Preface:

Dear Professor Zhang and TA Mr. Yin:

In xgboost, it is not only time-consuming but also **impossible to check the ratio of completion**. Due to these two inconveniences, I do not use gridsearchcv. Instead, I choose to test each parameter in each value separately, so that I can better control the process.

Classification

(a): Initial Data Pre-processing:

Features selection:

source_type, grid_distance, target_lng, target_lat, urgency, source_lng, source_lat.

Dependent variable:

action_type:

Data processing:

(1): **Data transformation**:

The dependent variable [‘action_type’] consists of PICKUP and DELIVERY. A function named dummy(x) is created to transform the two label into 0 and 1.

The feature [‘source_type’] consists of ASSIGN, PICKUP and DELIVERY. It is transformed to 0-1 dummy variables by the function pd.get_dummies().

(2): **splitting data**:

The final ratio of train test data is 509604:25468. Therefore, I do not split the training data using train-test-split with a test_size=0.3. Instead, I split the data into 484136:25468, which approximates more the real data.

(3): **outlier elimination**:

The outlier detection has taken a lower_limit and an upper_limit rules.

The first quantile is Q_1, the third quantile is Q_3. The IQR is Q_3-Q_1.

Lower bound is Q_1-1.5*IQR, upper limit is Q_3+1.5*IQR.

Any value outside the lower bound and upper bound in train data would be defined as an outlier and eliminated.

(b): Baseline Model:

The experiment has taken Logistic Regression as a baseline model.

The model has been appropriately processed, and the F1-score in Kaggle is 0.71589.

(c): Any Model (gradient boosting tree):

In this part, the gradient boosting tree is the main theory to fit the model.

Still using the data that already been processed.

Inside the xgboost, there are five parameters that need to be finetuned.

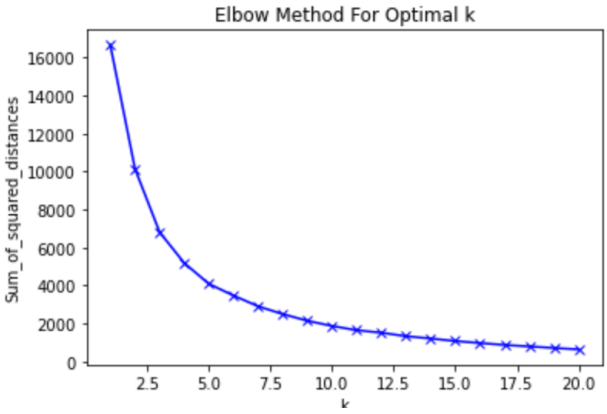
The benchmark of five parameters takes: colsample_bynode=0.8, learning_rate=0.1, gamma=0.001, max_depth=20, n_estimators=200. The in-sample F1-score is: 0.939.

After finetuning, the five parameters are: colsample_bynode=0.9, learning_rate=0.3, gamma=0.01, max_depth=18, n_estimators=500. The in-sample F1-score is: **0.96328.** And out-of-sample F1-score (in kaggle) is **0.96582.**

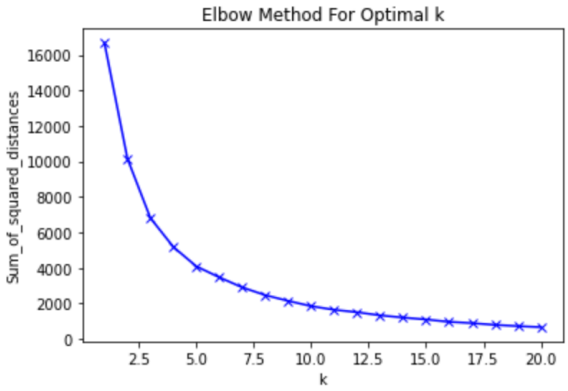
(d): Feature Engineering:

Method: Clustering, Elbow Approach.

First, I use elbow approach to judge the source location and target location:



(source-lng and Source-lat)



(target-lng and target-lat**)**

From the elbow approach, we can set the k at 5.

After finetuning, the best parameters are: max_depth = 15, n_estimators = 500. learning_rate = 0.3. colsample_bynode=1.0. gamma=0.01.

The in-sample F1-score is 0.96576, the out-of-sample F1-score is **0.9652.**

F1-score	without clustering	with clustering
in-sample	0.96327	0.96576
Out-of-sample	0.96582	0.9652

Therefore, the clustering does not necessarily make the model prediction better. Probably because of the parameter finetuning, or that I have reached the highest F1-score for the given features.

(e): Model Interpretation:

	features	importance
0	source_type_ASSIGN	6668.531866
2	source_type_DELIVERY	637.634091
1	grid_distance	9.972416
3	target_lat	5.954661
6	target_lng	5.828138
9	target_integrated	3.663543
4	urgency	2.335235
8	source_integrated	2.144315
7	source_lng	2.001847
5	source_lat	1.977068
10	source_type_PICKUP	1.470371

(the features' importance)

According to the importance, the most relevant features in predicting the type of action is 'source_type'.

