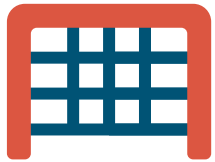
Top 5 Leagues analysis and predictions

Rodolfo Ferreira 03/12/20

GOOOOOAAAAAL !!!



Objectives of the project:

1. Provide valuable insights that can be useful to agents, investors, betting companies, managers, scouts and football professionals in general.
2. Predict season average ratings and most important stats for each position with regression models.
3. Predict level of season performance with classification models.
4. Find the best performing young stars. (in Tableau)

Data source
(football – api)

Top 5 Leagues

01 Premier League

02 La Liga

03 Bundesliga

04 Ligue I

05 Serie A

Collecting the data

Country → League ID → Team ID → Player ID


```
{"api":{"results":73,"leagues":[{"league_id":30,"name":"Primera Division","type":"League"
,"country":"Spain","country_code":"ES","season":2017,"season_start":"2017-08-18","season_
end":"2018-05-20","logo":"https:\/\/media.api-sports.io\/football\/leagues\/140.png","fla
g":"https:\/\/media.api-sports.io\/flags\/es.svg","standings":1,"is_current":0,"coverage"
:{"standings":true,"fixtures":{"events":true,"lineups":true,"statistics":true,"players_st
atistics":true},"players":true,"topScorers":true,"predictions":true,"odds":false}},{"leag
ue_id":33,"name":"Segunda Division","type":"League","country":"Spain","country_code":"ES"
,"season":2017,"season_start":"2017-08-18","season_end":"2018-06-16","logo":"https:\/\/me
dia.api-sports.io\/football\/leagues\/141.png","flag":"https:\/\/media.api-sports.io\/fla
gs\/es.svg","standings":1,"is_current":0,"coverage":{"standings":true,"fixtures":{"events
":true,"lineups":true,"statistics":true,"players_statistics":true},"players":true,"topSco
rers":true,"predictions":true,"odds":false}},{"league_id":64,"name":"Primera Division","t
ype":"League","country":"Spain","country_code":"ES","season":2016,"season_start":"2016-08
-19","season_end":"2017-05-21","logo":"https:\/\/media.api-sports.io\/football\/leagues\/
140.png","flag":"https:\/\/media.api-sports.io\/flags\/es.svg","standings":1,"is_current"
:0,"coverage":{"standings":true,"fixtures":{"events":true,"lineups":true,"statistics":tru
e,"players_statistics":true},"players":true,"topScorers":true,"predictions":true,"odds":f
alse}},{"league_id":65,"name":"Segunda Division","type":"League","country":"Spain","count
ry_code":"ES","season":2016,"season_start":"2016-08-19","season_end":"2017-06-24","logo":
```

Time to clean !!! 🕒

```
pl_df['goals_scored'] = pl_df['goals'].apply(lambda x: x['total'])
pl_df['goals_conceded'] = pl_df['goals'].apply(lambda x: x['conceded'])
pl_df['assists'] = pl_df['goals'].apply(lambda x: x['assists'])
pl_df['saves'] = pl_df['goals'].apply(lambda x: x['saves'])
pl_df.drop(['goals'], axis=1, inplace=True)
pl_df['total_shots'] = pl_df['shots'].apply(lambda x: x['total'])
pl_df['shots_on_target'] = pl_df['shots'].apply(lambda x: x['on'])
pl_df.drop(['shots'], axis=1, inplace=True)
pl_df['total_passes'] = pl_df['passes'].apply(lambda x: x['total'])
pl_df['key_passes'] = pl_df['passes'].apply(lambda x: x['key'])
pl_df['passing_accuracy'] = pl_df['passes'].apply(lambda x: x['accuracy'])
pl_df.drop(['passes'], axis=1, inplace=True)
pl_df['total_tackles'] = pl_df['tackles'].apply(lambda x: x['total'])
pl_df['blocks'] = pl_df['tackles'].apply(lambda x: x['blocks'])
pl_df['interceptions'] = pl_df['tackles'].apply(lambda x: x['interceptions'])
pl_df.drop(['tackles'], axis=1, inplace=True)
pl_df['total_duels'] = pl_df['duels'].apply(lambda x: x['total'])
pl_df['duels_won'] = pl_df['duels'].apply(lambda x: x['won'])
pl_df.drop(['duels'], axis=1, inplace=True)
pl_df['dribbles_attempts'] = pl_df['dribbles'].apply(lambda x: x['attempts'])
pl_df['successful_dribbles'] = pl_df['dribbles'].apply(lambda x: x['success'])
pl_df.drop(['dribbles'], axis=1, inplace=True)
pl_df['fouls_drawn'] = pl_df['fouls'].apply(lambda x: x['drawn'])
pl_df['fouls_committed'] = pl_df['fouls'].apply(lambda x: x['committed'])
pl_df.drop(['fouls'], axis=1, inplace=True)
pl_df['yellow_card'] = pl_df['cards'].apply(lambda x: x['yellow'])
pl_df['red_card'] = pl_df['cards'].apply(lambda x: x['red'])
pl_df['yellowred_card'] = pl_df['cards'].apply(lambda x: x['yellowred'])
pl_df.drop(['cards'], axis=1, inplace=True)
pl_df['penalties_won'] = pl_df['penalty'].apply(lambda x: x['won'])
pl_df['penalties_commited'] = pl_df['penalty'].apply(lambda x: x['commited'])
pl_df['penalties_scored'] = pl_df['penalty'].apply(lambda x: x['success'])
pl_df['penalties_missed'] = pl_df['penalty'].apply(lambda x: x['missed'])
pl_df['penalties_saved'] = pl_df['penalty'].apply(lambda x: x['saved'])
pl_df.drop(['penalty'], axis=1, inplace=True)
```

