Problem Statement:

This dataset utilizes data from 2014 Major League Baseball seasons in order to develop an algorithm that predicts the number of wins for a given team in the 2015 season based on several different indicators of success. There are 16 different features that will be used as the inputs to the machine learning and the output will be a value that represents the number of wins.

-- Input features: Runs, At Bats, Hits, Doubles, Triples, Homeruns, Walks, Strikeouts, Stolen Bases, Runs Allowed, Earned Runs, Earned Run Average (ERA), Shutouts, Saves, and Errors

-- Output: Number of predicted wins (W)

In [1]:
```
#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-
#df=df.dropna()
df=df_
df
```

Out[2]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86
5	93	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34	88
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97
10	88	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45	119

In [3]:
```
#Checking out the feature names and datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   W       30 non-null     int64  
 1   R       30 non-null     int64  
 2   AB      30 non-null     int64  
 3   H       30 non-null     int64  
 4   2B      30 non-null     int64  
 5   3B      30 non-null     int64  
 6   HR      30 non-null     int64  
 7   BB      30 non-null     int64  
 8   SO      30 non-null     int64  
 9   SB      30 non-null     int64  
 10  RA      30 non-null     int64  
 11  ER      30 non-null     int64  
 12  ERA     30 non-null     float64
 13  CG      30 non-null     int64  
 14  SHO     30 non-null     int64  
 15  SV      30 non-null     int64  
 16  E       30 non-null     int64  
dtypes: float64(1), int64(16)
memory usage: 4.1 KB
```

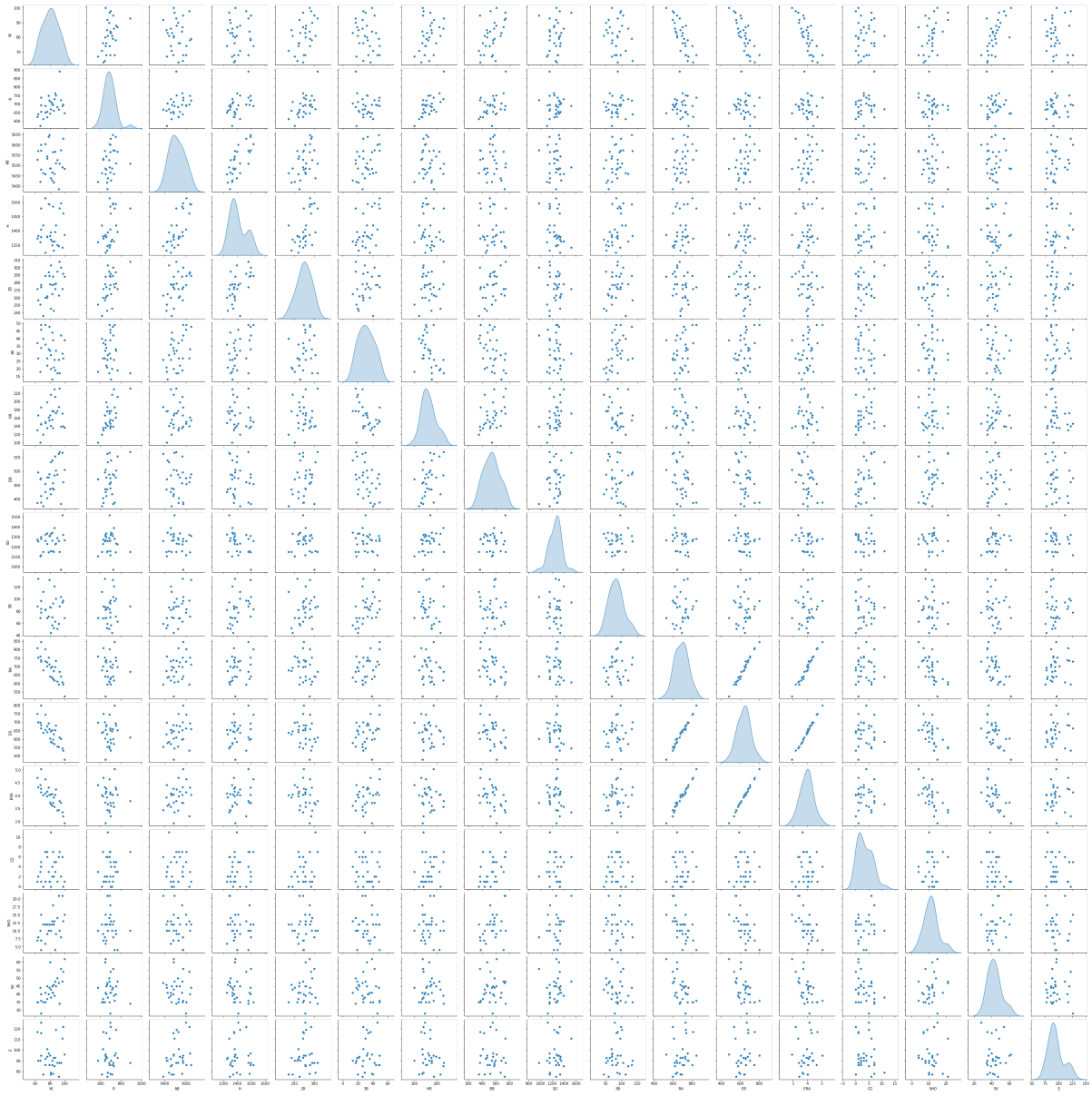
In [4]:
```
#Checking out the
df.describe()
```

