Following is a machine learning model which can predict the strength of a mixture for given composition of ingredients like cement, slag, ash, water, superplastic, coarseagg, fineagg, age.

**Factors Affecting Strength of Concrete**

The initial factor that must be addressed when creating concrete is the raw materials that you wish to use to make said concrete. Concrete is formed of four parts; cement, water, coarse aggregate, and sand (or fine aggregate), and the quality of all these components will affect the strength of the concrete.

**Cement**: The quality of the cement used should be tested to make sure it is the appropriate type of cement for the desired role, though in terms of strength the most important aspects are the fineness of the cement, as the finer the cement is then the faster the mixture will hydrate, and consequently the quicker it will strengthen as it dries, and the manufacturing date as cement loses its strengthening capabilities over time.

**Slag**: As the amount of slag increases, the late-age compressive strength of concrete mixtures increases. However, after an optimum point, any further increase in slag does not improve the late-age compressive strength. This optimum replacement ratio of slag is a crucial factor for its efficient use in the concrete industry.

**Ash**: Ash/Fly ash use in concrete improves the workability of plastic concrete, and the strength and durability of hardened concrete. Fly ash use is also cost effective. When fly ash is added to concrete, the amount of portland cement may be reduced.

**Water**: Adding more water to the concrete increases workability but more water also increases the potential for segregation (settling of coarse aggregate particles), increased bleeding, drying shrinkage and cracking in addition to decreasing the strength and durability.

**Superplastic**: Superplasticizers (SPs), also known as high range water reducers, are additives used in making high strength concrete. Plasticizers are chemical compounds that enable the production of concrete with approximately 15% less water content. Superplasticizers allow reduction in water content by 30% or more.

**Coarse aggregate**: Compressive strength of a concrete increases with increase in coarse aggregate size. Coarse aggregate size 13.2 mm, 19 mm, 25 mm, and 37.5 mm gave average compressive strength of 21.26 N/mm2, 23.41 N/mm2, 23.66 N/mm2 and 24.31N/mm2 respectively.

**Fine** **aggregate**: The fine aggregate can generally contain less material passing 300 um and 150 um sieve because of the higher cement content. Proportionally, the amount of fine aggregate should also be somewhat less than that used for normal strength concrete.

**Age**: It increases with increase in age. The strength measured after days, months and years shows an increase. It takes 28 days for concrete to attain full strength.

Following is a sample of components of the data which is being used.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cement** | **Slag** | **Ash** | **Water** | **Superplastic** | **Coarseagg** | **Fineagg** | **Age** | **Strength** |
| 141.3 | 212 | 0 | 203.5 | 0 | 971.8 | 748.5 | 28 | 29.89 |
| 168.9 | 42.2 | 124.3 | 158.3 | 10.8 | 1080.8 | 796.2 | 14 | 23.51 |
| 250 | 0 | 95.7 | 187.4 | 5.5 | 956.9 | 861.2 | 28 | 29.22 |
| 266 | 114 | 0 | 228 | 0 | 932 | 670 | 28 | 45.85 |
| 154.8 | 183.4 | 0 | 193.3 | 9.1 | 1047.4 | 696.7 | 28 | 18.29 |

**Machine Learning model:**

**Step – 1: Importing necessary libraries**

# Dataframe manipulation and analysis libraries

import pandas as pd

import numpy as np

# Data visualization libraries

import matplotlib.pyplot as plt

import seaborn as sns

import scipy.stats as sci

# Libraries to filter warnings

import warnings

warnings.filterwarnings('ignore')

#Multicolinearity test package

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

#Data preparation libraries

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split,cross\_val\_score,GridSearchCV

#Model evaluation libraries

from sklearn.metrics import r2\_score, mean\_squared\_error

#Machine Learning models

from sklearn.linear\_model import LinearRegression, Lasso, Ridge

from sklearn.preprocessing import PolynomialFeatures

from sklearn.svm import SVR

from sklearn.tree import DecisionTreeRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor

from xgboost import XGBRegressor

import xgboost

# Feature decomposition library

from sklearn.decomposition import PCA

#Clustering Libraries

from sklearn.cluster import KMeans

#Recursive feature elemination library

from sklearn.feature\_selection import RFE

#Learning curve analysis library

from sklearn.model\_selection import learning\_curve

**Step 2: Exploratory Data Analysis – EDA**

|  |  |  |
| --- | --- | --- |
| Column | Non-Null Count | Dtype |
| Cement | 1030 non-null | float64 |
| Slag | 1030 non-null | float64 |
| Ash | 1030 non-null | float64 |
| Water | 1030 non-null | float64 |
| Superplastic | 1030 non-null | float64 |
| Coarseagg | 1030 non-null | float64 |
| Fineagg | 1030 non-null | float64 |
| Age | 1030 non-null | int64 |
| Strength | 1030 non-null | float64 |

**Analysis:**

* We have 1030 rows
* We have 9 columns
* There is no null values
* All values are in the form of continuous data

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **cement** | **slag** | **ash** | **water** | **superplastic** | **coarseagg** | **fineagg** | **age** | **strength** |
| **count** | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 |
| **mean** | 281.167864 | 73.89583 | 54.18835 | 181.5673 | 6.20466 | 972.918932 | 773.5805 | 45.66214 | 35.81796 |
| **std** | 104.506364 | 86.27934 | 63.997 | 21.35422 | 5.973841 | 77.753954 | 80.17598 | 63.16991 | 16.70574 |
| **min** | 102 | 0 | 0 | 121.8 | 0 | 801 | 594 | 1 | 2.33 |
| **25%** | 192.375 | 0 | 0 | 164.9 | 0 | 932 | 730.95 | 7 | 23.71 |
| **50%** | 272.9 | 22 | 0 | 185 | 6.4 | 968 | 779.5 | 28 | 34.445 |
| **75%** | 350 | 142.95 | 118.3 | 192 | 10.2 | 1029.4 | 824 | 56 | 46.135 |
| **max** | 540 | 359.4 | 200.1 | 247 | 32.2 | 1145 | 992.6 | 365 | 82.6 |

**Analysis**

* Cement, slag, ash, age tend to have outliers because the mean and median are not same
* For ash, slag and superplastic there is no chance of outliers in the lower whisker region as min and Q1 are same
* Cement, slag, ash, superplastic and age might have outliers in higher whisker region.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **cement** | **slag** | **ash** | **water** | **superplastic** | **coarseagg** | **fineagg** | **age** | **strength** |
| **datatype** | float64 | float64 | float64 | float64 | float64 | float64 | float64 | int64 | float64 |
| **count** | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 | 1030 |
| **min** | 102 | 0 | 0 | 121.8 | 0 | 801 | 594 | 1 | 2.33 |
| **Q1** | 192.375 | 0 | 0 | 164.9 | 0 | 932 | 730.95 | 7 | 23.71 |
| **Q2** | 272.9 | 22 | 0 | 185 | 6.4 | 968 | 779.5 | 28 | 34.445 |
| **Q3** | 350 | 142.95 | 118.3 | 192 | 10.2 | 1029.4 | 824 | 56 | 46.135 |
| **max** | 540 | 359.4 | 200.1 | 247 | 32.2 | 1145 | 992.6 | 365 | 82.6 |
| **mean** | 281.17 | 73.9 | 54.19 | 181.57 | 6.2 | 972.92 | 773.58 | 45.66 | 35.82 |
| **stddev** | 104.51 | 86.28 | 64 | 21.35 | 5.97 | 77.75 | 80.18 | 63.17 | 16.71 |
| **skew** | 0.51 | 0.8 | 0.54 | 0.07 | 0.91 | -0.04 | -0.25 | 3.27 | 0.42 |
| **kurt** | -0.52 | -0.51 | -1.33 | 0.12 | 1.41 | -0.6 | -0.1 | 12.17 | -0.31 |
| **range** | 438 | 359.4 | 200.1 | 125.2 | 32.2 | 344 | 398.6 | 364 | 80.27 |
| **IQR** | 157.625 | 142.95 | 118.3 | 27.1 | 10.2 | 97.4 | 93.05 | 49 | 22.425 |
| **Skew comment** | Moderately positively skewed | Moderately positively skewed | Moderately positively skewed | Approximately normally distributed(+ve) | Moderately positively skewed | Approximately normally distributed(-ve) | Approximately normally distributed(-ve) | Highly positively skewed | Approximately normally distributed(+ve) |
| **Kurt comment** | Moderately platykurtic | Moderately platykurtic | Highly platykurtic | Mesokurtic curve | Highly leptokurtic | Moderately platykurtic | Mesokurtic curve | Highly leptokurtic | Mesokurtic curve |
| **outlier comment** | No outliers | Have outliers | No outliers | Have outliers | Have outliers | No outliers | Have outliers | Have outliers | Have outliers |

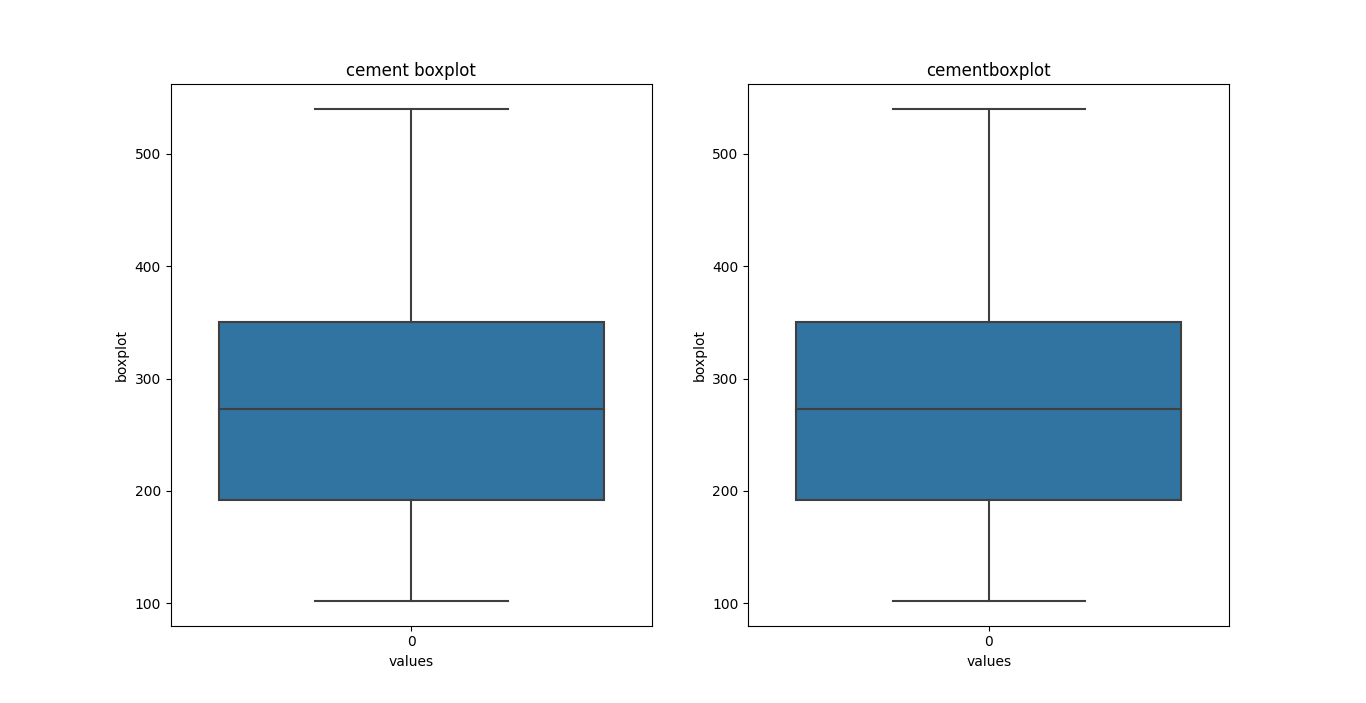
**Analysis**:

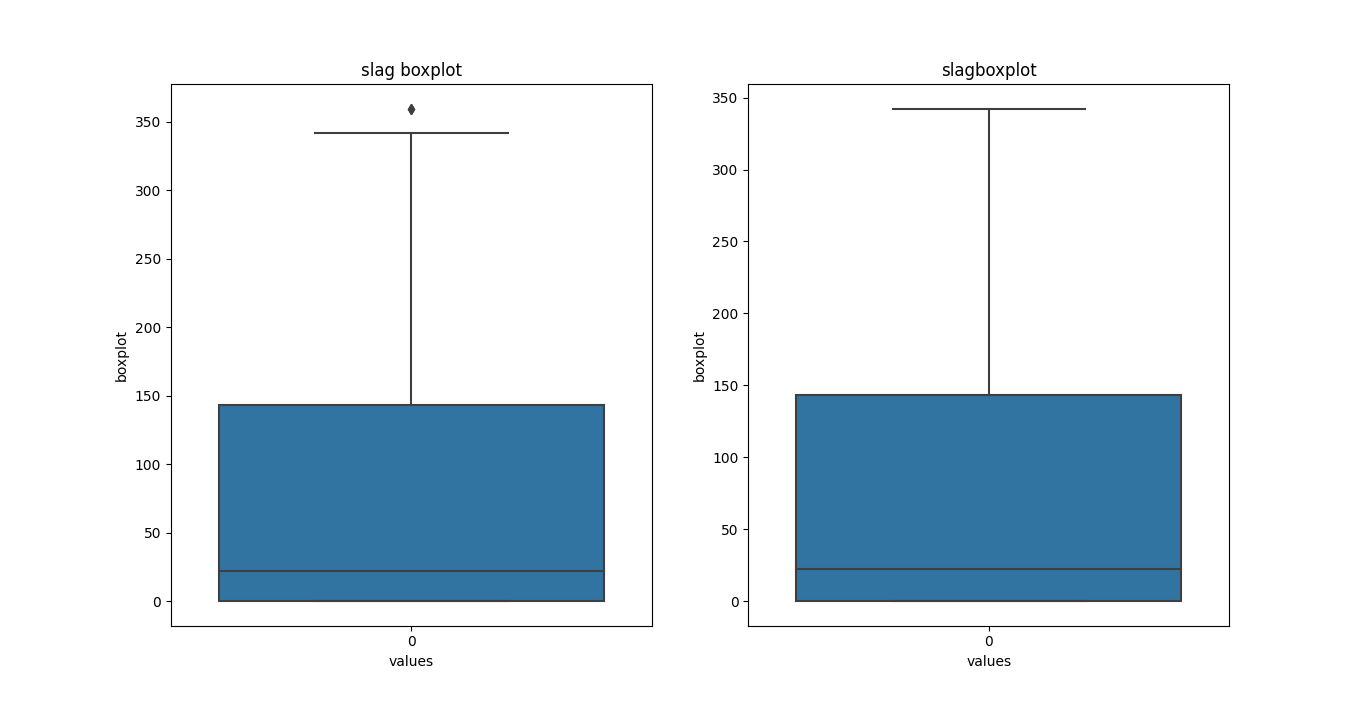
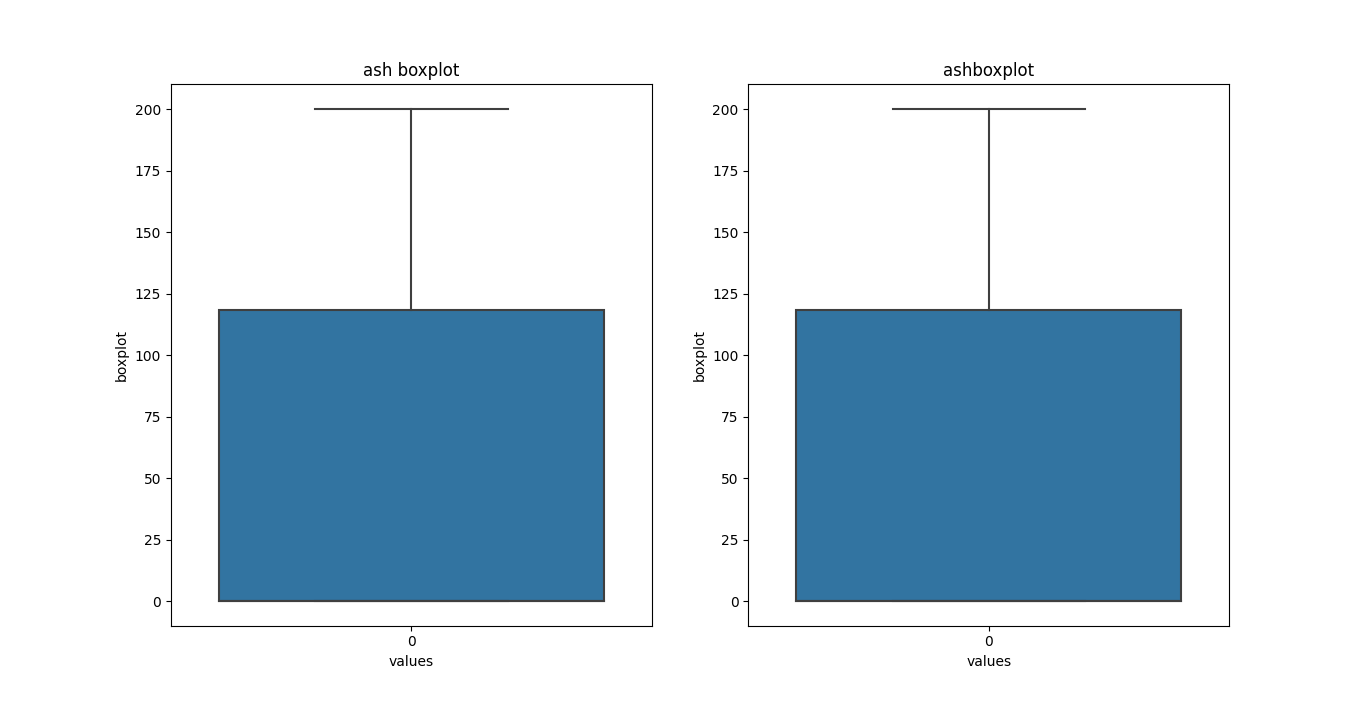
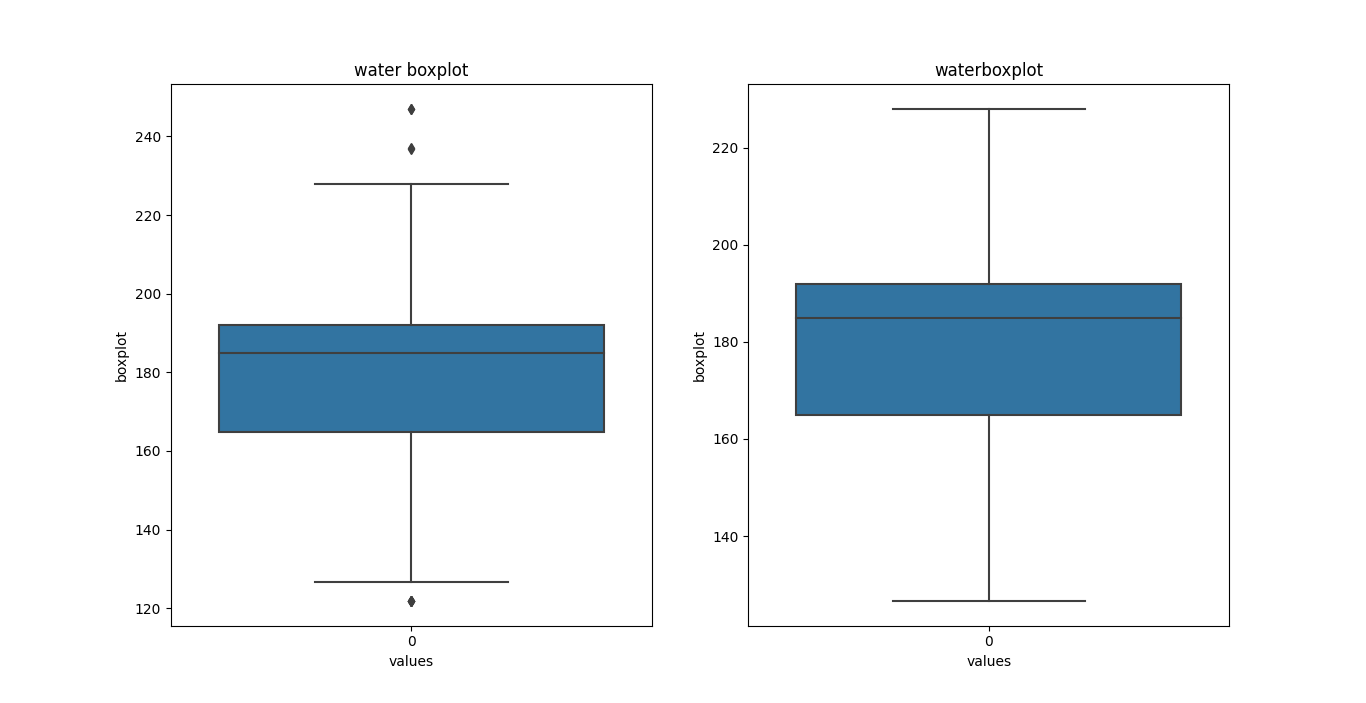
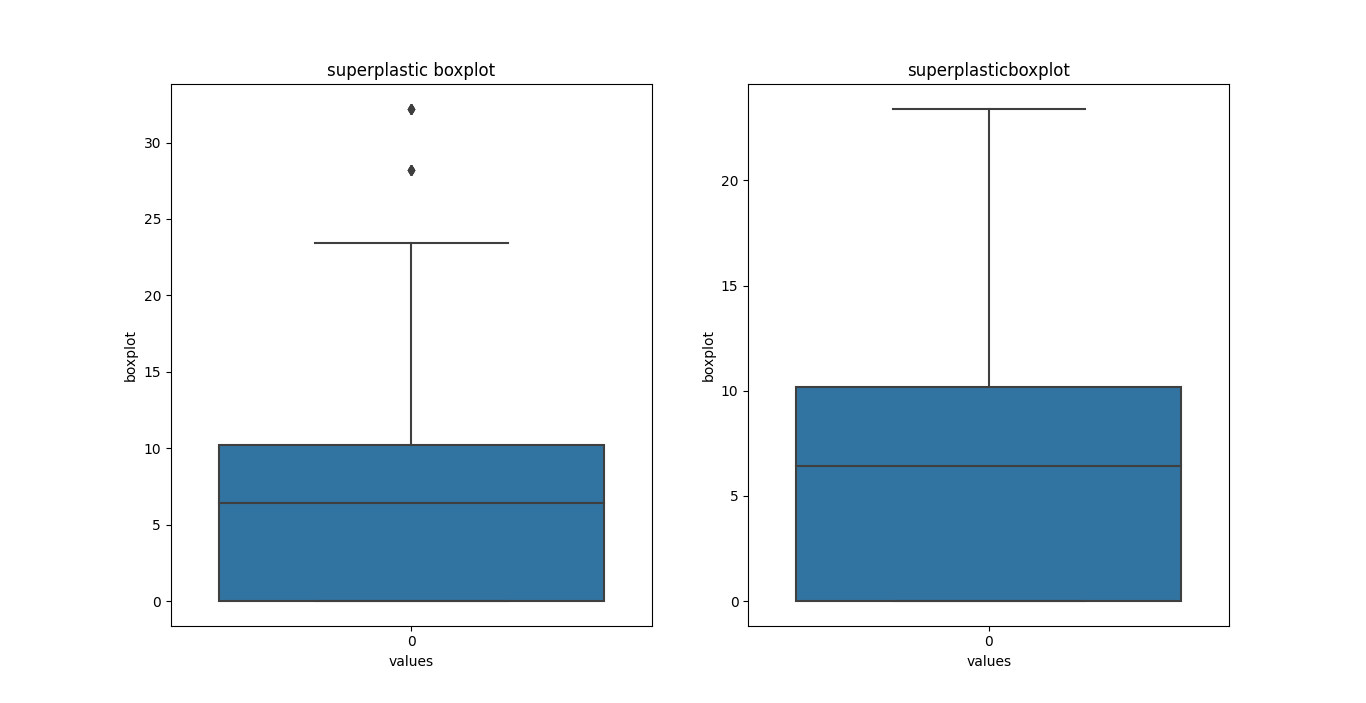
