Concrete Dataset

Week3 Task 4

Week 3 task 4

Task 05

Importing the libraries

In [1]:
```
#import libraries fro the dataset exploration
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve,auc
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import warnings  # Ignoring Warnings
warnings.filterwarnings("ignore")
```

Loading the dataset

In [2]:
```
#reading the csv file
dset =pd.read_csv("C:\\Users\\Roxton\\Desktop\\concrete.csv")
dset
```

Out[2]:

	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5) (kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Fine Aggregate (componen 7)(kg in a m^3 mixture
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5
...	...	...	...	...	...	...	..
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5

1030 rows × 9 columns

In [68]:
```
dset.describe().T
```

Out[68]:

	count	mean	std	min	25%	50%	75%	max
cement	1030.0	281.167864	104.506364	102.00	192.375	272.900	350.000	540.0
slag	1030.0	73.895825	86.279342	0.00	0.000	22.000	142.950	359.4
ash	1030.0	54.188350	63.997004	0.00	0.000	0.000	118.300	200.1
water	1030.0	181.567282	21.354219	121.80	164.900	185.000	192.000	247.0
superplastic	1030.0	6.204660	5.973841	0.00	0.000	6.400	10.200	32.2
coarseagg	1030.0	972.918932	77.753954	801.00	932.000	968.000	1029.400	1145.0
fineagg	1030.0	773.580485	80.175980	594.00	730.950	779.500	824.000	992.6
age	1030.0	45.662136	63.169912	1.00	7.000	28.000	56.000	365.0
strength	1030.0	35.817961	16.705742	2.33	23.710	34.445	46.135	82.6

In [3]:
```
dset.rename(columns=dict(zip(dset.columns, ['cement', 'slag', 'ash', 'water', 'su
       'fineagg', 'age', 'strength'])), inplace=True)
```

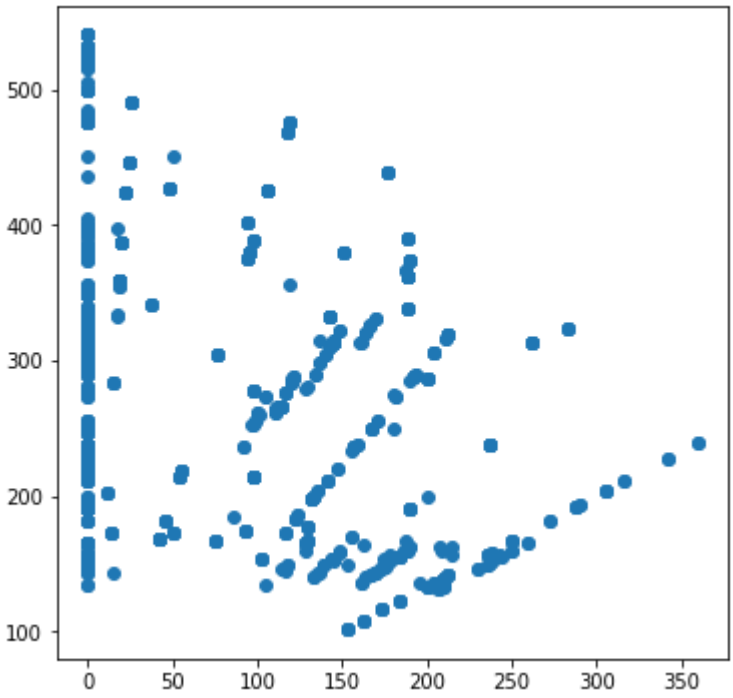
Checking for null values

In [4]:
```
#assesing any missing values so that we drop themsct
print(dset.isnull())
```

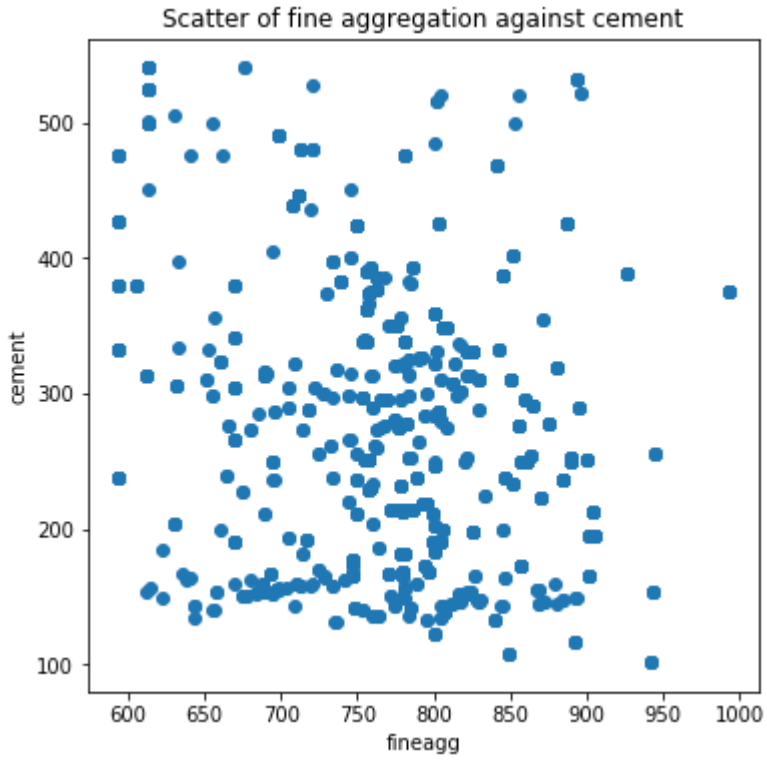
```
      cement   slag    ash  water  superplastic  coarseagg  fineagg    age  \
0      False  False  False  False         False      False    False  False
1      False  False  False  False         False      False    False  False
2      False  False  False  False         False      False    False  False
3      False  False  False  False         False      False    False  False
4      False  False  False  False         False      False    False  False
...      ...    ...    ...    ...           ...        ...      ...    ...
1025   False  False  False  False         False      False    False  False
1026   False  False  False  False         False      False    False  False
1027   False  False  False  False         False      False    False  False
1028   False  False  False  False         False      False    False  False
1029   False  False  False  False         False      False    False  False

      strength
0        False
1        False
2        False
3        False
4        False
...        ...
1025     False
1026     False
1027     False
1028     False
1029     False

[1030 rows x 9 columns]
```

Scatter Plots

In [5]:
```
#scatter plots
g=plt.figure(figsize=(6,6))
p=g.add_subplot(111)
y=dset["cement"]
x=dset["slag"]
plt.scatter(x,y)

plt.show()
```



In [6]:
```
#scatter plots
g=plt.figure(figsize=(6,6))
p=g.add_subplot(111)

y=dset["cement"]
x=dset["fineagg"]
plt.scatter(x,y)
plt.title('Scatter of fine aggregation against cement')
plt.xlabel('fineagg')
plt.ylabel('cement')
plt.show()
```



In [7]:
```
#we use shape to determine size of the dataset
dset.shape
```

Out[7]:
```
(1030, 9)
```

In [8]:
```
#lets count the values of each column
dset['cement'].value_counts()
```

Out[8]:
```
362.6    20
425.0    20
251.4    15
310.0    14
446.0    14
         ..
313.8     1
147.8     1
260.9     1
136.4     1
321.3     1
Name: cement, Length: 278, dtype: int64
```

In [9]:
```
dset["slag"].value_counts()
```

Out[9]:
```
0.0      471
189.0     30
106.3     20
24.0      14
20.0      12
         ...
178.1      1
148.9      1
98.8       1
128.9      1
169.4      1
Name: slag, Length: 185, dtype: int64
```

In [10]:
```
dset['slag'].value_counts()
```

Out[10]:
```
0.0      471
189.0     30
106.3     20
24.0      14
20.0      12
         ...
178.1      1
148.9      1
98.8       1
128.9      1
169.4      1
Name: slag, Length: 185, dtype: int64
```

In [11]:
```
dset.describe()
```

Out[11]:

	cement	slag	ash	water	superplastic	coarseagg	fineagg
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000

In [12]:
```
#data dimensions
print("Number of rows    :",dset.shape[0])
print("Number of columns :",dset.shape[1])
```

```
Number of rows    : 1030
Number of columns : 9
```

In [13]:
```
dset.dtypes
```

Out[13]:
```
cement          float64
slag            float64
ash             float64
water           float64
superplastic    float64
coarseagg       float64
fineagg         float64
age               int64
strength        float64
dtype: object
```

Checking for null values

In [14]:
```
#checking for null values
dset.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[14]:
```
cement          0
slag            0
ash             0
water           0
superplastic    0
coarseagg       0
fineagg         0
age             0
strength        0
dtype: int64
```

Plot of null values

In [15]:
```
plt.figure(figsize=(15,10))
sns.heatmap(dset.isnull(), yticklabels=False, cbar=False, cmap='Reds')
plt.xticks(fontsize=14)
plt.title(' Missing Values by Heat Map', fontsize=12)
plt.show()
```

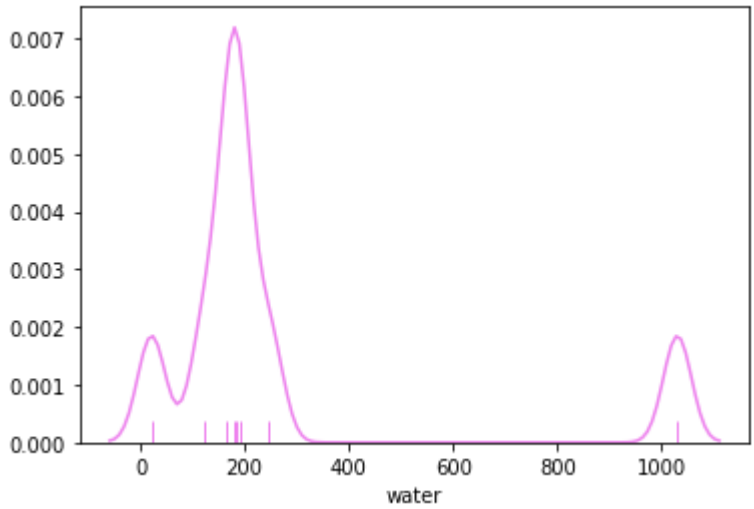
In [16]:
```
#statistics of water
st=dset['water'].describe()
st
```

Out[16]:
```
count    1030.000000
mean      181.567282
std        21.354219
min       121.800000
25%       164.900000
50%       185.000000
75%       192.000000
max       247.000000
Name: water, dtype: float64
```

Distribution Plots

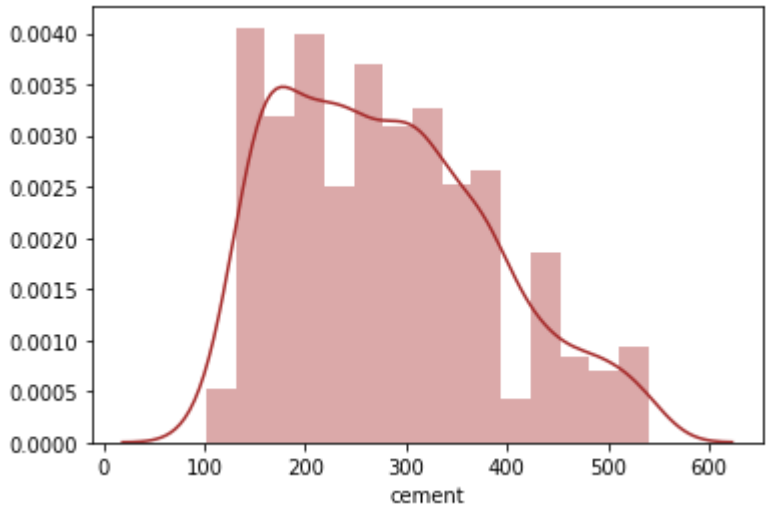
In [17]:
```
#statistical distibution of water in a graph
sns.distplot(st, hist=False, bins=20,rug=True,color ='violet')
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5a3e21ec8>
```



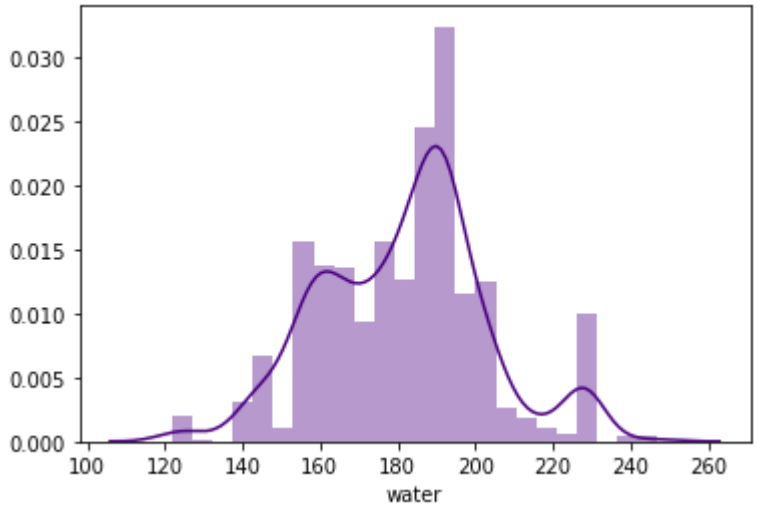
In [18]:
```
#cement distribution
sns.distplot(a=dset['cement'],color='brown')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a3ea33c8>



