report-predictive-modelling

October 25, 2023

[3]:
```
Phase4: Exploratory Data Analysis

1) In Coding file - Perform the Exploratory Data Analysis and submit the␣
 ↪assignment in ipynb and pdf format as
"Report - Exploratory Data Analysis" . And this needs to be done by eod of␣
 ↪starting of new task (Check document for timeline).Mention Below details in␣
 ↪the document:

Name Renuka Hatwalne
Batch March 2023
Project Name Cement_Live project

Go through the Dataset and implement data cleaning/data pre-processing␣
 ↪techniques which comprises of below pointers:

1. Explore each independent feature w.r.t dependent variable(Compression␣
 ↪Strength)
2. Perform the Uni-variant, Bi-variant and Multi-variant data analysis of each␣
 ↪features.
3. Plot the co-relation of other features with Compression Strength.
4. Check the outliers in the dataset using Boxplot and various techniques.
5. With the help of graph check the skeweness of the dataset
6. Check distribution of each independent variables  using scatter plot and␣
 ↪also using QQ plot to understand column distribution.
7. Ploting the Dataset Distribution to check every density and skewness.
8. Scaling the Dataset.
9. If columns are not gaussian distributed then make it normal distribution.
10. If dataset is too noisy then apply power transformer (Yeo-Jhonson)
11. Once obtained the cleaned and normally distributed datset.
12. Select the important feature for modelling.
Note:
1. Provide access to munirainsideaiml@gmail.com
2. Deadline: Submission will be done by 11:59 PM (9th Oct, 2023)
3. Maintain Professional decorum
```

```
  Cell In[3], line 3
```

```
    1) In Coding file - Perform the Exploratory Data Analysis and submit the␣
 ↪assignment in ipynb and pdf format as
      ^
SyntaxError: unmatched ')'
```

[ ]:
```
#1. Explore each independent feature w.r.t dependent variable(Compression␣
 ↪Strength)
```

[6]:
```
import pandas as pd
import matplotlib.pyplot as plt
# Replace 'your_file.csv' with the actual name of your CSV file
file_path = 'Material.csv'
# Read the CSV file into a Pandas DataFrame
df = pd.read_csv(file_path)
# Display the first few rows of the DataFrame to verify it was read correctly
df.head()
```

[6]:

	Material Quantity (gm)	Additive Catalyst (gm)	Ash Component (gm)	\
0	486.42	180.60	21.26	
1	133.32	260.14	185.60	
2	559.97	2.84	111.76	
3	391.43	351.05	76.39	
4	394.78	352.61	194.35	

	Water Mix (ml)	Plasticizer (gm)	Moderate Aggregator	Refined Aggregator	\
0	201.66	16.11	1151.17	708.50	
1	175.99	6.27	1090.57	1010.25	
2	295.23	11.95	1024.93	810.69	
3	299.14	19.00	1134.88	881.34	
4	235.54	17.02	1098.24	781.01	

	Formulation Duration (hrs)	Compression Strength MPa
0	344.43	79.89
1	28.86	59.80
2	237.68	77.86
3	208.81	71.74
4	266.84	76.07

[8]:
```
df.columns = df.columns.str.replace(' ', '_')
```

[9]:
```
df = df.drop_duplicates()
```

[10]:
```
df = df.fillna(df.mean())
```

[11]:
```
df
```

[11]:

	Material_Quantity_(gm)	Additive_Catalyst_(gm)	Ash_Component_(gm)	\
0	486.42	180.60	21.26	
1	133.32	260.14	185.60	
2	559.97	2.84	111.76	
3	391.43	351.05	76.39	
4	394.78	352.61	194.35	
…	…	…	…	
6134	188.78	162.30	142.65	
6135	349.87	291.45	77.82	
6136	358.29	22.70	17.99	
6137	445.25	275.59	178.86	
6138	560.23	266.56	167.14	

