**Business Trip Report**

**Scipy 2019 austin, texas  
(SCIPY: Python-based ecosystem of open-source software for  
mathematics, science, and engineering)**

**July 2019**

**EUNJAE JANG**

**Quant TEAM**

****

|  |
| --- |
| Scipy Conference Schedule |

* **Period : 2019. 7. 8 (Mon) ~ 2019. 7. 12 (Fri), 6-day**
* **Attending : Eunjae Jang Sr. Manager, Quant Team**
* **Location : AT&T Convention Center, Austin, Texas**
* **Event : Scipy 2019 Conference**

|  |
| --- |
| Summary |

* + **Background**Within Python data science community, Scipy is the largest annual gathering community for developing open-source data analytics tools. During the event, topics such as machine learning, mathematics, computation efficiency, and data science engineering that cover all fields of science and commercial applications were conversed while various knowledge sharing activities were held for similar-interest practitioners.

Since the ecosystem of open-source data analytics software is continually evolving to better serve practitioners in relevant fields including finance, such event provides ample opportunities to learn new data science trends and novel methodologies that help extract insights from data.

* + **Conference Overview**  
    This conference was held with a 2-day tutorial session and a 3-day presentation session by practitioners in academia and corporations such as Nvidia, Microsoft, and IBM.  
    4 attending tutorials were: Bayesian Statistics Made Simple (Allen Downey), Bayesian Data Science Simulation (Eric Ma), Hands-On Satellite Imagery Analysis (Sara Safavi), and  
    Network Analysis Made Simple (Eric Ma)

Good amount of topics emphasized the importance for showing good data visualization with added interactive functions to better understand and communicate data. Most other talks showcased new tools or Scipy ecosystem best-practice usages. Also methods for speeding up research processes and enhancing computation were discussed. Lastly some talks discussed about ethical responsibility for data scientists in an era of human mimicking machines.

* + **Thoughts after Conference**Efficient programming and key-understanding of mathematical fundamentals are key-ingredient skillsets for data analysts, to say the least for financial quant specialists. Another important element is to continually be exposed to new ideas from other practitioners and learn new methods to understand data from a different angle.

Throughout this business trip, I’ve learned a great deal on the real-world applications with Bayesian statistics. This is a mathematical field which is generally known to many graduate-level engineers, but at the same time it is not widely used in many fields including finance. Bayesian statistics is different form conventional statistics in that it uses probability to represent uncertainty in any event or hypothesis. Due to this nature, the inference from Bayesian statistics can be much richer in data and also account for limited assumptions such as assuming normal distribution. However, the downside is that computational complexity grows substantially.

Thanks to tools such as PyMC, Bayesian statistical analysis can be reproduced very easily and multi-thread computational capability from local machines can be fully utilized. In a field such as finance where uncertainty holds great importance, figuring ways to apply Bayesian methods will be a fun challenge for KIC Quant.

Scipy conference has also touched upon topics like data cleansing, which accounts for nearly 50% of Quant research time consumption. Normally when we try to draw useful information from data, data-cleansing can become a great burden for tasks such as ticker-mapping, variable ordering, processing missing values and extreme values. Some useful packages like PyJanitor are good examples which could speed up the process for data cleansing.

Other useful topics were dashboard visualization, efficient multi-task computations, and network analysis methodologies.

* **Order of Topic Summaries**

Much of the tutorials and conference talks involved explicit coding examples and github repositories which were unsuited for written report. Therefore, the contents within this report will solely touch upon theories, concepts and general overviews. Topics are as following:

* [Bayesian Statistics and Python Programming (pg5)](#bayes1)
* [Bayesian Statistics Modelling with PyMC3 (pg14)](#bayes2)
* [NLP and Field of Study Today (pg19)](#nlp1)
* [Transfer Learning in NLP (pg 22)](#nlp2)
* [Hands-on Satellite Imagery Analysis (pg 25)](#satellite)

For a lack of better representation, other summaries and demonstrations will be shared among team members.

**Conference Topics Summary**

**Bayesian Statistics and Python Programming**

* + **Conjoint Probability**

Writing means, the probability of occurrences A and B being both true.

In case of two independent occurrences, such as tossing a coin, the probability of both 2 occurrences having heads is:

when two occurances are independent

However, if two occurrences are dependent, such as A being raining today and B being raining tomorrow, it is intuitive to infer that .

Thus, the probability of a conjunction is:

* + **Bayes Theorem**

Bayes theorem can be derived by picking up from above concept.

so, rewriting this is:

which leads to Bayes Theorem:

A simple example which utilizes this equation is, the cookie problem:

Suppose there are two bowls of cookies. Bowl 1 contains 30 vanilla cookies and 10 chocolate cookies. Bowl 2 contains 20 of each.

Now suppose you choose one of the bowls at random and, without looking, select a cookie at random. The cookie is vanilla. What is the probability that it came from Bowl 1?

Denoting *B1* for the hypothesis that the cookie came from Bowl 1 and V for the vanilla cookie, we could write the problem as solving following equation:

Then assuming that selecting either Bowl 1 or 2 is random, .

Selecting vanilla cookie from Bowl 1 is and selecting vanilla cookie out of total cookies, .