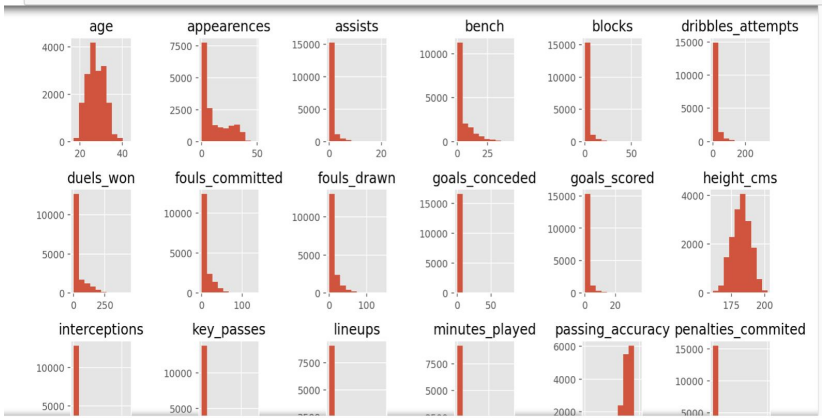
Concatenate all dataframes and observe last 3 seasons . . . Perfect !!!

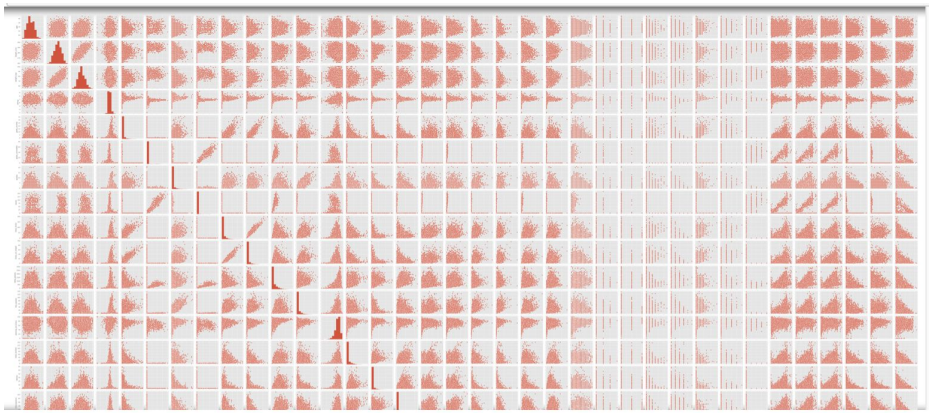
```
topleagues.season.value_counts()
```

```
2019-2020    6656
2018-2019    5325
2017-2018    5096
```

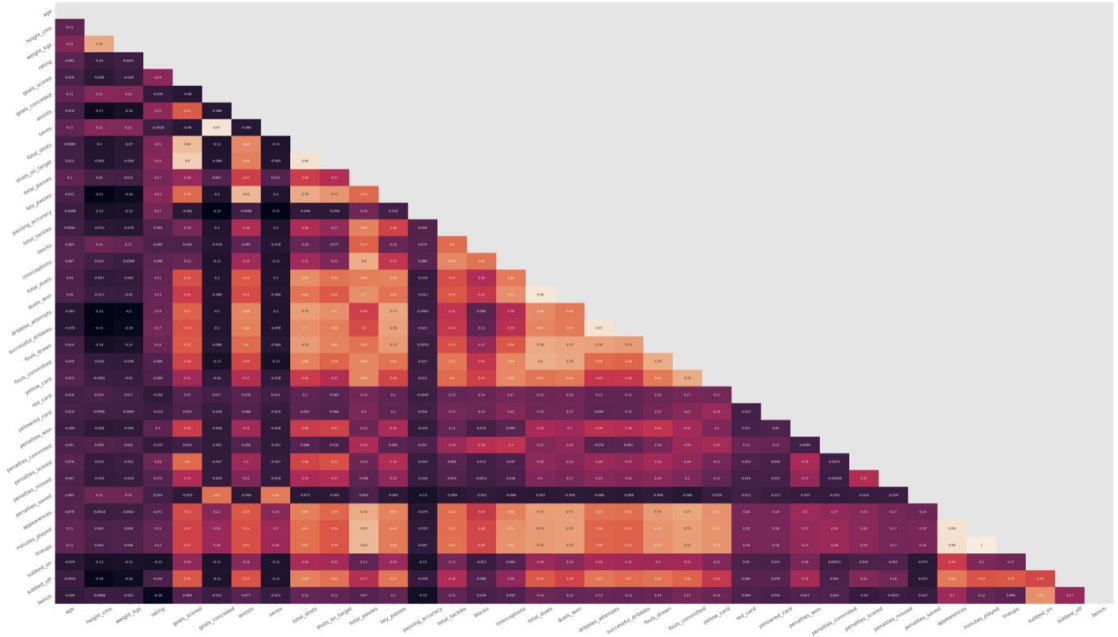
player_name	position	age	birth_country	nationality	height_cms	weight_kgs	rating	...	penalties_missed	penalties_saved	appearences	minutes_played
Sergio Rico	Goalkeeper	27	Spain	Spain	194.0	90.0	6.350000	...	0	0	2	180
Sergio Rico	Goalkeeper	27	Spain	Spain	194.0	90.0	6.751724	...	0	0	29	2610
Sergio Rico	Goalkeeper	27	Spain	Spain	194.0	90.0	6.695833	...	0	1	24	2155