This makes sense because the status for an order is ASSIGN-PICKUP-DELIVERY. If source_type is ASSIGN, then the action_type must be PICKUP; if source_type is PICKUP, then the action_type must be delivery.

What's more, after finetuning, the highest out-of-sample F1-score using XGBoost is 0.96582. When I use clustering, after finetuning, the out-of-sample highest F1-score using XGBoost is 0.9652. I think the reason that clustering features (named as 'target_integrated' and 'source_integrated', see the table above) does not influence the F1-score that much is because. their weights are too small (3.66 and 2.14 respectively), comparing to the importance of 'source_type' and 'grid_distance'.

Regression

(a): Initial Data Pre-processing:

Features selection:

wave_index, courier_wave_start_lng, courier_wave_start_lat, level, speed, max_load, source_tracking_id, source_lng, source_lat, target_lng, target_lat, grid_distance, urgency, hour, source_type, source_type, weather_grade.

Dependent variable:

expected_use_time.

Data processing:

(1): **Data transformation**:

The feature [‘weather_grade] consists of Normal Weather, slightly bad weather, bad weather and very bad weather. They should be processed by the function ‘pd.get_dummies()’.

The feature [‘source_type’] consists of ASSIGN, PICKUP and DELIVERY. It is transformed to 0-1 dummy variables by the function pd.get_dummies().

(2): **splitting data**:

The final ratio of train test data is 509604:25468. Therefore, I do not split the training data using train-test-split with a test_size=0.3. Instead, I split the data into 484136:25468, which approximates the real data more.

(3): **outlier elimination**:

The outlier detection has taken a lower_limit and an upper_limit rules.

The first quantile is Q_1, the third quantile is Q_3. The IQR is Q_3-Q_1.

Lower bound is Q_1-1.5*IQR, upper limit is Q_3+1.5*IQR.

Any value outside the lower bound and upper bound in train data would be defined as an outlier and eliminated.

(b): Baseline Model:

Linear Regression:

The experiment has taken Linear Regression as baseline model.

The data has been properly processed, the mae is 219.8212.

(c): Any Model :

Polynomial:

Using package ‘PolynomialFeatures’, the dataframe X_train, y_train are transformed into X_train_poly and y_train_poly so that the data can be process in the package properly. However, the mae is 219.82296, which does not change a lot.

Tree Regression:

Using DecisionTreeRegressor, setting the ccp_alpha at 0.005 and max_depth at 4, the mae is

196.2749.

Xgboost:

The data should be processed in xgb.DMatrix so that it could run in XGBRegressor.

After finetuning, the five parameters are as follow:

colsample_bynode=0.9,
learning_rate=0.1,
gamma=0.5,
max_depth=15,
n_estimators=17.

The mae is 176.47736.

And the mae in Kaggle is 181.664.

(d): Feature Engineerling:

Using PolynomialFeatures to make the train data into second order model.

After finetuning, the best parameters are: max_depth = 15, n_estimators = 18. learning_rate = 0.1. colsample_bynode=0.9. gamma=1.

The in-sample MAE with polynomial is 176.94887, the out-of-sample MAE with polynomial is 181.72844.

MAE	without polynomial	with polynomial
in-sample	176.47733	176.94887
Out-of-sample	181.66473	181.72844

Therefore, the polynomial does not necessarily make the model prediction better. Probably because of the parameter finetuning, or that I have reached the highest possible MAE within the given features.

(e): Model Interpretation:

	features	importance
0	source_type_ASSIGN	3.153250e+09
19	source_type_PICKUP	6.740365e+06
1	grid_distance	4.530666e+06
2	source_type_DELIVERY	4.330676e+06
3	urgency	1.370349e+06
9	weather_grade_Very Bad Weather	9.970780e+05
17	source_tracking_id	6.235317e+05
16	target_lat	5.576708e+05
14	target_lng	5.176822e+05
12	hour	4.596205e+05
4	source_lat	4.364769e+05
8	source_lng	3.999825e+05
5	max_load	3.671858e+05
7	speed	3.654809e+05
18	weather_grade_Slightly Bad Weather	3.504200e+05
6	courier_wave_start_lat	3.363931e+05
13	weather_grade_Normal Weather	3.308178e+05
10	courier_wave_start_lng	2.772796e+05
15	level	2.761404e+05
20	weather_grade_Bad Weather	2.702527e+05
11	wave_index	2.357736e+05

(the features' importance)

According to the importance, the top several most relevant features in predicting the expected_use_time are **'source_type', 'grid_distance', 'urgency','weather_grade'.**
This makes sense. When source_type is ‘ASSIGN’, the order has two steps to finish, therefore may needs longer time. When grid_distance is large, urgency is high and weather_grade is ‘Very Bad Weather’, the expected_use_time is sure to be very long.

What’s more, after finetuning, the lowest MAE using XGBoost is 181.66473. When I use polynomial, after finetuning, the highest F1-score using XGBoost is 181.72844. Although using polynomial makes the prediction a little bit worse, I think I have reached the limit of XGBoost and using polynomial would only makes the best MAE fluctuate in a small range. Polynomial may improve the prediction in other method, such as Linear Regression.