Jialin Li
Professor Phillips (Renyu) Zhang
Kaggle Competition Report
Due: 2021/4/12

Classification Problem

Before writing the code, I plan the steps for the classification problem:

1. **Data Preparation**
   a) Drop unnecessary data (rough drop, a further drop will be decided by testing on baseline model)
   b) Dummy textual information to binary int
   c) Split the training data frame into "training set" and "testing set"
   d) Standardization (in case I need KNN classification)
2. **Choose model** and **features**
   a) Fit different baseline model for all features
   b) Choose the best Model and find each feature's importance, abandon insignificant features
3. **Feature Engineering** and **Fine Tune** for the best model
   a) Feature engineering
      i. PCA/ Polynomial Features
   b) Fine-tune
      i. Grid Search or Manually fine-tune
   c) Re-determine feature and parameters
4. **Retrain** the model on the whole training data frame using the best model found
5. **Make predictions** on the testing data frame

STEP 1

Dropping Features:

At first, I **drop** all id-related features since they obviously cannot provide important information about the action. Meanwhile, since the "source type" may not be detectable in real-life scenario, I **drop** the 'source_type' information to avoid an overly optimistic prediction.

Dummy Features:

I **dummy** "weather_grade".

Standardize before training:

After **splitting** the data, I **standardize** them using standardscaler.

STEP 2

Decide on the best baseline model:

I **train** 5 baseline models and observe their performances on fscore. (Rank of fscore: BINARY LR < CART < RANDOM FOREST < KNN < XGBOOST). In other words, Xgboosts performs best in simulating the real action determine pattern of the courier. Therefore, I choose to use Xgboost.

```
# # Fit a Baseline Model with all features
# # XgBoost
# import xgboost as xgb
# data_train=xgb.DMatrix(data=X_train_st,label=y_train)
# data_test=xgb.DMatrix(data=X_test_st,label=y_test)

# Action_type_xgbt=xgb.XGBClassifier().fit(X_train_st,y_train)

# y_pred=Action_type_xgbt.predict(X_test_st)
# fbeta=fbeta_score(y_test, y_pred, beta=0.5)
# print('The Fbeta_score for XgBoost is', fbeta)
# print(X_train.columns.values)
# print('The features importance is shown here:',Action_type_xgbt.feature_importances_ )
```

```
In previous trials: The Fbeta_score for XgBoost is 0.8124462981252886

From the Fbeta_score listed above, xgBoost should be the best model. Further fine tune for it.
```

Decide on the best feature combinations:

I print the feature importance on the baseline model and narrow down the feature needed. In this dataset, I choose to abandon those features with importance less than 0.03. This improves the model's fscore by 0.02

```
# Fit a Baseline Model with all features
# XgBoost
import xgboost as xgb
data_train=xgb.DMatrix(data=X_train_st,label=y_train)
data_test=xgb.DMatrix(data=X_test_st,label=y_test)

Action_type_xgbt=xgb.XGBClassifier().fit(X_train_st,y_train)

y_pred=Action_type_xgbt.predict(X_test_st)
fbeta=fbeta_score(y_test, y_pred, beta=0.5)
print('The Fbeta_score for XgBoost is', fbeta)
print(X_train.columns.values)
print('The features importance is shown here:',Action_type_xgbt.feature_importances_ )
```

```
C:\Users\gzLij\AppData\Local\Programs\Python\Python37\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XG
BClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[18:46:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation m
etric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavi
or.
The Fbeta_score for XgBoost is 0.8428487284676511
['courier_wave_start_lng' 'courier_wave_start_lat' 'target_lng'
 'target_lat' 'grid_distance' 'urgency']
The features importance is shown here: [0.08255818 0.0897366  0.06423002 0.05700229 0.42079422 0.2856787 ]
```

STEP 3

Feature Engineering:

Since PCA and Polynomial features didn't give me good results, I don't use feature engineering on this question anymore. (Previously, I tried cluster hour information by dividing them into rush hour and not-busy hour. But it gives a worse result.)

Fine-tune:

Later, with 40+ times manual trials, I finalized **my parameter list**: (gamma=2, max_depth=12, learning_rate=1 , n_estimators=120).

```python
import xgboost as xgb
data_train=xgb.DMatrix(data=X_train_st,label=y_train)
data_test=xgb.DMatrix(data=X_test_st,label=y_test)

clf=xgb.XGBClassifier(gamma=2,max_depth=12,learning_rate=1,n_estimators=120)
poly_pipe=make_pipeline(clf)
Action_type_xgbt=poly_pipe.fit(X_train_st,y_train)
y_pred=Action_type_xgbt.predict(X_test_st)
fbeta=fbeta_score(y_test, y_pred, beta=0.5)
print('The Fbeta_score for XgBoost is', fbeta)
print(X_train.columns.values)
print('The features importance is shown here:',clf.feature_importances_ )
```

```
C:\Users\gzLij\AppData\Local\Programs\Python\Python37\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XG
BClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[18:46:49] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation m
etric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavi
or.
The Fbeta_score for XgBoost is 0.9396499026068216
['courier_wave_start_lng' 'courier_wave_start_lat' 'target_lng'
 'target_lat' 'grid_distance' 'urgency']
The features importance is shown here: [0.08957425 0.0890241  0.12085743 0.1237528  0.31945175 0.25733972]
```

STEP 4 & STEP 5

At last, I retrain the model with the above parameters on the whole dataframe_train and predict the dataframe_test data based using the finalized model.

