Main

Libraries

In [82]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from matplotlib.lines import Line2D
from textwrap import wrap
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

Loading The Data

In [70]:
```
df_concrete = pd.read_csv("train.csv").drop(columns=["id"])
print(f"Duplicates: {df_concrete.duplicated().sum():4}")
print(f"Length: {len(df_concrete)}")
```

```
Duplicates:    0
Length: 5407
```

Data Familiarization

In [71]:
```
df_concrete.head()
```

Out[71]:

	CementComponent	BlastFurnaceSlag	FlyAshComponent	WaterComponent	Superplastici
0	525.0	0.0	0.0	186.0	
1	143.0	169.0	143.0	191.0	
2	289.0	134.7	0.0	185.7	
3	304.0	76.0	0.0	228.0	
4	157.0	236.0	0.0	192.0	

Useful Functions

In [72]:
```
def plot_histogram(data, rows, cols):

    _, axes = plt.subplots(rows, cols, figsize=(12, 10))

    axes = axes.flatten()

    for ax, col in zip(axes, data.columns):

        current_data = data[col]

        sns.histplot(current_data, kde=True, bins=30, ax=ax)

        mean_val = current_data.mean()
        median_val = current_data.median()
        std_val = current_data.std()
        min_val = current_data.min()
        max_val = current_data.max()

        ax.axvline(mean_val, color='r', linestyle='--', linewidth=2, label=f'Mean:
        ax.axvline(median_val, color='g', linestyle='-', linewidth=2, label=f'Media
        ax.axvline(mean_val - std_val, color='b', linestyle=':', linewidth=2, label
        ax.axvline(mean_val + std_val, color='b', linestyle=':', linewidth=2)


        dummy_min = Line2D([0], [0], color='none', label=f'Min: {min_val:.2f}')
        dummy_max = Line2D([0], [0], color='none', label=f'Max: {max_val:.2f}')

        handles, labels = ax.get_legend_handles_labels()
        handles.extend([dummy_min, dummy_max])
        labels.extend([f'Min: {min_val:.2f}', f'Max: {max_val:.2f}'])
        ax.legend(handles=handles, labels=labels, loc='center left', bbox_to_anchor

        ax.set_title(f'Distribution of {col}')
        ax.set_xlabel(col)
        ax.set_ylabel('Frequency')

    plt.tight_layout()

    plt.show()
```

Descriptive Statistics

Statistical Summary of the Data

In [73]:
```
df_concrete.describe().transpose()
```

Out[73]:

	count	mean	std	min	25%	50%	75%
CementComponent	5407.0	299.168189	105.537682	102.00	213.70	297.20	375.0(
BlastFurnaceSlag	5407.0	58.610579	83.417801	0.00	0.00	0.00	122.6(
FlyAshComponent	5407.0	31.872795	54.605003	0.00	0.00	0.00	79.0(
WaterComponent	5407.0	185.076235	18.517583	121.80	175.10	187.40	192.0(
SuperplasticizerComponent	5407.0	4.108441	5.692296	0.00	0.00	0.00	8.0
CoarseAggregateComponent	5407.0	992.000718	77.148010	801.00	938.20	978.00	1047.0(
FineAggregateComponent	5407.0	771.219974	78.725253	594.00	734.30	781.20	821.0(
AgeInDays	5407.0	51.751618	70.006975	1.00	7.00	28.00	56.0(
Strength	5407.0	35.452071	16.401896	2.33	23.64	33.95	45.8

Distributions

In [ ]:
```
df_concrete_scaled = df_concrete.copy()
ss = StandardScaler()
df_concrete_scaled[df_concrete.columns] = ss.fit_transform(df_concrete[df_concrete.

df_concrete_robustScaled = df_concrete.copy()
rs = RobustScaler()
df_concrete_robustScaled[df_concrete.columns] = rs.fit_transform(df_concrete[df_con

df_concrete_minMax = df_concrete.copy()
mm = MinMaxScaler()
df_concrete_minMax[df_concrete.columns] = mm.fit_transform(df_concrete[df_concrete.

plot_histogram(df_concrete, 5, 2)
plot_histogram(df_concrete_scaled, 5, 2)
plot_histogram(df_concrete_robustScaled, 5, 2)
plot_histogram(df_concrete_minMax, 5, 2)
```

Box Plot(Outliers)

In [ ]:
```
plt.figure(figsize=(10, 6))
ax = df_concrete.boxplot()
plt.title('Combined Boxplots')
plt.ylabel('Values')
new_labels = [
    "\n".join(wrap(label.get_text(), 10))
    for label in ax.get_xticklabels()
]
ax.set_xticklabels(new_labels, rotation=0)
plt.show()
```

Correlation Analysis

In [ ]:
```
correlation_table = df_concrete.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_table, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.tight_layout()
plt.show()
```

Scatterplot Matrix

In [ ]:
```
sns.set_theme(font_scale=0.75)
sns.pairplot(df_concrete, hue='Strength', height=1.5, markers='.', plot_kws={'s': 3
plt.show()
```

Feature Engineering

In [ ]:
```

```

Baseline Model

In [97]:
```
train = pd.read_csv("train.csv").drop(columns=["id"])
test = pd.read_csv("test.csv").drop(columns=["id"])

X = train.drop(columns=["Strength"])
y = train["Strength"]

X_scaled = X.copy()

ss = StandardScaler()

X_scaled[X.columns] = ss.fit_transform(X[X.columns])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Linear Regression Baseline Model

In [98]:
```
# Linear Regression
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
pred = model_lr.predict(X_test)
r2_score_result = r2_score(y_test, pred)
rmse = np.sqrt(np.mean((y_test - pred) ** 2))
print('The result of rmse is: ', rmse)
print('The result of r2 is: ', r2_score_result)

plt.figure(figsize=(8, 6))
```

```
plt.scatter(y_test, pred, alpha=0.5, edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', l
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Predictions vs. Actual Values')
plt.legend()
plt.tight_layout()
plt.show()
```

```
The result of rmse is:  14.483769165661862
The result of r2 is:  0.1825270446731332
```