CS 6350.002

BIG DATA MANAGEMENT AND ANALYTICS

SPRING 2018

PROJECT:

FACEBOOK V: CHECK-INS PREDICTION

PRIYANK SHAH (pss160530)

ABHISHEK JAGWANI (alj160130)

MOHAMMED ATHER AHMED SHAREEF (mxs166331)

JAHNAVI MESA (jxm169230)

Introduction and Problem Description

With the dawn of data-driven economy, big data has emerged as the secret to success that every company is keen on implementing. Since its inception, Facebook's growth is due in part to big data. It has become a producer of big data and will continue to fuel its growth with it. Facebook has become one of the world's largest repositories of personal data, with an ever-growing range of potential uses. That's why the monetization of data in the social network has become paramount.

Check-ins of users have gained hotness on the social media platform and an important task which is associated with it is to recommend a check-in location to the user. This makes us interested to consider the 'Facebook V: Predicting Check Ins' dataset on Kaggle. In most of the recommender system, we develop a user profile, but here the task is to recommend check-ins to the user simply based on the location and the time of a user.

The main objective of the project is to predict which place a person would like to check-in to, based upon the data which is provided by the Facebook. Facebook has generated an artificial data consisting of more than 100,000 places located in a 10 km by 10 km square. For a given position of coordinates, the job is to provide a ranked list of the most likely places. Data provided was fabricated to resemble location signals coming from mobile devices, giving a flavour of what it takes to work with real data complicated by inaccurate and noisy values. Inconsistent and inaccurate location data can obstruct activity for services like Facebook Check-In.

Dataset Description

In this section, we are examining the dataset as well as its features. The dataset consists of nearly 30 million check-ins with around 38000 unique places in a 10km X 10 km grid.

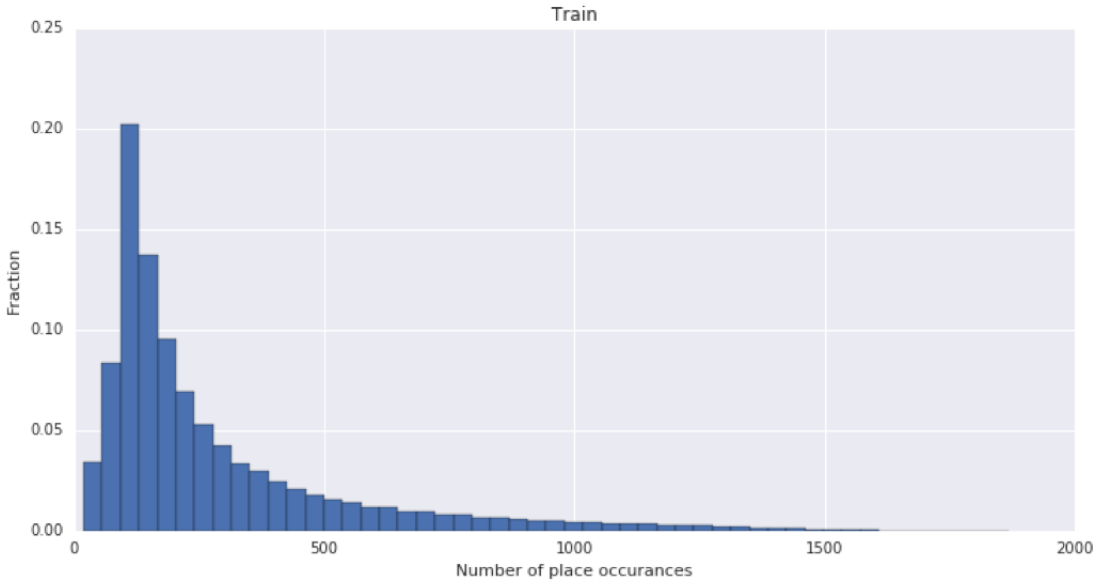
The train and test data are split based upon the time and there is no concept of person in the dataset. All the row id's in the dataset are the events and not people.

The training data consists of approximately 29 million observations where the location (x, y), accuracy, and timestamp is given along with the target variable, the check in location. The test data contains 8.6 million observations where the check in location should be predicted based on the location, accuracy and timestamp.