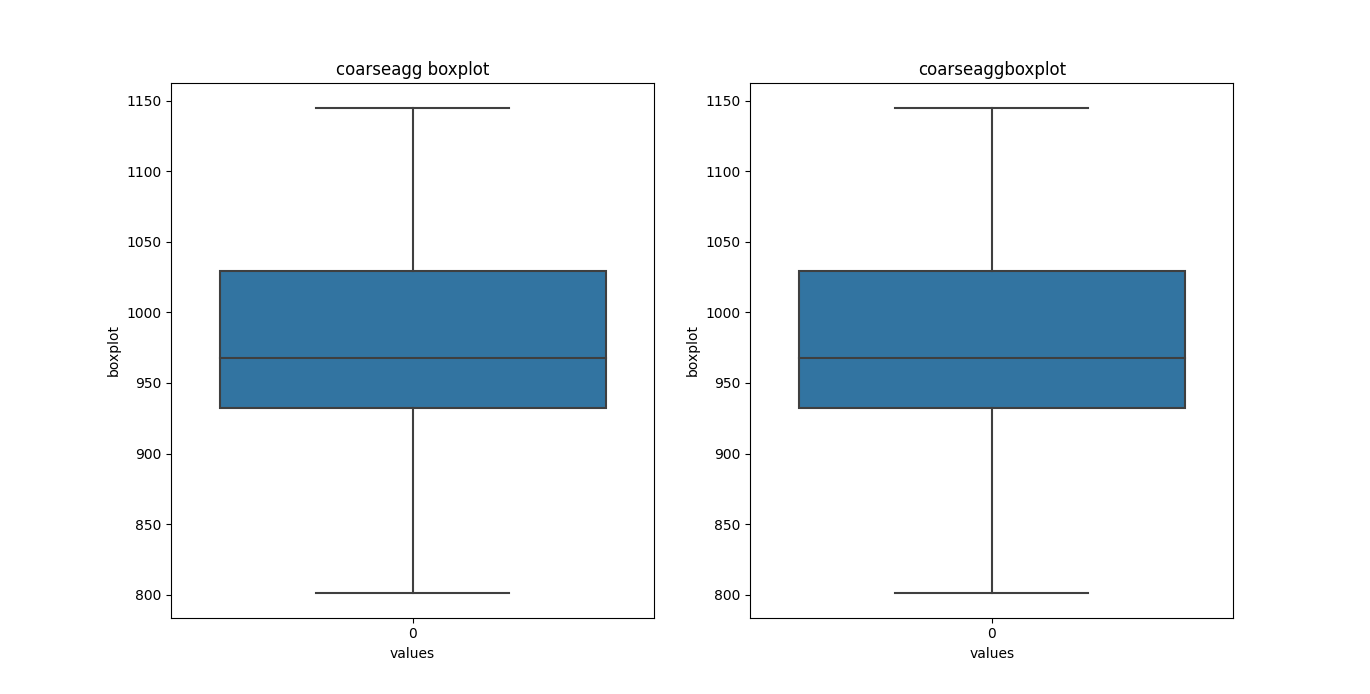
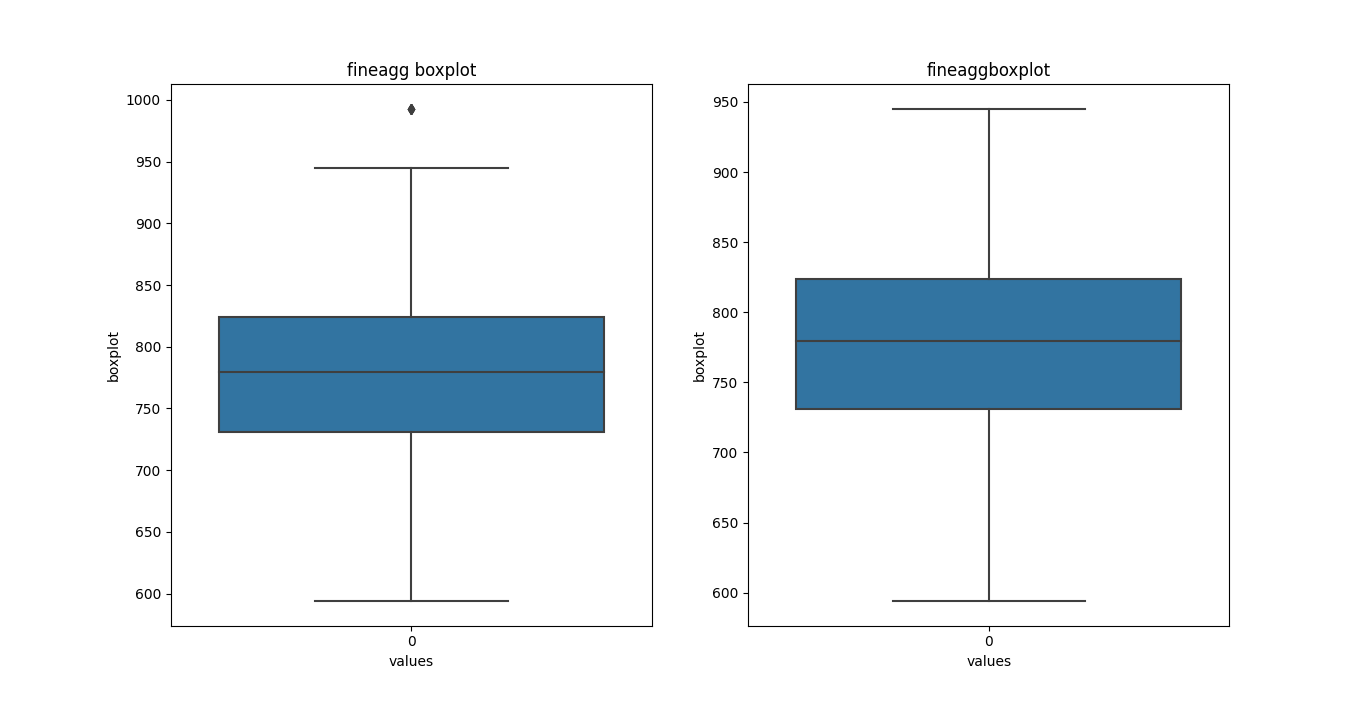
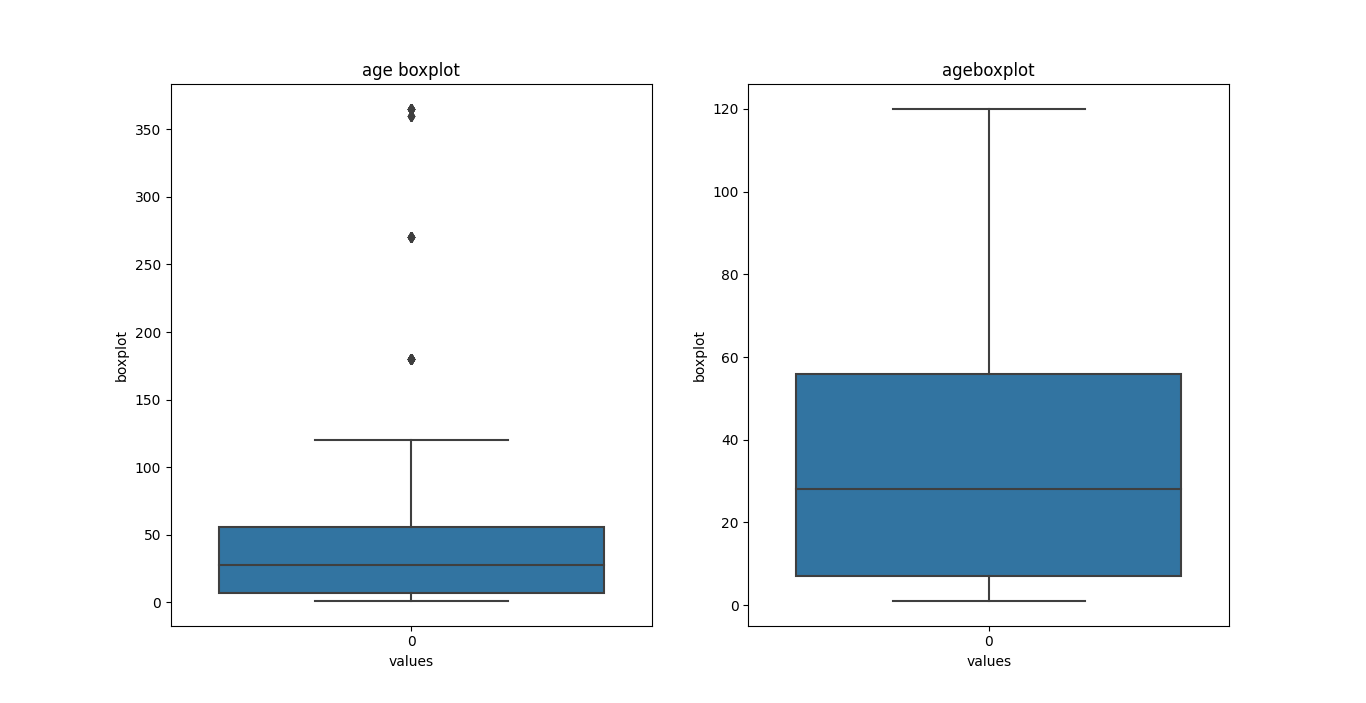
* \* We have outliers in slag, water, superplastic, fineagg and strength.

