Lab Report

Data Mining and Machine Learning Lab

Course Code: CSE322

Submitted By

Name: SHAKIL RANA

ID: 202-15-3816

Section: PC-A

Department: Computer Science and Engineering

Submitted To

Dr. Md Zahid Hasan

Associate Professor & Program Director MIS

Department of Computer Science and Engineering

Faculty of Science and Information Technology

Date of Submission: 12-6-2022

Polycystic ovary syndrome

Title: Polycystic ovary syndrome Classification with machine learning.

Problem Statement: Polycystic ovary syndrome (PCOS) is a hormonal disorder common among women of reproductive age. Women with PCOS may have infrequent or prolonged menstrual periods or excess male hormone (androgen) levels. The ovaries may develop numerous small collections of fluid (follicles) and fail to release eggs regularly.

The exact cause of PCOS is unknown. Early diagnosis and treatment and weight loss may reduce the risk of long-term complications such as type 2 diabetes and heart disease. [Mayo Clinic]

This dataset is composed of 44 different features with more than 500 records. Such features include pimples, hair growth, cycles, vitamin d3, etc.

Pre-processing technique:

Here I am using some basic preprocessing technique such as,

1.Basic Data Cleaning

2.Handling missing data

3.Data Exploration

4.Outlier detection

5.Interactions between features

6.Dimensionality reduction using PCA

First Load and Check Data, If you see the Missing optional dependency 'xlrd' error.You just need to install a required package before trying to use pd.read_excel.

```
data =pd.read_excel("/content/drive/MyDrive/Colab Notebooks/Dataset with disease data.xlsx")
data
```

```
!pip install openpyxl
```

Then Check null values within the dataset

```
data.isnull().sum().sort_values(ascending=False).head(10)
```

```
Unnamed: 44                539
FSH/LH                     532
Waist:Hip Ratio            532
BMI                        299
Marraige Status (Yrs)        1
Fast food (Y/N)              1
Skin darkening (Y/N)         0
PRL(ng/mL)                   0
Vit D3 (ng/mL)               0
Endometrium (mm)             0
dtype: int64
```

Now Replace the null or not available values with the forward fillna and mean function of these features and also drop some useless column.

```
data=data.drop(['Unnamed: 44'],axis=1)
data=data.drop(['FSH/LH'],axis=1)
data=data.drop(['Waist:Hip Ratio'],axis=1)
```

```
data['BMI'].fillna(data['BMI'].mode(), inplace=True)
```

```
data=data.fillna(method='ffill')
```

