Project Report
on
Facebook check—ins prediction

By
Harika Nukala
Sarika Padmashali

December 2016

Introduction

The purpose of this project is to predict which place a person would check into. Check-ins allow visitors to your business to "check in" on Facebook — meaning that their friends will see that they have visited your business. To predict the facebook check-ins we started off by deciding how to parse the huge corpus of data which was provided to us by Kaggle. The next step was to understand the datas set and analyze which machine learning algorithm to use inorder to predict. The third step was to incorporate knowledge representation aspect so that we can automate the things which we would have otherwise done manually inorder to improve the performance. The fourth step was to decide a search algorithm which would be feasible to search the best path to take inorder to improve the accuracy of the model. Each of these steps is explained in detail in the following sections.

Input Dataset:

Facebook created an artificial world consisting of more than 100,000 places located in a 10 km by 10 km square. For a given set of co-ordinates, our task is to return a ranked list of the most likely places. Data was fabricated to resemble location signals coming from mobile devices, giving us a flavor of what it takes to work with real data complicated by inaccurate and noisy values. Inconsistent and erroneous location data can disrupt experience for services like Facebook Check In.

The dataset consists of 30 million (simulated) check ins on facebook. It has 6 columns with headers – row_id, x, y, accuracy, time, place_id.

Description of columns:

row_id – specifies the ID
x, y – specifies the location of the place the users have checked into
time – Time is given in minutes
place_id – a set of different classes of places

Data Parsing and Cleaning:

Since the data had 30 millions records and there were around 100k different classes for place_ids we had to figure out a way to prune the dataset. With this huge number of classes most machine learning techniques would not work as expected and would fail so we thought we must work on a smaller set of classes for place_ids and decided to work on 250 by 250 meters grid in our simulated facebook world. The amount of data after pruning is still large enough to predict efficiently – 17710 records remain after cleaning.

Conversion of timestamp:

The time feature in the data is in minutes. So, we first converted the time into hour, week, month, year and day format. So that we can analyze data based hours, weeks etc and create some insights of it.

The unit of timestamp was in minutes. In the data set which we retrieved from kaggle, there was no description of the timestamp. So our first challenge was to find that the timestamp was in minutes. Usually, the timestamp is in milliseconds, however, in this particular set it was in minutes.

Visualizing using Matplotlib to extract features:

After data cleaning, we visualized the data set using the python Matplotlib library. Figure 1 shows the checkins plotted by place ids using the co ordinates x,y in the set.

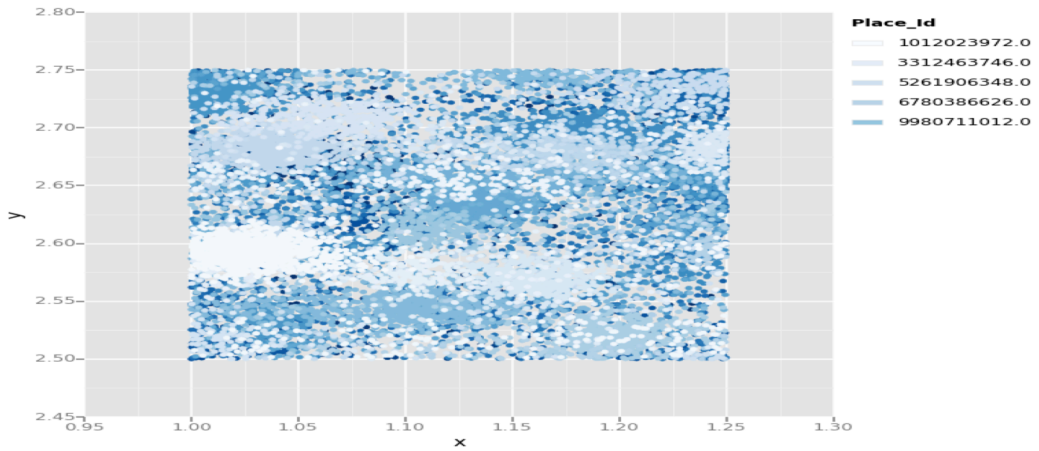

Fig.1. Plotting place_ids by considering co-ordinates

As we can see from the figure 1 that there are clusters in the data. However, most of the clusters are overlapping and hence, we cannot use the x, y features alone to predict the check—ins. We need to make use of the time dimension to see clear clusters.

