Facebook V: Predicting Check Ins

Identify the correct place for check ins

CS 6375- Machine Learning Project Report

Nikhil Kalekar NLK180002
Aditya Singh AXS180021
Habil Damania HXD170005
Shobhit Jain SXJ180033

December 2018

Contents

1	**Introduction**	**4**
2	**Pre-Requisites**	**4**
3	**Project Scope and Dataset Description**	**5**
3.1	Project Scope	5
3.2	Dataset Description	5
4	**Methodology**	**6**
4.1	Data Analysis and preprocessing	6
4.2	Dividing the grids into cells	8
4.3	Splitting data into training and testing	8
4.4	Feature Engineering	8
5	**Training and testing**	**9**
5.1	K-Nearest Neighbor	9
5.2	Gaussian Naïve Bayes	9
5.3	Random Forest Classifier	10
5.4	AdaBoost	10
5.5	XGB	10
6	**Conclusion and Future Scope**	**10/11**
Bibliography		**11**
APPENDIX I		**12**

List of Figures

Fig 1: Distribution of data in the Dataset----------6

Fig 2: Top place-IDs in the prediction----------7

Fig 3: Bottom place_ids in the distribution----------7

Fig 4: Check ins on a tiny grid map ----------7

Fig 5: Difference between the dispersion in accuracy attribute and log of accuracy attribute ------7

1. INTRODUCTION

Facebook has become one of the leading social networking websites today with more than 2.27 Billion active users. Each user has specific preferences and ways in which a user utilizes Facebook. One of the most common social activity that users on Facebook do is checking in to popular places through Facebook and letting people know where they are and what's going on in their life.. To make the popular Facebook check in more efficient for users, Facebook has decided to take the next step forward in identifying the correct place where the person is most likely going to check in to.

Facebook collects the data about its millions of active users on its platform. This data includes the users search history, posts, likes, comments, etc. For the purpose of this competition, Facebook has provided data about the Check Ins done over a time period. The challenge is to use this provided data of over 29 million check ins to over 100,000 places to identify the ranked list of the top 3 places a person is going to check in to depending on the persons location and time stamp. This is a multi-class prediction problem where the number of classes are 100,000.

For the purpose of this competition, Facebook has created a virtual city of an area of 10km * 10km with 100,000 places in it. Each data point in the dataset represents one check in. The output of the competition is to provide a ranked list of the top 3 places most likely a person is going to check in based on the persons location.

2. PRE-REQUISITES

2.1 Sci-kit learn:

Sci-kit Learn is a scientific library in python which consists of methods for various data science and machine learning models. We can specify the models and the parameters of the models as parameters to the function and train our models.

2.2 Pandas:

Pandas is a data processing library in python which is commonly used to read data from csv and json files and store it in dataframes. The pandas dataframe provides various flexibilities to modify, load, display, and handle the data in the dataframe.

2.3 Numpy:

It is a package used in python. It helps in holding a multi-dimensional generic data into a one big holder. This is a good way to store generic data.

3. PROJECT SCOPE AND DATASET DESCRIPTION

3.1 Project Scope

The challenge was to predict the top three places that the user is most likely going to check in based on the location (i.e. the x, y coordinates and the location accuracy) of the user and the timestamp associated with it. The scope for our project was drawn out as follows: Firstly, analyze the dataset and its attributes. Then, perform pre-processing on the dataset to reduce the computational complexity of the dataset and save on computing and time space. The next step in the scope of our project was to divide the entire 10 x10 grid into multiple cells and then process each cell separately. Finally, run the classifier on every cell and return the ranked list of the top three cells.

3.2 Dataset Description

The dataset was provided by Kaggle for the purpose of this challenge. The dataset was divided in two parts across two files. One was for training and the other file was for testing. Some of the attributes in the dataset were left vague on purpose and was to be considered as part of the challenge. Two such attributes were the time attribute and the accuracy attribute. After analyzing the data we figured that the time attribute was a timestamp associated with the Check Ins which means when did the event take place relative to the first event. The second attribute was the location accuracy which was the tolerance associated with the x and y coordinate attributes in the dataset. The other attributes in the dataset were the row_id, x, y, and place_id which was the class of our dataset. The place_id distinguished the 100,000+ places and was unique to every place in the virtual city.