Histograms and Pairplot

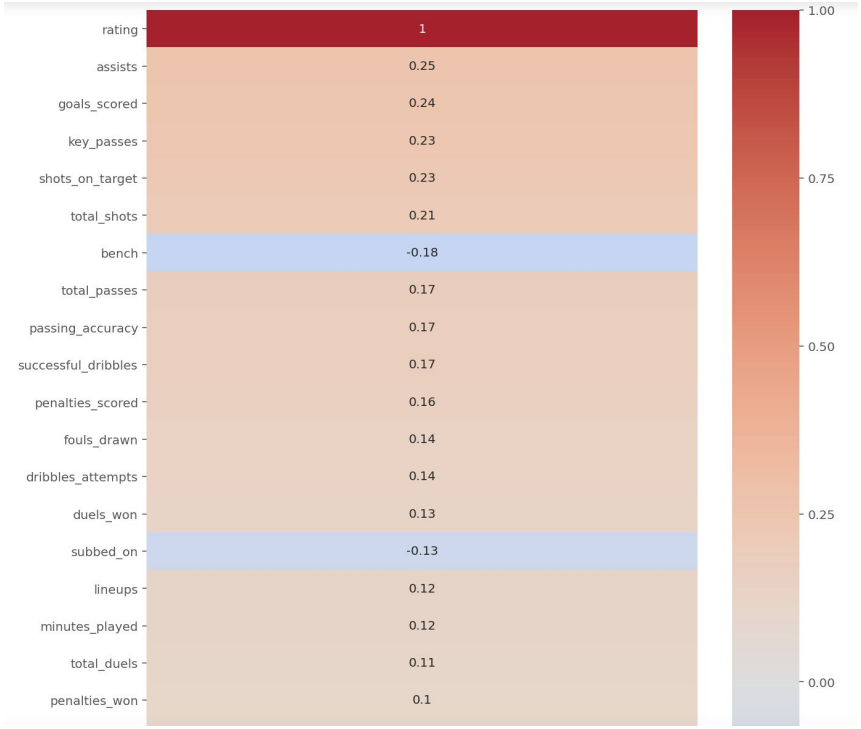


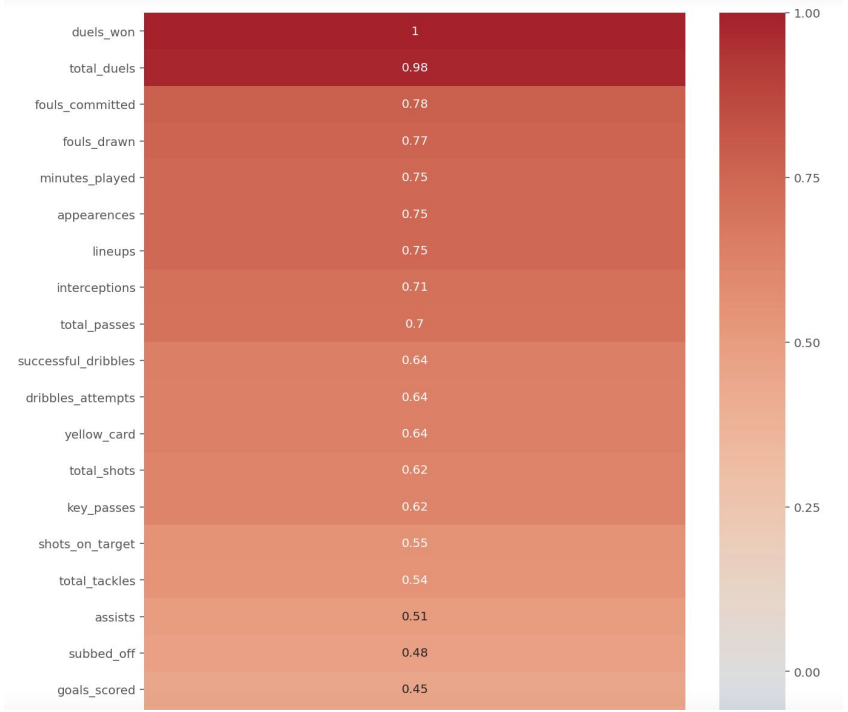


Heatmap and correlations

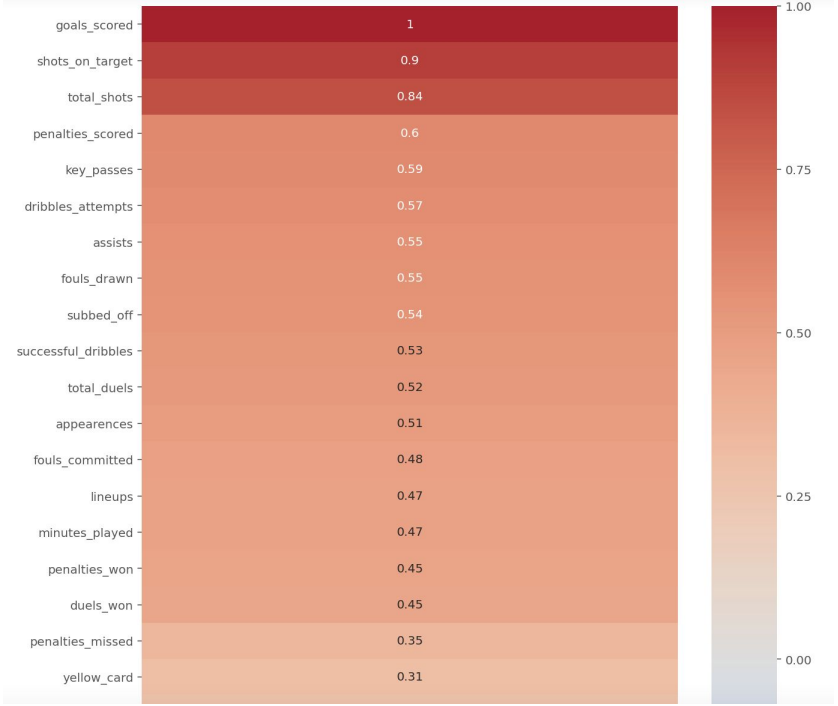


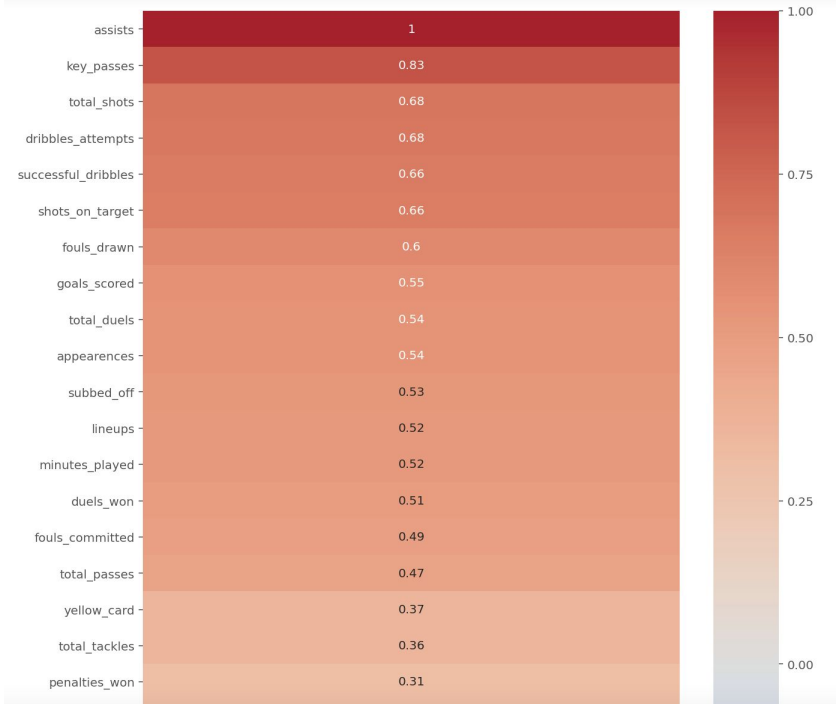
Heatmap and correlations (rating & duels_won)





Heatmap and correlations (goals_scored & assists)

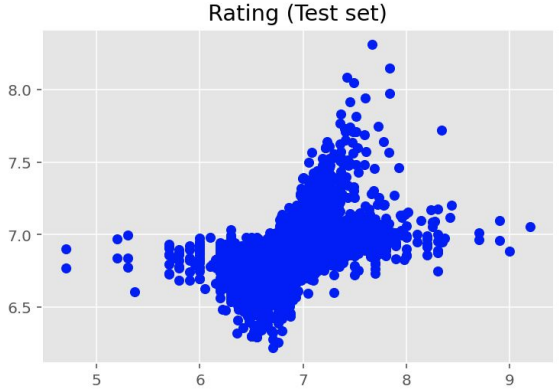




Modeling : rating (Decision Tree Regressor > Linear Regression)

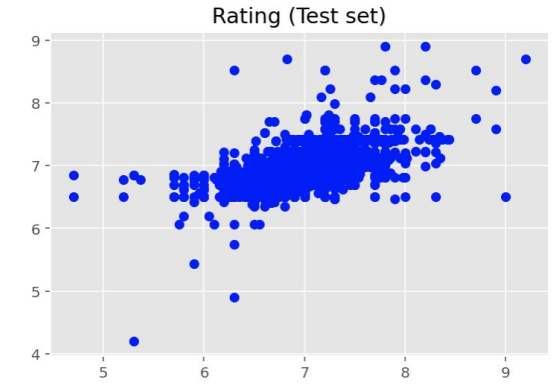
```
# fit a model
model = LinearRegression()
model.fit(X_train, y_train)
# Perform 5-fold cross validation
cv_scores = cross_val_score(model, X_train, y_train, cv=5)

print('Training Score:', model.score(X_train, y_train))
print('Cross validation score:', cv_scores)
print('Mean cross validation score:', cv_scores.mean())
print('Test Score:', model.score(X_test, y_test))
```

```
Training Score: 0.2202203971361817
Cross validation score: [0.20795205 0.21667938 0.2038173  0.23730764 0.21902085]
Mean cross validation score: 0.2169554443734551
Test Score: 0.2315316922926125
```



