Mining Pinterest

Social Media Mining CSE 400

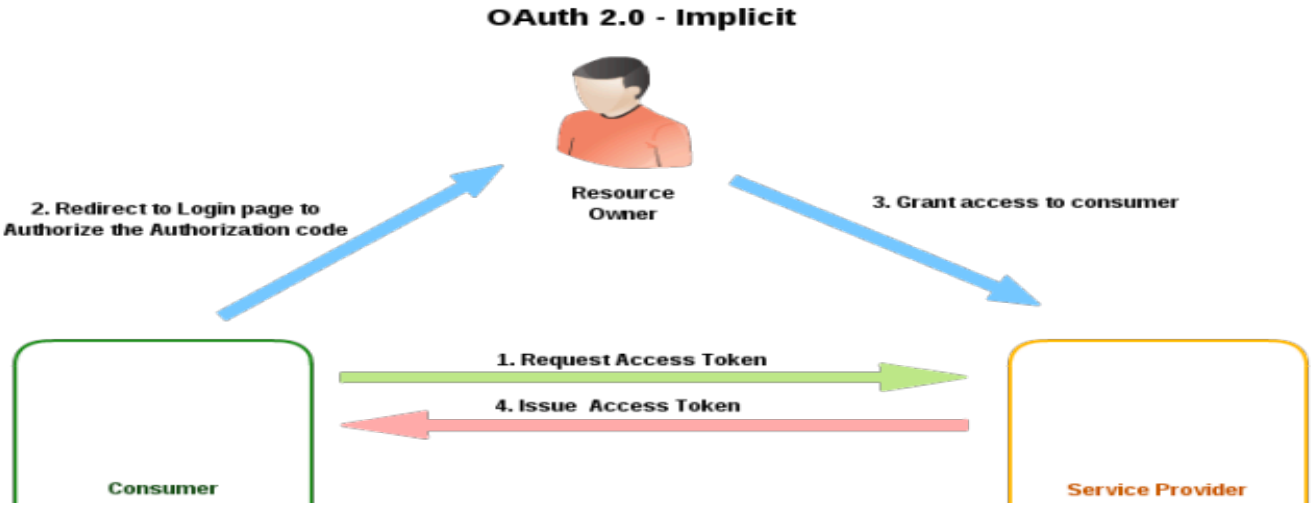
Tatiana Ensslin | May 4, 2016

Introduction

MinePin makes API requests to the Pinterests API, which uses OAUTH 2.0. There are 7 data mining functions, which mine the pin count, creation date, and username of a specific board. These board categories are: Bunnies, Cat, Dog, Summer, Winter, Fall, and Spring. It requires an Access Token to allow for any data mining or API requests to be made. This means that to data mine an individual user's board, one must get their authentication by having them sign into their account.To get a personal Access Token, necessary for MinePin, one can find it at https://developers.pinterest.com/tools/access_token/. Because of this limitation, MinePin requires MANUAL data collection of each user's username. These can be found for Bunnies, Summer, Winter, Spring, and Fall through file I/O. The files have been provided in this folder. Cat and Dog username data can be found in a List under each specific function definition. Mine Pin contains three Pinterest API request functions, each providing different data. These include: returnUserInfo(), getBoardInfo(), retrievePins(). MinePin also contains a parser() helper function, which intakes the date and time in isoformat and parses it to the YMD without any spaces.

OAUTH 2.0:

Pinterest uses OAUTH 2.0, which requires each user to give access, including yourself in order to retrieve the access token necessary for mining. This means that if one would like to data mine a user's Pinterest data other than their own, they would need an access token validated for both the user and the developer who is mining. Access tokens can be obtained by registering an App with the Pinterest Developer website. Below is a picture that shows the OAUTH 2.0 process, followed by a description if one were to use it with Pinterest:



Step 1/2. Get authorization from your user. The program will redirect the user to Pinterest and ask for their permission to read or change their account.

Step 3. Get an authorization code. If your user approves your request, you'll receive temporary credentials (known as an authorization code) for your user's Pinterest account.

Step 4. Exchange for an access token. The program will call the API to exchange the authorization code for an access token, which is a permanent credential (unless the user revokes access). You'll use the access token to perform actions on Pinterest on your user's behalf.

Pinterest API Request Functions:

MinePin makes API requests to the Pinterest API. Required for the requests are an access token, a user name, and a board name (category). All requests begin with the path: https://api.pinterest.com/v1/. MinePin contains three GET API requests. These are in the function returnUserInfo(), getBoardInfo(), and retrievePins().

```
def returnUserInfo(user):

    r = urllib2.urlopen("https://api.pinterest.com/v1/users/"+user+"/?access_token="+access_token+"fields=first_name%2Cid%2Clast_name%2Curl%2
    data = json.load(r)
    return data
    #print r

# Retrieves the boards information, such as URL, date of creation, and pin count ..etc
def getBoardInfo(user,board):

    r = urllib2.urlopen("https://api.pinterest.com/v1/boards/"+user+"/"+board+"/?access_token="+access_token+"fields=id%2Cname%2Curl%2Ccounts
    data = json.load(r)
    return data

# Retrieves the information about all pins on a board, meaning title, URL, creation, id .. etc
def retrievePins(user,board):

    r = urllib2.urlopen("https://api.pinterest.com/v1/boards/"+user+"/"+board+"/pins/?access_token="+access_token+"fields=id%2Clink%2Cnote%2C
    data = json.load(r)
    return data
```

Each of these functions require a username (user), as well as the token, however, to retrieve any information about the pins on the board, or get information about a board, a board name is required. The fields attributes are the different information you are allowed to mine off of that account/board. Adding and removing them control the amount of data received. Each request is called and then converted to JSON, in order to be parsed for data handling.

