PROJECT

EXPEDIA USER BEHAVIOR PREDICTION

DATE

10/21/2016

CHENG JI

Agenda

- Overview
- Dealing with big data with AWS
- Dealing missing values with big data
- Model selection / optimization with big data
- Conclusion / Next Step

Overview

- The goal: predict hotel choices of Expedia users based on previous activities and characteristics.

Data

- (insanely) huge dataset: 4g csv file with over 37 million rows, 24 columns.
- Features:
- date_time, site_name, posa_continent, user_location_country, user_location_region, user_location_city, orig_destination_distance, user_id, is_mobile, is_package, channel, srch_ci, srch_co, srch_adults_cnt, srch_children_cnt, srch_rm_cnt, srch_destination_id, srch_destination_type_id, hotel_continent, hotel_country, hotel_market, is_booking, cnt, hotel_cluster.
- A lot missing values (about 1/3) in orig_destination_distance, which I believe intuitively will be very important in predicting hotel choices.
- Everything label encoded.

Dealing with big data

- Utilize AWS EC2 and S3 services
- Machine: 64 cores, 256g ram
- Write a script to ping AWS server every 120s to keep connection alive

Dealing with big data

- Game Plan:
- Take a small sample from the dataset
- Build a missing value imputation model based on the sample, and impute the whole dataset
- Conduct feature selection, model selection, model optimization with the sample
- Use the best optimized model to fit the whole dataset and make prediction

Dealing with big data

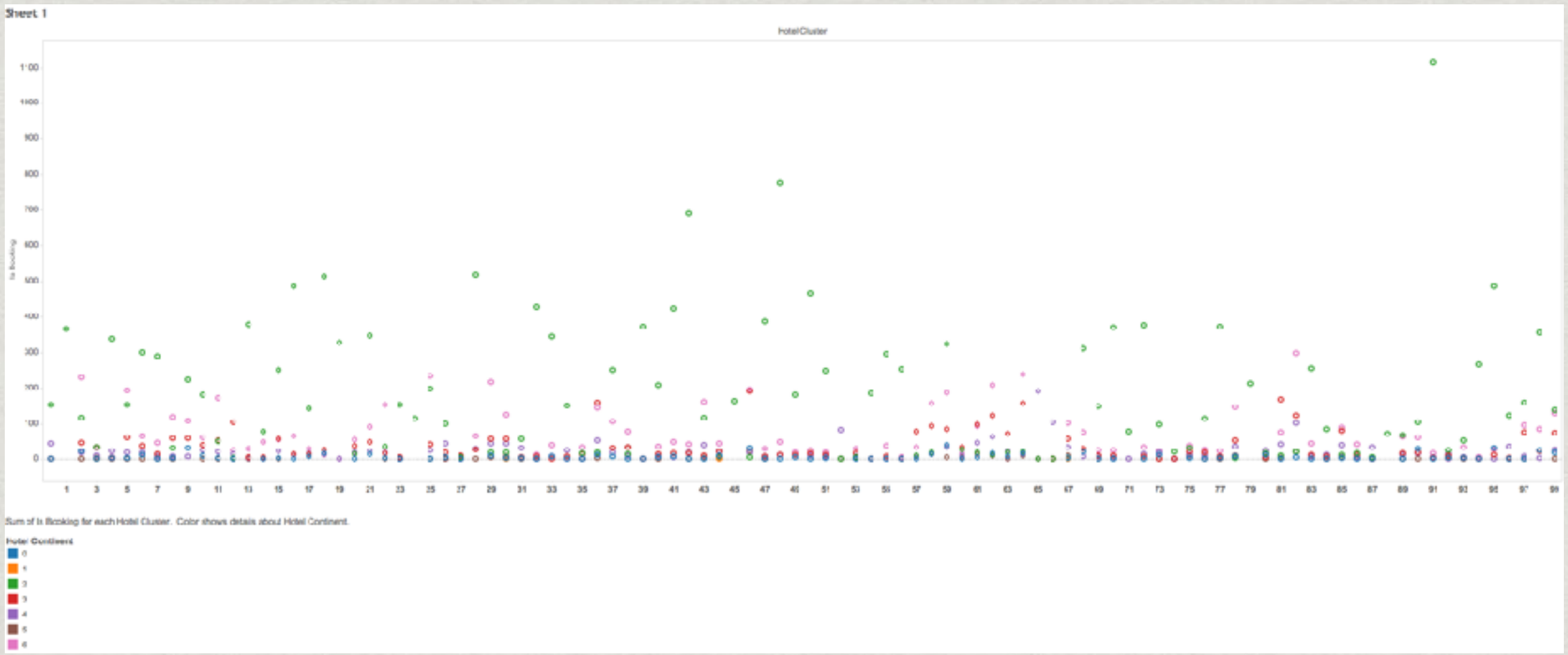
- Initially tried 10% sample (3.7 million rows), still too much to handle. Go with 1% in the end.
- All computationally expensive methods are unavailable due to budget / time constraint: cross validation, grid search, etc.