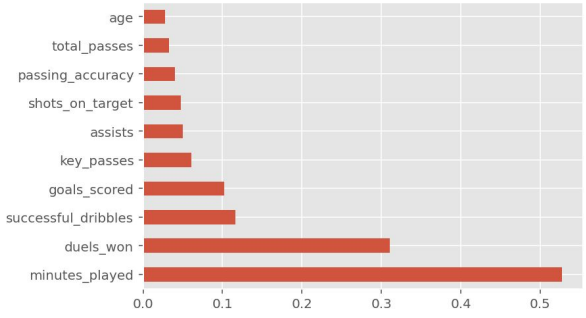

```
# Improving results with Decision Tree Regressor
dtr8 = DecisionTreeRegressor(max_depth=8)
dtrN = DecisionTreeRegressor(max_depth=None)
dtr8.fit(X_train, y_train)
dtrN.fit(X_train, y_train)
# cross validate the 2 models
dtr8_scores = cross_val_score(dtr8, X_train, y_train, cv=5)
dtrN_scores = cross_val_score(dtrN, X_train, y_train, cv=5)

# score the models
print(dtr8_scores, np.mean(dtr8_scores))
print(dtrN_scores, np.mean(dtrN_scores))
```

```
[0.3708624  0.35526991 0.30989264 0.35773266 0.3222955 ] 0.3432106216819707
[ 0.096505    0.04690949  0.05671737  0.03471497 -0.01882831] 0.043203703872142604
```

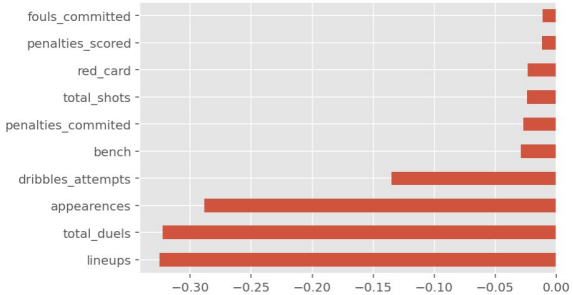


Modeling : rating (coefficients)

```
#10 features that impact the most
df_coef['coefficients'].sort_values(ascending = False)[:10].plot(kind='barh');
```



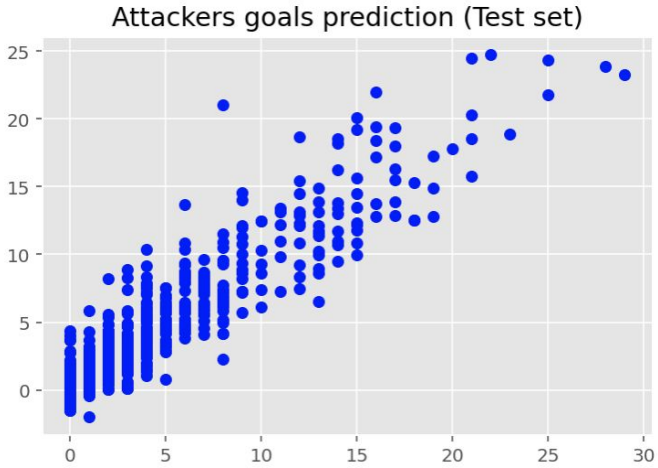

```
# 10 features that impact the least
df_coef['coefficients'].sort_values(ascending = True)[:10].plot(kind='barh');
```



Modeling : goals_scored (Attackers)

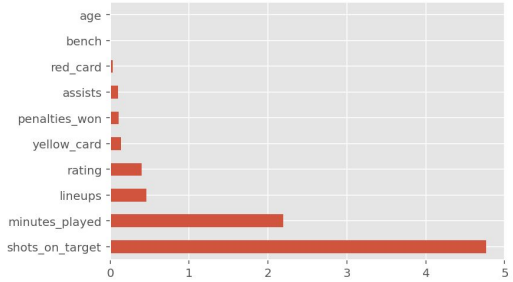
```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-4, 4, 10), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```

```
Best alpha: 0.005994842503189409
Cross-validated training scores: [0.88869296 0.87651146 0.86648526 0.89388014 0.87644349]
Mean cross-validated training score: 0.8804026621170709
Training score: 0.8868852485933866
Test Score: 0.8719898690804146
```

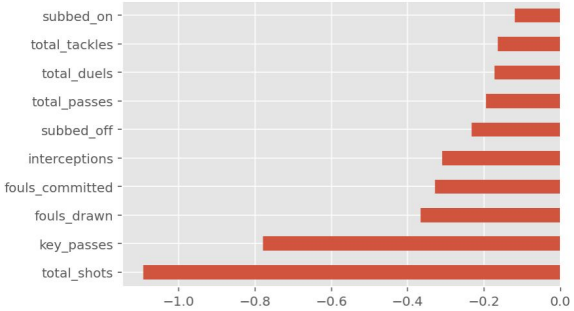


Modeling : goals_scored (Attackers , coefficients)

```
#10 features that impact the most
df_coef['coefficients'].sort_values(ascending = False)[:10].plot(kind='barh');
```

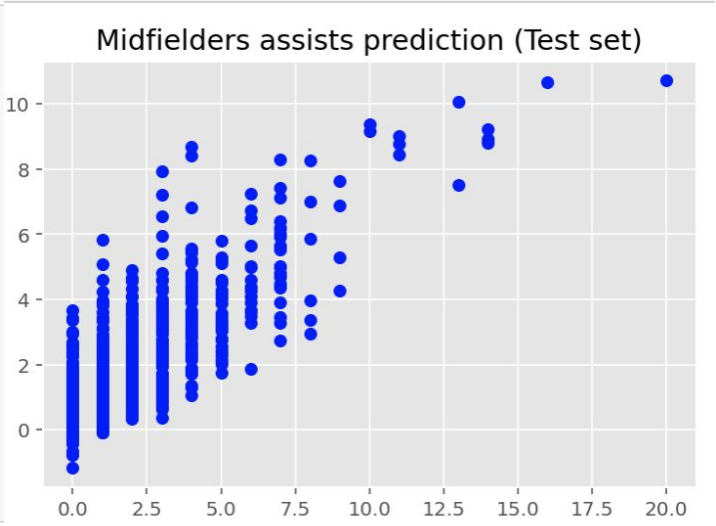



