baseline

September 16, 2019

[1]:

```
import gc
gc.collect()
```

[1]:

```
62
```

[2]:

```
# Load libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪multilabel_confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

import xgboost as xgb
import lightgbm as lgb
```

[3]:

```
# Load datasets

TRAIN_FILEPATH = 'data/train.csv'
train_df = pd.read_csv( TRAIN_FILEPATH, header=0 )
display(train_df.shape)
```

```
(15120, 56)
```

[4]:

```
# Split training set on train/validation sets (60-20-20, 70-30)

# save target value
train_label = train_df['Cover_Type']  # {1;2;...;6;7}, 2160 entires each
train_df = train_df.drop( ['Id', 'Cover_Type'], axis=1 )

# train_test_split
```

```
VALIDATION_SIZE = 0.3

X_tr, X_val, y_tr, y_val = train_test_split(
    train_df, train_label,
    test_size=VALIDATION_SIZE,
    shuffle=True
)
```

[5]:

```
# 1. sklearn.LogisticRegression, l2 regularization

logreg_l2_model = LogisticRegression(
    C=1.0,
    solver='lbfgs',  # multinomial loss
    penalty='l2',
    max_iter=1000,
    multi_class='multinomial',
    n_jobs=-1
)

logreg_l2_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('logreg_model', logreg_l2_model)
])

%time logreg_l2_pipeline.fit( X_tr, y_tr )
```

```
CPU times: user 54.3 ms, sys: 63.5 ms, total: 118 ms
Wall time: 8.36 s
```

[5]:

```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('logreg_model',
                 LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=1000,
                                    multi_class='multinomial', n_jobs=-1,
                                    penalty='l2', random_state=None,
                                    solver='lbfgs', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
```

[6]:

```
%time logreg_y_val_pred = logreg_l2_pipeline.predict( X_val )
```

```
CPU times: user 16.1 ms, sys: 19.1 ms, total: 35.2 ms
Wall time: 23.7 ms
```

[7]:

```
display( accuracy_score(y_val, logreg_y_val_pred) )

print( classification_report(y_val, logreg_y_val_pred) )

display( multilabel_confusion_matrix(y_val, logreg_y_val_pred) )
```

```
0.7074514991181657
```

```
              precision    recall  f1-score   support

           1       0.66      0.67      0.66       671
           2       0.61      0.53      0.57       667
           3       0.63      0.54      0.58       617
           4       0.81      0.88      0.84       627
           5       0.73      0.80      0.76       681
           6       0.61      0.67      0.64       631
           7       0.87      0.87      0.87       642

    accuracy                           0.71      4536
   macro avg       0.70      0.71      0.70      4536
weighted avg       0.70      0.71      0.70      4536
```

```
array([[[3630,  235],
        [ 222,  449]],

       [[3648,  221],
        [ 315,  352]],

       [[3726,  193],
        [ 283,  334]],

       [[3779,  130],
        [  76,  551]],

       [[3657,  198],
        [ 138,  543]],

       [[3638,  267],
        [ 210,  421]],

       [[3811,   83],
        [  83,  559]]])
```

[8]:

```
# 2. sklearn.LogisticRegression, l1 regularization
```

```
logreg_l1_model = LogisticRegression(
    C=1.0,
    solver='liblinear',  # one-vs-rest scheme
    penalty='l1',
    max_iter=1000,
    multi_class='ovr'  # can't use 'multinomial'
)

logreg_l1_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('logreg_model', logreg_l1_model)
])

%time logreg_l1_pipeline.fit( X_tr, y_tr )
```

```
CPU times: user 20.6 s, sys: 54 ms, total: 20.6 s
Wall time: 20.6 s
```

[8]:

```
Pipeline(memory=None,
         steps=[('scaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('logreg_model',
                 LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=1000,
                                    multi_class='ovr', n_jobs=None,
                                    penalty='l1', random_state=None,
                                    solver='liblinear', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
```

[9]:

```
%time logreg_y_val_pred = logreg_l1_pipeline.predict( X_val )
```

```
CPU times: user 33.6 ms, sys: 19.6 ms, total: 53.2 ms
Wall time: 11 ms
```

[10]:

```
display( accuracy_score(y_val, logreg_y_val_pred) )

print( classification_report(y_val, logreg_y_val_pred) )

display( multilabel_confusion_matrix(y_val, logreg_y_val_pred) )
```

```
0.6783509700176367
```

```
              precision    recall  f1-score   support
```

```
           1       0.64      0.62      0.63       671
           2       0.58      0.49      0.53       667
           3       0.60      0.54      0.57       617
           4       0.80      0.89      0.84       627
           5       0.64      0.72      0.68       681
           6       0.60      0.62      0.61       631
           7       0.87      0.87      0.87       642

    accuracy                           0.68      4536
   macro avg       0.67      0.68      0.68      4536
weighted avg       0.67      0.68      0.67      4536
```

```
array([[[3629,  236],
        [ 252,  419]],

       [[3634,  235],
        [ 341,  326]],

       [[3697,  222],
        [ 284,  333]],

       [[3770,  139],
        [  68,  559]],

       [[3576,  279],
        [ 191,  490]],

       [[3643,  262],
        [ 242,  389]],

       [[3808,   86],
        [  81,  561]]])
```