Data Mining Functions:

There are 7 data mining functions, which mine the pin count and creation date of a specific username and a specific board. MinePin requires MANUAL data collection of each user's username. These user names are provided and can be found for Bunny, Summer, Winter, Spring, and Fall through file I/O. Cat and Dog username data can be found in a List under each specific function definition.

```
def bunnies(): #preform data mining on BUNNIES category
    with open('bunnies.txt', 'r') as f: #read in 150 usernames from file
        bunniesUsers = [line.rstrip('\n') for line in f] #save into List

    bunnyPins = [] #collects pin count per board

    t=0
    for x in range(0, 150):
        data = getBoardInfo(bunniesUsers[t],'bunnies') #search 150 users' boards
        print t
        print bunniesUsers[t]
        print "Bunny pins:"
        print data["data"]["counts"]["pins"] #enter data and counts to print pin
        print "Bunnies board created at:"
        date = data["data"]["created_at"]
        dateOnly = parser(date)
        bunnyPins.append([dateOnly,data["data"]["counts"]["pins"],bunniesUsers[t]])
        t=t+1

    print bunnyPins[0:150] #print out all 150 users' information

    sortedBunnies = sorted(bunnyPins,key=itemgetter(0)) #sort the creation dates
    print "sortedBunnies", sortedBunnies

    #save data to a file to be opened in excel and plotted
    with open('bunnydata.txt', 'w') as outfile:
        s = str(sortedBunnies)
        s = s.split("], [") #parse between each tuple to create a new line
        json.dump(s, outfile , indent = 3)
    return sortedBunnies
```

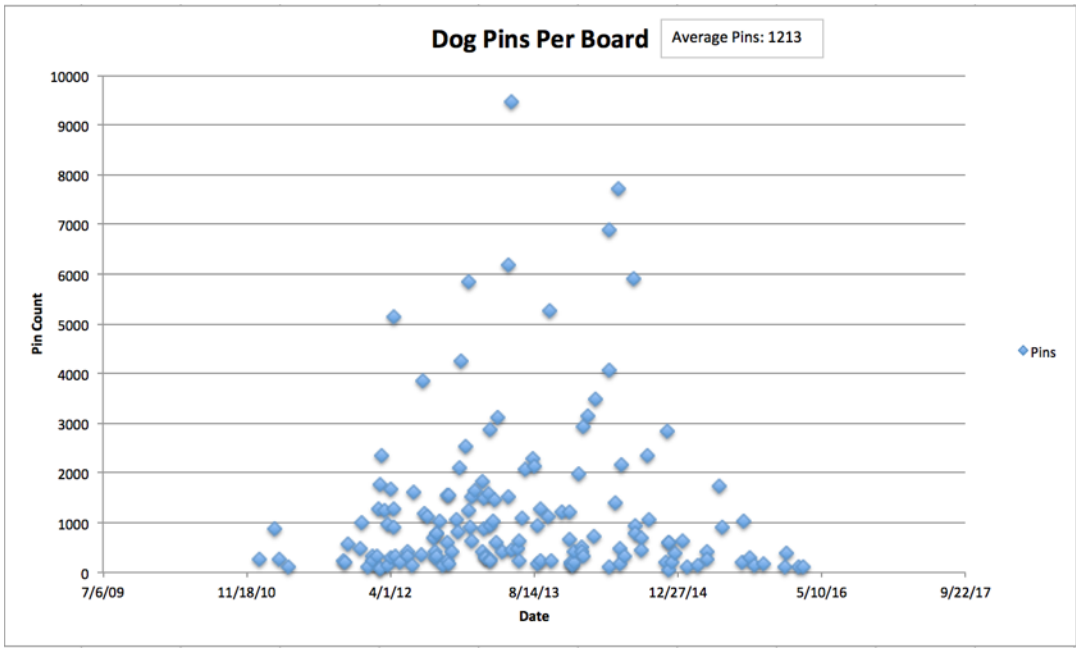
Seen left is the bunnies() function. It opens the file filled with usernames, creates a list to store the pin count and data in, and then for all 150 usernames in the file, it calls the getBoardInfo() request function for a board names “bunnies.” It then uses JSON to parse the data down to pins and creation date. The date is then taken in by a helper function which parses the date to YYYY,MM,DD format to be stored with the pin count and username in a tuple in a list. This list is then sorted by date from oldest to youngest. The list (sortedBunnies) is then stored, one tuple per line, in a file named bunnydata.txt, which will be created in the current directory.

Each of the seven functions work in this manner, using an API request function to pull in data, parse the data, and then handle the data so that it will be saved into a .txt file ready for Excel formatting.

Results:

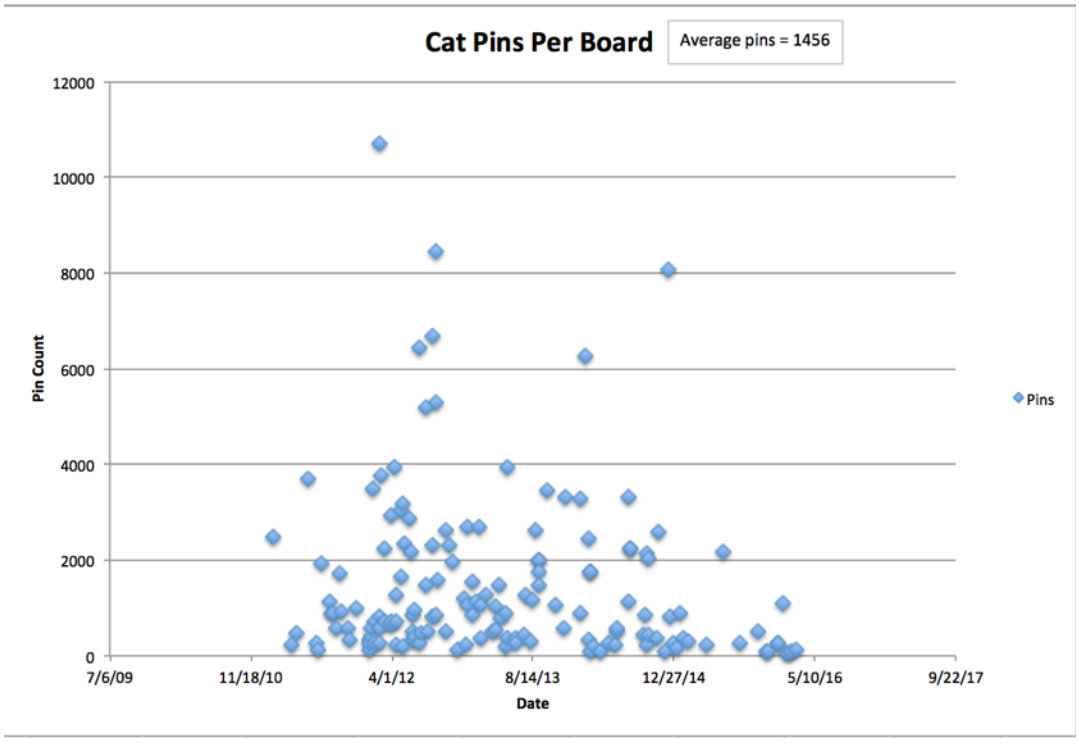
The following section contains a graphical representation of all data mined per category:

- **Dog:** Highly congested 2012-2014. Average pins: 1213.