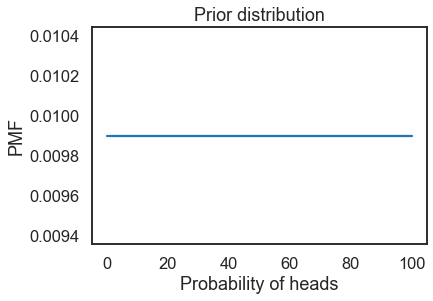
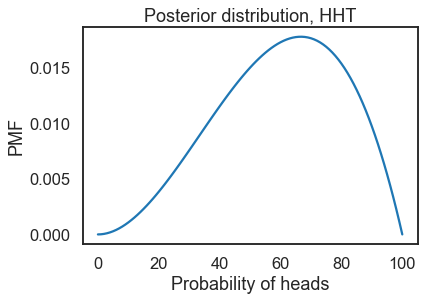
Thus,

* + **Bayesian Theorem with Diachronic Interpretation**

A more general way of interpreting Bayesian Theorem is to utilize it for updating the probability of a hypothesis, *H*, in light of some body of data, *D*.

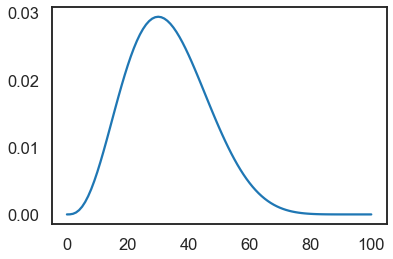
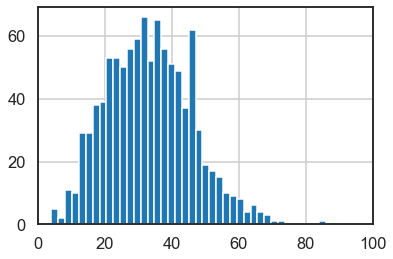
This way of thinking about Bayes’s theorem is called the diachronic interpretation. “Diachronic” means that something is happening over time; in this case the probability of the hypotheses changes, over time, as we see new data.

In this interpretation, each term has a name:

1. is the probability of the hypothesis before we see the data, called the prior probability, or just **prior**.
2. is what we want to compute, the probability of the hypothesis after we see the data, called the **posterior**.
3. is the probability of the data under the hypothesis, called the **likelihood**.
4. is the probability of the data under any hypothesis, called the **normalizing constant**
   * Examples of Bayesian Statistics using Python
5. Euro Problem  
     
     
   This problem requires proceeding in two steps. The first is to estimate the probability that the coin lands face up. The second is to evaluate whether the data support the hypothesis that the coin is biased.  
     
   Any given coin has some probability, *x*, of landing heads up when spun on edge. If a coin is perfectly balanced, we expect x to be close to 50%, but for a lopsided coin, x might be substantially different.  
     
   The key point in using Bayesian Statistics for solving this problem is not about the search for the exact value of *x*, per se, however the distribution of *x*.  
     
   Thus, *x* should be on a range between 0 to 100, where is the hypothesis that probability of heads is . Within python setup, one could starting the prior with a uniform distribution where the probability of is the same for all *x*, as such:  
     
      
     
   Then with incoming data, new information can be updated into the likelihood probability of coin toss resulting in heads with a function like:  
   If is true, the probability of heads is and the probability of tails is .  
     
    **def** Likelihood(self, data, hypo):  
    x = hypo  
    **if** data == ‘H’**:**  
    return x / 100.0  
    **else:**  
    return 1 – x / 100.0  
     
   If so, having updated with 3 sequential occurrences data, ie. **Head –** **Head -** **Tail**, the updated posterior distribution would look like:  
     
      
     
   In such matter, one could argue with 3 input data that, the maximum probability for having a head with a coin toss is 67% with a general average probability of 60% likelihood.  
     
   Again, the key take-away from this exercise is that, while one wishes to find the probability for the result of a coin toss, the output result from a Bayesian statistical analysis is not a single value but rather a probability distribution which holds the information for the general likelihood.

"When spun on edge 250 times, a Belgian one-euro coin came up heads 140 times and tails 110. 'It looks very suspicious to me,' said Barry Blight, a statistics lecturer at the London School of Economics. 'If the coin were unbiased, the chance of getting a result as extreme as that would be less than 7%.' "

*From “The Guardian” quoted by MacKay, Information Theory, Inference, and Learning Algorithms.*

1. Random Selection with a Custom Probability Distribution  
   Numpy library in python, gives some powerful tools for generating random selection data.  
     
   Let’s say we would like to select a number between 0~100, under a custom probability distribution, such that the distribution could look like below:  
     
      
     
   Then, numpy has following function to make a random selection within above distribution:  
     
     
   Using this, one can create a 1000 sample of selections as following:  
     
      
   As can be observed, the sample roughly follows the distribution as designed to be.

np.random.choice(a, size=None, replace=True, p=None)

key-Parameters

-----------

a : 1-D array-like or int

If an ndarray, a random sample is generated from its elements.

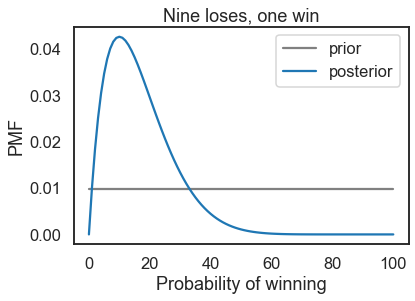
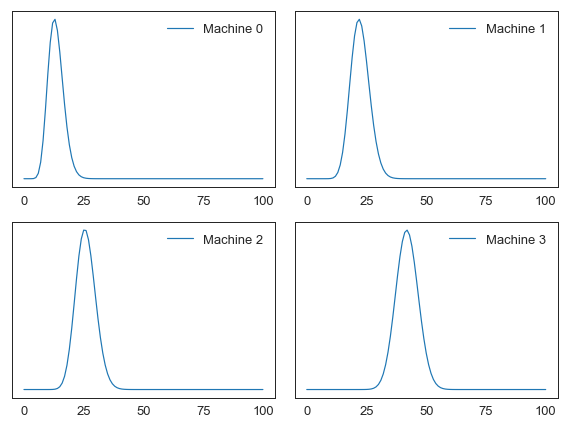
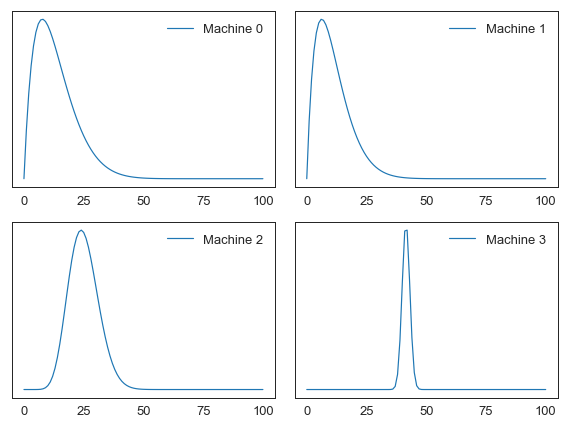
If an int, the random sample is generated as if a were np.arange(a)