```
# 10 features that impact the least
df_coef['coefficients'].sort_values(ascending = True)[:10].plot(kind='barh');
```



Modeling : assists (Midfielders)

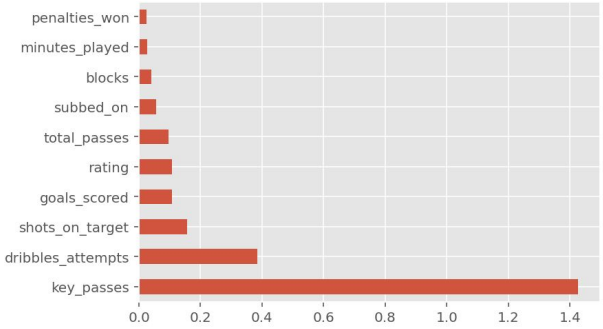
```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-4, 4, 10), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```



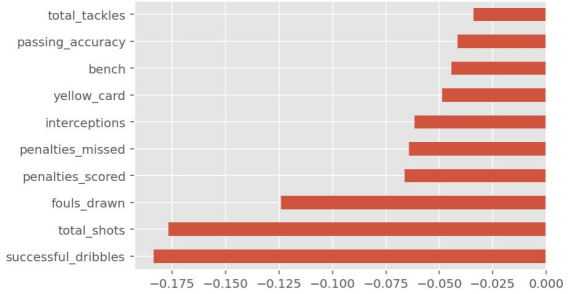

```
Best alpha: 0.000774263682681127
Cross-validated training scores: [0.72094953 0.72575787 0.64093404 0.71992693 0.61508483]
Mean cross-validated training score: 0.6845306409517158
Training score: 0.6995891223235444
Test Score: 0.7215333190777067
```

Modeling : assists (Midfielders , coefficients)

```
#10 features that impact the most
df_coef['coefficients'].sort_values(ascending = False)[:10].plot(kind='barh');
```




```
# 10 features that impact the least
df_coef['coefficients'].sort_values(ascending = True)[:10].plot(kind='barh');
```



Modeling : goals_scored (Midfielders vs Attackers)

```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-4, 4, 10), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```

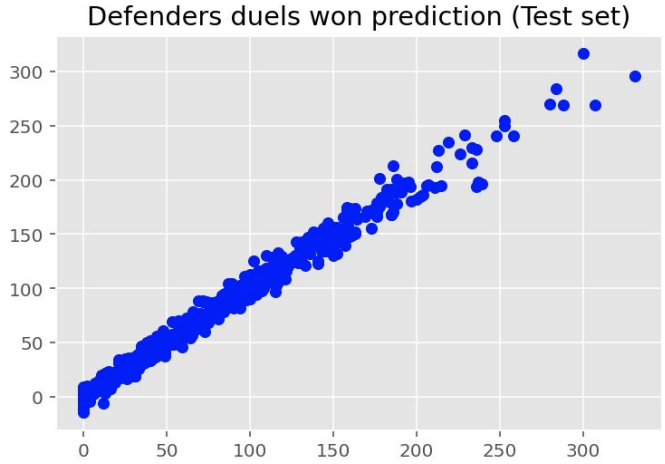
```
Best alpha: 0.005994842503189409
Cross-validated training scores: [0.78328661 0.74603311 0.74928854 0.71479918 0.72608131]
Mean cross-validated training score: 0.7438977475673354
Training score: 0.7537502451235687
Test Score: 0.7048872438499272
```

```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-4, 4, 10), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```

```
Best alpha: 0.005994842503189409
Cross-validated training scores: [0.88869296 0.87651146 0.86648526 0.89388014 0.87644349]
Mean cross-validated training score: 0.8804026621170709
Training score: 0.8868852485933866
Test Score: 0.8719898690804146
```

Modeling : duels_won (Defenders)

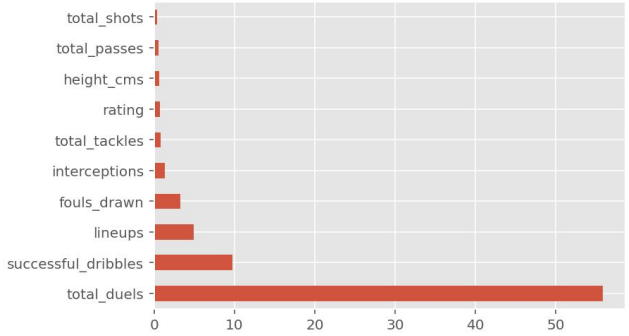
```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-5, 5, 100), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```



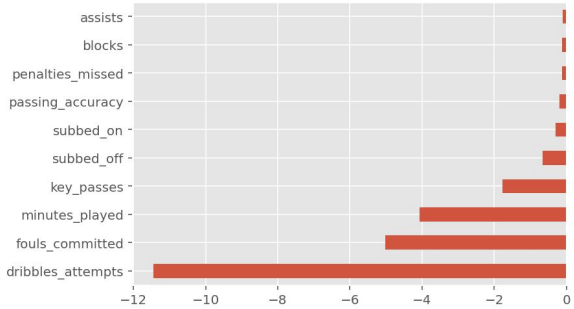

```
Best alpha: 0.00041320124001153346
Cross-validated training scores: [0.99026871 0.98926753 0.98988053 0.98722047 0.99091526]
Mean cross-validated training score: 0.989510499092819
Training score: 0.9898938186831282
Test Score: 0.9898683684066514
```

Modeling : duels_won (Defenders , coefficients)