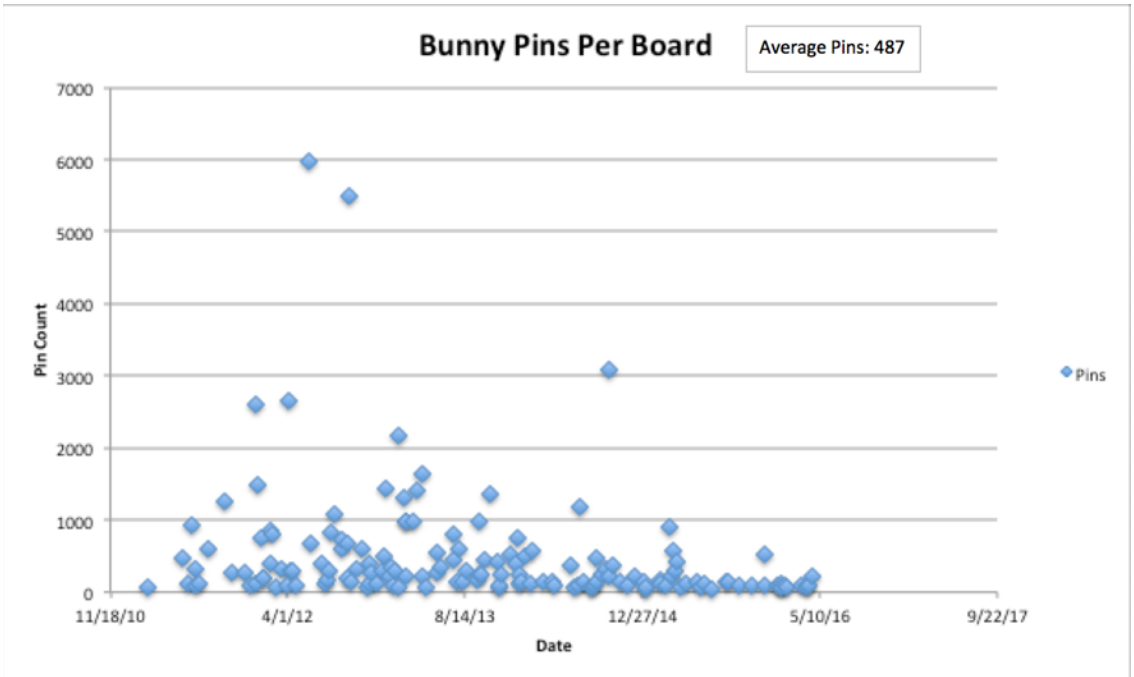


-

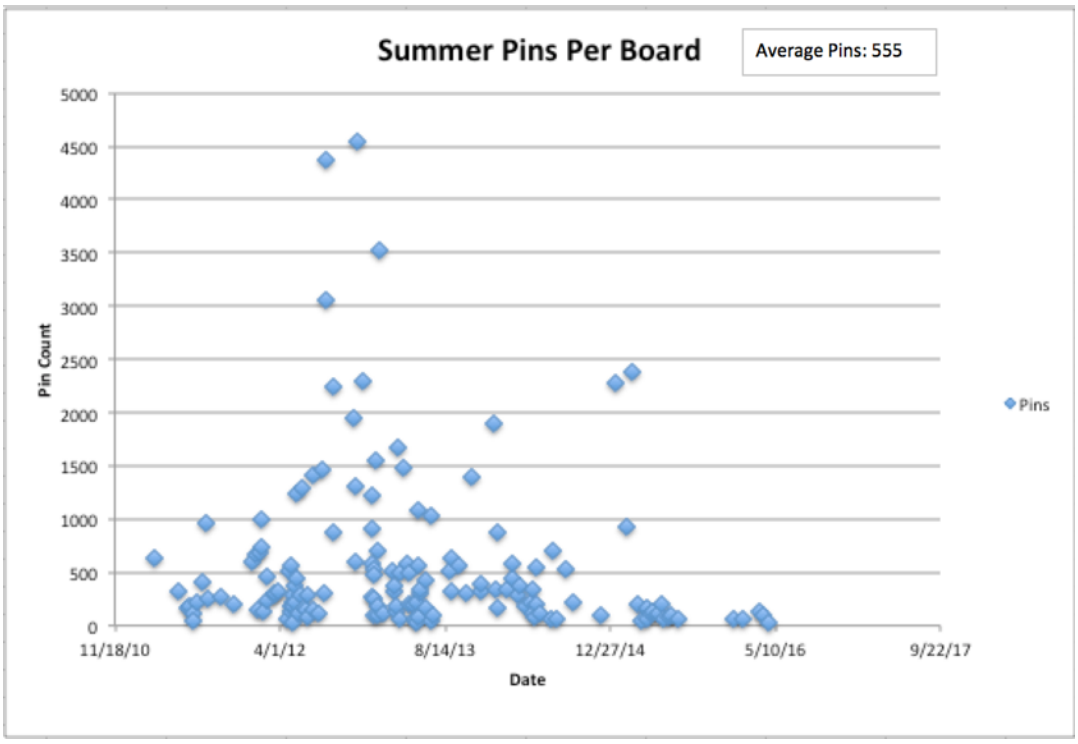
- **Cat:** Highly congested 2012-2013. Moderately congested 2014. Average pins: 1456.