The description of the fields of our data is as follow:

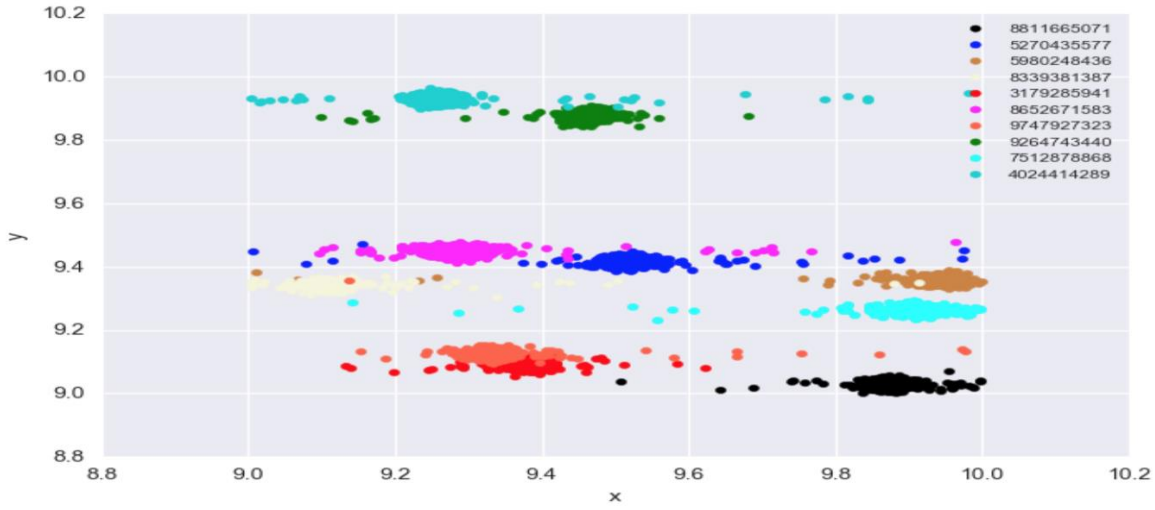
- Row_id: This field specifies the record number of check-ins.
- X: This field specifies the x- coordinate bounded between 0 and 10
- Y: This field specifies the y- coordinate bounded between 0 and 10
- Accuracy: This field specifies the accuracy with which the check-ins was made. This represents the (x,y) location accuracy which is dependent on the environmental factors, device characteristics etc.
- Time: Here, we assumed that it is the number of minutes passed since some reference time.
- Place_id: This field represent the place at which the check-ins was made and is acting as a unique identifier.

The given data has approximately **38000** place_ids into which every check-ins are classified. For making the dataset computationally viable for our resources, we are considering only the 1km X 1 km grid from the bottom-left corner of the artificial world given by Facebook. This makes our dataset to approximately **250000** check-ins with nearly **8000** place_ids.

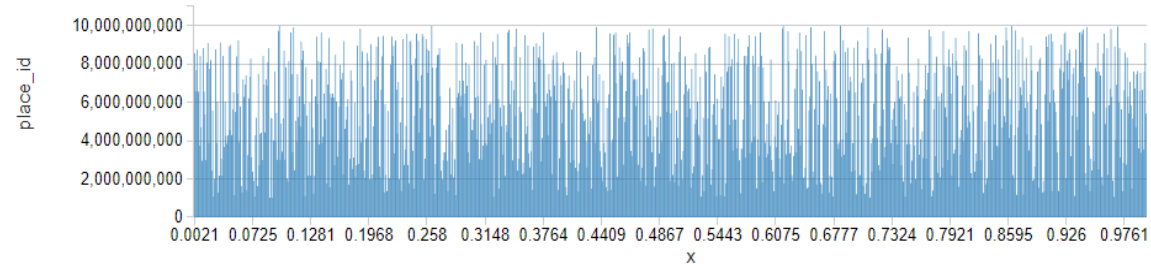


The above figure represents the fraction of the number of places occurring in the train dataset. The long tail represents that the place is very famous for the check-ins. We also observe that some places are more popular as compared to others in terms of check-ins.

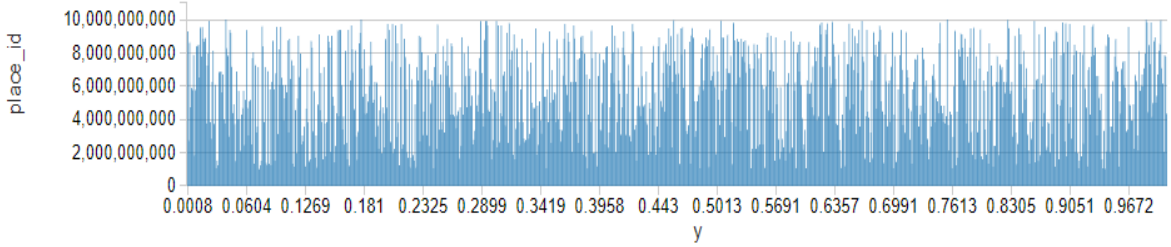
1) x, y co-ordinates



The above diagram shows the distribution of the place_ids with the grid location of x and y.
The above distribution for the ten most places shows that the variations along x axis is much more as compared to y-axis.



