# Capstone Project

Machine Learning Engineer Nanodegree

Yung-Chun, Lu
Aug 4th, 2016

## Definition

### Project Overview

- Project Origin: Kaggle - Facebook V: Predicting Check Ins
- Description: Facebook and Kaggle launched a machine learning engineering competition of identifying the correct place for check ins. For giving a flavor of what it takes to work with real data, Facebook created an artificial world consisting of more than 100,000 places located in a 10 km by 10 km square and data was fabricated to resemble location signals coming from mobile devices.
- Data Sets:
  - Train.csv, test.csv
    - row_id: id of the check-in event
    - x, y: coordinates
    - accuracy: location accuracy
    - time: timestamp
    - place_id: id of the business (this is the target I'm predicting)

### Problem Statement

- Problem Definition:
  - Predicting the top 3 places a person would most likely to check in to.
  - Total number of places on the map is over 100000.
  - The probability of place could be modeled as

$$Pr(place\ id \mid x,\ y,\ time,\ accuracy)$$

  - This problem is categorized as supervised classification.
- Proposed Solution:
  - K-Nearest Neighbors
    - Since I do not have any assumptions about the data distribution, KNN could be the good start point.
    - The weight of coordinates should be much higher than others.
  - Gaussian Naive Bayes
    - When looking at histograms of features, we could see that all the shape look like a bell. So I think it's worth trying to know how well this method could achieve.
  - Kernel Density Estimation
    - Based on the result of GNB, the assumption that all conditional probabilities of features were independent may hold on this problem
    - KDE could help finding more accurate estimation of probabilities of features
  - Ensemble
    - At final step, I try to ensemble above three models with customized method.
    - This method is inspired by Larry Freeman

## Metrics

- Prediction is evaluated according to the Mean Average Precision @3

$$MAP@3 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(3,n)} P(k)$$

  - |U| is the number of check in events
  - P(k) is the precision at cutoff k
  - n is the number of predicted businesses.

# Analysis

## Data Exploration

In this section, I try to explore values distribution of following features:

- Basic Statistics
- Coordinates
- Accuracy
- Place_ids

### Basic Statistics

- From the tables, we can see that training and testing data are splitted by time.
- Training Data:
  - Size of  training data: (29118021, 6)

**Table.1** Description of training data

| | row_id | x | y | accuracy | time | place_id |
|---|---|---|---|---|---|---|
| **count** | 2.911802e+07 | 2.911802e+07 | 2.911802e+07 | 2.911802e+07 | 2.911802e+07 | 2.911802e+07 |
| **mean** | 1.455901e+07 | 4.999770e+00 | 5.001814e+00 | 8.284912e+01 | 4.170104e+05 | 5.493787e+09 |
| **std** | 8.405649e+07 | 2.857601e+00 | 2.887505e+00 | 1.147518e+02 | 2.311761e+05 | 2.611088e+09 |
| **min** | 0.000000e+0- | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000016e+09 |
| **25%** | 7.279505e+07 | 2.534700e+00 | 2.496700e+00 | 2.700000e+01 | 2.030570e+05 | 3.222911e+09 |
| **50%** | 1.455901e+07 | 5.009100e+00 | 4.988300e+00 | 6.200000e+01 | 4.339220e+05 | 5.518573e+09 |
| **75%** | 2.183852e+07 | 7.461400e+00 | 7.510300e+00 | 7.500000e+01 | 6.204910e+05 | 7.764307e+09 |
| **max** | 2.911802e+07 | 1.000000e+01 | 1.000000e+01 | 1.033000e+03 | 7.862390e+05 | 9.999932e+09 |

- Testing Data:
  - Size of  testing data: (8607230, 5)

