,v in invalid_ballot_count.items():
    print('election',k,':',len(v))
```

```
invalid results
election 5 : 55
election 1 : 356
election 3 : 33
```

In [34]:
```
print('invalid results')
for k,v in invalid_ranking.items():
    print('election',k,':',len(v))
```

```
invalid results
election 1 : 37
election 3 : 10
election 2 : 3
election 5 : 18
```

In [27]:
```
number_of_votes_cast = [len(vote_df) for vote_df in reslts[1]]
```

In [28]:
```
min(number_of_votes_cast)
```

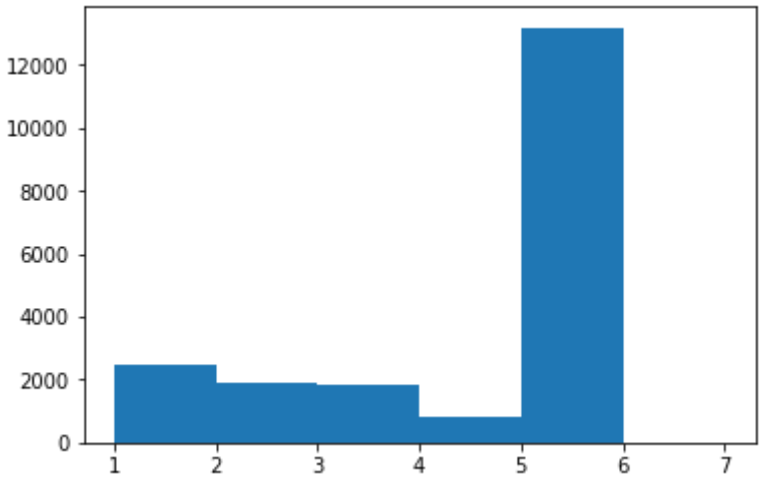
Out[28]: 1

In [29]:
```
max(number_of_votes_cast)
```

Out[29]: 5

In [30]:
```
plt.hist(number_of_votes_cast,bins=[1,2,3,4,5,6,7])
```

Out[30]:
```
(array([  2499.,   1875.,   1837.,    840.,  13186.,      0.]),
 array([1, 2, 3, 4, 5, 6, 7]),
 <a list of 6 Patch objects>)
```



categorize first round for an election

In [31]:
```
df = reslts[1][0]
df
```

Out[31]:

	CandidateId	IsAmbiguous	IsVote	MarkDensity	OutstackConditionIds	PartyId	Rank
0	4	False	True	93	[]	0	1
1	3	False	False	98	[]	0	2
2	2	False	False	99	[]	0	3
3	1	False	False	99	[]	0	4
4	5	False	False	99	[]	0	5

In [32]:
```
df[df['Rank']==1]['CandidateId'].values[0]
```

Out[32]: 4

In [33]:
```
start_time=time.time()
which_candidate=[]
for vote_df in reslts[1]:
    list_of_first_rank_candidates = vote_df[vote_df['Rank']==1]['Candida
teId'].values
    if len(list_of_first_rank_candidates)==1:
        which_candidate.append(list_of_first_rank_candidates[0])
    else:
        print('number of first rank candidates=',len(list_of_first_rank_
candidates))
        print(vote_df)

print('elapsed',round(time.time()-start_time,2),'seconds')
```

```
elapsed 24.73 seconds
```

In [35]:
```
len(which_candidate)
```

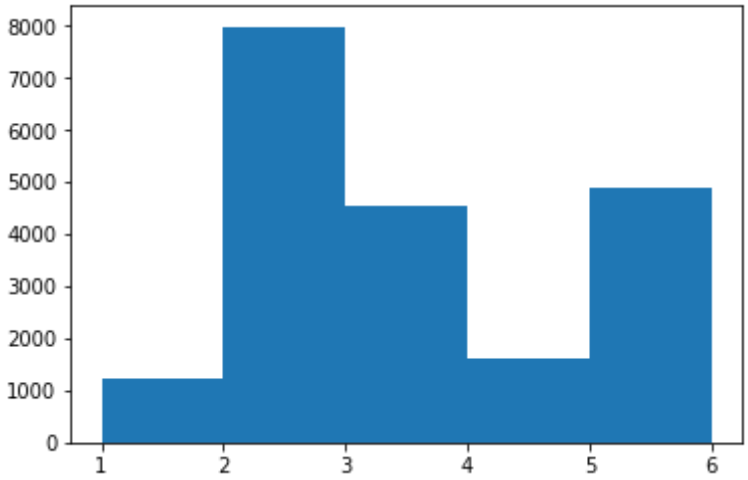
Out[35]: 20237

In [36]:
```
which_candidate[0:10]
```

Out[36]: [4, 2, 4, 3, 2, 2, 2, 2, 3, 2]

In [39]:
```
plt.hist(which_candidate,bins=[1,2,3,4,5,6])
```

Out[39]:
```
(array([ 1212.,  7985.,  4541.,  1620.,  4879.]),
 array([1, 2, 3, 4, 5, 6]),
 <a list of 5 Patch objects>)
```



In [46]:
```
values, counts = numpy.unique(which_candidate, return_counts=True)
print('values',values)
print('counts',counts)
print('total number of votes=',sum(counts))
list(zip(values,counts))
```

```
values [1 2 3 4 5]
counts [1212 7985 4541 1620 4879]
total number of votes= 20237
```

Out[46]:
```
[(1, 1212), (2, 7985), (3, 4541), (4, 1620), (5, 4879)]
```

In [49]:
```
percentages = [x/sum(counts) for x in counts]
percentages
```

Out[49]:
```
[0.05989029994564412,
 0.39457429460888471,
 0.22439096704056927,
 0.080051391016455004,
 0.24109304738844689]
```

Objective for this notebook: separate the elections into the following categories:

1. Leading candidate in the first round has greater than 50% first choice votes
2. Leading candidate in the first round has between 45-50% first choice votes
3. Leading candidate in the first round has less than 45% of first choice votes

In [52]:
```
def which_cat(percentages):
    if max(percentages)>0.5:
        print("Leading candidate in the first round has greater than 50% first choice votes")
    elif max(percentages)<=0.5 and max(percentages)>=0.45:
        print("Leading candidate in the first round has between 45-50% first choice votes")
    elif max(percentages)<0.45:
        print("Leading candidate in the first round has less than 45% of first choice votes")
    else:
        raise Exception("invalid outcome")
    return
```

In [53]:
```
which_cat(percentages)
```

```
Leading candidate in the first round has less than 45% of first choice votes
```

categorize first round for each election

In [60]:
```
start_time=time.time()
which_candidate=[]
for election_id,election in reslts.items():
    for vote_df in election:
        list_of_first_rank_candidates = vote_df[vote_df['Rank']==1]['CandidateId'].values
        if len(list_of_first_rank_candidates)==1:
            which_candidate.append(list_of_first_rank_candidates[0])
        else:
            print('number of first rank candidates=',len(list_of_first_rank_candidates))
            print(vote_df)
    values, counts = numpy.unique(which_candidate, return_counts=True)
    percentages = [x/sum(counts) for x in counts]
    which_cat(percentages)
print('elapsed',round(time.time()-start_time,2),'seconds')
```

```
Leading candidate in the first round has less than 45% of first choice votes
Leading candidate in the first round has less than 45% of first choice votes
Leading candidate in the first round has less than 45% of first choice votes
Leading candidate in the first round has less than 45% of first choice votes
Leading candidate in the first round has less than 45% of first choice votes
elapsed 47.06 seconds
```