PUBG Finish Placement Prediction

Peng Liu, Xuxu Pan, Min Che

What is PUBG

PlayerUnknown's BattleGrounds (PUBG) :

- A Battle Royale-style video game
- 100 players are dropped onto an island empty-handed
- Explore and eliminate other players until only one is left standing
- The play zone continues to shrink until the last minute of the game

Outlines

- Ask
- Acquire
- Process
- Model
- Deliver
- Conclusion
- Future works

Ask

Whose Chicken Dinner?

The best killer?

The best hide-and-seeker?

Can we predict the finish placement of the PUBG players?

Acquire

Data source: https://www.kaggle.com/c/pubg-finish-placement-prediction/data

Target: winPlacePerc, player's percentile in one game (float, 0 to 1).

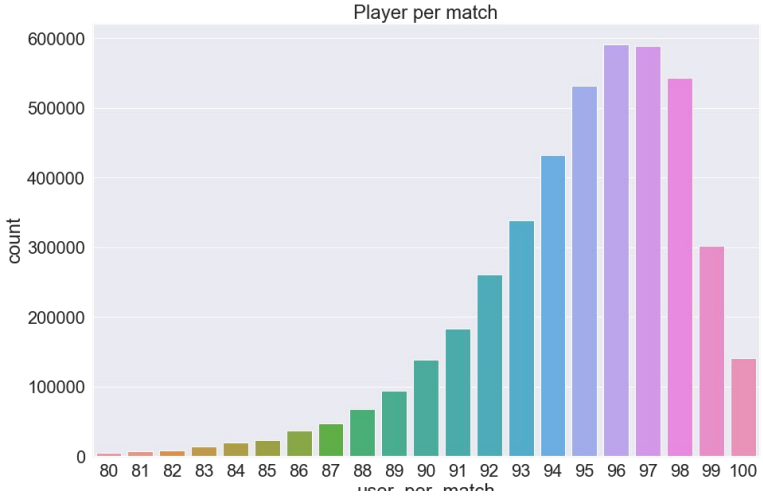
Number of observations: >4 M, each row contains a player's post-game stats.

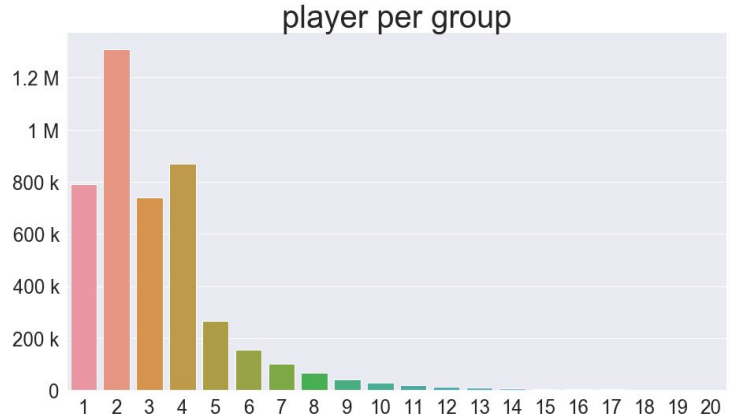
Number of raw features: 28, including player's stats such as assists, boosts, damage, kills, player per match, player per group, walk distance...

Process

1. **Feature Engineering**
   - Add :
     - **player per match**
     - **player per group**
   - Get rid of anormal matches.
     - player per match < 80
     - player per group > 4
   - Down sampling: 4M to 165K
   - Drop highly correlated features.
   - 24 numerical features.