p : 1-D array-like, optional

The probabilities associated with each entry in a.

If not given the sample assumes a uniform distribution over all

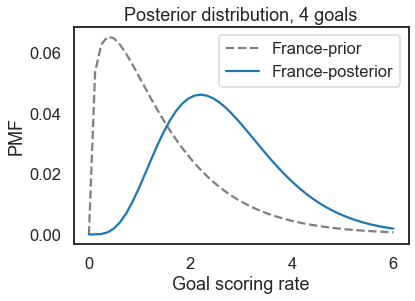
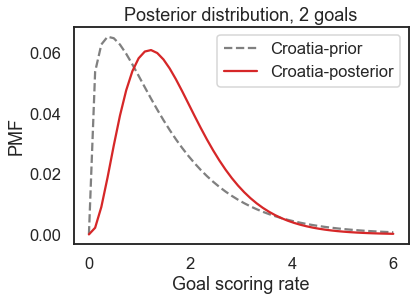
entries in a.

1. Bandit Problem  
     
     
   Considering that one could argue that an investment strategy is analogical to creating a pick-and-choose slot-machine strategy (ie. choosing the right investment strategy), this example is crucial for applying Bayesian Statistics for investment strategy.  
     
   All being equal in process with that of the “euro problem,” we’ll use the same selecting a number between 0~100 as our prior for eventually creating a success probability distribution for each slot machine, and the same likelihood update function. Therefore, if one has a slot-machine which ran a sequence of 1 win and 9 loses, hereafter denoted as ‘WLLLLLLLLL’, the posterior distribution would look like:  
     
      
   \*where grey line is the initial prior distribution and blue is the updated posterior after updating the occurrence data.  
     
   A distribution as such, gives meaningful information such as that the overall weighted-average probability of winning is 16.7% whereas the occurrence with the highest percentage for wining is 10%.  
     
   Now, say we have 4 slot machines with 10, 20, 30, 40% of winning probabilities and create random sequences of win or lose occurrence for each slot machines. For 100 sequence of occurrences for each slot machine, the winning distributions for each slot-machine would look like this:  
     
      
   Resulting average winning probabilities are, 13.39%, 22.32%, 25.89%, 41.96%.  
     
   Since, these winning probability distributions are updated after every round of plays by 4 slot-machine, one can use this data to choose which slot-machine to bet on with every round of play with the updated distribution.  
     
   By setting up the right kind of generating functions, following 1000 iterations of slot-machine plays can set-off following result:  
     
   num\_plays = 1000  
   count\_win = 0  
   for \_ in range(num\_plays):  
    count\_win = choose\_play\_update(  
    beliefs, record=True, count=count\_win)  
     
      
     
   After the run, when counting how many times each slot-machine was selected, the result is:  
     
   Slot-Machine 0 : 13 times  
   Slot-Machine 1 : 16 times  
   Slot-Machine 2 : 46 times  
   Slot-Machine 3 : 925 times  
     
   And the total number of wins was 397 times which is approximately 40% win out of 1000 runs.  
     
   However, on the other hand, if one does a random pick and choose slot-machine strategy, the result would have come up as:  
     
   Slot-Machine 0 : 252 times  
   Slot-Machine 1 : 248 times  
   Slot-Machine 2 : 257 times  
   Slot-Machine 3 : 243 times  
     
   And the total number of wins are 235 times which is approximately 23.5% win out of 1000 runs.  
     
   Thus, updating the model and choosing slot-machine through Bayesian approach not only **improved the outcome** but also **successfully allocated one’s bet to the most winning-probable slot-machine** through empirical trials.

Suppose you have several "one-armed bandit" slot machines, and reason to think that they have different probabilities of paying off.

Each time you play a machine, you either win or lose, and **you can use the outcome to update your belief about the probability of winning.**

Then, to decide which machine to play next, you can use the "Bayesian bandit" strategy.

1. Synthetic Outcome Generation with Bayesian Statistics  
   When predicting complex probability, Bayesian statistics can become handy for generating likely occurrences and deriving the outcome probability.  
     
   Also, it is important to know the different probability distributions available and choose the appropriate distribution to approximate the phenomena.  
     
   One type of problem might be solving the World Cup problem:  
     
     
   In order to setup a probability distribution for the number of goals a team is likely to score, one could use a **Poisson distribution** which is non-negative and discrete (ideal for this case of soccer goals).  
     
   First one could set a prior distribution which would set the max-probable average number of goals for any team to be 1.4 goals (based on data from previous World Cups).  
   Then after, one could update the prior with 4 goals for France and 2 goals for Croatia, which would have an updated PMF as below:  
     
       
     
   Now that distributions for each team has been updated, this can become the basis for creating random game scores and predict the likelihood for the match-up outcome.

