Predicting 2019 OBP

Project Walk-Through

Introduction

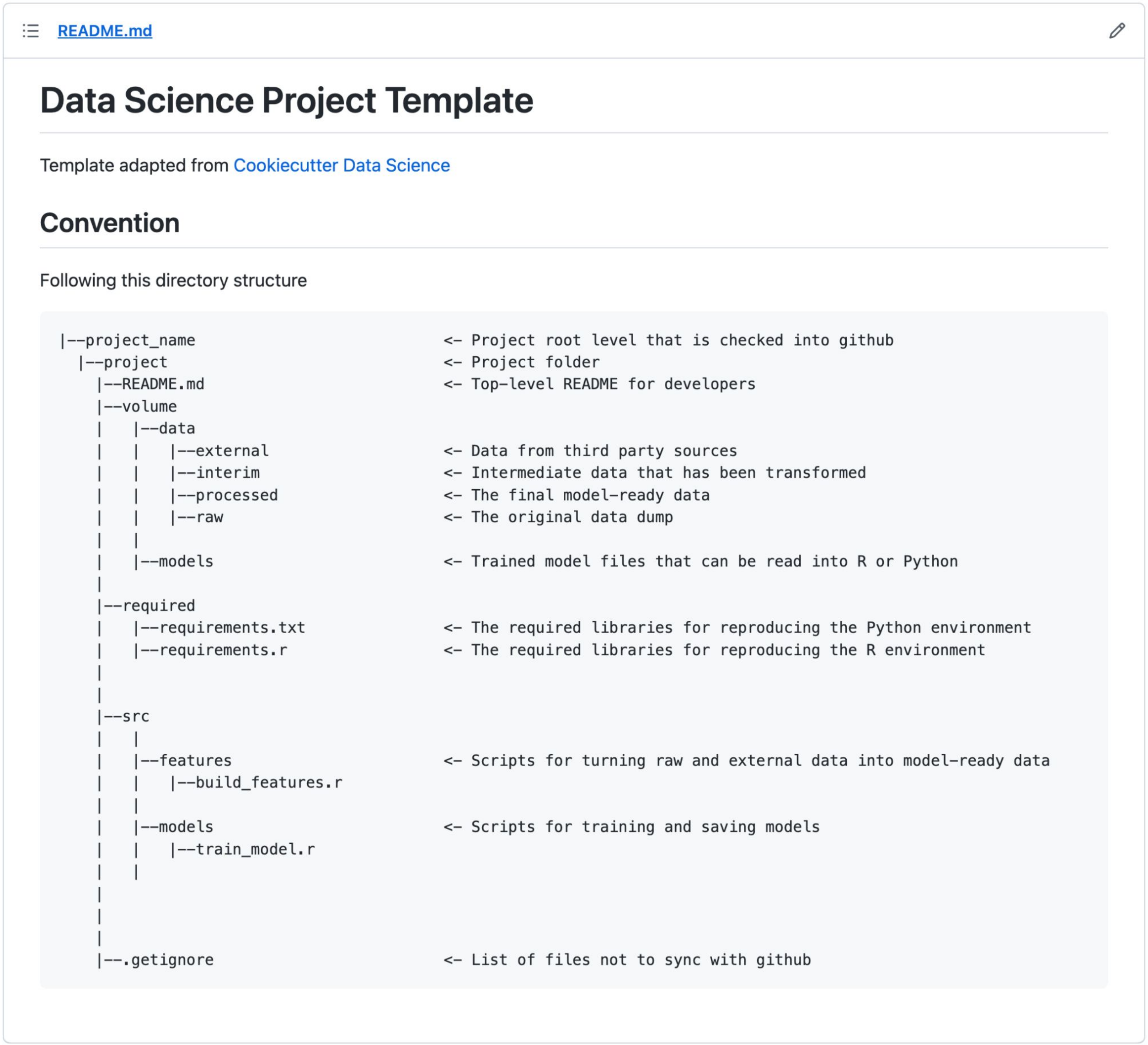
- The objective is to predict each player's on-base percentage at the end of the 2019 season given batting statistics in March/April 2019.
- I will work through two pipelines—One which uses only the features provided and one which uses additional batting statistics from the 2018 season.
- This is a regression problem that can be solved using various machine learning algorithms. The ones that I will use for this project are: (1) Decision Tree, (2) Random Forest, (3) Gradient Boosted Trees, (4) Penalized Linear Regression, (5) Support Vector Machine and (6) K-Nearest Neighbors.

