INTRODUCTION TO CONCRETE ANALYSIS WORKSHOP

Part 1 - Importing necessary libraries and loading the data

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

def load_data(file_path):
    try:
        data = pd.read_excel(file_path)
        return data
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        return None
```

We import pandas, numpy, matplotlib and seaborn to help with the execution of data processing, complex maths and visualizations of said data.
But we also imported from sklearn's machine learning libraries for model selection, processing the data and etc.

We load the data as we did in other workshops except we used an .XLS file, not a .CSV file so we used pd.read_excel( ).

Mustafa Aksu - 2021403018

INTRODUCTION TO CONCRETE ANALYSIS WORKSHOP

Part 2 - Creating Visualizations

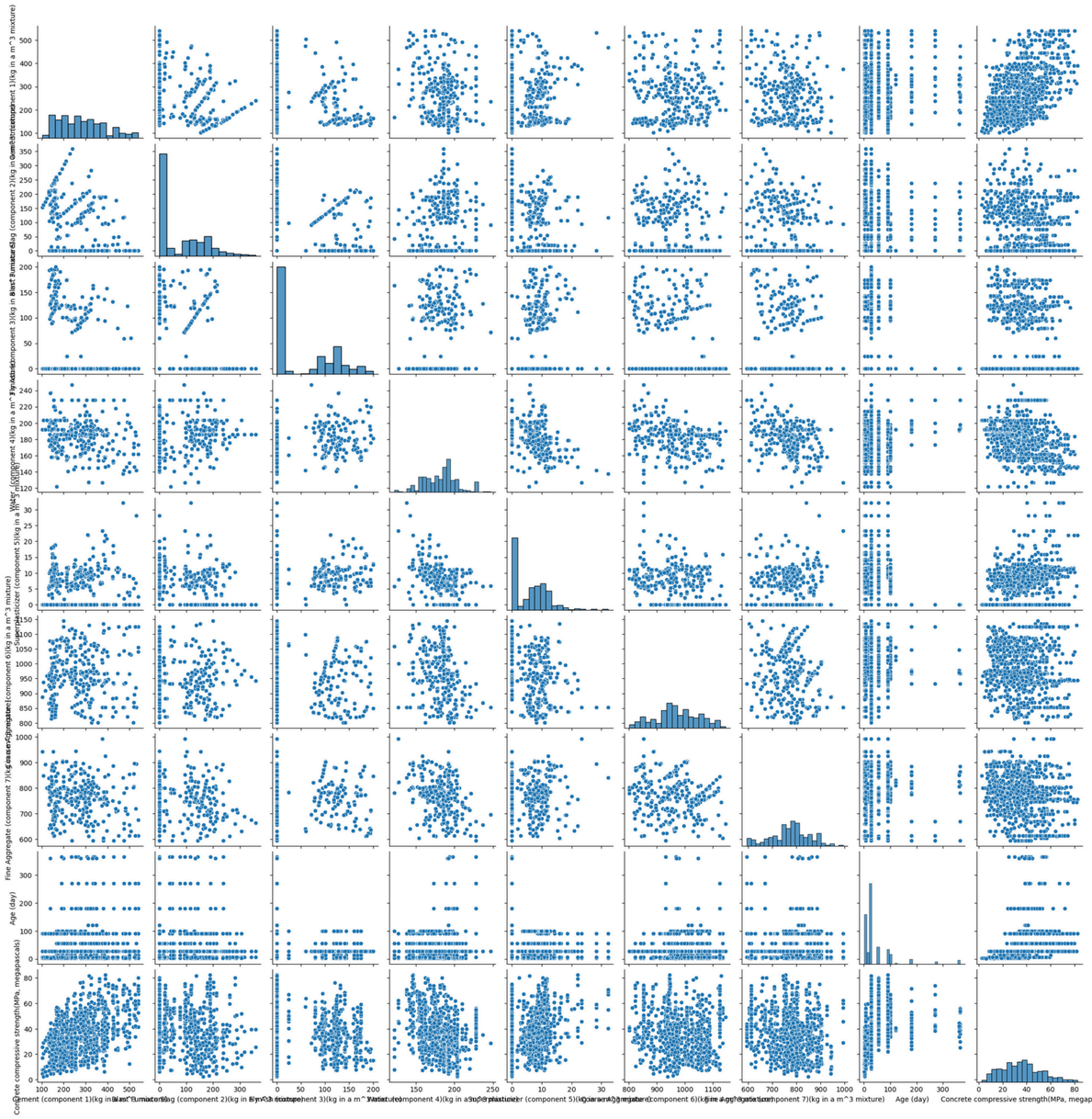
```python
def create_visualizations(data):