Then explore the variable of dtype object

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Sl. No                    541 non-null    int64
 1   Patient File No.          541 non-null    int64
 2   PCOS (Y/N)                541 non-null    int64
 3    Age (yrs)                541 non-null    int64
 4   Weight (Kg)               541 non-null    float64
 5   Height(Cm)                541 non-null    float64
 6   BMI                       242 non-null    float64
 7   Blood Group               541 non-null    int64
 8   Pulse rate(bpm)           541 non-null    int64
 9   RR (breaths/min)          541 non-null    int64
 10  Hb(g/dl)                  541 non-null    float64
 11  Cycle(R/I)                541 non-null    int64
 12  Cycle length(days)        541 non-null    int64
 13  Marraige Status (Yrs)     540 non-null    float64
 14  Pregnant(Y/N)             541 non-null    int64
 15  No. of aborptions         541 non-null    int64
 16    I   beta-HCG(mIU/mL)    541 non-null    float64
 17  II    beta-HCG(mIU/mL)    541 non-null    object
 18  FSH(mIU/mL)               541 non-null    float64
 19  LH(mIU/mL)                541 non-null    float64
 20  Hip(inch)                 541 non-null    int64
 21  Waist(inch)               541 non-null    int64
 22  TSH (mIU/L)               541 non-null    float64
 23  AMH(ng/mL)                541 non-null    object
 24  PRL(ng/mL)                541 non-null    float64
 25  Vit D3 (ng/mL)            541 non-null    float64
 26  PRG(ng/mL)                541 non-null    float64
 27  RBS(mg/dl)                541 non-null    float64
 28  Weight gain(Y/N)          541 non-null    int64
 29  hair growth(Y/N)          541 non-null    int64
 30  Skin darkening (Y/N)      541 non-null    int64
 31  Hair loss(Y/N)            541 non-null    int64
```

✓ 0s completed at 12:09 AM

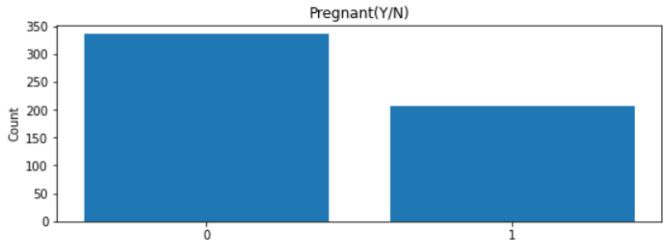
```
[ ]  29  hair growth(Y/N)          541 non-null    int64
     30  Skin darkening (Y/N)      541 non-null    int64
     31  Hair loss(Y/N)            541 non-null    int64
     32  Pimples(Y/N)              541 non-null    int64
     33  Fast food (Y/N)           540 non-null    float64
     34  Reg.Exercise(Y/N)         541 non-null    int64
     35  BP _Systolic (mmHg)       541 non-null    int64
     36  BP _Diastolic (mmHg)      541 non-null    int64
     37  Follicle No. (L)          541 non-null    int64
     38  Follicle No. (R)          541 non-null    int64
     39  Avg. F size (L) (mm)      541 non-null    float64
     40  Avg. F size (R) (mm)      541 non-null    float64
     41  Endometrium (mm)          541 non-null    float64
    dtypes: float64(17), int64(23), object(2)
    memory usage: 177.6+ KB
```

Now convert from object dtype to float

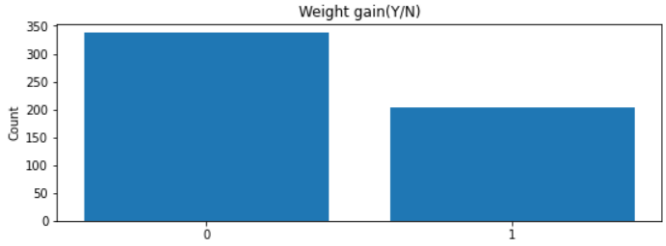
```
[ ]  data["AMH(ng/mL)"] = pd.to_numeric(data["AMH(ng/mL)"], errors='coerce')
     data["II    beta-HCG(mIU/mL)"] = pd.to_numeric(data["II    beta-HCG(mIU/mL)"], errors='coerce')
```

Now Visualization some particular column compare with pcos

```
category = ["Pregnant(Y/N)", "Weight gain(Y/N)", "hair growth(Y/N)", "Skin darkening (Y/N)", "Hair loss(Y/N)",
            "Pimples(Y/N)", "Fast food (Y/N)", "Reg.Exercise(Y/N)", "Blood Group"]
for c in category:
    bar_plot(c)
```




```
Pregnant(Y/N):
 0    335
1    206
Name: Pregnant(Y/N), dtype: int64
```



✓ 0s completed at 12:09 AM

Find out the cor relationship

```
corrmat = data.corr()

corrmat["PCOS (Y/N)"].sort_values(ascending=False) #How all the features correlate with
```

