PUBG Finish Placement Prediction

Mid-project review deck

Content

Highlights

Review progress

Demo/analysis

Lessons learned

Recommendations

Highlights

Interesting Insight from EDA - A tale of two strategies

Resources (e.g. weapons and other items) are not uniformly distributed throughout PUBG game map. The two areas with the most high-quality items are the **PRISON** and the **SCHOOL - These areas are very popular**

- Dropping in Prison or School - High Risk and High Reward - High chance to die early but If you survive, you will be geared up
- Dropping in sparse area of the map - Low Risk and Low Reward - Low chance to die early but takes time to gear up

Found from the EDA that Distance Walked and Weapons accumulated can be proxy to separate the two strategies - Only players dropping in sparse areas walk more

Review progress

Sprint 1 - Planned User Stories

US1: Spin up EC2 on AWS. Create requirements.txt to make reproducible environment. ✓

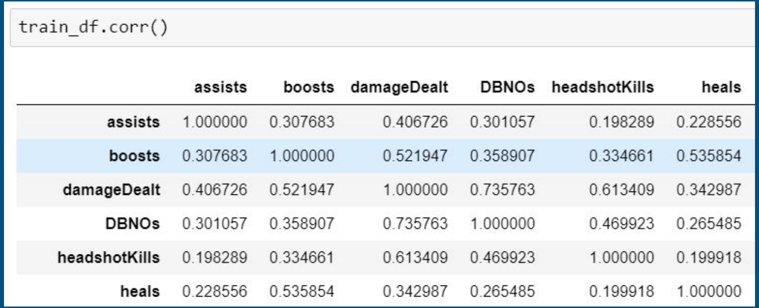
US2: Find appropriate PUBG dataset with relevant features ✓

US3: Upload the dataset into a RDS - Created schema in RDS. Yet to persist data ◕ 75%

US4: Data preprocessing - cleaning missing, incoherent and incorrect data ✓

US5: Exploratory Data Analysis - story generation and visualization ✓

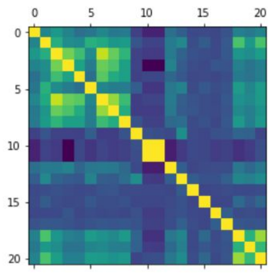
Demo/analysis

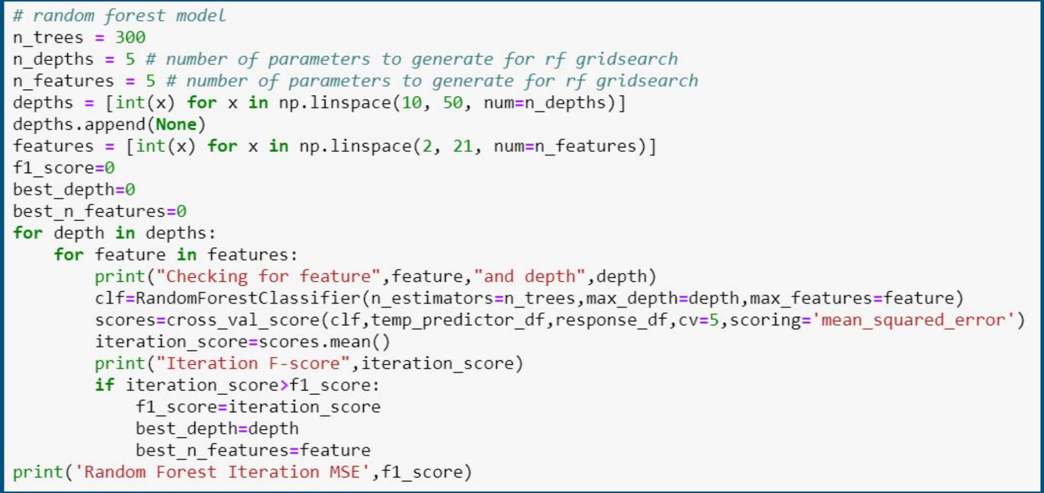


```
train_df.corr()
```

	assists	boosts	damageDealt	DBNOs	headshotKills	heals
assists	1.000000	0.307683	0.406726	0.301057	0.198289	0.228556
boosts	0.307683	1.000000	0.521947	0.358907	0.334661	0.535854
damageDealt	0.406726	0.521947	1.000000	0.735763	0.613409	0.342987
DBNOs	0.301057	0.358907	0.735763	1.000000	0.469923	0.265485
headshotKills	0.198289	0.334661	0.613409	0.469923	1.000000	0.199918
heals	0.228556	0.535854	0.342987	0.265485	0.199918	1.000000

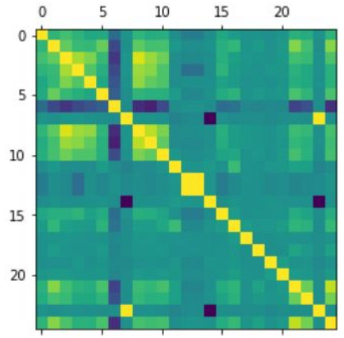
```
predictor_df = train_df.drop(['Id', 'groupId', 'matchId', 'killPlace','rankPoints','killPoints','winPoints'], axis=1)
plt.matshow(predictor_df.corr())
plt.show()
predictor_df.corr()
```





```
# random forest model
n_trees = 300
n_depths = 5 # number of parameters to generate for rf gridsearch
n_features = 5 # number of parameters to generate for rf gridsearch
depths = [int(x) for x in np.linspace(10, 50, num=n_depths)]
depths.append(None)
features = [int(x) for x in np.linspace(2, 21, num=n_features)]
f1_score=0
best_depth=0
best_n_features=0
for depth in depths:
    for feature in features:
        print("Checking for feature",feature,"and depth",depth)
        clf=RandomForestClassifier(n_estimators=n_trees,max_depth=depth,max_features=feature)
        scores=cross_val_score(clf,temp_predictor_df,response_df,cv=5,scoring='mean_squared_error')
        iteration_score=scores.mean()
        print("Iteration F-score",iteration_score)
        if iteration_score>f1_score:
            f1_score=iteration_score
            best_depth=depth
            best_n_features=feature
print('Random Forest Iteration MSE',f1_score)
```

```
plt.matshow(train_df.corr())
plt.show()
```



Lessons learned

Argparse was something that I was not aware of. It makes it easy to create generic scripts which can be used for more than one purpose

It's interesting to see PyCharm throwing warnings where there is a syntax error in docstring. Getting used to this is going to make reproducible research easier

Logging has proved vital in so many cases where the jupyter notebook connection gets disconnected but the log stores the essential information about the run. Especially useful when the code takes a long time to run

Recommendations

Plan for Sprint 2

US6: Feature Engineering - Leverage understanding from EDA to engineer the features to increase the models prediction power

US7: Prediction model selection and parameter tuning - Linear Regression, Random Forest and Neural Network

US8: Model Evaluation - cross validation to optimize variance/bias tradeoff

US9: Write unit tests to test each step of the model building process

US10: Front-end development - CSS/HTML UI with drop down and text boxes which accepts ingame statistics as user input and returns predicted finish placement

US11: Create a script to feed user input to the model to get predicted finish placement