

```

1  """
2  Version: 1.0
3  Created on: Sat Mar 20 18:55:27 2021
4  Author: Balaji Kannan
5  Description: Lower Back Pain Dataset from Kaggle
6      1. https://www.kaggle.com/sammyl23/lower-back-pain-symptoms-dataset
7      2. This pipeline is custom built for a single target variable with Datatype
         Object.
8      3. Binary classification problem.
9
10 """
11
12 %% Library
13
14 import pandas as pd
15 pd.set_option('display.max_columns', 20)
16 import numpy as np
17 import matplotlib.pyplot as plt
18 import seaborn as sns
19
20 %% IO Path & Dataframe Definitions
21
22 PATH = r"C:\DSML_Case_Studies\02_Logistic_Regression\Input"
23 FNAME = r"\Dataset_Lower_Back_Pain.csv" # Prefix LrBkPn
24
25 OUTPATH = r"C:\DSML_Case_Studies\02_Logistic_Regression\Output"
26 PREFIX = r"\LrBkPn_"
27
28 df = pd.read_csv(f"{PATH}{FNAME}")
29 df = df.round(decimals=3) # rounding the decimals
30
31 targetvar = 0
32 while targetvar <= 0:
33     targetvar = int(input("Enter # of Target Variables: ")) # user specifies # of
        targets
34
35 collst = []
36 for columns in df.columns:
37     collst.append(columns)
38
39 featlst = collst[0:len(collst)-targetvar]
40 targlst = collst[-targetvar:]
41
42 # Sanity Checks
43
44 print(df.head(), sep='\n')
45 print("List of Features:", featlst, sep='\n')
46 print("List of Targets:", targlst, sep='\n')
47
48 %% Exploratory Data Analysis
49
50 OFNAME1 = r"01_Descriptive_Stats.txt"
51
52 desc_stat = df.describe() # Univariate analyses
53 print(desc_stat)
54 FOUT1 = open(f"{OUTPATH}{PREFIX}{OFNAME1}", 'w+')
55 desc_stat.to_string(FOUT1)
56 FOUT1.close()
57
58 # Generating n*4 matrix of box plots
59
60 n_rows = len(collst)//4
61 fig, axes = plt.subplots(n_rows, 4, figsize = (15,15))
62 axes = axes.flatten()
63
64 FIG1 = r"_FIG01_Boxplot"
65 for i in range(0,len(df.columns)-targetvar):
66     sns.boxplot(x=targlst[0], y=df.iloc[:,i], data=df, orient='v', ax=axes[i])
67     plt.tight_layout()
68     plt.savefig(f"{OUTPATH}{PREFIX}{FIG1}")
69
70 # Linear Correlation Heatmap

```

```

71
72 cormethod = {0:'pearson', 1:'kendall', 2:'spearman'}
73 for i in range(0, 3, 1):
74     temp = 'linear_cor' + str(i)
75     temp = df.corr(method=cormethod[i])
76     ftemp = cormethod[i].title()
77     FIG2 = r"_FIG02_Corr_"
78     mask = np.zeros(temp.shape, dtype=bool)
79     mask[np.tril_indices(len(mask))] = False
80     plt.subplots(figsize=(20,15))
81     plt.title(f"{ftemp} Correlation")
82     sns.heatmap(temp, annot=True, vmin=-1, vmax=1, center=0,
83                 cmap='coolwarm', square=True, mask=mask)
84     plt.savefig(f"{OUTPATH}{PREFIX}{FIG2}{ftemp}")
85
86 # Non-Linear Correlation Predictive Power Score - Heatmap
87
88 import ppscore as pps
89
90 FIG3 = r"_FIG03_Predictive_Power_Score"
91
92 ppscorr = pps.matrix(df) # Predictive Power Score - PPS
93 matrix_df = pps.matrix(df)[['x', 'y', 'ppscore']].pivot(columns='x', index='y',
94                                                         values='ppscore')
95 plt.subplots(figsize=(20,15))
96 sns.heatmap(matrix_df, cmap="Greens", annot=True, linewidth=0,
97             annot_kws={"size":12}, fmt='.2g')
98 plt.savefig(f"{OUTPATH}{PREFIX}{FIG3}")
99
100 # Scatter Plot with Hue
101 FIG4 = r"_FIG04_Scatter_Plot"
102
103 grid1 = sns.PairGrid(df, hue=targlst[0])
104 grid1.map(plt.scatter)
105 grid1.map_diag(sns.kdeplot)
106 grid1.add_legend()
107 grid1.fig.suptitle("Scatter Plot", y=1.01)
108 grid1.savefig(f"{OUTPATH}{PREFIX}{FIG4}")
109
110 %% Pandas Profile Report
111
112 from pandas_profiling import ProfileReport
113
114 OFNAME2 = r"02_Descriptive_Stats.html"
115 report = ProfileReport(df) # Descriptive statistics report
116 report.to_file(f"{OUTPATH}{PREFIX}{OFNAME2}") # Rendering to HTML
117
118 %% High Dimensional Interactive Plot - HD Plot
119
120 import hiplot as hip
121
122 OFNAME3 = r"03_Parallel_Plot.html"
123 parplot = hip.Experiment.from_dataframe(df)
124 parplot.to_html(f"{OUTPATH}{PREFIX}{OFNAME3}")
125
126 %% Machine Learning Model - Pre-Processing
127
128 from sklearn import preprocessing
129 from sklearn.preprocessing import StandardScaler
130 from sklearn.model_selection import train_test_split
131 from sklearn.linear_model import LogisticRegression
132 from sklearn.metrics import classification_report
133
134 import statsmodels.api as sm
135
136 df.loc[df.Class_att=='Abnormal',targlst[0]] = 1
137 df.loc[df.Class_att=='Normal', targlst[0]] = 0
138 X = df.drop(columns=targlst)
139 y = df.filter(targlst, axis=1)
140
141