In the 2018 FIFA World Cup final, France defeated Croatia 4 goals to 2. Based on this outcome, we can answer the following questions:

1. How confident should we be that France is the better team?
2. If the same teams played again, what is the chance Croatia would win?

**Bayesian Statistics Modelling with PyMC3**

PyMC3 library is a powerful tool for testing Bayesian statistical hypothesis in a concise manner whilst harnessing multi-threading and GPU computation. In real-world problems, there are problems where more-than one parameter are unknown and these parameters can also be hypothesized. The beauty in PyMC3 is the ability to add more than one priors for problems with multiple parameters, and yet compute the outcome probability with added computational improvement through library utilities.

The basics of Bayesian model building follows the steps of:

1. To completely specify the model in terms of probability distributions. This includes specifying
   * what the form of the sampling distribution of the data is and
   * what form describes our uncertainty in the unknown parameters (This formulation is adapted from [Fonnesbeck's workshop](https://github.com/fonnesbeck/intro_stat_modeling_2017/blob/master/notebooks/2.%20Basic%20Bayesian%20Inference.ipynb" \t "_blank) as Chris said it so well there).
2. Calculate the posterior distribution.

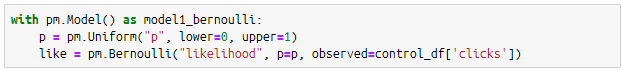
Under the hood, PyMC3 will compute the posterior using a sampling based approach called Markov Chain Monte Carlo (MCMC) or Variational Inference.

1. **Example 1 : Finding Before/After Statistical Change**  
   To verify from Bayesian statistics whether higher visual contrast button in a website will induce more clicks from users.

Say that we don't think enough people are clicking a button on my website, but we hypothesize that it's because the button is a similar color to the background of the page, meaning they're a difficult time finding the button to click.

In a **‘control’** group we show the original web-page, and count the number of clicks by user. In the **‘test’** group we will show the web-page with a higher contrast button and count the clicks by user.

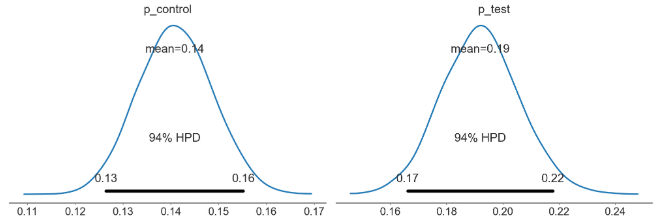
PyMC3 allows creating a Bayesian model wrapped in a python function as following:

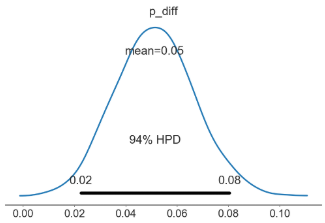


In above, we’ve set up a uniform distribution for the probability to click (0~100%) as prior and used a likelihood update function using the Bernoulli distribution.

By setting up two of same models, one for the control group and one for the test group, we will have two posterior distributions which we can compare against each other to verify the hypothesis.

The outcome is as following:





This shows that there is 5% difference in click rate when using higher contrast button.

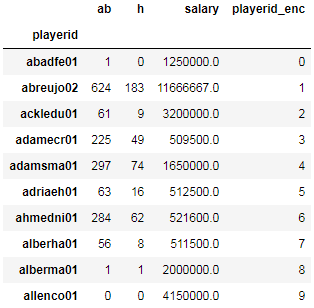
1. **Example 2 : Baseball Players**

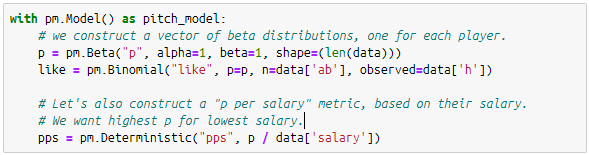
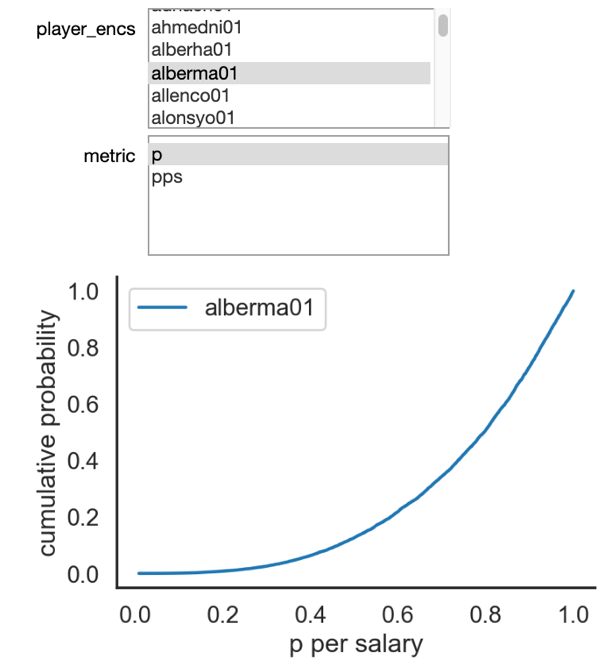
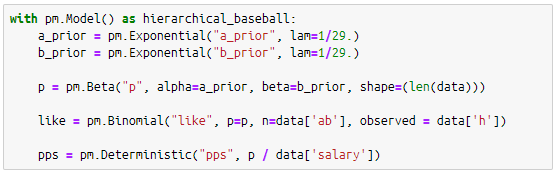
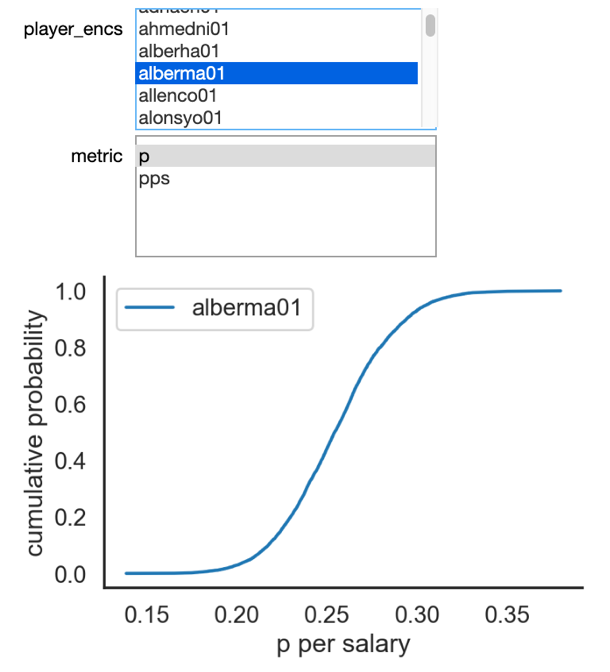
Further usage case for Bayesian statistics can be tested in PyMC3 with following example:

Goal of this analysis is to identify which player we want to make an offer to join our baseball team based on the batting stats.

We are going to make a decision on the basis of just batting average and their salaries.

‘ab’ column stands for at-bat, and ‘h’ stands for hit counts. One would assume that batting average has a direct impact on salary.

… (to 805 players)

* + - * ***Defining Model:***Since we have a number of trials (ab), with a number of successes (h), from which we want to estimate a player-specific property: p (probability of hitting a pitch), we can use Bernoulli distribution as our prior.  
          
        Code can be written as following:  
          
        The core strength behind this modeling in PyMC is that, by assigning a ‘shape’ to the function, we are setting up 805 distributions for each player, instead of just one for the whole.  
        The data richness that comes out of this is substantial compared with a simple 1-dimensional average hit per player.
      * ***Enhancing a Model by Defining Prior Parameters***Now, when a model is not well-rounded enough to take consideration for players with little to no data points, we might get hitting percentile outcome as following.  
        For instance, from our model we’ll analyze the ECDF (empirical cumulative distribution function) of the hitting probability relative to salary. For a player like ‘alberma01’ who has only had one hit chance and one hit (equivalent to 100% hit-ratio) the ECDF looks like:  
          
          
        This is however due to the fact that the model does not have a default distribution for players without enough datasets (meaning, we should assume his hit-ratio is not 100%). We could solve this through defining the parameters for the prior to adjust for these cases. This type model is called hierarchical model.  
          
        An example for setting up this hierarchical model is as following:  
          
          
        The change from this can be seen below with player, ‘alberma01’:  
          
        Meaning, with a prior that is also assumed with probable parameters that follow general player hit-ratio assumptions, even players with small data-points can be supplemented by a prior to create a reasonable distribution.
      * ***Analysis Conclusion:*** By finding the highest mean hit-probability relative to salary, this should become the most undervalued (attractive) player.

**NLP and Fields of Study Today**

There has been so much growth in the field of NLP (Natural Language Processing), to the point that computers can now finish a sentence written by humans. While powerful and accurate at the same time, it is becoming more and more important to understand NLP technology in order to fully harness the power and utilize its strength in assisting investment decisions.

In a general list, following sums up the different studies in NLP which which have made progress:

* Automatic speech recognition
* CCG
* Common sense
* Constituency parsing
* Coreference resolution
* Dependency parsing
* Dialogue
* Domain adaptation
* Entity linking
* Grammatical error correction
* Information extraction
* Language modeling
* Lexical normalization
* Machine translation
* Missing elements
* Multi-task learning
* Multi-modal
* Named entity recognition
* Natural language inference
* Part-of-speech tagging
* Question answering
* Relation prediction
* Relationship extraction
* Semantic textual similarity
* Semantic parsing
* Semantic role labeling
* Sentiment analysis
* Shallow syntax
* Simplification
* Stance detection
* Summarization
* Taxonomy learning
* Temporal processing
* Text classification
* Word sense disambiguation

Although all studies are important in their own respect, these are some details for topics which are likely to be more important in their use-cases for NLP in finance:

1. **Information Extraction**Open Information Extraction approaches leads to creation of large Knowledge bases (KB) from the web. The problem with such methods is that their entities and relations are not canonicalized, which leads to storage of redundant and ambiguous facts. For example, an Open KB storing <Barack Obama, was born in, Honolulu> and <Obama, took birth in, Honolulu> doesn't know that Barack Obama and Obama mean the same entity. Similarly, took birth in and was born in also refer to the same relation. Problem of Open KB canonicalization involves identifying groups of equivalent entities and relations in the KB.  
     
   There are two apparent models for this: CESI (Vashishth et al., 2018) and Galarraga et al., 2014 (IDF).
2. **Lexical Normalization**  
   Lexical normalization is the task of translating/transforming a non-standard text to a standard register.  
   (example: new pix coming in tomoroe / new pictures coming tomorrow)  
     
   Datasets usually consists of tweets, since these naturally contain a fair amount of these phenomena. For lexical normalization, only replacements on the word-level are annotated. Some corpora include annotation for 1-N and N-1 replacements. However, word insertion/deletion and reordering is not part of the task.  
     