    features = data.columns[:-1]
    target = data.columns[-1]

    # Calculate number of rows needed for subplots
    n_features = len(features)
    n_rows = (n_features + 1) // 2  # 2 plots per row, rounded up

    # Create figure with subplots
    plt.figure(figsize=(15, 4*n_rows))

    # Create a scatter plot for each feature vs compressive strength
    for i, feature in enumerate(features, 1):
        plt.subplot(n_rows, 2, i)
        plt.scatter(data[feature], data[target], alpha=0.5)
        plt.xlabel(feature)
        plt.ylabel('Compressive Strength')
        plt.title(f'{feature} vs Compressive Strength')

    plt.tight_layout()  # Adjust spacing between subplots
```

For creating visualizations at first, I thought to use sns.pairplot function but since we had so many parameters, the plot would be so useless as shown below.

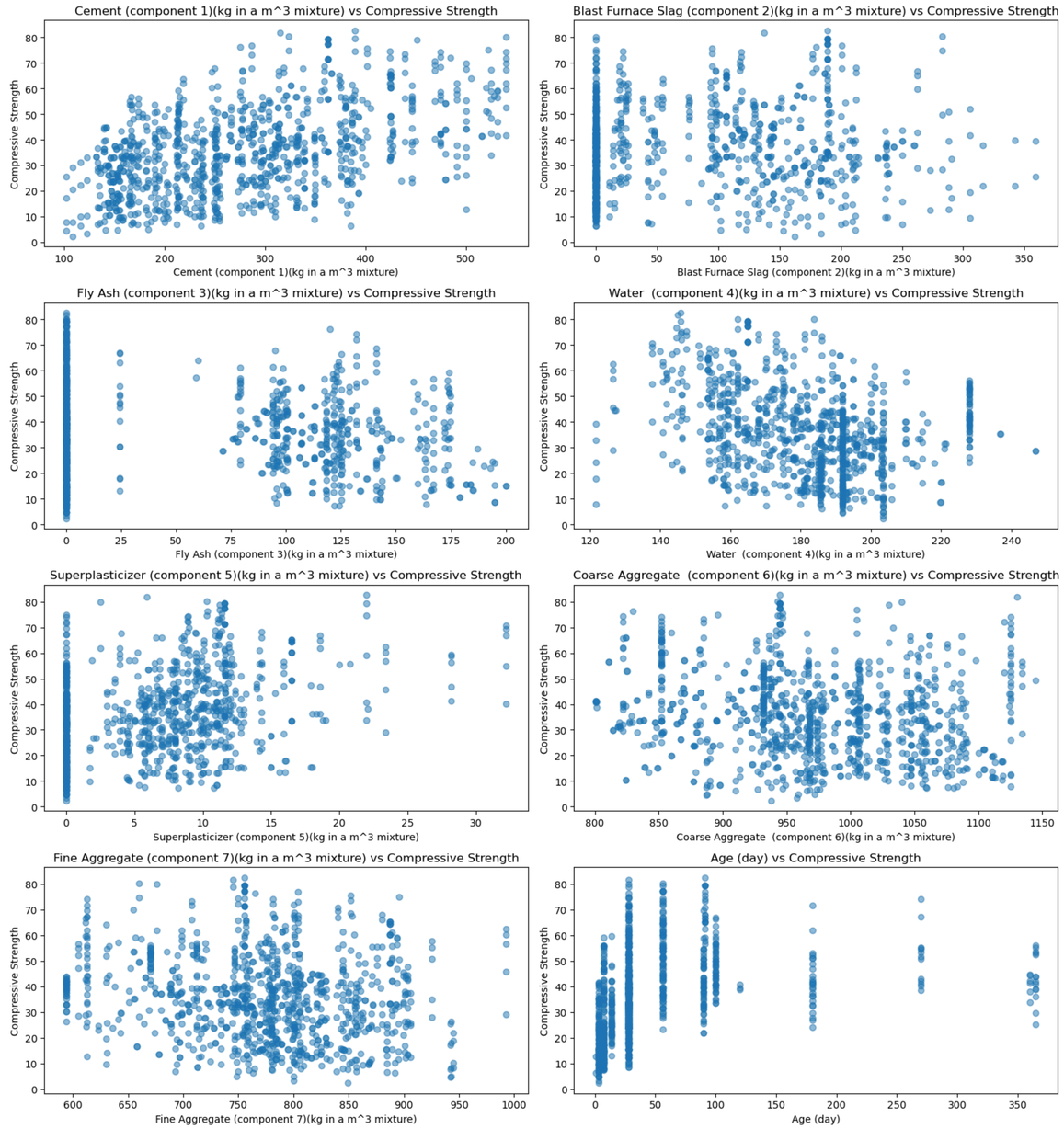
So I created a scatterplot comparing each feature vs compressive strength with the help of AI as shown below.

Mustafa Aksu - 2021403018

Pairplot



Scatterplot vs every feature



Mustafa Aksu - 2021403018

INTRODUCTION TO CONCRETE ANALYSIS WORKSHOP

Part 3 - Preprocessing the data and running the model