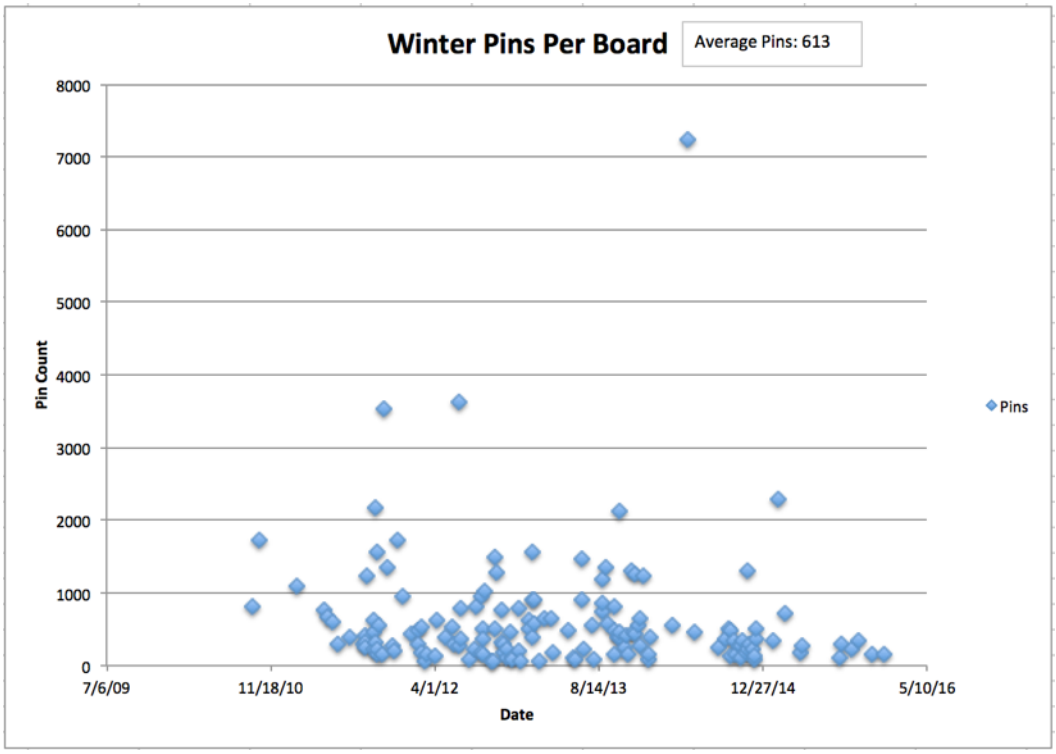
- **Bunnies:** Consistently pinned through 2010-2016. Average pins: 487.



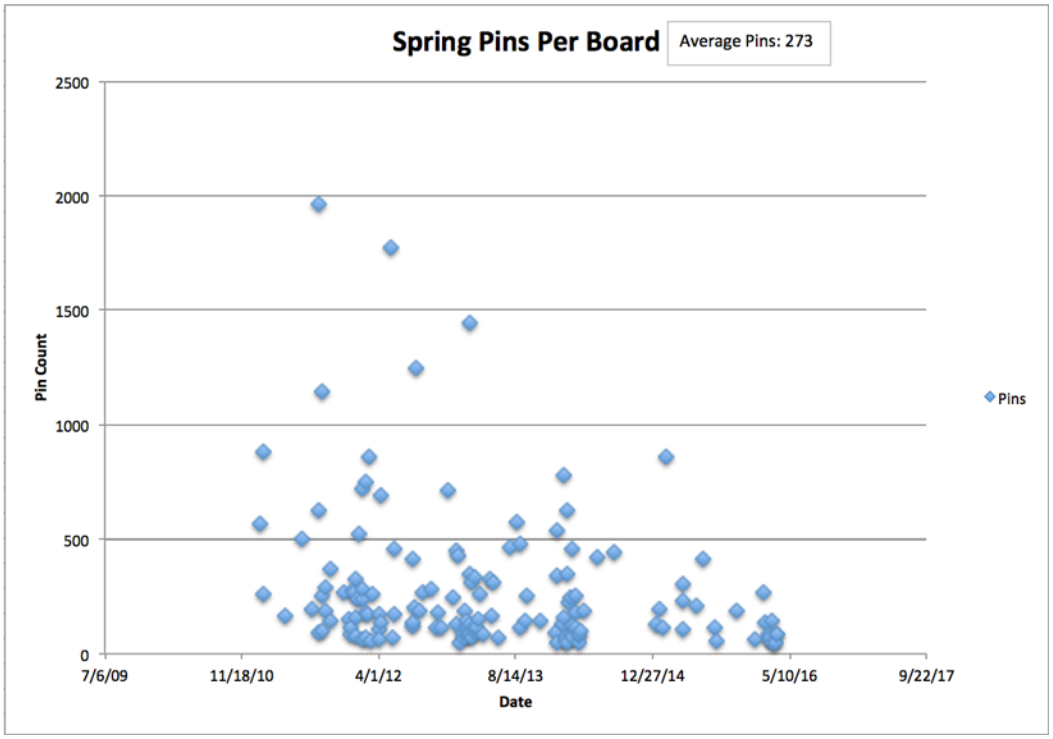
- **Summer:** High congestion 2012-6/2014. Average pins 555.



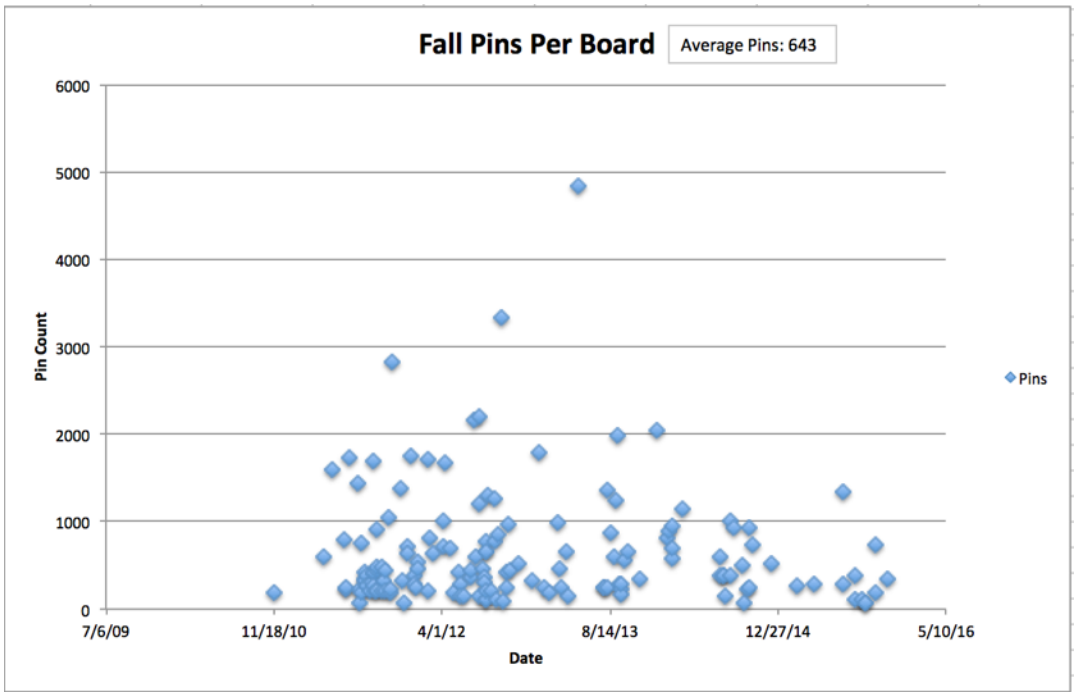
- **Winter:** Steady congestion from 2011-2013. Picks up in Winter 2014. Average pins: 613.