```
#10 features that impact the most
df_coef['coefficients'].sort_values(ascending = False)[:10].plot(kind='barh')
```



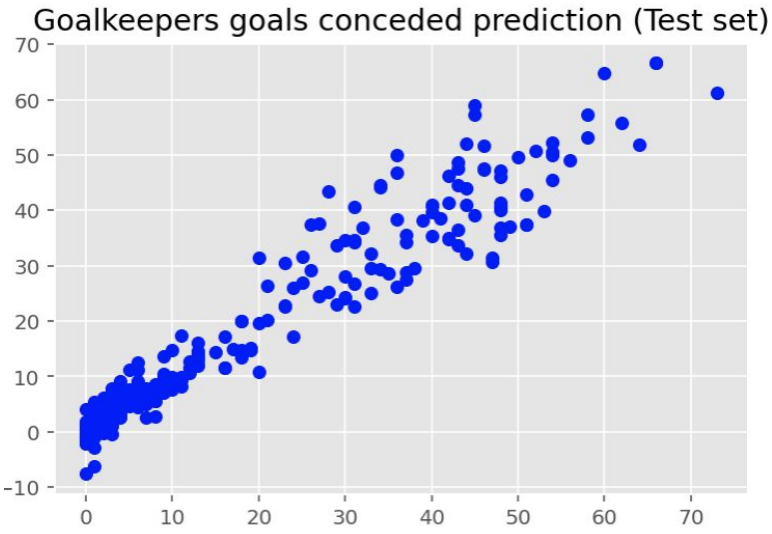

```
# 10 features that impact the least
df_coef['coefficients'].sort_values(ascending = True)[:10].plot(kind='barh');
```



Modeling : goals_conceded (Goalkeepers)

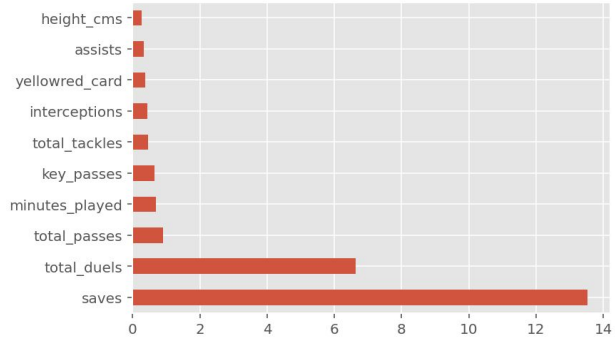
```
# create a LassoCV model instance
model = LassoCV(alphas=np.logspace(-4, 4, 1000), max_iter=10000, cv=5)
# fit the model
model.fit(X_train, y_train)
# get the best alpha
print('Best alpha:', model.alpha_)

alpha = model.alpha_
model = Lasso(alpha=alpha)

scores = cross_val_score(model, X_train, y_train, cv=5)
print("Cross-validated training scores:", scores)
print("Mean cross-validated training score:", scores.mean())

model.fit(X_train, y_train)

# evaluate on the training set
print('Training score:', model.score(X_train, y_train))
# evaluate on the test set
print("Test Score:", model.score(X_test, y_test))
```

```
Best alpha: 0.015072253093107555
Cross-validated training scores: [0.89800395 0.91940146 0.94177901 0.92953753 0.93239036]
Mean cross-validated training score: 0.9242224609563434
Training score: 0.931403859476435
Test Score: 0.9366874141821464
```

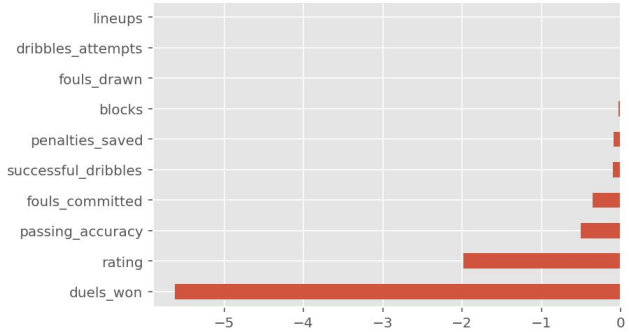


Modeling : goals_conceded (Goalkeepers , coefficients)

```
#10 features that impact the most
df_coef['coefficients'].sort_values(ascending = False)[:10].plot(kind='barh');
```




```
# 10 features that impact the least
df_coef['coefficients'].sort_values(ascending = True)[:10].plot(kind='barh');
```



Modeling : Performance/Rating

```
cr.rating.describe()
```

```
count    17077.000000
mean         6.874672
std          0.381241
min          3.000000
25%          6.680000
50%          6.850000
75%          7.062500
max          9.400000
Name: rating, dtype: float64
```

```
def rating_classification(rating, threshold=7.0):
    # over threshold will be considered a high performance
    if rating > threshold:
        return 1
    else:
        return 0
```

```
cr['performance'].value_counts(normalize=True)
```

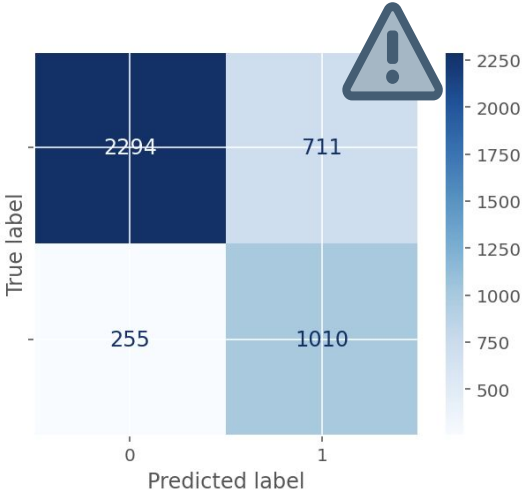
```
0    0.703636
1    0.296364
Name: performance, dtype: float64
```

Modeling : Performance/Rating