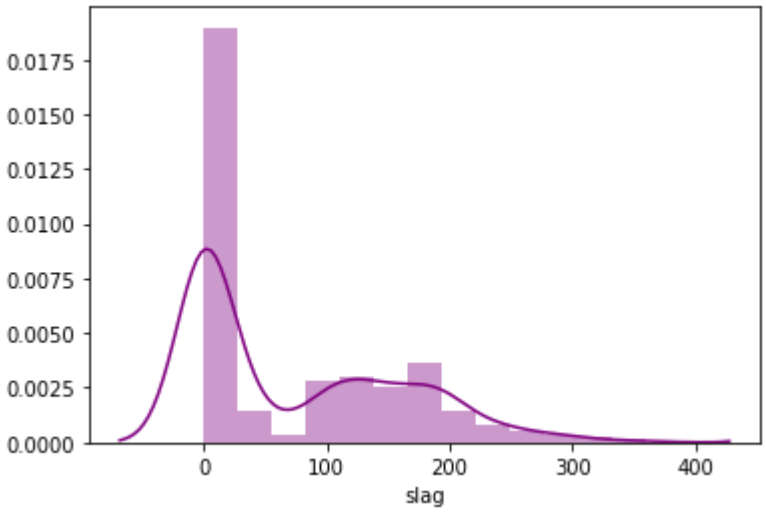
In [19]:
```
#water distribution
sns.distplot(a=dset['water'],color='indigo')
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a3f39148>



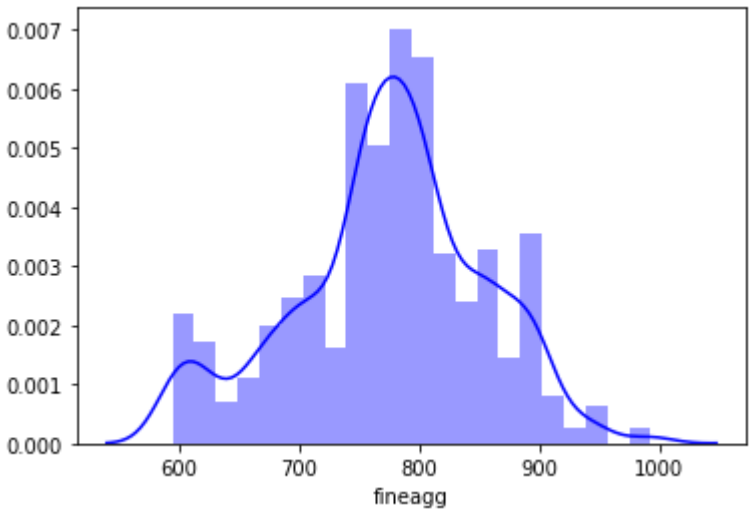
In [20]:
```
# slag distribution
sns.distplot(a=dset['slag'],color='purple')
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a3ffca08>



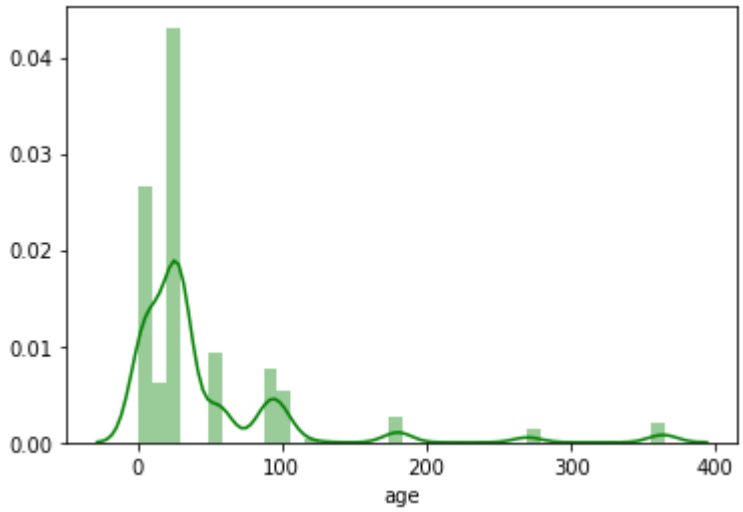
In [21]:
```
#fine coarse aggregation distribution
sns.distplot(a=dset['fineagg'],color='blue')
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a408e508>



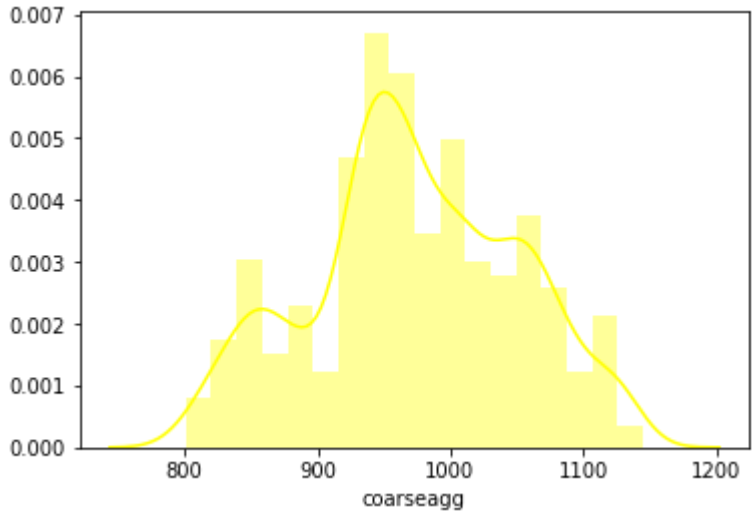
In [22]:
```
#age distribution
sns.distplot(a=dset['age'],color='green')
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a34db248>



In [23]:
```
#coarse aggregationdistribution
sns.distplot(a=dset['coarseagg'],color='yellow')
```

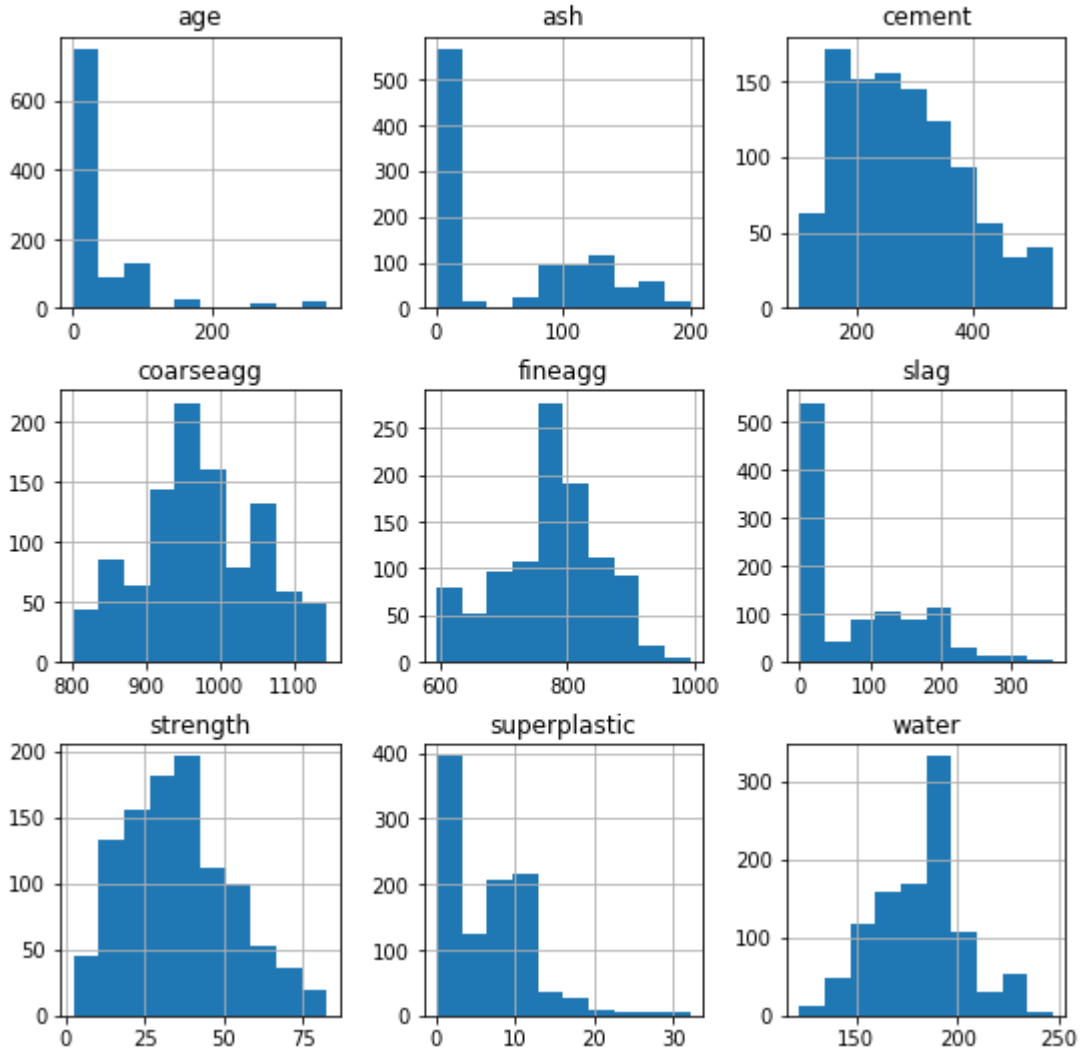
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a412d188>



In [24]:
```
#finding the correllations of columns
corr = dset.corr()
print(corr)
```