- **Spring:** Lightly congested. Picks up 4/2012-10/2013. Dies in Winter of 2013. Resumes lightly Spring 2014, 2015, and moderately in Spring 2016. Average pins: 273.

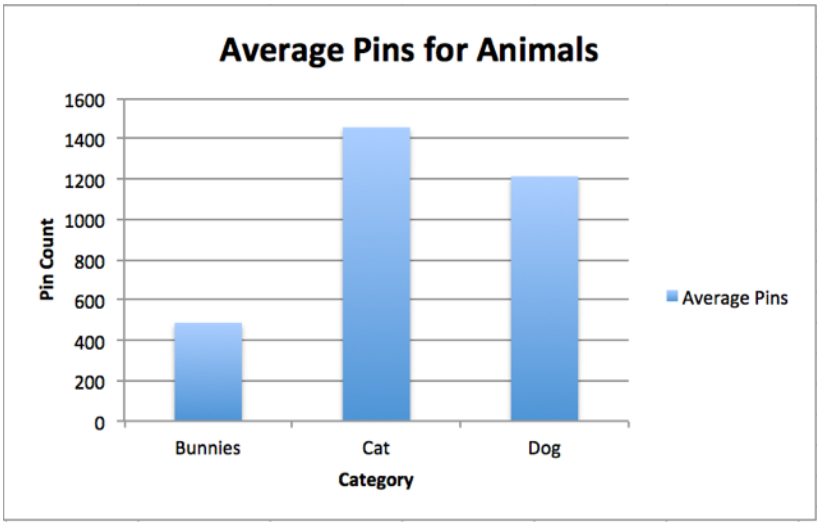


- **Fall:** Begins heavily congested in 2011 through 2012. Lightly congested from 2013 till present day. Average pins: 643.

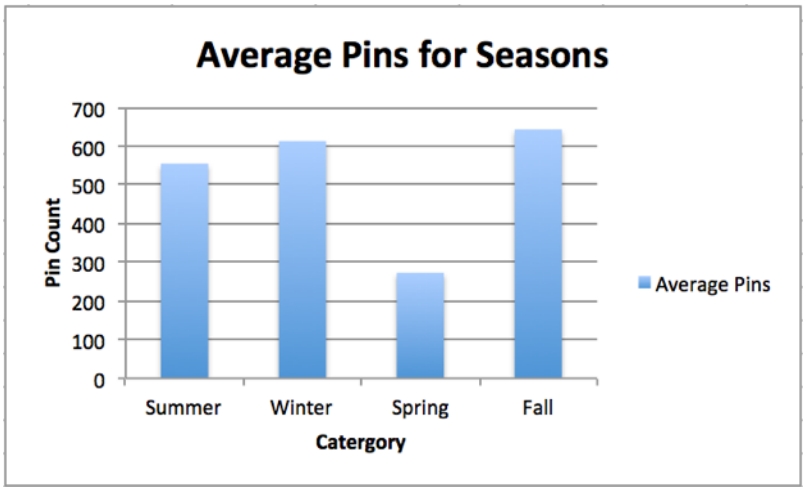


Discussion:

There are 7 data groups, split into three animals and the four seasons. For the animals group, based on average pin count, most users slightly prefer to post about cats (1456 pins) than dogs (1213 pins), with bunnies falling behind by more than less than almost half the amount of pins (487 pins):



As for the four seasons, Fall was pinned the most (643 pins), followed by Winter (613 pins), Summer (555 pins), and then Spring (273 pins). This can be seen graphically below as well:



Interestingly enough, most Summer boards were created through the months of December to March, die off in April, and pick up again beginning in May until August each year. This data could point to the fact that most people are thinking about the Summer year long, anticipating for its arrival and then posting about its adventures through its months.

As for Winter boards, most were created from August until the end of December, with very few being created from January to August each year. This data also could be explaining how beginning in August, most people look forward to the winter season as it is only one season away. After Winter begins, people are over the hype of the snow and look forward to Spring.

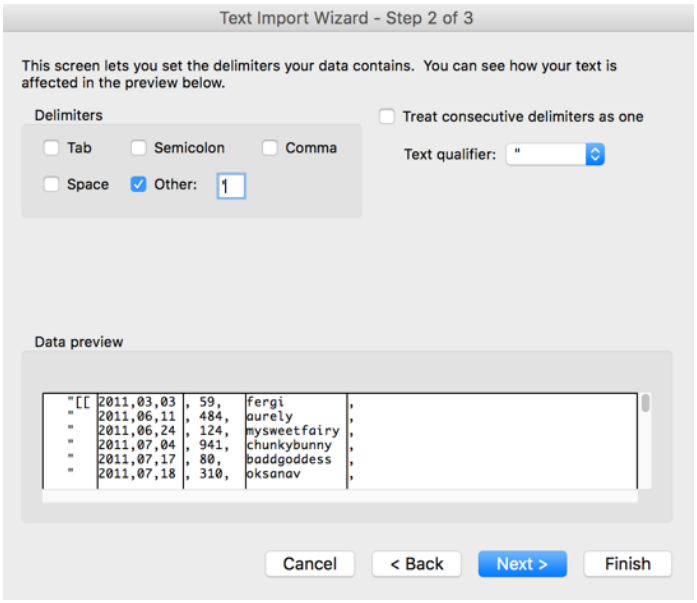
Furthermore, Spring boards are mostly created in the months of February and March, and some in May. They tend to be created just weeks before the season begins in February. Additionally, according to the data, Spring is the most underrepresented and posted season overall.