Introduction

- For each modeling script, the basic methodology is the same: (1) Create a model specification, (2) perform necessary data preprocessing, (3) tune hyperparameters using a tuning grid, (4) evaluate the performance of each hyperparameter combination using 10-fold cross-validation.
- Performance metrics (I will be focusing on root mean square error) on the 10 folds should provide a good estimate as to how the model will perform on the test set/unseen data.
- All models in both pipelines will be trained on the same training set, evaluated using the same cross-validation folds and tested on the same test set

Introduction

- All required R packages can be installed at './project/required/requirements.R'
- All files ending in '1' refer to the first pipeline and files ending in '2' refer to the second pipeline.
- More information about the structure of the submitted project ('obp_project.zip') can be found on the next slide.
- **When running code, make sure working directory is set at the repo level folder.**



README.md

Data Science Project Template

Template adapted from Cookiecutter Data Science

Convention

Following this directory structure

```
|--project_name                           <- Project root level that is checked into github
  |--project                              <- Project folder
    |--README.md                          <- Top-level README for developers
    |--volume
    |   |--data
    |   |   |--external                   <- Data from third party sources
    |   |   |--interim                    <- Intermediate data that has been transformed
    |   |   |--processed                  <- The final model-ready data
    |   |   |--raw                        <- The original data dump
    |   |
    |   |--models                         <- Trained model files that can be read into R or Python
    |
    |--required
    |   |--requirements.txt               <- The required libraries for reproducing the Python environment
    |   |--requirements.r                 <- The required libraries for reproducing the R environment
    |
    |
    |--src
    |   |
    |   |--features                       <- Scripts for turning raw and external data into model-ready data
    |   |   |--build_features.r
    |   |
    |   |--models                         <- Scripts for training and saving models
    |   |   |--train_model.r
    |   |
    |
    |
    |
    |--.getignore                         <- List of files not to sync with github
```

Feature Script (Pipeline 1)

- ‘./project/src/features/obp_feature_1.Rmd’
- The provided dataset includes 320 observations and 28 features that can be used to predict the target variable ‘FullSeason_OBP’.
- Wrangling must be performed to convert the data into a tidy, model-ready format.
- ‘%’ is removed from features such as ‘MarApr_BB.’ and ‘MarApr_K.’
- Variables are converted from characters to real numbers where needed.

Feature Script (Pipeline 1)

- ‘./project/src/features/obp_feature_1.Rmd’
- Once the data is in tidy format, we can perform the initial train and test split.
- 80% of the data (256 obs.) will be used to train the model and the remaining 20% (64 obs.) will be used to provide a final, unbiased check of the model’s performance on previously unseen data.
- Train and test sets are sent to ‘./project/volume/data/interim’ for further use.

The Null Model

- ‘./project/src/models/obp_null_model.Rmd’
- Before any candidate models can be evaluated, the performance of a null model must first be established.
- The average ‘FullSeason_OBP’ within the training set is 0.3245. This value will be the prediction for every observation in the test set, simulating what randomly guessing 2019 OBP would be like.
- This results in an rmse of about 0.035. This will be used to compare to all candidate models.