2. **Train-validation-test split (0.49+0.21+0.3)**
   - Train(0.49) +validation(0.21)+ test(0.3)





Model

- LinearRegressor
  - Normalized
- DecisionTreeRegressor
  - RandomizedSearchCV hyperparameter tuning
- RandomForestRegressor
  - RandomizedSearchCV hyperparameter tuning

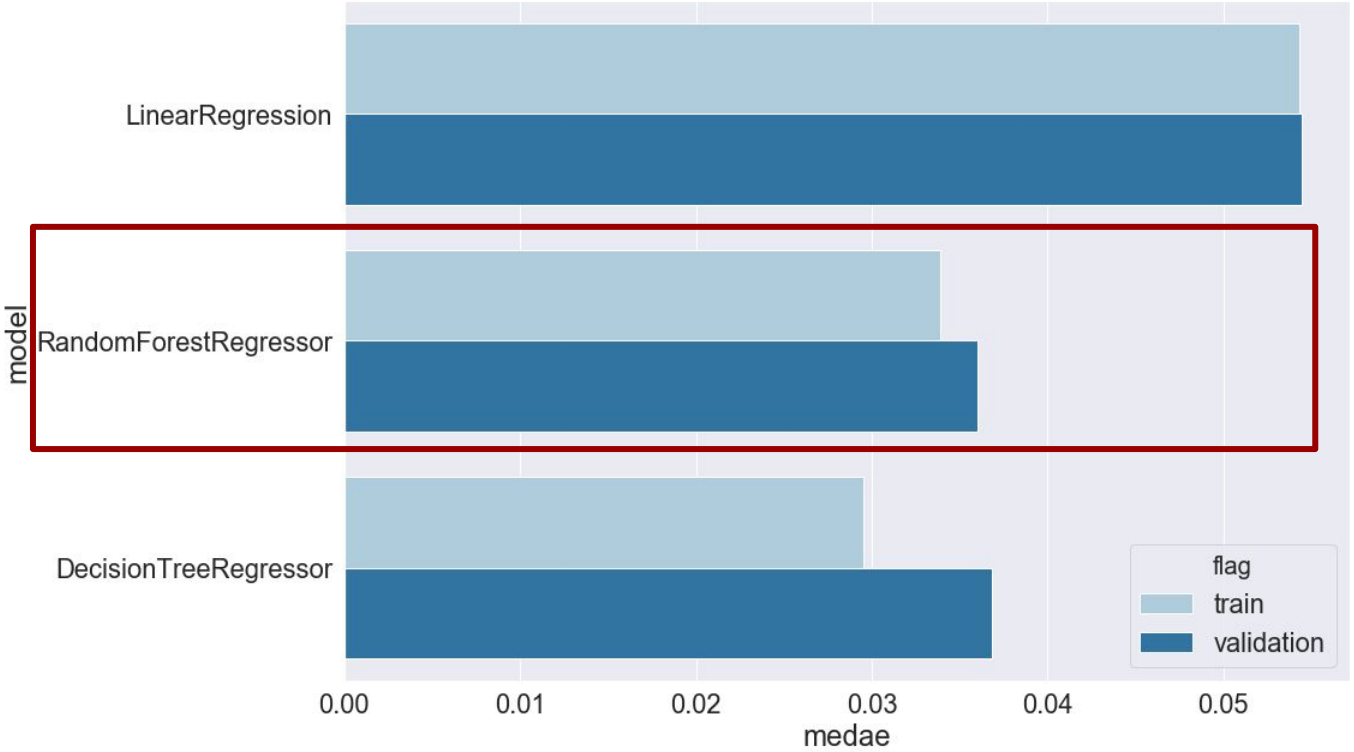
```
def make_pipelines():
    pipe_lr = Pipeline([('scl', StandardScaler()),
                        ('lr', LinearRegression())])
    pipe_rf = Pipeline([('rf', RandomForestRegressor(**para_rf))])
    pipe_dt = Pipeline([('dt', DecisionTreeRegressor(**para_dt))])

    pipelines = [pipe_lr, pipe_rf, pipe_dt]

    return pipelines
```

```
cv=5
n_iter=20

def make_random_cv_dt():

    param_dist = {"max_features":["auto","sqrt","log2", None],
                  "min_samples_split": np.arange(start=2,stop=20,step=2),
                  "min_samples_leaf": np.arange(start=1,stop=50,step=5),
                  "max_depth": [3, 4, 5, 6, 7, 8, 9,10,None]}

    clf_random_cv = RandomizedSearchCV(DecisionTreeRegressor(random_state=42),
                                       param_distributions=param_dist,
                                       cv=cv,
                                       n_iter=n_iter,
                                       verbose=1)
    return clf_random_cv
```