```
PCOS (Y/N)                1.000000
Follicle No. (R)          0.648327
Follicle No. (L)          0.603346
Skin darkening (Y/N)      0.475733
hair growth(Y/N)          0.464667
Weight gain(Y/N)          0.441047
Cycle(R/I)                0.401644
Fast food (Y/N)           0.378720
Pimples(Y/N)              0.286077
AMH(ng/mL)                0.264716
Weight (Kg)               0.211938
Hair loss(Y/N)            0.172879
Waist(inch)               0.164598
Hip(inch)                 0.162297
BMI                       0.151999
Avg. F size (L) (mm)      0.132992
Endometrium (mm)          0.106648
Avg. F size (R) (mm)      0.097690
Pulse rate(bpm)           0.091821
Hb(g/dl)                  0.087170
Vit D3 (ng/mL)            0.085494
Height(Cm)                0.068254
Reg.Exercise(Y/N)         0.065337
LH(mIU/mL)                0.063879
Sl. No                    0.060998
Patient File No.          0.060998
RBS(mg/dl)                0.048922
```

ML-Algorithm:

Let's now begin to train our Classification model. We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case, the class column.

Model Building

```
y= data['PCOS (Y/N)']
X= data.drop('PCOS (Y/N)', axis=1)
```

+ Code + Text

Now let's split the data into a training set and a testing set. We will train the model on the training set and then use the test set to evaluate the model.

1.logisticRegreesion

```
[ ] from sklearn.model_selection import train_test_split
    xtrain,xtest,ytrain,ytest= train_test_split(X,y,test_size=.3,random_state=1)
```

```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression(max_iter=150)
logmodel.fit(xtrain,ytrain)
```

```
LogisticRegression(max_iter=150)
```

2. Random forest

```
from sklearn.ensemble import RandomForestClassifier
tr= RandomForestClassifier(n_estimators= 80, criterion="entropy")
tr.fit(xtrain, ytrain)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=80)
```

```
y_pred= tr.predict(xtest)
```

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(ytest, y_pred)
```

```
cm
```

```
array([[110,   3],
       [ 10,  40]])
```

3. Naive_Bayes

```
[ ]  from sklearn.naive_bayes import GaussianNB
     gnb = GaussianNB()
     gnb.fit(xtrain, ytrain)
```

```
     GaussianNB()
```

```
[ ]  pred = gnb.predict(xtest)
```

```
[ ]  from sklearn.metrics import classification_report,confusion_matrix
     print(confusion_matrix(ytest,pred))
     print(classification_report(ytest,pred))
```

```
     [[107   6]
      [ 22  28]]
```

And also show some combine algorithm technique

```
# Models to be used for ML
models = [('Logistic Regression', LogisticRegression()),
          ('Decision Tree Classifier', DecisionTreeClassifier()),
          ('Random Forest', RandomForestClassifier()),
          ('Linear Discriminant Analyzer', LinearDiscriminantAnalysis()),
          ('Ada Boost', AdaBoostClassifier()),
          ('KNN', KNeighborsClassifier()),
          ]

models_score = []
for name, model in models:
    model = model #Model Object create
    model.fit(xtrain,ytrain)
    model.predict(xtest)
    models_score.append([name, accuracy_score(ytest, model.predict(xtest))])

    print("Model: ",name)
    print('Validation Accuracy: ', accuracy_score(ytest, model.predict(xtest)))
    print('Training Accuracy: ', accuracy_score(ytrain, model.predict(xtrain)))

    plt.figure()
    cf_matrix = confusion_matrix(ytest, model.predict(xtest))
    plt.title('Confusion Matrix: {}'.format(name))
    sns.heatmap(cf_matrix, annot = True, fmt = 'g', cmap = sns.cubehelix_palette(as_cmap=True))
    plt.show()
```

Result:

The accuracy of the model is 93% for Random Forest .From the confusion matrix, we saw that our train and test data are balanced. Most of the classification methods hit accuracy with this dataset. Others algorithms can't give that much accuracy.

Random Forest Classification	Test Accuracy: 93%
Decision Tree Classification	Test Accuracy: 84%

KNN Classification	Test Accuracy: 71%
Naive Bayes Classification	Test Accuracy: 84%
Logistic Regression Classification	Test Accuracy: 84%