	Water_Mix_(ml)	Plasticizer_(gm)	Moderate_Aggregator	\
0	201.66	16.11	1151.17	
1	175.99	6.27	1090.57	
2	295.23	11.95	1024.93	
3	299.14	19.00	1134.88	
4	235.54	17.02	1098.24	
…	…	…	…	
6134	163.66	15.98	1003.82	
6135	188.26	25.82	925.10	
6136	208.58	34.91	1081.07	
6137	191.77	18.07	865.15	
6138	175.49	10.63	1165.87	

	Refined_Aggregator	Formulation_Duration_(hrs)	Compression_Strength_MPa
0	708.50	344.43	79.89
1	1010.25	28.86	59.80
2	810.69	237.68	77.86
3	881.34	208.81	71.74
4	781.01	266.84	76.07
…	…	…	…
6134	1002.47	357.91	50.61
6135	1005.31	104.20	54.24
6136	792.44	302.76	56.57
6137	833.10	374.63	58.21
6138	894.53	360.96	58.96

[6111 rows x 9 columns]

[13]:
```
import pandas as pd
from scipy.stats import zscore
df_standardized = df.apply(zscore)
```

[14]:
```
df_standardized
```

[14]:

	Material_Quantity_(gm)	Additive_Catalyst_(gm)	Ash_Component_(gm)	\
0	0.692801	-0.122121	-1.240761	
1	-1.679314	0.478770	0.998448	
2	1.186907	-1.465023	-0.007657	
3	0.054661	1.165557	-0.489589	
4	0.077166	1.177342	1.117670	
…	…	…	…	
6134	-1.306735	-0.260370	0.413234	
6135	-0.224538	0.715304	-0.470105	
6136	-0.167973	-1.314989	-1.285316	
6137	0.416222	0.595488	0.906612	
6138	1.188654	0.527270	0.746922	

	Water_Mix_(ml)	Plasticizer_(gm)	Moderate_Aggregator	\
0	-0.551920	-0.133391	1.570917	
1	-1.174617	-0.981269	0.945634	
2	1.717882	-0.491843	0.268346	
3	1.812729	0.115630	1.402834	
4	0.269934	-0.054979	1.024774	
…	…	…	…	
6134	-1.473716	-0.144592	0.050529	
6135	-0.876974	0.703286	-0.761720	
6136	-0.384056	1.486539	0.847611	
6137	-0.791830	0.035496	-1.380297	
6138	-1.186746	-0.605583	1.722595	

	Refined_Aggregator	Formulation_Duration_(hrs)	Compression_Strength_MPa
0	-0.922174	1.522128	1.433540
1	1.772742	-1.311449	0.185721
2	-0.009520	0.563594	1.307453
3	0.621452	0.304364	0.927331
4	-0.274590	0.825429	1.196274
…	…	…	…
6134	1.703259	1.643168	-0.385083
6135	1.728623	-0.634954	-0.159619
6136	-0.172510	1.147963	-0.014899
6137	0.190623	1.793301	0.086964
6138	0.739252	1.670555	0.133547

[6111 rows x 9 columns]