**Table.2** Description of testing data

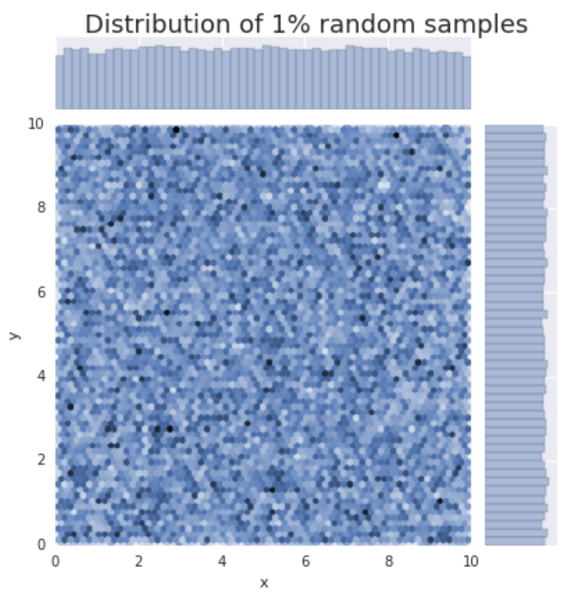
| | row_id | x | y | accuracy | time |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| **count** | 8.607230e+06 | 8.607230e+06 | 8.607230e+06 | 8.607230e+06 | 8.607230e+06 |
| **mean** | 4.303614e+06 | 4.991417e+00 | 5.006705e+00 | 9.265208e+01 | 8.904637e+05 |
| **std** | 2.484693e+06 | 2.866409e+00 | 2.886888e+00 | 1.242906e+02 | 6.446783e+04 |
| **min** | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 | 7.862420e+05 |
| **25%** | 2.151807e+06 | 2.517000e+00 | 2.502400e+00 | 4.200000e+01 | 8.332200e+05 |
| **50%** | 4.303614e+06 | 4.988000e+00 | 5.000900e+00 | 6.400000e+01 | 8.874620e+05 |
| **75%** | 6.455422e+06 | 7.463600e+00 | 7.505300e+00 | 7.900000e+01 | 9.454910e+05 |
| **max** | 8.607229e+06 | 1.000000e+01 | 1.000000e+01 | 1.033000e+03 | 1.006589e+06 |

## Coordinate

- From Fig. 1, we can see that the check-ins are roughly uniformly distributed on the map.

**Fig.1** Distribution of random samples



- Using distribution of top 3 places to show that **standard deviation of x is much more than y.**

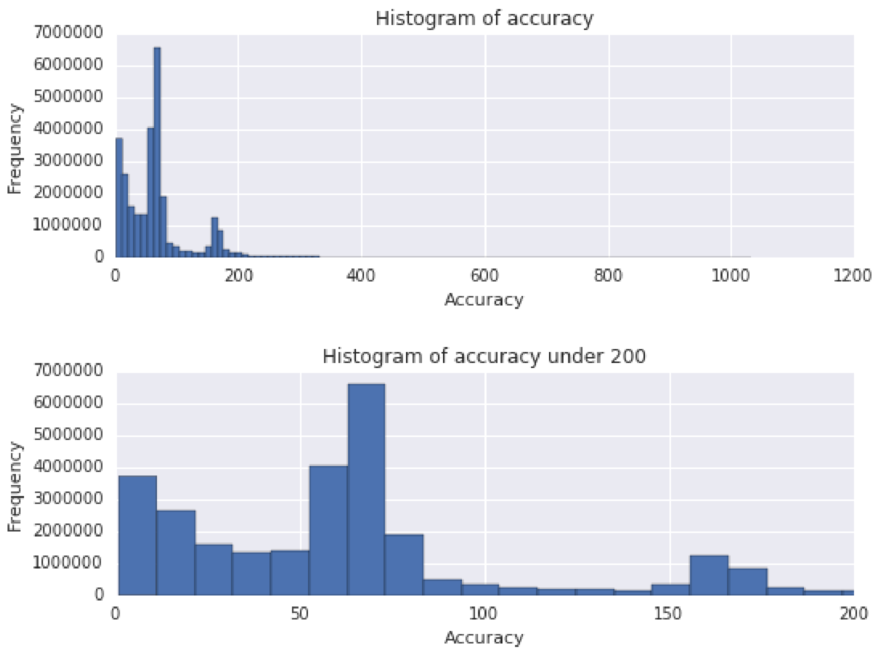
**Table.3** Standard Deviation of x and y on top 3 places

| | **1st place** | **2nd place** | **3rd place** |
|---|---|---|---|
| **Std of x** | 0.3561 | 0.2406 | 0.3401 |
| **Std of y** | 0.0115 | 0.0157 | 0.0177 |

## Accuracy

- From Fig. 2, we could know that majority of value is under 200.

