Project 1

Sareh Jalalizad

Question 1

Code

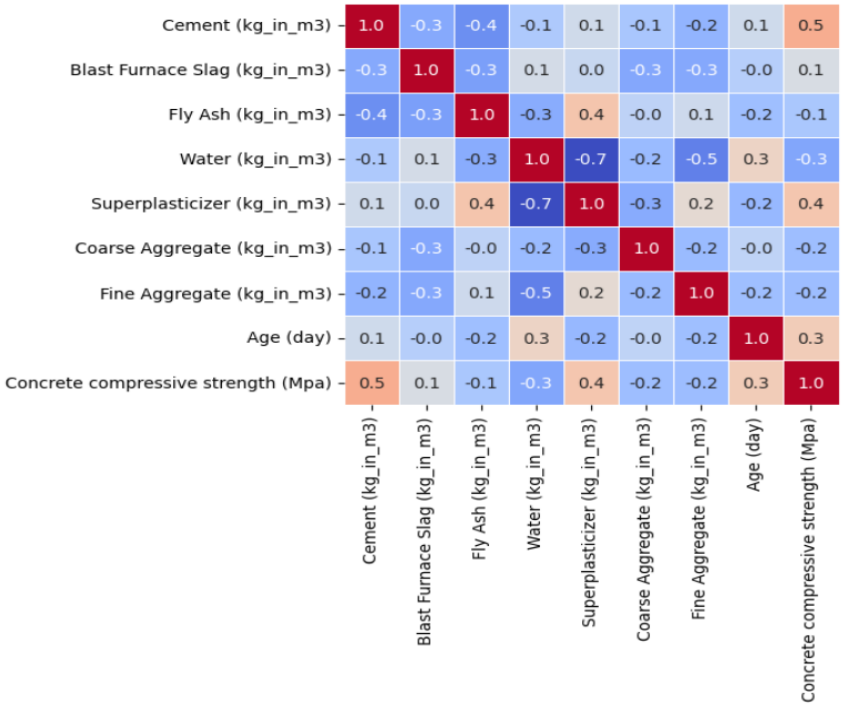
```
# Generate a correlation matrix.
# Write your code between the lines (~ 1 line)
################################################
# Generate a correlation matrix
correlation_matrix = dataset.corr()

# Find the two features with the highest correlation to the target
highest = correlation_matrix['Concrete compressive strength (Mpa)'].nlargest(3)[1:3]

print("Correlation Matrix:")
print(correlation_matrix)
print("\nTwo features with highest correlation to the target:")
print(highest)


sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".1f", linewidths=0.5)
plt.show()
################################################
```

Output



Correlation Matrix:

	Cement (kg_in_m3)	Blast Furnace Slag (kg_in_m3)	Fly Ash (kg_in_m3)	Water (kg_in_m3)	Superplasticizer (kg_in_m3)	Coarse Aggregate (kg_in_m3)	Fine Aggregate (kg_in_m3)	Age (day)	Concrete compressive strength (Mpa)
Cement (kg_in_m3)	1.0	-0.3	-0.4	-0.1	0.1	-0.1	-0.2	0.1	0.5
Blast Furnace Slag (kg_in_m3)	-0.3	1.0	-0.3	0.1	0.0	-0.3	-0.3	-0.0	0.1
Fly Ash (kg_in_m3)	-0.4	-0.3	1.0	-0.3	0.4	-0.0	0.1	-0.2	-0.1
Water (kg_in_m3)	-0.1	0.1	-0.3	1.0	-0.7	-0.2	-0.5	0.3	-0.3
Superplasticizer (kg_in_m3)	0.1	0.0	0.4	-0.7	1.0	-0.3	0.2	-0.2	0.4
Coarse Aggregate (kg_in_m3)	-0.1	-0.3	-0.0	-0.2	-0.3	1.0	-0.2	-0.0	-0.2
Fine Aggregate (kg_in_m3)	-0.2	-0.3	0.1	-0.5	0.2	-0.2	1.0	-0.2	-0.2
Age (day)	0.1	-0.0	-0.2	0.3	-0.2	-0.0	-0.2	1.0	0.3
Concrete compressive strength (Mpa)	0.5	0.1	-0.1	-0.3	0.4	-0.2	-0.2	0.3	1.0