[17]:
```
df_standardized.columns
```

[17]:
```
Index(['Material_Quantity_(gm)', 'Additive_Catalyst_(gm)',
       'Ash_Component_(gm)', 'Water_Mix_(ml)', 'Plasticizer_(gm)',
       'Moderate_Aggregator', 'Refined_Aggregator',
       'Formulation_Duration_(hrs)', 'Compression_Strength_MPa'],
```

```
      dtype='object')
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

[19]:
```
#2.Correlation Analysis (Numeric-Numeric)
df_standardized.corr()
```

[19]:

	Material_Quantity_(gm)	Additive_Catalyst_(gm)	\
Material_Quantity_(gm)	1.000000	0.009915	
Additive_Catalyst_(gm)	0.009915	1.000000	
Ash_Component_(gm)	-0.020792	0.053135	
Water_Mix_(ml)	0.006496	0.029454	
Plasticizer_(gm)	0.048439	0.140341	
Moderate_Aggregator	-0.007077	-0.023353	
Refined_Aggregator	-0.017512	0.010720	
Formulation_Duration_(hrs)	0.068994	0.162147	
Compression_Strength_MPa	0.129241	0.179600	

	Ash_Component_(gm)	Water_Mix_(ml)	\
Material_Quantity_(gm)	-0.020792	0.006496	
Additive_Catalyst_(gm)	0.053135	0.029454	
Ash_Component_(gm)	1.000000	-0.011093	
Water_Mix_(ml)	-0.011093	1.000000	
Plasticizer_(gm)	0.161586	-0.024860	
Moderate_Aggregator	-0.007257	-0.032200	
Refined_Aggregator	0.041248	-0.053895	
Formulation_Duration_(hrs)	0.103810	0.028321	
Compression_Strength_MPa	0.094633	-0.025387	

	Plasticizer_(gm)	Moderate_Aggregator	\
Material_Quantity_(gm)	0.048439	-0.007077	
Additive_Catalyst_(gm)	0.140341	-0.023353	
Ash_Component_(gm)	0.161586	-0.007257	
Water_Mix_(ml)	-0.024860	-0.032200	
Plasticizer_(gm)	1.000000	-0.019667	
Moderate_Aggregator	-0.019667	1.000000	
Refined_Aggregator	0.056319	-0.004989	
Formulation_Duration_(hrs)	0.156638	0.005475	
Compression_Strength_MPa	0.206862	-0.030385	

	Refined_Aggregator	Formulation_Duration_(hrs)	\
Material_Quantity_(gm)	-0.017512	0.068994	
Additive_Catalyst_(gm)	0.010720	0.162147	
Ash_Component_(gm)	0.041248	0.103810	
Water_Mix_(ml)	-0.053895	0.028321	

```
Plasticizer_(gm)                    0.056319                    0.156638
Moderate_Aggregator                -0.004989                    0.005475
Refined_Aggregator                  1.000000                    0.007182
Formulation_Duration_(hrs)          0.007182                    1.000000
Compression_Strength_MPa           -0.010433                    0.269526
```

```
                            Compression_Strength_MPa
Material_Quantity_(gm)                      0.129241
Additive_Catalyst_(gm)                      0.179600
Ash_Component_(gm)                          0.094633
Water_Mix_(ml)                             -0.025387
Plasticizer_(gm)                            0.206862
Moderate_Aggregator                        -0.030385
Refined_Aggregator                         -0.010433
Formulation_Duration_(hrs)                  0.269526
Compression_Strength_MPa                    1.000000
```

[27]:
```
X = df_standardized.drop("Compression_Strength_MPa", axis = 1).values
```

[28]:
```
X
```

[28]:
```
array([[ 0.69280058, -0.1221213 , -1.24076099, …,  1.57091722,
        -0.92217389,  1.52212782],
       [-1.67931397,  0.47876987,  0.99844761, …,  0.9456337 ,
         1.7727418 , -1.31144941],
       [ 1.18690714, -1.46502317, -0.00765658, …,  0.2683464 ,
        -0.00951959,  0.56359447],
       …,
       [-0.16797267, -1.31498924, -1.28531625, …,  0.8476107 ,
        -0.17250952,  1.14796309],
       [ 0.41622184,  0.5954881 ,  0.90661199, …, -1.38029718,
         0.19062311,  1.79330072],
       [ 1.18865381,  0.52727026,  0.74692157, …,  1.7225949 ,
         0.73925168,  1.67055458]])
```

[29]:
```
y = df_standardized['Compression_Strength_MPa'].values
```

[30]:
```
y
```

[30]:
```
array([ 1.43353962,  0.18572074,  1.30745339, …, -0.01489922,
        0.08696354,  0.13354712])
```

[51]:
```
df_standardized
```

[51]:
```
      Material_Quantity_(gm)  Additive_Catalyst_(gm)  Ash_Component_(gm)  \
0                   0.692801               -0.122121           -1.240761
1                  -1.679314                0.478770            0.998448
```