   Known models are: MoNoise (van der Goot & van Noord, 2017), Joint POS + Norm in a Viterbi decoding (Li & Liu, 2015), Syllable based (Xu et al., 2015), unLOL (Yang & Eisenstein, 2013)
3. **Named Entity Recognition**Named entity recognition (NER) is the task of tagging entities in text with their corresponding type. Approaches typically use BIO notation, which differentiates the beginning (B) and the inside (I) of entities. O is used for non-entity tokens.
4. **Part-of-Speech Tagging**Part-of-speech tagging (POS tagging) is the task of tagging a word in a text with its part of speech. A part of speech is a category of words with similar grammatical properties. Common English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, etc.
5. **Relationship Extraction**Relationship extraction is the task of extracting semantic relationships from a text. Extracted relationships usually occur between two or more entities of a certain type (e.g. Person, Organisation, Location) and fall into a number of semantic categories (e.g. married to, employed by, lives in).  
     
   Example: Elevation Partners, the $1.9 billion private equity group that was founded by Roger McNamee 🡪 (founded\_by, Elevation Partners, Roger\_McNamee).
6. **Sentiment Analysis**Sentiment analysis is the task of classifying the polarity of a given text.  
   One of corpuses which are used for testing this is IMBd dataset which is a binary sentiment analysis dataset consisting of 50,000 reviews from the Internet Movie Database (IMDb) labeled as positive or negative.  
     
   Some of the best models utilize pre-training models, which is also a crucial concept for building highly advanced NLP algorithms.
7. **Text Classification**Text classification is the task of assigning a sentence or document an appropriate category. The categories depend on the chosen dataset and can range from topics.  
     
   It could be said that sentiment analysis is a part of this text classification study.
8. **Word Sense Disambiguation**The task of Word Sense Disambiguation (WSD) consists of associating words in context with their most suitable entry in a pre-defined sense inventory. The de-facto sense inventory for English in WSD is WordNet. For example, given the word “mouse” and the following sentence:  
     
   “A mouse consists of an object held in one's hand, with one or more buttons.”  
   We would assign “mouse” with its electronic device sense (the 4th sense in the WordNet sense inventory).

As shown above, the field of NLP is not a one-size fits all blackbox artificial intelligence, but rather a group of studies which encompasses the various tasks of language processing. It is usually a group of these tasks which are combined to create a model which would interpret text-documents that relate to investment textual-data, and filter them into a type of indicator or sentiment that could assist investment decision making.  
  
For instance, “Lexical Normalization”, “Part-of-Speech Tagging” are studies which are used to refine the text-information to enhance further meaning extraction. “Information Extraction”, “Named Entity Recognition”, “Relationship Extraction”, and “Word Sense Disambiguation” are fields that can improve accuracy to find the correct subject in relation within the text and the proper meanings behind expressions used. Lastly, “Text Classification” and “Sentiment Analysis” tasks will translate text into targeted information that hold informational value regarding the subject written within the context. This informational value may hold crucial ideas about the future investment return.

**Transfer Learning in NLP**

1. **Background**

In the last couple of years we’ve started to see deep learning making significant inroads into areas where computers have previously seen limited success. Rather than requiring a set of fixed rules that are defined by the programmer, deep learning uses neural networks that learn rich non-linear relationships directly from data. Most notable is the success of deep learning in computer vision, as seen for example in the rapid progress in image classification in the Imagenet competition.



Deep learning has also seen some success in NLP, for example in automatic translation. A common feature of successful NLP tasks is that large amounts of labeled data are available for training a model. However, until now such applications were limited to those institutions that were able to collect and label huge datasets and had the computational resources to process them on a cluster of computers for a long time.

One particulaar area that is still challenging with deep learning for NLP, curiously enough, is the exact area where it’s been most successful in computer vision: classification. This refers to any problem where your goal is to categorize things (such as images, or documents) into groups (such as images of cats vs dogs, or reviews that are positive vs negative, and so forth). In NLP, current approaches are good at identifying, for instance, when a movie review is positive or negative, a problem known as sentiment analysis. Models struggle, however, as soon as things get more ambiguous, as often there is not enough labeled data to learn from.

1. **How does Transfer Learning Improve NLP**

Transfer learning facilitates those that don’t have masses of data and computational resources to deal with some specific NLP problems.

**Transfer learning** refers to the use of a model that has been trained to solve one problem (such as classifying images from Imagenet) as the basis to solve some other somewhat similar problem.

Because the fine-tuned model doesn’t have to learn from scratch, it can generally reach higher accuracy with much less data and computation time than models that don’t use transfer learning.

Simple use-case of transfer learning:

* + - * Using a single layer of weights, in order words, word2vec embeddings from Google to conduct an NLP study.

However, full neural networks in practice contain many layers, so only using transfer learning for a single layer was clearly just scratching the surface of what’s possible.

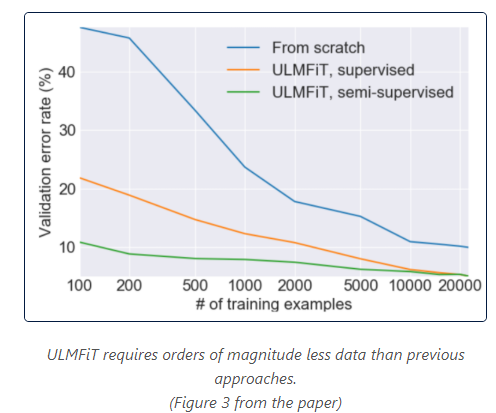
1. **Transfer Learning Application by Fast.ai**

A project from fast.ai researchers has released a paper on Universal Language Model Fine-tuning for Text Classification (ULMFiT), pre-trained models, and full source code in the Python programming language. It’s paper has been peer-reviewed and accepted for presentation at the Annual Meeting of the Association for Computational Linguistics (ACL 2018).