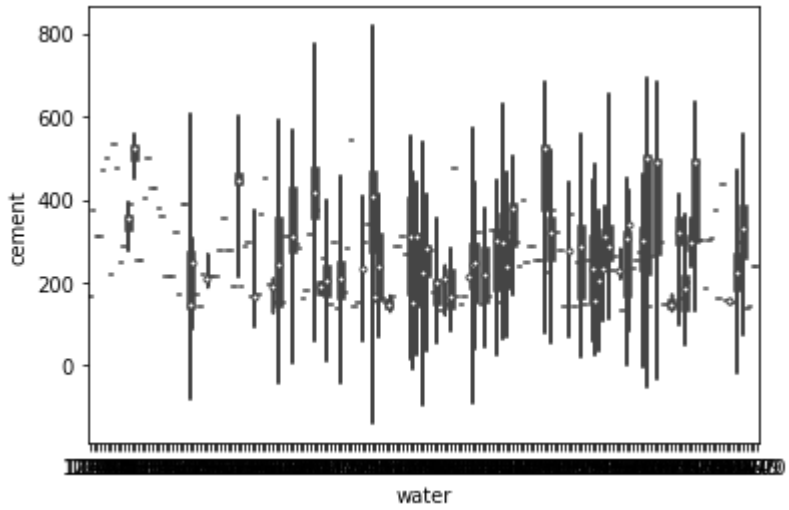
```
                cement      slag       ash     water  superplastic  coarseagg
\
cement        1.000000 -0.275216 -0.397467 -0.081587      0.092386  -0.109349
slag         -0.275216  1.000000 -0.323580  0.107252      0.043270  -0.283999
ash          -0.397467 -0.323580  1.000000 -0.256984      0.377503  -0.009961
water        -0.081587  0.107252 -0.256984  1.000000     -0.657533  -0.182294
superplastic  0.092386  0.043270  0.377503 -0.657533      1.000000  -0.265999
coarseagg    -0.109349 -0.283999 -0.009961 -0.182294     -0.265999   1.000000
fineagg      -0.222718 -0.281603  0.079108 -0.450661      0.222691  -0.178481
age           0.081946 -0.044246 -0.154371  0.277618     -0.192700  -0.003016
strength      0.497832  0.134829 -0.105755 -0.289633      0.366079  -0.164935

                fineagg       age  strength
cement        -0.222718  0.081946  0.497832
slag          -0.281603 -0.044246  0.134829
ash            0.079108 -0.154371 -0.105755
water         -0.450661  0.277618 -0.289633
superplastic   0.222691 -0.192700  0.366079
coarseagg     -0.178481 -0.003016 -0.164935
fineagg        1.000000 -0.156095 -0.167241
age           -0.156095  1.000000  0.328873
strength      -0.167241  0.328873  1.000000
```

In [25]:
```
#histogramsfor individual columns
bargraphs =dset.hist(figsize=(9,9))
plt.show()
```

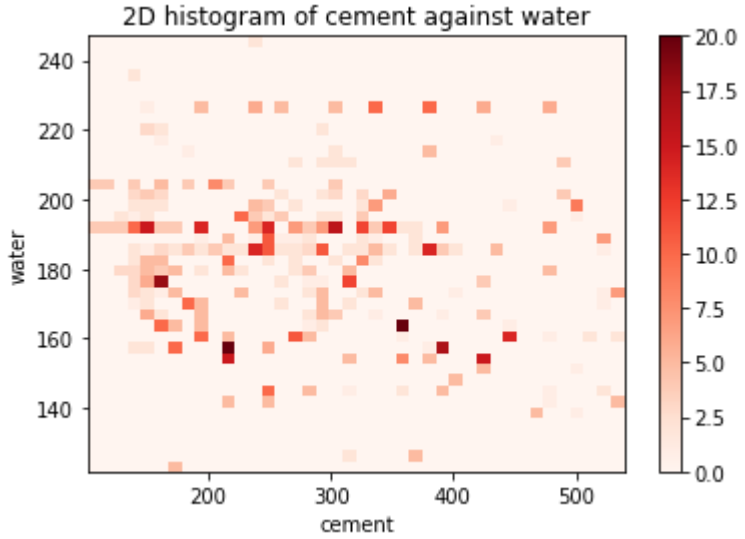


In [26]:
```
#violin plots cement against water
#plt.subplot(2,4,4)
sns.violinplot(y= 'cement', x='water', data = dset)
```