Decision Tree (Pipeline 1)

- ‘./project/src/models/obp_dt_model_1.Rmd’
- Hyperparameters to tune: cost_complexity, tree_depth, min_n
- Best performing hyperparameters: cost_complexity = 1e-10, tree_depth = 4, min_n = 11
- Performance metric: rmse = 0.03067357

Random Forest (Pipeline 1)

- './project/src/models/obp_rf_model_1.Rmd'
- Hyperparameters to tune: mtry, trees, min_n
- Best performing hyperparameters: mtry = 11, trees = 223, min_n = 2
- Performance metric: rmse = 0.02838279

Gradient Boosted Trees (Pipeline 1)

- ‘./project/src/models/obp_gbt_model_1.Rmd’
- Hyperparameters to tune: mtry, trees, min_n, tree_depth, learn_rate, loss_reduction, sample_size
- Best performing hyperparameters: mtry = 19, trees = 1555, min_n = 2, tree_depth = 4, learn_rate = 0.3, loss_reduction = 0.0, sample_size = 0.95
- Performance metric: rmse = 0.02910991

Penalized Linear Regression (Pipeline 1)

- ‘./project/src/models/obp_lr_model_1.Rmd’
- Hyperparameters to tune: penalty, mixture
- Best performing hyperparameters: penalty = 0.002335721, mixture = 0.5263158
- Performance metric: rmse = 0.02849681

Support Vector Machine (Pipeline 1)

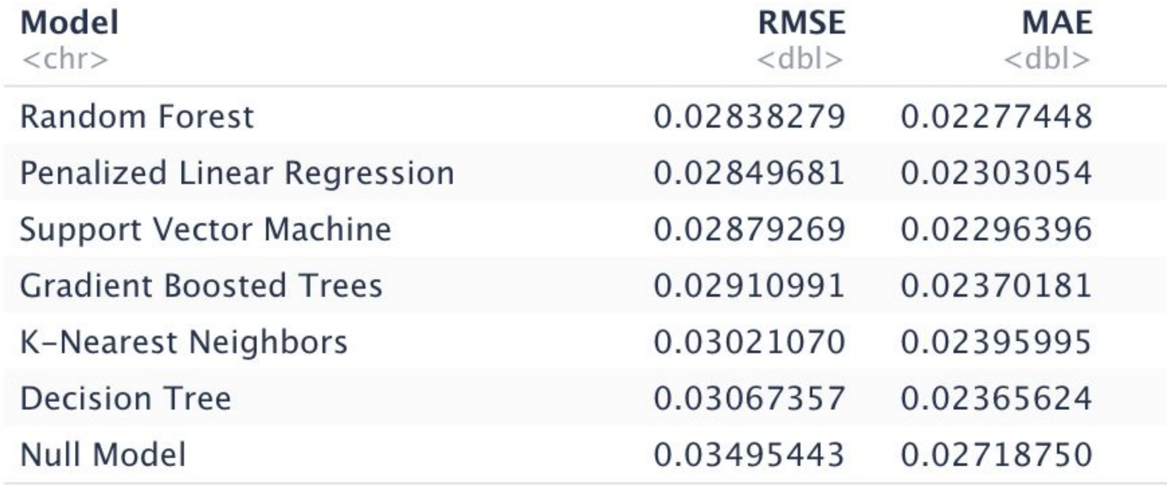
- ‘./project/src/models/obp_svm_model_1.Rmd’
- Hyperparameters to tune: cost, rbf_sigma
- Best performing hyperparameters: cost = 1.0, rbf_sigma = 0.005994843
- Performance metric: rmse = 0.02879269

K-Nearest Neighbors (Pipeline 1)

- ‘./project/src/models/obp_knn_model_1.Rmd’
- Hyperparameters to tune: neighbors, weight_func, dist_power
- Best performing hyperparameters: neighbors = 10, weight_func = rectangular, dist_power = 1.333333
- Performance metric: rmse = 0.03021070

Final Model Selection (Pipeline 1)

- ‘./project/src/models/obp_FINAL_model_1.Rmd’
- The validation error of each model was saved to compare. This metric provides of an estimate of how the model with perform in production.