It is a transfer learning method that dramatically improves over previous approaches to text classification, and the code and pre-trained models allow anyone to leverage this new approach to better solve problems such as:

* + - * Finding documents relevant to a legal case;
      * Identifying spam, bots, and offensive comments;
      * Classifying positive and negative reviews of a product;
      * Grouping articles by political orientation;

The result has shown remarkable improvements without the masses of data and computational resources that might be needed otherwise.



Paper is found in <https://arxiv.org/abs/1801.06146>

As can be learned, within an organization like KIC, conducting NLP strategies can be a challenging task when considering size and limited data resources. Without massive datasets, KIC is unlikely to train an NLP model from scratch by lack of testing data and access limitation to cloud computation.

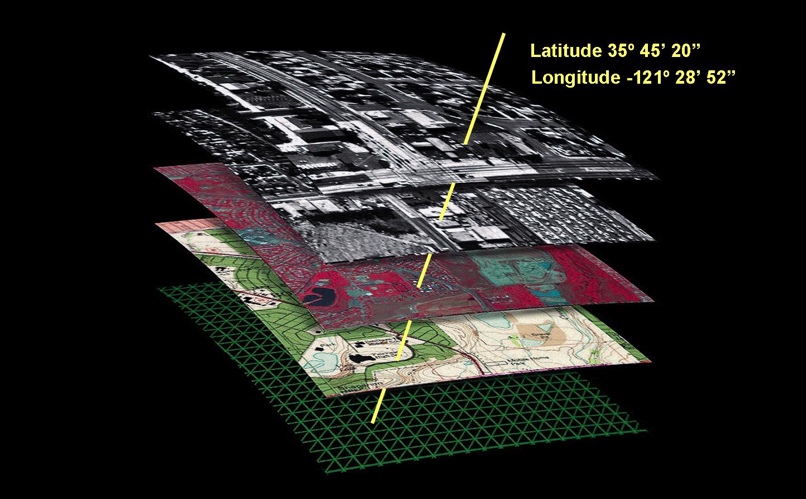
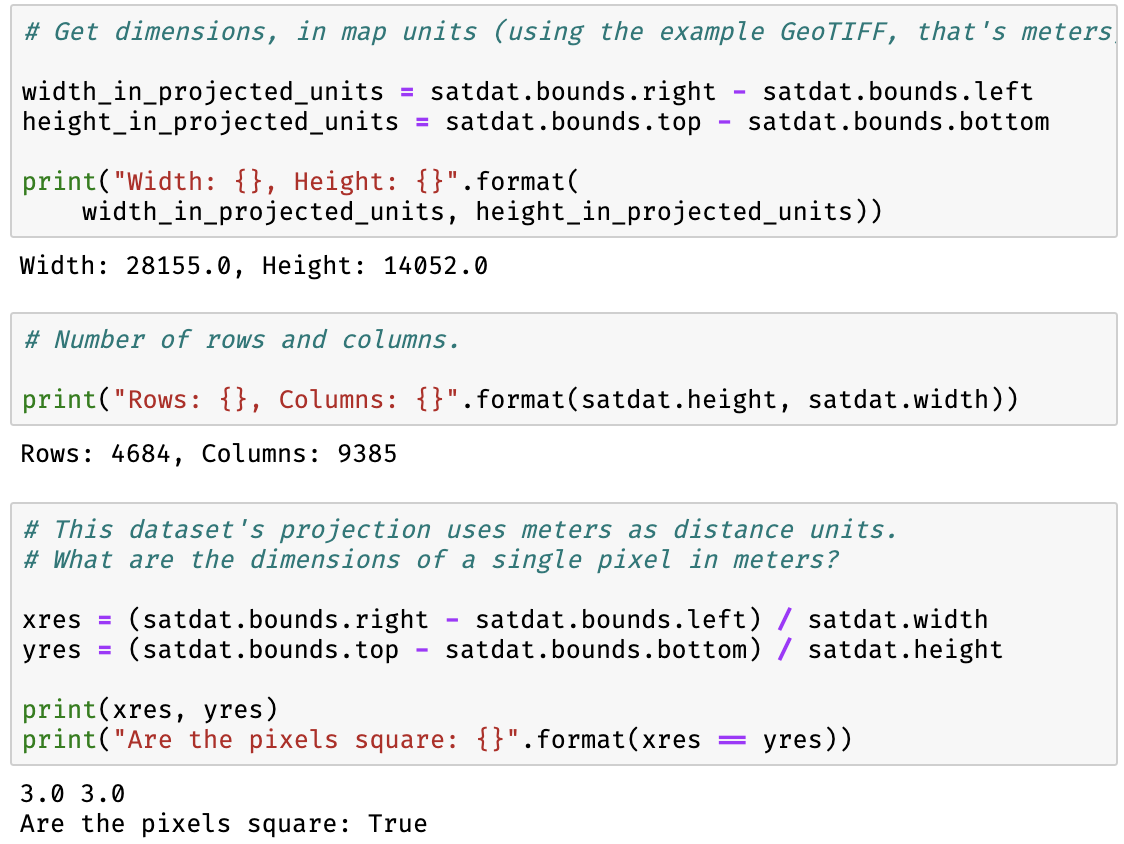
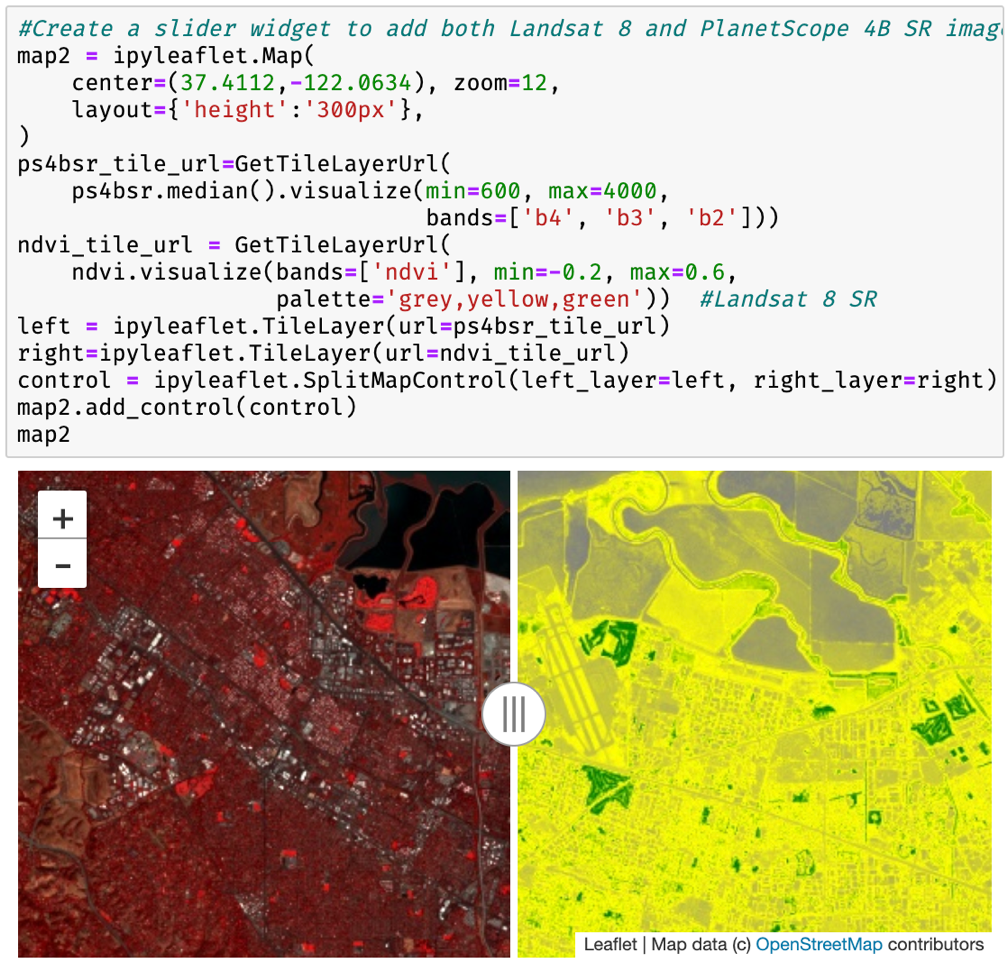
For this reason, although applying deep learning for investment strategy purposes might still be too advanced at current stage (for creating a non-biased, non-overfit model), when we actually plan to do so, transfer learning seems to be a good starting point.