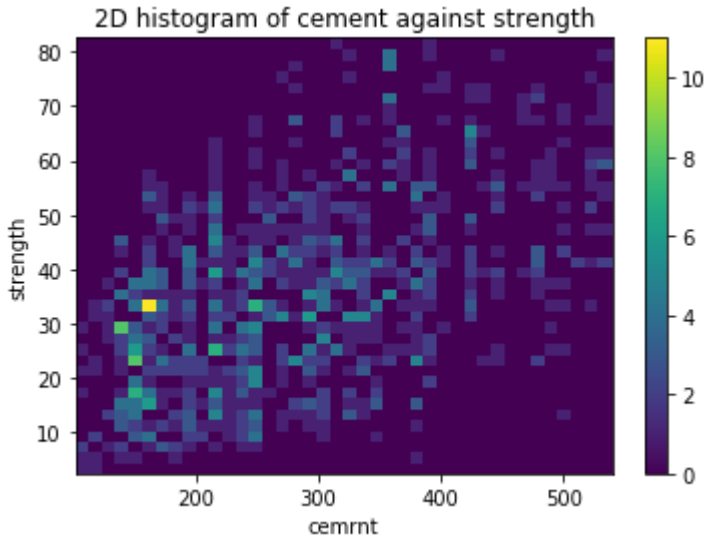
Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5a462a888>
```



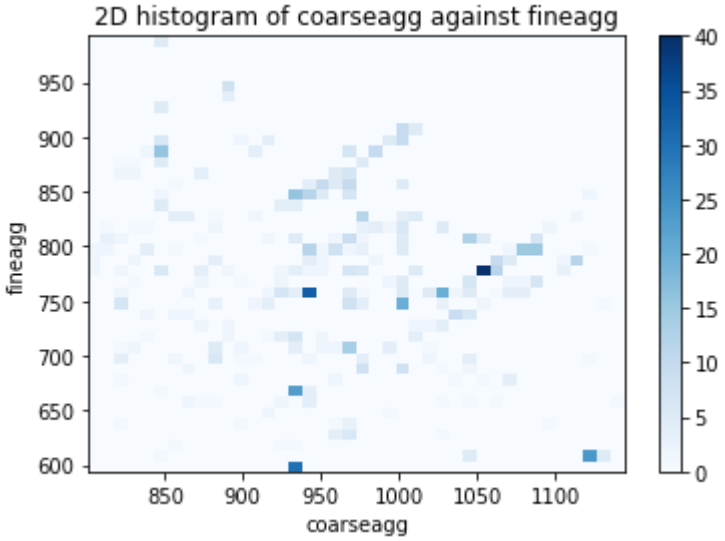
In [27]:
```
#2d histogram
plt.hist2d(dset.cement, dset.water, bins=40, cmap='Reds')
plt.colorbar()
plt.title('2D histogram of cement against water')
plt.xlabel('cement')
plt.ylabel('water')
plt.show()
```



In [28]:
```
#2d histogram
plt.hist2d(dset.cement, dset.strength, bins=40, cmap='viridis')
plt.colorbar()
plt.title('2D histogram of cement against strength')
plt.xlabel('cemrnt')
plt.ylabel('strength')
plt.show()
```

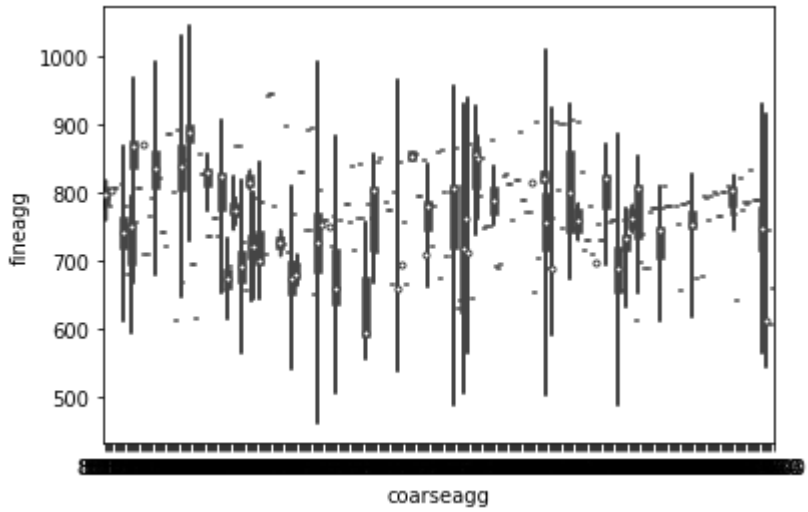


In [29]:
```
#2d histogram
plt.hist2d(dset.coarseagg, dset.fineagg, bins=40, cmap='Blues')
plt.colorbar()
plt.title('2D histogram of coarseagg against fineagg')
plt.xlabel('coarseagg')
plt.ylabel('fineagg')
plt.show()
```

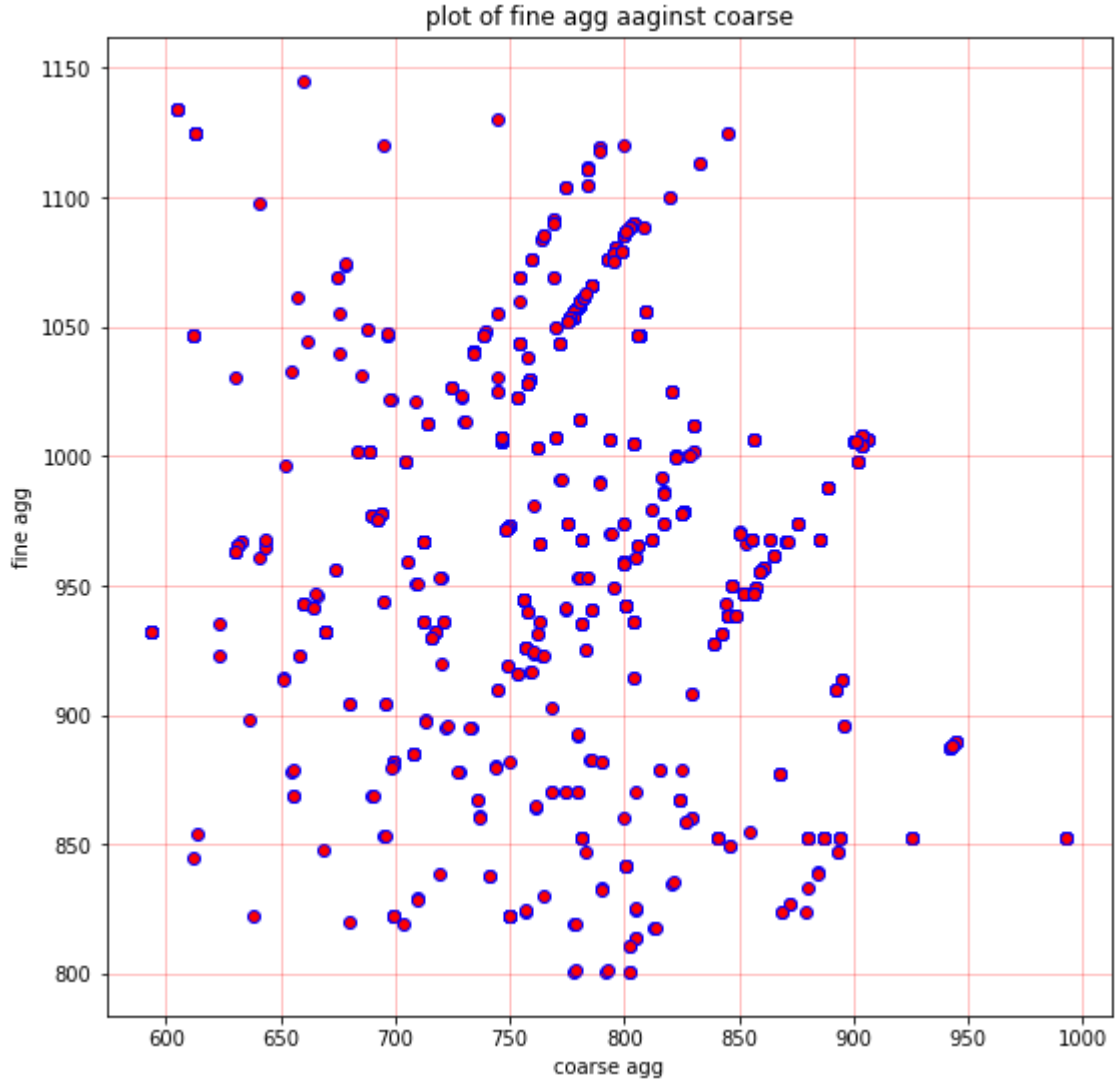


In [30]:
```
#violin plots cement against water
#plt.subplot(2,4,4)
sns.violinplot(y= 'fineagg', x='coarseagg', data = dset)
```

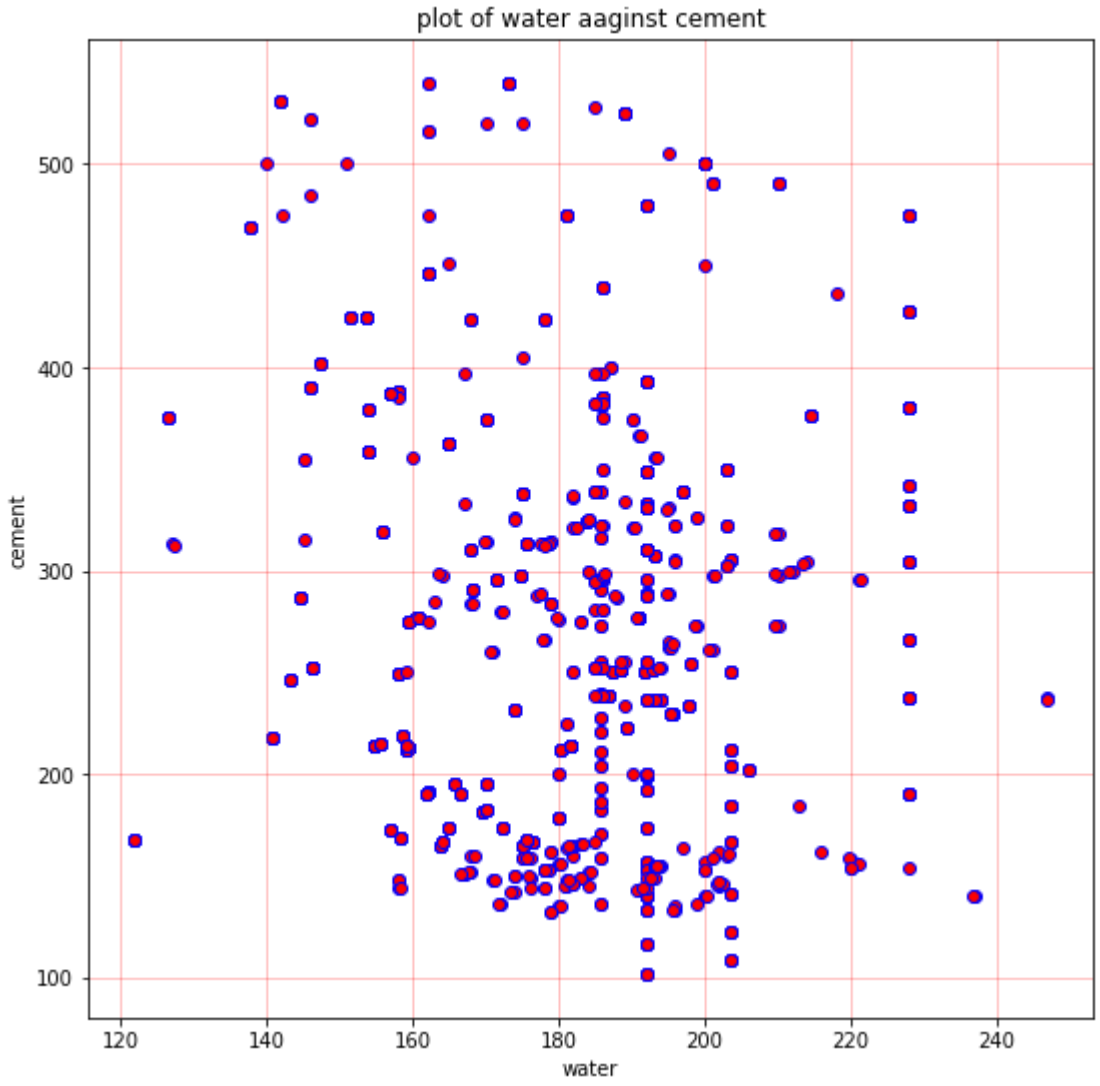
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0xb5a5fc6888>



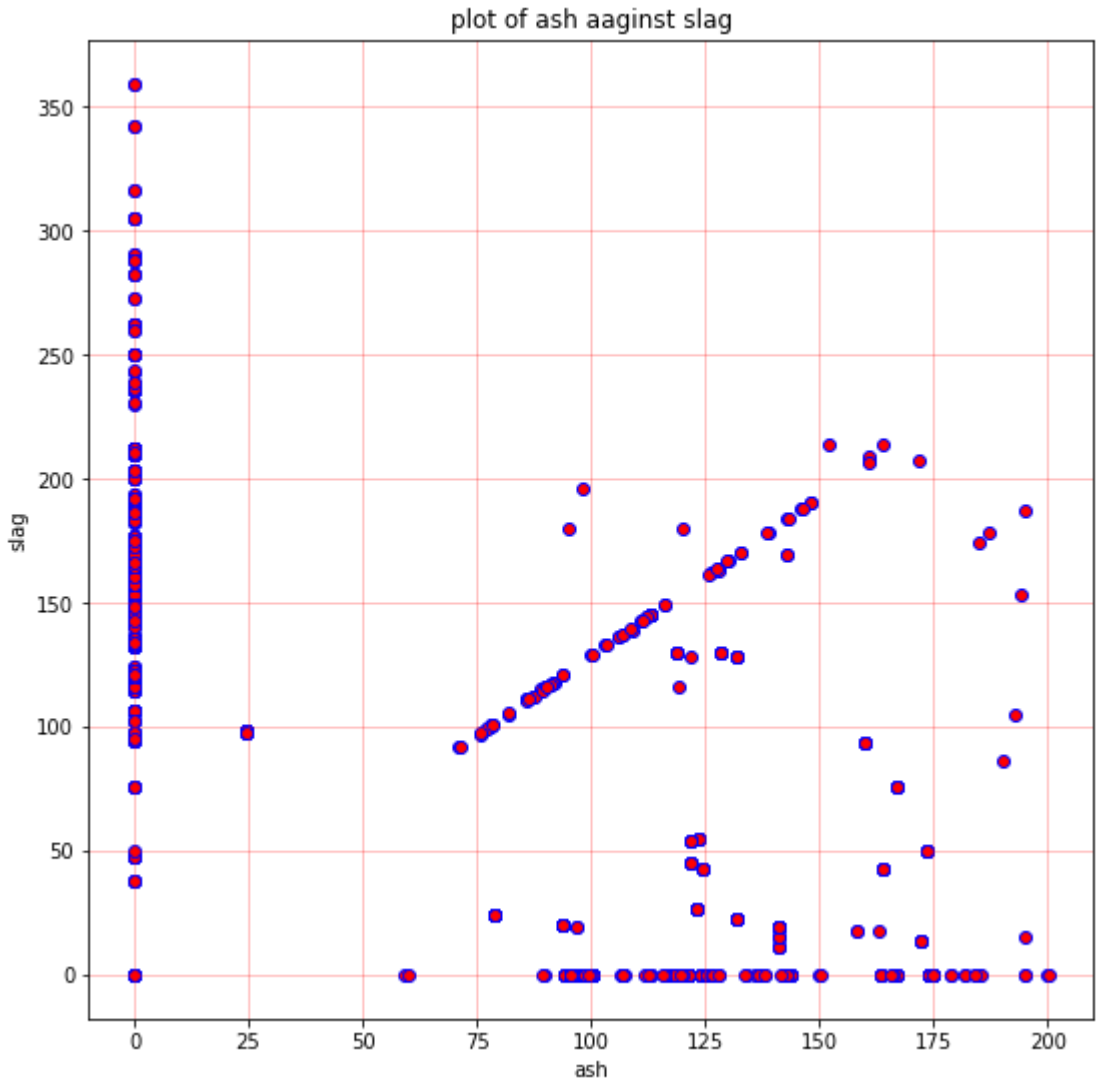
In [31]:
```
#fig =plt.figure(figsize=(13,9))
#scatter plot
g=plt.figure(figsize=(9,9))
p=g.add_subplot(111)
y=dset['coarseagg']
x=dset['fineagg']
plt.ylabel('fine agg')
plt.xlabel('coarse agg')
plt.title('plot of fine agg aaginst coarse')
plt.grid(True,alpha =0.3,color='r')
plt.scatter(x,y,edgecolor='b',cmap='virdis_r',facecolor='r')