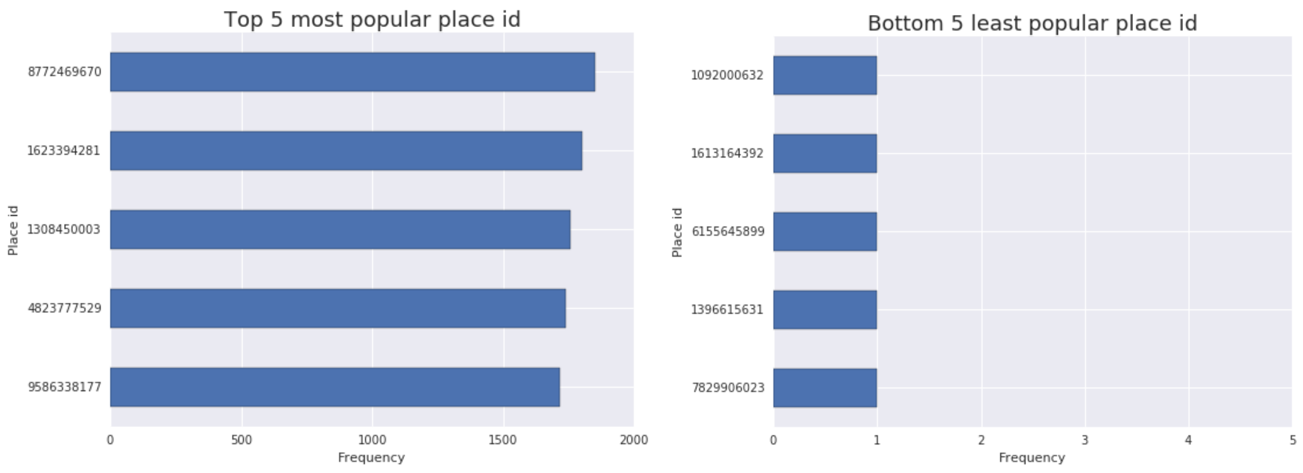
**Fig. 2** Histogram of accuracy



### Place ids

- Number of unique place is 108390, which is huge number for a classification problem
- From Fig. 3, we could know that number of place check-ins are not uniformly distributed.

**Fig. 3** Histogram of top 5 most and 5 least popular places



## Exploratory Visualization

In this section, I try to focus representing following things:

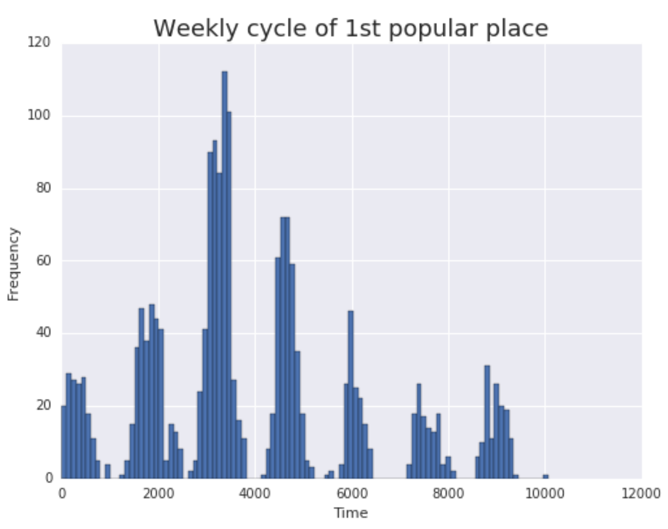
- Explore Time: Explain why the unit of time would be minute.
- Explore Accuracy: Explain why I use log function to convert accuracy.

