

```

1  #PCOS or polycystic ovary syndrome :
2  #PCOS is a hormonal imbalance that affects ovulation. This can cause irregular periods, excess androgen, and cysts in the ovaries. It's a common condi
3
4  # PCOS Dataset Source : https://www.kaggle.com/datasets/cm037divya/pcos-dataset
5
6  """Importing libraries"""
7  import pandas as pd
8  import matplotlib.pyplot as plt
9  import seaborn as sns
10 import numpy as np
11 import tensorflow as tf
12 from tensorflow import keras
13 from sklearn.model_selection import train_test_split
14 from sklearn.preprocessing import StandardScaler
15 from sklearn.metrics import confusion_matrix
16 from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score
17 from sklearn.svm import SVC
18 from sklearn.metrics import accuracy_score
19 from sklearn.naive_bayes import GaussianNB
20 from sklearn.tree import DecisionTreeClassifier
21 from sklearn.ensemble import RandomForestClassifier
22
23 """Reading the dataset using pandas"""
24
25 dataset = pd.read_csv("/content/PCOS/PCOS_extended_dataset.csv")
26
27 """Analysis of Dataset"""
28
29 dataset.head(10)
30
31 dataset.tail(10)
32
33 """Getting the dimension of dataset"""
34
35 rows = dataset.shape[0]
36 columns = dataset.shape[1]
37 print("There are", rows, "rows and", columns, "Columns in dataset")
38
39 """Getting metadata about data (Datatype of column and count of non-null values)"""
40
41 dataset.info()
42
43 """Getting summary statistics of dataset"""
44
45 dataset.describe()
46
47 """Printing the columns name of dataset"""
48
49 print(dataset.columns)
50
51 """Cleaning"""
52
53 dataset_1 = dataset.drop(['Sl. No', 'Patient File No.', 'Weight (Kg)', 'Height (Cm) ', 'Hip (inch)', 'Waist (inch)', 'Marraige Status (Yrs)'], axis=1)
54
55 dataset_1.isna().sum()
56
57 """Updating column datatypes from object to float"""
58
59 for col in dataset_1:
60     if dataset_1[col].dtypes == object:
61         dataset_1[col] = pd.to_numeric(dataset_1[col], errors = "coerce") # The errors="coerce" parameter instructs pandas to convert any values that c
62
63 """Select all columns except the first ('PCOS(Y/N)') as it represents the target variable"""
64
65 columns = dataset_1.columns[1:]
66
67 """Create a grid of boxplots to visually inspect the distribution of each feature (excluding target) and identify potential outliers."""
68
69 plt.figure(figsize = (35,35))
70 for i, col in enumerate(columns):
71     plt.subplot(9, 4, i+1)
72     sns.boxplot(x = dataset_1[col])
73 plt.show()
74
75 """Calculate pairwise correlation coefficients"""
76
77 correlation = dataset_1.corr()
78 correlation
79
80 """Generate a heatmap of the correlation matrix with annotations"""
81
82 plt.figure(figsize = (40,40))
83 sns.heatmap(correlation, annot=True) # it's a 37x37 correlation matrix,
84
85 """Removing the outliers from the dataset"""
86
87 outliers_columns = ['Pulse rate(bpm) ', 'FSH(mIU/mL)', 'LH(mIU/mL)', 'FSH/LH', 'TSH (mIU/L)', 'Vit D3 (ng/mL)', 'BP _Systolic (mmHg)', 'BP _Diastolic (mmHg)']
88 lower_range = [60,0,0.05,0.39,0.4,5,80,50,1]
89 upper_range = [100,4000,30,17,7,90,145,105,19]
90
91 outliers_df = pd.DataFrame({'Columns': outliers_columns, 'Lower Range': lower_range, 'Upper Range': upper_range})
92 print(outliers_df.to_string())
93
94 def get_outliers_index(data, lower, upper):
95     store=[]
96     for i in range(len(data)):
97         if (data[i]>upper or data[i]<lower):
98             store.append(i)
99     return store
100
101 outlier_index=[]
102 for i in range(len(outliers_columns)):
103     data=outliers_columns[i]
104     lower=lower_range[i]
105     upper=upper_range[i]
106     indexes=get_outliers_index(dataset_1[data], lower, upper)
107     for i in indexes:

```

```

107     for j in indices:
108         if j not in outlier_index:
109             outlier_index.append(j)
110
111 print("Total number of Outliers found : ", len(outlier_index))
112
113 Cleaned_Dataset = dataset_1.drop(outlier_index)
114
115 """Cleaned Dataset"""
116
117 rows = Cleaned_Dataset.shape[0]
118 columns = Cleaned_Dataset.shape[1]
119 print("There are", rows, "rows and", columns, "Columns in Cleaned Dataset")
120
121 plt.figure(figsize = (40,40))
122 sns.heatmap(Cleaned_Dataset.corr(), annot=True) # it's a 37x37 correlation matrix,
123
124 """Splitting Dataset"""
125
126 Dataset = Cleaned_Dataset.dropna()
127 X=Dataset.iloc[:,1:].values # Independent variables
128 Y=Dataset.iloc[:,0].values # Target variable
129
130 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
131
132 sc=StandardScaler()
133 X_train=sc.fit_transform(X_train)
134 X_test=sc.transform(X_test)
135
136 #Implementing ANN
137
138 ann_model = keras.Sequential([
139     keras.layers.Dense(16, activation='relu'),
140     keras.layers.Dense(1, activation='sigmoid')
141 ])
142 ann_model.build(input_shape=(1,36))
143 ann_model.summary()
144
145 ann_model.compile(optimizer="adam", loss="binary_crossentropy", metrics=['accuracy'])
146 ann_model.fit(X_train, Y_train, epochs=70, batch_size=32)
147 Model_Name=[]
148 Accuracy_Model=[]
149
150 loss, accuracy = ann_model.evaluate(X_test, Y_test)
151 print('Loss:', loss)
152 print('Accuracy:', accuracy)
153 predicted=ann_model.predict(X_test)
154 predicted = [1 if x > 0.5 else 0 for x in predicted.flatten()]
155 cm = confusion_matrix(Y_test,predicted)
156 sns.heatmap(cm,
157     annot=True,
158     fmt='g',
159     xticklabels=['PCOS +', 'PCOS -'],
160     yticklabels=['PCOS +', 'PCOS -'])
161 plt.ylabel('Prediction', fontsize=13)
162 plt.xlabel('Actual', fontsize=13)
163 plt.title('Confusion Matrix', fontsize=17)
164 plt.show()
165
166
167 Model_Name.append('ANN')
168 Accuracy_Model.append(accuracy)
169
170 """Implementing SVM"""
171
172 kernel_Name= ['linear', 'sigmoid', 'rbf', 'poly']
173 Differ_Kernel_Accuracy=[]
174
175 for Kernel in kernel_Name:
176     classifier=SVC(kernel=Kernel)
177     classifier.fit(X_train,Y_train)
178     predicted=classifier.predict(X_test)
179     Differ_Kernel_Accuracy.append(accuracy_score(Y_test,predicted))
180
181 SVM_Accuracy = pd.DataFrame({'Kernel Used': kernel_Name, 'Accuracy': Differ_Kernel_Accuracy})
182 print(SVM_Accuracy.to_string())
183
184 print("SVM gives Max accuracy of ", max(Differ_Kernel_Accuracy), " with kernel ", kernel_Name[np.array(Differ_Kernel_Accuracy).argmax()], "\n")
185 SVM_Model=SVC(kernel=kernel_Name[np.array(Differ_Kernel_Accuracy).argmax()])
186 SVM_Model.fit(X_train, Y_train)
187 predicted=SVM_Model.predict(X_test)
188 cm = confusion_matrix(Y_test, predicted)
189 sns.heatmap(cm,
190     annot=True,
191     fmt='g',
192     xticklabels=['PCOS +', 'PCOS -'],
193     yticklabels=['PCOS +', 'PCOS -'])
194 plt.ylabel('Prediction', fontsize=13)
195 plt.xlabel('Actual', fontsize=13)
196 plt.title('Confusion Matrix', fontsize=17)
197 plt.show()
198
199
200 Model_Name.append('SVM')
201 Accuracy_Model.append(max(Differ_Kernel_Accuracy))
202
203 """Implementing Naive Bayes Classifier"""
204
205 gnb_model = GaussianNB()
206 gnb_model.fit(X_train, Y_train)
207
208 y_pred = gnb_model.predict(X_test)
209
210 cm = confusion_matrix(Y_test, y_pred)
211
212 sns.heatmap(cm,
213     annot = True,
214     fmt='g',

```

```

214     plt.ylabel('Prediction', fontsize=13)
215     xticklabels=['PCOS +','PCOS -'],
216     yticklabels=['PCOS +','PCOS -'])
217 plt.ylabel('Prediction', fontsize=13)
218 plt.xlabel('Actual', fontsize=13)
219 plt.title('Confusion Matrix', fontsize=17)
220 plt.show()
221
222 accuracy = accuracy_score(y_pred, Y_test)
223 print('Accuracy : ', accuracy)
224
225 Model_Name.append('Naive Bayes')
226 Accuracy_Model.append(accuracy)
227
228 """Implementing the Decision Tree Model"""
229
230 DT_Model = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
231 DT_Model.fit(X_train, Y_train)
232
233 y_pred = DT_Model.predict(X_test)
234
235 cm = confusion_matrix(Y_test, y_pred)
236
237 sns.heatmap(cm,
238             annot = True,
239             fmt='g',
240             xticklabels=['PCOS +','PCOS -'],
241             yticklabels=['PCOS +','PCOS -'])
242 plt.ylabel('Prediction', fontsize=13)
243 plt.xlabel('Actual', fontsize=13)
244 plt.title('Confusion Matrix', fontsize=17)
245 plt.show()
246
247 accuracy = accuracy_score(y_pred, Y_test)
248 print('Accuracy : ', accuracy)
249
250 Model_Name.append('Decision Tree')
251 Accuracy_Model.append(accuracy)
252
253 """Implementing Random Forest Model"""
254
255 RF_model = RandomForestClassifier(n_estimators = 10, criterion = "entropy")
256 RF_model.fit(X_train, Y_train)
257
258 y_pred = RF_model.predict(X_test)
259
260 cm = confusion_matrix(Y_test, y_pred)
261
262 sns.heatmap(cm,
263             annot = True,
264             fmt='g',
265             xticklabels=['PCOS +','PCOS -'],
266             yticklabels=['PCOS +','PCOS -'])
267 plt.ylabel('Prediction', fontsize=13)
268
269 plt.xlabel('Actual', fontsize=13)
270 plt.title('Confusion Matrix', fontsize=17)
271 plt.show()
272
273 accuracy = accuracy_score(y_pred, Y_test)
274 print('Accuracy : ', accuracy)
275
276 Model_Name.append('Random Forest')
277 Accuracy_Model.append(accuracy)
278
279 Models = pd.DataFrame({'Model Name': Model_Name, 'Accuracy': Accuracy_Model})
280 print(Models.to_string())

```