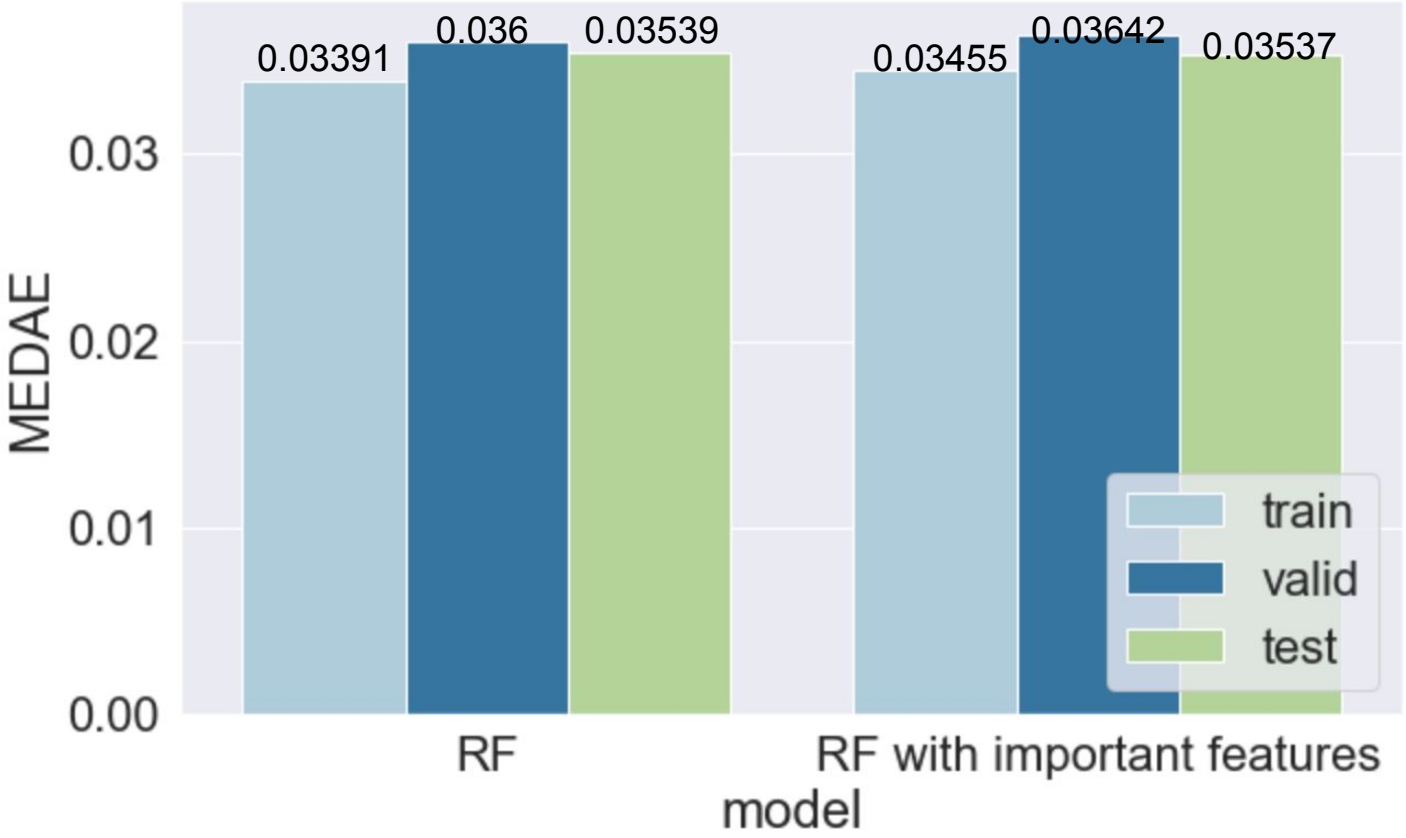
```
cv=5
n_iter=20

def make_random_cv_rf():

    param_dist = {"max_features":["auto","sqrt","log2", None],
                  "n_estimators":np.arange(start=50,stop=300,step=10),
                  "min_samples_split": np.arange(start=2,stop=20,step=2),
                  "min_samples_leaf": np.arange(start=1,stop=50,step=5),
                  "max_depth": [3, 4, 5, 6, 7, 8, 9,10,None],
                  "bootstrap": [True, False]
                 }
    clf_random_cv = RandomizedSearchCV(RandomForestRegressor(random_state=42),
                                       param_distributions=param_dist,
                                       cv=cv,
                                       n_iter=n_iter,
                                       verbose=1)
    return clf_random_cv
```