Dealing with missing value

- Over 13 million missing value in orig_destination_distance, which I believe is too important to drop.
- Solution: Random Forest Regressor imputation.
- Why Random Forest Regressor: easy to handle categorical features
- Features used: user_location_country, user_location_region, srch_destination_id, hotel_continent, hotel_country, hotel_market
- r2 on test set: 0.99
- Build model on non-missing value subset of sample data, and impute values on the whole dataset

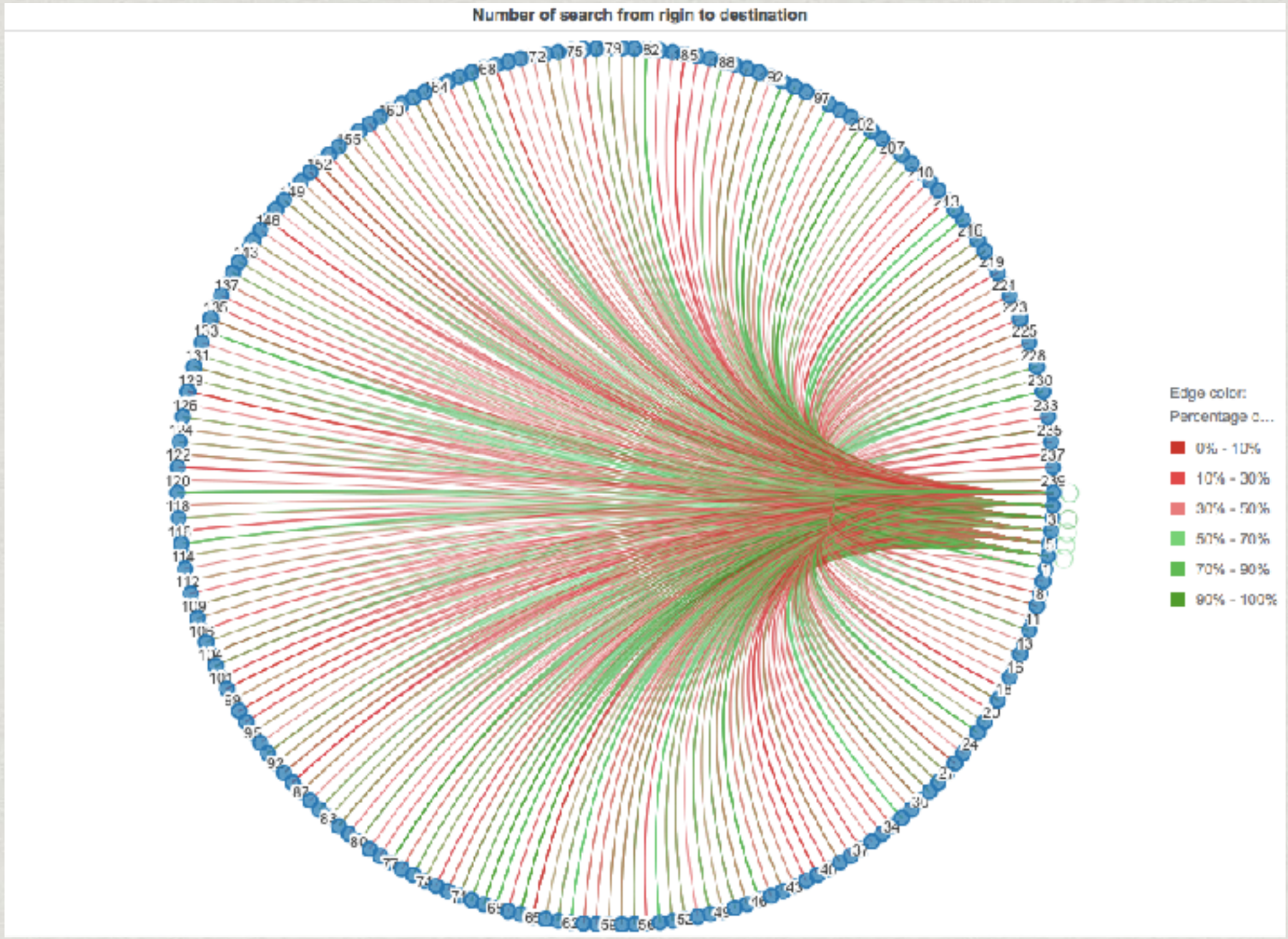
EDA

- Booking numbers by hotel cluster and hotel continent



made with Tableau