**Hands-on Satellite Imagery Analysis**

Most of the materials dealt during this subject were not ideal for demonstration in words. Summary written here will be minimally detailed.

Satellite imageries require terra-bytes of data processing which can be mind-boggling for researchers to utilize. Putting aside the depth and scale of the data, some key-traits of satellite imagery are:

* + - * Satellite imageries are a collection of metadata
      * Typically used measuring metric is distance per pixel
      * Zooming images use techniques of expanding collage of image-patches which are in more detail
      * Satellite imagery data-vendors provide a web-API platforms which allow easy access through various languages such as python (ex. Google Earth)

1. **What is meta-data?**Meta-data contains geographically assigned data-points which also hold information other than imagery data (ie. RGB).  
      
     
   An example is an imagery with 4 bands, that is, layers that depict blue, green, red, and NIR. NIR is infrared data which can be used to detect vegetation, thus monitor the changes in forestation.  
     
   These metadata are collected by means of advanced sensor on the satellite and also data-processing by vendors. Metadata allows users to get hold of richer datasets that is also easier to analyze for the given purpose.
2. **An illustration of measuring the distance per pixel by python:**  
   While the rows of the matrix data is 4684, the height of the satellite image height has 14052 units. We can say that the image pixel is equivalent to 3 distance units in real life.
3. **API and Programming for Analysis**While many proprietary vendors exist for providing satellite imageries, the most well-known and open API is Google Earth Engine (GEE).  
   One can use date ranges and metadata to filter images and add image tiles to render from GEE. In GEE environment image collections have their own characteristic setup and are often composed of multiple single images. They often have the similar band structure and generally share a similar metadata structure for filtering and querying.  
     
     
   Above right side shows the ‘nir (infrared)’ band of the image which captures the vegetation area with a distinct green color.

Although satellite imagery is hard to access and the techniques for analyzing requires a certain amount of specialty, the concepts behind are pretty straight forward and similar to any other data analytics task.

Recent developments in machine learning capabilities are gaining traction within the landscape of finance because satellite imagery is essentially the next frontier for monitoring global activities in real-time. Below is an example of counting dump trucks in a mining field, measuring the activities within the area:



As such, ideas for quantifying non-structured data are countless.

The takeaway for this tutorial has been to understand the key concepts and tools to analyze such imagery data.

This tutorial has been a beginner’s course which is nowhere close to analyzing real-live satellite imageries for investment purposes; however, it has offered an idea for what skillsets are needed and what level of data infrastructure is required to build the capability for extracting information and adding value to our system. At current stage, the possibility of acquiring satellite imagery for investing seems highly unlikely in the near future due to the level of complexity for data processing and the insufficiency of metadata for relevant information.