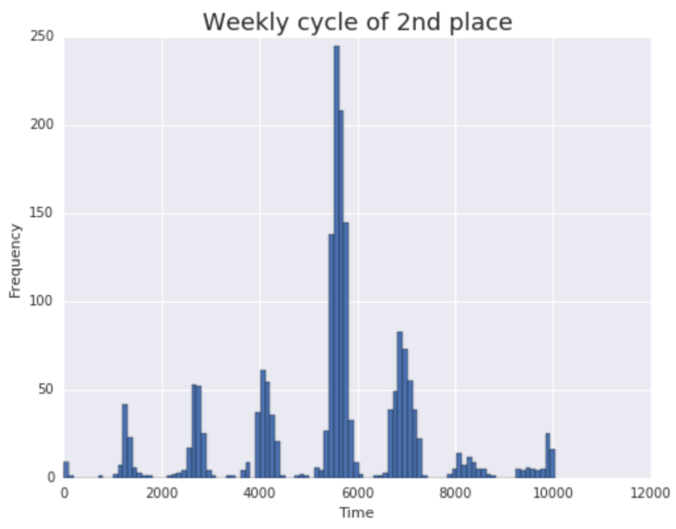
### Explore Time

Because the column of time is intentionally left vague without defining the unit of time, this section provides two methods to conclude the unit of time would be minute.

- **Method 1: Converted the time to weekdays in order to visualize the weekly cycle.** From Fig.4, we could see that there are 7 different sections representing 7 days a week.

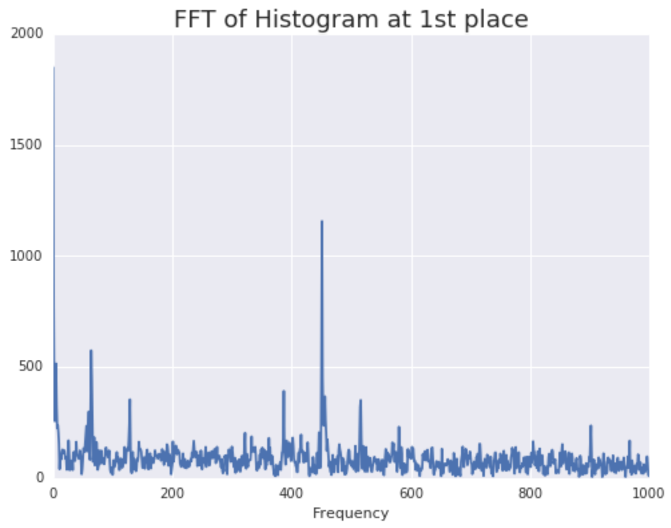
**Fig. 4** Histogram of 1st and 2nd popular places.





- **Method 2: Used Fourier transform to extract the dominant frequency in order to find the period.** From Fig.5, the 1st period is 10151.96874 closing to 10080 minutes a week. The 2nd period is 1440.634 closing to 1440 minutes a day.

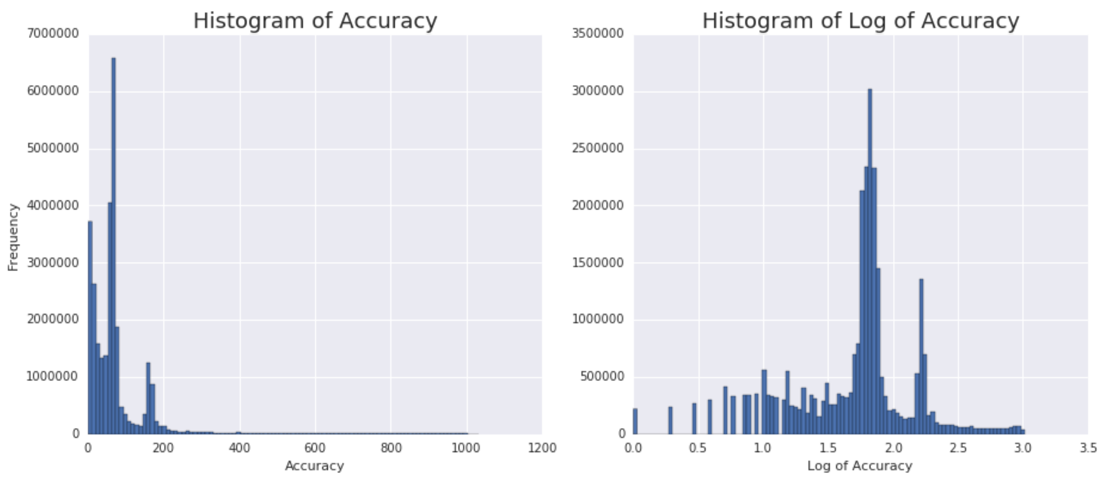
**Fig. 5** Fourier transform of histogram.