Adding Time Dimension to our feature set:

We thought adding the time dimension to the set would help us get proper clusters. So we decided to add the hour component to the plot. Figure 2 is a plot of checkins by place ids after including the hour dimension to the graph. We can now see clear non-overlapping clusters. Hence, we concluded time would play an important feature in our model.

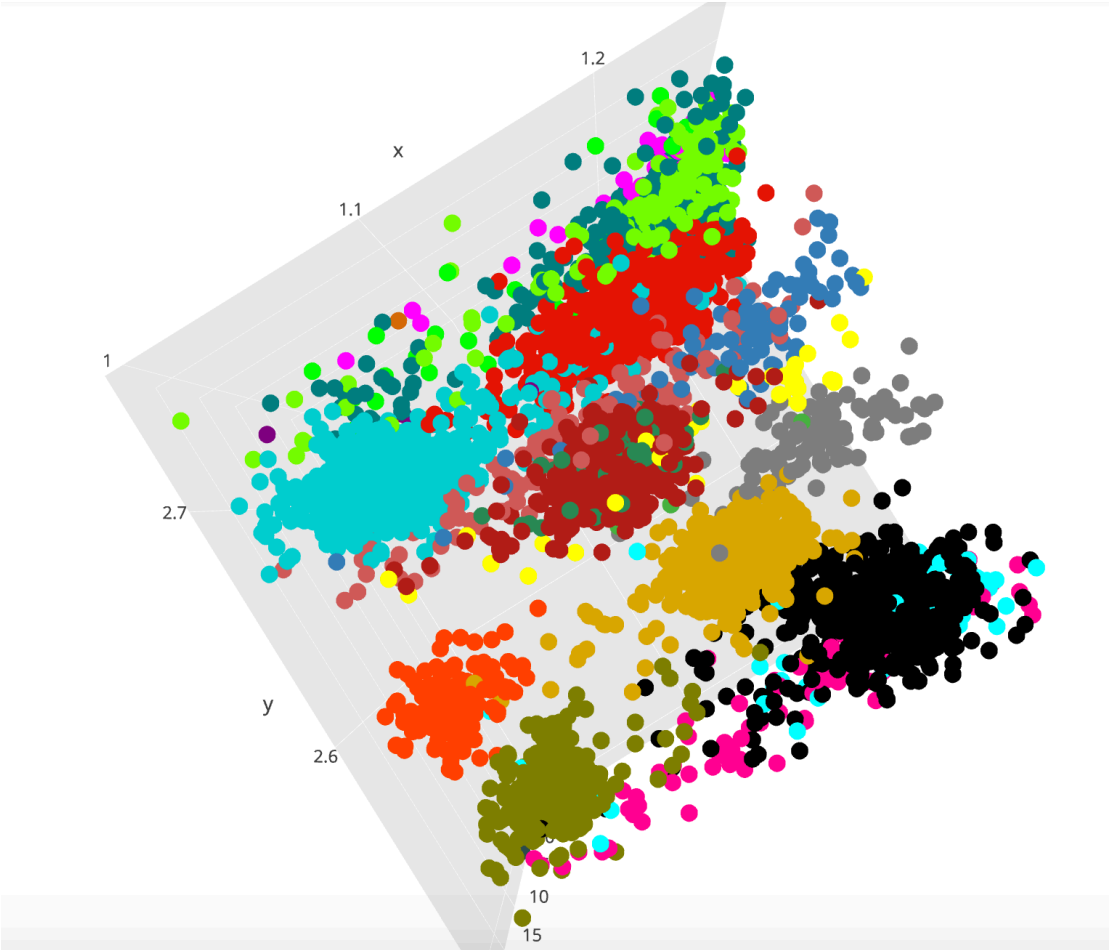

Fig.2. Plotting place_ids by adding the hour dimension

We can see that for certain places check—ins stop for a few hours and then starts picking, for some places there are peaks in

certain hours of the day and experience a downfall in certain hours etc.