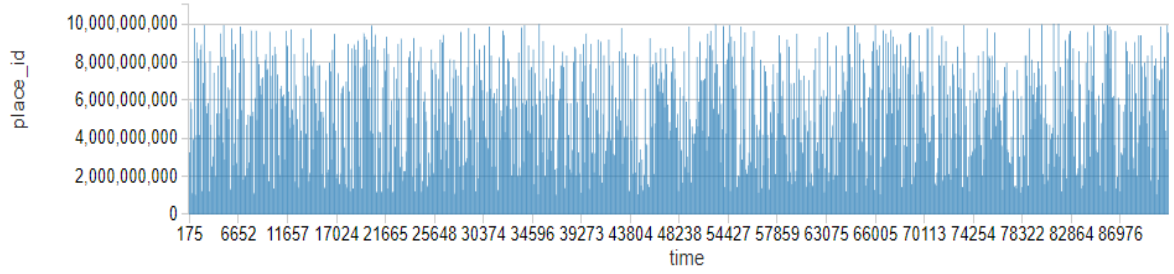
The above diagram shows the distribution of the place_ids in a bar chart against x co-ordinates.



The above diagram shows the distribution of the place_ids in a bar chart against y co-ordinates.

2) Time

The time factor from the data is intentionally left out as it has a variety of values and it has been started from some given reference time which is totally unknown. Thus, we pick up the most popular places based on the number of check-ins a user has done in different places to observe the varying difference n check-in patterns. Time in the data set is varied across a span of 24 hours for various days. The check-ins are varied by time based upon the popularity of the place during day/night time.



The above diagram shows the distribution of the place_ids in a bar chart against time.

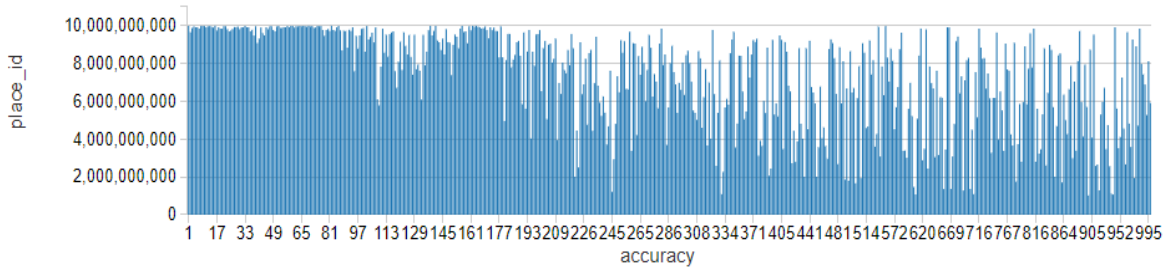
3) Accuracy

Accuracy values in the dataset are very above 100. As a result, we can:

i) Normalize the accuracy values for a specific range or

ii) Consider the same values and consider the accuracy values based on comparison between them.

To analyse the full data set, bar plots can be generated for place_id v/s accuracy values. But due to a varied number of different labels/classes, the bar plot can be very compact and very difficult to analyse the data over accuracy.



The above diagram shows the distribution of the place_ids in a bar chart against accuracy.

Due to high number of classes/labels present, the plots are very compact.

Pre-processing Techniques

- Our dataset was clean and we used 1km x 1 km grid out of 10km x 10km available area grid area in-order to make the computation viable for our available resources.
- We are using string indexer for place_id as our values in that field was Long Integer.
- We are converting our 4 fields (x, y, accuracy and time) to feature vectors for doing feature selection

Proposed Solution and Methods

The attribute which we need to predict has approximately 38000 classes. Hence, we are suing multi-class classifiers to solve this challenge. The following are the multi-classifiers which we are using in our project:

1) **Random forest classifier:** Random forest classifier are kind of ensemble methods which use decision tree. It is one of the most successful machine learning technique for classification and regression. To reduce the risk of overfitting, they combine the decision trees. The parameters which we tuned to find the best classification are:
    - numTrees: It is the number of trees in the forest.
        1. We can decrease the variation in the predictions by increasing the number of tress in the forest.
        2. But, the amount of time taken to train the model also increases roughly linearly with the increase in the number of trees in the forest.
    - maxDepth: It represents the maximum depth of each tree in the forest
        1. The model will become more powerful and expensive as the depth of each tree will increase. But, deep trees are prone to overfitting and will take more time to train on the training data.

2) **Logistic Regression**: One of the popular method to predict the categorical response. It is the special case of the generalized linear model that can predict the probability of the outcomes. Here, it is using multinomial logistic regression to predict multiclass outcome. The parameters which we tuned here are as follows:
    - maxIterations: It is the max number of times this classifier can run.
    - Regularization: It is used to control the overfitting

3) **Decision Trees:** Decision Trees are basically a type of classifiers that majorly focus on classification area using axis parallel cuts. They can be very effective when the data is not featured scaled or when you have high number of different labels/classes with few feature attributes. The parameters which we tuned here are as follows:
    - maxBins: By incrementing this parameter, we can actually tune and set the discretizing continuous features.
    - maxDepth: It helps to adjust the maximum depth that each tree can grow during fitting and training of a model.

