ISEN 613 Project - Fall'24

Team Members

Vedant Samirbhai Shah (UIN 335007238)

Yash Dalmai (UIN 335007845)

Harsh Tarang Shah (UIN 136003995)

Howdy, for the project analysis of the concrete dataset, it contains `1030 observations` and `9 variables.` The dataset is based on predicting the strength of the concrete as a function of variables. The variables included in the dataset are Cement, Slag, Ash, Water, Superplastics, Coarseagg, Fineagg, Age, and Strength. The first 8 variables are the input variables, and strength is the output variable. We can start the analysis by first performing descriptive analysis where we initially calculate the basic statistical metrics like Mean, Median, Variance, and Range of each variable. Then we will plot histograms of each variable where the frequency of the variable is calculated and the plot can give the skewness of the data, monotonous curves, normality, and other parameters in the histogram. Then in the descriptive analysis, we will plot the **scatter matrix**, where we get individual relations of all the variables with each other on a single matrix. After plotting the scatter matrix, we will plot a **box plot** to detect the outliers, skewness, spread, and dispersion in the dataset. Then we plot the **histograms** and scatter plots of strength with the variables to understand the relation of the variables with strength individually, and to explain feature selection, the relationship between the variables can be explained by the heatmap plotting of the dataset.

We then plot the Regression for the dataset where at the start, we plot **linear regression** of each predictor with strength to give the output of the correlation and the spread of the points on the graph. We also plot the **residual plots** for the graphs where we get the spread of residuals with the regression line for each predictor. We then move forward to the refined model of **multiple linear regression** where from the summary we think which variables are important are then only used for this model. We then perform the **forward subset selection** and **cross-validation (Validation Set Approach & K-Fold)** for the model, which provides us with the variables that will influence strength as a whole. Then we perform the **Ridge Regression** and **Lasso** on the dataset where we get our key findings and also use cross-validation to check for the **TEST MSE**.

Afterward, then we use classification techniques to plot the graphs for strength as a binary variable where the strength < 40MPa is plotted. We use classification techniques like **Linear Discriminant Analysis, Quadratic Discriminant Analysis, and KNN**.

Then we plot a **Regression Tree** to get insights into our analysis which is used for plotting the conclusion of the project.

In the end, we report our findings, also the analysis for **extensions** is explained, and the **limitations** of the current dataset are provided in-depth where future implementation can provide better results.

In [446…

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from ISLP.models import ModelSpec as MS, Stepwise, sklearn_selected
from statsmodels.api import OLS
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.anova import anova_lm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.base import BaseEstimator, RegressorMixin
from sklearn.metrics import accuracy_score
from sklearn.linear_model import ElasticNetCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis, LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor, plot_tree
from ISLP import confusion_table
from functools import partial
```

Statistical Analysis

We start by first examining the dataset with basic statistical functions where we can get a rough interpretation of the data, its varaibility and the distribution.

In [448…

```
concrete_data = pd.read_csv('concrete.csv')
concrete_data
```

Out[448…

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
0	141.3	212.0	0.0	203.5	0.0	971.8	748.5	28	29.89
1	168.9	42.2	124.3	158.3	10.8	1080.8	796.2	14	23.51
2	250.0	0.0	95.7	187.4	5.5	956.9	861.2	28	29.22
3	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85
4	154.8	183.4	0.0	193.3	9.1	1047.4	696.7	28	18.29
…	…	…	…	…	…	…	…	…	…
1025	135.0	0.0	166.0	180.0	10.0	961.0	805.0	28	13.29
1026	531.3	0.0	0.0	141.8	28.2	852.1	893.7	3	41.30
1027	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1028	342.0	38.0	0.0	228.0	0.0	932.0	670.0	270	55.06
1029	540.0	0.0	0.0	173.0	0.0	1125.0	613.0	7	52.61

1030 rows × 9 columns

Concrete dataset is firstly loaded and the dataset contains a total of 1030 observations and the parameters that are in the dataset are also listed.

In [450…

```
concrete_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   cement        1030 non-null   float64
 1   slag          1030 non-null   float64
 2   ash           1030 non-null   float64
 3   water         1030 non-null   float64
 4   superplastic  1030 non-null   float64
 5   coarseagg     1030 non-null   float64
 6   fineagg       1030 non-null   float64
 7   age           1030 non-null   int64  
 8   strength      1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.6 KB
```

The dataset variables datatype are categorized in this code cell, where we can find that only `Age is an integer datatype` other variables are of float type.

In [452…
```
concrete_data.describe()
```

Out[452…

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
count	1030.000000	1030.000000	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	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

Here the standard statstical parameters like Mean, Standard Deviation, Minimum and Maximum Value for each varaible.

In [454…

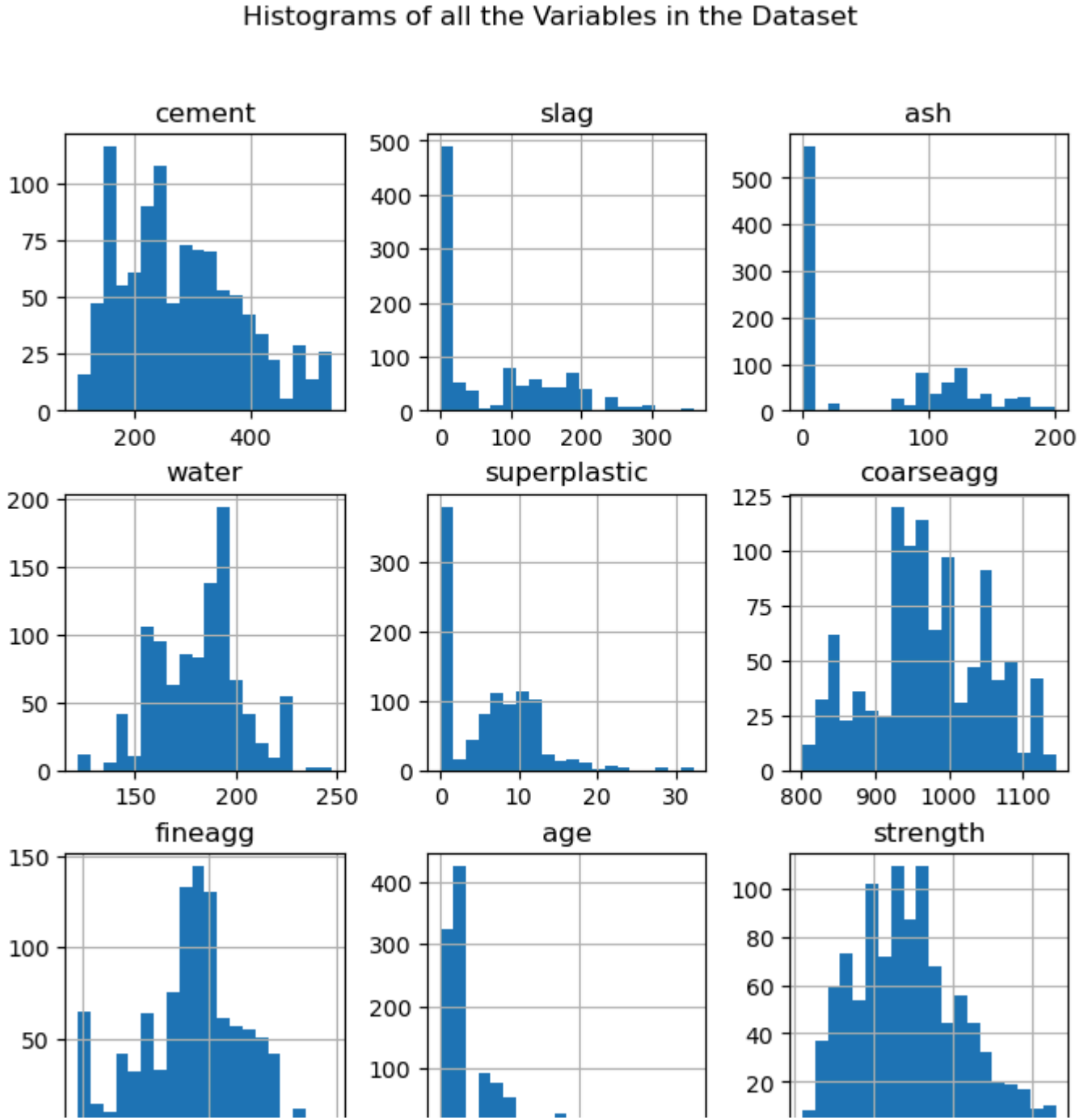
```
statistical_analysis = concrete_data.describe().T
statistical_analysis['variance'] = concrete_data.var()
statistical_analysis['range'] = statistical_analysis['max'] - statistical_analysis['min']
statistical_analysis['median'] = concrete_data.median()
statistical_analysis['mode'] = concrete_data.mode().iloc[0]
print(statistical_analysis[['mean', 'range', 'variance', 'median', 'mode']])
```

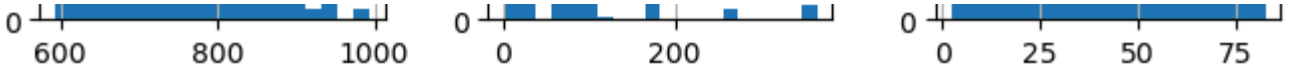
```
                    mean   range      variance   median   mode
cement        281.167864  438.00  10921.580220  272.900  362.6
slag           73.895825  359.40   7444.124812   22.000    0.0
ash            54.188350  200.10   4095.616541    0.000    0.0
water         181.567282  125.20    456.002651  185.000  192.0
superplastic    6.204660   32.20     35.686781    6.400    0.0
coarseagg     972.918932  344.00   6045.677357  968.000  932.0
fineagg       773.580485  398.60   6428.187792  779.500  594.0
age            45.662136  364.00   3990.437729   28.000   28.0
strength       35.817961   80.27    279.081814   34.445   33.4
```

Major Statstical metrics like Mean, Median, Mode, Variance and Range are calculated here. `Superplastics have a low variance and range as compared to other variables. Slag, Ash and Coarseagg have 0 mode so no values are repeating in the dataset.`

In [456…

```
concrete_data.hist(bins=20, figsize=(8, 8))
plt.suptitle('Histograms of all the Variables in the Dataset')
plt.show()
```


Histograms of all the Variables in the Dataset
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
strength



Here we have plotted a histogram for the frequency of the dataset for each variable.

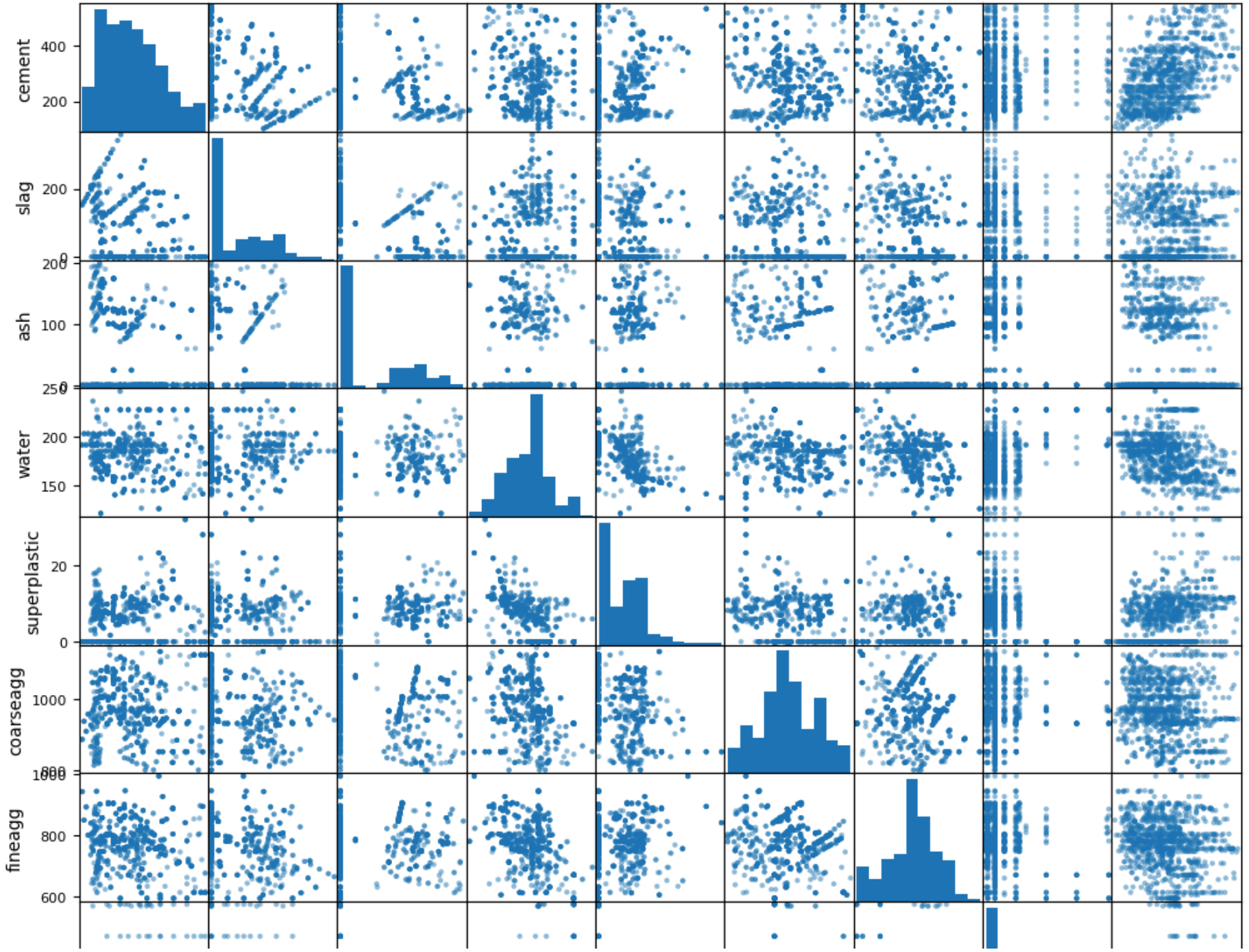
Our findings from this histogram reveal that `Slag, Ash, Superplastic, and Age are concentrated and highly rightly skewed`, which can suggest that these variables are limited to less valued use in the mixture.

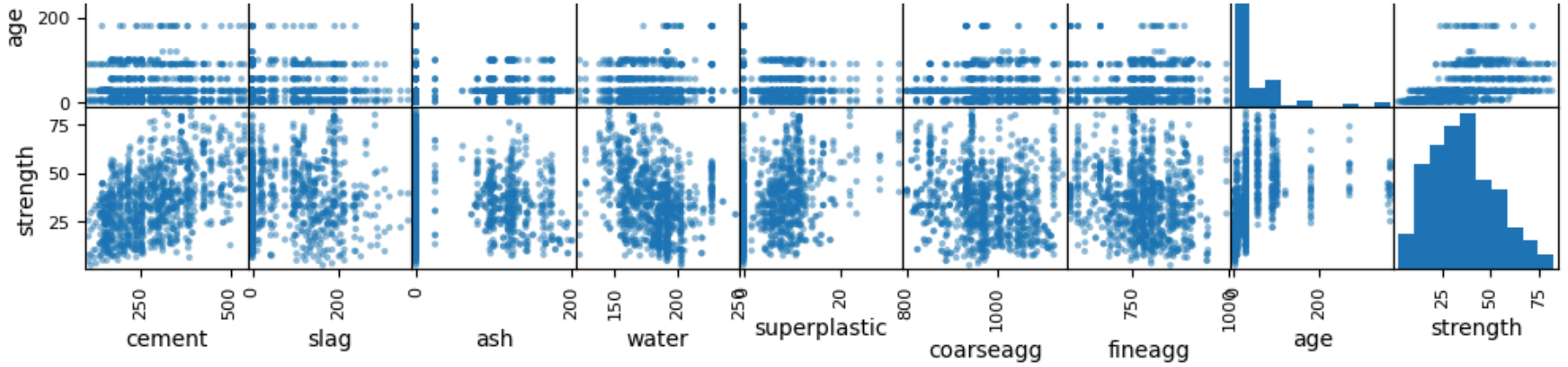
`Cement and Strength graphs are moderately distributed and slightly skewed towards the right.`

`Coarseagg and Fineagg are uniformly distributed.`

In [458…

```
pd.plotting.scatter_matrix(concrete_data[['cement', 'slag', 'ash', 'water', 'superplastic', 'coarseagg',
       'fineagg', 'age', 'strength']], figsize=(12, 12));
plt.show()
```


cement
slag
ash
water
superplastic
coarseagg
fineagg
400
200
200
0
200
100
0
250
200
150
20
0
1000
800
1000
800
600



Some key findings from this scatter matrix can be noted -

`Cement v/s Strength` - There is a positive correlation between the two, as the value of cement increases the strength also increases.

`Water v/s Strength` - There is a blank space where the water level is too low or too high, which suggests that water concentration is between fixed values.

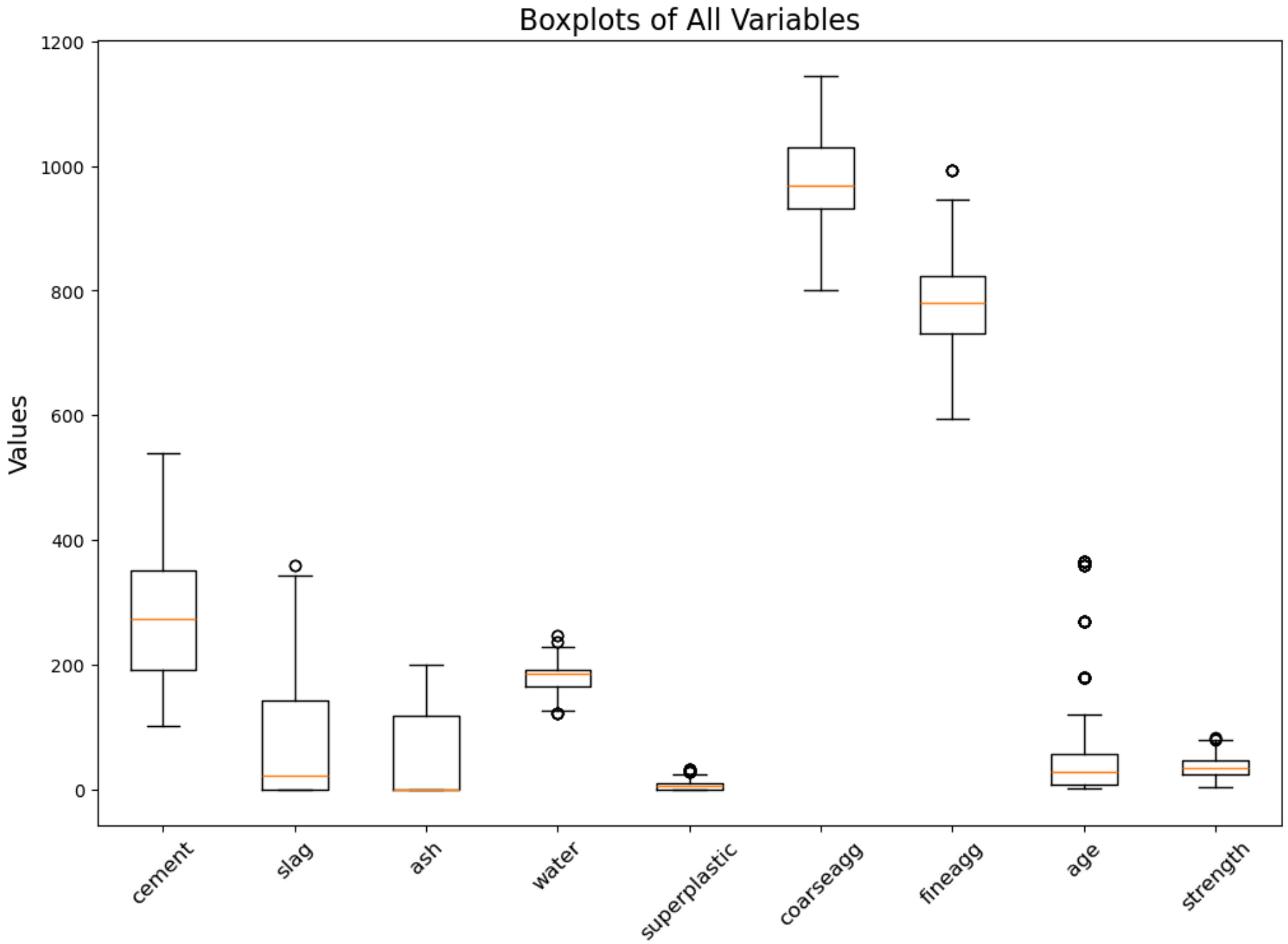
`Superplastic v/s Strength` - There is a cloud of points for the value of superplastic between 0-20 which suggests that over a certain value, there are no readings and also the value of strength does not change much with a change in the value of the superplastic.

`Age v/s Strength` - A key finding from this is that the readings are of some certain days only and on these days the strength is highly variable, a same day can have different values of strength due to the influence of other variables. Also for starting days age is positively related and then it becomes a constant.

In [460…