plt.show()
#set grid
```



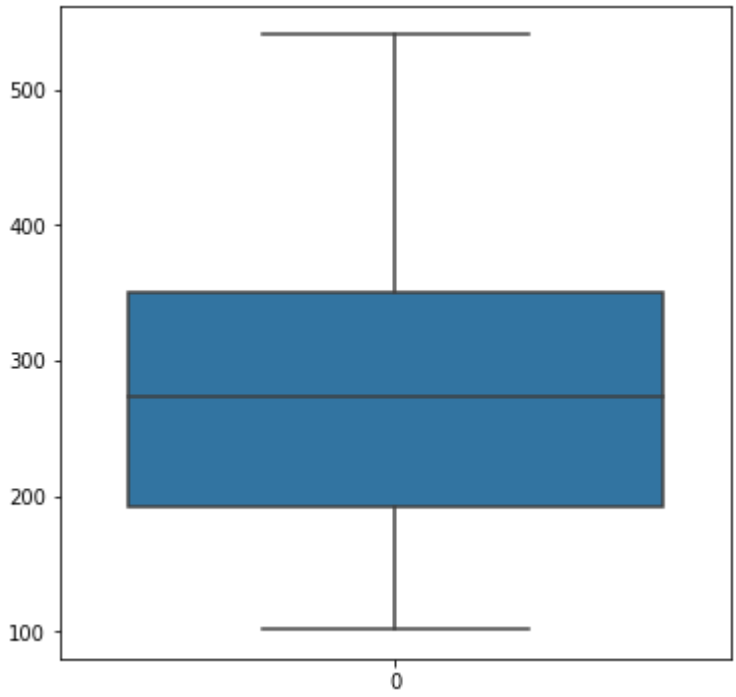
In [32]:
```
#fig =plt.figure(figsize=(9,9))
#scatter plot
g=plt.figure(figsize=(9,9))
p=g.add_subplot(111)
y=dset['cement']
x=dset['water']
plt.ylabel('cement')
plt.xlabel('water')
plt.title('plot of water aaginst cement')
plt.grid(True,alpha =0.3,color='r')
plt.scatter(x,y,edgecolor='b',cmap='virdis_r',facecolor='r')
plt.show()
#set grid
```



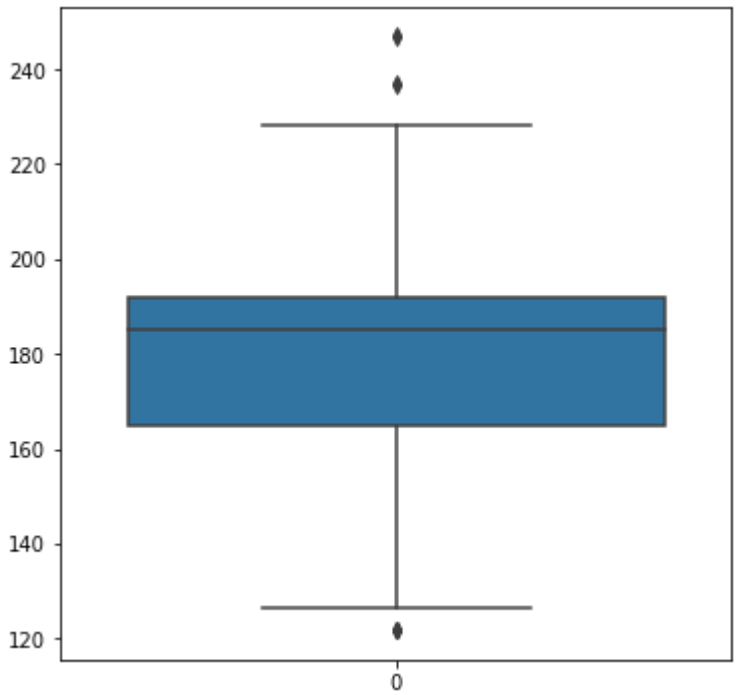
In [33]:
```
#fig =plt.figure(figsize=(13,9))
#scatter plot
g=plt.figure(figsize=(9,9))
p=g.add_subplot(111)
y=dset['slag']
x=dset['ash']
plt.ylabel('slag')
plt.xlabel('ash')
plt.title('plot of ash aaginst slag')
plt.grid(True,alpha =0.3,color='r')
plt.scatter(x,y,edgecolor='b',cmap='virdis_r',facecolor='r')
plt.show()
#set grid
```



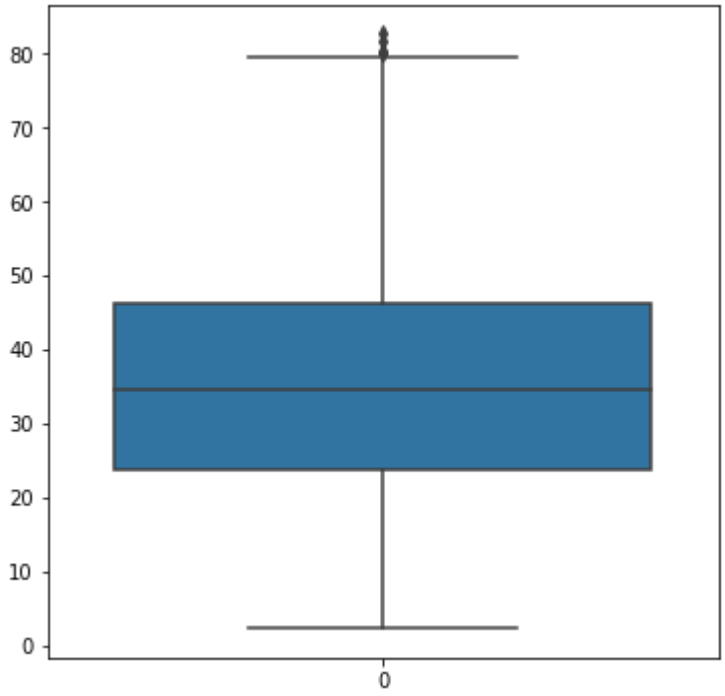
In [34]:
```
plt.figure(figsize=(6,6))
sns.boxplot(data=dset['cement'])
plt.show()
```



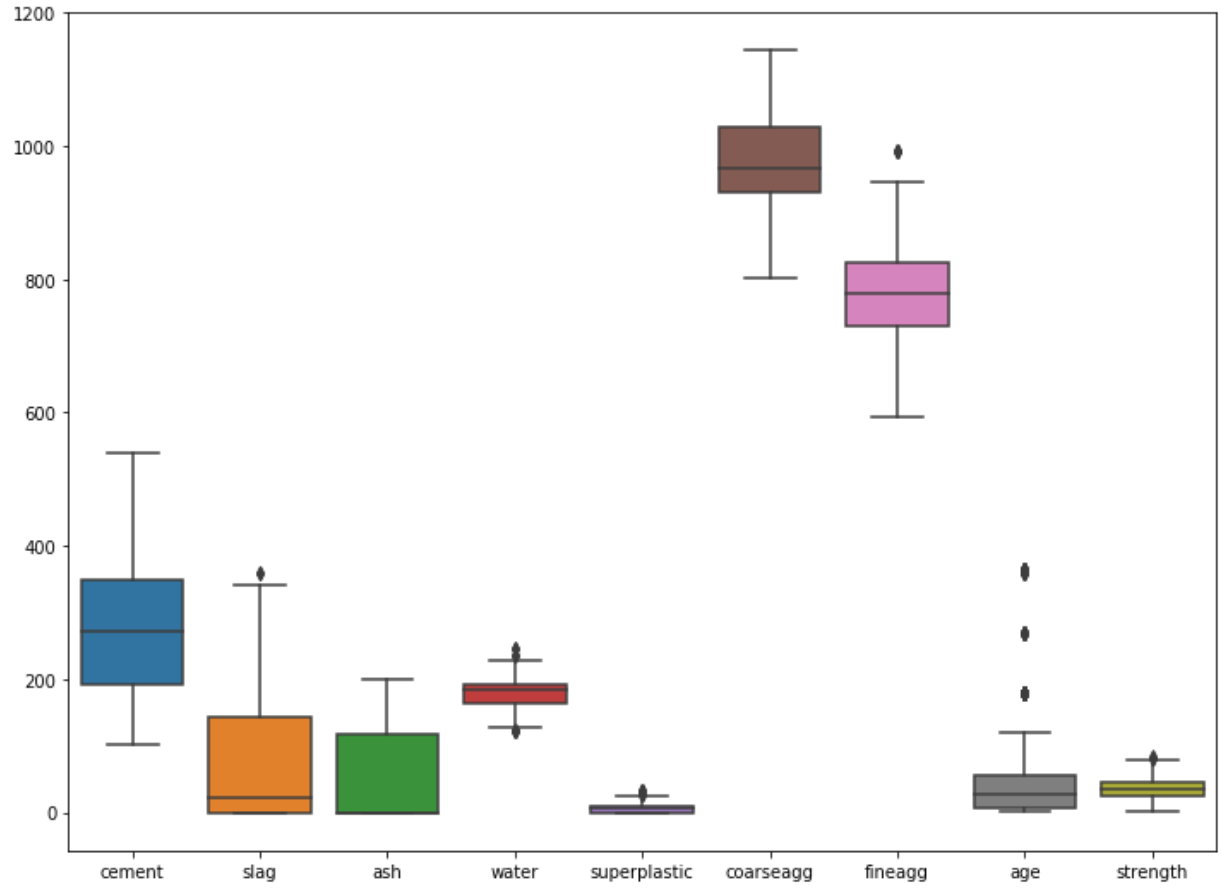
In [35]:
```
plt.figure(figsize=(6,6))
sns.boxplot(data=dset['water'])
plt.show()
```



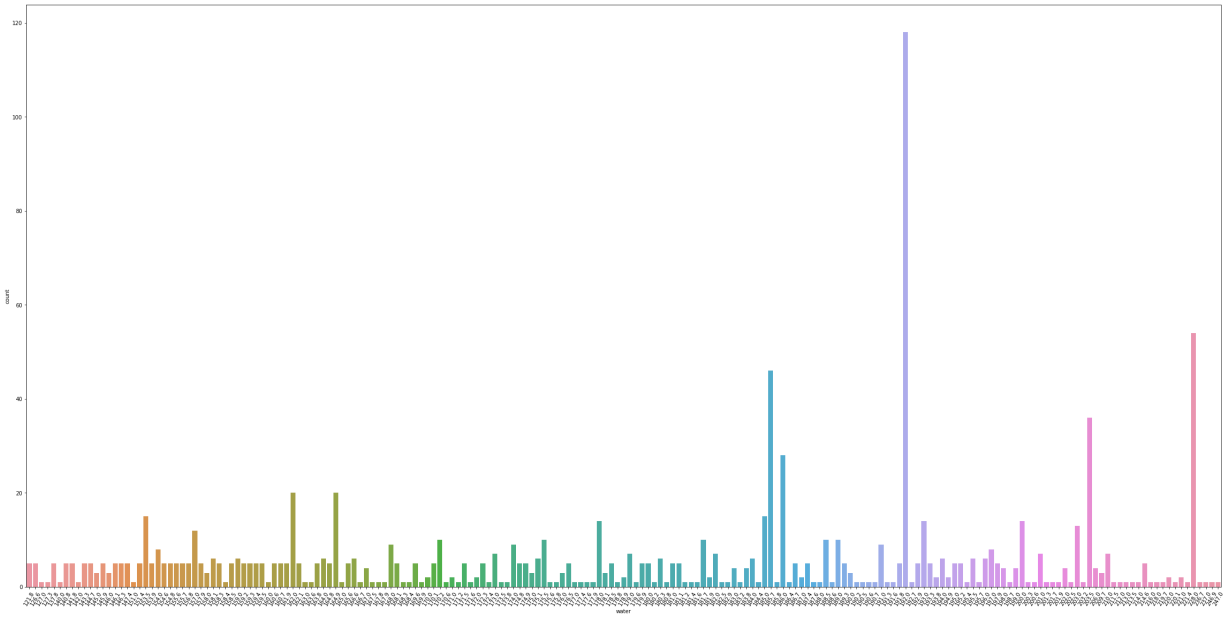
In [36]:
```
plt.figure(figsize=(6,6))
sns.boxplot(data=dset['strength'])
plt.show()
```



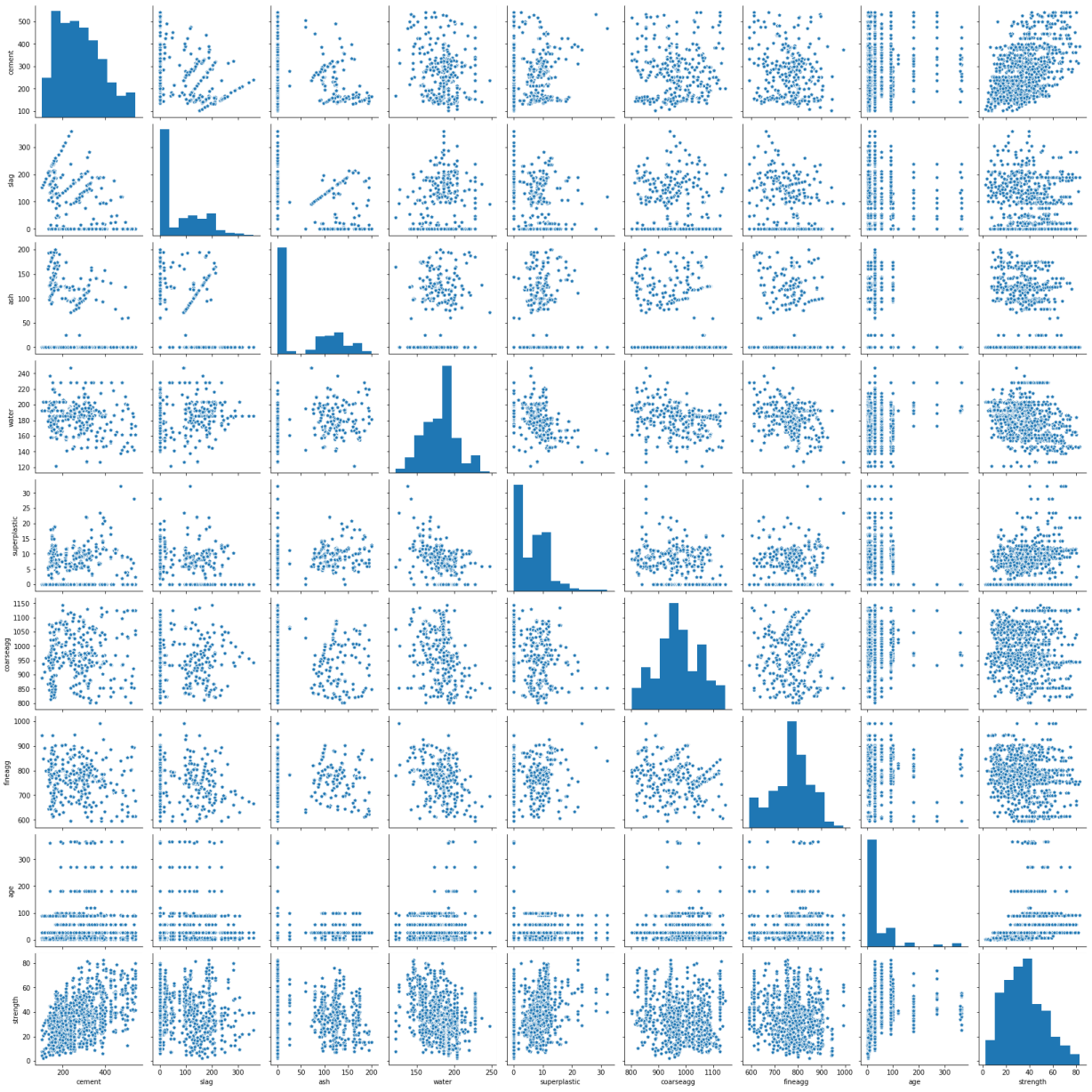
In [37]:
```
#boxplot of the the prevailed data
plt.figure(figsize=(12,9))
sns.boxplot(data=dset)
plt.show()
```



In [38]:
```
plt.figure(figsize=(40,20))
sns.countplot(x=dset['water'],data=dset)
plt.xticks(rotation=60)
plt.show()