[27]:

```
# 3. sklearn.RandomForestClassifier

rfc_model = RandomForestClassifier(
    n_estimators=1000,
    n_jobs=-1
)

%time rfc_model.fit( X_tr, y_tr )
```

```
CPU times: user 23.4 s, sys: 380 ms, total: 23.8 s
Wall time: 3.66 s
```

[27]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=-1, oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

[28]:
```
%time rfc_y_val_pred = rfc_model.predict( X_val )
```

```
CPU times: user 1.56 s, sys: 60.2 ms, total: 1.62 s
Wall time: 407 ms
```

[29]:
```
display( accuracy_score(y_val, rfc_y_val_pred) )

print( classification_report(y_val, rfc_y_val_pred) )

display( multilabel_confusion_matrix(y_val, rfc_y_val_pred) )
```

```
0.8619929453262787
```

```
              precision    recall  f1-score   support

           1       0.81      0.75      0.78       671
           2       0.79      0.72      0.75       667
           3       0.87      0.79      0.83       617
           4       0.92      0.97      0.95       627
           5       0.88      0.95      0.91       681
           6       0.83      0.90      0.86       631
           7       0.93      0.96      0.95       642

    accuracy                           0.86      4536
   macro avg       0.86      0.86      0.86      4536
weighted avg       0.86      0.86      0.86      4536
```

```
array([[[3744,  121],
        [ 168,  503]],

       [[3739,  130],
        [ 187,  480]],

       [[3844,   75],
        [ 129,  488]],

       [[3856,   53],
        [  16,  611]],
       [[3767,   88],
        [  37,  644]],

       [[3790,  115],
        [  65,  566]],

       [[3850,   44],
        [  24,  618]]])
```

[14]:

```
display(
    pd.DataFrame({
        'feature_name': X_tr.columns,
        'feature_imp': rfc_model.feature_importances_
    }).sort_values( by='feature_imp', ascending=False ).head()
)
```

```
                        feature_name  feature_imp
0                          Elevation     0.231019
5    Horizontal_Distance_To_Roadways     0.088981
9 Horizontal_Distance_To_Fire_Points     0.070936
3    Horizontal_Distance_To_Hydrology     0.063224
4      Vertical_Distance_To_Hydrology     0.054716
```

[15]:

```
# 4. xgb.XGBClassifier

eval_set = [ (X_val, y_val) ]

xgb_model = xgb.XGBClassifier(
    gamma=0.025,
    learning_rate=0.35,
    max_depth=5,
    n_estimators=1000,
    objective='multi:softmax',
    n_jobs=4
)

%time xgb_model.fit( X_tr, y_tr, eval_set=eval_set, eval_metric='merror',␣
 ↪verbose=False )
```

```
CPU times: user 3min 1s, sys: 495 ms, total: 3min 1s
Wall time: 45.7 s
```

[15]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0.025,
              learning_rate=0.35, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=4,
              nthread=None, objective='multi:softprob', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

[16]:

```
%time xgb_y_val_pred = xgb_model.predict( X_val )
```

```
CPU times: user 1.33 s, sys: 4.03 ms, total: 1.33 s
Wall time: 335 ms
```

[17]:

```
display( accuracy_score(y_val, xgb_y_val_pred) )

print( classification_report(y_val, xgb_y_val_pred) )

display( multilabel_confusion_matrix(y_val, xgb_y_val_pred) )
```

```
0.8538359788359788
```

```
              precision    recall  f1-score   support

           1       0.77      0.71      0.74       671
           2       0.73      0.69      0.71       667
           3       0.85      0.83      0.84       617
           4       0.95      0.96      0.96       627
           5       0.89      0.94      0.92       681
           6       0.84      0.89      0.86       631
           7       0.93      0.96      0.95       642

    accuracy                           0.85      4536
   macro avg       0.85      0.86      0.85      4536
weighted avg       0.85      0.85      0.85      4536
```

```
array([[[3721,  144],
        [ 194,  477]],

       [[3700,  169],
        [ 208,  459]],

       [[3831,   88],
        [ 107,  510]],

       [[3880,   29],
        [  22,  605]],

       [[3776,   79],
        [  38,  643]],

       [[3795,  110],
        [  70,  561]],

       [[3850,   44],
        [  24,  618]]])
```

[18]:
```
display(
    pd.DataFrame({
        'feature_name': X_tr.columns,
        'feature_imp': xgb_model.feature_importances_
    }).sort_values( by='feature_imp', ascending=False ).head()
)
```

```
       feature_name  feature_imp
13  Wilderness_Area4     0.081424
43       Soil_Type30     0.076421
25       Soil_Type12     0.065155
16        Soil_Type3     0.062327
0          Elevation     0.059893
```

[19]:
```
# 5. lgb.LGBMClassifier

lgb_model = lgb.LGBMClassifier(
    learning_rate=0.2,
    max_depth=-1,
    n_estimators=1000,
    objective='multiclass',
    n_jobs=8,
    verbose=0
)

%time lgb_model.fit( X_tr, y_tr )
```

```
CPU times: user 1min 27s, sys: 607 ms, total: 1min 27s
Wall time: 11.7 s
```

[19]:
```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.2, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=8, num_leaves=31,
               objective='multiclass', random_state=None, reg_alpha=0.0,
               reg_lambda=0.0, silent=True, subsample=1.0,
               subsample_for_bin=200000, subsample_freq=0, verbose=0)
```

[20]:

```
%time lgb_y_val_pred = lgb_model.predict( X_val )
```

```
CPU times: user 4.75 s, sys: 9 ţs, total: 4.75 s
Wall time: 625 ms
```

[21]:

```
display( accuracy_score(y_val, lgb_y_val_pred) )

print( classification_report(y_val, lgb_y_val_pred) )

display( multilabel_confusion_matrix(y_val, lgb_y_val_pred) )
```

```
0.8699294532627866
```

```
              precision    recall  f1-score   support

           1       0.79      0.74      0.77       671
           2       0.77      0.71      0.74       667
           3       0.87      0.84      0.86       617
           4       0.96      0.98      0.97       627
           5       0.89      0.96      0.92       681
           6       0.85      0.91      0.88       631
           7       0.94      0.97      0.95       642

    accuracy                           0.87      4536
   macro avg       0.87      0.87      0.87      4536
weighted avg       0.87      0.87      0.87      4536
```

```
array([[[3735,  130],
        [ 173,  498]],

       [[3725,  144],
        [ 191,  476]],

       [[3845,   74],
        [ 101,  516]],

       [[3883,   26],
        [  15,  612]],

       [[3777,   78],
        [  29,  652]],

       [[3804,  101],
        [  59,  572]],
       [[3857,   37],
        [  22,  620]]])
```

[22]:
```
display(
    pd.DataFrame({
        'feature_name': X_tr.columns,
        'feature_imp': lgb_model.feature_importances_
    }).sort_values( by='feature_imp', ascending=False ).head()
)
```

```
                       feature_name  feature_imp
5     Horizontal_Distance_To_Roadways        23481
9  Horizontal_Distance_To_Fire_Points        23437
0                           Elevation        22266
4      Vertical_Distance_To_Hydrology        15688
3    Horizontal_Distance_To_Hydrology        14445
```

[23]:
```
###################################################################################
```

[33]:
```
# Blend baseline LGBclf, XGBclf and sklearn RandomForestClassifier models into␣
 ↪submission

# Load test set
TEST_FILEPATH = 'data/test.csv'
test_df = pd.read_csv( TEST_FILEPATH, header=0 )
display(test_df.shape)

# Save 'Id' for submission
test_ids = test_df['Id']
test_df = test_df.drop( ['Id'], axis=1 )
```

```
(565892, 55)
```

[41]:
```
print('random forest classifier...')
rfc_model = RandomForestClassifier(
    n_estimators=1000,
    n_jobs=-1
)
rfc_model.fit( train_df, train_label )
# rfc_pred = rfc_model.predict( test_df )

print('xgboost classifier...')
xgb_model = xgb.XGBClassifier(
    gamma=0.025,
    learning_rate=0.35,
    max_depth=9,
    n_estimators=1000,
    objective='multi:softmax',
    n_jobs=4
)
xgb_model.fit( train_df, train_label )
# xgb_pred = xgb_model.predict( test_df )

print('lgbm classifier...')
lgb_model = lgb.LGBMClassifier(
    learning_rate=0.2,
    max_depth=-1,
    n_estimators=1000,
    objective='multiclass',
    n_jobs=8,
    verbose=0
)
lgb_model.fit( train_df, train_label )
# lgb_pred = lgb_model.predict( test_df )
```

[45]:

```
rfc_pred_proba = rfc_model.predict_proba( test_df )
xgb_pred_proba = xgb_model.predict_proba( test_df )
lgb_pred_proba = lgb_model.predict_proba( test_df )
```

[46]:

```
blended_pred_proba = ( rfc_pred_proba + xgb_pred_proba + lgb_pred_proba ) / 3.0
```

[78]:

```
# display(blended_pred_proba.shape, blended_pred_proba)

def get_argmax( x_array ):
    return np.argmax( x_array )
get_argmax_vect = np.vectorize(get_argmax)

y_pred_max_proba = np.array( [get_argmax(x_row)+1 for x_row in␣
 ↪blended_pred_proba] )
```

[84]:

```
submission = pd.DataFrame(
    y_pred_max_proba, index=test_ids, columns=['Cover_Type']
)
submission.to_csv('submission_baseline_rfc_lgb_xgb.csv', index_label='Id')
```

[ ]:

```
# Kaggle public leaderboard score (score - classification accuracy): 0.77121
```