```
def preprocess_data(data):

    data = data.fillna(data.mean())

    target_column = data.columns[-1]

    X = data.drop(target_column, axis=1)
    y = data[target_column]

    X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

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

    return X_train_scaled, X_test_scaled, y_train, y_test

    def train_and_evaluate_model(X_train_scaled, X_test_scaled,
    y_train, y_test):

    model = LinearRegression()
    model.fit(X_train_scaled, y_train)

    y_pred = model.predict(X_test_scaled)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error: {mse:.4f}")
    print(f"R² Score: {r2:.4f}"
```

We first fill the missing values from the data with the mean of column.
Then we select the target column, which is the last column(compressive strength)
Then we separate the features and targets where X is the features and it's every parameter other than compressive strength and Y is the target as mentioned above.
Then we split the data for training and testing the model, where the test size is 20% of the data and training is the %80.
At last, we standardized the data with use of standardscaler , fit_transform and transform.

Then we trained a linear regression model with said scaled data collected predicted values from the model.

After that we compared our predicted values with the test values and with the library's help we calculated MSE and R^2 score to evaluate the model.

Mustafa Aksu - 2021403018

INTRODUCTION TO CONCRETE ANALYSIS WORKSHOP

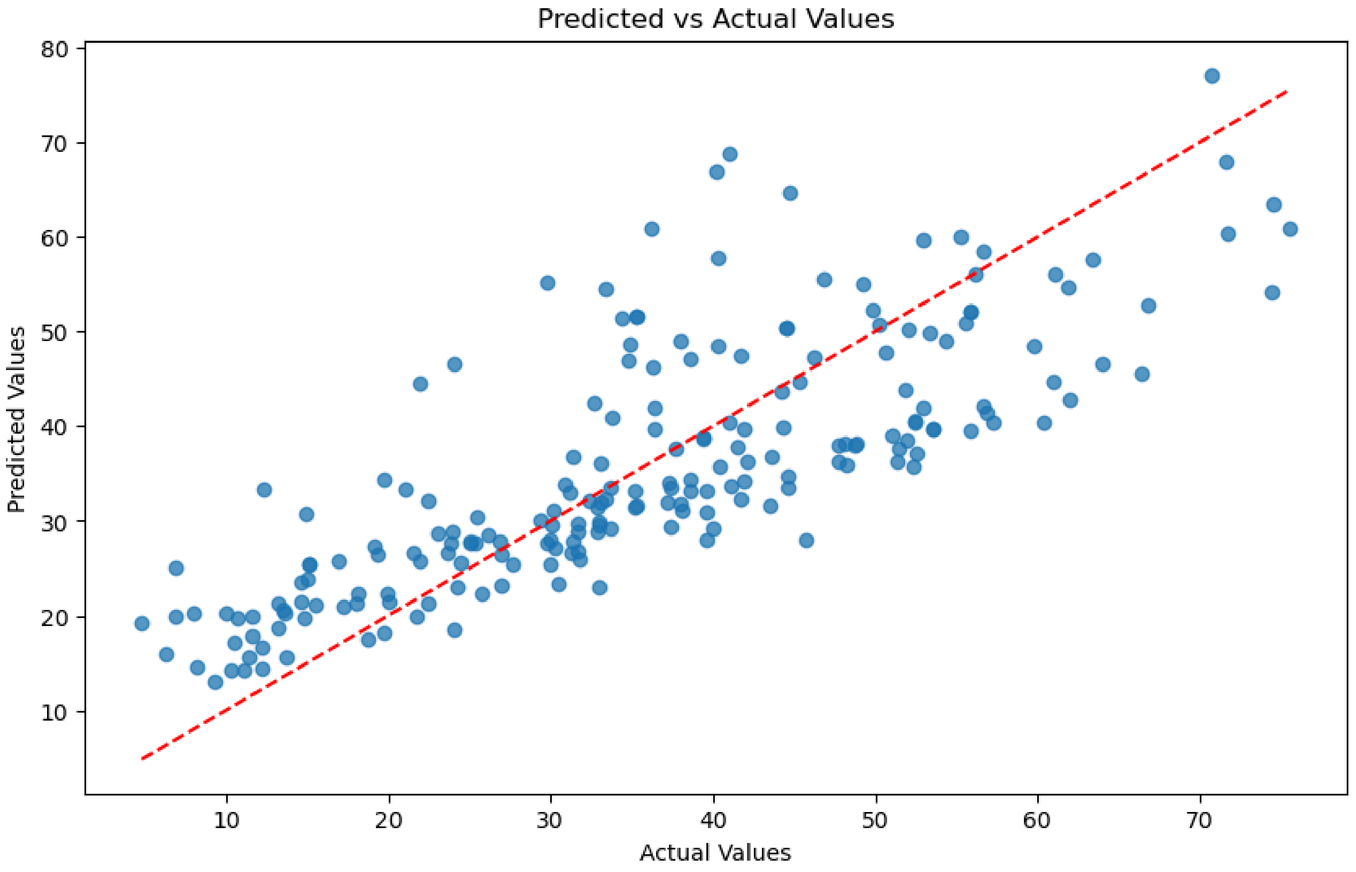
Part 4 - Plotting the Results

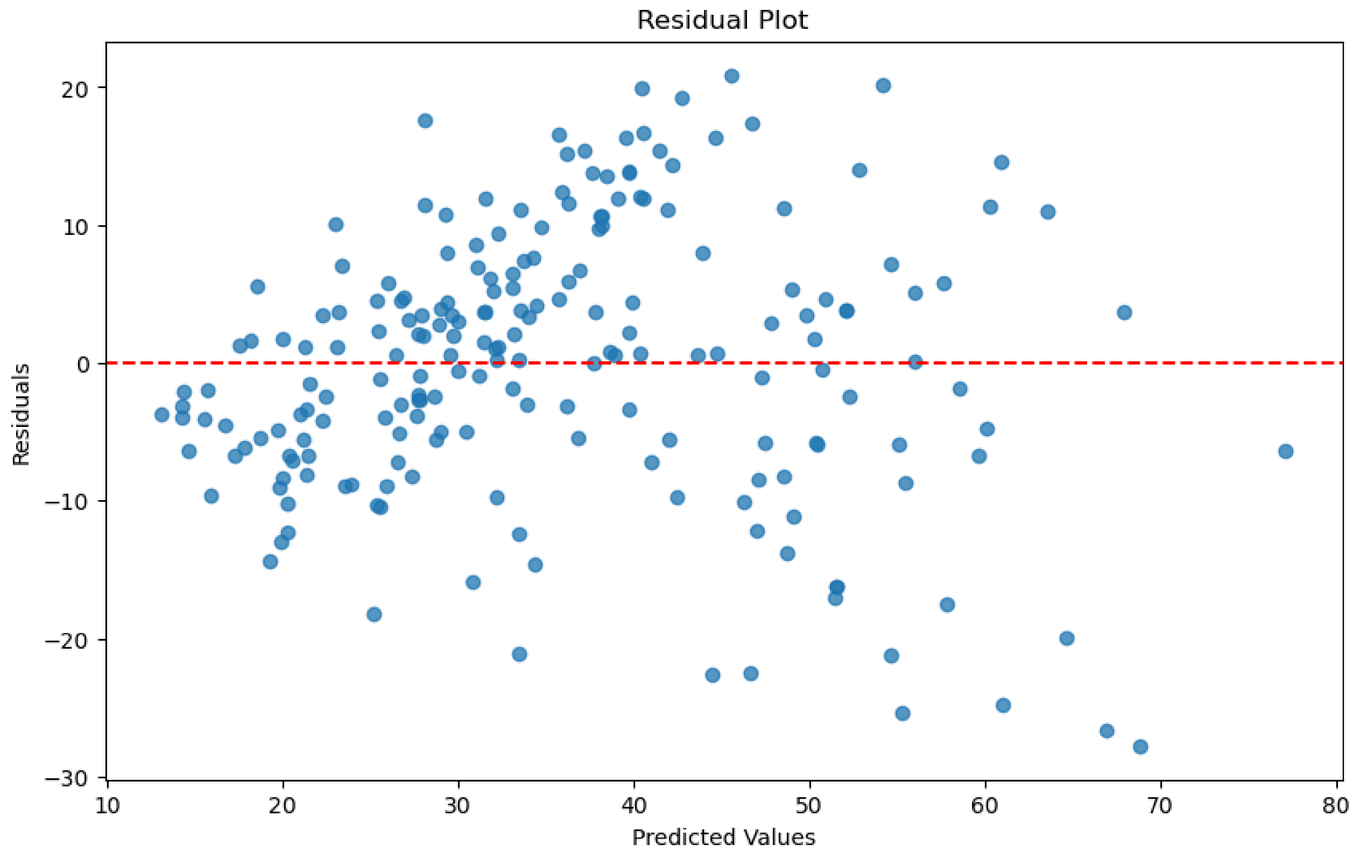
```
def plot_results(y_test, y_pred):
    """Create visualization plots for model results."""
    # Plot predicted vs actual values
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, alpha=0.75)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(),
 y_test.max()], 'r--')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.title('Predicted vs Actual Values')