## Explore accuracy

I use coefficient of variation to measure the level of dispersion around mean. Before applying log function, the coefficient of variation of  accuracy is 1.385, after applying, the score drops to 0.292.

**Fig. 6** Histogram of accuracy and log of accuracy



## Algorithms and Techniques

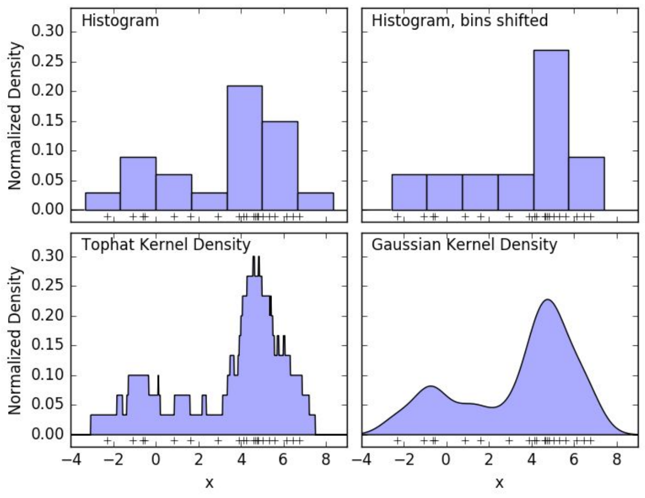
For this supervised classification problem, following are those algorithms I'm trying to use

- KNN(K-Nearest Neighbor)
  - Description:
    - When a prediction is required for a unseen data instance, the KNN algorithm will search through the training dataset for the k-most similar instances.
    - The similarity measure is dependent on the type of data. For real-valued data, the Euclidean distance can be used. Other types of data such as categorical or binary data, Hamming distance can be used.
    - It's a non parametric learning algorithm, so it means that KNN does not make any assumptions on the underlying data distribution.
    - It is also a lazy algorithm. What this means is that it does not use the training data points to do any generalization.
  - Discussion:
    - Since I do not have any assumptions about the data distribution, KNN could be the good start point.
    - The weight of y should be more than x because of the deviation of x is much more than y.
    - Since I only have limited knowledge about each feature but do not know exactly how important are they, the weight will be set based on trial and error.
  - Reference:
    - Tutorial To Implement k-Nearest Neighbors in Python From Scratch
    - A Detailed Introduction to KNN
    - K-Nearest Neighbors: dangerously simple
- Gaussian Naive Bayes
  - Description:
    - It's a simple classifier based on applying Bayes Theorem with independence assumptions between features and probability models of each feature are Gaussian.
    - Equation of Bayes Theorem

$$P(y \mid X) = \frac{P(X \mid y)\, P(y)}{P(X)}$$

      - X: features, y: class
      - P(y|X): posterior probability
      - P(y): prior probability of class

        - P(X): prior probability of features
        - P(X|y): class conditional model, which is gaussian distribution
    - Discussion:
        - From the below histograms, we could see that all the shape look like a bell. So I think it's worth trying to know how well this method could achieve.
        - And I assume each feature is independent although I do not prove it yet.
    - Reference:
        - Wiki
        - Intro of GNB
        - A simple explanation of Naive Bayes Classification
- Kernel Density Estimation
    - Description:
        - Kernel density estimation is a data smoothing way to estimate the probability density function of a random variable.