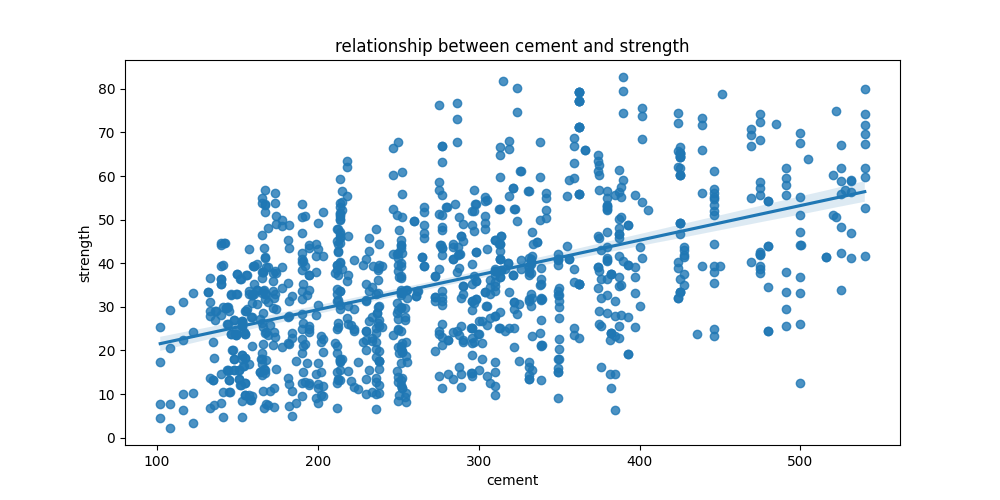
**Step 3: Outliers treatment**

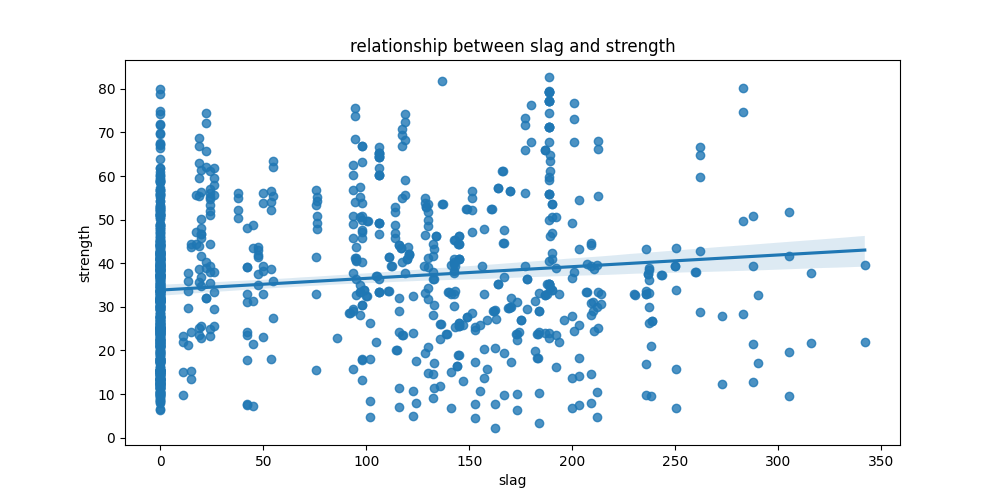
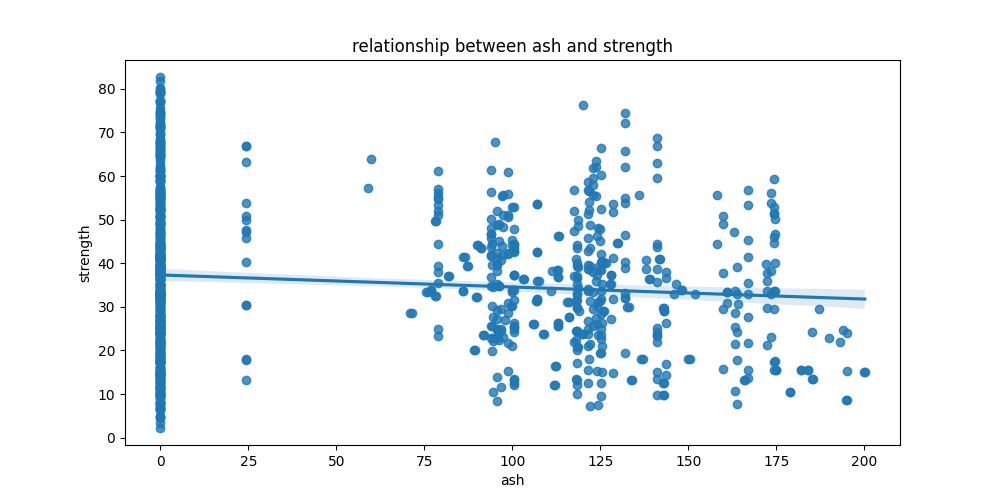
**Checking for outliers using boxplot and replacing them**

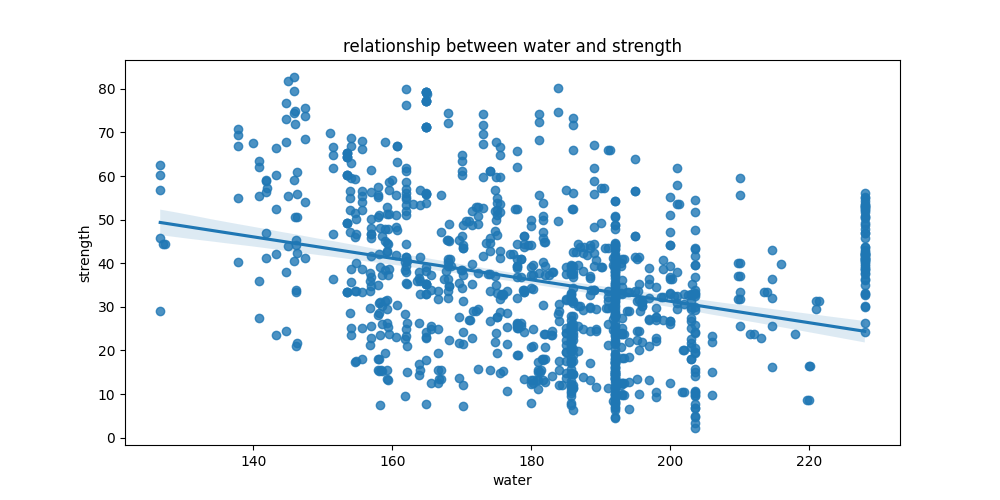
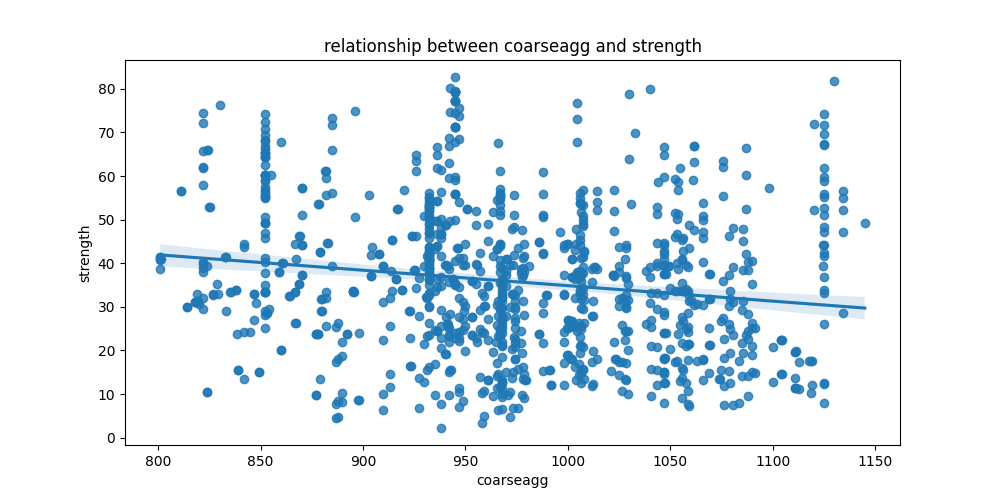
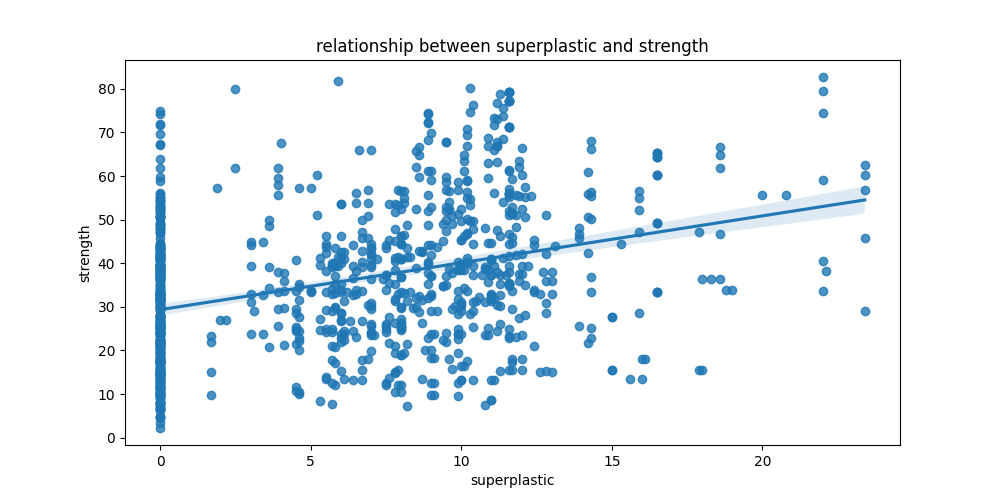
1. **Feature cement does not have any outliers**

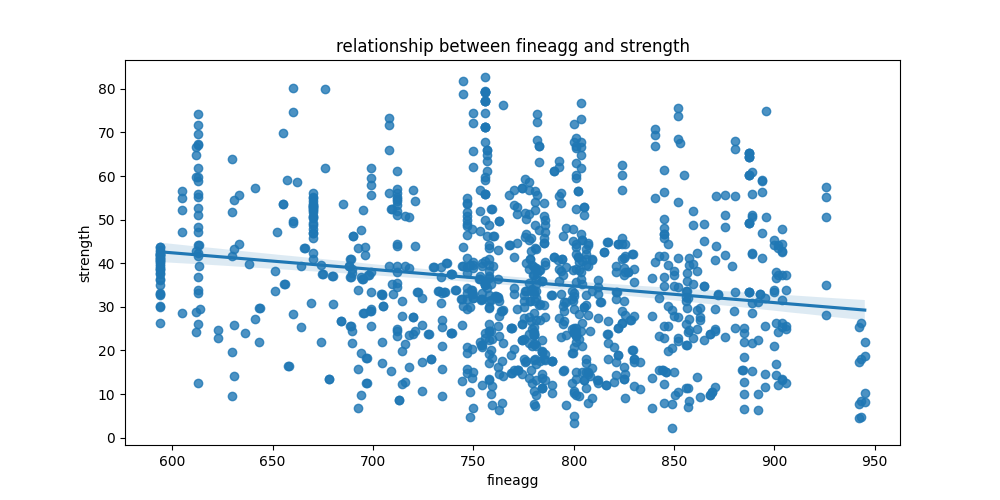


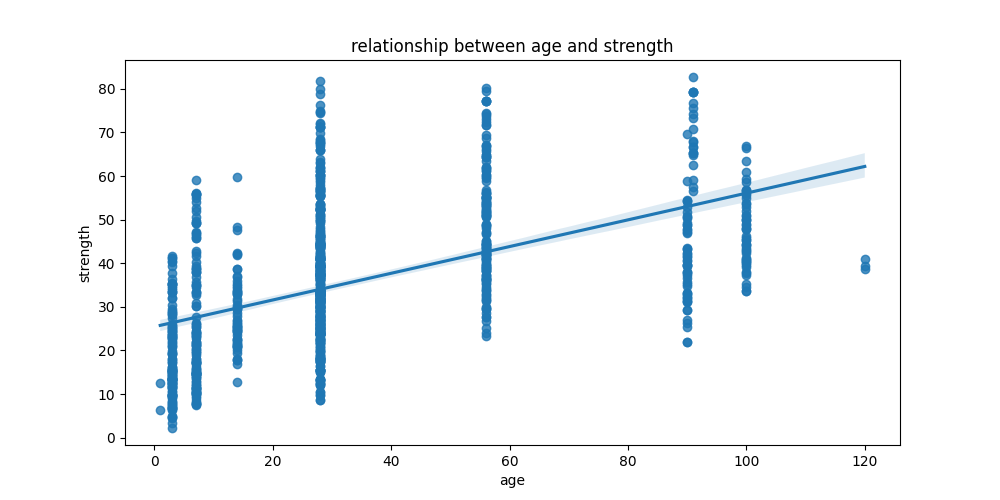
1. **Feature slag has outliers**
   1. Total number of outliers in this slag is 2
   2. Outliers percentage in slag is 0.0%
2. **Feature ash does not have any outliers**
3. **Feature water has any outliers**
   1. Total number of outliers in this water is 9
   2. Outliers percentage in water is 1.0%
4. **Feature superplastic has any outliers**
   1. Total number of outliers in this superplastic is 10
   2. Outliers percentage in superplastic is 1.0%
5. **Feature coarseagg does not have any outliers**
6. **Feature fineagg has any outliers**
   1. Total number of outliers in this fineagg is 5
   2. Outliers percentage in fineagg is 0.0%
7. **Feature age has outliers**
   1. Total number of outliers in this age is 59
   2. Outliers percentage in age is 6.0%