Model

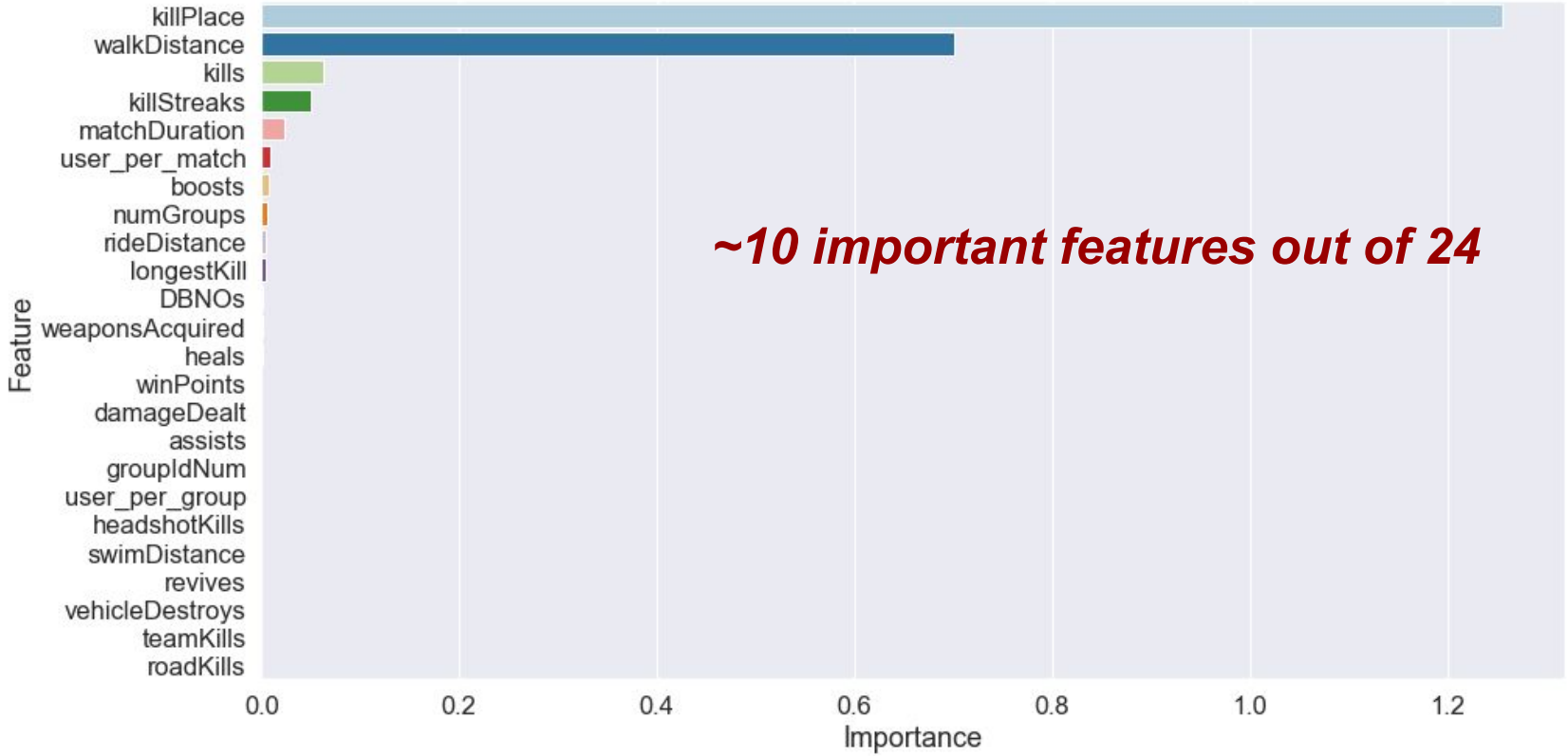
North star metrics: Median Absolute Error



