Mariners Machine Learning Model

February 17, 2020

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report,␣
↪precision_recall_curve, auc, matthews_corrcoef
from sklearn.pipeline import Pipeline

import xgboost as xgb
from xgboost import XGBClassifier

from joblib import dump
```

[2]:
```
train_df = pd.read_csv('../Data/model_data.csv')
```

This notebook is dedicated to tuning the hyper-parameters of the XGBoost model for use in the main Challenge notebook. Here we will tune 5 hyper-parameters: n_estimators, max_depth, gamma, learning rate, and colsample_bytree to get a better model than just a base XGBoost Classifier. I chose these parameters in specific, because in doing some research, these were the common parameters tuned for an imbalanced classification problem.

[3]:
```
X = train_df.drop(['pitcher_id', 'batter_id', 'stadium_id', 'umpire_id',
                   'catcher_id', 'pitch_call', 'is_strike', 'pitch_id'], axis=1)
y = train_df['is_strike']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
↪random_state=34)

X_train = pd.get_dummies(X_train, prefix=['pitcher', 'batter', 'is'],␣
↪columns=['pitcher_side', 'batter_side', 'pitch_type'])
X_test = pd.get_dummies(X_test, prefix=['pitcher', 'batter', 'is'],␣
↪columns=['pitcher_side', 'batter_side', 'pitch_type'])
```

[8]:
```
steps = [('xgb', XGBClassifier(seed=34))]
param_grid = {'xgb__n_estimators': np.arange(100, 1100, 100)}
pipeline = Pipeline(steps)
cv_1 = GridSearchCV(pipeline, param_grid, cv=3)
cv_1.fit(X_train, y_train)
print(cv_1.best_params_, cv_1.best_score_)
n_estimators = cv_1.best_params_['xgb__n_estimators']
```

```
{'xgb__n_estimators': 900} 0.8690059652694787
```

[10]:
```
steps = [('xgb', XGBClassifier(n_estimators=n_estimators, seed=34))]
param_grid = {'xgb__max_depth': np.arange(3,10,2)}
pipeline = Pipeline(steps)
cv_2 = GridSearchCV(pipeline, param_grid, cv=3)
cv_2.fit(X_train, y_train)
print(cv_2.best_params_, cv_2.best_score_)
max_depth = cv_2.best_params_['xgb__max_depth']
```

```
{'xgb__max_depth': 5} 0.8697434097759663
```

[11]:
```
steps = [('xgb', XGBClassifier(n_estimators=n_estimators, max_depth=max_depth,␣
 ↪seed=34))]
param_grid = {'xgb__gamma': [0, .1, .25, .5, 1]}
pipeline = Pipeline(steps)
cv_3 = GridSearchCV(pipeline, param_grid, cv=3)
cv_3.fit(X_train, y_train)
print(cv_3.best_params_, cv_3.best_score_)
gamma = cv_3.best_params_['xgb__gamma']
```

```
{'xgb__gamma': 0.5} 0.8699524670372843
```

[12]:
```
steps = [('xgb', XGBClassifier(n_estimators=n_estimators, max_depth=max_depth,␣
 ↪gamma=gamma, seed=34))]
param_grid = {'xgb__learning_rate': np.arange(0.05, 0.35, .05)}
pipeline = Pipeline(steps)
cv_4 = GridSearchCV(pipeline, param_grid, cv=3)
cv_4.fit(X_train, y_train)
print(cv_4.best_params_, cv_4.best_score_)
learning_rate = cv_4.best_params_['xgb__learning_rate']
```

```
{'xgb__learning_rate': 0.1} 0.8699524670372843
```

[19]:
```
steps = [('xgb', XGBClassifier(n_estimators=n_estimators, max_depth=max_depth,␣
 ↪gamma=gamma, seed=34))]
param_grid = {'xgb__colsample_bytree': np.arange(0.5, 1.1, .1)}
pipeline = Pipeline(steps)
cv_5 = GridSearchCV(pipeline, param_grid, cv=3)
```

```
cv_5.fit(X_train, y_train)
print(cv_5.best_params_, cv_5.best_score_)
colsample_bytree = cv_5.best_params_['xgb__colsample_bytree']
```

```
C:\Users\David C. Buehler\Anaconda3\lib\site-
packages\sklearn\model_selection\_validation.py:536: FitFailedWarning: Estimator
fit failed. The score on this train-test partition for these parameters will be
set to nan. Details:
xgboost.core.XGBoostError: value 1.1 for Parameter colsample_bytree exceed bound
[0,1]

  FitFailedWarning)
```

```
{'xgb__colsample_bytree': 0.7999999999999999} 0.8700880106227418
```

[26]:
```
steps = [('xgb', XGBClassifier(max_depth=max_depth, gamma=gamma,␣
 ↪learning_rate=learning_rate,
                                colsample_bytree=colsample_bytree, seed=34))]
param_grid = {'xgb__n_estimators': np.arange(100, 1100, 100)}
pipeline = Pipeline(steps)
cv_final = GridSearchCV(pipeline, param_grid, cv=3)
cv_final.fit(X_train, y_train)
print(cv_final.best_params_, cv_final.best_score_)
n_estimators = cv_final.best_params_['xgb__n_estimators']
```

```
{'xgb__n_estimators': 800} 0.8701109839502493
```

[30]:
```
xgb = XGBClassifier(n_estimators=n_estimators, max_depth=max_depth,
                    gamma=gamma, learning_rate=learning_rate,␣
 ↪colsample_bytree=colsample_bytree, seed=34)
xgb.fit(X_train, y_train)
```

[30]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.7999999999999999,
              gamma=0.5, learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=800, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=34,
              silent=None, subsample=1, verbosity=1)
```

Now that we have our model with its optimal hyperparameters and fitted, we'll save this model for use in the other notebook to look at metrics on it, as well as make predictions on the testing set.

[31]:
```
model_filename = 'xgboost_model.pkl'
dump(xgb, model_filename)
```

[31]:
```
['xgboost_model.pkl']
```