Delivery Time Prediction Report

Wei Guo, guowei123@gmail.com, 12/21/2017

Features used in models

- Continuous
  - 'total_items'
  - 'subtotal'
  - 'num_distinct_items'
  - 'min_item_price'
  - 'max_item_price'
  - 'total_onshift_dashers'
  - 'total_busy_dashers'
  - 'total_outstanding_orders'
  - 'estimated_store_to_consumer_driving_duration'
- Categorical
  - 'market_id'
  - 'store_id_rebinned'
  - 'store_primary_category_rebinned'
  - 'order_protocol'
  - 'created_at_month'
  - 'created_at_dayOfWeek'
  - 'created_at_hour'
  - 'created_at_isWeekend'
  - 'created_at_isHoliday'
  - 'estimated_order_place_duration_rebinned'

Feature engineering

- The target variable has 7 missing values and quite a few outliers. The 7 missing values were removed. The outlying target variable is capped at 3600 seconds (1 hour), which aligns with majority of the records. This was proved to be useful in model's performance.
- For continuous features, missing values were imputed with mean. For categorical features, missing values were imputed with a new level such as "Unknown".

- Categorical features with too many levels (high cardinality) such as store_id and store_primary_category are handled this way:
    1. Calculate the number of orders in each level (count)
    2. Count can be used as a continuous feature but I re-group the original feature based on the counts
    3. Save the count table for scoring use
- For some categorical features such as order_protocol, re-grouping into fewer levels so that each level has enough data.
- Categorical features were One-Hot encoded and continuous features were scaled.

Modeling

I have tried the following models and found that GBM is the best:

- Linear Regression
- Linear Regression with LASSO
- Linear Regression with Ridge
- Random Forest
- GBM

Hyper-parameters in all models (except Linear Regression) were tuned using Grid Search. Model performance is shown in table below:

Model	RMSE in validation (in seconds)
Linear Regression	589.96
Linear Regression with LASSO	590.10
Linear Regression with Ridge	589.96
Random Forest ('bootstrap': True,'max_depth': 5, 'max_features': 20, 'min_samples_leaf': 10, 'min_samples_split': 50, 'n_estimators': 1000)	644.03
GBM {'n_estimators': 500 'learning_rate': 0.1, 'max_depth': 5, 'min_samples_split': 10 }	566.91

How to score new data

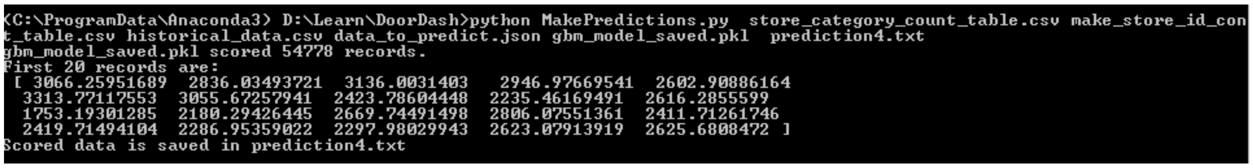
0. Required packages:
    a. pandas
    b. numpy

c. matplotlib
d. sklearn
e. math
f. json
g. sys

1. Put those files in the same directory
    a. MakePredictions.py
    b. ScoringUtilities.py
    c. gbm_model_saved.pkl
    d. data_to_predict.json
    e. historical_data.csv
    f. make_store_id_cont_table.csv
    g. store_category_count_table.csv
2. Open command window (I tested in Anaconda Command Prompt in Windows OS) and CD to above directory
3. Run this command:

    ```
    python  MakePredictions.py  store_category_count_table.csv make_store_id_cont_table.csv historical_data.csv data_to_predict.json gbm_model_saved.pkl predictions.txt
    ```

4. In seconds, a file "predictions.txt" will be generated in the directory, in command window you will see the following screen



```
(C:\ProgramData\Anaconda3) D:\Learn\DoorDash>python MakePredictions.py  store_category_count_table.csv make_store_id_con
t_table.csv historical_data.csv data_to_predict.json gbm_model_saved.pkl  prediction4.txt
gbm_model_saved.pkl scored 54778 records.
First 20 records are:
 [ 3066.25951689  2836.03493721  3136.0031403   2946.97669541  2602.90886164
  3313.77117553  3055.67257941  2423.78604448  2235.46169491  2616.2855599
  1753.19301285  2180.29426445  2669.74491498  2806.07551361  2411.71261746
  2419.71494104  2286.95359022  2297.98029943  2623.07913919  2625.6808472 ]
Scored data is saved in prediction4.txt
```

Potential improvement

- Features to try
    - Restaurant info:
        - ratings/age/work hours/location/popularity
    - Destination info:
        - Location
        - Residential or commercial (office building)
    - Accurate weather info
        - hourly update
    - Driver info:
        - history of driving (accidents, tickets, claims, ratings)
    - Vehicle info:

        - make, age
        - Real time speed of the delivery vehicle
    - Route traffic status:
        - congestion, accidents, construction blocks
        - the trip is Highway or local
        - Speed limit on the route, especially lower ones
- Modeling
    - Fine tune model hyper-parameters
        - Due to time limit, I only made a simple and small grid
        - Try to make the grid larger
        - Try other other methods such as random search
    - Ensemble several models
    - Try XGBoost
        - ran out of memory in my machine
    - Deep Neural Nets may help
        - No time to contruct complex graph
    - Fine tune features:
        - Transformation
        - better binning
    - Tried cross-validation but out of memory, did train-test split instead

Real time implementation thoughts

- The scoring algorithm must be very fast, algorithms like KNN should be avoided since it takes long time to lookup in the training data to find similar points.
- Too complex models might slow the scoring process and make it hard to maintain and debug.
- For features that need to look up values in some “look-up” table, the look-up table has to be cached in advance and accessible quickly.
- If API call is needed to create features, the API’s latency has to be considered. The ability of processing huge volume of calls needs to be verified.
- Since new data is accumulated very fast, how often and in what method the model should be updated. There are several choices such as batch learning, SGD to consider for **online learning**.

- How to parallel model’s online training process. Models like Random Forest might be better choice.
- Develop model performance tracking system since the data pattern might shift and performance may change.
- There should be some mechanism that filters outliers, so that model updating is not be adversely impacted.

Thanks for reading! Any comments and feedback are greatly appreciated ☺