ECE 5984 SP22 – Prof. Jones – HW2

1. Print to the Python console a more complete set of statistics on the data. You may use numpy operations on a numpy array, or operations on a Pandas DataFrame (or Series) for this functionality. The output that I want is as follows (not all columns are shown):

**Python Console Output**

A picture containing text, outdoor

Description automatically generated

1. Write a similar report to an Excel workbook. I used operations on a pandas DataFrame for this functionality. Note: “cardinality” is the number of distinct values. The output that I want in the spreadsheet is as follows (note, yours will have numbers 😊😊):

**Report Excel File:**



1. From this DQR, determine a few things about each column in the data set. For every column in the data, tell me each of the following:
   1. The type of the feature (ID, target, or feature – and what type of feature – continuous, binary, interval, categorical, etc.)

Member = ID

Sex = Binary

Age = Numeric

CP = Categorical

Trestbps = Numeric

Chol = Numeric / Categorical

FBS = Binary

Restecg = Categorical

Thalach = Numeric

Exang = Binary

Oldpeak = Numeric

Slope = Categorical

Ca = Categorical

Thal = Categorical

Bt = Categorical

Target = Binary

* 1. How many values are missing and how many are invalid?

Total Missing Values: 65

Total Invalid Stats: 22 (not including stats that don’t make sense)

Member = 2

Age = 0

Sex = 1

CP = 1

Trestbps = 1

Chol = 3

FBS = 7

Restecg = 1

Thalach = 35

Exang = 1

Oldpeak = 1

Slope = 2

Ca = 5

Thal = 0

Bt = 5

Target = 0

* 1. What should be done about the missing values – BE SPECIFIC (don’t just say “replace missing values” but tell me how)

For numeric features such as Age, Blood Pressure, or Cholesterol, it would be best to replace those missing values with the average. For categorical and binary features, it would be best to replace those missing values with the mode. And for ID features such as Member, it would be best to not replace those values because there is no meaningful statistic that would replace an ID.

* 1. Whether the feature contains a significant number of outliers

Member: None

Age: None

Sex: None

CP: None

Trestbps: None

Chol: None:

Fbs: None

Restecg: None

Thalach: None

Exang: None

Oldpeak: None

Slope: None

CA: Has cardinality of 5 however it should be cardinality of 4 based on the website.

Thal: Has cardinality of 4 however it should be cardinality of 3 based on the website.

Bt: None

Target: None

* 1. Whether the feature should be ignored in modeling (by removing the column).

Member: Yes

Age: No

Sex: No

CP: No

Trestbps: No

Chol: No

Fbs: No

Restecg: No

Thalach: No

Exang: No

Oldpeak: No

Slope: No

CA: No

Thal: No

Bt: No

Target: No

1. Calculate and write to Excel workbooks the covariance and correlation matrices for the numeric values in this data set. Create data frames with a row and a column for each numeric value; the entries in the cells are the covariances and the correlations for each pair of numeric features. Include the target value if it’s numeric.

Covariance Output:



Correlation Output:



1. From these workbooks, determine the three feature values (predictors) that are most highly correlated with the target; list them and their correlation. Note that either a large positive or a large negative correlation with the target indicates a good predictor. Also, find the three predictors that are the most highly correlated with each other; list them and their cross-correlation.

**Three features highly correlated to target:** CP, Exang, and Oldpeak

**Three predictors highly correlated:**

Slope and Oldpeak: -0.579058

Slope and Thalach: 0.385144

Oldpeak and Thalach: -0.34325

Python Code:

**simple\_stats.py**

import pandas

import stats\_report as sr

filename = r"C:\Users\agarc\OneDrive\Documents\GitHub\Virginia\_Tech\_Masters\ECE\_5984\_Appl\_Machine\_Learning\_SP22\Homework\_2\Heart Disease.xlsx"

df = pandas.read\_excel(filename) # read Excel spreadsheet

print('File {0} is of size {1}'.format(filename, df.shape))

labels = df.columns

report = sr.StatsReport()

# Create a simple data set summary for the console

for thisLabel in labels: # for each column, report stats

thisCol = df[thisLabel]

report.addCol(thisLabel, thisCol)

print(report.to\_string())

report.statsdf.to\_excel("Report\_Andrew\_Garcia.xlsx")

covariance = df.cov()

correlation = df.corr()

covariance.to\_excel("Covariance\_Andrew\_Garcia.xlsx")

correlation.to\_excel("Correlation\_Andrew\_Garcia.xlsx")

labels = correlation.columns

for thisLabel in labels:

if thisLabel == "member":

pass

else:

thisCol = correlation[thisLabel]

v = thisCol.sort\_values()

max\_v = v[-2]

min\_v = v[0]

min\_v\_abs = abs(min\_v)

if max\_v > min\_v\_abs:

max\_overall = max\_v

print(f"\n Label: {thisLabel} - Highest: {max\_v}")

elif min\_v\_abs > max\_v:

max\_overall = min\_v\_abs

print(f"\n Label: {thisLabel} - Highest: {min\_v}")

**stats\_report.py**

import pandas

class StatsReport:

def \_\_init\_\_(self):

self.statsdf = pandas.DataFrame()

self.statsdf['stat'] = ['cardinality', 'mean', 'median', 'n\_at\_median', 'mode', 'n\_at\_mode', 'stddev', 'min', 'max', 'nzero', 'nmissing']

pass

def addCol(self, label, data):

self.statsdf[label] = [self.cardinality\_(data), self.mean\_(data), self.median\_(data), self.n\_at\_median(data), self.mode\_(data), self.n\_at\_mode(data), self.std\_(data), self.min\_(data), self.max\_(data), self.nzero\_(data), self.nmissing\_(data)]

def to\_string(self):

return self.statsdf.to\_string()

def cardinality\_(self, d):

try:

return d.nunique()

except:

return "N/A"

def mean\_(self, d):

try:

return d.mean()

except:

return "N/A"

def median\_(self, d):

try:

return d.median()

except:

return "N/A"

def n\_at\_median(self, d):

try:

n = d == d.median()

return n.sum()

except:

return "N/A"

def mode\_(self, d):

try:

return int(d.mode())

except:

return "N/A"

def n\_at\_mode(self, d):

try:

n = d == int(d.mode())

return n.sum()

except:

return "N/A"

def std\_(self, d):

try:

return d.std()

except:

return "N/A"

def min\_(self, d):

try:

return d.min()

except:

return "N/A"

def max\_(self, d):

try:

return d.max()

except:

return "N/A"

def nzero\_(self, d):

try:

n = d == 0

return n.sum()

except:

return "N/A"

def nmissing\_(self, d):

try:

n = d.isna()

return n.sum()

except:

return "N/A"