The following were the attributes in the training dataset and their description in brief:

- row_id: id of the check-in event. Each instance in the dataset has a unique id
- x: x coordinate location on the map for a particular check in.

- y: y coordinate location on the map for a particular check in.

(Note: x and y together describe the location of the check in on the map).

- accuracy: The accuracy or tolerance of a location.
- time: timestamp of the check-in.
- place_id: id of the business or place where a check-in happens. This is the target

class of the dataset which is to be predicted. Every place in the block is recognized by its place_id.

The testing data was similar to the training data except it had its class label missing (which means it only had 5 of the 6 attributes).

The size of the training dataset was approximately 29 million data points where each point represented a particular event/check in. There is no data or representation about any person in the dataset and the dataset solely consists of events rather than individual details.

The test data provided consists of approximately 10 million data points each having no class attribute (or place_id attribute).

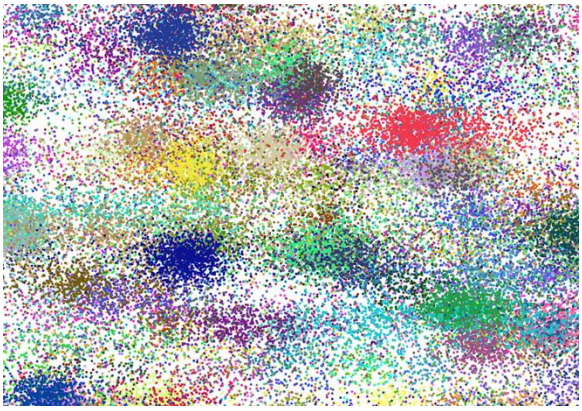

Fig 1: Distribution of data in the dataset

Fig 1 shows the distribution of the data in the dataset. As seen in Fig 1, a few spots on the grid are more densely colored compared to other spots. This shows the presence of popular places that has had a lot of check ins.

4. METHODOLOGY

4.1 Data Analysis and Preprocessing: Analysis

Before Preprocessing the data, we performed a low-level analysis on the data in hand. This gave us a deep idea of the data that we were dealing with. The following are the various analysis that we performed on our data.

4.1.1 Displaying the top 10 places in our data distribution

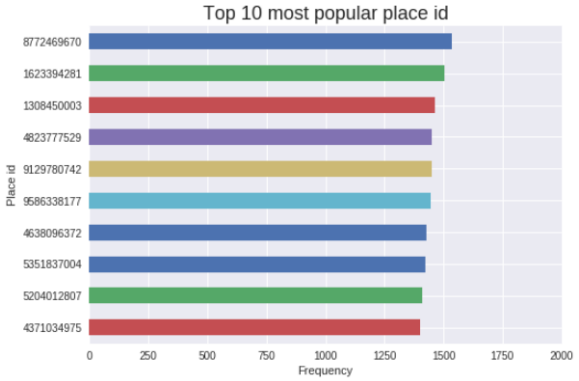

Fig 2: Top place_ids in the distribution

The place_ids in the dataset is not evenly distributed which means that some place_ids are present more times in some data points compared to others. Fig 2 displays the frequency of the classes in the data distribution showing that some places are more visited than the other classes.

4.1.2 Displaying Bottom 10 places in our data distribution

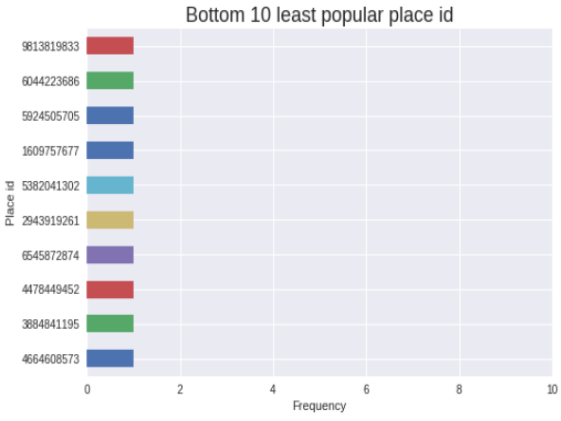

Fig 3: Bottom place_ids in the distribution

Fig 3 displays the place_ids in the dataset which have the least frequency which means that these are the places least visited.