        - A major problem with histogram(upper-left panel) is that the choice of binning can have a disproportionate effect on the resulting visualization(upper-right panel). We can solve this problem and recover a smoother distribution by using KDE. By using different kernels, the visualization will be different. Bottom-left panel is top-hat kernel and bottom-right panel is Gaussian kernel.
        - The bandwidth of the kernel is a free parameter which exhibits a strong influences on the resulting estimate.
    - Discussion:
        - This problem could be approximately modeled as following

$$P(place\,|\,x,\ y,\ time,\ accuracy)$$
$$\approx P(x,\ y,\ time,\ accuracy\,|\,place)\ \times P(place)$$
$$\approx P(x\,|\,place)\ \times P(y\,|\,place)\ \times P(time\,|\,place)\ \times P(accuracy\,|\,place)\ \times P(place)$$

        - Based on the result of GNB, the assumption that all conditional probabilities of features were independent could hold on this problem.
    - Reference:
        - Wiki
        - KDE on sklearn
- Ensemble
    - Discussion:
        - Based on following equation, finding the place_ids such that P(place_id) are top 3 value.

$$P(place) = \sum_{M} \sum_{i=0}^{k} \frac{W(M)}{i+1}$$

        - M: ensembled model
        - W(M): weight of model
        - i: the order of each row
        - k: the number of place_ids of each row
    - Reference:
        - Larry Freeman

## Benchmark

I use the most voted and forked methods on Kaggle forums as my benchmark.

- KNN by Sandro
    - Leader Board Score : 0.56736
    - Features this method used: [x, y, hour, weekday, day, month, year]
    - Corresponding weights: [500, 1000, 4, 3, 1./22, 2, 10]
    - This method divided the whole map into 20*40 grid cells for computational issues.
    - In each cell, it do not consider those place-ids if the number of occurrences is less than 5.

# Methodology

## Data Preprocessing

I do following three things for preprocessing data

- Since it has been found that the unit of time is minute, we could add more fields related with time, which are hour, weekday, month, year.
- Dividing the whole map into small grid cells and labeling the number on each cell.
- Converting accuracy into log scale

## Implementation

- KNN
    1. Dividing the whole map into many small cells.
    2. In each cell, splitting data by time into training and validation data sets.
    3. Training a KNN classifier in each cell without feature weighting, and calculating the accuracy.
    4. Training with feature weighting. The order of weights is y > x > year > hour > weekday > month > day.
- GNB
    1. Dividing the whole map into many small cells.
    2. In each cell, splitting data by time into training and validation data sets.
    3. Training a GNB classifier in each cell with feature selection, and calculating the accuracy.
    4. Training with feature selection. Selected features are x, y, hour, weekday, log of accuracy.
- KDE
    1. Dividing the whole map into many small cells.
    2. In each cell, splitting data by time into training and validation data sets.
    3. In each cell, using kde to approximate the conditional probabilities of features.
    4. In each cell, calculating the probability of place_id
    5. Multiplying all the probabilities, sorting in descending order and picking the top 3 place ids.
- Ensemble

1. Based on the method of Larry Freeman, ensemble the result of above models.

## Refinement

Because it will take too much time generating one submission file for validating improvement, I will instead using accuracy as metric for simplicity.

$$Accuracy \ = \sum \frac{\# \ of \ Right \ Predictions * 100}{\# \ of \ Total \ Label}$$

Training and validating data are splitted by time in order to keep similar characteristics between testing data.

- KNN:
  - Refinement #1: Selecting features and using weights. The accuracy is boosted from 4.x to 43.x
    - Features: [x, y, hour, weekday, day, accuracy, month, year]
    - Corresponding weights: [600, 1200, 10, 10, 0.01, 30, 2, 15]
  - Refinement #2: Choosing different parameters for KNN. The accuracy is boosted from 43.x to 45.x
  - Refinement #3: Transforming scales of hour, weekday, and month into sin and cos for solving problems of periods. The accuracy is boosted from 45.x to 46.x
- GNB:
  - Refinement #1: Selecting features. The accuracy is boosted from 9.x to 42.x
    - Features: [x, y, sin of hour, cos of hour, weekday, accuracy]
  - Refinement #2: Adding probability of place. The accuracy is boosted from 42.x to 43.x

$$P(place \,|\, x,\ y,\ time,\ accuracy)$$
$$\approx P(x \,|\, place)\ \times P(y \,|\, place)\ \times P(time \,|\, place)\ \times P(accuracy \,|\, place)\ \times P(place)$$