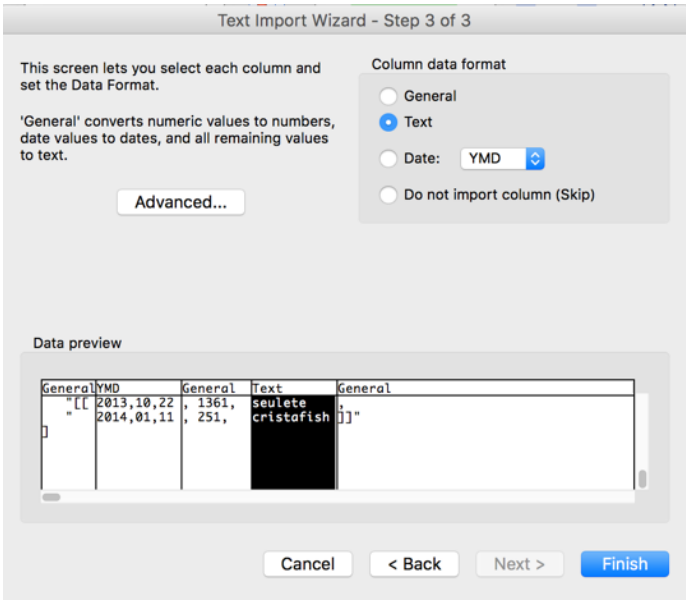
Lastly, Fall boards tend to be created in August and September, with a few being created in October and November. Overall, there is a pattern that boards tend to be created one month before each season begins. This could be because people begin to prepare, become excited, and are looking into things related to said season a month before it begins.

All of this data is most likely representing just the Northern Hemisphere, where the seasons are Summer (June-August), Fall (September-Novermber), Winter (December-February), and Spring (March-May).

Excel Parsing and Charting:

Once the data files were created, they were exported to Excel in order to be charted for data trends and patterns. In order to do this, the date was placed in a specific format of YYYY,MM,DD which is selected in Excel and reformatted into MM/DD/YYYY. This allows us to chart the pins over time from oldest to newest. To parse the data, delimiters are set in the Text Import Wizard for the ' character. This separates the tuple into three sections.





However, as you can see above, the pin count contains two commas. These are removed using the replace function in excel one the data is imported in, replacing "," with "". On the last step of the Text Import Wizard, highlight the dates and set them in the column data format of YMD, and text for the usernames.

Once the data is pulled into excel, shift the date and pin cells down one, allowing space to create a Series header for each: Date and Pins. Highlight all the cells (B2:B151-C2:C151) and select scatter chart. Add axis headers. Now the chart is ready to be used for data analysis.

Appendix:

MinePin.py

```
# -*- coding: utf-8 -*-

# A Python script that data mines Pinterest.com using OAUTH 2.0 access token and GET requests.
# Created by Tatiana Ensslin
# April 23rd, 2016
# Social Media Mining


import urllib2 #for urlopen
import sys
import json
import time
from dateutil.parser import parse #to parse the date
from datetime import date
from operator import itemgetter #to sort
from datetime import datetime
from time import gmtime, strftime
from datetime import datetime, timedelta, date

#Pinterest useds AOUTH 2.0 .. aqcuire access token through token generator under app development
access_token='Ae3zx8MeHSlYVUJs2S861AU19fAeFEh8FWEFUTRDCv0QnYAt5AAAAAA&'

# Returns any user's information including the first name, last name, id, URL
def returnUserInfo(user):

        r = urllib2.urlopen("https://api.pinterest.com/v1/users/"+user+"/?access_token="+access_token+"fields=first_name%2Cid%2Clast_name%2Curl%2Ccreated_at")
        data = json.load(r)
        return data
        #print r

# Retrieves the boards information, such as URL, date of creation, and pin count ..etc
def getBoardInfo(user,board):

        r = urllib2.urlopen("https://api.pinterest.com/v1/boards/"+user+"/"+board+"/?access_token="+access_token+"fields=id%2Cname%2Curl%2Ccounts%2Ccreated_at")
        data = json.load(r)
        return data

# Retrieves the information about all pins on a board, meaning title, URL, creation, id .. etc
def retrievePins(user,board):

        r = urllib2.urlopen("https://api.pinterest.com/v1/boards/"+user+"/"+board+"/pins/?access_token="+access_token+"fields=id%2Clink%2Cnote%2Curl")
        data = json.load(r)
        return data

def parser(date): #a function that takes in the date as a YY-MM-DDTHH:MM:SS and parses it to recieve just YY-MM-DD
        p = parse(date) #parse date
        p = str(p) #convert date to string
        parsedDate = p.split(" ", 1) #split the date and time
        dateOnly = parsedDate[0] #retreive the date only
        #dateOnly.replace('/', '-')
        dateOnly = dateOnly.replace("-","")
        print dateOnly
        dateArray = []
        for i in dateOnly:
                dateArray.append(i)
        #print dateArray
        dateArray2 = [] #create a temp array to add in ',' for dates. this is CRUCIAL for excel data charting
        dateArray2.append(dateArray[0])
        dateArray2.append(dateArray[1])
        dateArray2.append(dateArray[2])
        dateArray2.append(dateArray[3])
        dateArray2.append(",")
        dateArray2.append(dateArray[4])
        dateArray2.append(dateArray[5])
        dateArray2.append(",")
        dateArray2.append(dateArray[6])
        dateArray2.append(dateArray[7])
        dateOnly = "".join(dateArray2)
```

```
		return dateOnly

def dog(): #preform data mining on DOGS category

		# dogUsers contains 150 list of users with boards named dogs
		dogUsers =
['playbuzz','paola_gambetti','birob','funnyordie','leasheffield','chicoulino','kittyklan','gwylfa','carinageuther','enmibolso','thubtnguyen','doublecloth','newfinish','warda278','justintpalmer','alisaquint','alari
design','christelkelz','lisskis','carleencoulter','retrogoddess','whippets4ever','muttimamma','angelicasara','wunderworker','sagegardencon','ellenbrok','apricou','Peggylholland','fetchsilversp','paperandstyl
eco','born2rock1974','mamapenny','caitlin_cawley','equip2survive','StephLaughlin','luvleo10','gabbiandrade','fineanddapper','neyselima','eziosgrandma','kinaps','judioliver','AlessandrD','bibibibibibibib',
'monikan754','l0503','espritk','avlogginggirl','karolinabeaudet','marrycullen','runintoflowers','johnsparkssandf','Meggssmithh','littlelargo','TheButtonMaker','gracie_baldwin','nellieviloria','carol1977','co
quidv','nds3535','enchantedmuse','fuchia57','wendi_gratz','ozbarb','willemijn26','jonathan930','fritela','deboernicole','msjoni56','hrtmb','thedreamerco','wilko2412','mllenorma','brandyo71','paigenyna','f
edericandco','disneymagic','freeallorcas','mmbfotografia','yannbros','sextoysestore','almapropia','philip1227','Souyzee','plainsimplehome','creddie','Hoosierpin84','charisseandrews','tay7lor','shaunashaw'
,'debmalewski','nancylilypad','unodedos','namolio','emilyvvhite0637','hella49','meryltruett','thehananaa','HerPoisonMind','leerburg','MuffinMerriam','dfco','vionavandijk','jurakoncius','resmarie','zbrow
n115','wallinbuerkle','peppermintsplay','julier','marybethlarson','jannyroo','trendswb','POPSUGARPets','ivabrozkova','crescentmoon06','ameliamims','iminclover','equestrianco','Scarrick23','vall','blank
2325','ginavalerio','montanameador','filmingbear321','shadowdog','loldamn','englandathomeuk','kateuhry','bioyoshimoto','czarchild51','esdesigns','ticktock7','ahirose','rocioespina','cuudulieutrsang','em
perornorton47','shorterdigital','sue_aylesbury','jaxonmike','agneteemilie','lesaiz','shaehopkins','FaithOConnor9','ltree21','JeanneMelanson','gtrzilla','lizziekailey','dolerf','cnccookbook']
		dogPins = []#collects pin count per board

		t=0
		for x in range(0, 150):
			data = getBoardInfo(dogUsers[t],'dogs') #search 150 users' boards who all have a board titled dogs
			print t
			print dogUsers[t]
			print "Dog pins:"
			print data["data"]["counts"]["pins"] #enter data and counts to print pin
			print "Dogs board created at:"
			date = data["data"]["created_at"]
			dateOnly = parser(date)
			dogPins.append([dateOnly,data["data"]["counts"]["pins"],dogUsers[t]]) #store a list of [created date, pins, user] in a tuple
			t=t+1

		print dogPins[0:150] #print out all 150 users' information
		sortedDogs = sorted(dogPins,key=itemgetter(0)) #sort the creation dates
		print "sortedDogs", sortedDogs

		#save data to a file to be opened in excel and plotted
		with open('dogdata.txt', 'w') as outfile:
			s = str(sortedDogs)
			s = s.split("], [") #parse between each tuple to create a new line
			json.dump(s, outfile , indent = 3)
		return sortedDogs

def cat(): #preform data mining
		catUsers =
['paola_gambetti','juwely','pribomilcar','laineyok','warda278','joyceeuverman','GGBeautifulLife','ozbarb','elizspinohza','sapperlott','11je12je','christelkelz','genevievekoenig','cisnerosmtz','LebenTina','stre
etform','amandaonwriting','teenwitches','honeybea1982','muttimamma','enmibolso','funnyordie','whippets4ever','BrokenDarkenss','perpetualkid','popupshop','applering','minandaminanda','liten','Fran
kieZeus','asemar','peterfwisser','socovina','theforestideas','SandraMeier','unodedos','cynthikelly','maryelizadoane','fetchsilversp','Souyzee','swtharvestmoon','SnoodleDoodles','jwmchristie','samegarr','A
nkeWeckmann','retrogoddess','swallowtail8','archidaniels','deboernicole','Busy_Beesz','AlessandrD','karosj','Hanatirusato','keyen2','kenderfrau','letilor','babayaga1','carlavangalen','alaridesign','kasemier
','MarciaCarrillo','queenzofia','InAustralia','judioliver','nuria1966','2lifestyle','morejoea','conquette','catsandkittys','l0503','bethany_s','sirenasnails','eziosgrandma','gracie_baldwin','queenmanu','thedream
erco','Seamssewcool','enid','avlogginggirl','swipurr','label29','nancydooren','iraco','mullenbry','lilmsjess','mixzublue','margot59','mzpeach','glutenfreemaman','noraburns','Turnike','avlogginggirl','paulam
ildon','mixzublue','Lazygirlsbeauty','tessalee','minimimi18','Meggssmithh','chihuahuabites','Eljy','eziosgrandma','fritela','arielmeier','susanelise','inyrdreemz','catslikeus','thcrazyteaparty','motleycrafter','j
uhbueno','zatras','FunnyBratt','napaneedlepoint','ltblogged','pamgreer','nuria1966','hopihapi','ravenlost','mllenorma','littlelargo','fofysfactory','hgaller','jcphoto','candilou31','TanuLee','Hoosierpin84','wil
lemijn26','msjoni56','marier24','shotbyshola','mamacitalopez62','imreagnes','jogo10','emilyvvhite0637','karakulia','swsana','misshenriques','junyatate','ofallonlibrary','ticktock7','FaithOConnor9','catlove