```
# logistic regression with L1 penalty
```

```
model_cvl1 = LogisticRegressionCV(Cs=np.logspace(-4, 4, 15),
                          penalty='l1', max_iter=10000, cv=5, class_weight='balanced',
                          solver='liblinear',
                          scoring='accuracy')

model_cvl1.fit(X_train, y_train)

cv_scores = cross_val_score(model_cvl1, X_train, y_train, cv=5, scoring='accuracy')

print('Training score:', model_cvl1.score(X_train,y_train))
print('Test Score:', model_cvl1.score(X_test, y_test))
print('Cross validation score:', cv_scores)
print('Mean cross validation score:', cv_scores.mean())
```

```
Training score: 0.7759818849066916
Test Score: 0.7737704918032787
Cross validation score: [0.77517564 0.7763466  0.76688793 0.76962124 0.77235455]
Mean cross validation score: 0.7720771946701879
```



Modeling : Performance/Rating – Precision

```
model = LogisticRegression(max_iter=1000)
```

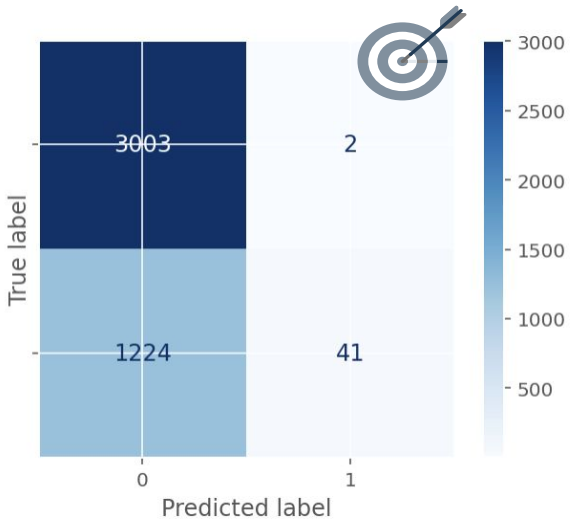
```
params = {'C': np.logspace(-5, 5, 15),
          'penalty': ['l2']}
```

```
gs = GridSearchCV(estimator=model,
                  param_grid=params,
                  cv=5,
                  scoring='precision',
                  return_train_score=True)
```

```
gs.fit(X_train, y_train)
```

```
Best Parameters:
{'C': 5.1794746792312125e-05, 'penalty': 'l2'}
Best estimator C:
5.1794746792312125e-05
Best estimator mean cross validated training score:
0.9595271601463551
Best estimator score on the full training set:
0.9508196721311475
Best estimator score on the test set:
0.9
```

Modeling : Performance/Rating – Precision




```
tp: 41
fp: 2
tn: 3003
fn: 1224
Number of correct classifications: 3044
Number of classification errors: 1226
```

```
print(classification_report(y_test, predictions_test))
```

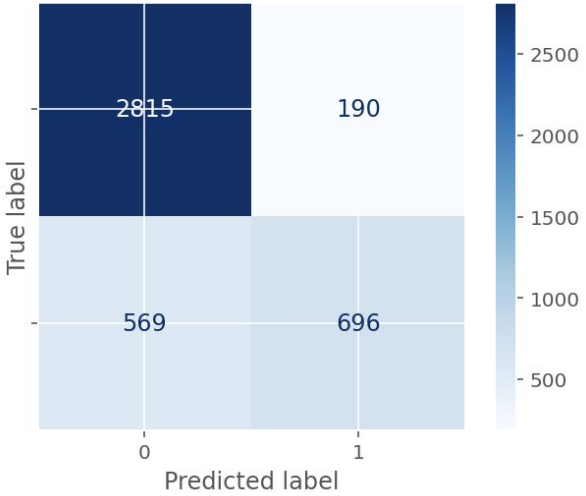
	precision	recall	f1-score	support
0	0.71	1.00	0.83	3005
1	0.95	0.03	0.06	1265
accuracy			0.71	4270
macro avg	0.83	0.52	0.45	4270
weighted avg	0.78	0.71	0.60	4270

Modeling : Performance/Rating – Precision

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf_params = {
    'n_estimators': [50, 100, 200, 300],
            'max_features': [1.0, 0.9, 0.8, 0.7],
            'max_depth': [None]+list(range(1,50,8))}
rf = RandomForestClassifier(random_state=1, class_weight='balanced')
rf_gridsearch = GridSearchCV(rf,
                             rf_params,
                             n_jobs=2,
                             cv=3,
                             verbose=3,
                            scoring='precision')
rf_gridsearch.fit(X_train, y_train)
```

Modeling : Performance/Rating – Precision




```
tp: 696
fp: 190
tn: 2815
fn: 569
Number of correct classifications: 3511
Number of classification errors: 759
```

```
print(classification_report(y_test, predictions_test))
```

	precision	recall	f1-score	support
0	0.83	0.94	0.88	3005
1	0.79	0.55	0.65	1265
accuracy			0.82	4270
macro avg	0.81	0.74	0.76	4270
weighted avg	0.82	0.82	0.81	4270

Further actions:

#Models

Apply different models

#Upscale dataset

Different leagues
Different continents
More seasons

#rodmarialvas@gmail.com

linkedin.com/in/rodolfosf/

Thank you

Diego Maradona, in his greatest ever moment.