```
fig, ax = plt.subplots(figsize=(12, 8))
ax.boxplot(
    [
        concrete_data['cement'],
        concrete_data['slag'],
        concrete_data['ash'],
        concrete_data['water'],
        concrete_data['superplastic'],
        concrete_data['coarseagg'],
        concrete_data['fineagg'],
        concrete_data['age'],
```

```
        concrete_data['strength']
    ],
    labels=['cement', 'slag', 'ash', 'water', 'superplastic', 'coarseagg', 'fineagg', 'age', 'strength']
)

ax.set_title('Boxplots of All Variables', fontsize=16)
ax.set_ylabel('Values', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.show()
```


Boxplots of All Variables
Values
1200
1000
800
600
400
200
0
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
strength

Here are key findings can be noted as -

`Cement` - Moderately distributed across the values, the mean is also slightly skewed.

`Slag` — Slag is highly skewed, and many observations are 0, so data is concentrated to the bottom. Therefore, most of the mix must not contain slag as a variable. Also, an outlier exists for slag.

`Ash` - Ash is more concentrated on 0 same as slag, inferring the use is limited to mixing in the solution.

`Water` - Water is distributed in a short region, which suggests that it can influence the strength with some change in its value, and also has some outliers on lower and higher ends.

`Superplastic` - Superplastic is also concentrated on the lower end and near 0, suggesting major combinations not using it.

`Coarseagg` - Coarseagg is evenly distributed across the observations.

`Fineagg` - Fineagg is also evenly distributed and has an outlier on the higher end.

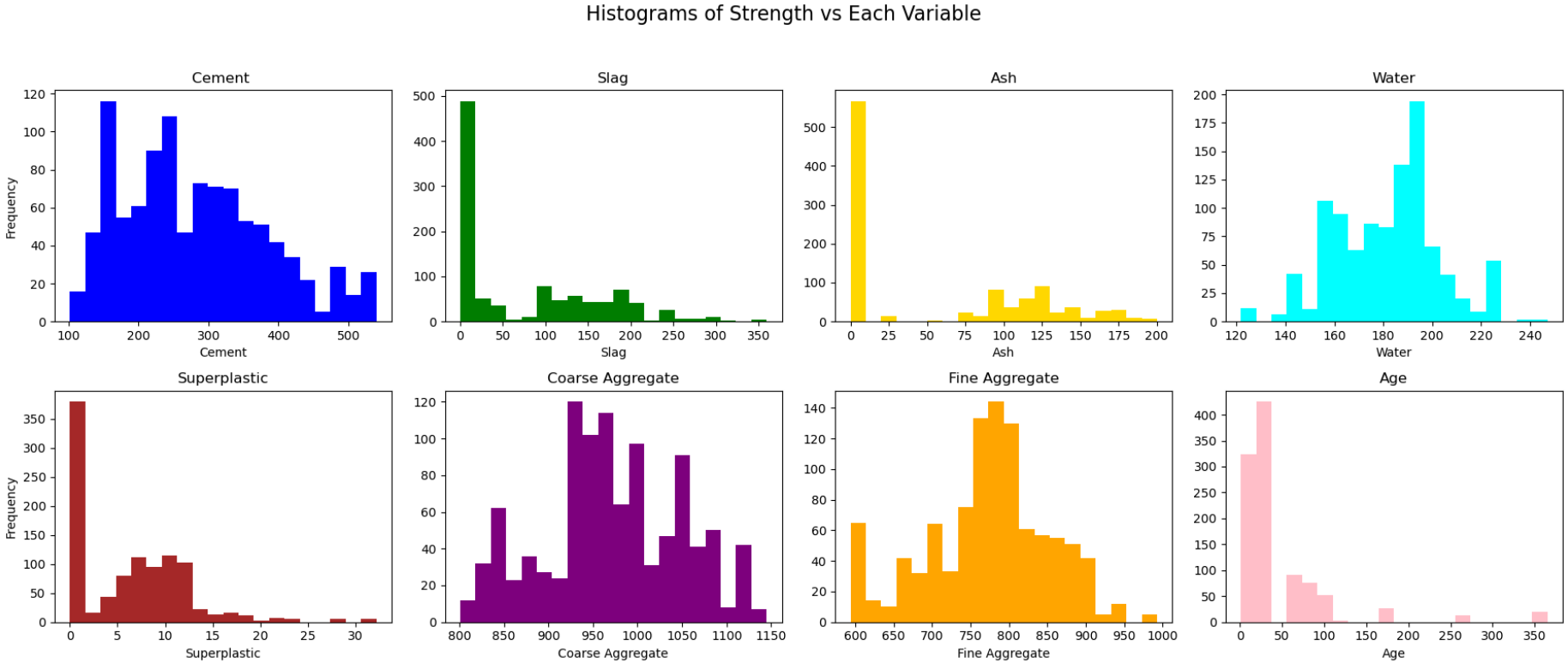
`Age` — The Age box plot suggests that most of the observations are within a small time period, while some outliers are at the higher end.

`Strength` - Strength is highly concentrated and also inferred that there can be major combinations that can affect the strength while also some variables can have a negative effect on the strength which can decrease the strength of the combined variables, so choosing the right variables is important for us, which does not cancel out the strength and there proportions in such a way that the overall strength increases.

In [462…

```
fig, axes = plt.subplots(2, 4, figsize=(18, 8))
fig.suptitle('Histograms of Strength vs Each Variable', fontsize=16)

axes[0, 0].hist(concrete_data['cement'], bins=20, color='blue')
axes[0, 0].set_title('Cement ')
axes[0, 0].set_xlabel('Cement')
axes[0, 0].set_ylabel('Frequency')

axes[0, 1].hist(concrete_data['slag'], bins=20, color='green')
axes[0, 1].set_title('Slag')
axes[0, 1].set_xlabel('Slag')

axes[0, 2].hist(concrete_data['ash'], bins=20, color='gold')
```

```
axes[0, 2].set_title('Ash')
axes[0, 2].set_xlabel('Ash')

axes[0, 3].hist(concrete_data['water'], bins=20, color='cyan')
axes[0, 3].set_title('Water')
axes[0, 3].set_xlabel('Water')

axes[1, 0].hist(concrete_data['superplastic'], bins=20, color='brown')
axes[1, 0].set_title('Superplastic')
axes[1, 0].set_xlabel('Superplastic')
axes[1, 0].set_ylabel('Frequency')

axes[1, 1].hist(concrete_data['coarseagg'], bins=20, color='purple')
axes[1, 1].set_title('Coarse Aggregate')
axes[1, 1].set_xlabel('Coarse Aggregate')

axes[1, 2].hist(concrete_data['fineagg'], bins=20, color='orange')
axes[1, 2].set_title('Fine Aggregate')
axes[1, 2].set_xlabel('Fine Aggregate')

axes[1, 3].hist(concrete_data['age'], bins=20, color='pink')
axes[1, 3].set_title('Age')
axes[1, 3].set_xlabel('Age')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Here the histograms are plotted with a higher number of bins and they are more spread to get a better understanding of the data for each variable. Also, they are color-coded to the same shade which will follow for their regression analysis, which can help in better understanding for the reader.

Some key findings can be noted as -

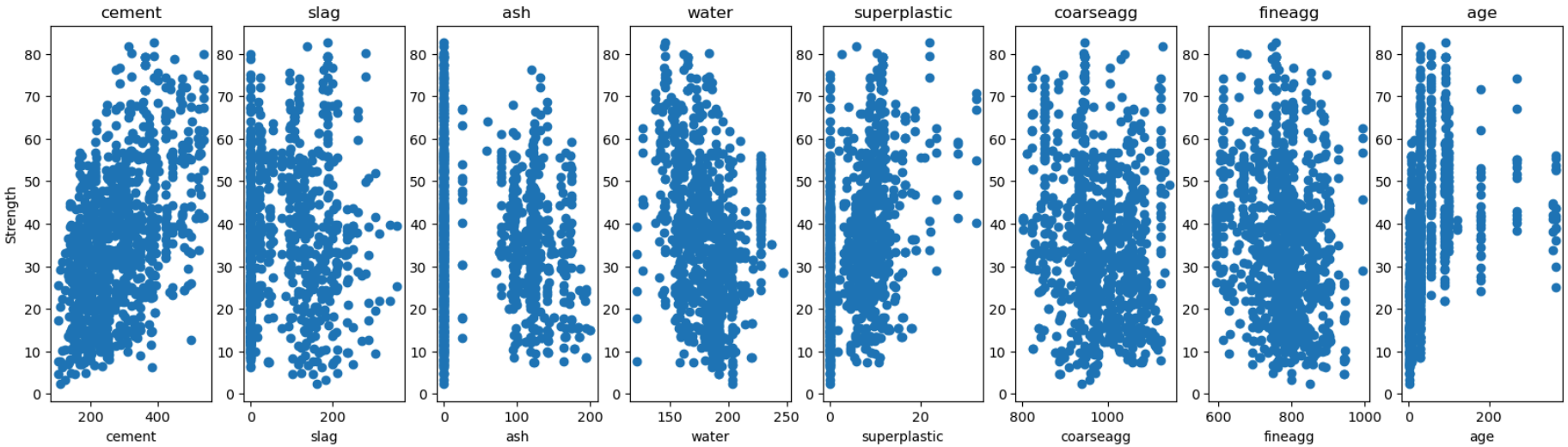
`Age` has many missing data on some specific days. So for future understanding, this might help us in giving future recommendations for the database.

`Superplastic, Slag and Ash` - Data for higher values is very limited, so minor changes in values at lower levels can have higher variations in strength.

In [464…

```
variables = ['cement', 'slag', 'ash', 'water', 'superplastic', 'coarseagg', 'fineagg', 'age']
fig, axes = plt.subplots(nrows=1, ncols=len(variables), figsize=(20, 5))
```

```
for i, var in enumerate(variables):
    axes[i].scatter(concrete_data[var], concrete_data['strength'])
    axes[i].set_title(var)
    axes[i].set_xlabel(var)
    if i == 0:
        axes[i].set_ylabel('Strength')
plt.show()
```



Here we have a scatter plot of each variable v/s strength, the findings for each plot are -

`Cement v/s Strength` - Positively correlated, inferred from the graph that an increase in the value of the cement leads to an increase in the value of strength.

`Slag v/s Strength` - Weak and random correlation where variability is high, and lower values lead to strength higher values also lead to the same level of strength.

`Ash v/s Strength` - Adding no ash can lead to a highly variable strength, so it can or cannot make a difference by using or not using it. Also, the data lags some level of ash in between 0 to 100, where there are not many ash content observations.

`Water v/s Strength` - An inverse/negative trend is seen, where an increase in the water level can lead to being less concentrated and affect on strength as the particles can not hold themselves together. So finding an optimal level of water can affect the strength.

`Superplastic v/s Strength` - A positive trend can be seen in the starting and then variability increases further as the value of superplastic increases.

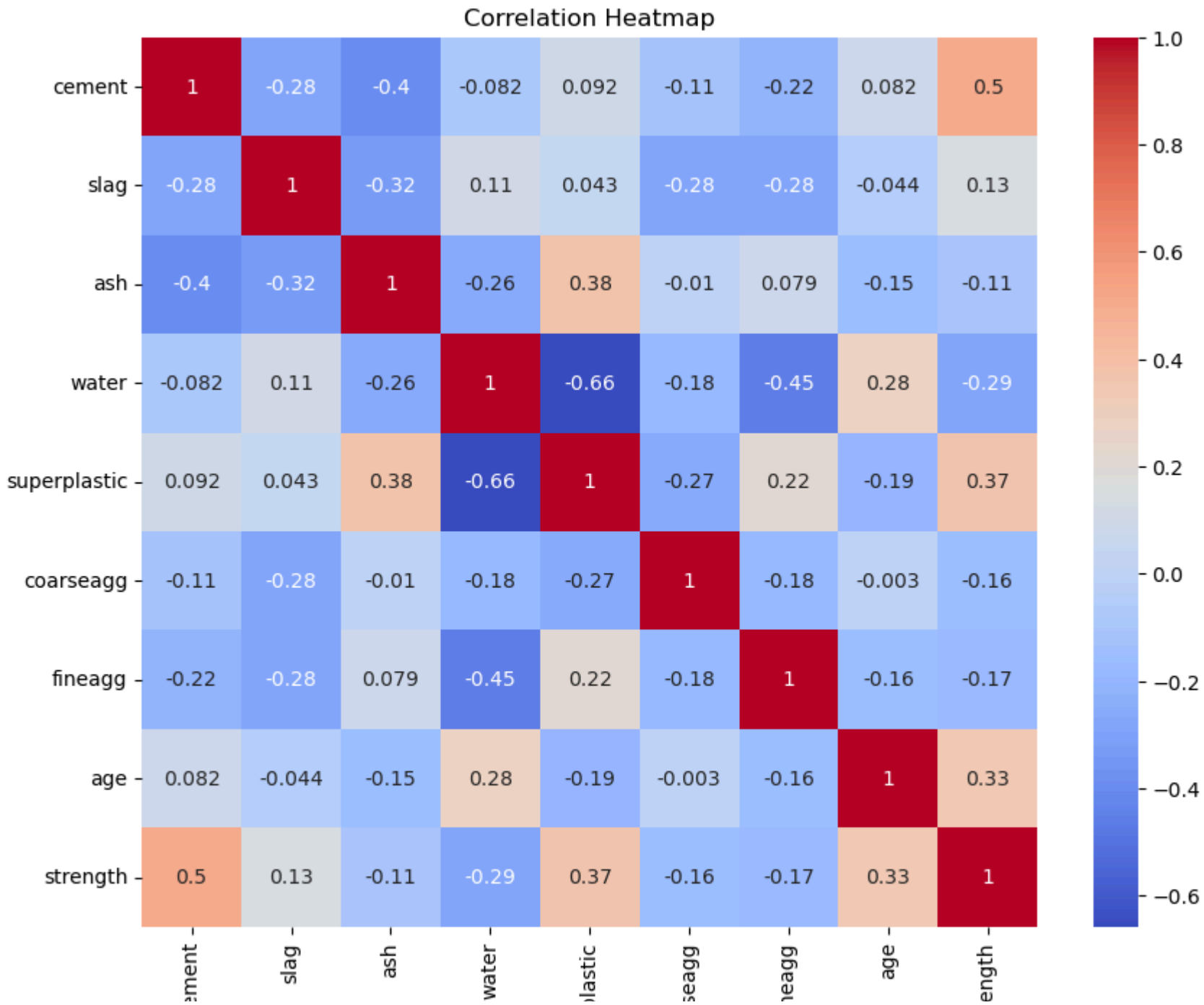
`Coarseaggerate v/s Strength` - A random cloud of points can be seen, so it can be interpreted that with an increase in the coarseagg, the strength does not have a large impact.

`Fineaggerate v/s Strength` - A similar random points can lead to show that the value of strength is just not directly dependent on the fine aggregate but it works with other variables in combination with other variables.

`Age v/s Strength` - A positive strength is seen in the starting days of testing, and then a slight increase and constant behavior can be seen of age and strength. As in the starting period over time cement becomes stronger and this can be seen in strength values as the cement binds itself better over a time up to a certain limit.

In [466…
```
correlation_matrix = concrete_data.corr()
```

In [467…
```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap
cement 1 -0.28 -0.4 -0.082 0.092 -0.11 -0.22 0.082 0.5
slag -0.28 1 -0.32 0.11 0.043 -0.28 -0.28 -0.044 0.13
ash -0.4 -0.32 1 -0.26 0.38 -0.01 0.079 -0.15 -0.11
water -0.082 0.11 -0.26 1 -0.66 -0.18 -0.45 0.28 -0.29
superplastic 0.092 0.043 0.38 -0.66 1 -0.27 0.22 -0.19 0.37
coarseagg -0.11 -0.28 -0.01 -0.18 -0.27 1 -0.18 -0.003 -0.16
fineagg -0.22 -0.28 0.079 -0.45 0.22 -0.18 1 -0.16 -0.17
age 0.082 -0.044 -0.15 0.28 -0.19 -0.003 -0.16 1 0.33
strength 0.5 0.13 -0.11 -0.29 0.37 -0.16 -0.17 0.33 1
slag
ash
age
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Here in the Correlation Heatmap, the values of relation can give us an idea of how related are the variables to each other, considering the strength here, the `positive relation is seen with Cement, Age, and Superplastic`. There is a high `negative correlation with water` and a minor decreasing negative correlation with coarse aggregate, ash, and fine aggregate. Also a minor positive relation between slag and strength.

Other key relations can be seen between `water and superplastic where they are highly negative` suggesting they should be maintained in a ratio which will eventually increase the strength.

`Ash and superplastic are positively correlated`, which can suggests that they can be used together which eventually affects strength.

Regression Analysis

Here in the `Regression Analysis` section, we will plot linear regression, and multiple linear regression and also find key inputs that influence the strength. We first plot the linear regression of every predictor with strength which will give us key findings, also we will plot the multiple linear regression to see the spread of observations with strength. Our metric of measurement will be R^2 which shows how much variability of the current model is captured by the regression. We then perform the cross-validation for higher level terms like Cement, Cement^2, and Cement^3 to find the MSE for each variable, here we will use two methods Validation Set Approach and K Fold Cross Validation.

In [471…

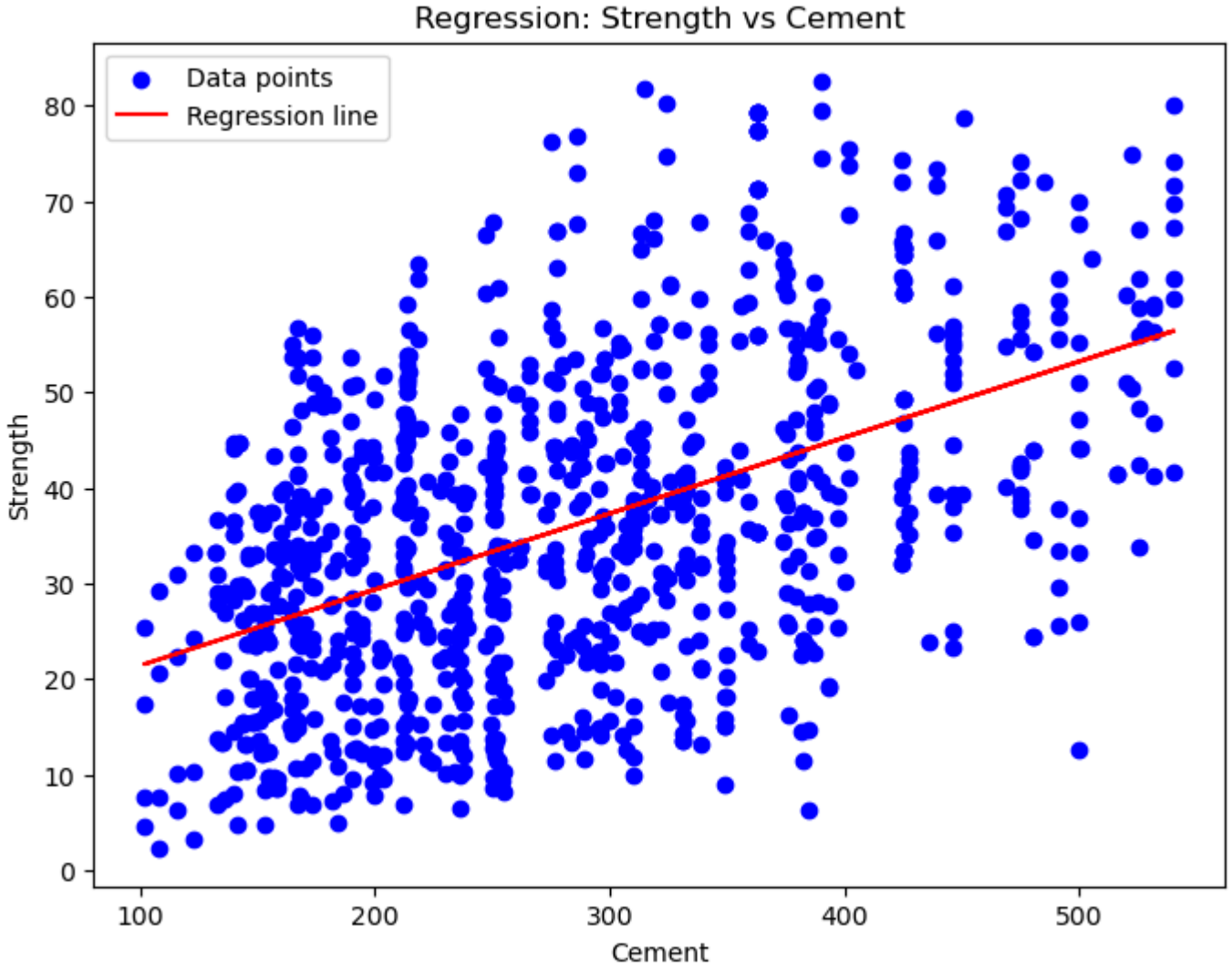
```
print(concrete_data.isnull().sum())

X = concrete_data.drop(columns=['strength'])
y = concrete_data['strength']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

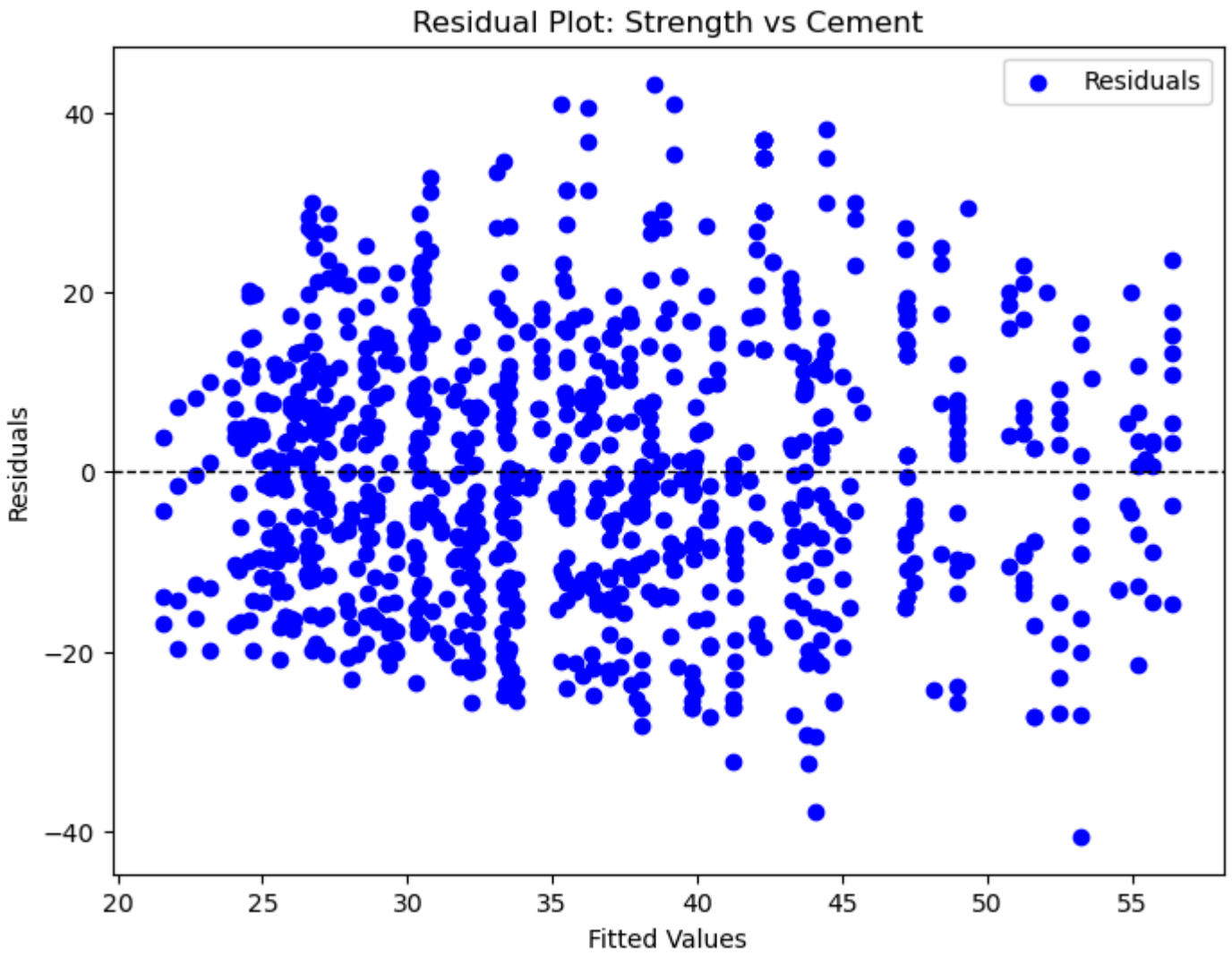
```
cement          0
slag            0
ash             0
water           0
superplastic    0
coarseagg       0
fineagg         0
age             0
strength        0
dtype: int64
```

Here we have first divided our variables X and Y, where X is all the variables that will help predict the Strength and Y is the strength. Also, X and Y are divided into the training and test data, where 30% of the data is the test data and the remaining is the training data. Also, we get to see that we don't have any missing values in the dataset for each variable. We then move forward to perform Linear Regression for each predictor with strength to see the relation between them.

In [473…

```
X_cement = sm.add_constant(concrete_data['cement'])
y_strength = concrete_data['strength']
model_cement = sm.OLS(y_strength, X_cement)
results_cement = model_cement.fit()
plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['cement'], concrete_data['strength'], label='Data points', color='blue')
plt.plot(concrete_data['cement'], results_cement.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Cement')
plt.xlabel('Cement')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_cement.fittedvalues, results_cement.resid, label='Residuals', color='blue')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Cement')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```


Regression: Strength vs Cement
Data points
Regression line
Strength
Cement
0
10
20
30
40
50
60
70
80
100
200
300
400
500



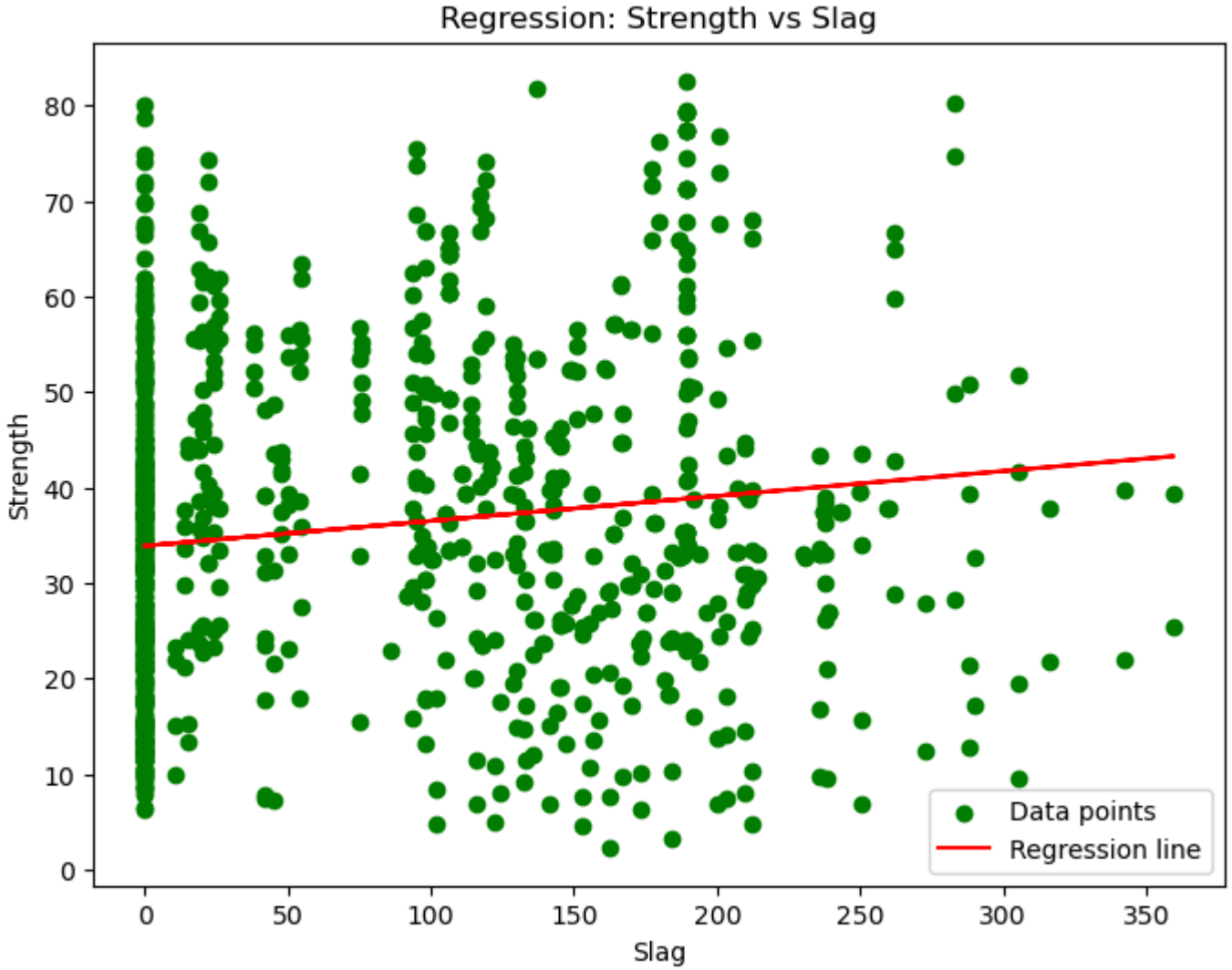
In the graphs, the points are equally variable in positive and negative distance from the linear regression line, which portrays the model as not biased. Using polynomial regression lines like using Cement^2 can also capture the data much better and have lower RSS (Residual Sum of Squares).

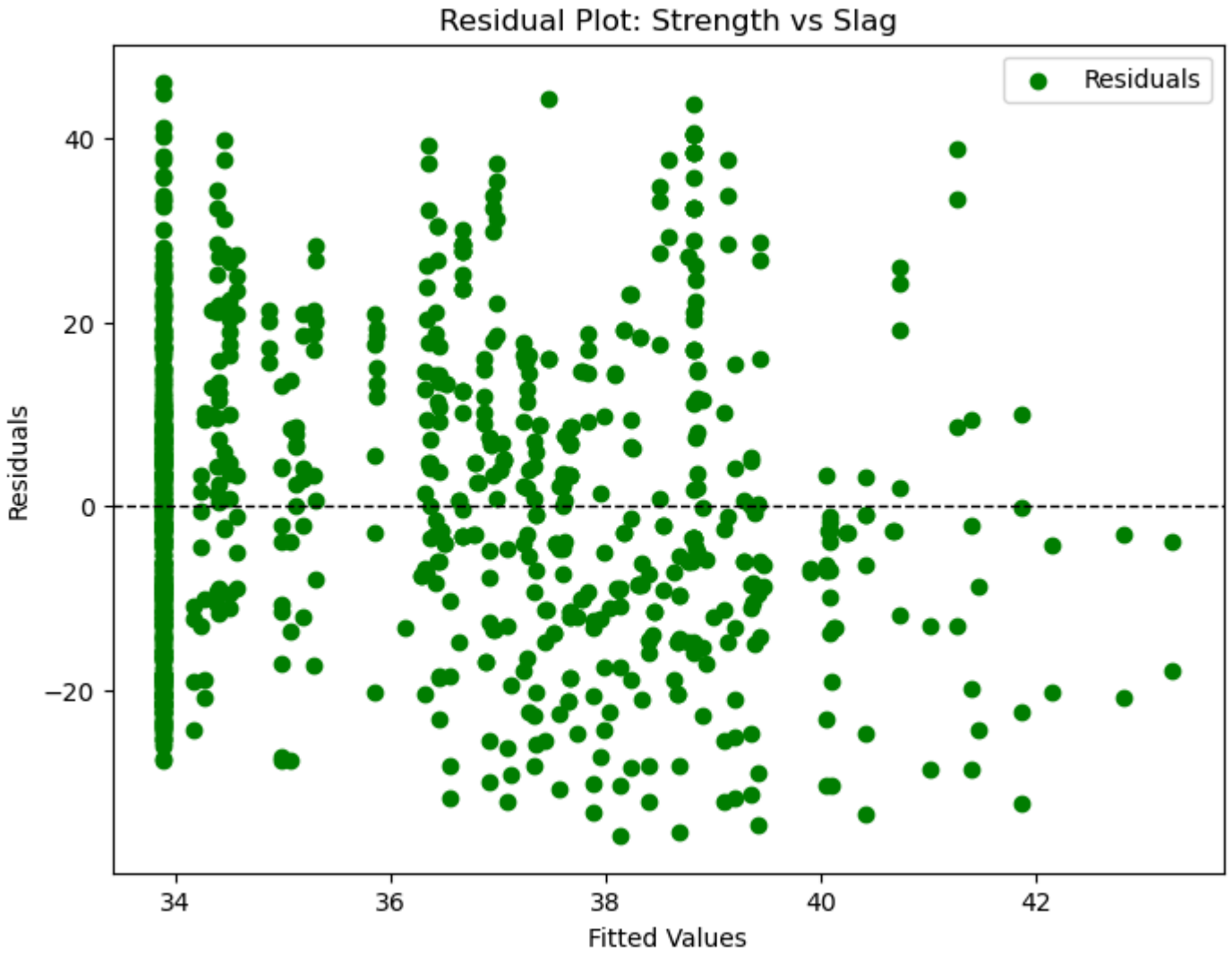
The dataset is more concentrated in values of `Cement less than 350`, and the `variability increases with the increase in the content of the cement`. The overall trend is `positively increasing` with the increase in the content of cement. This is due to the

increase in the content of cement causes an increase in strength.

In [475…

```
X_slag = sm.add_constant(concrete_data['slag'])

model_slag = sm.OLS(y_strength, X_slag)
results_slag = model_slag.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['slag'], concrete_data['strength'], label='Data points', color='green')
plt.plot(concrete_data['slag'], results_slag.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Slag')
plt.xlabel('Slag')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_slag.fittedvalues, results_slag.resid, label='Residuals', color='green')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Slag')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```

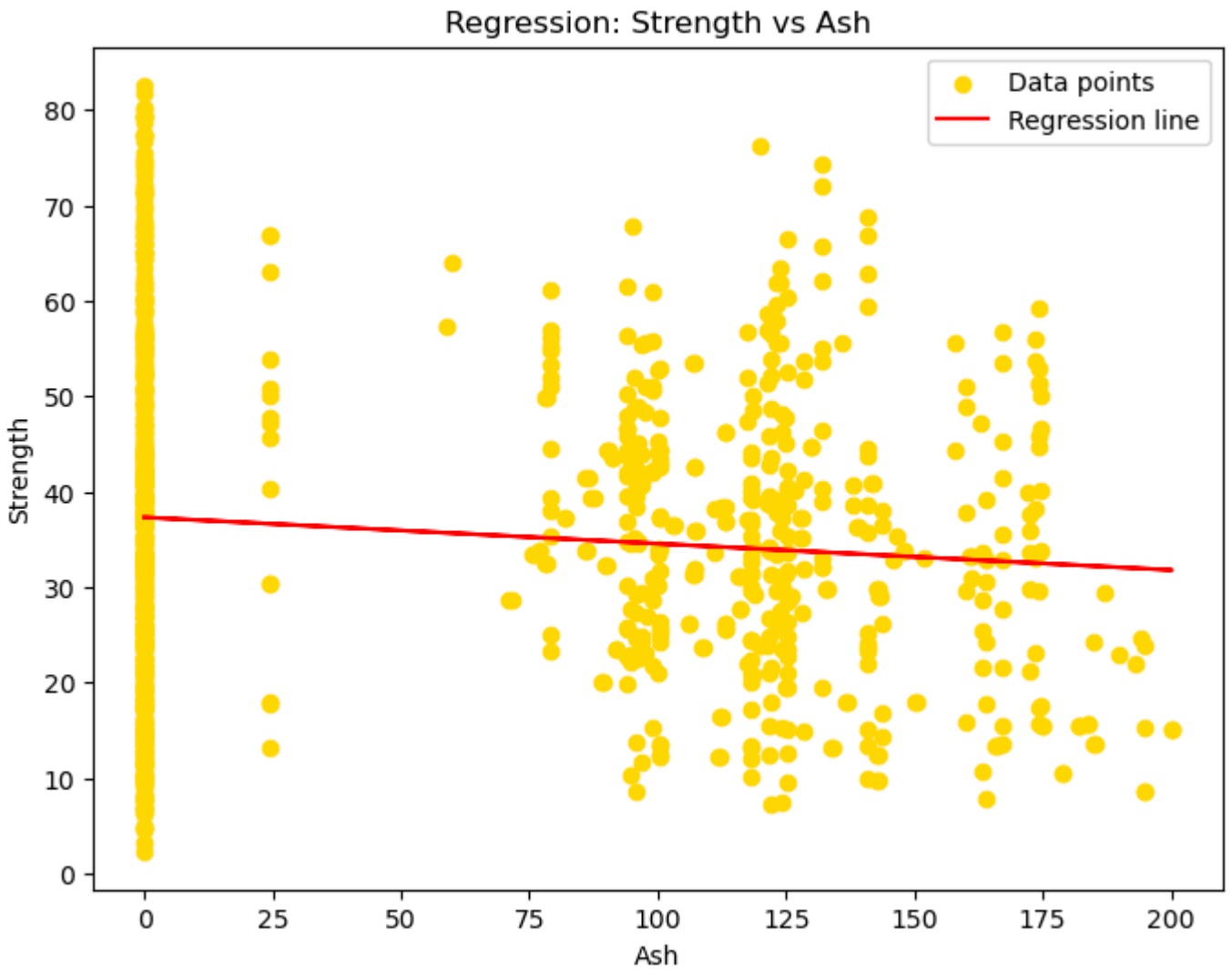

Regression: Strength vs Slag
Strength
Slag
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350
Data points
Regression line

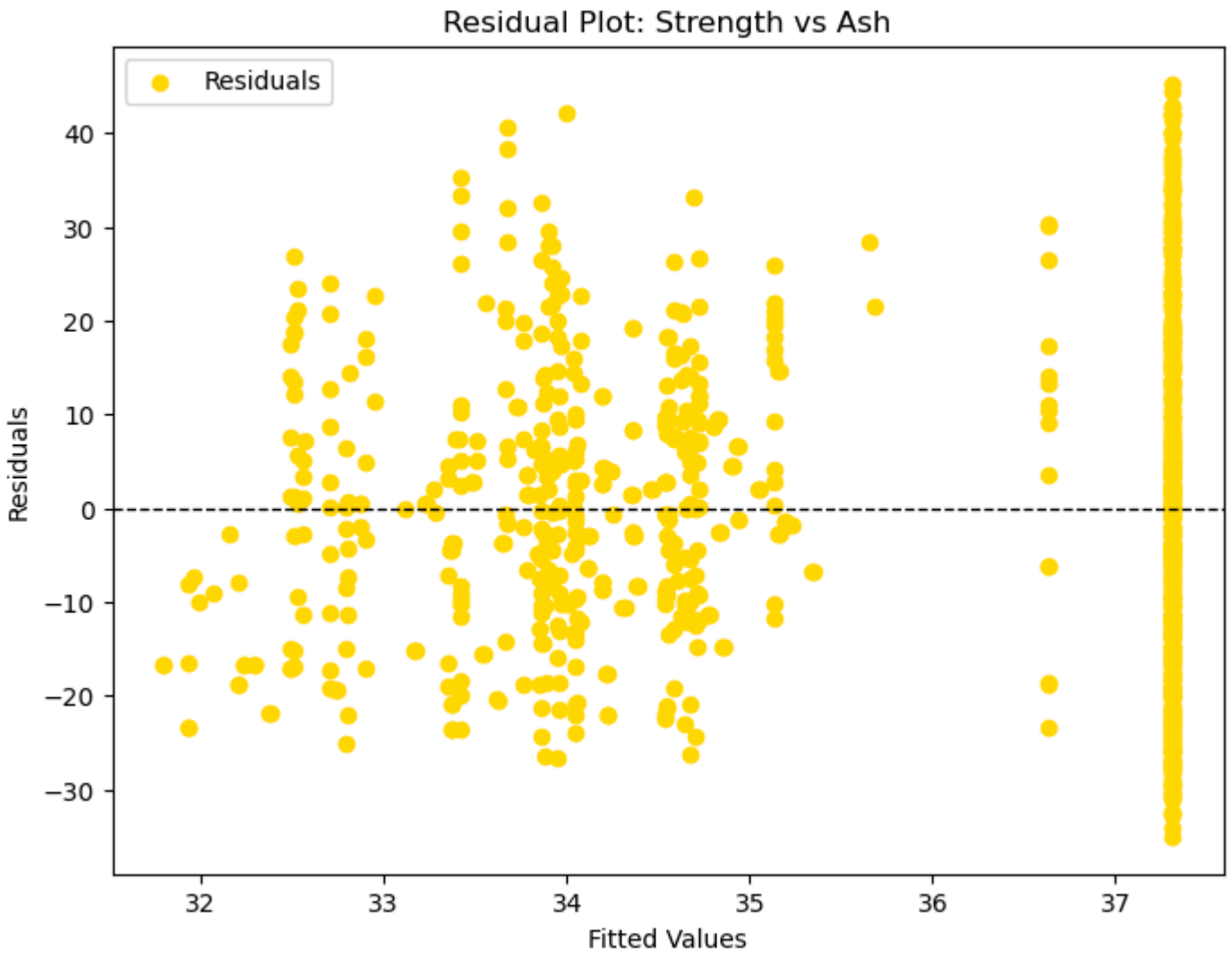


The slope of the regression line of Slag is almost flat, suggesting minor increase in strength with increase in slag content. Also slag is very widely spread across the line, which suggest `variability in the slag useage for mixture`. Also a `high number of mixing observations do not include slag` content in their mix, suggesting `less importance of slag for increasing strength` rather useage limited to increase in volume or any other parameters.

In [477…

```
X_ash = sm.add_constant(concrete_data['ash'])

model_ash = sm.OLS(y_strength, X_ash)
results_ash = model_ash.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['ash'], concrete_data['strength'], label='Data points', color='gold')
plt.plot(concrete_data['ash'], results_ash.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Ash')
plt.xlabel('Ash')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_ash.fittedvalues, results_ash.resid, label='Residuals', color='gold')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Ash')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```


Regression: Strength vs Ash
Data points
Regression line
Strength
Ash
0
10
20
30
40
50
60
70
80
0
25
50
75
100
125
150
175
200

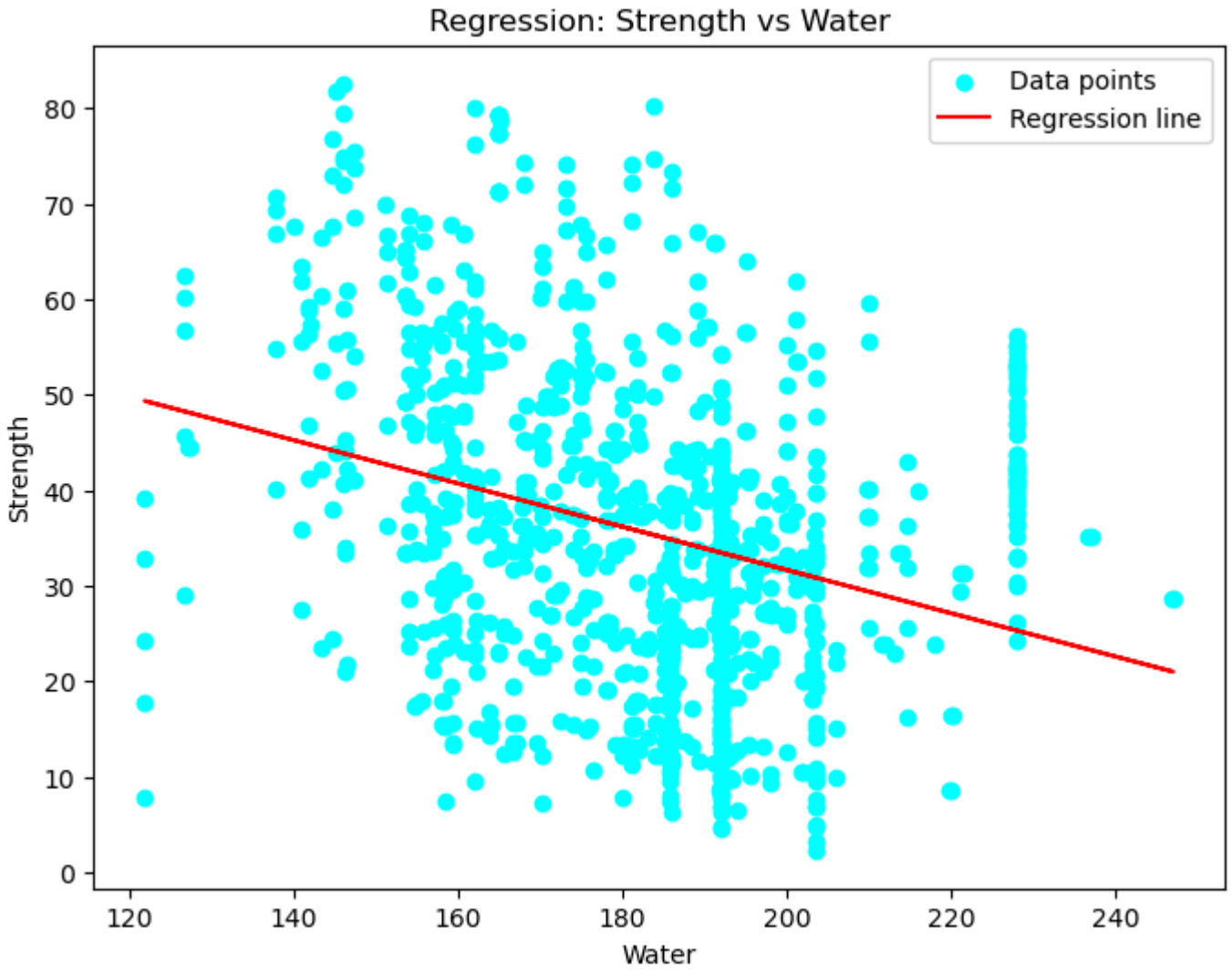


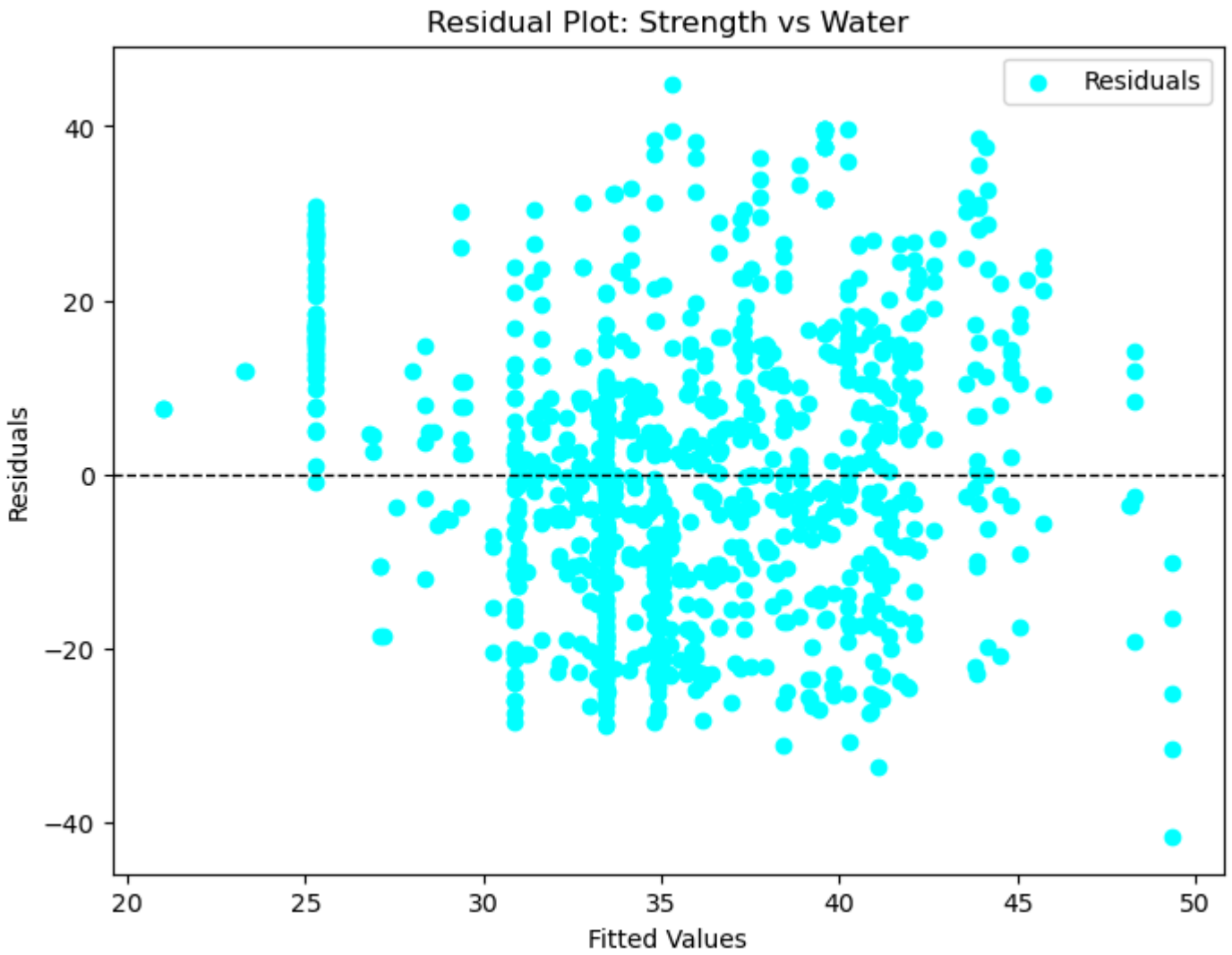
Here from this plot, we can interpret that Ash is weekly `negatively correlated` with the `strength of the Concrete`. Also, a lot of the observations contain 0 ash content, indicating not much was used in the mixture. Also, in the distribution we can see that there is almost no data for Ash content between `25-75`.

In [479…

```
X_water = sm.add_constant(concrete_data['water'])
```

```
model_water = sm.OLS(y_strength, X_water)
results_water = model_water.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['water'], concrete_data['strength'], label='Data points', color='cyan')
plt.plot(concrete_data['water'], results_water.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Water')
plt.xlabel('Water')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_water.fittedvalues, results_water.resid, label='Residuals', color='cyan')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Water')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```


Regression: Strength vs Water
Data points
Regression line
Strength
Water
0
10
20
30
40
50
60
70
80
120
140
160
180
200
220
240

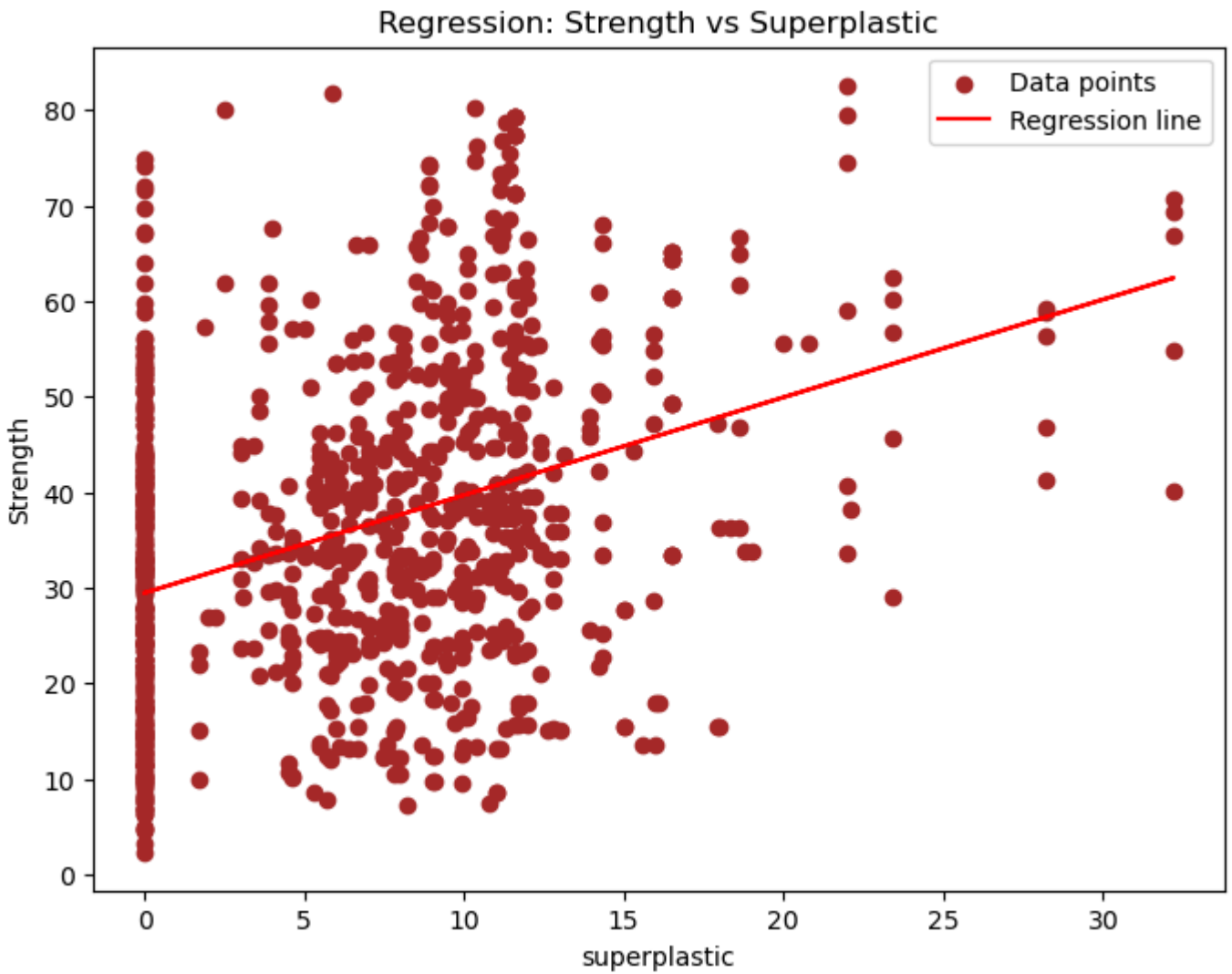


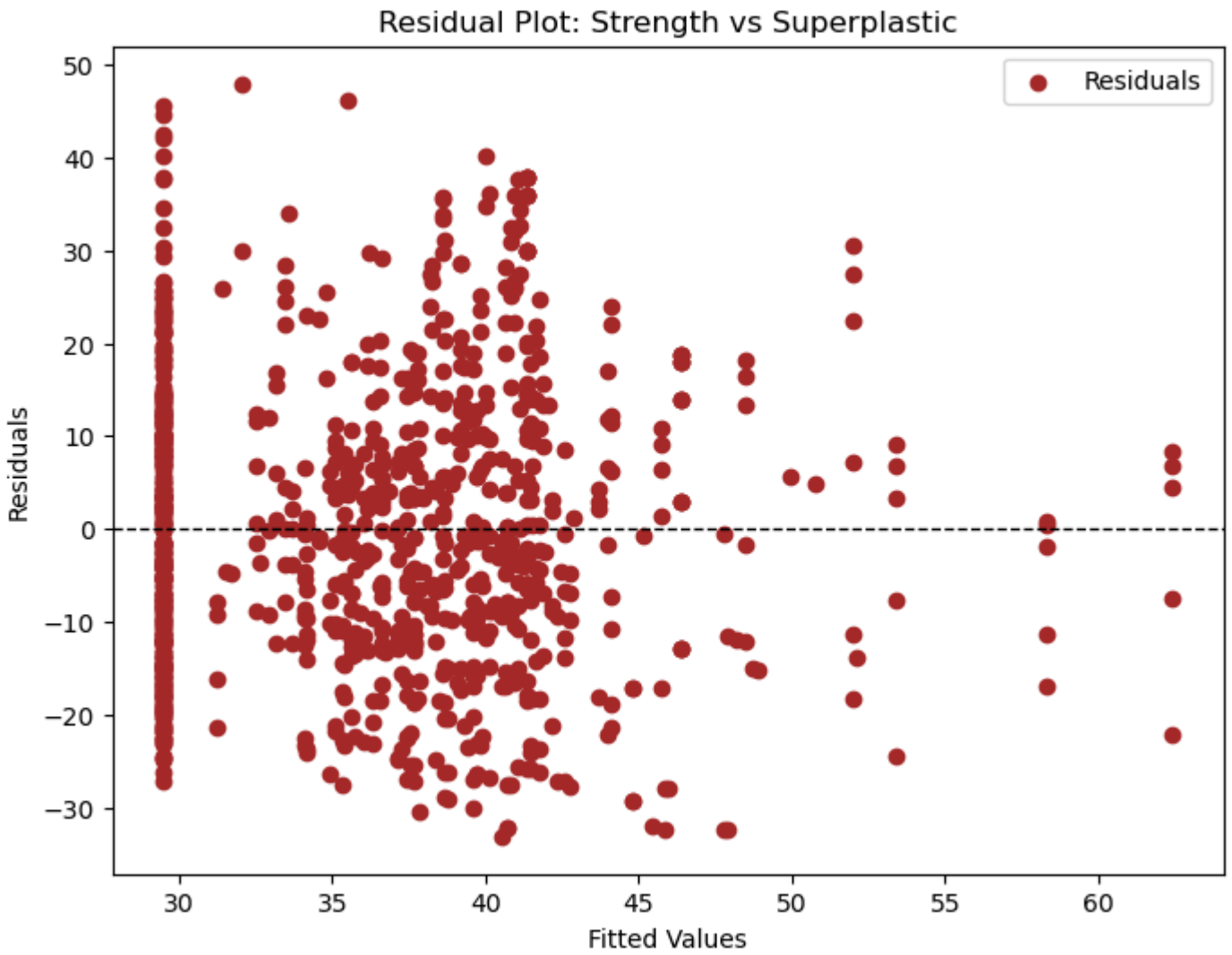
The Regression Plot shows a `highly negative trend` with an increase in water content. This shows that an increase in the content of water will decrease the strength of the concrete. The concrete will get diluted and not able to hold the components firmly which shows a decrease in strength. Also, the data is highly concentrated in between, showing missing values in starting and end.

In [481…

```
X_Superplastic = sm.add_constant(concrete_data['superplastic'])
```

```
model_Superplastic = sm.OLS(y_strength, X_Superplastic)
results_Superplastic = model_Superplastic.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['superplastic'], concrete_data['strength'], label='Data points', color='brown')
plt.plot(concrete_data['superplastic'], results_Superplastic.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Superplastic')
plt.xlabel('superplastic')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_Superplastic.fittedvalues, results_Superplastic.resid, label='Residuals', color='brown')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Superplastic')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```

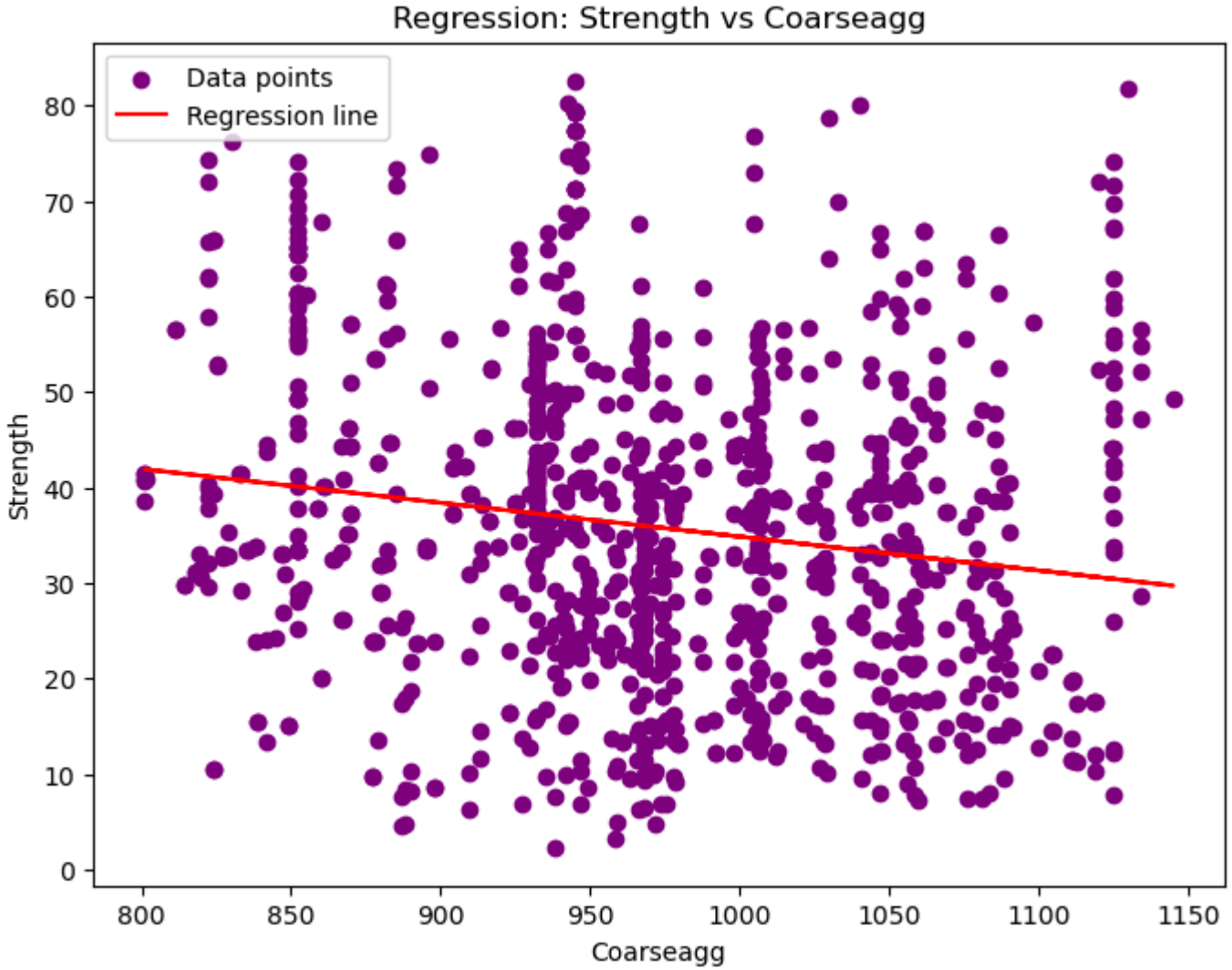

Regression: Strength vs Superplastic
Data points
Regression line
Strength
superplastic
0
10
20
30
40
50
60
70
80
0
5
10
15
20
25
30

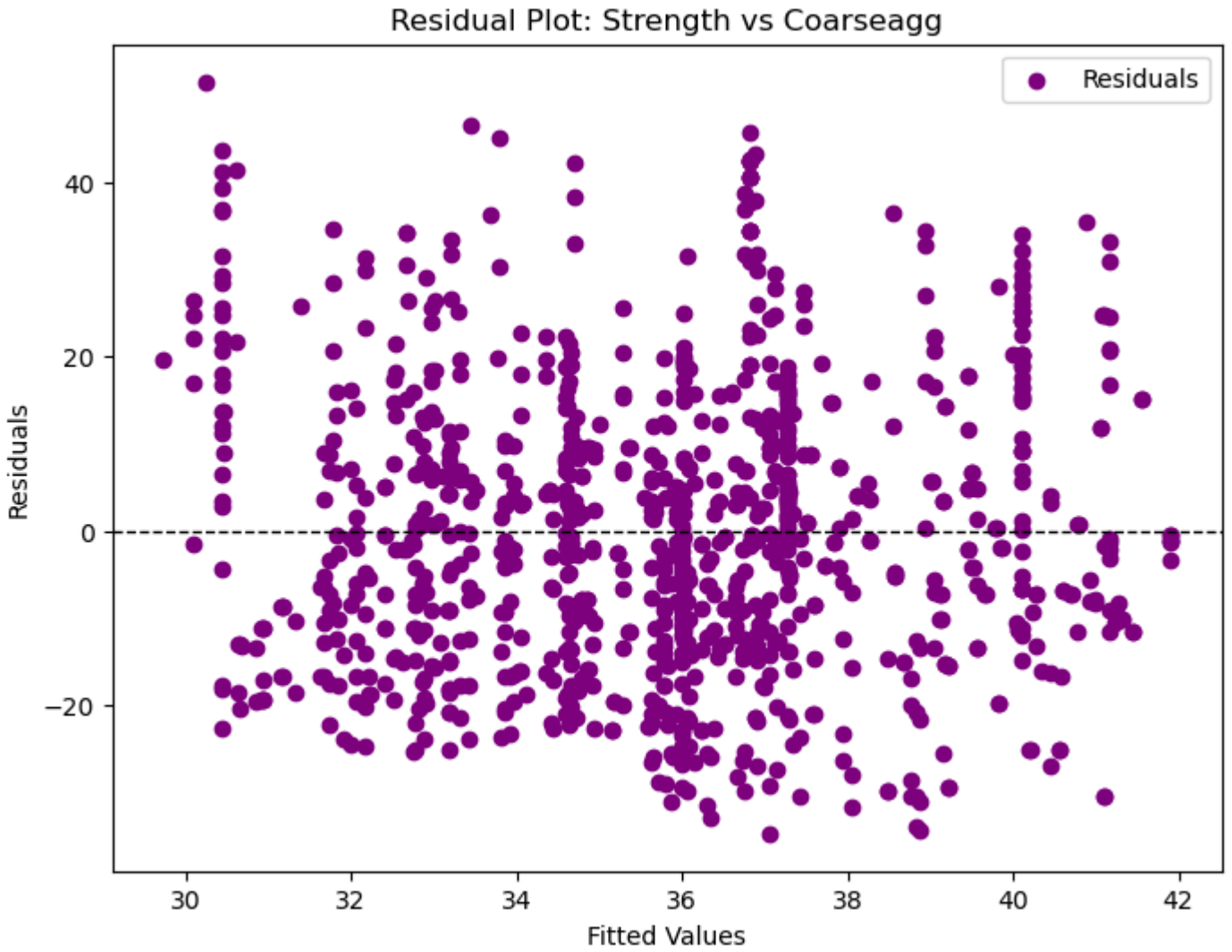


Here a `strong positive trend` is seen in between the Superplastic and the Strength of Concrete. This suggests that the superplastic can be used in higher amounts to increase the strength of the concrete, but some scatter suggests that this is not only due to an increase in superplastic, so it is collinear with any other predictor. Also, the data lags higher value observations, so our outputs might change seeing more higher data. The data for Residual is also evenly spread showing that the model captures the `variability effectively`.

In [483…

```
X_coarseagg = sm.add_constant(concrete_data['coarseagg'])

model_coarseagg = sm.OLS(y_strength, X_coarseagg)
results_coarseagg = model_coarseagg.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['coarseagg'], concrete_data['strength'], label='Data points', color='purple')
plt.plot(concrete_data['coarseagg'], results_coarseagg.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Coarseagg')
plt.xlabel('Coarseagg')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_coarseagg.fittedvalues, results_coarseagg.resid, label='Residuals', color='purple')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Coarseagg')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```

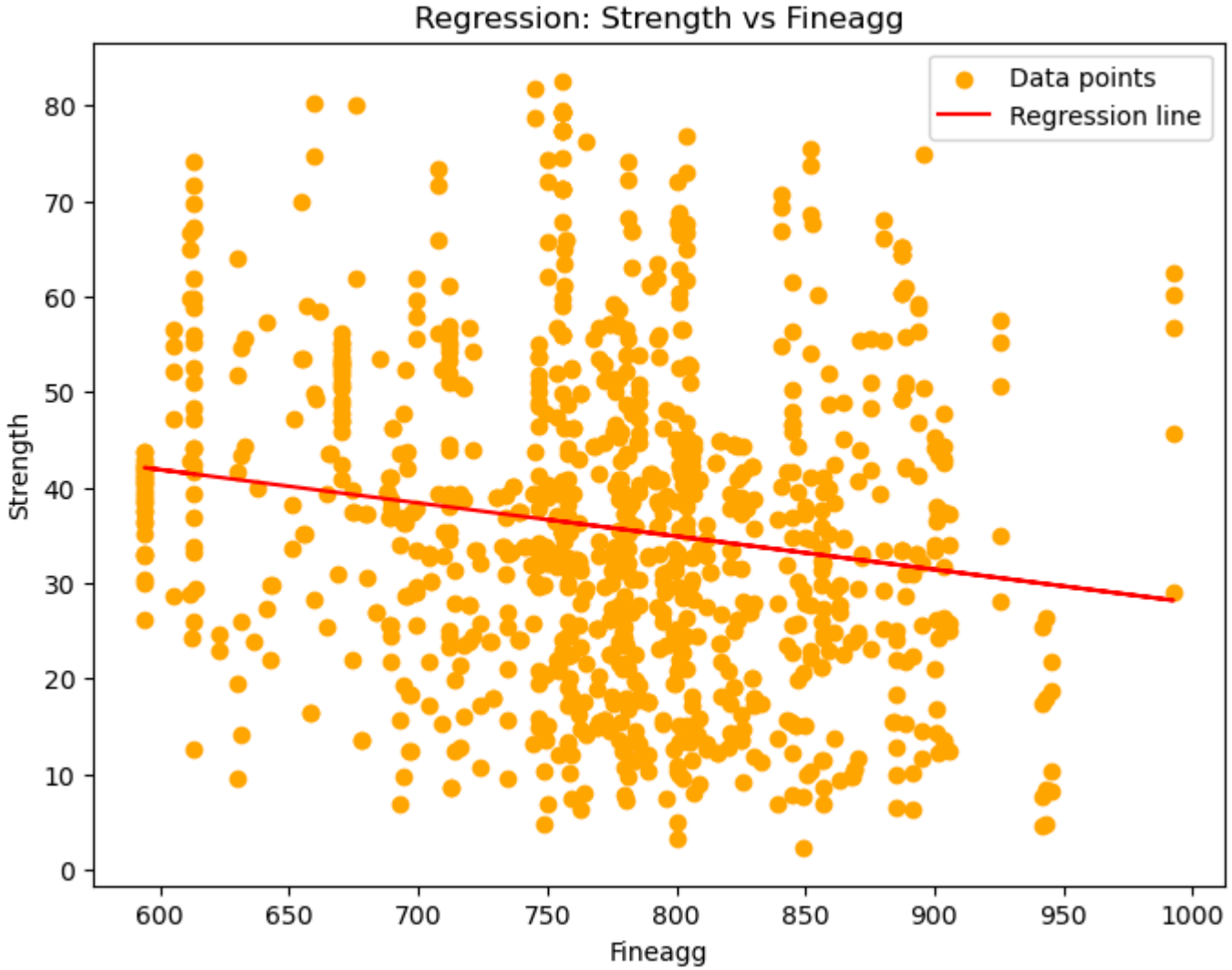

Regression: Strength vs Coarseagg
Data points
Regression line
Strength
Coarseagg
0
10
20
30
40
50
60
70
80
800
850
900
950
1000
1050
1100
1150

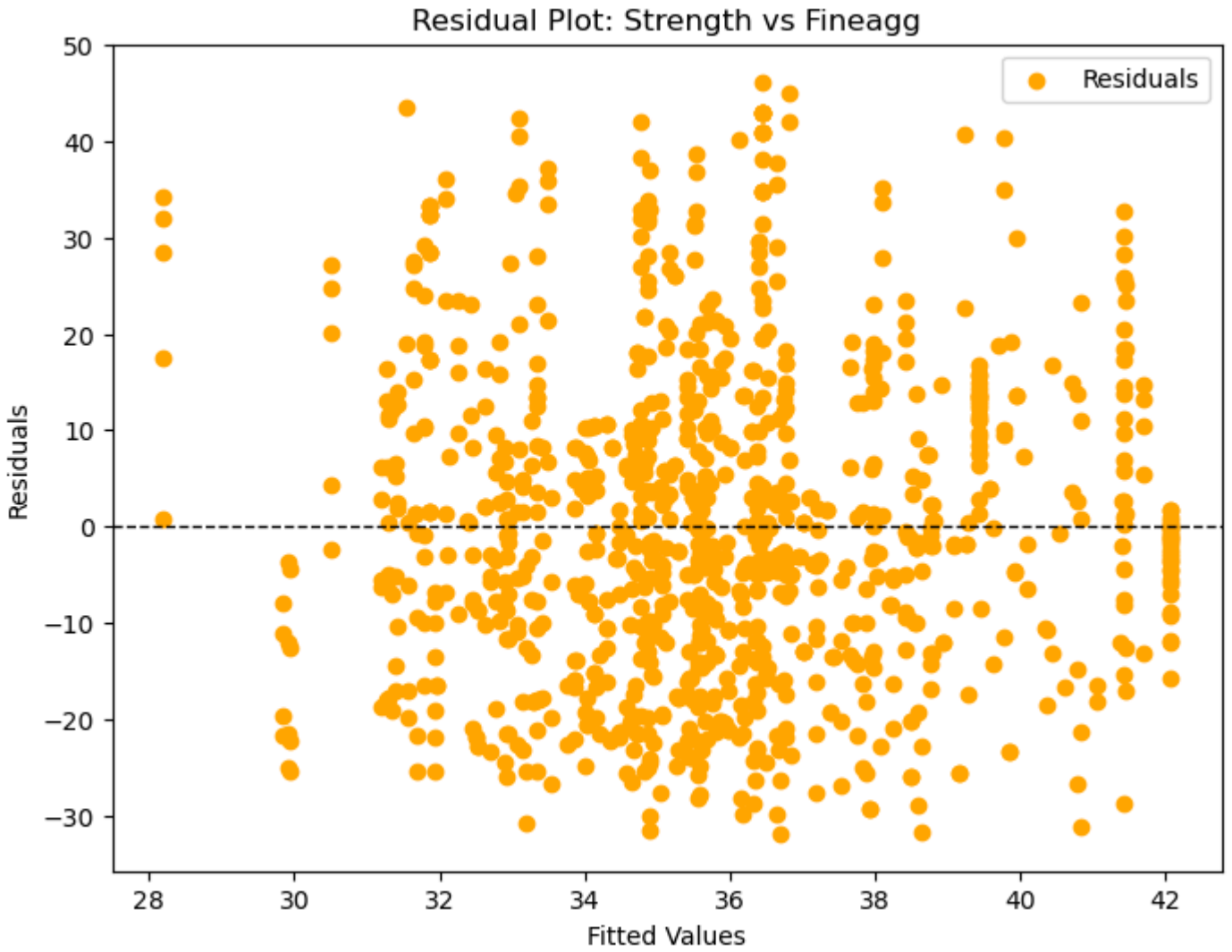


There is a weak `negative correlation` between the Strength of Concrete and the quantity of Coarseagg. The data is distributed on both the positive and negative side of the regression line, showcasing that the data is variable and random. So it can also be interpreted that coarseagg might not directly influence the strength of the concrete, but other variable combinations can help improve the strength and maintain binding in the concrete which can firmly hold the concrete.

In [485…

```
X_fineagg = sm.add_constant(concrete_data['fineagg'])

model_fineagg = sm.OLS(y_strength, X_fineagg)
results_fineagg = model_fineagg.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['fineagg'], concrete_data['strength'], label='Data points', color= 'orange')
plt.plot(concrete_data['fineagg'], results_fineagg.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Fineagg')
plt.xlabel('Fineagg')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_fineagg.fittedvalues, results_fineagg.resid, label='Residuals', color='orange')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Fineagg')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```


Regression: Strength vs Fineagg
Data points
Regression line
Strength
Fineagg
0
10
20
30
40
50
60
70
80
600
650
700
750
800
850
900
950
1000



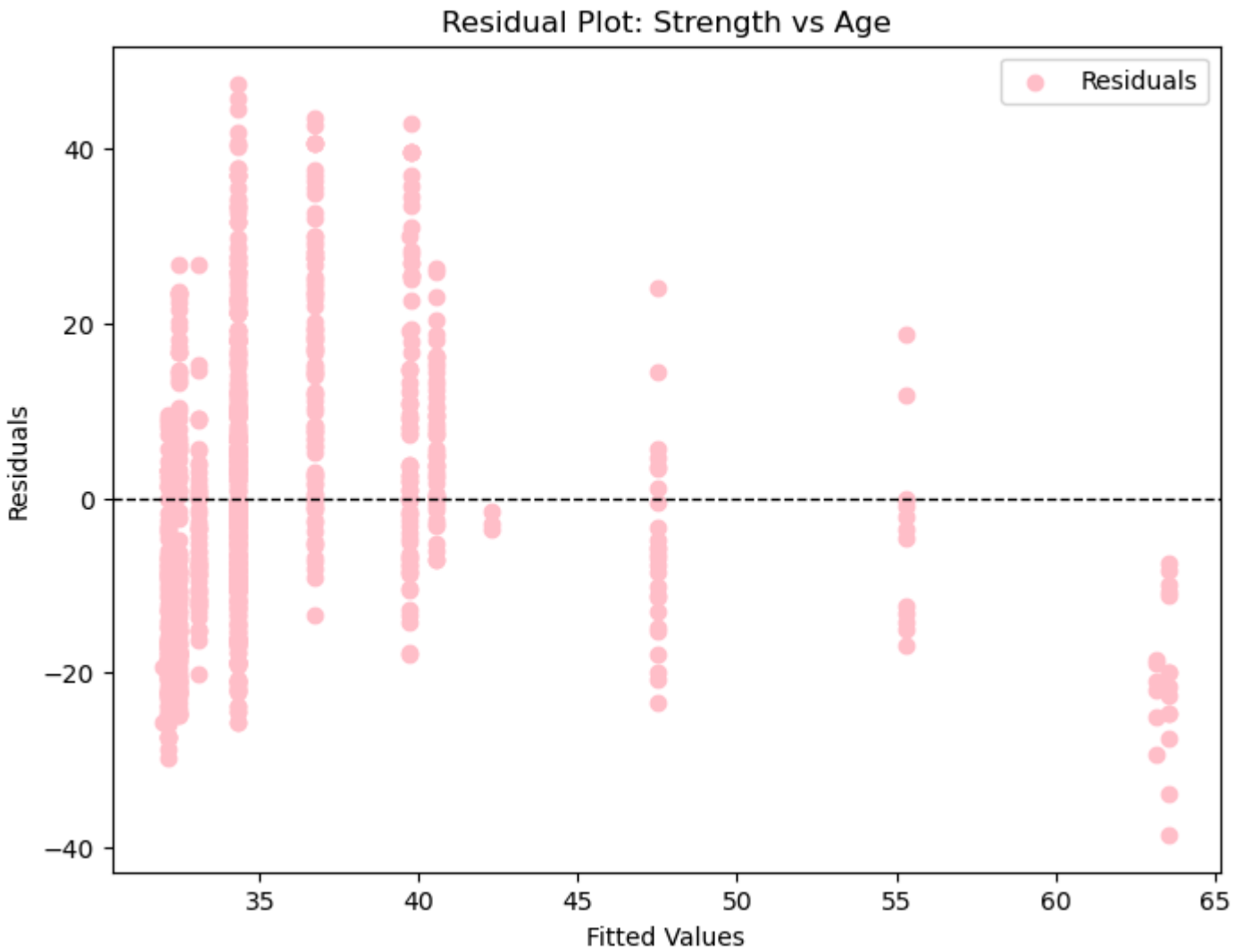
This negative slope suggests a `weak correlation` between the Fineagg and the Strength of the Concrete. The data is a little concentric in between values suggesting, major observations using fineagg as a component, as it binds the other materials better. Fineagg and Coarseagg can also be used in combination to increase the strength and also provide volume to the concrete.

In [487…

```
X_age = sm.add_constant(concrete_data['age'])
```

```
model_age = sm.OLS(y_strength, X_age)
results_age = model_age.fit()

plt.figure(figsize=(8, 6))
plt.scatter(concrete_data['age'], concrete_data['strength'], label='Data points', color= 'pink')
plt.plot(concrete_data['age'], results_age.fittedvalues, color='red', label='Regression line')
plt.title('Regression: Strength vs Age')
plt.xlabel('Age')
plt.ylabel('Strength')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(results_age.fittedvalues, results_age.resid, label='Residuals', color='pink')
plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.title('Residual Plot: Strength vs Age')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.legend()
plt.show()
```

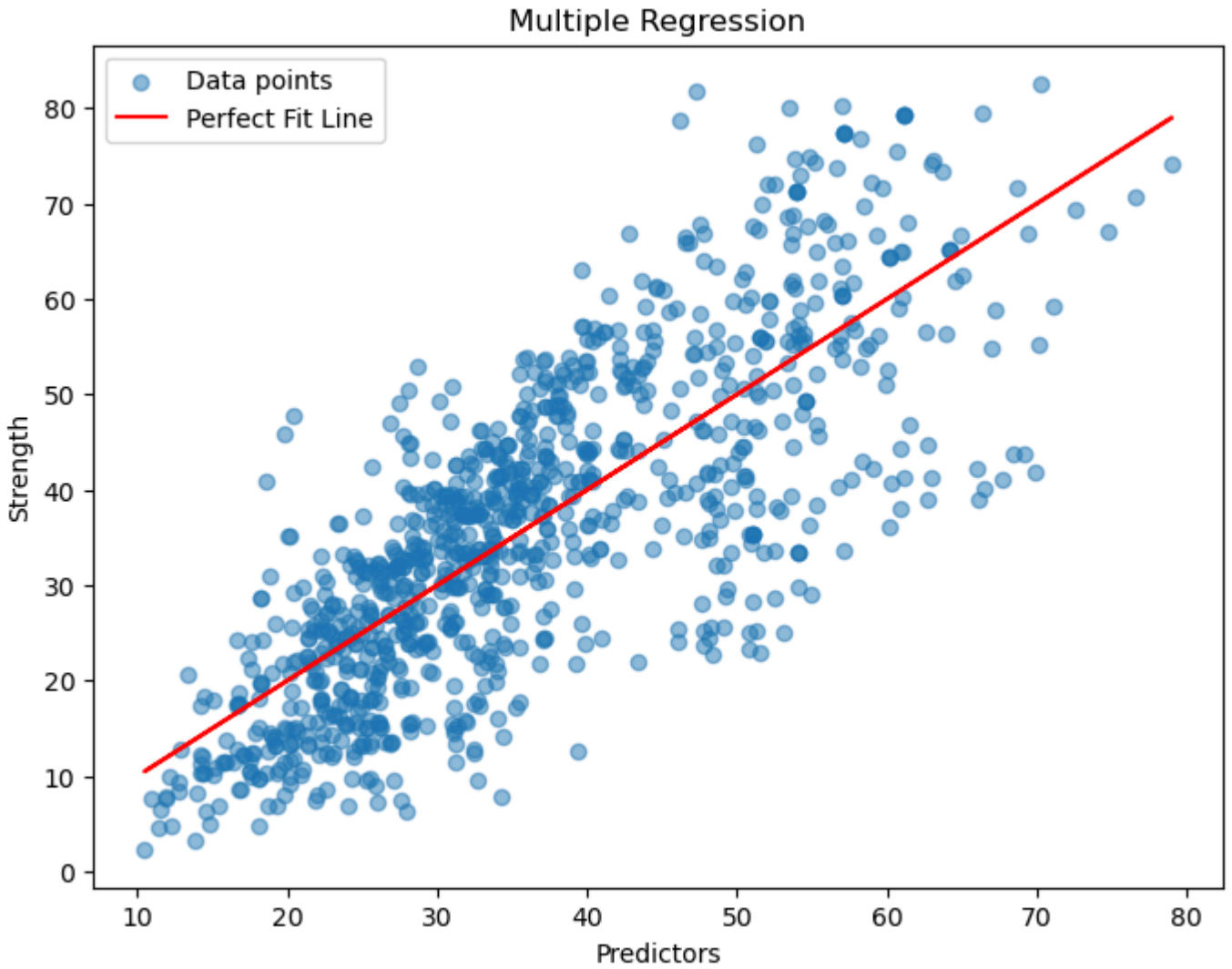

Regression: Strength vs Age
Data points
Regression line
Strength
Age
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350



Here we can see a `strong positive trend` between the Age of Testing and the Strength of Concrete. This can be explained as the number of days the concrete is left for binding, the `concrete gains strength over time`. This is why there is a positive trend between the two, but over time the strength of concrete reaches its maximum so after testing after reaching the strength does not increase, which can seen as the data over the right is below the line, and in future, if plotted a polynomial regression line a decreasing trend could be seen. Also, the strength testing of Concrete is not done continuously, as on a few days there are no observations, and on a particular day in starting the strength is variable suggesting different combinations can show different strengths.

In [489…

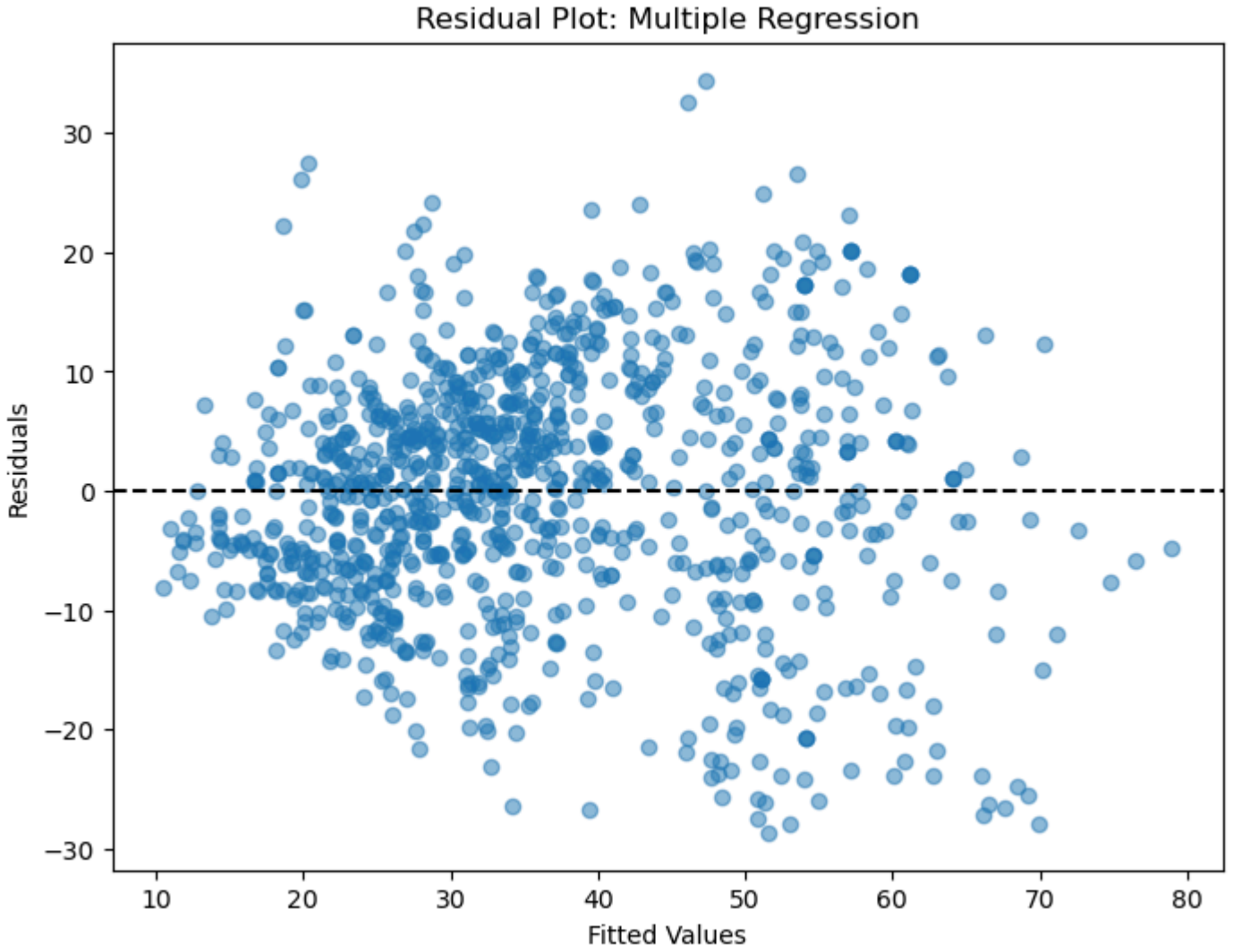
```
predictors = ['cement', 'slag', 'ash', 'water', 'superplastic', 'coarseagg', 'fineagg', 'age']
X = concrete_data[predictors]
X = sm.add_constant(X)
model = sm.OLS(y, X)
results = model.fit()

plt.figure(figsize=(8, 6))
plt.scatter(results.fittedvalues, y, alpha=0.5, label='Data points')
plt.plot(results.fittedvalues, results.fittedvalues, color='red', label='Perfect Fit Line')
plt.title('Multiple Regression')
plt.xlabel('Predictors')
plt.ylabel('Strength')
plt.legend()
plt.show()
```



This graph represents the spread of points where the `X axis is the fitted values of all the predictors` v/s, and the `Y axis represents the Strength`. The relationship is positively correlated. We can see that the strength increases as we add predictors of higher value. Also, the variability in strength is less for lower values of predictors and is higher for higher values of predictors. The `points are much concentrated` on the value of predictors between 10-40. Fitting a higher order regression model can also help capture more data and reduce the RSS. For predicting strength greater than 60, the variability is higher, and the data is also limited for the same.

In [491…

```
plt.figure(figsize=(8, 6))
plt.scatter(results.fittedvalues, results.resid, alpha=0.5)
plt.axhline(0, color='black', linestyle='--')
plt.title('Residual Plot: Multiple Regression')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.show()
```



Here in the residual plot, we can see, that there are some `concentrated points in the 20-40 region`. Also as the fitted values of the predictors increase the residual value also increases. There are also some outliers or `higher residual terms after 50`. This higher residual terms can be fitted to the model by incuding higher order regression.

In [493…

```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               strength   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.613
Method:                 Least Squares   F-statistic:                     204.3
Date:                Mon, 02 Dec 2024   Prob (F-statistic):          6.29e-206
Time:                        23:49:37   Log-Likelihood:                -3869.0
No. Observations:                1030   AIC:                             7756.
Df Residuals:                    1021   BIC:                             7800.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -23.3312     26.586     -0.878      0.380     -75.500      28.837
cement         0.1198      0.008     14.113      0.000       0.103       0.136
slag           0.1039      0.010     10.247      0.000       0.084       0.124
ash            0.0879      0.013      6.988      0.000       0.063       0.113
water         -0.1499      0.040     -3.731      0.000      -0.229      -0.071
superplastic   0.2922      0.093      3.128      0.002       0.109       0.476
coarseagg      0.0181      0.009      1.926      0.054      -0.000       0.037
fineagg        0.0202      0.011      1.887      0.059      -0.001       0.041
age            0.1142      0.005     21.046      0.000       0.104       0.125
==============================================================================
Omnibus:                        5.378   Durbin-Watson:                   1.870
Prob(Omnibus):                  0.068   Jarque-Bera (JB):                5.304
Skew:                          -0.174   Prob(JB):                       0.0705
Kurtosis:                       3.045   Cond. No.                     1.06e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.06e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

This result table gives us an in depth summary of the multiple regression model -

R^2 value - 0.616 which means `61.6 percent of the variability in strength is explained by the model`. This value can be made better by removing unnecessary predictors, adding polynomial terms, or checking for multicollinearity.

`P Values (Preferred = X < 0.05)` - According to the null hypothesis test, we can conclude that predictors having P values > 0.05 are not very significant, and removing them would make the model more interpretable.

Cement, Slag, Ash, Water, Superplastic, and Age are the significant variables and Coarseagg and Fineagg are not very significant.

In [495…
```
vif_data = pd.DataFrame()
vif_data['predictors'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

```
    predictors          VIF
0        const  6731.811539
1       cement     7.488944
2         slag     7.276963
3          ash     6.170634
4        water     7.003957
5  superplastic     2.963776
6    coarseagg     5.074617
7      fineagg     7.005081
8          age     1.118367
```

Validation Indication Factor (VIF) Analysis gives us the multicollinearity between the predictors. The higher the value for each predictor the higher their multicollinearity (VIF>5). Here we can see from our results that `Cement, Slag, Ash, Water, Coarseagg, and Fineagg` have higher values of VIF, which suggest they are collinear and harm our regression model. Adjusting these predictors can maintain the R^2 values and also decrease the model needs to see for analysis which results in the usage of less computing power. `Age and Superplastic` has the least VIF meaning they are not collinear and must be kept in the multiple linear regression model.

Cross Validation method with Validation Set Approach and Train/Test Split

Here we will perform the two cross-validation methods - Validation Set Approach and and K Fold Cross Validation to find the MSE for each predictor with Strength and higher degree predictors.

In [499…
```
train_data, test_data = train_test_split(concrete_data, test_size=0.3, random_state=42)

def eval_mse(degree, predictor, response, train, test):
    X_train = np.vander(train[predictor], N=degree + 1, increasing=True)
```

```
    X_test = np.vander(test[predictor], N=degree + 1, increasing=True)

    y_train = train[response]
    y_test = test[response]

    model = sm.OLS(y_train, X_train)
    results = model.fit()

    predictions = results.predict(X_test)
    mse = np.mean((y_test - predictions) ** 2)

    return mse

response = 'strength'

mse_results_all = {}

for predictor in predictors:
    mse_results_all[predictor] = {}
    for degree in range(1, 4):
        mse_results_all[predictor][f"Degree {degree}"] = eval_mse(degree, predictor, response, train_data, test_data)

print("Validation Set MSE for Polynomial Fits (All Predictors):")
for predictor, mse_values in mse_results_all.items():
    print(f"\nPredictor: {predictor}")
    for degree, mse in mse_values.items():
        print(f"  {degree}: {mse:.4f}")
```

```
Validation Set MSE for Polynomial Fits (All Predictors):

Predictor: cement
  Degree 1: 206.3869
  Degree 2: 207.2340
  Degree 3: 207.2572

Predictor: slag
  Degree 1: 262.2533
  Degree 2: 264.8401
  Degree 3: 265.0710

Predictor: ash
  Degree 1: 266.5704
  Degree 2: 261.3579
  Degree 3: 261.3802

Predictor: water
  Degree 1: 248.2029
  Degree 2: 243.5102
  Degree 3: 227.4783

Predictor: superplastic
  Degree 1: 230.9562
  Degree 2: 230.8715
  Degree 3: 233.7906

Predictor: coarseagg
  Degree 1: 259.1197
  Degree 2: 254.2319
  Degree 3: 250.1695

Predictor: fineagg
  Degree 1: 258.5902
  Degree 2: 257.1306
  Degree 3: 255.7456

Predictor: age
  Degree 1: 234.1013
  Degree 2: 194.6891
  Degree 3: 173.4661
```

Here we have split the data in a 70/30 split, where `70% is the training data and 30% is test data` for the Validation Set Approach, where we do split the data and fit the test data on the linear model to calculate its Test MSE. Here we can see two trends where increasing the degree of predictor if the TEST MSE decreases means that the curve fits the data well and can be used in the future, this trend is seen in `Age`. The other trend is that the TEST MSE is decreasing then we can see that the curve is overfitting the data and thus is not a good sign for the model, a simpler model must be used which would explain the model much better. Such a trend can be seen in `Strength, Slag, Superplastic.`

Cross Validation method with K-Fold approach

In [502…

```
class StatsmodelsRegressor(BaseEstimator, RegressorMixin):
    def __init__(self, degree, predictor):
        self.degree = degree
        self.predictor = predictor

    def fit(self, X, y):
        self.X_ = np.vander(X[self.predictor], N=self.degree + 1, increasing=True)
        self.y_ = y
        self.model_ = sm.OLS(self.y_, self.X_).fit()
        return self

    def predict(self, X):
        X_poly = np.vander(X[self.predictor], N=self.degree + 1, increasing=True)
        return self.model_.predict(X_poly)

    def score(self, X, y):
        predictions = self.predict(X)
        return -np.mean((y - predictions) ** 2)

kf = KFold(n_splits=10, shuffle=True, random_state=42)

cv_results_all = {}

for predictor in predictors:
    cv_results_all[predictor] = {}
    for degree in range(1, 4):
        regressor = StatsmodelsRegressor(degree=degree, predictor=predictor)
```

```
        mse_scores = cross_val_score(regressor, concrete_data[[predictor]], concrete_data[response], cv=kf,
                                     scoring='neg_mean_squared_error')
        mean_mse = -np.mean(mse_scores)
        cv_results_all[predictor][f"Degree {degree}"] = mean_mse

print("K-Fold Cross-Validation MSE for Polynomial Fits (All Predictors):")
for predictor, mse_values in cv_results_all.items():
    print(f"\nPredictor: {predictor}")
    for degree, mse in mse_values.items():
        print(f"  {degree}: {mse:.4f}")
```

```
K-Fold Cross-Validation MSE for Polynomial Fits (All Predictors):

Predictor: cement
  Degree 1: 210.2931
  Degree 2: 211.5878
  Degree 3: 212.3283

Predictor: slag
  Degree 1: 274.6702
  Degree 2: 269.3164
  Degree 3: 267.7379

Predictor: ash
  Degree 1: 276.7642
  Degree 2: 274.6721
  Degree 3: 275.0403

Predictor: water
  Degree 1: 256.9058
  Degree 2: 243.2755
  Degree 3: 232.2017

Predictor: superplastic
  Degree 1: 242.7176
  Degree 2: 242.8886
  Degree 3: 243.5069

Predictor: coarseagg
  Degree 1: 272.1696
  Degree 2: 269.6776
  Degree 3: 267.8712

Predictor: fineagg
  Degree 1: 271.7332
  Degree 2: 271.6017
  Degree 3: 271.6226

Predictor: age
  Degree 1: 249.5295
  Degree 2: 208.4046
  Degree 3: 192.1845
```

In K Fold Cross Validation, we divide the dataset into K sets (here K = 10), here the K-1 set is used as the training data set and K^{th} is used as the Test data set and we also have the shuffled the dataset to keep randomness. Here we can see the Test MSE for different order terms for each predictor. Here the Test MSE increases for `Strength and Superplastic` showcasing overfitting for higher order terms. Other variables have a decrease in Test Mse showcasing that higher-order terms fit the observation better, this interpretation is valid using both the K-Fold and Validation Set Approach Cross Validation.

Forward Subset Selection

In [505…

```
design = MS(X.columns).fit(X)
Y = np.array(y)

sigma2 = sm.OLS(Y, X).fit().scale

def nCp(sigma2, estimator, X, Y):
    """Negative Cp statistic"""
    n, p = X.shape
    Yhat = estimator.predict(X)
    RSS = np.sum((Y - Yhat) ** 2)
    return -(RSS + 2 * p * sigma2) / n

neg_Cp = partial(nCp, sigma2)
strategy = Stepwise.first_peak(design, direction='forward', max_terms=len(design.terms))

stepwise_model = sklearn_selected(sm.OLS, strategy, scoring=neg_Cp)
stepwise_model.fit(X, Y)

print("Selected predictors:", stepwise_model.selected_state_)
```

```
Selected predictors: ('age', 'ash', 'cement', 'slag', 'superplastic', 'water')
```

The Forward Subset Selection is an iterative process of adding and removing predictors to form a subset of predictors that are the most influential on the response. Subset selection provides a balance between the number of predictors and the performance of the model, as having a balanced Flexible as well as Interpretable model is important for the overall model. The results of Subset Selection are the most influential predictors set which are - `Age, Ash, Cement, Slag, Superplastic, Water` predictors must be used for the multiple linear

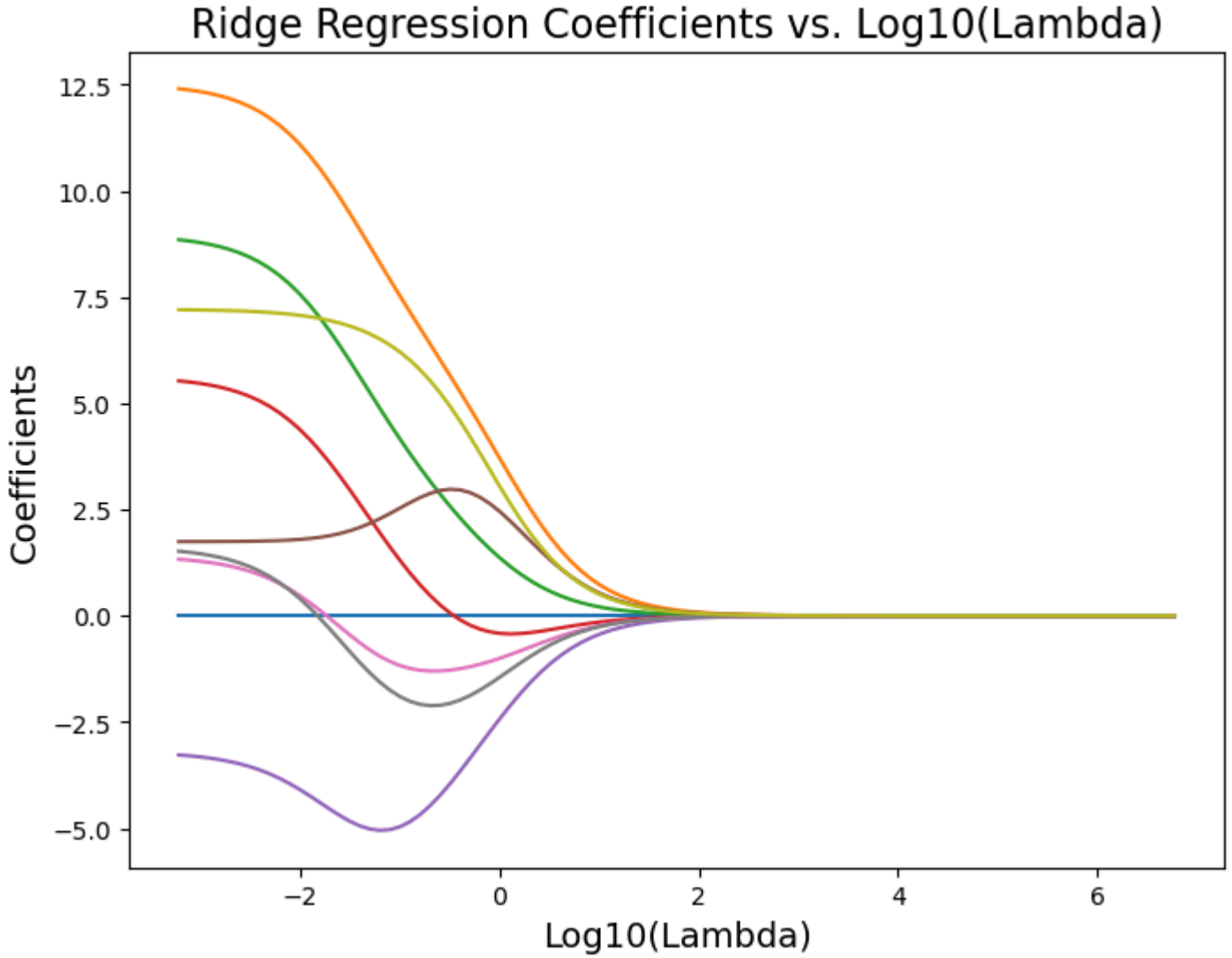
regression model. Age is used here as it has a strong linear relationship with Strength, and so does Cement, Ash, Slag, and Superplastic. Some nonsignificant variables like Coarseagg and Finegg are not selected in the Subset.

Ridge Regression and Lasso

In [508…

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

lambdas = 10**np.linspace(8, -2, 100) / y.std()
ridge_model = ElasticNetCV(l1_ratio=0, alphas=lambdas, cv=5)
ridge_model.fit(X_scaled, y)

print(f"Best Lamda for Ridge: {ridge_model.alpha_}")

soln_array = ridge_model.path(X_scaled, y, l1_ratio=0, alphas=lambdas)[1]

plt.figure(figsize=(8, 6))
plt.plot(np.log10(ridge_model.alphas_), soln_array.T)
plt.xlabel('Log10(Lambda)', fontsize=14)
plt.ylabel('Coefficients', fontsize=14)
plt.title('Ridge Regression Coefficients vs. Log10(Lambda)', fontsize=16)
plt.show()

y_pred_ridge = ridge_model.predict(X_scaled)
ridge_mse = np.mean((y - y_pred_ridge) ** 2)
print(f"Ridge MSE: {ridge_mse}")
warnings.filterwarnings('ignore')
```

```
Best Lamda for Ridge: 0.0005985965797148694
```



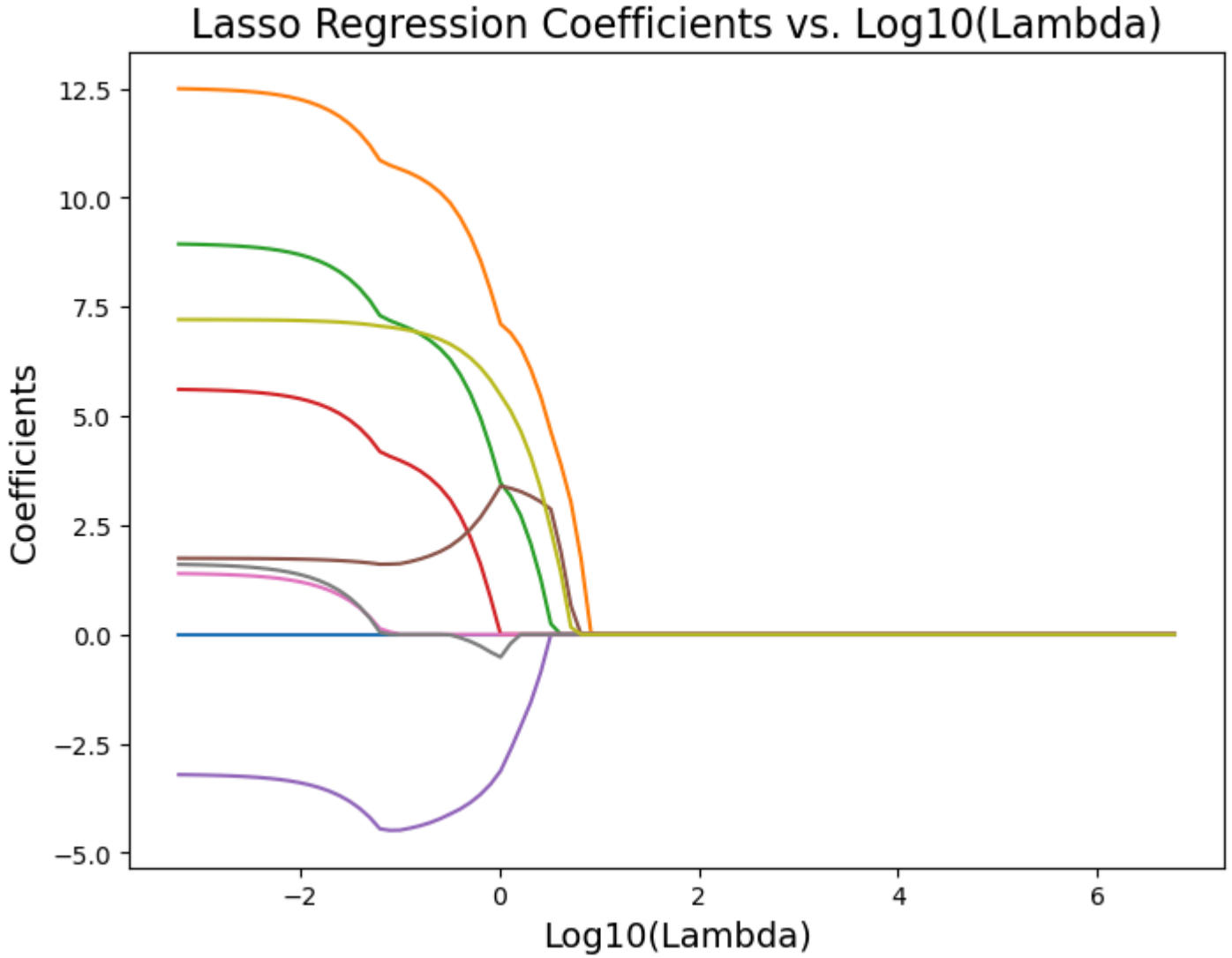

```
Ridge MSE: 107.19898806446675
```

With the increase in the value of Lambda, the model shrinks towards zero, this can help reduce the overall variance in the model. The model when increasing the value of Lambda moves from overfitting to underfitting of the predictors. The best lambda is the lambda at which the model balances overfitting and underfitting. Ridge regression stabilizes the multicollinearity among the predictors like Cement, Slag, and Water. Calculating the MSE can give us an in-depth idea about the model's accuracy and interpretability.

In [510…

```
lasso_model = ElasticNetCV(l1_ratio=1, alphas=lambdas, cv=5)
lasso_model.fit(X_scaled, y)

print(f"Best Lamda for Lasso: {lasso_model.alpha_}")

soln_array = lasso_model.path(X_scaled, y, l1_ratio=1, alphas=lambdas)[1]

plt.figure(figsize=(8, 6))
plt.plot(np.log10(lasso_model.alphas_), soln_array.T)
plt.xlabel('Log10(Lambda)', fontsize=14)
plt.ylabel('Coefficients', fontsize=14)
plt.title('Lasso Regression Coefficients vs. Log10(Lambda)', fontsize=16)
plt.show()

y_pred_lasso = lasso_model.predict(X_scaled)
lasso_mse = np.mean((y - y_pred_lasso) ** 2)
print(f"Lasso MSE: {lasso_mse}")
```

```
Best Lamda for Lasso: 0.0007553432142929846
```



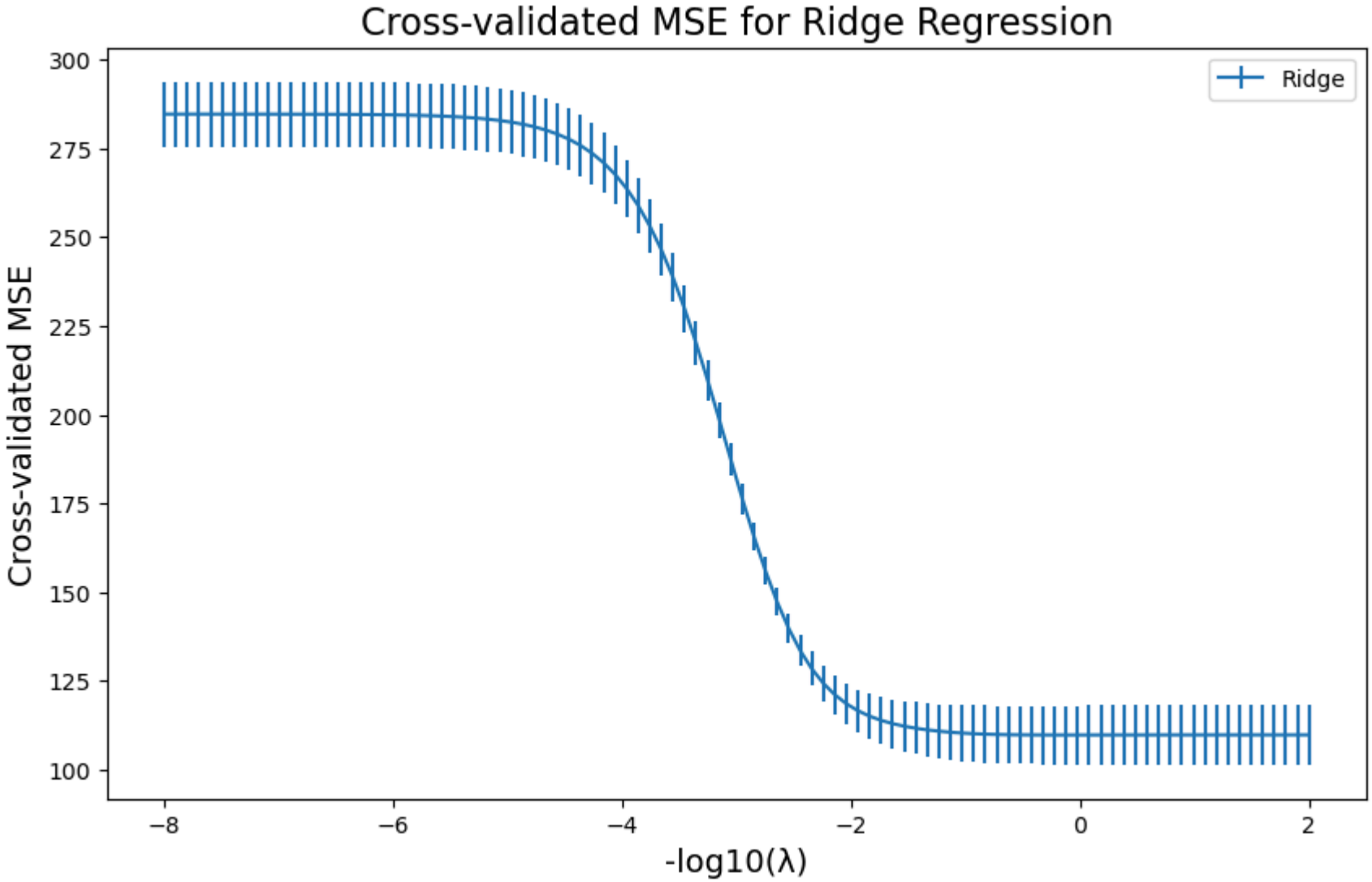

```
Lasso MSE: 107.19729748733856
```

Lasso is used to select significant predictors and eliminate irrelevant ones. Here we have used the 5-fold cross-validation to find the best value of Lamba which reduces the overfitting. Here from the lasso graph, we can see some coefficients shrink to 0 faster than others, the faster the drop predicts there is little contribution of such predictors.

In [512…

```
ridge = Ridge()
param_grid = {'alpha': np.logspace(-2, 8, 100)}
ridge_cv = GridSearchCV(ridge, param_grid, scoring='neg_mean_squared_error', cv=5, n_jobs=-1)
ridge_cv.fit(X_train_scaled, y_train)

best_lambda_ridge = ridge_cv.best_params_['alpha']
cv_results = ridge_cv.cv_results_

print(f"Best λ for Ridge Regression: {best_lambda_ridge}")

mean_mse = -cv_results['mean_test_score']
std_mse = cv_results['std_test_score']

plt.figure(figsize=(10, 6))
plt.errorbar(-np.log10(param_grid['alpha']), mean_mse, yerr=std_mse / np.sqrt(5), label='Ridge')
plt.xlabel('-log10(λ)', fontsize=14)
plt.ylabel('Cross-validated MSE', fontsize=14)
plt.title('Cross-validated MSE for Ridge Regression', fontsize=16)
plt.legend()
plt.show()

y_pred_ridge = ridge_cv.best_estimator_.predict(X_test_scaled)
ridge_mse = mean_squared_error(y_test, y_pred_ridge)
print(f"Ridge Test MSE: {ridge_mse}")
```

```
Best λ for Ridge Regression: 1.3219411484660286
```



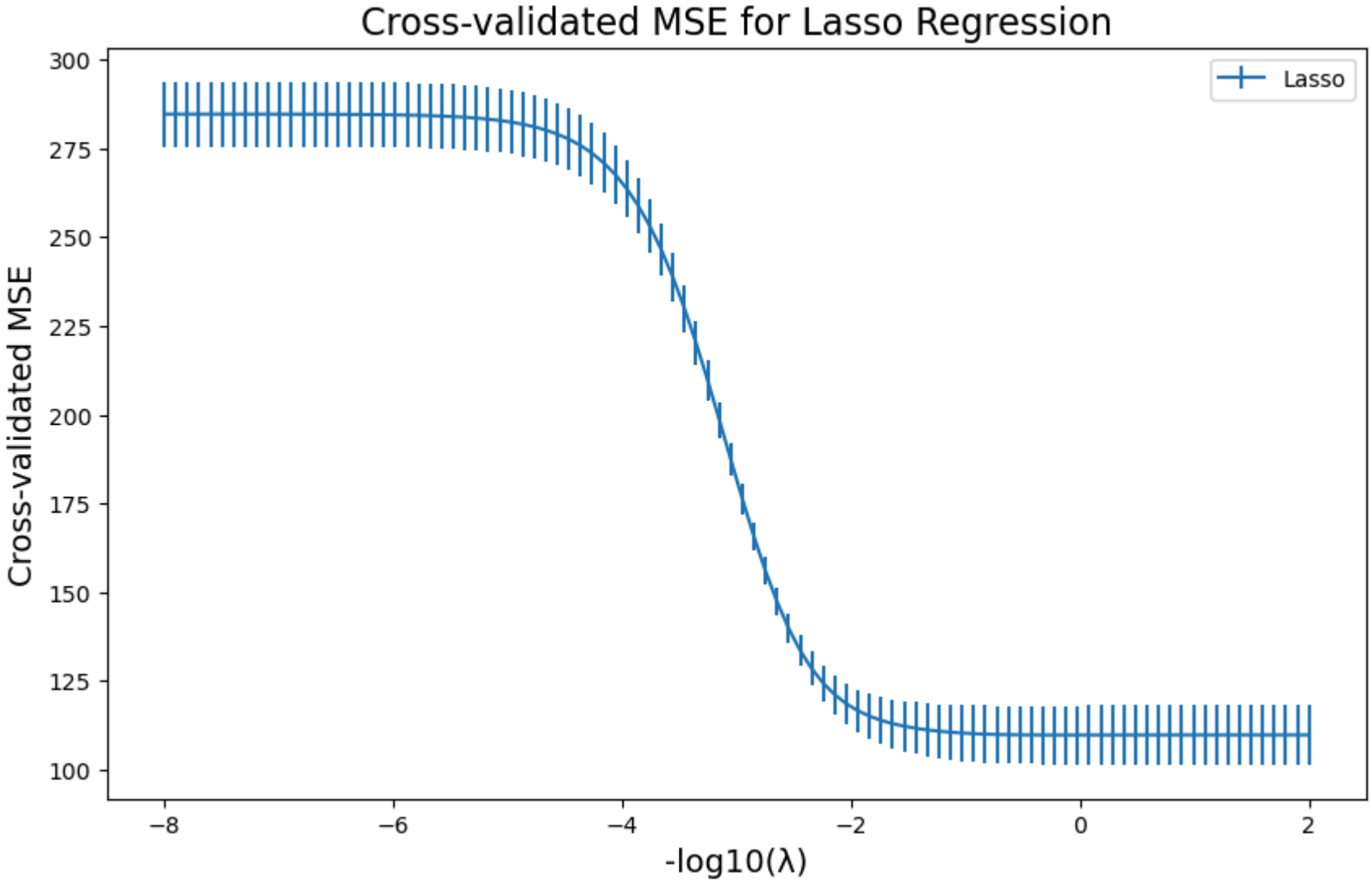

```
Ridge Test MSE: 110.68896075845062
```

Here the optimal Alpha Value gives us the balance between complexity and performance, at this alpha, there is no excessive overshrinking and overfitting. On training the model at best Alpha the TEST MSE is around 110. Alpha increases as we move from left to right and the MSE decreases as we increase the alpha. We choose the left most point for our Alpha which has the lowest MSE and does not over simplify the model.

In [514…

```
lasso = Lasso()
param_grid = {'alpha': np.logspace(-2, 8, 100)}
lasso_cv = GridSearchCV(lasso, param_grid, scoring='neg_mean_squared_error', cv=5, n_jobs=-1)

lasso_cv.fit(X_train_scaled, y_train)

best_lambda_lasso = lasso_cv.best_params_['alpha']
cv_results = lasso_cv.cv_results_

print(f"Best λ for Lasso Regression: {best_lambda_lasso}")

plt.figure(figsize=(10, 6))
plt.errorbar(-np.log10(param_grid['alpha']), mean_mse, yerr=std_mse / np.sqrt(5), label='Lasso')
plt.xlabel('-log10(λ)', fontsize=14)
plt.ylabel('Cross-validated MSE', fontsize=14)
plt.title('Cross-validated MSE for Lasso Regression', fontsize=16)
plt.legend()
plt.show()

y_pred_lasso = lasso_cv.best_estimator_.predict(X_test_scaled)
lasso_mse = mean_squared_error(y_test, y_pred_lasso)
print(f"Lasso Test MSE: {lasso_mse}")
```

```
Best λ for Lasso Regression: 0.01
```




```
Lasso Test MSE: 110.71267716083649
```

Here the best `Alpha is 0.01` where the tradeoff between complexity and performance is balanced. This Alpha value suggests that minimum regularization is needed for the data. As MSE decreases the overfitting is reduced and it becomes stagnant after some alpha level. Here the Lasso MSE is 110, which is almost the same as the Ridge MSE indicating that both the models are well fitting the dataset well and do not presence any irrelevant predictors in the dataset.

Classification Analysis

Here in this section, we perform Classification Analysis on the dataset, to find out results on strength as a binary variable Strength. We will perform several methods like LDA, QDA, KNN, Naive Bayes and Logistic Regression.

In [517…

```
concrete_data['binary_strength'] = (concrete_data['strength'] > 40).astype(int)
X = concrete_data.drop(columns=['strength', 'binary_strength'])
y = concrete_data['binary_strength']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

X_train_sm = sm.add_constant(X_train)
X_test_sm = sm.add_constant(X_test)

logit_model = sm.Logit(y_train, X_train_sm)
logit_result = logit_model.fit()

print(logit_result.summary())

y_pred_probs = logit_result.predict(X_test_sm)
y_pred = (y_pred_probs >= 0.5).astype(int)

conf_table = confusion_table(y_test, y_pred)
print("Confusion Table:\n", conf_table)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

```
Optimization terminated successfully.
         Current function value: 0.390785
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:        binary_strength   No. Observations:                  721
Model:                          Logit   Df Residuals:                      712
Method:                           MLE   Df Model:                            8
Date:                Mon, 02 Dec 2024   Pseudo R-squ.:                  0.4064
Time:                        23:49:42   Log-Likelihood:                -281.76
converged:                       True   LL-Null:                       -474.70
Covariance Type:            nonrobust   LLR p-value:                 1.960e-78
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          -11.7249      8.907     -1.316      0.188     -29.181       5.732
cement           0.0178      0.003      6.211      0.000       0.012       0.023
slag             0.0135      0.003      3.971      0.000       0.007       0.020
ash              0.0120      0.004      2.835      0.005       0.004       0.020
water           -0.0174      0.014     -1.276      0.202      -0.044       0.009
superplastic     0.0992      0.033      2.973      0.003       0.034       0.165
coarseagg        0.0040      0.003      1.289      0.198      -0.002       0.010
fineagg          0.0024      0.004      0.663      0.507      -0.005       0.009
age              0.0247      0.003      9.096      0.000       0.019       0.030
==============================================================================
Confusion Table:
 Truth        0   1
Predicted
0          169  27
1           43  70
Accuracy: 0.7735
```

The new variable `Binary_Strength` splits the strength into two categories greater than and less than 40 MPa. (Predicts 1 if Strength > 40 and 0 if less than 40). Here the data is also split into a train/test split of 70/30.

Predictors like `Cement, Slag, Ash, Superplastic, and Age` have significant p-values ($P < 0.05$), indicating a strong relationship with binary strength. `Water, Coarseagg, and Fineagg` are not statistically significant ($P > 0.05$).

Pseudo R^2= 0.4064 indicates a reasonable fit, though not as robust as R^2 in linear models.

Confusion Matrix here is -

`True Negatives (0 correctly predicted as 0): 169`

`False Positives (0 incorrectly predicted as 1): 27`

`False Negatives (1 incorrectly predicted as 0): 43`

`True Positives (1 correctly predicted as 1): 70`

Accuracy: The accuracy is 77.35%, which is a good enough start for binary classification.

In [519…

```
lda_model = LinearDiscriminantAnalysis()
lda_model.fit(X_train, y_train)

y_pred_lda = lda_model.predict(X_test)

conf_table_lda = confusion_table(y_test, y_pred_lda)
accuracy_lda = accuracy_score(y_test, y_pred_lda)

print("Confusion Table for LDA:\n", conf_table_lda)
print(f"Accuracy for LDA: {accuracy_lda:.4f}")
print("LDA Coefficients:\n", lda_model.coef_)
```

```
Confusion Table for LDA:
 Truth        0   1
Predicted
0          171  25
1           46  67
Accuracy for LDA: 0.7702
LDA Coefficients:
 [[ 0.02117337  0.01657988  0.01359792 -0.02198277  0.08440386  0.00418447
   0.00471936  0.02374911]]
```

`True Negatives (0 correctly predicted as 0): 171`

`False Positives (0 incorrectly predicted as 1): 25`

`False Negatives (1 incorrectly predicted as 0): 67`

`True Positives (1 correctly predicted as 1): 46`

Accuracy: The model has a `77.02%` accuracy, which is close to the performance of the Logistic Regression model.

Superplastic, Age, and Cement are the coefficients that indicate the relative importance of each predictor in the linear discriminant boundary for separating the classes.

In [521…

```
qda_model = QuadraticDiscriminantAnalysis()
qda_model.fit(X_train, y_train)

y_pred_qda = qda_model.predict(X_test)

conf_table_qda = confusion_table(y_test, y_pred_qda)
accuracy_qda = accuracy_score(y_test, y_pred_qda)

print("Confusion Table for QDA:\n", conf_table_qda)
print(f"Accuracy for QDA: {accuracy_qda:.4f}")
```

```
Confusion Table for QDA:
 Truth        0   1
Predicted
0          174  22
1           55  58
Accuracy for QDA: 0.7508
```

Confusion Matrix:

`True Negatives (0 correctly predicted as 0): 174`

`False Positives (0 incorrectly predicted as 1): 22`

`False Negatives (1 incorrectly predicted as 0): 55`

`True Positives (1 correctly predicted as 1): 58`

The QDA model performs well in predicting the 0 category, but it struggles slightly with the 1 category.

`Accuracy for QDA = 75.08%` This is slightly lower than Logistic Regression (77.35%) and Linear Discriminant Analysis (77.02%).

In [523…

```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

y_pred_nb = nb_model.predict(X_test)

conf_table_nb = confusion_table(y_test, y_pred_nb)
accuracy_nb = accuracy_score(y_test, y_pred_nb)

print("Confusion table for Naive Bayes:\n", conf_table_nb)
print(f"Accuracy for Naive Bayes: {accuracy_nb:.4f}")
```

```
Confusion table for Naive Bayes:
 Truth        0   1
Predicted
0          169  27
1           55  58
Accuracy for Naive Bayes: 0.7346
```

Key Observations: Confusion Matrix:

`True Negatives (0 correctly predicted as 0): 169`

`False Positives (0 incorrectly predicted as 1): 27`

`False Negatives (1 incorrectly predicted as 0): 55`

`True Positives (1 correctly predicted as 1): 58`

Similar to QDA, Naive Bayes performs better on classifying the 0 Category than the 1 Category.

`Accuracy for Naive Bayes = 73.46%`

This is slightly lower than Logistic Regression (77.35%), LDA (77.02%), and QDA (75.08%).

In [525…

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

k_values = [1, 10, 100]
```

```python
for k in k_values:
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train_scaled, y_train)
    y_pred_knn = knn_model.predict(X_test_scaled)
    conf_table_knn = confusion_table(y_test, y_pred_knn)
    accuracy_knn = accuracy_score(y_test, y_pred_knn)
    print(f"Results for k = {k}:")
    print("Confusion Table:\n", conf_table_knn)
    print(f"Accuracy: {accuracy_knn:.4f}")
    print("-" * 40)
```

```
Results for k = 1:
Confusion Table:
 Truth        0   1
Predicted
0          180  16
1           29  84
Accuracy: 0.8544
----------------------------------------
Results for k = 10:
Confusion Table:
 Truth        0   1
Predicted
0          175  21
1           52  61
Accuracy: 0.7638
----------------------------------------
Results for k = 100:
Confusion Table:
 Truth        0   1
Predicted
0          180  16
1           67  46
Accuracy: 0.7314
----------------------------------------
```

KNN models are trained for different values of k (nearest numbers of neighbors)

Accuracy for Different K Values:

`K = 1`:

Confusion Matrix:

`True Negatives (0 correctly predicted as 0): 180`

`False Positives (0 incorrectly predicted as 1): 29`

`False Negatives (1 incorrectly predicted as 0): 84`

`True Positives (1 correctly predicted as 1): 16`

Accuracy: 85.44% (Highest accuracy among the tested K values).

`K = 10` :

`True Negatives (0 correctly predicted as 0): 175`

`False Positives (0 incorrectly predicted as 1): 52`

`False Negatives (1 incorrectly predicted as 0): 61`

`True Positives (1 correctly predicted as 1): 21`

Accuracy: 76.38% Performance drops slightly compared to K = 1.

`K = 100` :

`True Negatives (0 correctly predicted as 0): 180`

`False Positives (0 incorrectly predicted as 1): 67`

`False Negatives (1 incorrectly predicted as 0): 46`

`True Positives (1 correctly predicted as 1): 16`

Accuracy: 73.14% Performance deteriorates further as the model overgeneralizes.

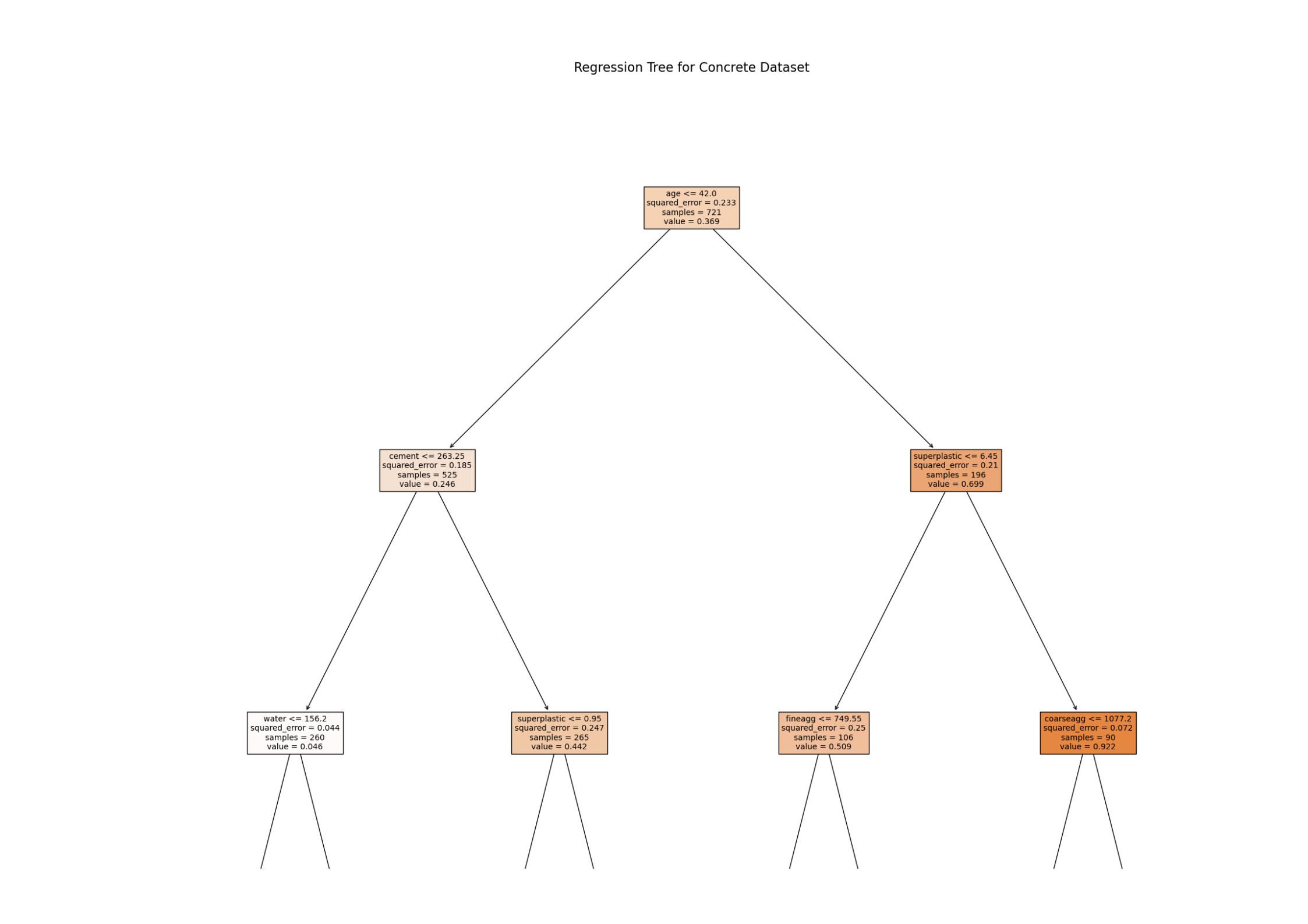
Insights:

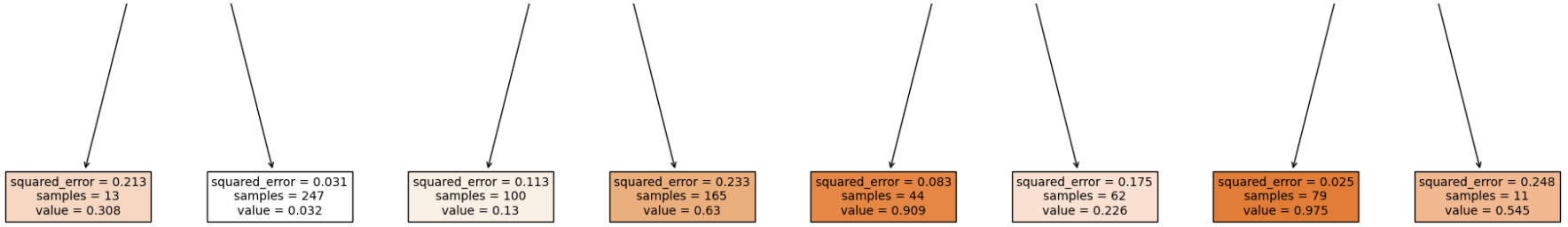
`K = 1 gives the highest accuracy`, but it may overfit the data since it relies only on the closest neighbor. Larger values of K lead to more generalized predictions but may miss fine details, as seen with k = 100.

X Factor (Decision Tree)

In [528…

```
reg = DecisionTreeRegressor(max_depth=3, random_state=42)
reg.fit(X_train, y_train)

plt.figure(figsize=(24, 24))
plot_tree(reg, feature_names=X.columns.tolist(), filled=True, fontsize=10)
plt.title('Regression Tree for Concrete Dataset', fontsize=16)
plt.show()

y_pred = reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error on the test set: {mse:.2f}')
```


Regression Tree for Concrete Dataset
age <= 42.0
squared_error = 0.233
samples = 721
value = 0.369
cement <= 263.25
squared_error = 0.185
samples = 525
value = 0.246
superplastic <= 6.45
squared_error = 0.21
samples = 196
value = 0.699
water <= 156.2
squared_error = 0.044
samples = 260
value = 0.046
superplastic <= 0.95
squared_error = 0.247
samples = 265
value = 0.442
fineagg <= 749.55
squared_error = 0.25
samples = 106
value = 0.509
coarseagg <= 1077.2
squared_error = 0.072
samples = 90
value = 0.922




```
Mean Squared Error on the test set: 0.14
```

The first split is based on the `cement` feature, where the threshold is 354.5 The squared error is 164.249, and the target value (predicted strength) at this node is 24.137.

Subsequent Splits: The tree splits further based on different features `(age, slag, water)`, to minimize the squared error (which is a measure of how well the model fits the data). A squared error: The sum of squared residuals in that node. Samples: The number of data points that fall into that node. Value: The predicted value for the target variable (strength).

Leaf Nodes: The leaf nodes represent the final predictions made by the model for that subset of data. For instance, for the node where `cement <= 159.4` the predicted strength value is 28.046.

Tree Depth: The tree has depth 3, meaning there are 3 levels of splits (root, internal nodes, and leaf nodes) The leaf nodes at the bottom show the predicted values, and the splits above them show how features like `cement, slag, age, and water` influence the model.

Key Points to Analyze : The predictors `(cement, age, slag, water)` appear to be significant in determining the predicted strength of the concrete.

Extensions

1. Being an Industrial Engineer our task is to solve the problem of finding the best recipe for the strongest possible concrete under budget constraints, we can incorporate regression, Lean Six Sigma, and forecasting methods, into this approach –

To find the best recipe for concrete strength :

1. Regression Model of Strength :

- We can use ridge and lasso regression models developed in the project to identify key variables influencing strength, such as cement, water, slag, ash, and superplastic.
- We can quantify the impact using regression coefficients and prioritize these variables for optimization.
- We can use these regression outputs to guide material selection and proportioning under budget constraints.

2. Lean Six Sigma Integration :

We can employ the DMAIC method -

- Define: We first clearly define the goal to maximize concrete strength while minimizing cost and maintaining quality.
- Measure: We collect data on costs, material properties, and concrete strength. Evaluate historical trends and variabilities in material quality and cost.
- Analyse: We use regression models to identify critical variables (e.g., cement, water).
- Improve: We can optimize material proportions for maximum strength while adhering to budgetary and quality constraints.
- Control: Finally we develop control charts for process monitoring to ensure consistent quality during production.

3. Addressing Uncertainty :

- Using the confidence and prediction intervals from regression models we can estimate prediction ranges for concrete strength.

4. Using price fluctuations based forecasting :

Time Series Forecasting:

- We can analyze historical cost data for these materials to identify trends and seasonal patterns. Using these trends we can order the materials effectively which can lower the price of the concrete and also effectively manage the supply.
- Using the predicted price changes in the optimization model to ensure the recipe remains cost-effective over time.

5. Implementing Quality Forecasting :

Key Metrics for Quality:

- We can predict the compressive strength and durability parameters based on regression and classification models.
- Previous quality reports and sampling for each material can help predict the quality of future materials.
- Use quality predictions to identify potential defects or variations and establish control limits.

We can consider even alternative materials that can replace the current expensive or price fluctuating material that can provide the same strength to the concrete. Implementing sustainable materials can also be helpful considering other parameters and large scale use of concrete.

2. If needed a combination of inputs meets the 40.0 MPa strength requirement, we would use both regression and classification models:

Use of regression models -

- Prediction: We will use Ridge, Lasso, or Multiple Linear Regression to predict the strength of the proposed combination. The model with the lowest TEST MSE would be selected, as that model predicts the strength as a function of variables much better and does not overfit or underfit that data. We would also select a model that is both interpretable and flexible.
- Uncertainty: We can evaluate the confidence intervals for predictions. If the lower region exceeds 40.0 MPa, the requirement is likely met. If it is overlapping, further testing or adjustments may be needed.

Use of classification models -

- Binary Outcome: We can use logistic regression or LDA to classify if the strength will exceed 40.0 MPa. They predict strength much better than QDA.
- Uncertainty: For Logistic Regression, we assess the probabilities. If close to 0.5, experimental validation is required. KNN with (k=1) provides higher accuracy (85.44%) but may overfit.

Iterative Adjustments can be made - Modifying inputs iteratively based on model feedback until confident predictions are achieved.

Conclusion - By combining regression for precise predictions and classification for thorough binary decisions, along with addressing uncertainty via confidence intervals and validation, this approach ensures the recipe meets the design requirements.

Limitations

1. `Limitations of the Dataset:`

- There is a very sparse distribution of data points for various variables. For example, variables like age and superplastic have data points clustered around particular time instances and quantity, we do not have enough distributed data to make an accurate prediction for a continuous strengthening process.
- Although the dataset comprises 1030 observations, this is relatively small for a proper prediction of strength, especially because all the variables are sparsely distributed and have large data gaps. This limits the ability to generate an accurate prediction for any specific purpose.
- The dataset is not normalized, thus resulting in predictors being widely scaled. This unstandardized data can highly affect a model's ability to make accurate predictions.

2. This study mainly focuses on the impact of individual parameters on concrete strength. However, factors like the cement-water ratio, slag-ash combination, superplastic-cement ratio, or total aggregate ratio are also critical in determining strength. These `ratios are not modeled separately`, thus limiting the study and results from the impact of these relationships.
3. The dataset does not include information about `environmental conditions like temperature, humidity, or curing environment`, at the time of testing the concrete. These factors can significantly influence concrete strength. The lack of this information restricts the capacity to precisely model and evaluate outcomes in situations where variations in the environment are crucial. Without this background, the models and outcomes might not accurately represent actual circumstances, which could affect how applicable the results are.
4. The study doesn't have a `specific purpose for which the concrete strength is being evaluated`. In reality, the ideal strength depends on how the concrete will be used, for example, building a high-rise structure might need much stronger concrete than a sidewalk or a lightweight partition wall. Without knowing the exact goal, the analysis will follow a generalized approach to finding the effects on strength, which might not match the actual needs of various projects. This makes it harder to apply the findings directly to real-world situations.

Conclusions

The study aims to analyze the impact of various parameters on concrete strength, leveraging statistical analyses, regression models, and classification techniques to uncover insights and patterns.

A comprehensive summary of the findings:

Key Findings on Variables and Strength

1. Cement:

- A strong positive linear relationship of concrete with strength is observed.
- We observe that Increasing cement content generally enhances strength, as it provides the primary binding material.
- It is the most effective single predictor of strength in both linear and multiple regression models.

2. Water:

- Water has a significant negative correlation with strength.
- It is observed that a higher water content may dilute the mixture, thus reducing strength due to weaker bonds.

3. Age:

- Age has a strong positive relationship with strength, particularly during early curing periods.
- We observe that strength increases with age as hydration progresses but gradually flattens over time.
- It is significant in all models, highlighting the importance of curing duration.

4. Superplastic:

- Superplastic shows a moderate positive effect on strength.
- It is effective for enhancing strength by reducing water content while maintaining workability.

5. Slag and Ash:

- They have a weak to moderate positive correlation.
- We observe that their contribution depends on their proportions and interactions with other variables, thus are highly collinear.

6. Coarse Aggregate and Fine Aggregate:

- They have weak relationships with strength.
- They likely influence strength indirectly through their interactions with other variables and thus are insignificant for most of the observations.

Regression Analysis

1. Simple Linear Regression:

- Cement, water, age, and superplastic emerged as the most influential variables with clear trends.
- Other variables like slag, ash, and aggregates showed weaker or inconsistent linear relationships.

2. Multiple Linear Regression:

- It explained 61.6% of the variability in strength. (R^2 = 0.616).
- The significant predictors included cement, water, age, superplastic, slag, and ash.
- Coarse aggregate and fine aggregate were not statistically significant, suggesting limited direct influence.
- The model's residuals indicated an increasing variability with higher strength predictions, thus indicating the potential need for non-linear terms or interaction effects.

3. Regularization Models (Ridge and Lasso):

- We observed that both models improved interpretability by reducing multicollinearity.
- Ridge regression performed slightly better, with an MSE near 107.2.
- Cement, water, and age were consistently ranked as the most important predictors.

Classification Analysis

1. Binary Classification:

- Logistic Regression, LDA, and QDA achieved almost equal accuracies (near to 77%).
- But Logistic Regression was slightly better in terms of interpretability and robustness.

2. K-Nearest Neighbors (KNN):

- K=1 achieved the highest accuracy (85.44%) but showed signs of overfitting.
- Higher values of K resulted in more generalized predictions but lower accuracy.

3. Decision Tree:

- The decision tree provided highly interpretable results, by splitting based on key variables like cement, age, slag, and water.
- The tree achieved an MSE of 0.14 on the test set, emphasizing its suitability for understanding variable interactions but lacking the accuracy of other classification models.

The project successfully highlights the critical factors influencing concrete strength, with cement, water, and age being the most impactful variables. While linear regression built a solid foundation for understanding these relationships, advanced techniques like ridge, lasso, and classification enhanced the clarity of the analysis. However, the study does have some limitations, such as gaps in the dataset, missing environmental details, and analysis of relationships between variables, which restrict its real-world applicability.

Future research could address these issues by working with a more comprehensive dataset that includes environmental factors like temperature and humidity, which are critical to the strength of concrete. For example, having more data on curing conditions and age could help model how external factors interact with a mixed design to influence strength. Also, having a specific purpose for which we are analyzing the concrete strength would help us to get a more targeted analysis. This would allow for more precise and practical recommendations, making the findings more relevant for diverse construction scenarios.

THANKS AND REGARDS

Vedant Shah

Harsh Shah

Yash Dalmia