We used "F-1 measure" metric to select the best parameter values as each of the classifier model is tuned for various parameter values accordingly.

Further, Cross validation techniques are used to generalize the dataset. For tuning, various K number folds are used.

The Following measures are taken into consideration to choose the best classifier amongst all used:

1) Accuracy

2) Precision

3) Recall

4) F1 – Score

We used following to achieve the functionalities of our project:

- Scala, Spark (Languages)
- Spark MLlib (Library)
- CrossValidator, ParamGridBuilder (Tuning)
- StringIndexer, VectorAssembler (Feature Extraction)
- Pipeline (Serializable Execution)
- MultiClassClassifierEvaluator (Evaluation of Metrics)
- AWS (Jar file Execution)

Experimental Results and Analysis

Classifier	Best Parameters	Number of Folds	Accuracy	Weighted Precision	Weighted Recall	Weighted F-1-Score
Random Forest	Max Depth: 8 Num Trees: 20	10	0.4725	0.4087	0.4625	0.3874
Random Forest	Max Depth: 8 Num Trees: 30	10	0.4795	0.4129	0.4895	0.3941
Random Forest	Max Depth: 8 Num Trees: 40	12	0.4801	0.4212	0.4701	0.3997
Logistic Regression	Max Iterations: 100 Regularization: 0.0	10	0.4311	0.3814	0.4311	0.3765
Logistic Regression	Max Iterations: 100 Regularization: 0.0	12	0.4359	0.3885	0.4359	0.3788
Logistic Regression	Max Iterations: 100 Regularization: 0.0	8	0.4337	0.3869	0.4337	0.3801
Decision Trees	Max Bins: 50 Max Depth: 8	10	0.4508	0.4057	0.4508	0.3991
Decision Trees	Max Bins: 50 Max Depth: 8	12	0.4622	0.4083	0.4622	0.4002
Decision Trees	Max Bins: 35 Max Depth: 8	10	0.4599	0.4046	0.4599	0.3984

***Reason behind getting very low accuracy is due to a large number of classes/labels. Additionally, a reference paper link has been mentioned in "References" section which also shows similar accuracy results.**

Following are the best results derived from the above observations:

1) Random Forest

- numTrees: 40
- maxDepth: 8
- Accuracy: 48.01%

2) Logistic Regression

- maxIterations: 100
- Regularization: 0.0
- Accuracy: 43.59%

3) Decision Trees

- maxBins: 50
- maxDepth: 8
- Accuracy: 46.22%

4) The accuracy results from the above observations are in the following order:

Random Forest > Decision Tree > Logistic Regression

5) Random Forests and Decision Trees performs better than Logistic Regression for precision metric.

6) The Recall values are higher for Random Forests and Decision Trees compared to Logistic Regression.

7) There is very minute difference between all the 3 models when F-1 score is taken into consideration.

Conclusion

After analysis of results, we can conclude that Random Forests performs best for our dataset and outperforms other models with the following tuning parameters: Number of Trees: 40, Max Depth: 8. An accuracy of 48.01% was achieved in comparison to best accuracy for Decision Trees 46.22% and Logistic Regression 43.59%. According to the best Kaggle submissions, Random Forests performs best in almost of all the cases given. The base part for the project is to pre-process, analyse the data - to further select a proportion of data for training and testing with the best accuracy results – and playing with tuning parameters. This project gives us a low accuracy due to high number of classes/labels.

Contribution of Team Members

Priyank Shah: Design, Implementation and Evaluation of Decision Trees Model

Mohamed Ather Ahmed Shareef: Design, Implementation and Evaluation of Random Forests Model

Abhishek Jagwani: Data Pre-processing, Data Analysis, Graph Plots and Integration of various Machine Learning models.

Jahnavi Mesa: Design, Implementation and Evaluation of Logistic Regression Model

References

- Reference paper
  https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a102.pdf
- Random Forest Model
  https://spark.apache.org/docs/latest/mllib-ensembles.html
- Logistic Regression Model
  https://spark.apache.org/docs/2.1.0/ml-classification-regression.html
- Metric Evaluation
  https://spark.apache.org/docs/2.2.0/mllib-evaluation-metrics.html
- Databricks Plots and Visualization
  https://docs.databricks.com/user-guide/visualizations/index.html
- Decision Tree Model
  https://spark.apache.org/docs/latest/mllib-decision-tree.html