r24','thepinksamurai','enchantedmuse','bunchou2','teresasoares','hella49','aylinkilic','marilynfranko','kittykataqua','paola_sigal']
		catPins = []#collects pin count per board

		t=0
		for x in range(0, 150):
			data = getBoardInfo(catUsers[t],'cats') #search 150 users' boards
			print t
			print catUsers[t]
			print "Cat pins:"
			print data["data"]["counts"]["pins"] #enter data and counts to print pin
			print "Cats board created at:"
			date = data["data"]["created_at"]
			dateOnly = parser(date)
			catPins.append([dateOnly,data["data"]["counts"]["pins"],catUsers[t]]) #store a list of [created date, pins, user] in a tuple
			t=t+1

		print catPins[0:150] #print out all 150 users' information

		sortedCats = sorted(catPins,key=itemgetter(0)) #sort the creation dates
```

```
		print "sortedCats", sortedCats

		#save data to a file to be opened in excel and plotted
		with open('catdata.txt', 'w') as outfile:
			s = str(sortedCats)
			s = s.split('], [') #parse between each tuple to create a new line
			json.dump(s, outfile , indent = 3)

		return sortedCats

def getDateTimeFromISO8601String(s):
    d = dateutil.parser.parse(s)
    return d

def bunnies(): #preform data mining on BUNNIES category
		with open('bunnies.txt', 'r') as f: #read in 150 usernames from file
				bunniesUsers = [line.rstrip('\n') for line in f] #save into List

		bunnyPins = [] #collects pin count per board

		t=0
		for x in range(0, 150):
			data = getBoardInfo(bunniesUsers[t],'bunnies') #search 150 users' boards
			print t
			print bunniesUsers[t]
			print "Bunny pins:"
			print data["data"]["counts"]["pins"] #enter data and counts to print pin
			print "Bunnies board created at:"
			date = data["data"]["created_at"]
			dateOnly = parser(date)
			bunnyPins.append([dateOnly,data["data"]["counts"]["pins"],bunniesUsers[t]]) #store a list of [created date, pins, user] in a tuple
			t=t+1

		print bunnyPins[0:150] #print out all 150 users' information

		sortedBunnies = sorted(bunnyPins,key=itemgetter(0)) #sort the creation dates
		print "sortedBunnies", sortedBunnies

		#save data to a file to be opened in excel and plotted
		with open('bunnydata.txt', 'w') as outfile:
			s = str(sortedBunnies)
			s = s.split('], [') #parse between each tuple to create a new line
			json.dump(s, outfile , indent = 3)
		return sortedBunnies

def summer():
		with open('summer.txt', 'r') as f: #read in 150 usernames from file
				summerUsers = [line.rstrip('\n') for line in f] #save into List

		summerPins = [] #collects pin count per board

		t=0
		for x in range(0, 150):
			data = getBoardInfo(summerUsers[t],'summer') #search 150 users' boards
			print t
			print summerUsers[t]
			print "Summer pins:"
			print data["data"]["counts"]["pins"] #enter data and counts to print pin
			print "Summer board created at:"
			date = data["data"]["created_at"]
			dateOnly = parser(date)
			summerPins.append([dateOnly,data["data"]["counts"]["pins"],summerUsers[t]]) #store a list of [created date, pins, user] in a tuple
			t=t+1

		print summerPins[0:150] #print out all 150 users' information

		sortedSummer = sorted(summerPins,key=itemgetter(0)) #sort the creation dates
		print "sortedSummer", sortedSummer

		#save data to a file to be opened in excel and plotted
		with open('summerdata.txt', 'w') as outfile:
			s = str(sortedSummer)
```

```
                s = s.split('"], ["') #parse between each tuple to create a new line
                json.dump(s, outfile , indent = 3)
        return sortedSummer

def winter(): #preform data mining
        with open('winter.txt', 'r') as f: #read in 150 usernames from file
                winterUsers = [line.rstrip('\n') for line in f] #save into List

        winterPins = [] #collects pin count per board

        t=0
        for x in range(0, 150):
                data = getBoardInfo(winterUsers[t],'winter') #search 150 users' board
                print t
                print winterUsers[t]
                print "winter pins:"
                print data["data"]["counts"]["pins"] #enter data and counts to print pin
                print "winter board created at:"
                date = data["data"]["created_at"]
                dateOnly = parser(date)
                winterPins.append([dateOnly,data["data"]["counts"]["pins"],winterUsers[t]]) #store a list of [created date, pins, user] in a tuple
                t=t+1

        print winterPins[0:150] #print out all 150 users' information

        sortedWinter = sorted(winterPins,key=itemgetter(0)) #sort the creation dates
        print "sortedWinter", sortedWinter

        #save data to a file to be opened in excel and plotted
        with open('winterdata.txt', 'w') as outfile:
                s = str(sortedWinter)
                s = s.split('"], ["') #parse between each tuple to create a new line
                json.dump(s, outfile , indent = 3)
        return sortedWinter

def fall(): #preform data mining
        with open('fall.txt', 'r') as f: #read in 150 usernames from file
                fallUsers = [line.rstrip('\n') for line in f] #save into List

        fallPins = [] #collects pin count per board

        t=0
        for x in range(0, 150):
                data = getBoardInfo(fallUsers[t],'fall') #search 150 users' boards who all have a board titled dogs
                print t
                print fallUsers[t]
                print "fall pins:"
                print data["data"]["counts"]["pins"] #enter data and counts to print pin
                print "fall board created at:"
                date = data["data"]["created_at"]
                dateOnly = parser(date)
                fallPins.append([dateOnly,data["data"]["counts"]["pins"],fallUsers[t]]) #store a list of [created date, pins, user] in a tuple
                t=t+1

        print fallPins[0:150] #print out all 150 users' information

        sortedFall = sorted(fallPins,key=itemgetter(0)) #sort the creation dates
        print "sortedFall", sortedFall

        #save data to a file to be opened in excel and plotted
        with open('falldata.txt', 'w') as outfile:
                s = str(sortedFall)
                s = s.split('"], ["') #parse between each tuple to create a new line
                json.dump(s, outfile , indent = 3)
        return sortedFall

def spring(): #preform data mining
        with open('spring.txt', 'r') as f: #read in 150 usernames from file
                springUsers = [line.rstrip('\n') for line in f] #save into List

        springPins = [] #collects pin count per board
```

```
		t=0
		for x in range(0, 150):
			data = getBoardInfo(springUsers[t],'spring') #search 150 users' boards
			print t
			print springUsers[t]
			print "spring pins:"
			print data["data"]["counts"]["pins"] #enter data and counts to print pin
			print "spring board created at:"
			date = data["data"]["created_at"]
			dateOnly = parser(date)
			springPins.append([dateOnly,data["data"]["counts"]["pins"],springUsers[t]]) #store a list of [created date, pins, user] in a tuple
			t=t+1

		print springPins[0:150] #print out all 150 users' information

		sortedSpring = sorted(springPins,key=itemgetter(0)) #sort the creation dates
		print "sortedSpring", sortedSpring


		#save data to a file to be opened in excel and plotted
		with open('springdata.txt', 'w') as outfile:
			s = str(sortedSpring)
			s = s.split("], [") #parse between each tuple to create a new line
			json.dump(s, outfile , indent = 3)
		return sortedSpring

def main():

		#Mine different types of boards for pin counts under the following ANIMALS. Save data to a txt file to be parsed and displayed in excel
		dog()
		cat()
		bunnies()

		#Mine all four seasons for pin count. Save data to a txt file to be parsed and displayed in excel.
		#summer()
		#winter()
		#spring()
		#fall()


if __name__ == "__main__":# main()
    import sys
    main()
```