Out[4]:

	W	R	AB	H	2B	3B	HR	
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30
mean	80.966667	688.233333	5516.266667	1403.533333	274.733333	31.300000	163.633333	469
std	10.453455	58.761754	70.467372	57.140923	18.095405	10.452355	31.823309	57
min	63.000000	573.000000	5385.000000	1324.000000	236.000000	13.000000	100.000000	375
25%	74.000000	651.250000	5464.000000	1363.000000	262.250000	23.000000	140.250000	428
50%	81.000000	689.000000	5510.000000	1382.500000	275.500000	31.000000	158.500000	473
75%	87.750000	718.250000	5570.000000	1451.500000	288.750000	39.000000	177.000000	501
max	100.000000	891.000000	5649.000000	1515.000000	308.000000	49.000000	232.000000	570

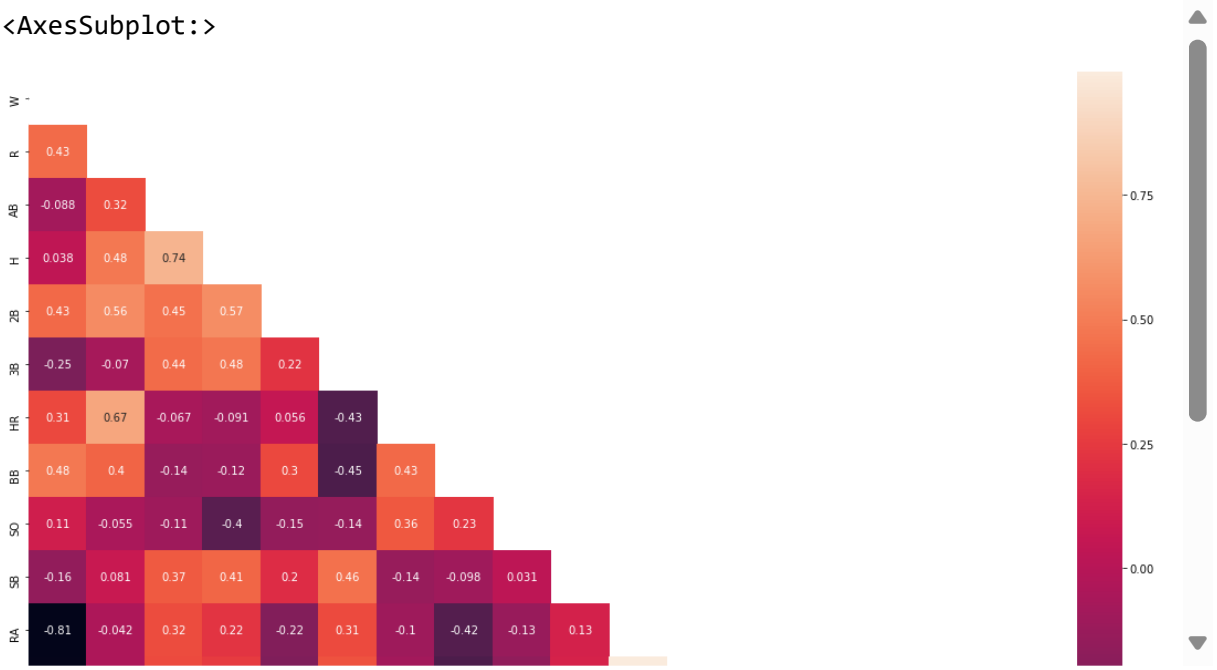
In [5]:
```
#creating scatterplots between various features to understand relasionship betw
sns.pairplot(df,diag_kind="kde")
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x2528a4ae5e0>



In [6]:
```
#correlation matrix
plt.figure(figsize=(20,15))
#ax=subplot(111)
matrix = np.triu(df.corr())
sns.heatmap(df.corr(), annot=True, mask=matrix)
```

Out[6]: <AxesSubplot:>



In [21]:
```
#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=
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='ex
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.078016 (0.936354)
LASSO: 0.409270 (0.621218)
EN: 0.460030 (0.549268)
KNN: 0.168013 (0.810888)
CART: -0.411674 (1.548366)
GBM: -0.174232 (1.125008)
XGB: -0.431659 (1.455410)
RFG: -0.028752 (1.070464)
```

In [24]:
```
#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
```

In [ ]:
```

```