Two features with highest correlation to the target:

Concrete compressive strength (Mpa)	1.0
Cement (kg_in_m3)	0.5

Two features with highest correlation to the target:

Cement (kg_in_m3)	0.5
Superplasticizer (kg_in_m3)	0.4

Question 2

Code

```
# Split the dataset into input features and the output target.
# Write your code between the lines (~ 2 lines)
################################################
target_column = "Concrete compressive strength (Mpa)"
y_dataset = dataset.pop(target_column)
X_dataset = dataset
################################################
print("Dataset separated.")
```

Output

```
Dataset separated.
```

Question 3

Code

```
# Convert raw values to their Z-scores
# Calculate the Z-scores of each input feature column.
# Write your code between the lines (~ 3 lines)
################################################
X_dataset_mean = X_dataset.mean()
X_dataset_std = X_dataset.std()
X_dataset_norm = (X_dataset - X_dataset_mean) / X_dataset_std
################################################
print("Dataset normalized.")
```

Output

```
Dataset normalized.
```

Question 4

Code

```
from sklearn.model_selection import train_test_split

# Split the dataset into the training set (80%) and the test set (20%).
# Write your code between the lines (~ 1 line)
################################################
X_train_norm, X_test_norm, y_train, y_test = train_test_split(X_dataset_norm, y_dataset, test_size=0.2, random_state=100)
################################################
print("Dataset split.")
```

Output

```
Dataset split.
```

Question 5

Code

```
# Create a linear regression object
model_LR = LinearRegression()

# Train the model using the training data
# Write your code between the lines (~ 1 line)
################################################
model_LR.fit(X_train_norm, y_train)
################################################

# Print model's parameters
print(model_LR.intercept_, "\n")
print(model_LR.coef_)
```

Output

```
35.917516549806784

[12.9748385   8.94444004  5.97546413 -2.86774364  1.72072076  1.60620725
  2.05493992  7.24036633]
```

Question 6

Code and output

```
# Predict the output for the test data
y_pred_LR = model_LR.predict(X_test_norm)

# Evaluate the performance using the mean squared error
print(mean_squared_error(y_test, y_pred_LR))
```

```
113.1787593778991
```

Question 7

Code

```
# find the number of input features
n_features = X_train_norm.shape[1]


# Create the neural network
model_NN = tf.keras.Sequential([
    layers.Dense(32, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(l=0.001), input_shape=(n_features,)),
    # Write your code between the lines (~ 2 lines)
    ################################################
    layers.Dense(16, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(l=0.001)),
    layers.Dense(8, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(l=0.001)),
    ################################################
    layers.Dense(1)
])
```

Question 8

Code

```
[20] # compile the model
     model_NN.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_squared_error'])

     # fit the model
     # Write your code between the lines (~ 1 line)
     ################################################
     model_NN.fit(X_train_norm, y_train, epochs=200, batch_size=30)
     ################################################
```

Question 9

Code and output

```
# Evaluate the performance using the mean squared error
model_NN.evaluate(X_test_norm, y_test)
```

```
7/7 [==============================] - 0s 2ms/step - loss: 35.4831 - mean_squared_error: 35.3441
[35.48311233520508, 35.344139099121094]
```

Question 10

The neural network model performed better against the test set in predicting the output. The mean squared error (MSE) for the neural network model on the test set (approximately 35.34) was lower than the MSE for the multivariate linear regression model (approximately 113.18), which indicates that the neural network was more accurate in its predictions compared to the linear regression model.

Regarding overfitting, for the multivariate linear regression model, the mean squared error on the training set was approximately 105.97, and the MSE on the test set was approximately 113.18, and the difference between these two values is relatively small, indicating that the model did not overfit.
For the neural network model, the MSE on the training set was approximately 14.39, and for the test set was approximately 35.34. The difference between these two values is somewhat larger, indicating that the neural network model may exhibit slight overfitting.