4.1.3 Map of check ins on a tiny grid:

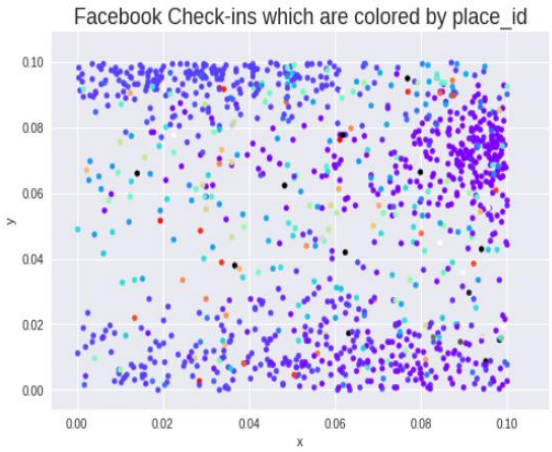

Fig 4: Check ins on a tiny grid map

We then explored a tiny grid of the entire distribution. As seen in Fig 4, the grid or the map shows the place_ids where each check ins have happened. Some of the check ins are overlapping each other in the grid. The density of distribution is more towards some places in the grid (For example: density can be clearly seen to be more towards the top right).

4.1.4 Dispersion of attribute vs log of attribute:

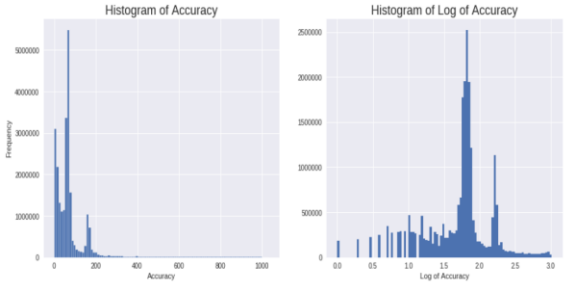

Fig 5: difference between the dispersion in accuracy attribute and log of accuracy attribute

As seen in Fig 5 the dispersion of data is more when we compute the log of accuracy attribute compared to the accuracy attribute.

Preprocessing

The first step in any machine learning project is to perform data preprocessing. It includes data cleaning, transformation, handling null values, etc. After analyzing the data in our dataset, our first step was data preprocessing. The first step in our data pre-processing was to handle the null values in our dataset. After handling the null values, the next step was to remove the redundant data tuples in the dataset. Since, every data instance in our dataset is a unique event, there is no way that we can have redundant data tuples.

The next step in our data preprocessing was the most non-trivial stage. Since, the time value for each event in the dataset is written as a timestamp, the value of the time attribute for the data instance goes to a very large number. Processing such a high valued attribute could increase the computational complexity of our model. For this purpose, we added hyperparameters such as hours, days, minutes and weekday and month in our dataset. These hyperparameters were derived from the time attribute to represent time in a higher dimension so that it would ease the computational complexity of the model.

As mentioned earlier, the accuracy column was kept vague by Kaggle and it was done so as a part of the challenge. The values of location accuracy ranged from a few tens to hundreds. To get the values of accuracy into a comprehensive range, we have computed an extra hyperparameter column as log_accuracy. For every value of accuracy, we have computed its logarithm to reduce the complexity of computation (Similar motivation as the time attribute).

4.2 Dividing the grids into cells:

The number of place_ids in the entire grid were more than 100,000 and there were more than 29 million recorded events in the dataset. That would be too much to handle at once. Therefore, we figured that an ideal solution for this would be to divide the entire grid into multiple cells and then process these cells separately. That is, return the ranked list of the top three places from each cell and then go on aggregating each cell until the entire grid is covered to finally return the top three places from the entire grid.

4.3 Splitting the data into training and testing

The test dataset provided by Kaggle had 5 attributes with the place_id (class) attribute missing. Therefore, there is no way that the accuracy can be computed on the test data. For this purpose, we extracted the validation data from our training dataset. We defined a method named train_test_split() to perform this function. This method returned the X_train, X_validation and the y_train and y_validation which were the training and validation datasets.

4.4 Feature Engineering

Feature engineering is one of the most important steps in any machine learning project. This not only decreases the computational complexity of a model but also in many cases increases the accuracy. Feature engineering Is the process by which we give more preference to some attributes compared to others. We performed feature engineering on our dataset by assigning weights to our attributes. The attributes with more weight held more significance compared to the others. We are initially assigning weights to our attributes randomly and then using trial and error to find the best weights for the attributes.