    # Plot residuals
    residuals = y_test - y_pred
    plt.figure(figsize=(10, 6))
    plt.scatter(y_pred, residuals, alpha=0.75)
    plt.axhline(y=0, color='r', linestyle='--')
    plt.xlabel('Predicted Values')
    plt.ylabel('Residuals')
    plt.title('Residual Plot')
```

We plotted a predicted vs actual values with the help of matplotlib as shown below.

Also we plotted a chart where we compared the residuals vs predicted values. Residuals as I've learned meant the difference between observed value and predicted value.

Mustafa Aksu - 2021403018





Mustafa Aksu - 2021403018

INTRODUCTION TO CONCRETE ANALYSIS WORKSHOP

Part 5 - Executing the script

```python
def main():
    concrete_excel = "../../datasets/concrete_strength/Concrete_Data.xls"
    data = load_data(concrete_excel)

    print("\nFirst few rows of the dataset:")
    print(data.head())
    print("\nSummary statistics:")
    print(data.describe())

    # Create visualizations
    create_visualizations(data)

    # Preprocess the data
    X_train_scaled, X_test_scaled, y_train, y_test = preprocess_data(data)

    # Train and evaluate the model
    model, y_pred = train_and_evaluate_model(X_train_scaled, X_test_scaled, y_train, y_test)

    # Create result visualizations
    plot_results(y_test, y_pred)

if __name__ == "__main__":
    main()
```

Mustafa Aksu - 2021403018