A	B	C
order	action_type_DELIVERY	
0	0	
1	1	
2	0	
3	1	
4	0	
5	0	
6	1	
7	1	
8	0	
9	1	
10	0	

Regression Problem

Before writing the code, I plan the steps for the regression problem:

1. **Data Preparation**
    a) Drop unnecessary data (rough drop, further drop will be decided by testing on baseline model)
    b) Dummy textual information to binary int
    c) Split the training data frame into "training set" and "testing set"
    d) Standardization
2. **Choose model** and **features**
    a) Fit different baseline model for all features
    b) Choose the best Model and find each feature's importance, abandon insignificant features
3. **Feature Engineering** and **Fine Tune** for the best model
    a) Feature engineering
        i. PCA/ Polynomial Features
    b) Fine-tune
        i. Grid Search or Manually fine-tune
    c) Re-determine features and parameters
4. **Retrain** the model on the whole training data frame using the best model found
5. **Make predictions** on the testing data frame

STEP 1

Dropping features:

At first, I **drop** all id-related features since they obviously cannot provide important information about the expected use time.

Dummy features:

I dummy ['source_type','action_type','weather_grade'].

Standardize before training:

After **splitting** the data, I **standardize** them using standardscaler.

STEP 2

Model Selection:

I train 4 baseline models and observe their performances on the MAE score. (MAE Score Rank: Logistic Regression > Tree Regressor > Random Forest ≈ KNN > Xgboost) Therefore, I choose to use Xgboost.

Feature Selection:

I print the feature importance on the baseline model. When narrowing down the features, the result is not as good as the original one. So I just keep my features to be:
['grid_distance' , 'urgency','hour' , 'expected_use_time','speed','weather_grade_Slightly Bad Weather' , 'weather_grade_Very Bad Weather' , 'source_type_DELIVERY' , 'source_type_PICKUP']

```python
import xgboost as xgb
data_train=xgb.DMatrix(data=X_train_st,label=y_train)
data_test=xgb.DMatrix(data=X_test_st,label=y_test)


xgb_reg=xgb.XGBRegressor(objective='reg:squarederror',colsample_bynode=0.8, gamma=0.001,max_depth=5,n_estimators=20)

xgb_reg.fit(X_train_st,y_train)
y_pred=xgb_reg.predict(X_test_st)
MAE_score=MAE(y_test, y_pred)
print(MAE_score)
print(X_train.columns.values)
print('feature importance: ' ,xgb_reg.feature_importances_)
```

```
201.55847429384917
['wave_index' 'courier_wave_start_lng' 'courier_wave_start_lat' 'group'
 'level' 'speed' 'grid_distance' 'urgency' 'hour' 'source_type_DELIVERY'
 'source_type_PICKUP' 'weather_grade_Normal Weather'
 'weather_grade_Slightly Bad Weather' 'weather_grade_Very Bad Weather']
feature importance:  [0.01206657 0.0053394  0.00683972 0.00314411 0.00309758 0.01142361
 0.20503186 0.09495062 0.01257664 0.47133318 0.15356338 0.00450212
 0.00287434 0.01325687]
```

STEP 3

I manually fine tune for the xgboost and find that the **best parameter combination** is (colsample_bynode=0.8, gamma=0.001,max_depth=6,n_estimators=5).

```python
import xgboost as xgb
data_train=xgb.DMatrix(data=X_train_st,label=y_train)
data_test=xgb.DMatrix(data=X_test_st,label=y_test)

xgb_reg=xgb.XGBRegressor(colsample_bynode=0.8, gamma=0.001,max_depth=6,n_estimators=5)

xgb_reg.fit(X_train_st,y_train)
y_pred=xgb_reg.predict(X_test_st)
MAE_score=MAE(y_test, y_pred)
print(MAE_score)
print(X_train.columns.values)
print(xgb_reg.feature_importances_)
```

```
195.42389335326789
['wave_index' 'courier_wave_start_lng' 'courier_wave_start_lat' 'group'
 'level' 'speed' 'grid_distance' 'urgency' 'hour' 'source_type_DELIVERY'
 'source_type_PICKUP' 'weather_grade_Normal Weather'
 'weather_grade_Slightly Bad Weather' 'weather_grade_Very Bad Weather']
[1.1109776e-02 1.3143742e-03 1.1016952e-03 3.3782487e-04 3.6354913e-04
 5.5715553e-03 2.0762073e-01 9.5536508e-02 6.7298920e-03 4.7606575e-01
 1.8716000e-01 0.0000000e+00 0.0000000e+00 7.0883459e-03]
```

STEP 4 & STEP 5

I **retrain** the model with the above parameters on the whole dataframe_train and **predict** the dataframe_test data based using the finalized model.

order	expected_use_time
0	460.1804
1	342.898
2	396.779
3	350.4547
4	456.3596
5	135.3047
6	450.0731
7	409.1506
8	526.4426
9	303.8695
10	428.1351