**Step 4: Relationship study**







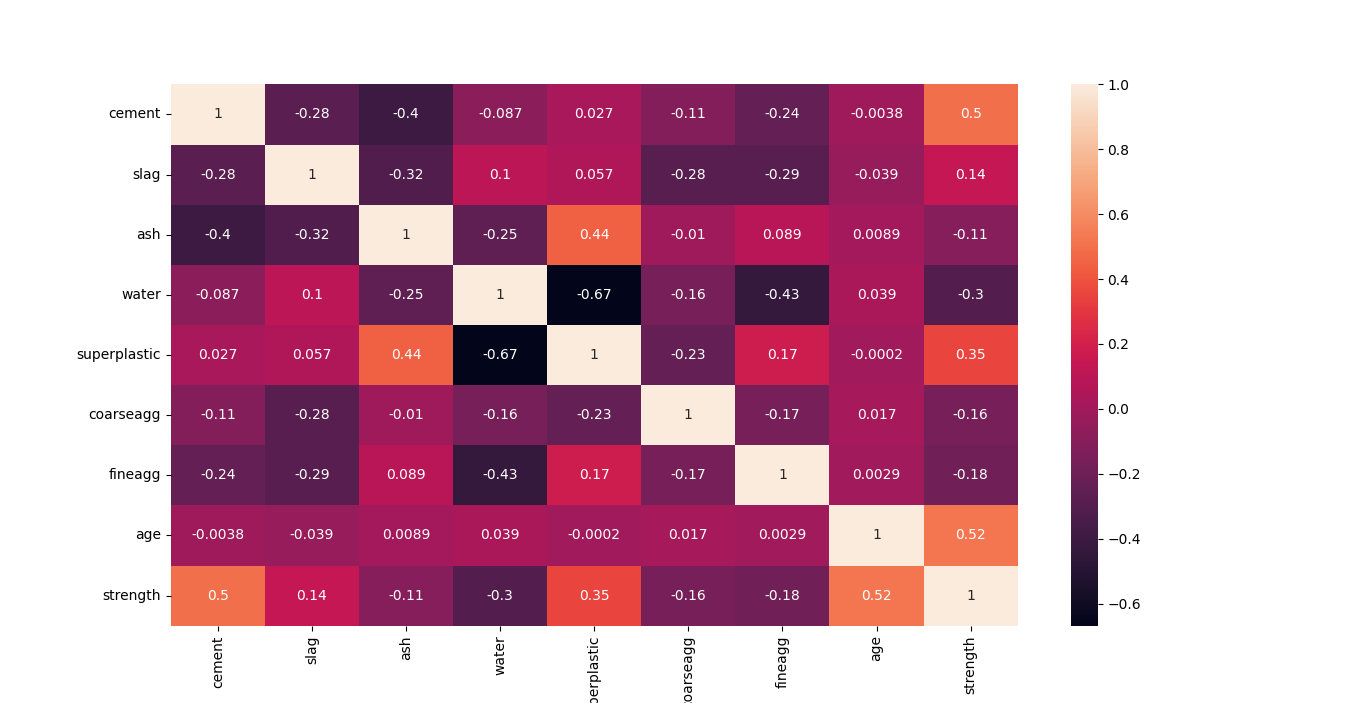


**Analysis:**

* Cement and Strength have a strong positive correlation
* Slag and strength have mild positive correlation
* ash and strength have a mild negative correlation.
* Water and strength have a strong negative correlation.
* Superplastic and strength have a strong positive correlation.
* Coarseagg and strength have a mild negative correlation.
* Fineagg and strength have a mild negative correlation.
* Age and strength have a strong positive correlation.

**Multi-collinearity-Check**

**Stage-1: Correlation heatmap**



**Analysis**

Independent variables pairs with correlation greater than 30% are listed below

1. ash and superplastic

2. cement and ash

3. slag and ash

4. water and superplastic

5. water and fineagg

Many independent features have collinearity greater than 30% which indicates that there is a chance of multicollinearity.

**Multicollinearity test**

* Stage 2: Variance Inflating Factor (VIF)

formula for VIF = 1/(1-R2)

1. Regress every independent variable with each other and find the R2 score

2. find out VIF using above formula

3. if VIF is more than 5 for any independent variable we can conclude that multi-collinearity exists.

|  |  |  |
| --- | --- | --- |
| **Position** | **VIF** | **Independent variables** |
| **5** | 86.535959 | coarseagg |
| **3** | 84.176659 | water |
| **6** | 70.274934 | fineagg |
| **0** | 14.826526 | cement |
| **4** | 5.980732 | superplastic |
| **2** | 4.486699 | ash |
| **1** | 3.43077 | slag |
| **7** | 2.449118 | age |

**Analysis**:

* Cement water, superplastic, coarseagg and fineagg have VIF score > 5 so we can conclude that multi-collinearity exists.

**Correlation with target feature**

|  |  |  |
| --- | --- | --- |
|  | **independent variables** | **correlation** |
| **7** | age | 0.519136 |
| **0** | cement | 0.497832 |
| **4** | superplastic | 0.353588 |
| **1** | slag | 0.137201 |
| **2** | ash | -0.105755 |
| **5** | coarseagg | -0.164935 |
| **6** | fineagg | -0.179536 |
| **3** | water | -0.304172 |

**Analysis:**

* Age, cement, superplastic, water and fineagg have higher correlation with strength.

**Step 5: Feature engineering**

Applying Principal component analysis and adding target feature

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PC\_1** | **PC\_2** | **PC\_3** | **PC\_4** | **PC\_5** | **PC\_6** | **Strength** |
| **0** | 1.826467 | 1.284669 | -1.3064 | -0.07124 | -0.53415 | -0.79045 | 29.89 |
| **1** | -1.87079 | -0.59563 | -1.09661 | -1.29665 | -0.12902 | -0.48972 | 23.51 |
| **2** | -0.74987 | -0.4639 | -0.52552 | 0.698966 | -0.78385 | 0.672137 | 29.22 |
| **3** | 2.776191 | 0.585938 | -0.46763 | 0.036635 | 0.034228 | 0.865712 | 45.85 |
| **4** | 0.907764 | 0.917339 | -0.89021 | -1.15575 | 0.415338 | -1.13323 | 18.29 |

Total independent features are reduced from 8 to 6

Above features are considered for model building and evaluation.

Explained variance ratio after PCA is:

[0.27182784 0.44776951 0.61076485 0.73733075 0.86207842 0.97452865]

**Model building:**

**1. Train-test split –** Splitting the data into train and test sets

**2. Cross-validation -** Evaluating machine learning models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data.

**3. Hyperparameter tuning-** Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors.

Results after initial testing

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Model name** | **r2\_score** | **RMSE** |
| **9** | XGBRegressor | 0.860129 | 6.380286 |
| **5** | RandomForestRegressor | 0.84496 | 6.717346 |
| **8** | GradientBoostingRegressor | 0.822697 | 7.183456 |
| **4** | KneighborRegressor | 0.796457 | 7.696695 |
| **6** | SVR | 0.711303 | 9.166357 |
| **2** | Ridge | 0.704407 | 9.275192 |
| **0** | LinearRegression | 0.7044 | 9.275304 |
| **3** | DecisionTreeRegressor | 0.69974 | 9.348126 |
| **1** | Lasso | 0.69752 | 9.382628 |
| **7** | AdaBoostRegressor | 0.697386 | 9.384702 |

Hyperparameter tuning:

Best parameters for different models are

model: Random forest

Best\_params: {'max\_depth': 11, 'max\_features': 7, 'n\_estimators': 200}

model: XGboost

Best\_params: {'alpha': 3, 'eta': 0.1, 'gamma': 0, 'max\_depth': 10, 'reg\_lambda': 3}

model: GradientBoosting

Best\_params: {'learning\_rate': 0.1, 'n\_estimators': 200}

model: Lasso

Best\_params: {'alpha': 0.1}

model: Ridge

Best\_params: {'alpha': 9}

model: AdaBoost

Best\_params: {'algorithm': 'ball\_tree', 'weights': 'distance'}

model: KNN

Best\_params: {'C': 2, 'gamma': 'scale'}

model: SVR

Best\_params: {'criterion': 'friedman\_mse', 'max\_depth': 11, 'max\_features': 4}

model: DicisionTreeRegressor

Best\_params: {'learning\_rate': 1, 'n\_estimators': 150}

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_names** | **cv\_score** | **cv\_std** |
| **9** | XGBRegressor | 0.891592 | 0.021769 |
| **1** | RandomForestRegressor | 0.867538 | 0.033461 |
| **5** | KNeighborsRegressor | 0.853697 | 0.022697 |
| **8** | GradientBoostingRegressor | 0.850181 | 0.024145 |
| **6** | SVR | 0.779467 | 0.037533 |
| **4** | DecisionTreeRegressor | 0.764166 | 0.039454 |
| **7** | AdaBoostRegressor | 0.715132 | 0.037002 |
| **3** | Ridge | 0.70664 | 0.045945 |
| **2** | Lasso | 0.706614 | 0.045198 |
| **0** | LinearRegression | 0.706585 | 0.046654 |

Results after cross validation:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_names** | **cv\_score** | **cv\_std** |
| **9** | XGBRegressor | 0.88178 | 0.02992 |
| **5** | RandomForestRegressor | 0.86717 | 0.0343 |
| **8** | GradientBoostingRegressor | 0.82573 | 0.03011 |
| **4** | KNeighborsRegressor | 0.79994 | 0.03127 |
| **3** | DecisionTreeRegressor | 0.74441 | 0.05332 |
| **6** | SVR | 0.72576 | 0.04063 |
| **2** | Ridge | 0.7066 | 0.04657 |
| **0** | LinearRegression | 0.70659 | 0.04665 |
| **7** | AdaBoostRegressor | 0.6995 | 0.04608 |
| **1** | Lasso | 0.68971 | 0.03667 |