Model <chr>	RMSE <dbl>	MAE <dbl>
Random Forest	0.02838279	0.02277448
Penalized Linear Regression	0.02849681	0.02303054
Support Vector Machine	0.02879269	0.02296396
Gradient Boosted Trees	0.02910991	0.02370181
K-Nearest Neighbors	0.03021070	0.02395995
Decision Tree	0.03067357	0.02365624
Null Model	0.03495443	0.02718750

- All models provided an improvement over the null model.

Final Model Results (Pipeline 1)

- ‘./project/src/models/obp_FINAL_model_1.Rmd’
- The most predictive model for 2019 OBP using all the provided features was the Random Forest model with mtry = 11, trees = 223 and min_n = 2.
- Using this model configuration, I trained the model on the entire training set and then created predictions for the test set.
- Test error = 0.02955591
- This was not too far off from the error estimated in cross-validation and much better than the null performance of 0.035.

Final Model Results (Pipeline 1)

- ‘./project/src/models/obp_FINAL_model_1.Rmd’
- The objective was to create a prediction for all players in the original dataset, so this model will now be applied to all of the observations.
- The resulting metric should be interpreted carefully as many of the observations being tested on were also included in the model training.
- Null error on all players: rmse = 0.03757081
- Test error on all players: rmse = 0.01638588
- This model provides a significant improvement in predicting 2019 OBP over the null model.

Feature Script (Pipeline 2)

- ‘./project/src/features/obp_feature_2.Rmd’
- To improve our ability to predict 2019 OBP, I expanded the feature pool to include batting statistics from the 2018 season
- The statistics were pulled from fangraphs.com and a glossary of these measurements can be found at https://library.fangraphs.com/offense/
- The added features include the same ones from March-April 2019, wOBA, wRC+, expected statistics (xBA, xSLG, xwOBA) and batted ball data (EV, LA, Barrel%, maxEV, HardHit%)
- The same wrangling from the feature script for the first pipeline is repeated

Feature Script (Pipeline 2)

- ‘./project/src/features/obp_feature_2.Rmd’
- With the introduction of 2018 statistics came certain observations with missing data for those columns.
- Some methods, including Random Forest and Penalized Linear Regression, cannot handle missing values.
- To remedy this problem, I replaced missing values with the average value for that column in the preprocessing stage.

Decision Tree (Pipeline 2)

- ‘./project/src/models/obp_dt_model_2.Rmd’
- Hyperparameters to tune: cost_complexity, tree_depth, min_n
- Best performing hyperparameters: cost_complexity = 1e-10, tree_depth = 4, min_n = 30
- Performance metric: rmse = 0.03138508

Random Forest (Pipeline 2)

- ‘./project/src/models/obp_rf_model_2.Rmd’
- Hyperparameters to tune: mtry, trees, min_n
- Best performing hyperparameters: mtry = 33, trees = 223, min_n = 6
- Performance metric: rmse = 0.02727044

Gradient Boosted Trees (Pipeline 2)

- ‘./project/src/models/obp_gbt_model_2.Rmd’
- Hyperparameters to tune: mtry, trees, min_n, tree_depth, learn_rate, loss_reduction, sample_size
- Best performing hyperparameters: mtry = 33, trees = 223, min_n = 2, tree_depth = 4, learn_rate = 0.3, loss_reduction = 0.0, sample_size = 0.95
- Performance metric: rmse = 0.02737531

Penalized Linear Regression (Pipeline 2)

- ‘./project/src/models/obp_lr_model_2.Rmd’
- Hyperparameters to tune: penalty, mixture
- Best performing hyperparameters: penalty = 0.002335721, mixture = 0.5263158
- Performance metric: rmse = 0.02736887

Support Vector Machine (Pipeline 2)