```

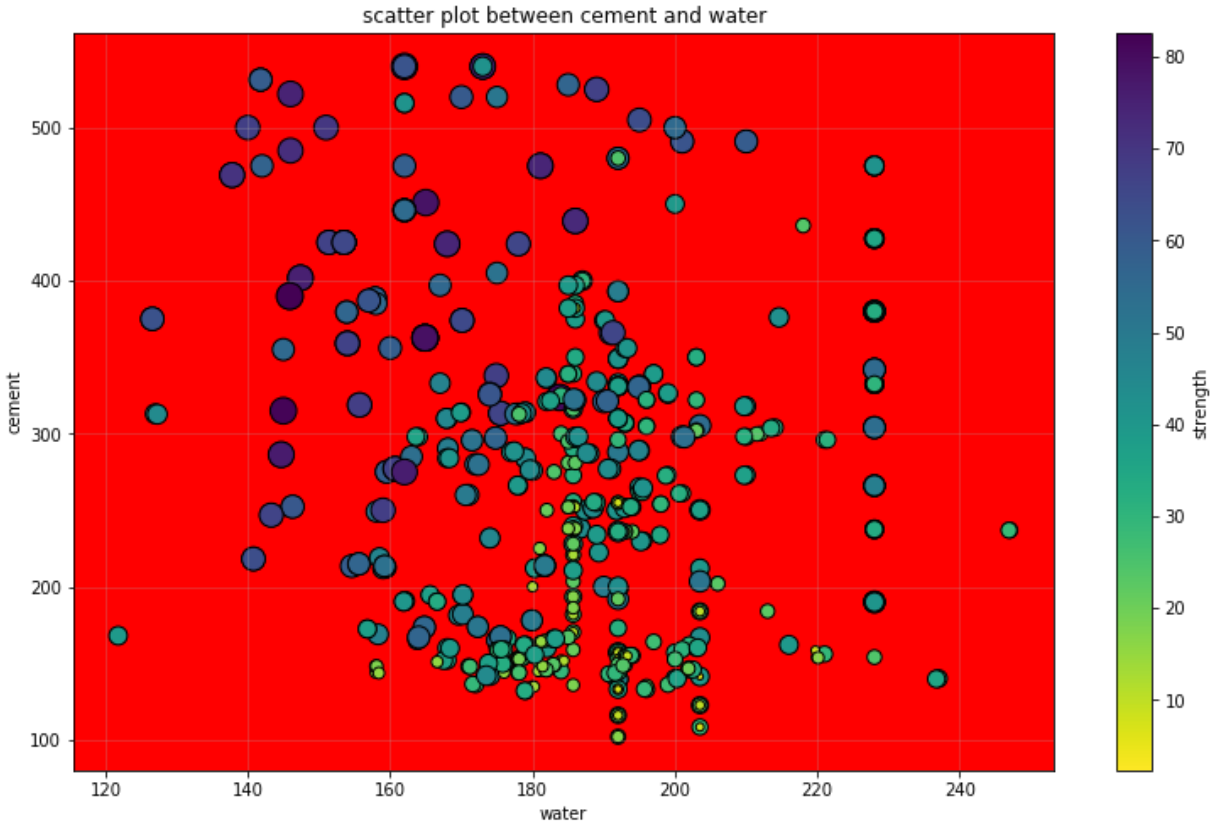


In [39]:
```
#pair plots betweeen variable
sns.pairplot(dset,markers="p")
plt.show()
```

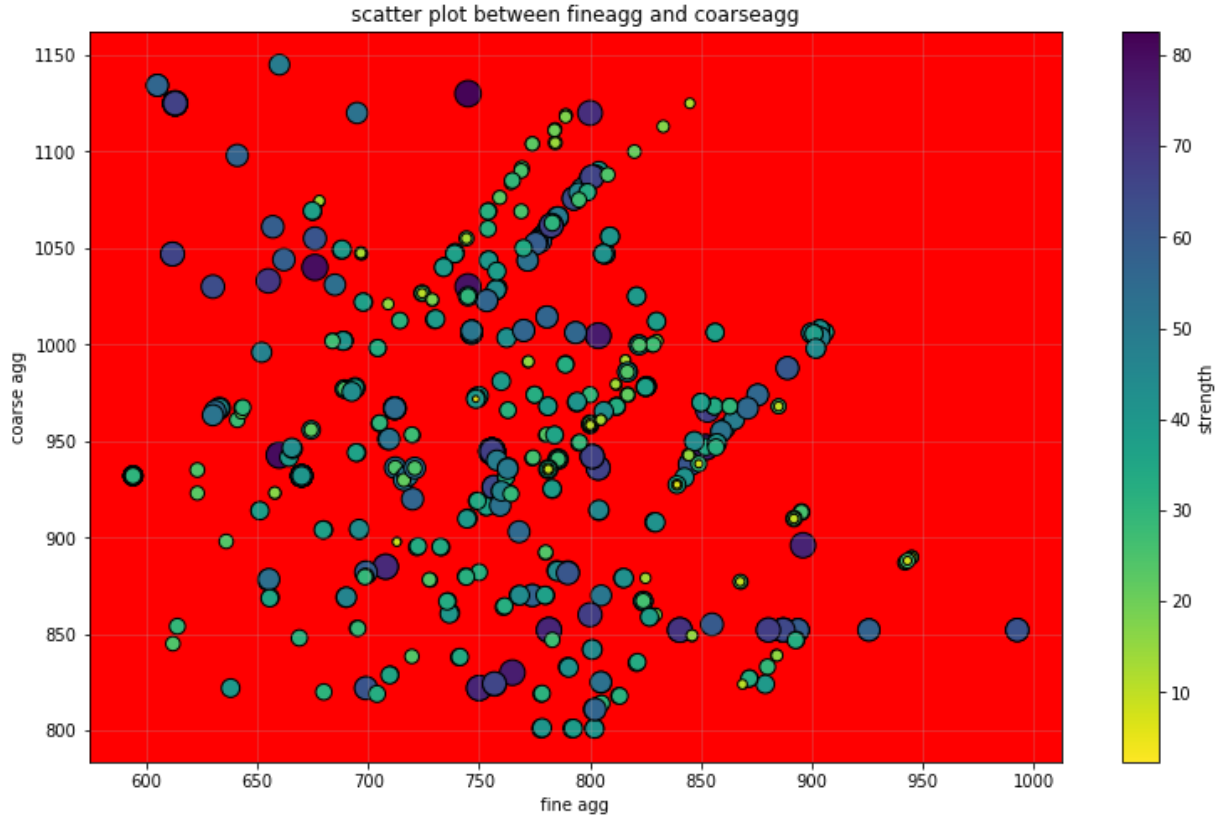


Scatter Plots

In [40]:
```
fig = plt.figure(figsize=(13,8))
ax = fig.add_subplot(111)
plt.scatter(dset["water"],dset["cement"],
            c=dset["strength"],s=dset["strength"]*3,
            linewidth=1,edgecolor="k",cmap="viridis_r")
ax.set_facecolor("r")
ax.set_xlabel("water")
ax.set_ylabel("cement")
lab = plt.colorbar()
lab.set_label("strength")
plt.title("scatter plot between cement and water")
plt.grid(True,alpha=.3)
plt.show()
```



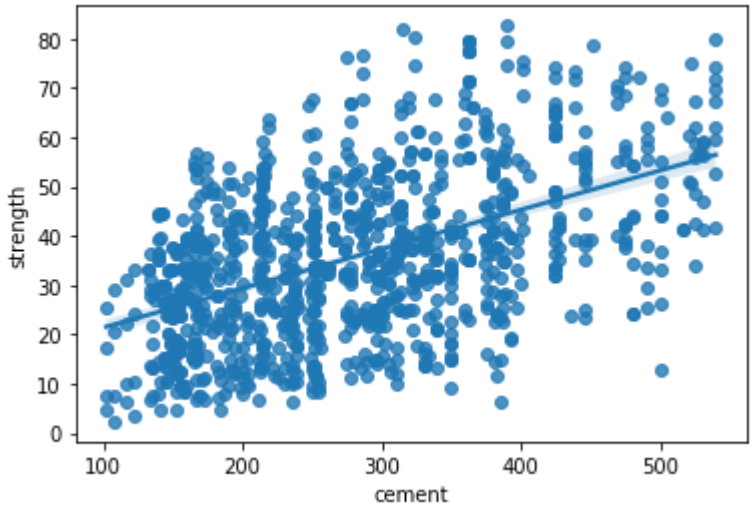
In [41]:
```
fig = plt.figure(figsize=(13,8))
ax = fig.add_subplot(111)
plt.scatter(dset["fineagg"],dset["coarseagg"],
            c=dset["strength"],s=dset["strength"]*3,
            linewidth=1,edgecolor="k",cmap="viridis_r")
ax.set_facecolor("r")
ax.set_xlabel("fine agg")
ax.set_ylabel("coarse agg")
lab = plt.colorbar()
lab.set_label("strength")
plt.title("scatter plot between fineagg and coarseagg")
plt.grid(True,alpha=.3)
plt.show()
```



Regression Plots

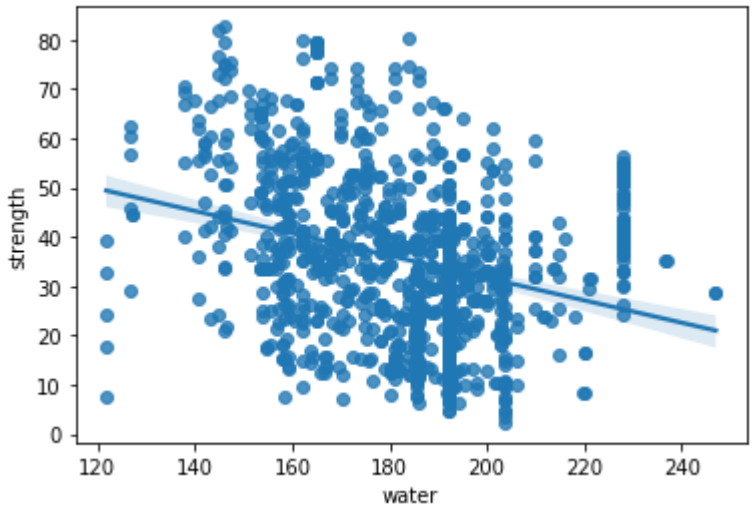
In [42]:
```
sns.regplot(x='cement',y='strength', data=dset)
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5aa4a7048>
```



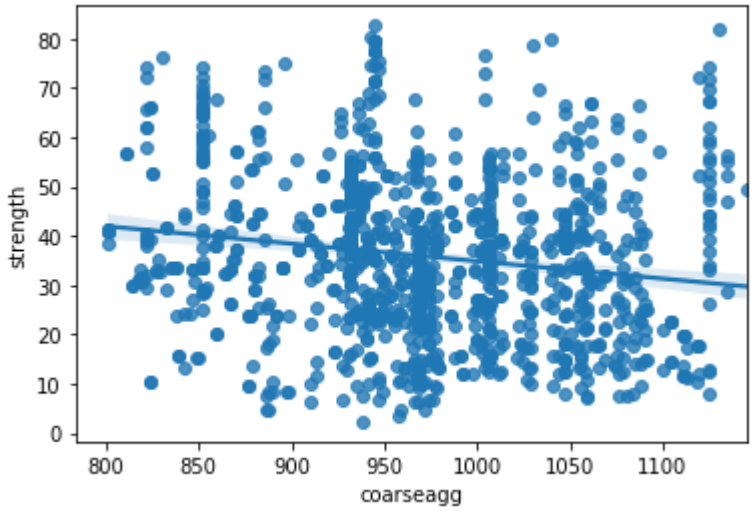
In [43]:
```
sns.regplot(x='water',y='strength', data=dset)
```

Out[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5ab61fc48>
```