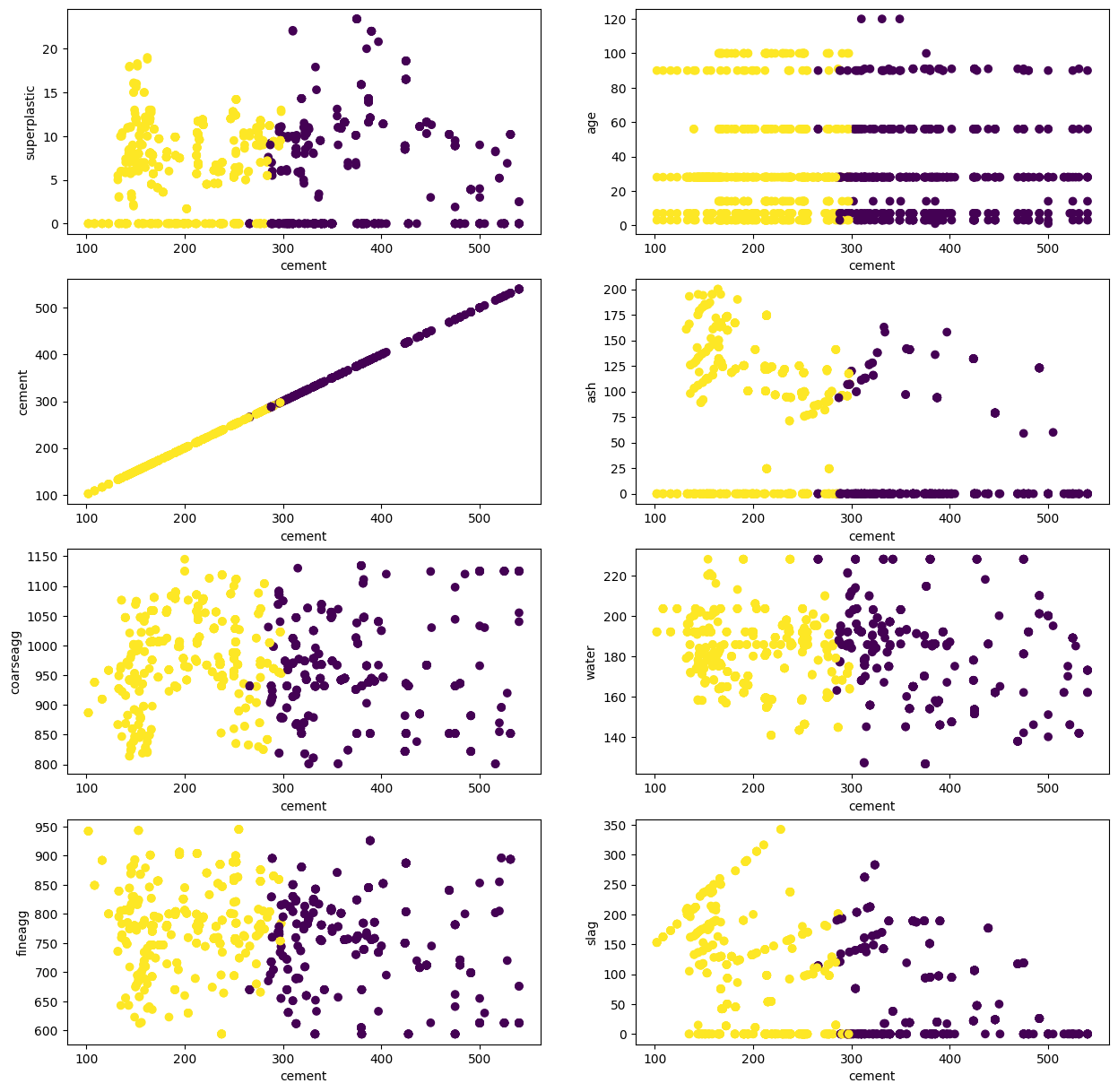
Results after hyperparameter tuning:

**Using clustering to check if it can help us improve accuracy**

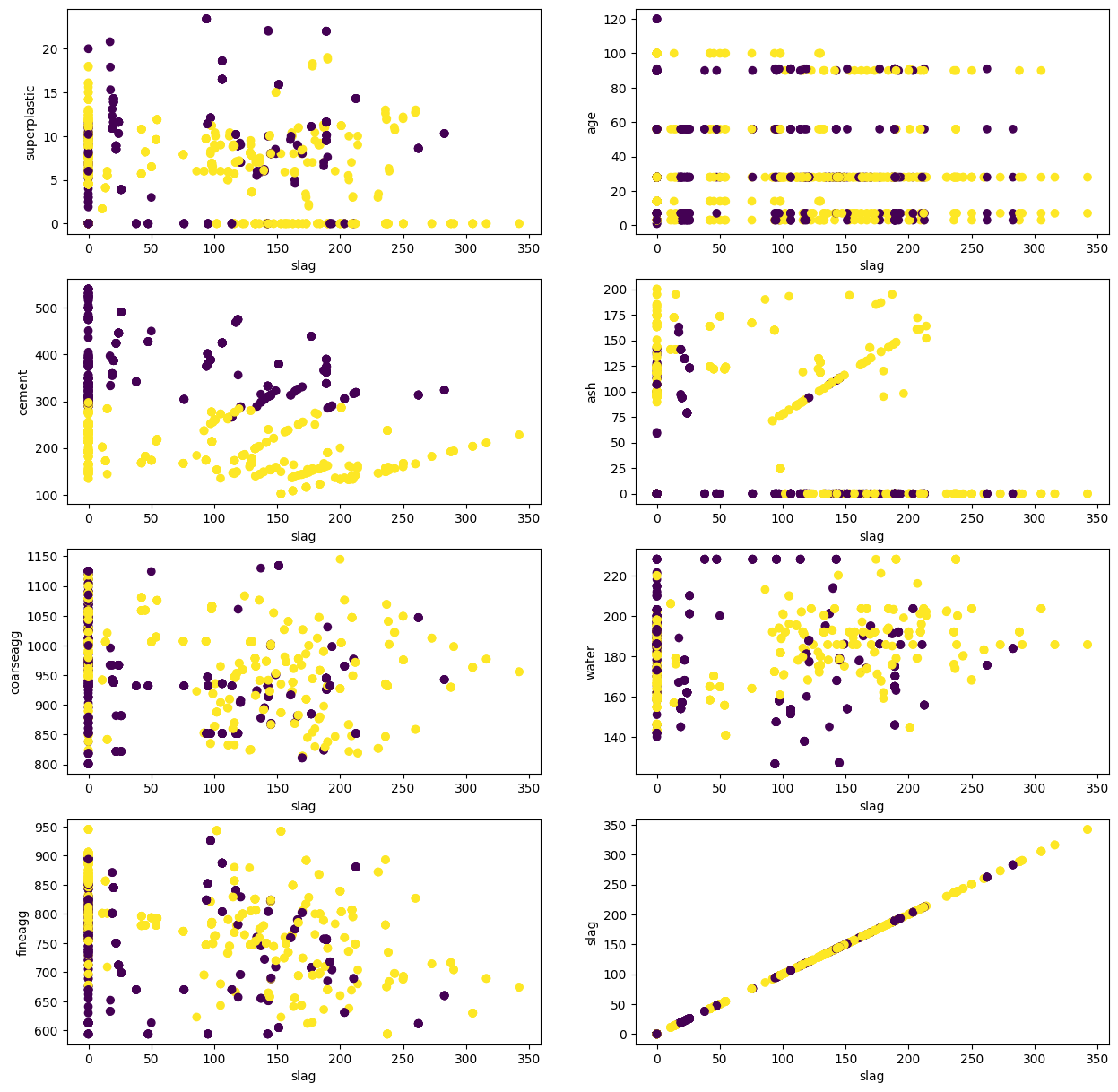
Checking features with target to see which one makes clear clusters

We are trying to use independent variables to see if there is an clusters being formed in the data so that we can use that as another independent variable in evaluation