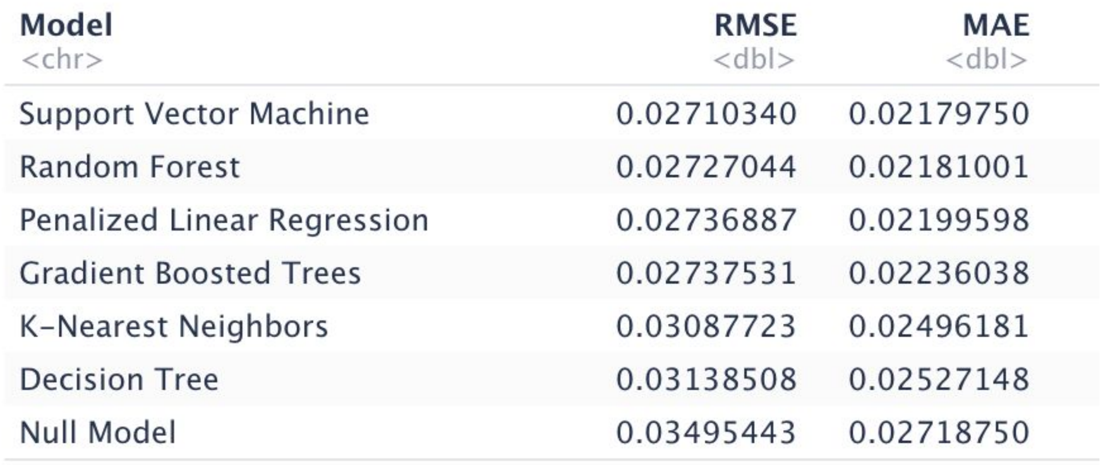
- ‘./project/src/models/obp_svm_model_2.Rmd’
- Hyperparameters to tune: cost, rbf_sigma
- Best performing hyperparameters: cost = 10.07937, rbf_sigma = 0.0004641589
- Performance metric: rmse = 0.0271034

K-Nearest Neighbors (Pipeline 2)

- ‘./project/src/models/obp_knn_model_2.Rmd’
- Hyperparameters to tune: neighbors, weight_func, dist_power
- Best performing hyperparameters: neighbors = 10, weight_func = gaussian, dist_power = 1.555556
- Performance metric: rmse = 0.03087723

Final Model Selection (Pipeline 2)

- ‘./project/src/models/obp_FINAL_model_2.Rmd’
- The validation error of each model was saved to compare. This metric provides of an estimate of how the model with perform in production.



Model <chr>	RMSE <dbl>	MAE <dbl>
Support Vector Machine	0.02710340	0.02179750
Random Forest	0.02727044	0.02181001
Penalized Linear Regression	0.02736887	0.02199598
Gradient Boosted Trees	0.02737531	0.02236038
K-Nearest Neighbors	0.03087723	0.02496181
Decision Tree	0.03138508	0.02527148
Null Model	0.03495443	0.02718750

- Again, all models provided an improvement over the null model.

Final Model Selection (Pipeline 2)

- ‘./project/src/models/obp_FINAL_model_2.Rmd’
- The most predictive model for 2019 OBP using all the provided features was the Support Vector Machine model with cost = 10.07937 and rbf_sigma = 0.0004641589.
- Using this model configuration, I trained the model on the entire training set and then created predictions for the test set.
- Test error = 0.0280255
- This is very close to the error estimated in cross-validation and much better than the null performance of 0.035.
- The second pipeline with the additional features outperformed the first pipeline

Final Model Results (Pipeline 2)

- ‘./project/src/models/obp_FINAL_model_2.Rmd’
- The objective was to create a prediction for all players in the original dataset, so this model will now be applied to all of the observations.
- The resulting metric should be interpreted carefully as many of the observations being tested on were also included in the model training.
- Null error on all players: rmse = 0.03757081
- Test error on all players: rmse = 0.0251984
- This model provides a significant improvement in predicting 2019 OBP over the null model.

Conclusion

- Tasked with predicting 2019 OBP, I would use the second pipeline as it achieved the best cross-validation results, surpassing both the first pipeline and the null model.
- I would be more confident in the cross-validation results as a measure of model performance than the test set results because the test set is rather small. In 10 folds, all observations in the train set are included in testing at some point (reduces variance)
- This model could be further improved upon by including more observations, including more features or by stacking models.