**[Datatrained-Baseball-Case-Study](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study)/Datatrained-Baseball Case Study (1).ipynb**

[Go to file](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/find/main)

[@Abhistha1996](https://github.com/Abhistha1996)

[Abhistha1996](https://github.com/Abhistha1996) [Datatrained-Baseball Case Study notebook](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/commit/4f6abc85ef2b7b12dccb5bf9e83e3e3acc9d160f" \o "Datatrained-Baseball Case Study notebook)

Latest commit [4f6abc8](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/commit/4f6abc85ef2b7b12dccb5bf9e83e3e3acc9d160f) on Mar 28, 2021[**History**](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/commits/main/Datatrained-Baseball%20Case%20Study%20(1).ipynb)

**1** contributor

1330 lines (1330 sloc)  756 KB

[Raw](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/raw/main/Datatrained-Baseball%20Case%20Study%20(1).ipynb)[Blame](https://github.com/Abhistha1996/Datatrained-Baseball-Case-Study/blame/main/Datatrained-Baseball%20Case%20Study%20(1).ipynb)

Top of Form

Bottom of Form

Top of Form

Bottom of Form

* © 2022 GitHub, Inc.
* [Terms](https://docs.github.com/en/github/site-policy/github-terms-of-service)
* [Privacy](https://docs.github.com/en/github/site-policy/github-privacy-statement)
* [Security](https://github.com/security)
* [Status](https://www.githubstatus.com/)
* [Docs](https://docs.github.com/)
* [Contact GitHub](https://support.github.com/?tags=dotcom-footer)
* [Pricing](https://github.com/pricing)
* [API](https://docs.github.com/)
* [Training](https://services.github.com/)
* [Blog](https://github.blog/)
* [About](https://github.com/about)
* *#Importing the relevant libraries*
* **import** pandas **as** pd
* **import** numpy **as** np
* **from** pandas.plotting **import** scatter\_matrix
* **import** matplotlib.pyplot **as** plt
* **from** sklearn **import** model\_selection
* **import** seaborn **as** sns
* **%matplotlib** inline
* **from** sklearn.model\_selection **import** train\_test\_split
* **from** sklearn.preprocessing **import** LabelEncoder, StandardScaler
* scaler **=** StandardScaler()
* **from** sklearn.model\_selection **import** RepeatedStratifiedKFold
* **from** sklearn.ensemble **import** RandomForestRegressor
* **from** sklearn.linear\_model **import** LinearRegression
* **from** sklearn.linear\_model **import** Lasso
* **from** sklearn.linear\_model **import** ElasticNet
* **from** sklearn.tree **import** DecisionTreeRegressor
* **from** sklearn.neighbors **import** KNeighborsRegressor
* **from** sklearn.ensemble **import** GradientBoostingRegressor
* **from** xgboost **import** XGBRegressor
* **from** sklearn.preprocessing **import** MinMaxScaler
* **from** sklearn.model\_selection **import** KFold
* **from** sklearn.model\_selection **import** cross\_val\_score
* **from** sklearn **import** metrics
* **import** warnings
* warnings**.**filterwarnings("ignore")
* In [2]:
* *#reading the train datset*
* df\_ **=** pd**.**read\_csv("https://raw.githubusercontent.com/dsrscientist/Data-Science-ML-Capstone-Projects/master/baseball.csv" )
* *#df=df.dropna()*
* df**=**df\_
* df
* *#Checking out the feature names and datatypes*
* df**.**info()
* *#Checking out the*
* df**.**describe()
* *#creating scatterplots between various features to understand relasionship between them*
* sns**.**pairplot(df,diag\_kind**=**"kde")
* *#correlation matrix*
* plt**.**figure(figsize**=**(20,15))
* *#ax=subplot(111)*
* matrix **=** np**.**triu(df**.**corr())
* sns**.**heatmap(df**.**corr(), annot**=True**, mask**=**matrix)
* *#Split the data to y and x with x is without the class's.*
* Y **=** df[['W']]
* X**=**df[['RA', 'ER', 'SV', 'SHO']]
* *#X = df.drop(['W'], axis=1)*
* In [22]:
* *#Splitting the train and test data and scaling respectively*
* X\_train, X\_test,Y\_train, Y\_test**=**model\_selection**.**train\_test\_split(X,Y,test\_size**=**0.2,random\_state**=**2)
* X\_train**=**scaler**.**fit\_transform(X\_train)
* X\_test**=**scaler**.**transform(X\_test)
* In [23]:
* *#spot checking algorithms*
* models **=** []
* models**.**append(('LR',LinearRegression()))
* models**.**append(('LASSO', Lasso()))
* models**.**append(('EN', ElasticNet()))
* models**.**append(('KNN', KNeighborsRegressor()))
* models**.**append(('CART', DecisionTreeRegressor()))
* models**.**append(('GBM', GradientBoostingRegressor()))
* models**.**append(('XGB', XGBRegressor()))
* models**.**append(('RFG', RandomForestRegressor()))
* results **=** []
* names **=** []
* **for** name, model **in** models:
* kfold **=** KFold(n\_splits**=**10, random\_state**=**21,shuffle**=True**)
* cv\_results **=** cross\_val\_score(model, X\_train, Y\_train, cv**=**kfold, scoring**=**'explained\_variance')
* results**.**append(cv\_results)
* names**.**append(name)
* msg **=** "%s: %f (%f)" **%** (name, cv\_results**.**mean(), cv\_results**.**std())
* print(msg)
* *#creating linear regression as a baseline model*
* model **=** LinearRegression()
* model**.**fit(X\_train, Y\_train)
* Y\_pred **=** model**.**predict(X\_test)
* score **=** model**.**score(X\_train, Y\_train)
* print('Training Score:', score)
* score **=** model**.**score(X\_test, Y\_test)
* print('Testing Score:', score)
* Training Score: 0.7085898578272409
* Testing Score: 0.5599992825843976
* In [25]:
* *#Finding out the mean absolute error*
* **from** sklearn.metrics **import** mean\_absolute\_error,mean\_squared\_error,r2\_score
* mae **=** np**.**round(mean\_absolute\_error(Y\_test,Y\_pred),3)
* print('Mean Absolute Error:', mae)
* Mean Absolute Error: 6.23
* In [26]:
* *#Finding out the mean Squarred Error*
* mse **=** np**.**round(mean\_squared\_error(Y\_test,Y\_pred),3)
* print('Mean Squared Error:', mse)
* Mean Squared Error: 46.457
* In [27]:
* *#Finding out the R2 Score*
* score **=** np**.**round(r2\_score(Y\_test,Y\_pred),3)
* print('R2 Score:', score)
* R2 Score: 0.56