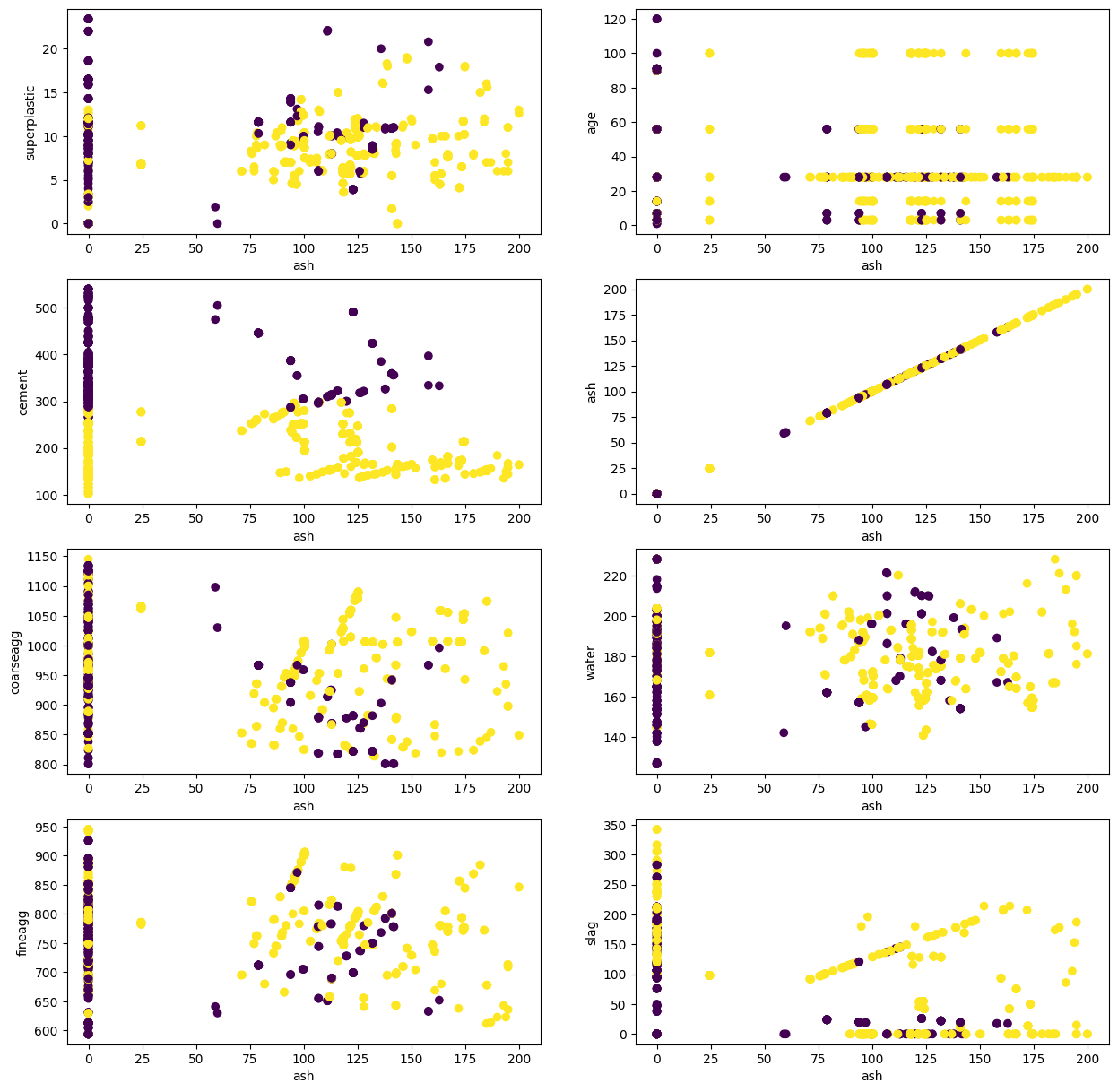
1. **Cement with strength**



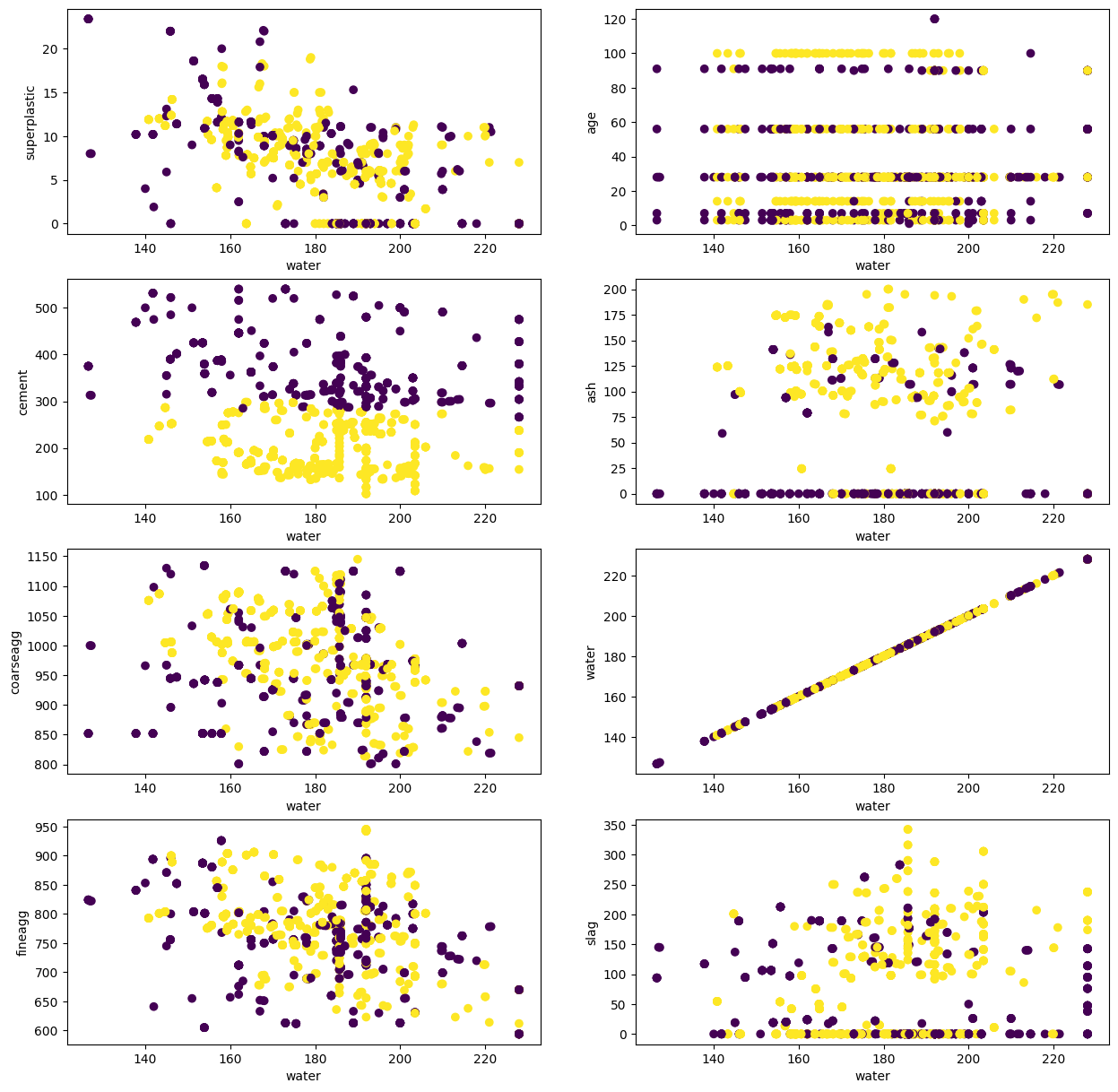
1. **Slag with Strength**



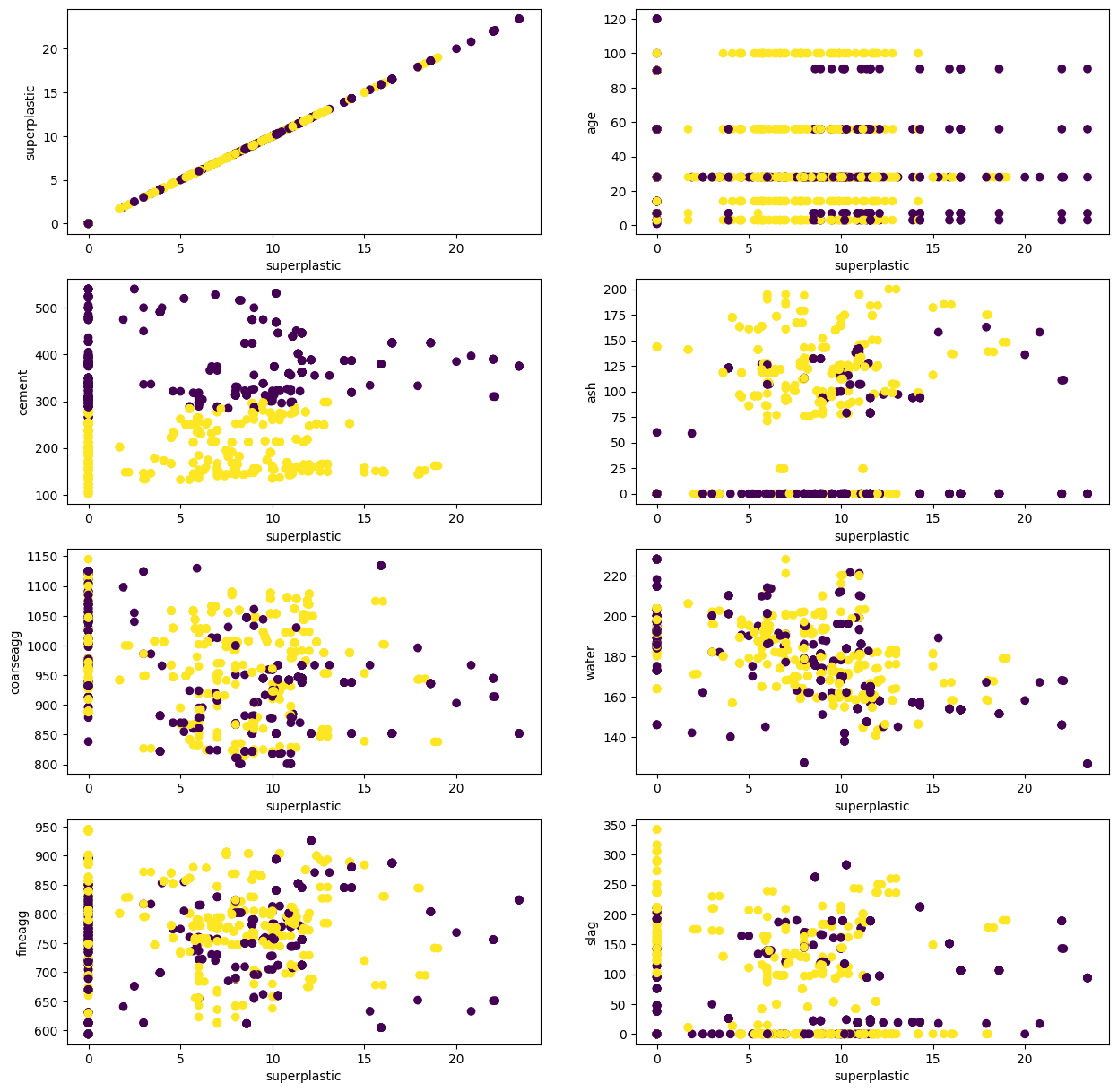
1. **Ash with Strength**



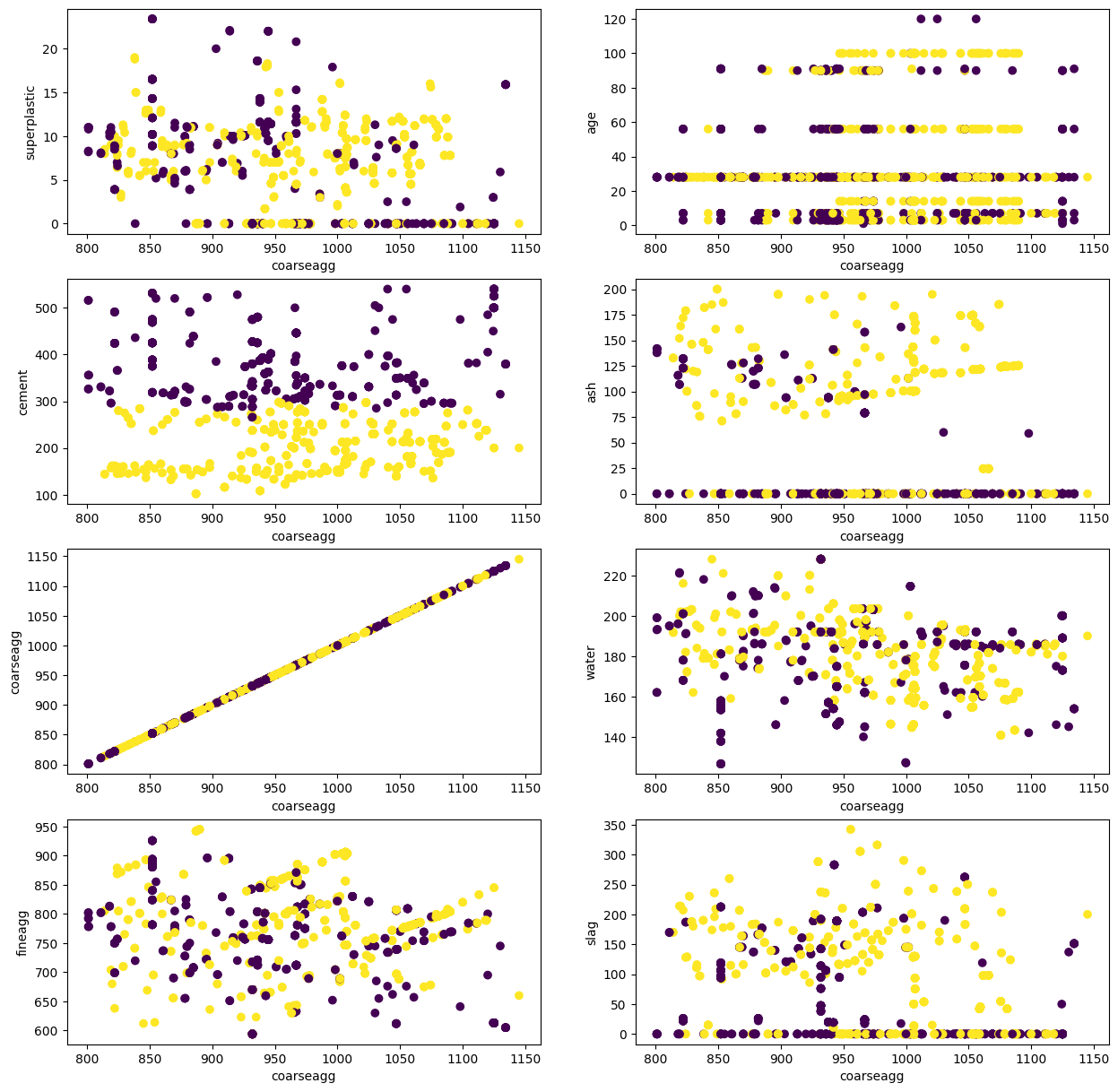
1. **Water with Strength**



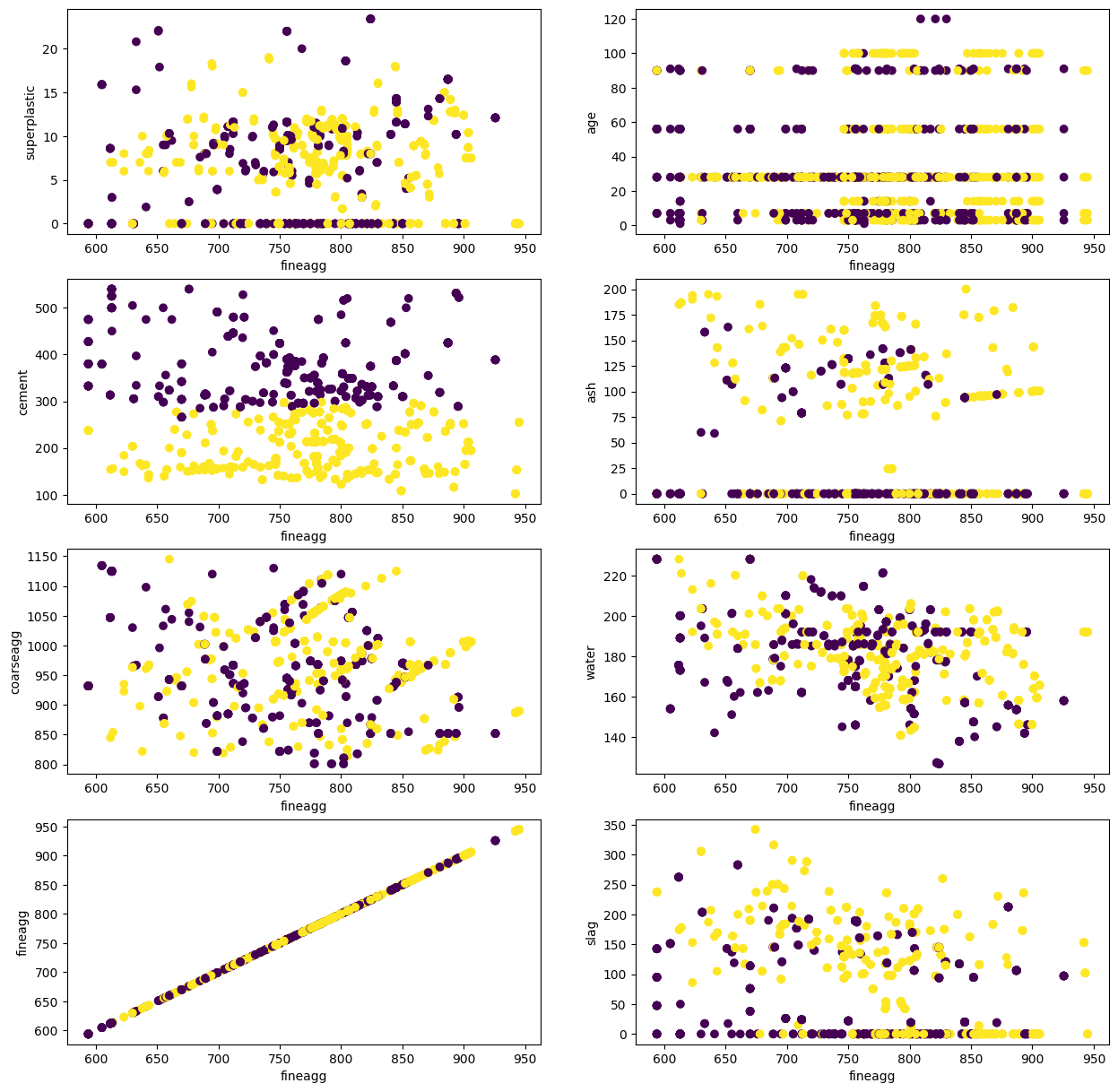
1. **Superplastic with Strength**



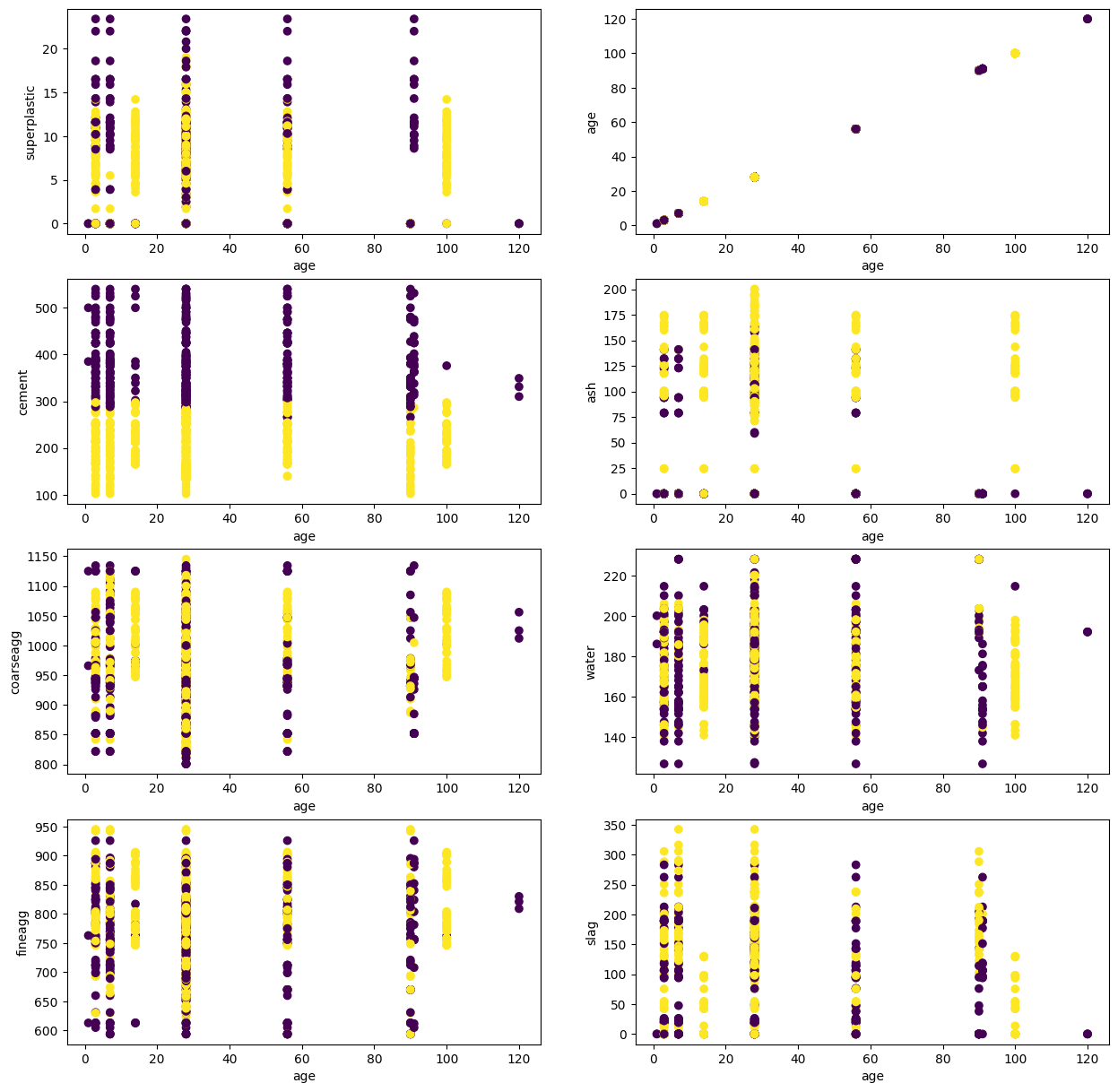
1. **Coarse aggregate with Strength**



1. **Fine aggregate with Strength**



1. **Age with Strength**



**Analysis**:

* Cement is forming clear clusters using all the independent variables.
* In column cluster, 1 represents data points belonging to cluster 1, 0 represents data points belonging to cluster 0.
* As values of cluster are less in scale compared to other values in different columns, we are trying to come up with a value which will we in range with respect to other value.
* In this case as cement is forming clear cluster, we are using that column.
* For this purpose, we have grouped cluster column with cement column and found mean and median of their respective cluster.

Model Evaluation on clustered data:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_name** | **r2\_score** | **RMSE** |
| **9** | XGBRegressor | 0.913379 | 5.020959 |
| **5** | RandomForestRegressor | 0.909214 | 5.14026 |
| **8** | GradientBoostingRegressor | 0.899472 | 5.409018 |
| **3** | DecisionTreeRegressor | 0.862543 | 6.324993 |
| **7** | AdaBoostRegressor | 0.768142 | 8.214605 |
| **1** | Lasso | 0.738715 | 8.72034 |
| **2** | Ridge | 0.738103 | 8.73054 |
| **0** | LinearRegression | 0.737821 | 8.735244 |
| **4** | KneighborRegressor | 0.704762 | 9.269623 |
| **6** | SVR | 0.224695 | 15.02147 |

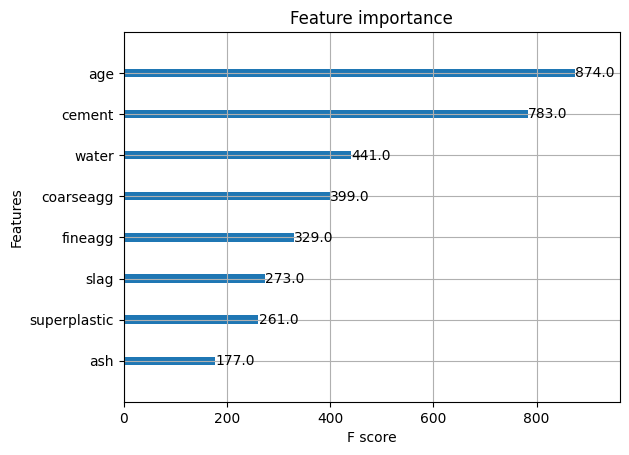
Cross validation in cluster data:

|  |  |  |
| --- | --- | --- |
| **model\_names** | **cv\_score** | **cv\_std** |
| XGBRegressor | 0.93437 | 0.01903 |
| RandomForestRegressor | 0.91691 | 0.02579 |
| GradientBoostingRegressor | 0.90313 | 0.01985 |
| DecisionTreeRegressor | 0.87339 | 0.04243 |
| AdaBoostRegressor | 0.77745 | 0.03581 |
| LinearRegression | 0.73653 | 0.03507 |
| Ridge | 0.73635 | 0.03502 |
| Lasso | 0.73626 | 0.03458 |