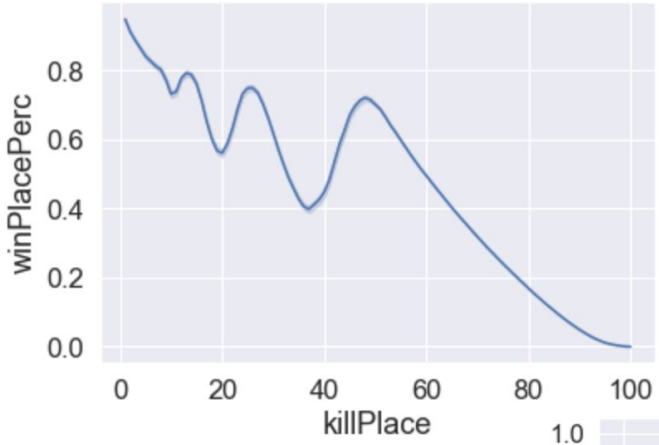
Deliver: Test set error of RF

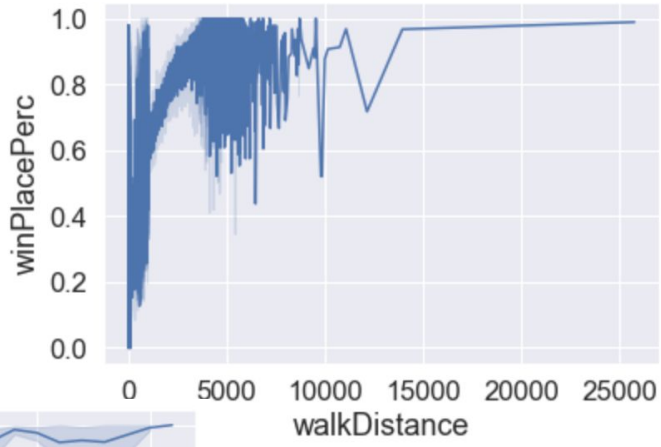


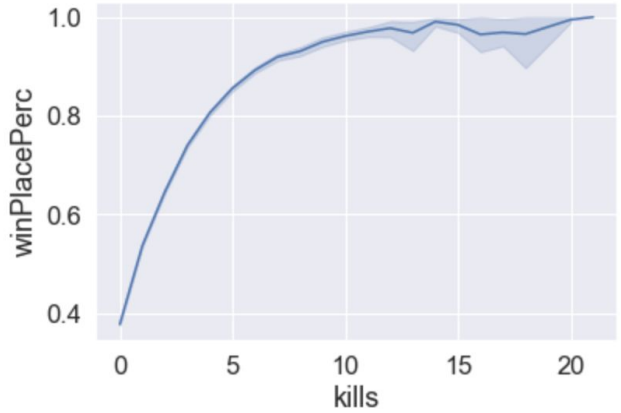
Deliver: Feature importance of RF



Deliver: Top 3 important features







Conclusions

- **Technical key takeaways:**
  - Random forest improves generality.
  - Feature engineering matters.

- **Non-technical key takeaways:**
  - The best killer wins!
  - Walk around and explore more.
  - Collect sufficient boosters boost yourself promptly.
  - Kill from afar.

Future works:

1. **Can we cluster players based on their strategies and behaviors?**

   i.e. killer or hide-and-seeker?

2. **If we label players by clustering, does it help to improve the regressor?**

Thanks and Q&A