In [44]:
```
sns.regplot(x='coarseagg',y='strength', data=dset)
```

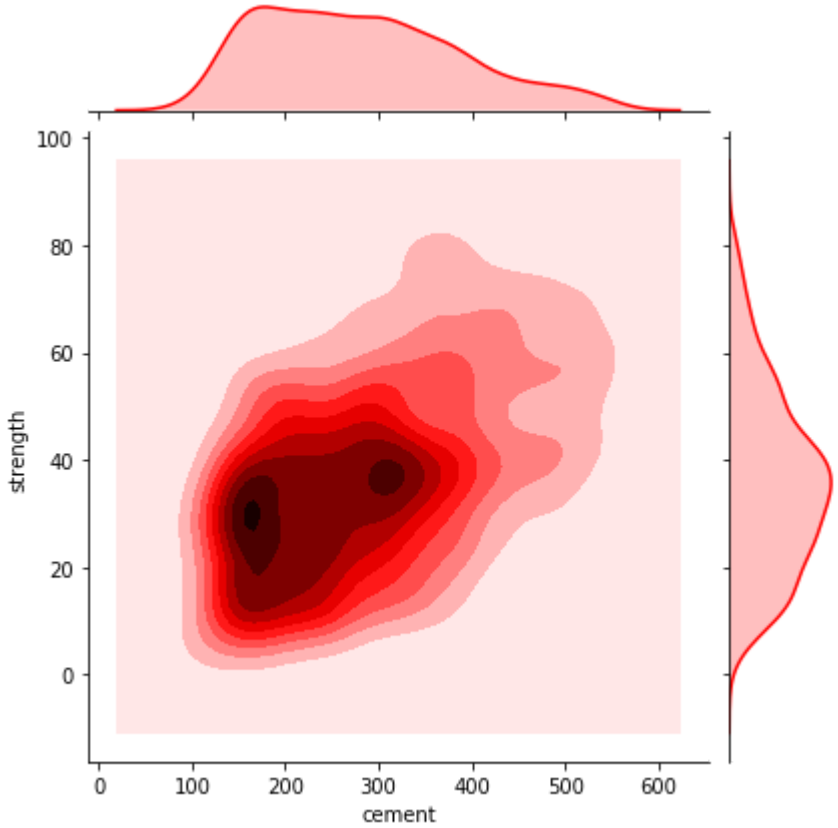
Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb5aafcd3c8>
```



Joint Plots

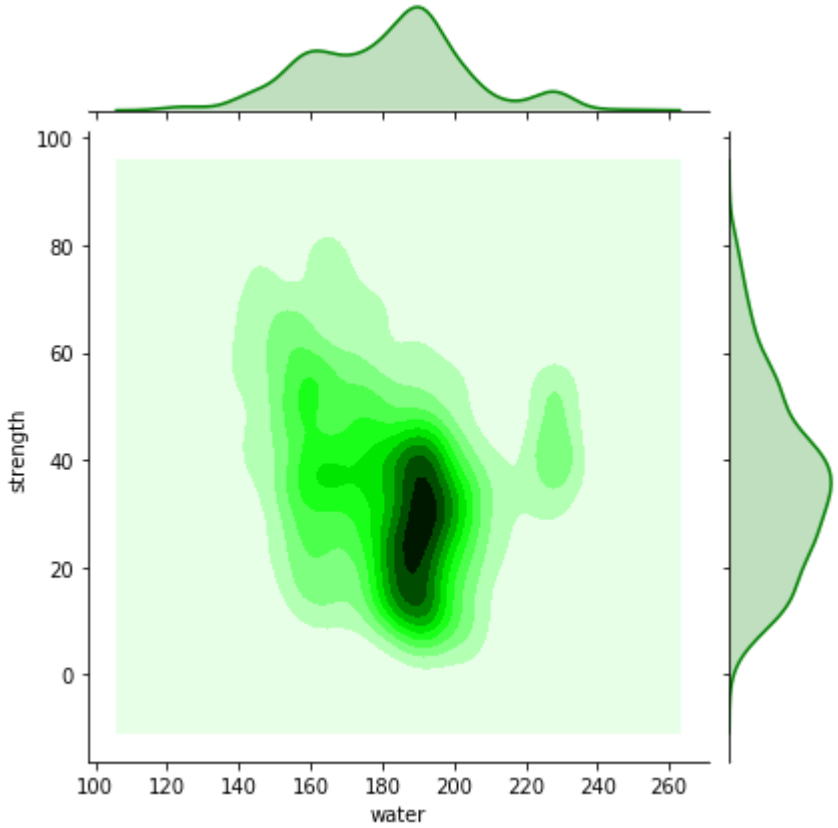
In [45]:
```
sns.jointplot(x='cement',y='strength',kind='kde',data=dset,color='r')
```

Out[45]:
```
<seaborn.axisgrid.JointGrid at 0xb5ab715b08>
```



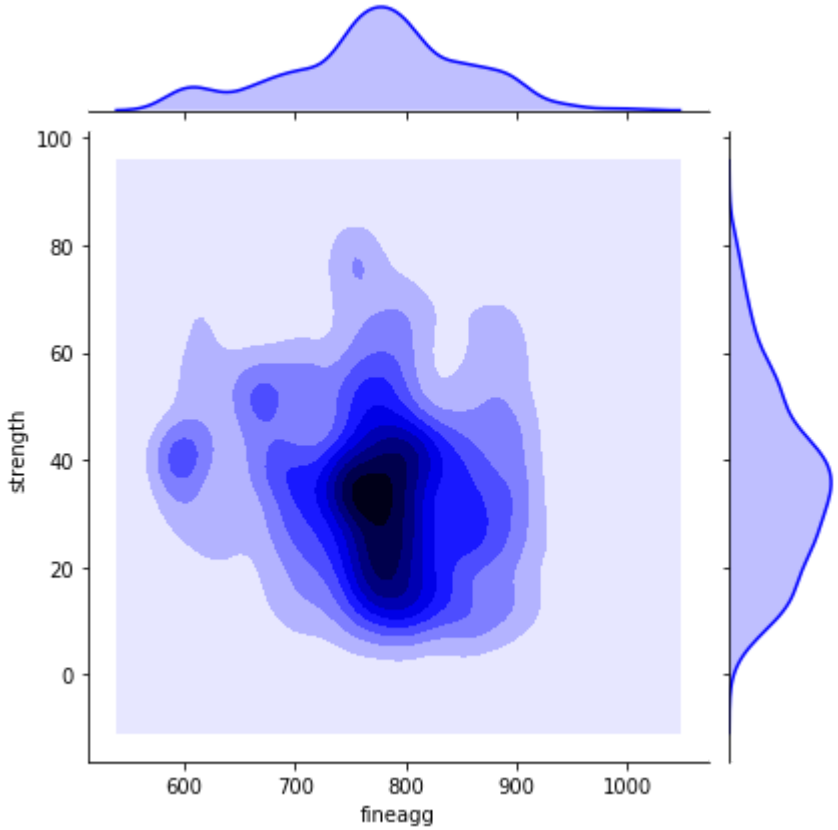
In [46]:
```
sns.jointplot(x='water',y='strength',kind='kde',data=dset,color='g')
```

Out[46]:
```
<seaborn.axisgrid.JointGrid at 0xb5ab705388>
```



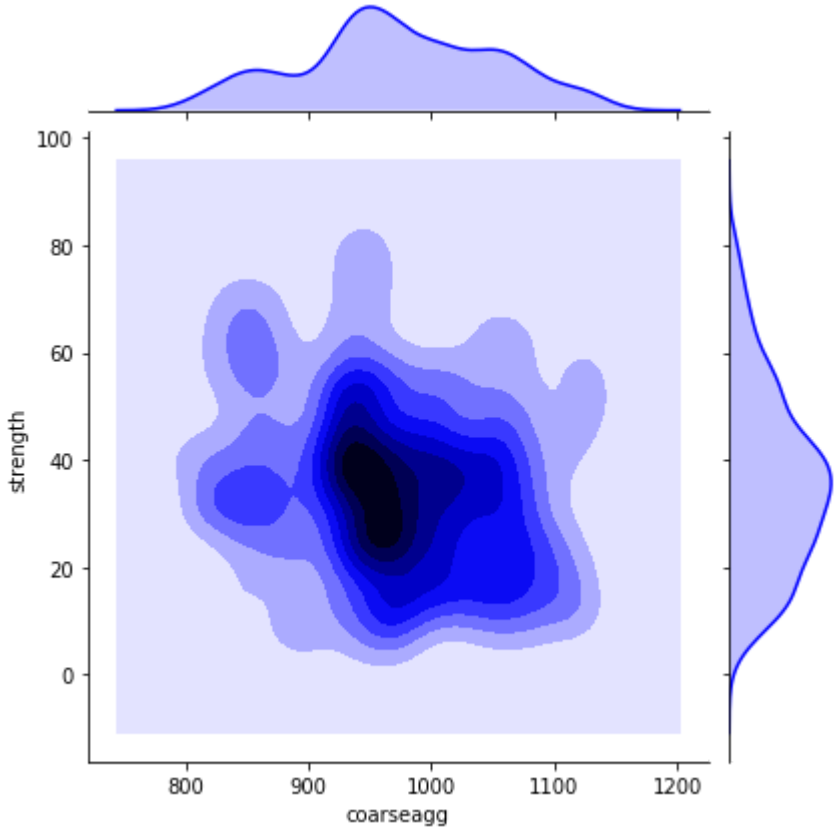
In [47]:
```
sns.jointplot(x='fineagg',y='strength',kind='kde',data=dset,color='blue')
```

Out[47]:
```
<seaborn.axisgrid.JointGrid at 0xb5ad04a488>
```

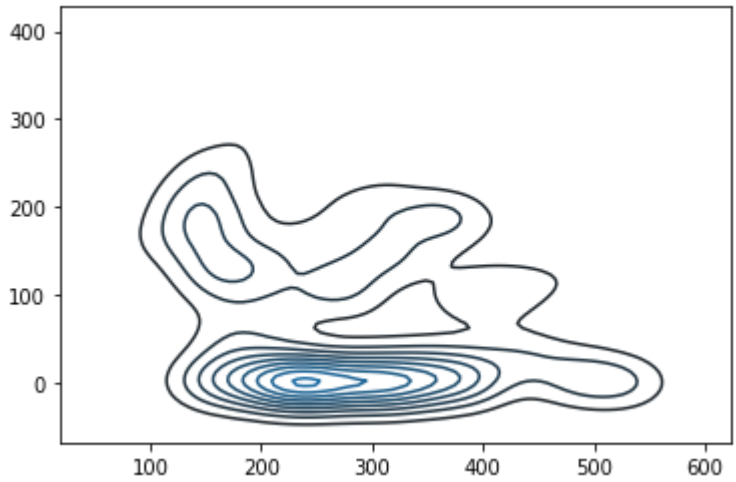


In [48]:
```
sns.jointplot(x='coarseagg',y='strength',kind='kde',data=dset,color='blue')
```

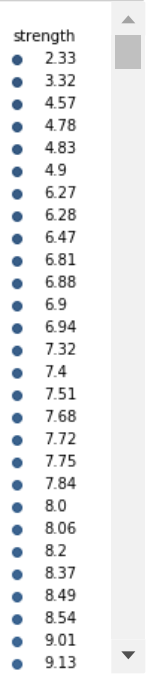
Out[48]:
```
<seaborn.axisgrid.JointGrid at 0xb5aa4bc488>
```



In [49]:
```
sns.kdeplot(dset);
```

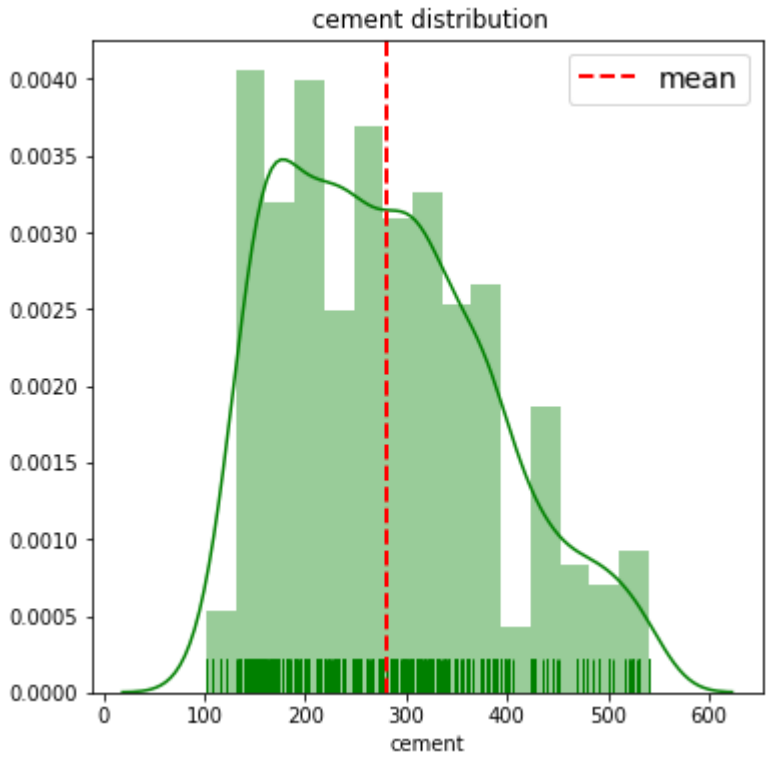


In [50]:
```
g = sns.PairGrid(dset, vars=['cement', 'water', 'slag', 'fineagg'],
                 hue='strength', palette='RdBu_r')
g.map(plt.scatter, alpha=0.8)
g.add_legend();
```

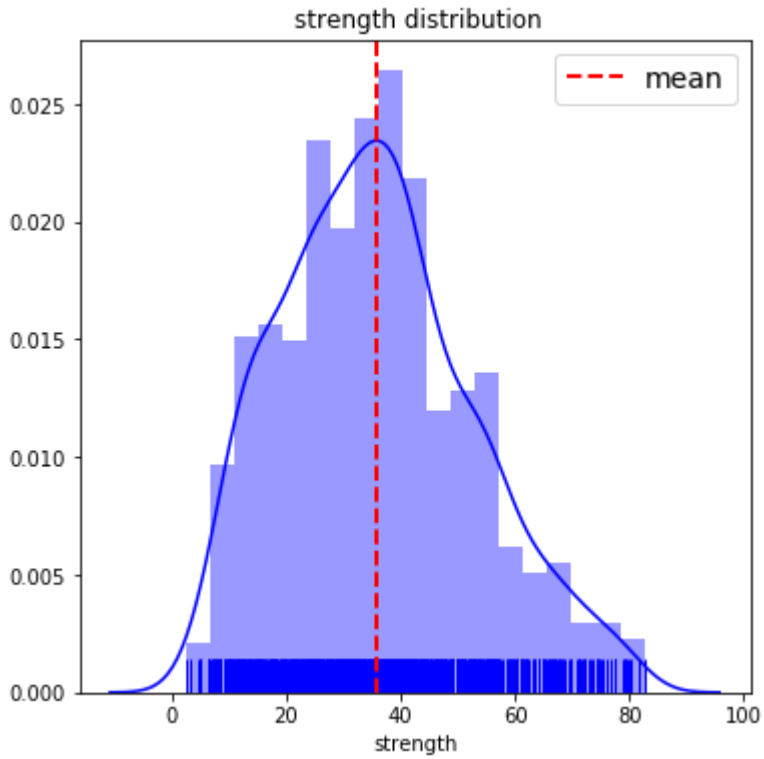


Distribution with mean

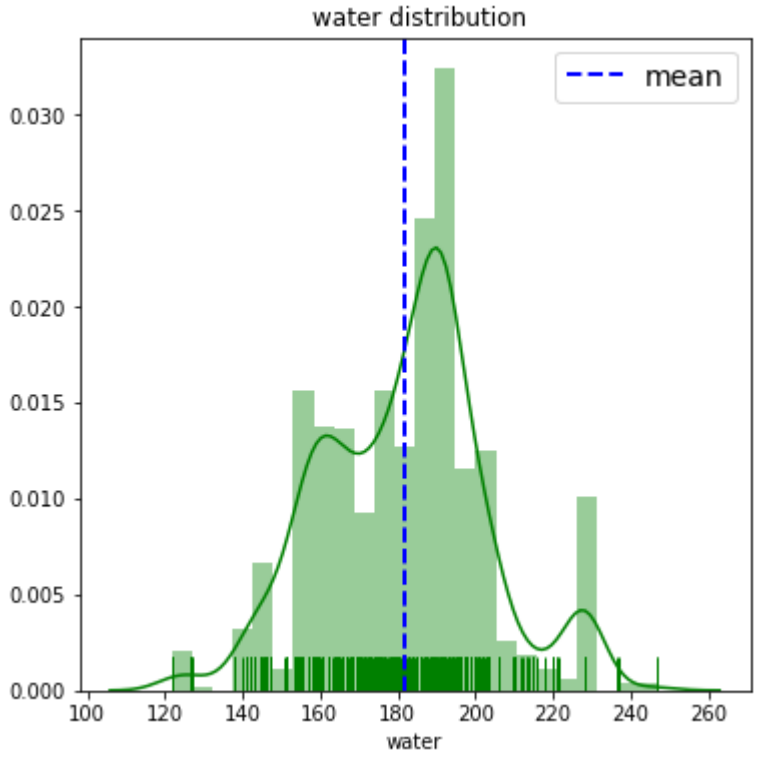
In [51]:
```
plt.figure(figsize=(6,6))
sns.distplot(dset["cement"],color="green",rug=True)
plt.axvline(dset["cement"].mean(),
            linestyle="dashed",color="red",
            label='mean',linewidth=2)
plt.legend(loc="best",prop={"size":14})
plt.title(" cement distribution")
plt.show()
```



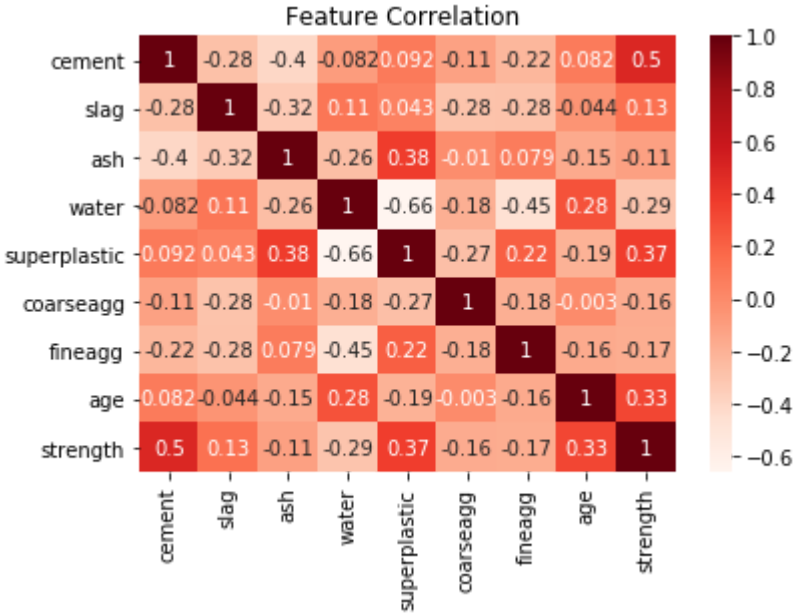
In [52]:
```
plt.figure(figsize=(6,6))
sns.distplot(dset["strength"],color="blue",rug=True)
plt.axvline(dset["strength"].mean(),
            linestyle="dashed",color="red",
            label='mean',linewidth=2)
plt.legend(loc="best",prop={"size":14})
plt.title(" strength distribution")
plt.show()
```



In [53]:
```
plt.figure(figsize=(6,6))
sns.distplot(dset["water"],color="green",rug=True)
plt.axvline(dset["water"].mean(),
            linestyle="dashed",color="blue",
            label='mean',linewidth=2)
plt.legend(loc="best",prop={"size":14})
plt.title(" water distribution")
plt.show()
```



In [54]:
```
corr = dset.corr()