Overview of the architecture:

Figure 3 shows the overview of the architecture to predict check—ins. Data is first ingested into system and goes through a python script which cleans the data and converts timestamp into hour, weekday, month and year. We then apply the machine learning technique K — nearest neighbors to predict the check-ins for the test case. For the first turn, the data set is divided into 60:40 train-test split. If the accuracy turns out to be < 90 % (90% is the threshold we have used), we then try to improve the accuracy by learning some knowledge from our knowledge repository. This is done using the knowledge base which we have developed using a series of experiments in the past. It also gathers more and more information as the number of turns increases I.e., the system learns with every turn it goes through. The inference engine uses A* searching technique to search through a tree like structure of knowledge which the machine learns and appends over time. Using this additional knowledge, the system applies the machine learning algorithm again to predict check—ins and hopes to improve the accuracy. If the accuracy reaches the threshold; it stops, else it continues to search for a new technique to improve the accuracy and keeps adding the knowledge acquired in the process. Each of the components in the architecture and its working is explained in detail in the following sections.

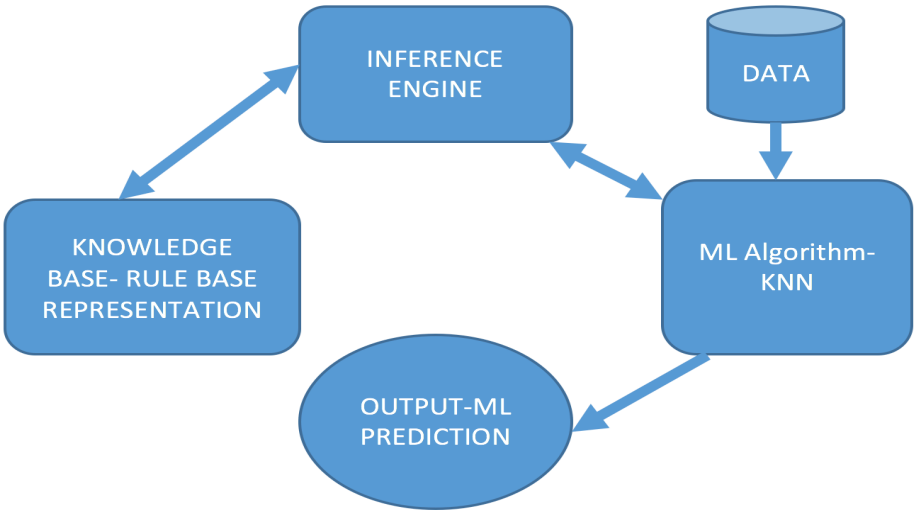

Fig. 3. Overview of the architcture

Machine Learning Algorithm K- Nearest Neighbors (KNN):

We have used the in-built KNN machine learning library in python sci-kit learn for prediction. KNN is a supervised learning algorithm I.e., it works on the data which has output labels. Also, since we are trying to find the closest popular check—ins we thought KNN would be the best machine learning technique to be used as KNN internally uses some distance function to calculate the proximity between records and we thought would work well for our model.

Figure 4 is a sample of the data set and how KNN works on it. In the figure 4 we can see that the test record is the closest to the records 1,3 and 5 (assuming K = 3, we just consider the 3 closest samples). Records 1, 3 and 5 have labels Bank of America(Bofa), walgreens and Bofa respectively. Hence, by majority vote we predict that our test sample will check—in to Bofa.

Training Sample:

Record	X	Y	Time	Place
1	9.8	5.0	11111	Bank of America
2	7.0	4.0	13444	San Jose State University
3	9.9	5.1	11113	Walgreens
4	7.8	5.2	11115	Walgreens
5	9.9	5.0	11113	Bank of America

Test Sample:

9.9	5.0	11113	?

For k = 3,
Our test sample is closest to the records 1,3 and 5. By majority of vote we can predict that the place the user is likely to check into is Bank of America.

Fig. 4. KNN working on our sample data set

Rule based Knowledge Representation:

After running the machine learning algorithm if the accuracy still falls below 90 %, then we are going to add the facts to the knowledge base — asserting the strategy we took and the accuracy we achieved using the ML algorithm in that turn. For example: In

the first turn, if we split the training and the testing data into 60:40, apply the machine learning algorithm and get an accuracy of 58%; then in our knowledge base we will assert — accuracy(0.4,58) which implies we achieved an accuracy of 58% by considering 40% of the data as the test set. In our knowledge base we also have a series of facts which are based on our experiments in the past.
By manually performing a series of steps to improve the accuracy we found that increasing the number of features helped in improving the accuracy by 20% so we assigned the highest probability to this improvement. We have assigned some probabilities to each of the path which our machine can take. The computatoion of probabilities phase in our system is explained in depth in the next section. Figure 5 shows the rules and facts which we have included in our system.

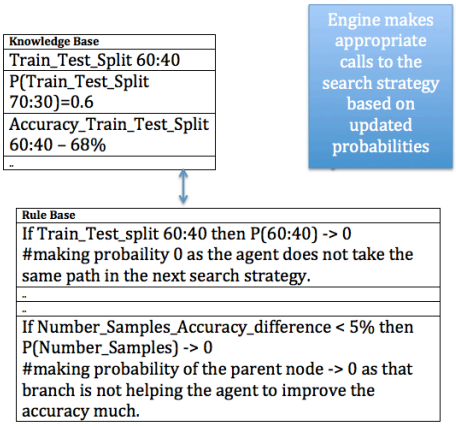

Fig.5. Knowledge Representation for our system

Inference Engine:

Figure 6 shows the inference engine technique we have used in our system. The searching strategy we used is A* algorithm. When the accuracy falls below some threshold, we usually do some optimizations and take some decisions manually to improve the accuracy. We have automated this using knowledge representation and searching technique. The searching tree shows the various strategies which can be deployed to improve the performance. The three parent strategies are: Increase the number of samples, randomize the sample set and increase the number of features. Each of these parent nodes have some leaf nodes which describe the steps

to be taken to improve the accuray. The system choses the path with the highest weight to improve the accuracy for that turn. The nodes of the tree represent the strategy and the weights represent the probabilities with which we can say that this path will improve the accuracy. We have assigned some probabilities intially based on our experiements. When some path is taken by the searching strategy we make the probability of that leaf node to 0 so that the searching algorithm does not take the same path again. If two succcessive turns uses the same parent strategy and if the difference between the accuracy is < 5% then we avoid that parent strategy altogether and go to the next strategy. For example: If the strategy 'increase the number of training samples' is selected twice and the difference in the accuracy of the previous and current turn is 2% we assert that the strategy is not working and make the probability of 'increase number of samples' branch to 0.

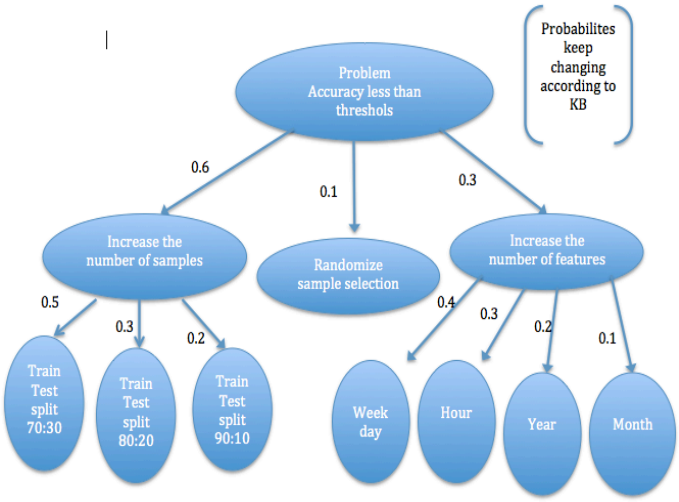

Fig.6. Inference engine describing searching strategy

Analysis And Conclusion:

We have validated our model by splitting the data in the training and testing dataset in ratio of 80% to 20%. Initially we were getting an accuracy of 53%. However by adding the searching strategy and knowledge representation to our model we have increased the accuracy. Based on the knowledge acquired during the execution turns, we have raised the accuracy to 92%. We have noticed that the "day of the week" plays an important role in predicting the popular destination a user can check into. For example: on a saturday a user is more likely to check into a pub than

to a corporate center.Figure 7 shows the effect of weekday on the plot.

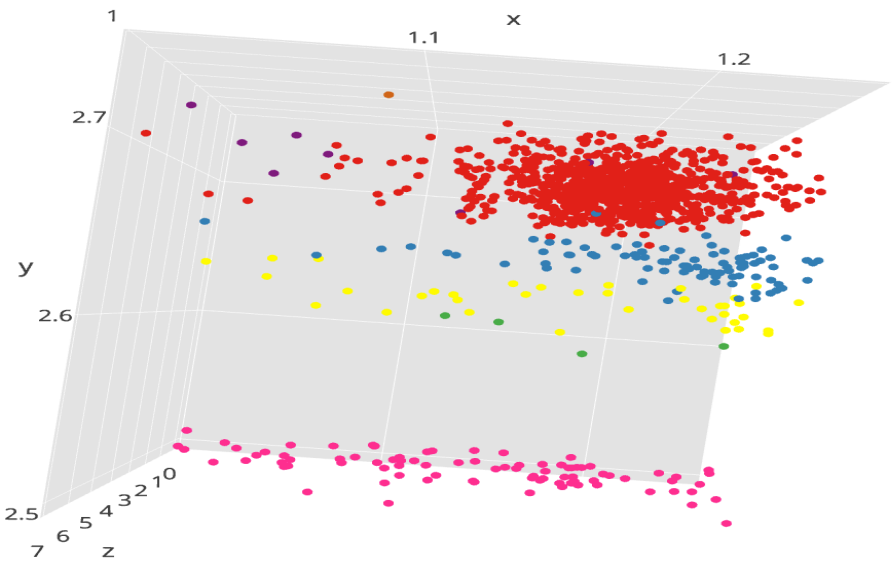

Fig.7. Plotting place ids based on weekday

Figure 8 is a snapshot of the running code evaluating our model

```
===Accuracy==== 53.0870445344
===INSERTED===
problem
inc num of samples
70:30
70:30 inc num of samples 1
===New Accuracy==== 53.0870445344
NEW BEST ACCURACY SO FAR 53.0870445344
===INSERTED===
problem
inc num of samples
80:20
80:20 inc num of samples 2
===New Accuracy==== 53.0870445344
NEW BEST ACCURACY SO FAR 53.0870445344
===INSERTED===
problem
inc num of samples
inc num of features
weekday
weekday inc num of features 1
=====New Accuracy==== 91.2322874494
NEW BEST ACCURACY SO FAR 91.2322874494
```

Fig.8. Model evaluation

Distribution of work:

We have contributed equally towards this project and have helped each other out in each of the modules coded. Since we are just two people in the project we met every week for the project and use to sit together and code. Even for the documentation and the presentation part we both have equal contributions.

Referemces:

[1] https://www.kaggle.com/c/facebook-v-predicting-check-ins

[2] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer: KNN Model-Based Approach in Classification

[3] Breiman, L. Machine Learning (2001) 45: 5. doi:10.1023/A:1010933404324