| KNeighborsRegressor | 0.69724 | 0.06063 |
| SVR | 0.24448 | 0.04537 |

Cross validation in cluster data post hyperparameter:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_names** | **cv\_score** | **cv\_std** |
| **9** | XGBRegressor | 0.93404 | 0.02108 |
| **8** | GradientBoostingRegressor | 0.92132 | 0.01917 |
| **1** | RandomForestRegressor | 0.91585 | 0.02451 |
| **4** | DecisionTreeRegressor | 0.79715 | 0.06219 |
| **5** | KNeighborsRegressor | 0.78643 | 0.04617 |
| **7** | AdaBoostRegressor | 0.78524 | 0.03126 |
| **0** | LinearRegression | 0.73653 | 0.03507 |
| **2** | Lasso | 0.73638 | 0.03497 |
| **3** | Ridge | 0.73636 | 0.03502 |
| **6** | SVR | 0.34999 | 0.04467 |

Applying feature importance to increase accuracy:

With the help of above graph features slag, superplastic, ash are dropped.

Cross validation in cluster data post hyperparameter:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_names** | **cv\_score** | **cv\_std** |
| **9** | XGBRegressor | 0.912672 | 0.02235 |
| **8** | GradientBoostingRegressor | 0.892403 | 0.021125 |
| **1** | RandomForestRegressor | 0.887221 | 0.026951 |
| **4** | DecisionTreeRegressor | 0.818769 | 0.055218 |
| **5** | KNeighborsRegressor | 0.773739 | 0.061818 |
| **7** | AdaBoostRegressor | 0.738537 | 0.046859 |
| **2** | Lasso | 0.702804 | 0.038205 |
| **3** | Ridge | 0.702798 | 0.038231 |
| **0** | LinearRegression | 0.702798 | 0.038232 |
| **6** | SVR | 0.348765 | 0.048318 |

Using RFE 'ash', 'water', 'coarseagg', 'fineagg' features are dropped.

Cross validation in cluster data post hyperparameter:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **model\_names** | **cv\_score** | **cv\_std** |
| **9** | XGBRegressor | 0.912549 | 0.025469 |
| **8** | GradientBoostingRegressor | 0.895459 | 0.023874 |
| **1** | RandomForestRegressor | 0.894085 | 0.029324 |
| **4** | DecisionTreeRegressor | 0.837571 | 0.059628 |
| **5** | KNeighborsRegressor | 0.812334 | 0.040242 |
| **7** | AdaBoostRegressor | 0.732052 | 0.041737 |
| **2** | Lasso | 0.704178 | 0.049496 |
| **3** | Ridge | 0.70417 | 0.049577 |
| **0** | LinearRegression | 0.704169 | 0.049584 |
| **6** | SVR | 0.60282 | 0.039403 |