sns.heatmap(corr, annot=True, cmap='Reds')

plt.title("Feature Correlation ")
plt.show()
```



THE SVM ALGORITHM

In [55]:
```
x = dset.iloc[:,0:-1].values # all columns but not the last which is strength
y = dset.iloc[:,-1].values # target value is last column
```

In [56]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state
```

In [57]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(824, 8)
(206, 8)
(824,)
(206,)
```

In [58]:
```
svm = SVR(kernel='rbf')
svm.fit(x_train,y_train,)
```

Out[58]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [59]:
```
svm.score(x_test,y_test)
```

Out[59]:
```
0.21646205043003333
```

Standard scaler

In [60]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [61]:
```
svm = SVR(kernel='rbf')
svm.fit(x_train,y_train)
```

Out[61]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [62]:
```
svm.score(x_test,y_test)
```

Out[62]:
```
0.6150828206262744
```

Hyperparameter tuning

In [63]:
```
param_grid = {'C':[1,10,100,200],
              'kernel':['rbf','poly','linear','sigmoid'],
              'degree':[1,2,4,6],
              'gamma':[0.01,0.1]}

grid=GridSearchCV(SVR(), param_grid=param_grid, cv=4)
grid.fit(x_train,y_train)

y_pred = grid.predict(x_test)

print("accuracy: {}".format(grid.score(x_test, y_test)))
print("Tuned Parameters: {}".format(grid.best_params_))
```

```
accuracy: 0.8811217153832156
Tuned Parameters: {'C': 200, 'degree': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

Hypothesis Testing

In [64]:
```
from scipy import stats
#testing for water and cement
dset[['water','cement']].describe()
ttest,pval = stats.ttest_rel(dset['water'], dset['cement'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
4.2205410769185945e-139
reject null hypothesis
```

In [65]:
```
#testing between Fineagg and Coarseagg
dset[['fineagg','coarseagg']].describe()
ttest,pval = stats.ttest_rel(dset['fineagg'], dset['coarseagg'])
print(pval)
if pval<0.05: #alpha value
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
5.862986131723774e-295
reject null hypothesis
```

In [66]:
```
dset[['age','strength']].describe()
ttest,pval = stats.ttest_rel(dset['age'], dset['strength'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
1.545311719208927e-07
reject null hypothesis
```

In [67]:
```
dset[['cement','strength']].describe()
ttest,pval = stats.ttest_rel(dset['cement'], dset['strength'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.0
reject null hypothesis
```

In [ ]: