Concrete Strength Data EDA

Concrete, the ubiquitous building block, is the foundation of countless structures. Its strength is paramount, ensuring safety and longevity. But what factors influence this strength?

To answer this question we'll use exploratory data analysis (EDA) of concrete mix data Through techniques like statistical summaries and visualizations, we'll unveil the distribution of each element.

Are there any outliers that stand out from the mix? We'll then delve deeper, examining how each ingredient interacts with the overall strength. Does a higher cement content necessarily translate to a stronger concrete? This initial exploration serves as a roadmap for further analysis. By understanding the data landscape, we can identify the most influential factors and potential roadblocks like missing values or inconsistencies. This knowledge will be invaluable when building models to predict concrete strength, ultimately leading to the creation of even more robust and reliable structures.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
import statsmodels.api as sm
from scipy import stats

data = pd.read_csv('/content/drive/MyDrive/ConcreteStrengthData.csv')
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1030,\n  \"fields\": [\n    {\n      \"column\": \"CementComponent \",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 104.50636449481532,\n        \"min\": 102.0,\n        \"max\": 540.0,\n        \"num_unique_values\": 278,\n        \"samples\": [\n          337.9,\n          290.2,\n          262.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"BlastFurnaceSlag\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 86.27934174810584,\n        \"min\": 0.0,\n        \"max\": 359.4,\n        \"num_unique_values\": 185,\n        \"samples\": [\n          94.7,\n          119.0,\n          136.3\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"FlyAshComponent\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 63.99700415268765,\n        \"min\": 0.0,\n        \"max\": 200.1,\n        \"num_unique_values\": 156,\n        \"samples\": [\n          98.0,\n          142.0,\n          195.0\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"WaterComponent\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 21.35421856503247,\n   \"min\":
121.8,\n   \"max\": 247.0,\n   \"num_unique_values\": 195,\n
\"samples\": [\n   195.4,\n   183.8,\n   127.3\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"SuperplasticizerComponent\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
5.97384139248552,\n   \"min\": 0.0,\n   \"max\": 32.2,\n
\"num_unique_values\": 111,\n   \"samples\": [\n   15.0,\n
28.2,\n   16.5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"CoarseAggregateComponent\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 77.75395396672077,\n
\"min\": 801.0,\n   \"max\": 1145.0,\n
\"num_unique_values\": 284,\n   \"samples\": [\n   852.1,\
n   913.9,\n   914.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"FineAggregateComponent\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
80.17598014240437,\n   \"min\": 594.0,\n   \"max\": 992.6,\n
\"num_unique_values\": 302,\n   \"samples\": [\n   710.0,\
n   695.4,\n   769.3\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"AgeInDays\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
63,\n   \"min\": 1,\n   \"max\": 365,\n
\"num_unique_values\": 14,\n   \"samples\": [\n   91,\n
100,\n   28\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Strength\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 16.705741961912512,\n   \"min\":
2.33,\n   \"max\": 82.6,\n   \"num_unique_values\": 845,\n
\"samples\": [\n   41.68,\n   39.59,\n   2.33\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"data"}

```
data.tail()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 5,\n \"fields\": [\
n   {\n   \"column\": \"CementComponent \",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
76.18921839735593,\n   \"min\": 148.5,\n   \"max\": 322.2,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   322.2,\n
260.9,\n   148.5\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"BlastFurnaceSlag\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 68.85228391273597,\n
\"min\": 0.0,\n   \"max\": 186.7,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   0.0,\n   100.5,\n

```
139.4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"FlyAshComponent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 46.32475580075948,\n        \"min\":
0.0,\n        \"max\": 115.6,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          115.6,\n          78.3,\n          108.6\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"WaterComponent\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
10.783320453366857,\n        \"min\": 175.6,\n        \"max\": 200.6,\
n        \"num_unique_values\": 5,\n        \"samples\": [\n
196.0,\n          200.6,\n          192.7\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SuperplasticizerComponent\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.9882152800941857,\n        \"min\": 6.1,\n        \"max\": 11.3,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          10.4,\n
8.6,\n          6.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"CoarseAggregateComponent\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 63.48728219100265,\n
\"min\": 817.9,\n        \"max\": 989.6,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          817.9,\n
864.5,\n          892.4\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"FineAggregateComponent\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 20.277006682446988,\n
\"min\": 761.5,\n        \"max\": 813.4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          813.4,\n
761.5,\n          780.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"AgeInDays\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 28,\n
\"max\": 28,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          28\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Strength\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7.371633468913115,\n        \"min\":
23.7,\n        \"max\": 44.28,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          31.18\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


check the size of data

```
data.shape

(1030, 9)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   CementComponent             1030 non-null   float64
 1   BlastFurnaceSlag            1030 non-null   float64
 2   FlyAshComponent             1030 non-null   float64
 3   WaterComponent              1030 non-null   float64
 4   SuperplasticizerComponent   1030 non-null   float64
 5   CoarseAggregateComponent    1030 non-null   float64
 6   FineAggregateComponent      1030 non-null   float64
 7   AgeInDays                   1030 non-null   int64  
 8   Strength                    1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

Details about the data :

```
print(f"There are {data.shape[0]} instances.")
print(f"There are {data.shape[1]} dataframe columns/attributes.")
```

```
There are 1030 instances.
There are 9 dataframe columns/attributes.
```

check if we have null value

```
data.isna().sum()
```

```
CementComponent              0
BlastFurnaceSlag             0
FlyAshComponent              0
WaterComponent               0
SuperplasticizerComponent    0
CoarseAggregateComponent     0
FineAggregateComponent       0
AgeInDays                    0
Strength                     0
dtype: int64
```

find the duplicated

```
data.duplicated().sum()
```

```
25
```

drop duplicates

```
data.drop_duplicates()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 1005,\n \"fields\": [\n   {\n   \"column\": \"CementComponent \",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 104.34426071285245,\n   \"min\": 102.0,\n   \"max\": 540.0,\n   \"num_unique_values\": 278,\n   \"samples\": [\n   337.9,\n   290.2,\n   262.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"BlastFurnaceSlag\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 86.17080699343222,\n   \"min\": 0.0,\n   \"max\": 359.4,\n   \"num_unique_values\": 185,\n   \"samples\": [\n   94.7,\n   119.0,\n   136.3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"FlyAshComponent\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 64.20796859777064,\n   \"min\": 0.0,\n   \"max\": 200.1,\n   \"num_unique_values\": 156,\n   \"samples\": [\n   98.0,\n   142.0,\n   195.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"WaterComponent\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 21.339334087611302,\n   \"min\": 121.8,\n   \"max\": 247.0,\n   \"num_unique_values\": 195,\n   \"samples\": [\n   195.4,\n   183.8,\n   127.3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SuperplasticizerComponent\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 5.919966720023434,\n   \"min\": 0.0,\n   \"max\": 32.2,\n   \"num_unique_values\": 111,\n   \"samples\": [\n   15.0,\n   28.2,\n   16.5\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"CoarseAggregateComponent\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 77.57966673714678,\n   \"min\": 801.0,\n   \"max\": 1145.0,\n   \"num_unique_values\": 284,\n   \"samples\": [\n   852.1,\n   913.9,\n   914.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"FineAggregateComponent\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 80.34043464964552,\n   \"min\": 594.0,\n   \"max\": 992.6,\n   \"num_unique_values\": 302,\n   \"samples\": [\n   710.0,\n   695.4,\n   769.3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"AgeInDays\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 63,\n   \"min\": 1,\n   \"max\": 365,\n   \"num_unique_values\": 14,\n   \"samples\": [\n   91,\n   100,\n   28\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"Strength\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 16.284815369229054,\n        \"min\":
2.33,\n        \"max\": 82.6,\n        \"num_unique_values\": 845,\n
\"samples\": [\n          41.68,\n          39.59,\n          2.33\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


Checking the column names

```
list(data.columns)
```

```
['CementComponent ',
 'BlastFurnaceSlag',
 'FlyAshComponent',
 'WaterComponent',
 'SuperplasticizerComponent',
 'CoarseAggregateComponent',
 'FineAggregateComponent',
 'AgeInDays',
 'Strength']
```

rename column names

The names are long and difficult to read, so we must change them to shorter, easy-to-read names

```
data.rename(columns={'CementComponent ': 'Cement', 'BlastFurnaceSlag':
"Slag",
                    'FlyAshComponent': 'Fly Ash', 'WaterComponent':
'Water',
                    'SuperplasticizerComponent': 'Super plasticizer',
'CoarseAggregateComponent': 'Coarse Aggregate',
                     'FineAggregateComponent': 'Fine Aggregate',
'AgeInDays': 'Age'}, inplace=True)
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1030,\n  \"fields\":
[\n    {\n      \"column\": \"Cement\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 104.50636449481532,\n
\"min\": 102.0,\n        \"max\": 540.0,\n
\"num_unique_values\": 278,\n        \"samples\": [\n          337.9,\
n          290.2,\n          262.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Slag\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 86.27934174810584,\n
\"min\": 0.0,\n        \"max\": 359.4,\n        \"num_unique_values\":
185,\n        \"samples\": [\n          94.7,\n          119.0,\n
136.3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Fly
```

Ash\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 63.99700415268765,\n   \"min\": 0.0,\n   \"max\":
200.1,\n   \"num_unique_values\": 156,\n   \"samples\": [\n
98.0,\n   142.0,\n   195.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Water\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 21.35421856503247,\n
\"min\": 121.8,\n   \"max\": 247.0,\n
\"num_unique_values\": 195,\n   \"samples\": [\n   195.4,\
n   183.8,\n   127.3\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Super plasticizer\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
5.97384139248552,\n   \"min\": 0.0,\n   \"max\": 32.2,\n
\"num_unique_values\": 111,\n   \"samples\": [\n   15.0,\n
28.2,\n   16.5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Coarse Aggregate\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 77.75395396672077,\n   \"min\":
801.0,\n   \"max\": 1145.0,\n   \"num_unique_values\": 284,\
n   \"samples\": [\n   852.1,\n   913.9,\n
914.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Fine Aggregate\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 80.17598014240437,\n   \"min\":
594.0,\n   \"max\": 992.6,\n   \"num_unique_values\": 302,\n
\"samples\": [\n   710.0,\n   695.4,\n   769.3\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 63,\n
\"min\": 1,\n   \"max\": 365,\n   \"num_unique_values\":
14,\n   \"samples\": [\n   91,\n   100,\n
28\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Strength\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 16.705741961912512,\n   \"min\":
2.33,\n   \"max\": 82.6,\n   \"num_unique_values\": 845,\n
\"samples\": [\n   41.68,\n   39.59,\n   2.33\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"data"}

convert categorical variables to categories

```
data['Age'] = data['Age'].astype('category')
data.describe(include='category')
```


{"summary":"{\n \"name\": \"data\",\n \"rows\": 4,\n \"fields\": [\
n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 476,\n   \"min\": 14,\n

\"max\": 1030,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   14,\n   425,\n   1030\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
data['Super plasticizer'] = data['Super plasticizer'].astype('category')
data.describe(include='category')
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 4,\n \"fields\": [\n   {\n   \"column\": \"Super plasticizer\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 461.6138357256348,\n   \"min\": 0.0,\n   \"max\": 1030.0,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   111.0,\n   379.0,\n   1030.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 476,\n   \"min\": 14,\n   \"max\": 1030,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   14,\n   425,\n   1030\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Top 10 Age

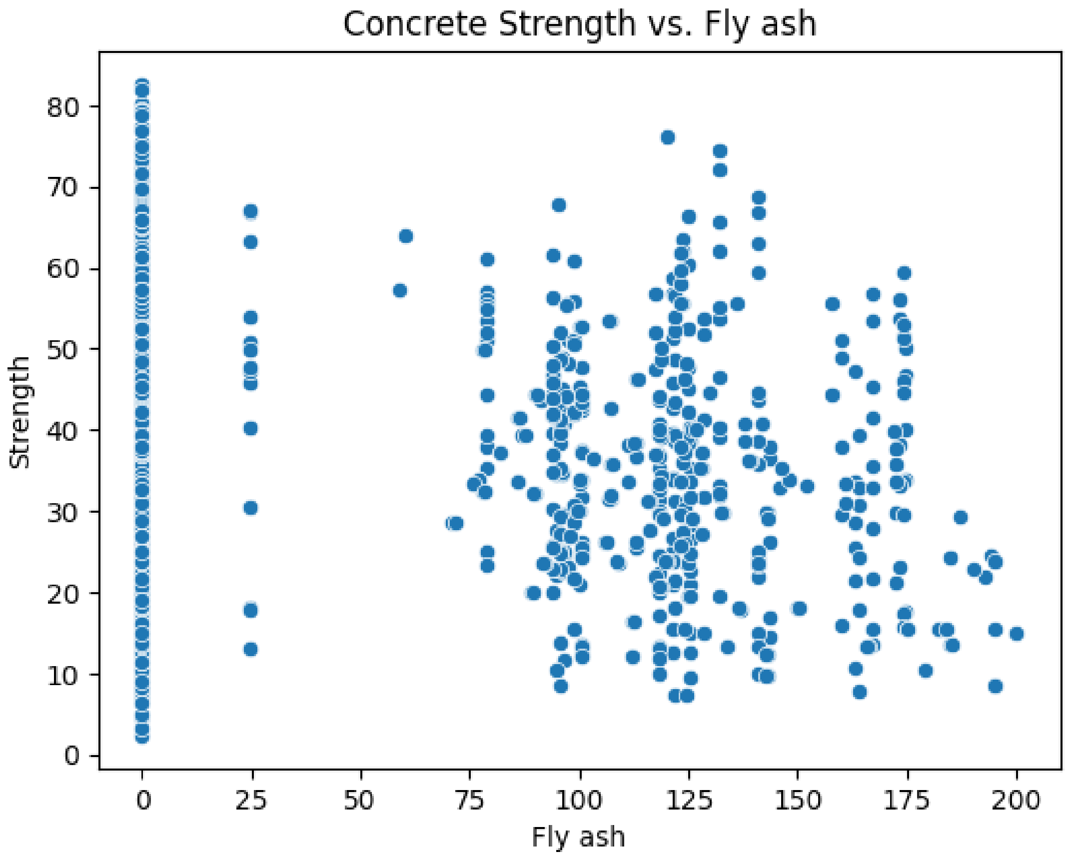
```
top10Age = data['Age'].value_counts().nlargest(10).to_frame()
top10Age
```

{"summary":"{\n \"name\": \"top10Age\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 10,\n   \"samples\": [\n   91,\n   3,\n   90\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 121,\n   \"min\": 14,\n   \"max\": 425,\n   \"num_unique_values\": 10,\n   \"samples\": [\n   22,\n   134,\n   54\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"top10Age"}

Scatterplots

A scatter plot is a visual representation of how two variables relate to each other. You can use scatter plots to explore the relationship between two variables, for example by looking for any correlation between them.

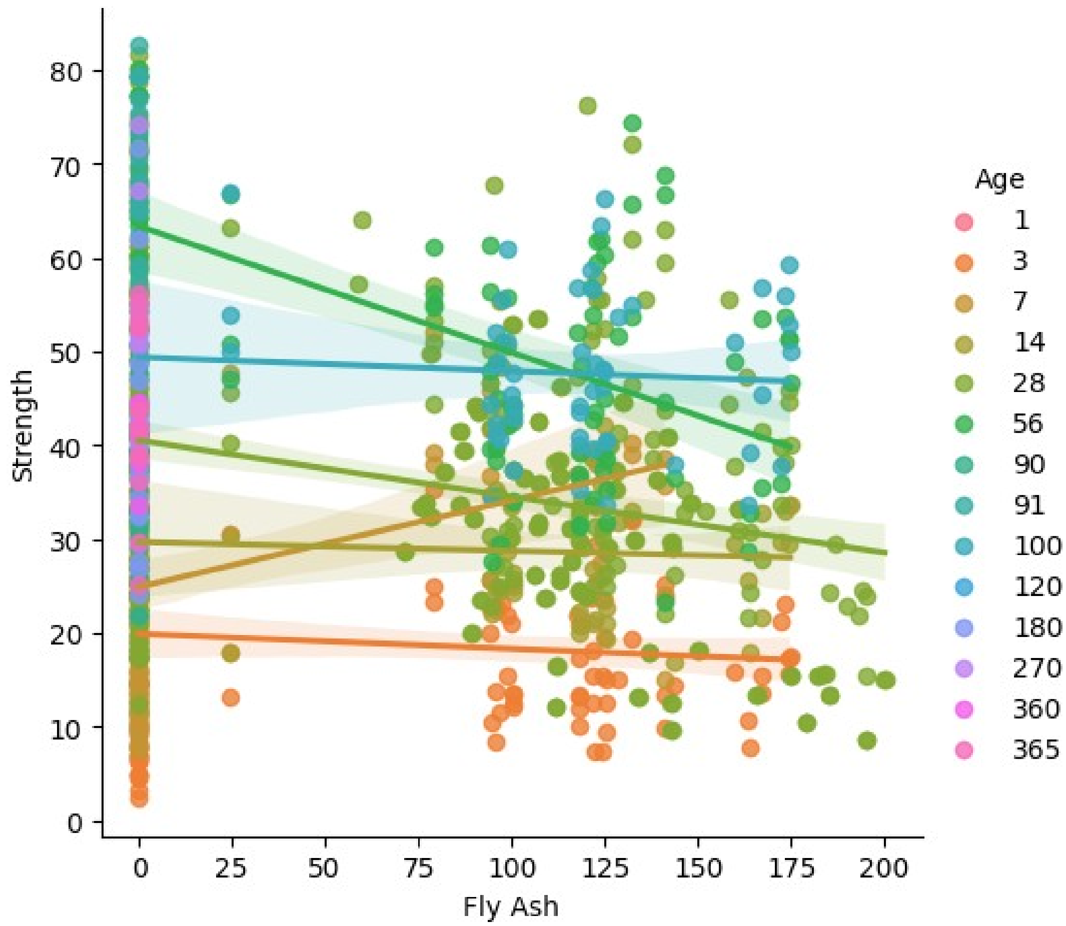
```
ax = sns.scatterplot(x="Fly Ash", y="Strength", data=data)
ax.set_title("Concrete Strength vs. Fly ash")
ax.set_xlabel("Fly ash");
```



Adding color as a third dimension

we conclude that the strength of cement is the best in Age 56-120

```
sns.lmplot(x="Fly Ash", y="Strength", hue="Age", data=data);
```



Corrleation matrix

A correlation matrix is a handy way to calculate the pairwise correlation coefficients between two or more (numeric) variables. The Pandas data frame has this functionality built-in to its corr() method, which I have wrapped inside the round() method to keep things tidy.

```
cormat = data.corr()
round(cormat,2)
```

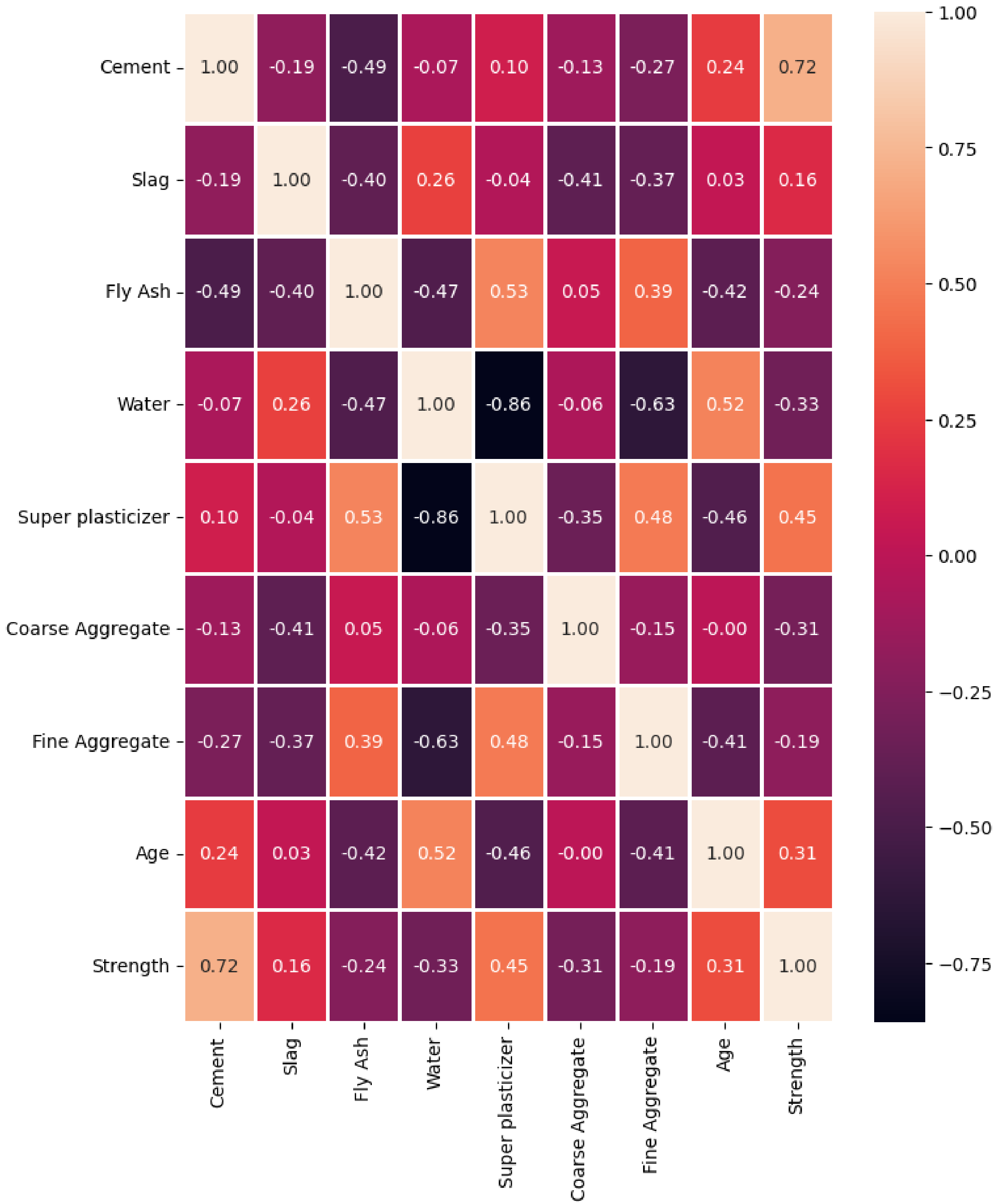
{"summary":"{\n \"name\": \"round(cormat,2)\",\n \"rows\": 9,\n \"fields\": [\n   {\n   \"column\": \"Cement\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.437667428280627,\n   \"min\": -0.4,\n   \"max\": 1.0,\n   \"num_unique_values\": 9,\n   \"samples\": [\n   0.08,\n   -0.28,\n   -0.11\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Slag\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.41368600545717177,\n   \"min\": -0.32,\n   \"max\": 1.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   -0.28,\n   1.0,\n   -0.04\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fly Ash\",\n

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.4345399866525519,\n        \"min\": -0.4,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          -0.15,\n          -0.32,\n          -0.01\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Water\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.4857840169366538,\n        \"min\": -0.66,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          0.28,\n          0.11,\n          -0.18\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Super plasticizer\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.4713396982125643,\n        \"min\": -0.66,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          -0.19,\n          0.04,\n          -0.27\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Coarse Aggregate\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.39526713892140225,\n        \"min\": -0.28,\n        \"max\": 1.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          -0.28,\n          1.0,\n          -0.11\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Fine Aggregate\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.42839169511609865,\n        \"min\": -0.45,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          -0.16,\n          -0.28,\n          -0.18\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.37625715201766524,\n        \"min\": -0.19,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          1.0,\n          -0.04,\n          -0.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Strength\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.41517399297697394,\n        \"min\": -0.29,\n        \"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          0.33,\n          0.13,\n          -0.16\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Correlation matrix to heat map

Python, and its libraries, make lots of things easy. For example, once the correlation matrix is defined , it can be passed to Seaborn's heatmap() method to create a heatmap (or headgrid). The basic idea of heatmaps is that they replace numbers with colors of varying shades, as indicated by the scale on the right. Cells that are lighter have higher values of r. This type of visualization can make it much easier to spot linear relationships between variables than a table of numbers. For example, if I focus on the "Strength" column, I immediately see that "Cement" and "FlyAsh" have the largest positive correlations whereas "Slag" has the large negative correlation.

```
plt.figure(figsize = (8, 10))
sns.heatmap(cormat.corr(), annot = True, fmt = '0.2f', annot_kws =
{'size' : 10}, linewidth = 2, linecolor = 'white')
plt.show()
```



Preparing the data

Define Y and X matrices

and add a constant column to the X matrix

```
import statsmodels.api as sm
Y = data['Strength']
X = data['Fly Ash']
X.head()
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: Fly Ash, dtype: float64
```

```
X = sm.add_constant(X)
X.head()
```

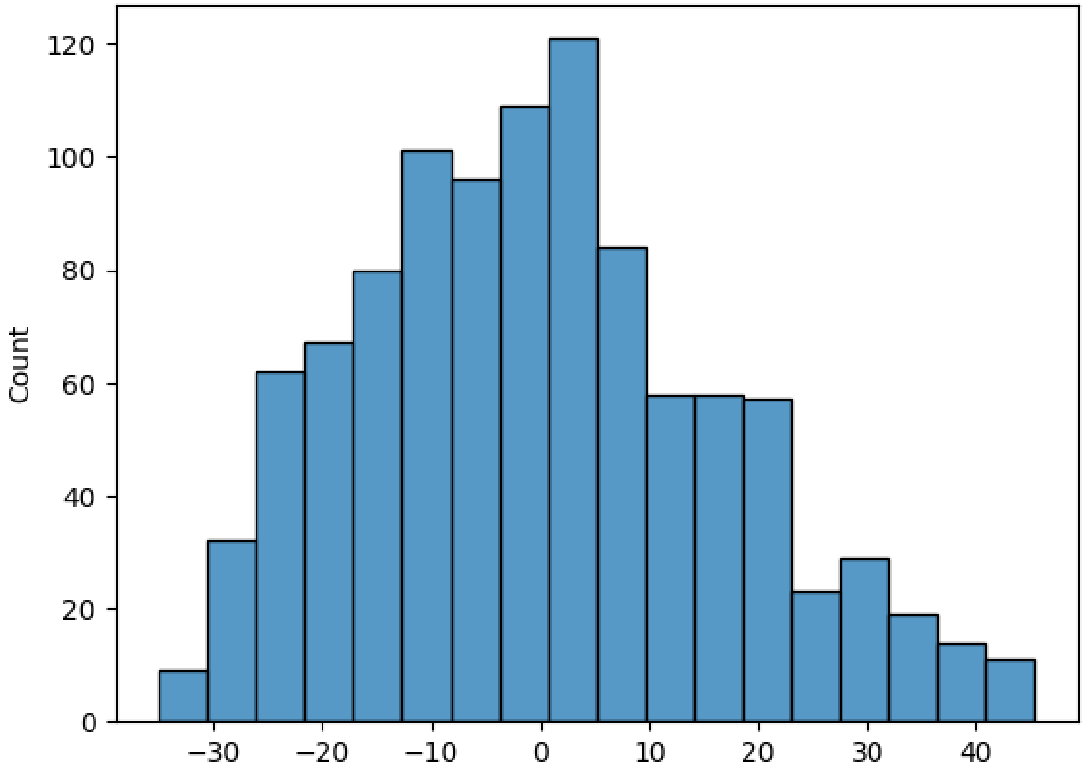

```
{"summary":"{\n  \"name\": \"X\",\n  \"rows\": 1030,\n  \"fields\": [\
n    {\n      \"column\": \"const\",\n      \"properties\": {\n        
\"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\": 1.0,\n        
\"max\": 1.0,\n        \"num_unique_values\": 1,\n        \"samples\": 
[\n          1.0\n        ],\n        \"semantic_type\": \"\",\n        
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Fly 
Ash\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        
\"std\": 63.99700415268765,\n        \"min\": 0.0,\n        \"max\": 
200.1,\n        \"num_unique_values\": 156,\n        \"samples\": [\n          
98.0\n        ],\n        \"semantic_type\": \"\",\n        
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"X"}
```


Running the model

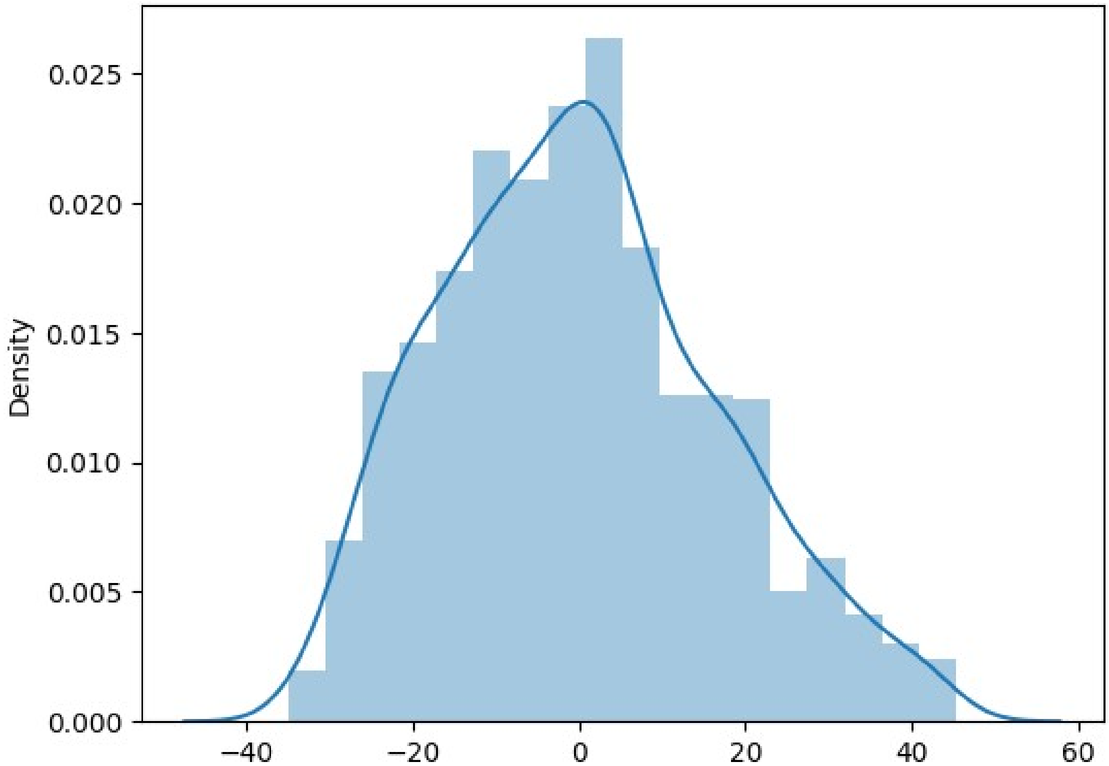
```
model = sm.OLS(Y, X, missing='drop')
model_result = model.fit()
model_result.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results                            
======================================================================
========
Dep. Variable:               Strength   R-squared:                       
0.011
Model:                            OLS   Adj. R-squared:                  
0.010
Method:                 Least Squares   F-statistic:                     
```

```
11.63
Date:                  Sat, 06 Apr 2024   Prob (F-statistic):
0.000675
Time:                          09:09:59   Log-Likelihood:
-4355.4
No. Observations:                  1030   AIC:
8715.
Df Residuals:                      1028   BIC:
8725.
Df Model:                             1

Covariance Type:              nonrobust

======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const         37.3139      0.679     54.978      0.000      35.982
38.646
Fly Ash       -0.0276      0.008     -3.410      0.001      -0.043
-0.012
======================================================================
========
Omnibus:                         29.013   Durbin-Watson:
0.848
Prob(Omnibus):                    0.000   Jarque-Bera (JB):
27.218
Skew:                             0.351   Prob(JB):
1.23e-06
Kurtosis:                         2.625   Cond. No.
110.
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
"""

sns.histplot(model_result.resid);
```



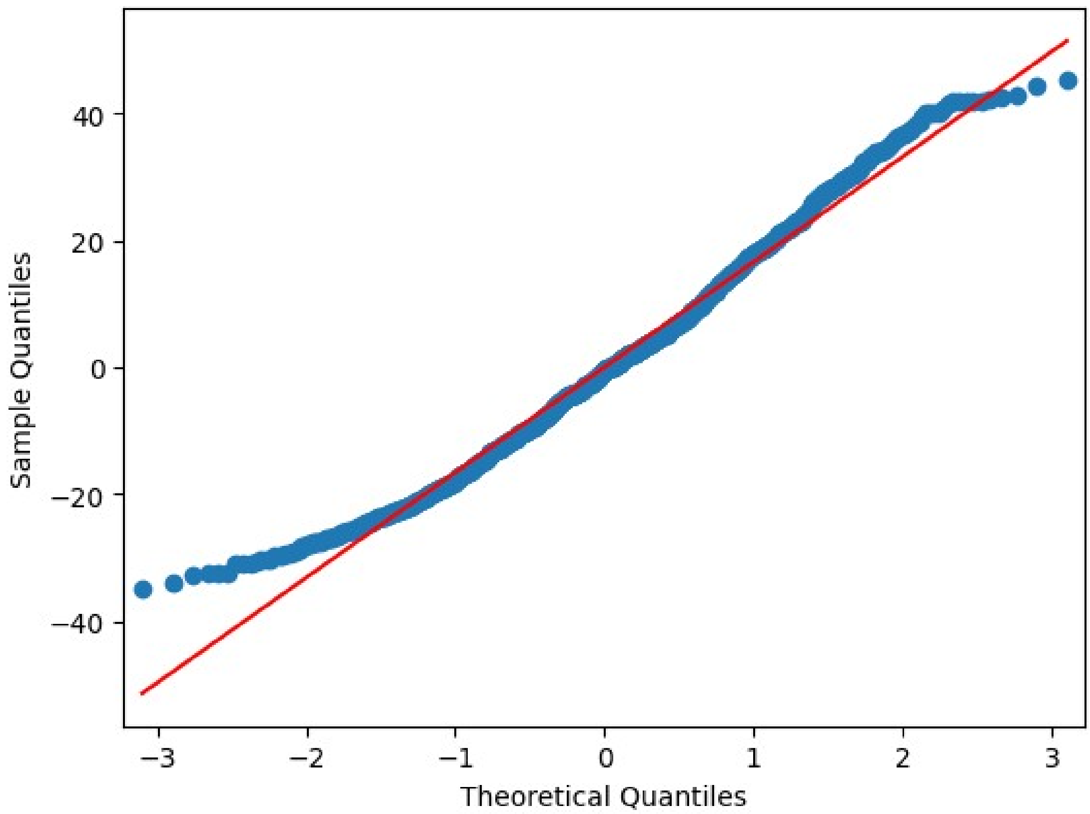

```
sns.distplot(model_result.resid)

<ipython-input-68-73949055c77d>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(model_result.resid)

<Axes: ylabel='Density'>
```



Q-Q Plot

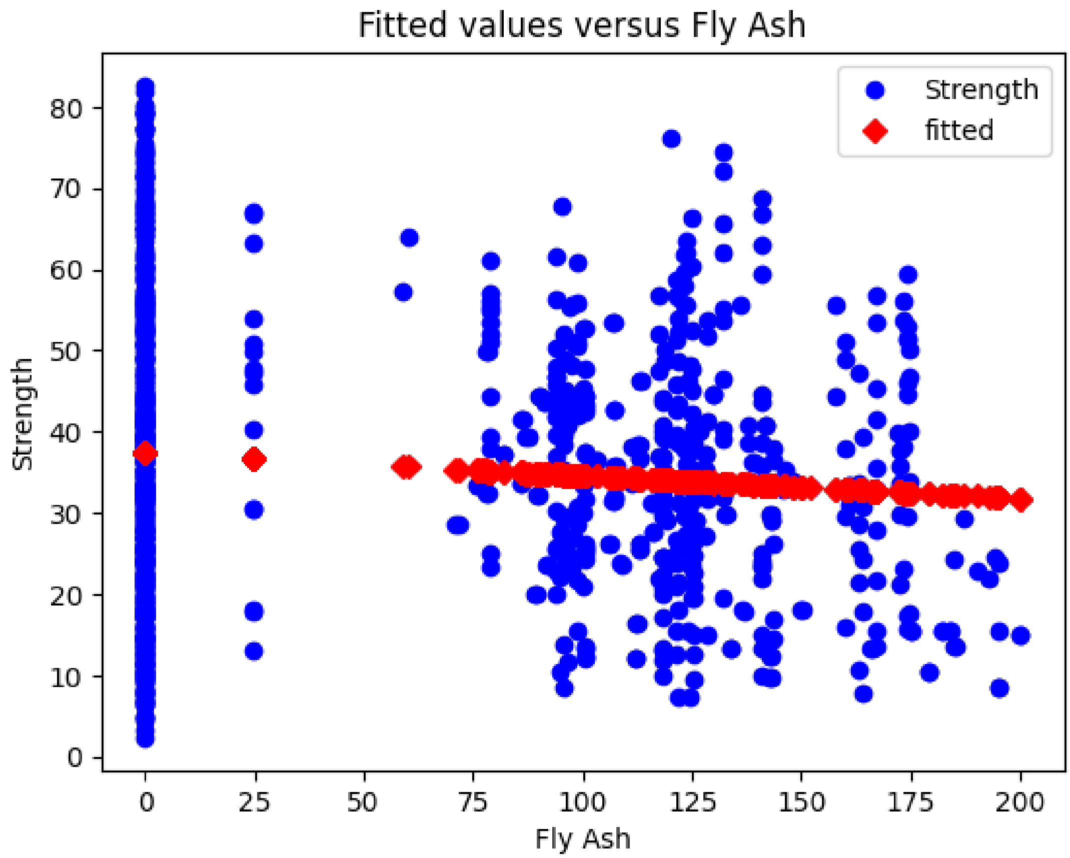
When the quantiles of two variables are plotted against each other, then the plot obtained is known as quantile – quantile plot or qqplot. This plot provides a summary of whether the distributions of two variables are similar or not with respect to the locations.

```
sm.qqplot(model_result.resid, line='s');
```



Fit Plot

```
sm.graphics.plot_fit(model_result,1, vlines=False);
```


Fitted values versus Fly Ash
Strength
fitted
Fly Ash
Strength
0
10
20
30
40
50
60
70
80
0
25
50
75
100
125
150
175
200