```
2                    1.186907               -1.465023           -0.007657
3                    0.054661                1.165557           -0.489589
4                    0.077166                1.177342            1.117670
…                         …                       …                   …
6134                -1.306735               -0.260370            0.413234
6135                -0.224538                0.715304           -0.470105
6136                -0.167973               -1.314989           -1.285316
6137                 0.416222                0.595488            0.906612
6138                 1.188654                0.527270            0.746922

      Water_Mix_(ml)  Plasticizer_(gm)  Moderate_Aggregator  \
0          -0.551920         -0.133391             1.570917
1          -1.174617         -0.981269             0.945634
2           1.717882         -0.491843             0.268346
3           1.812729          0.115630             1.402834
4           0.269934         -0.054979             1.024774
…                ...               …                    …
6134       -1.473716         -0.144592             0.050529
6135       -0.876974          0.703286            -0.761720
6136       -0.384056          1.486539             0.847611
6137       -0.791830          0.035496            -1.380297
6138       -1.186746         -0.605583             1.722595

      Refined_Aggregator  Formulation_Duration_(hrs)  Compression_Strength_MPa
0              -0.922174                    1.522128                  1.433540
1               1.772742                   -1.311449                  0.185721
2              -0.009520                    0.563594                  1.307453
3               0.621452                    0.304364                  0.927331
4              -0.274590                    0.825429                  1.196274
…                    …                           …                         …
6134            1.703259                    1.643168                 -0.385083
6135            1.728623                   -0.634954                 -0.159619
6136           -0.172510                    1.147963                 -0.014899
6137            0.190623                    1.793301                  0.086964
6138            0.739252                    1.670555                  0.133547

[6111 rows x 9 columns]
```

[ ]:
```
# New work
Hi Folks,

1) In Coding file - Perform Predictive Modelling and submit the assignment in␣
 ↪ipynb and pdf format as "Report - Predictive Modelling" . And this needs to␣
 ↪be done by eod of starting of new task (Check document for timeline).Mention␣
 ↪Below details in the document:

Name
```

```
Batch
Project Name

After Feature Engineering perform predictive modelling techniques which␣
 ↪comprises of below pointers:

1. Perform different regression model on top of selected features.
2. Plot graph of loss and accuracy.
3. Try not to overfit/underfit model on top of dataset.
4. Do hyper-parameter tuning, if model is not working well on selected features.
5. Once done with the modelling, then create a table at the end which comprises␣
 ↪of model name, value of hyperparameter, accuracy/MSE/MAD/etc (on Train and␣
 ↪Test data).
6. Do the comparison of different modelsand write the conclusion, so that you␣
 ↪can go with the best model for deployment.

Note:

1. Provide access to munirainsideaiml@gmail.com
2. Deadline: Submission will be done by 11:59 PM (21th Oct, 2023)
3. Maintain Professional decorum
```

[52]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[53]:
```
#Important feature for medelling from all above analysis:
   # Plasticizer_(gm)
   # Formulation_Duration_(hrs)
   #Water_Mix_(ml)
  #Additive_Catalyst_(gm)
  #Ash_Component_(gm)

#Using SVR
X =␣
 ↪df_standardized[['Plasticizer_(gm)','Formulation_Duration_(hrs)','Water_Mix_(ml)','Additive
 ↪values
```

[54]:
```
X
```

[54]:
```
array([[-0.13339072,  1.52212782, -0.55191996, -0.1221213 , -1.24076099],
       [-0.98126877, -1.31144941, -1.17461744,  0.47876987,  0.99844761],
       [-0.49184323,  0.56359447,  1.71788151, -1.46502317, -0.00765658],
       …,
       [ 1.4865389 ,  1.14796309, -0.38405605, -1.31498924, -1.28531625],
       [ 0.03549556,  1.79330072, -0.79182951,  0.5954881 ,  0.90661199],
       [-0.60558297,  1.67055458, -1.18674634,  0.52727026,  0.74692157]])
```

[55]:
```
y
```

[55]:
```
0       1.433540
1       0.185721
2       1.307453
3       0.927331
4       1.196274
          …
6134   -0.385083
6135   -0.159619
6136   -0.014899
6137    0.086964
6138    0.133547
Name: Compression_Strength_MPa, Length: 6111, dtype: float64
```

[59]:
```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