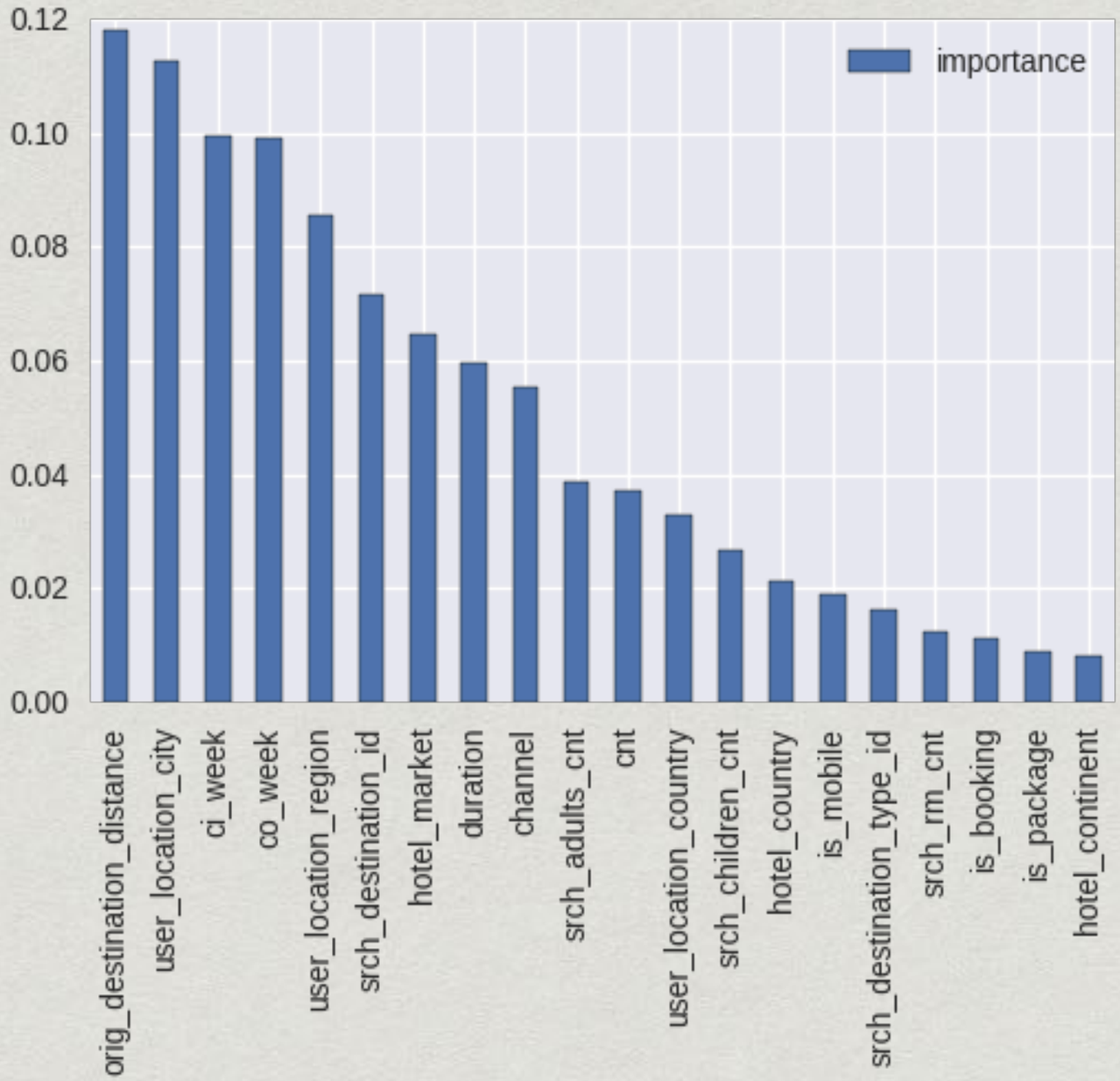
EDA



made with MicroStrategy

Feature Selection

- Random Forest Feature Importances



Model Selection

- Random Forest, Gradient Boosting, Neural Network, XGBoost.
- Also tried SVM, too computationally expensive, never finished training.

Model Selection

- Model performance with default parameters:
- RF: accuracy=0.0897, map5=0.1480
- XGB: accuracy = 0.1271, map5 = 0.2131
- GB: accuracy=0.1370, map5=0.2279
- NN (tuned): accuracy=0.1013, map5=0.1766
- map5: mean average precision at cutoff 5

Model Optimization

- Choose the best models (Gradient Boosting and XGBoost) to optimize.
- Assumption: the model performed the best with default parameters will perform the best after optimization, which may not necessarily be the case. Again, have to take efficiency / budget into consideration.
- Why neural network performed poorly: possibly stuck in local minima.

Model Tuning

- How to optimize model when gird search is too expensive? Apply yourself!
- manually tune gradient boosting and XGBoost:
  - Tree based parameters: max_depth, min_samples_split, min_samples_leaf, max_features, subsamples
  - Boosting setting parameters: learning rate, n_estimators