  - Refinement #3: When calculating probabilities of place, more recent check-in should give higher weight. I use exponential function to model weights. The accuracy is boosted from 43.0x to 43.3x

$$P(place)\ =\ \frac{1}{|U|}\sum_i e^{\alpha \frac{(t_i - t_{end})}{t_{end}}}$$

    - $t_i$: time of check-in given place_id
    - $t_{end}$: the last time of check-in in the cell
    - |U|: the number of total check-ins in the cell
    - $\alpha$: the constant for adjusting decay
- KDE:
  - Refinement: Based on the assumption of independent features, using KDE to get better estimation of probabilities of features. The accuracy is boosted from 43.x to 46.x
    - For setting bandwidth of kernel, I use Scott's Rule with slight modification.
    - Features: [x, y, hour, weekday, accuracy]
- Ensemble:
  - Refinement: Based on the shared method from Larry Freeman, combined this three models to get score of 47.x
    - Models: [KDE, KNN, GNB]
    - Weights: [68, 71, 19]

# Results

## Model Evaluation and Validation

In this section, I generate submission files of each model and will report the scores of each model. I run all files on Google Cloud, machine type is 4 vCPUs, 15GB.

**Table.4** Performance of Models

| | KNN | GNB | KDE | Ensemble |
|---|---|---|---|---|
| **Score on Kaggle Learderboard** | 0.57855 | 0.54893 | 0.58171 | 0.59250 |
| **Run Time** | 1 hour | 6 hours | 23 hours | 2.5 hours |

## Justification

In this section, I try to compare the score of mine on leardboard with benchmark.

**Table.5** Comparison of Performance

| Model | Score | Ranking on Kaggle Leardboard |
|---|---|---|
| **Benchmark** | 0.56736 | 585/1212 |
| **My Ensemble Model** | 0.59250 | 63/1212 |

- Frow above table, we can see that score of final solution is stronger than benchmark.
- The final solution is significant enough because the ranking has been improved to top 6%(63/1212).