[61]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,␣
↪random_state=50)
```

[62]:
```
X_train.shape
```

[62]:
```
(4277, 5)
```

[63]:
```
X_test.shape
```

[63]:
```
(1834, 5)
```

[64]:
```
svr_linear = SVR(kernel='linear', C=1.0, epsilon=0.2)
svr_poly = SVR(kernel='poly', degree=2, C=1.0, epsilon=0.2)
svr_rbf = SVR(kernel='rbf', C=1.0, epsilon=0.2)
```

[65]:
```
# Train SVR models
svr_linear.fit(X_train, y_train)
svr_poly.fit(X_train, y_train)
svr_rbf.fit(X_train, y_train)
```

[65]:
```
SVR(epsilon=0.2)
```

[66]:
```
# Make predictions
y_pred_linear = svr_linear.predict(X_test)
y_pred_poly = svr_poly.predict(X_test)
y_pred_rbf = svr_rbf.predict(X_test)
```

[67]:
```
# Model evaluation
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

mse_rbf = mean_squared_error(y_test, y_pred_rbf)
r2_rbf = r2_score(y_test, y_pred_rbf)

print("Linear Kernel - MSE: {:.2f}, R-squared (R2): {:.2f}".format(mse_linear,␣
 ↪r2_linear))
print("Polynomial Kernel - MSE: {:.2f}, R-squared (R2): {:.2f}".
 ↪format(mse_poly, r2_poly))
print("RBF Kernel - MSE: {:.2f}, R-squared (R2): {:.2f}".format(mse_rbf,␣
 ↪r2_rbf))
```

```
Linear Kernel - MSE: 0.95, R-squared (R2): 0.09
Polynomial Kernel - MSE: 0.95, R-squared (R2): 0.09
RBF Kernel - MSE: 0.79, R-squared (R2): 0.24
```

[ ]:
```
# Using Decision tree
```

[68]:
```
from sklearn.tree import DecisionTreeRegressor
```

[69]:
```
regressor = DecisionTreeRegressor(max_depth=5)
```

[70]:
```
# Train the model
regressor.fit(X_train, y_train)
```

[70]:
```
DecisionTreeRegressor(max_depth=5)
```

[71]:
```
# Make predictions
predictions = regressor.predict(X_test)
```

[72]:
```
# Evaluate the model using appropriate regression metrics
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 0.7234646447638587
R-squared: 0.3051218719402746
```

[86]:
```
# Random forest
```

[73]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

[75]:
```
# Create a Random Forest Regressor
regressor = RandomForestRegressor(n_estimators=50, random_state=42)

# Train the model on the training data
regressor.fit(X_train, y_train)
```

[75]:
```
RandomForestRegressor(n_estimators=50, random_state=42)
```

[76]:
```
predictions = regressor.predict(X_test)
```

[77]:
```
# Evaluate the model's performance using metrics such as Mean Squared Error␣
↪(MSE) and R-squared (R2)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Squared Error (MSE): 0.701902166032218
R-squared (R2): 0.32583234475441025
```

1 SVR

Linear Kernel - MSE: 0.95, R-squared (R2): 0.09 Polynomial Kernel - MSE: 0.95, R-squared (R2): 0.09 RBF Kernel - MSE: 0.79, R-squared (R2): 0.24 # Decision Tree Regressor Mean Squared Error: 0.7234646447638587 R-squared: 0.3051218719402746 # Random Forest Regressor Mean Squared Error (MSE): 0.701902166032218 R-squared (R2): 0.32583234475441025