```

```

142 def data_split(X,y):
143     """
144     Parameters
145     -----
146     X : Feature Variables
147     y : Target Variables
148     Returns
149     -----
150     Split dataframe into train and test
151     """
152     X_train, X_test, y_train, y_test = train_test_split(X, y.values.ravel(),
153                                                         test_size=0.3,
154                                                         random_state=39)
155     scaler = StandardScaler(copy=True, with_mean=True, with_std=True)
156     scaler.fit(X_train)
157     train_scaled = scaler.transform(X_train) # Only feature variables are scaled
158     test_scaled = scaler.transform(X_test) # only features varaibels are scaled
159     y_train=y_train.astype('int')
160     y_test=y_test.astype('int')
161     return(train_scaled, test_scaled, y_train, y_test)
162
163 %% Logistic Regression Model
164
165 def logistic_regression(x,y):
166     """
167     Parameters
168     -----
169     x : Feature Variables
170     y : Target Variables
171
172     Returns
173     -----
174     Logistic Regression Fit
175     """
176     logreg = LogisticRegression().fit(x, y)
177     return(logreg)
178
179 X_train_scaled, X_test_scaled, y_train, y_test = data_split(X,y)
180 logreg_result = logistic_regression(X_train_scaled, y_train)
181
182 print("Training set score:
183       {:.3f}".format(logreg_result.score(X_train_scaled,y_train)))
184 print("Test set score: {:.3f}".format(logreg_result.score(X_test_scaled,y_test)))
185
186 %% Logit with Stats Model
187 """
188 1. Scikit LogisticRegression is good in predicting target variable on a test set.
189 2. It did not interpret anything about the individual features.
190 3. Which variable(s) influence the Target variable more?
191 4. Use logit from stats model to answer questions 2 & 3
192 """
193
194 logit_model = sm.Logit(y_train, X_train_scaled)
195 result = logit_model.fit()
196 print(result.summary2())
197
198 """
199 1. Output such as:
200     Maximum Likelihood optimization failed to converge -
201     indicates that there is multicolliniarity.
202     Removing those variable would improve the convergence.
203 """
204
205 %% Feature Reduction Tryouts
206
207 features_to_drop = ['pelvic_incidence', 'lumbar_lordosis_angle'] # From Correlation
208 Heatmap
209
210 new_lst = targlst + features_to_drop
211 X = df.drop(columns=new_lst)
212 y = df.filter(targlst, axis=1)

```

```

211 X_train_scaled, X_test_scaled, y_train, y_test = data_split(X,y)
212 logreg_result = logistic_regression(X_train_scaled, y_train)
213
214 print("Training set score:
215 {:.3f)".format(logreg_result.score(X_train_scaled,y_train)))
216 print("Test set score: {:.3f)".format(logreg_result.score(X_test_scaled,y_test)))
217
218 logit_model = sm.Logit(y_train, X_train_scaled)
219 result = logit_model.fit()
220 print(result.summary2())
221
222 %% Consider only Variables for which P-value is < 0.05
223
224 features_to_drop1 = ['sacral_slope', 'pelvic_slope', 'Direct_tilt',
225                     'thoracic_slope', 'cervical_tilt', 'sacrum_angle',
226                     'scoliosis_slope', 'pelvic_tilt'] #
227 new_lst1 = targlst + features_to_drop + features_to_drop1
228 X = df.drop(columns=new_lst1)
229 y = df.filter(targlst, axis=1)
230 print(X.head())
231
232 X_train_scaled, X_test_scaled, y_train, y_test = data_split(X,y)
233 logreg_result = logistic_regression(X_train_scaled, y_train)
234
235 print("Training set score:
236 {:.3f)".format(logreg_result.score(X_train_scaled,y_train)))
237 print("Test set score: {:.3f)".format(logreg_result.score(X_test_scaled,y_test)))
238
239 logit_model = sm.Logit(y_train, X_train_scaled)
240 result = logit_model.fit()
241 print(result.summary2())
242
243 %% Model Metrics
244
245 # assigning the model predicted values to y_pred
246 y_pred = logreg_result.predict(X_test_scaled)
247
248 # assigning the string Normal and Abnormal to the 0 and 1 values respectively. This
249 # is useful in plotting
250 # the confusion matrix
251 y_pred_string = y_pred.astype(str)
252 y_pred_string[np.where(y_pred_string == '0')] = 'Normal'
253 y_pred_string[np.where(y_pred_string == '1')] = 'Abnormal'
254
255 y_test_string = y_test.astype(str)
256 y_test_string[np.where(y_test_string == '0')] = 'Normal'
257 y_test_string[np.where(y_test_string == '1')] = 'Abnormal'
258
259 from sklearn.metrics import confusion_matrix
260 ax= plt.subplot()
261 labels = ['Abnormal','Normal']
262 cm = confusion_matrix(y_test_string, y_pred_string, labels)
263 sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
264
265 # labels, title and ticks
266 ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
267 ax.set_title('Confusion Matrix');
268 ax.xaxis.set_ticklabels(['Abnormal', 'Normal']);
269 ax.yaxis.set_ticklabels(['Abnormal', 'Normal']);
270 plt.show()
271
272 print(classification_report(y_test, y_pred, target_names=labels))

```