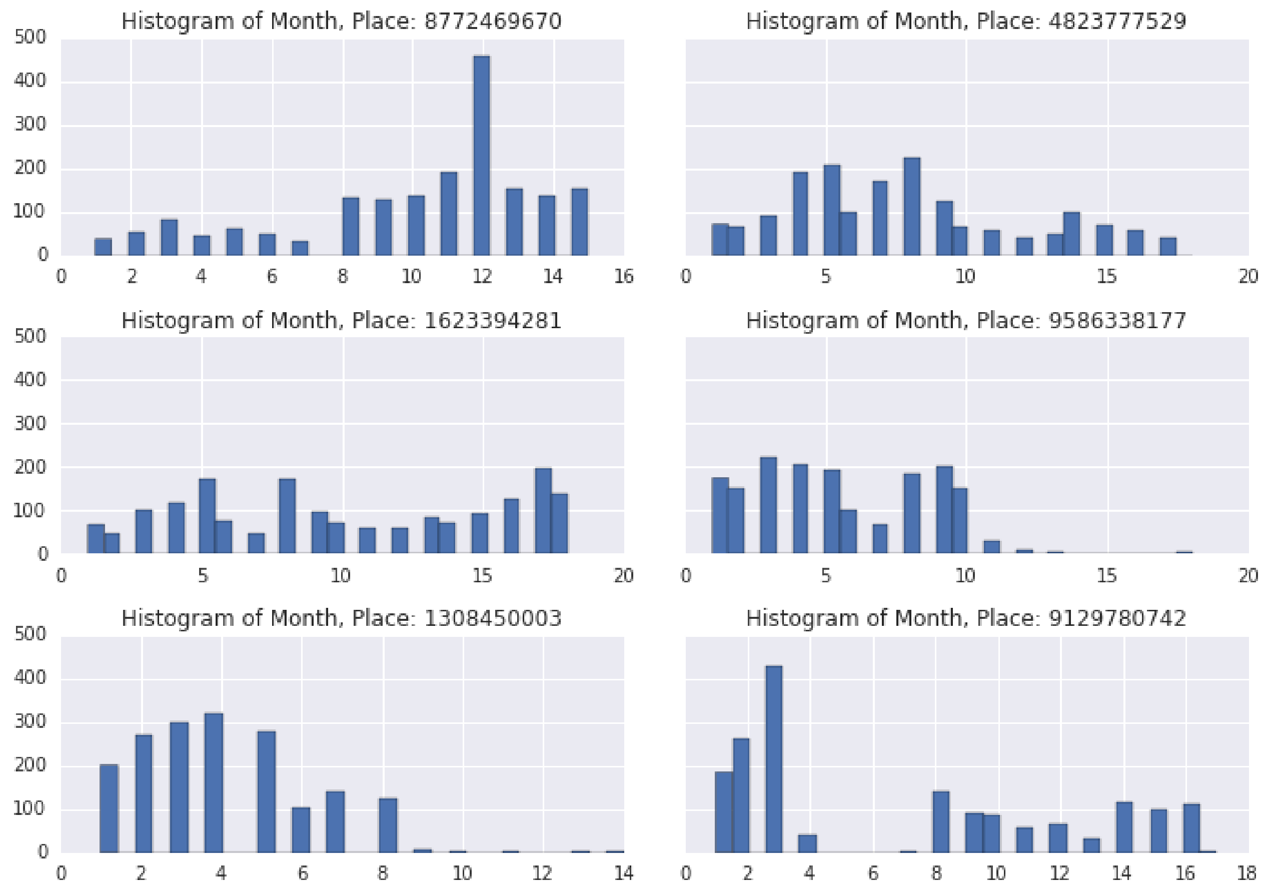
# Conclusion

## Free-Form Visualization

The most important insight into this data is the relative popularity of places, P(place), varied substantially over time (really it should be written as P(place, time)), and it seemed hard to forecast it from the training data, as following figures show.

- Reference
  - Discussion Thread
  - Blog on Kaggle

**Fig.7** Histogram of Month over top 6 popular places



## Reflection

The process used for this project can be summarized using the following steps:

1. Clearly knowing the problem definition
    a. Predicting the top 3 places a person would most likely to check in to.
    b. The probability of place could be modeled as

$$Pr(place\ id \mid x,\ y,\ time,\ accuracy)$$

    c. This problem is categorized as supervised classification.
2. Getting more information by data exploration
    a. Check-ins are roughly uniformly distributed on the map.
    b. The unit of time is minute.
    c. Majority of accuracy value is under 200.
    d. Training data and testing are splitted by time.
3. Implementing Algorithms
    a. Converting time of minute into hour, weekday, month and year.
    b. Dividing the whole map into small grid cells and labeling the number on each cell.
    c. Converting accuracy into log scale.
    d. Training three different models.
    e. Selecting features and adjust parameters of model by trial and error.

## Improvement

Based on shared methods on the forum of Kaggle, there are two aspects of ideas could help improving result:

## Learning the conditional distributions more efficient and accurate

- As Jack mentioned, estimation of histograms should consider effects of time because they are varying over time.
- Closer samples has to be weighted more, taking distribution of x as an example:

$$count(x) = count(x) + 0.8 * (count(x-1) + count(x+1)) + 0.6 * (count(x-2) + count(x+2))$$

- Reference:
    - Script by Jack

## Learning from the future

- As Markus mentioned, using just P(place_id) based on the train data, the predictions were already very good: About half of the time the first prediction was correct. So we could think of the top predictions as actually labeling the test data - just very noisily - and train on the test data using the top 10-20 or so predicted place_id's as labels with their probabilities as weights.
- This idea led to an approach of iterating between making predictions for the test data and learning from those predictions.
- Reference:
    - Script by Markus