5 Training and Testing

5.1 K-Nearest Neighbor Algorithm

The K-Nearest Neighbor Algorithm (KNN) is a supervised learning algorithm which assigns the data point the class of the its nearest neighbor(s). The K value is a hyperparameter and for our model we have taken the value of k to be 25. The value of k must be an odd number to avoid a draw/tie condition. The KNN is a lazy learner, which means that it just stores the data while training and performs computations only when the test data is provided. Hence, Computations are performed only when a query is executed. The reason we decided to start with KNN was that it is a non-parametric classifier. Although, the training data is of 29 million data points, KNN being non-parametric makes no assumptions about the distribution of the data. Therefore, KNN seemed as a good starting point for our project. The methodology we used in implementing KNN was as follows. We performed KNN on each cell in the entire grid. We had earlier divided the data in the cells into training and validation data. Using this data we trained our KNN model without performing any kind of feature engineering on it.

Results:

5.2 Gaussian Naïve Bayes

This classifier uses Bayes Theorem. It helps in classification problems related to binary and multi-class classification problems. This classifier involves lot of calculation and multiplying many different numbers.
Hence, we can use calculate the probability's log transform to avoid any underflow and for better results

Results:

5.3 Random Forest Algorithm

It is used for various tasks involving regression, classification and other works. It uses ensemble learning. Its working involves creating multiple decision trees at training time and outputting the results. Its working involves correcting the habit of overfitting of over-eager learners i.e. decision tree.

Results:

5.4 AdaBoost

Its full form is "Adaptive Boosting". It also uses ensemble learning like Random Forest. It was the 1st practical boosting algorithm. It helps in making weak classifiers to strong ones. Majorly used for classification problems.

Results:

5.5 XGBoost

This estimator is also known as the Xtreme Gradient Boosting Classifier. They form a set of boosted trees and have known to give good results on several machine learning problems. One of the drawbacks that we faced while modelling this classifier was that
they consumed too much of memory which killed the kernel after several hours of training. The booster we used was 'gblinear'.

Results:

6 FUTURE SCOPE

1. **Grid Search CV Implementation-**

We can use Grid Search CV for performing an extensive search on the values of parameters. The GridSearchCV is a library in machine learning which is used to select the best combination of parameters from the specified parameters on which the model would work the best.
Hence it help us to a great extent in tuning the parameters. It eases our life, as we don't need to perform hit and trial to improve accuracy. Currently, in our dataset we have 24 million+ data entries for training data and 8 million+ data entries for testing data. Also our dataset has 10,00,00+ class labels. Hence, in order to implement it we need

more computational power than the available memory of 15GB of CoLab.

2. Increasing number of cells in Grid

Number of cells can be increased and run on machine with more computational power. Currently we have made 30 cells in grid and are predicting amongst 1,00,000 data entries only. If the number of cells are increased then we can predict amongst greater number of testing data

3. Parameter-Tuning

The future scope would be to improve the top 3 check-ins prediction accuracy. The same can be fulfilled by using different parameters and their different values. Hit nd Trial Method can be used for the same as well.

7 CONCLUSION

In this project our main aim was to predict top 3 places most likely a person is going to Check-in based on a person's location. We have used multiple algorithms to accurately predict the top 3 places. It is performed using the dataset provide by Kaggle for their challenge –"Facebook V-Predicting Checkins". Our project has Regression Problem for us to resolve.

From initial data analysis we found that in our dataset we have 24 million+ data entries for training data and 8 million+ data entries for testing data. Also our dataset has 10,00,00+ class labels. So we used total 20% of total data for our project due to computational power implementation. After analyzing the data and performing preprocessing, various ML algorithms were used like KNN, Random Forest Classifier, Gaussian Naïve Bayes, Adaboost and XGB.

Now, the best model suggested predicted the test data with an accuracy of 52.3357% using Random Forest Classifier.

REFERENCES

[1] Machine Learning, Tom Mitchell, McGraw Hill, 1997.

[2] http://pandas.pydata.org/pandas-docs/stable/

[3] https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

[4] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[6] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

[7] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[8] https://docs.scipy.org/doc/numpy/

[9] https://scikit-learn.org/stable/modules/neighbors.html

[10] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[11] https://xgboost.readthedocs.io/en/latest/python/python_api.html

[12] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[13] https://scikit-learn.org/stable/modules/naive_bayes.html