Model Tuning

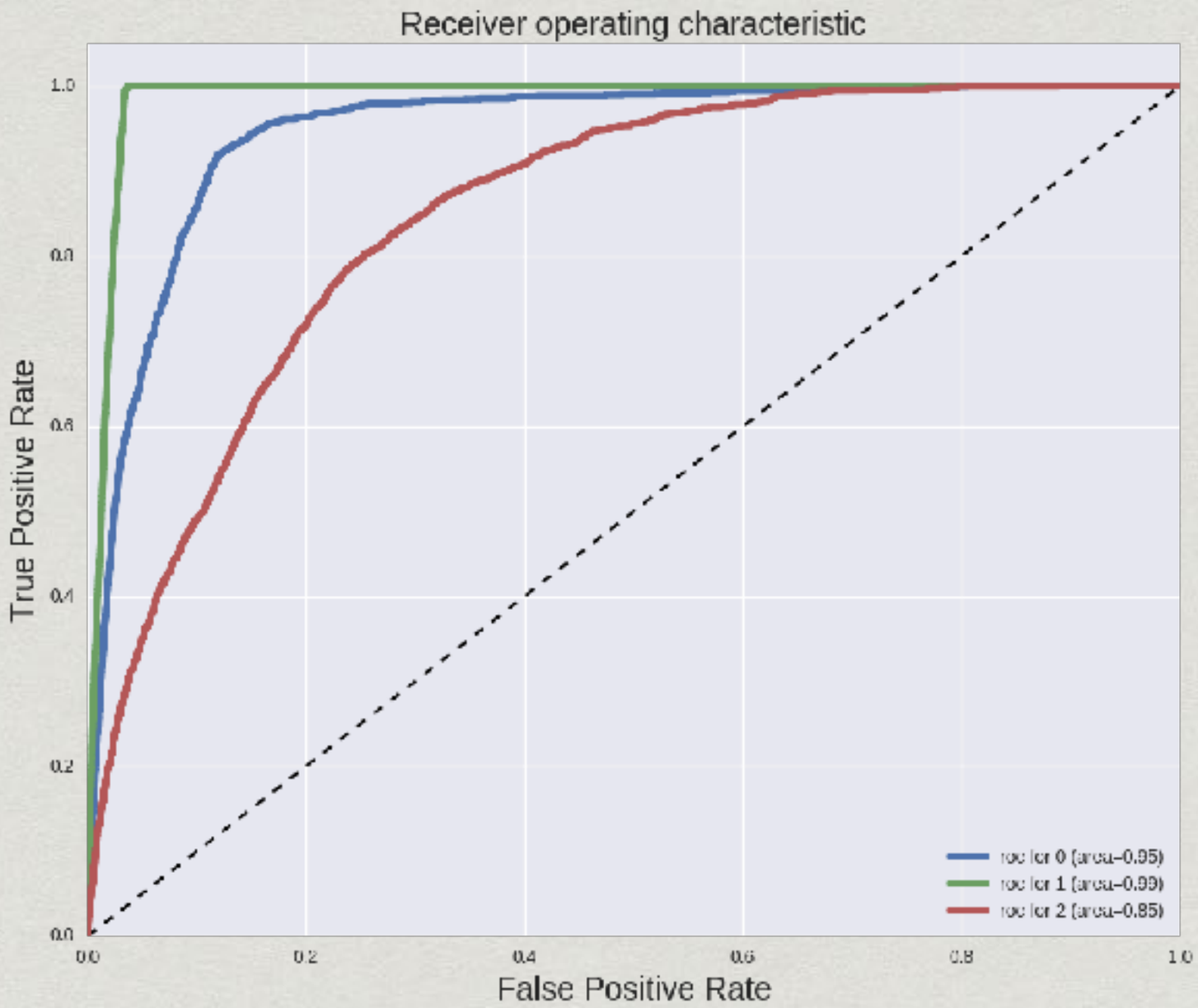
- Tree based parameters: underfit vs overfit.
- Boosting setting: lower learning rate always better given sufficient trees. However, too many trees may overfit.

Model Tuning

- Model performance after tuning:
- GB: accuracy=0.1339, map5=0.2268
- XGB: accuracy=0.1468, map5=0.2453
- Certainly not achieve optimum due to time / budget constraint.

Model Tuning

- ROC curve of XGBoost



Make prediction

- make prediction with the tuned model on the whole dataset.
- It took 1 hour to complete one training of the best model on the 1% sample. Too poor to fit the whole dataset.

Conclusion / Next Step

- Don’t tell me you know computational expense if you haven’t handle big data.
- Most efficient model on big data: Random Forest (parallelized) , least efficient: SVM (serialized and based on distance)
- Boosting performs better than neural network
- Next Steps (if I get rich):
- build models on 10% samples
- optimize all models