#!/usr/bin/env python

# coding: utf-8

# # ECE 2556 Homework 1 - Avery Peiffer

# In[1]:

import pandas as pd

import random

import statsmodels.api as sm

from statistics import mean

from statsmodels.formula.api import glm

from scipy import stats

# ## Question 1: For data in `scores.txt` (120x2), design a permutation algorithm (permutate 1000 times), and identify whether column 1 > column 2 (or column 1 < column 2) and report the p-value.

# In[2]:

# Set up the hyperparameters for the experiment, read in the data, and posit the null and alternative hypotheses.

n = 1000

alpha = 0.05

top\_x = n \* alpha

perms = []

data = pd.read\_csv('scores.txt', sep='\t', header=None, names=['col1', 'col2'])

samples = len(data)

orig\_diff = data['col1'].mean() - data['col2'].mean()

print(f'Original difference = {orig\_diff}.')

if orig\_diff < 0:

print(f'Since the original difference < 0: H\_0 = mean(col2) NOT > mean(col1) H\_a = mean(col2) > mean(col1).')

else:

print(f'Since the original difference > 0: H\_0 = mean(col1) NOT > mean(col2) H\_a = mean(col1) > mean(col2).')

# In[3]:

# Perform the permutations for the experiment.

for i in range(0, n):

perm = data['col1'].tolist() + data['col2'].tolist()

random.shuffle(perm)

size = int(len(perm) / 2)

new\_col\_1 = perm[:size]

new\_col\_2 = perm[size:]

mean1 = mean(new\_col\_1)

mean2 = mean(new\_col\_2)

if orig\_diff < 0:

perms.append(mean2 - mean1)

else:

perms.append(mean1 - mean2)

# In[4]:

# Tabulate and interpret the results of the experiment.

count = 0

for i in range(0, len(perms)):

if perms[i] > abs(orig\_diff):

count += 1

p\_value = count / len(perms)

print(f'The p-value is {p\_value}.')

if p\_value > alpha:

print(f'The p-value is greater than alpha; the experiment fails to reject the null hypothesis.')

if orig\_diff < 0:

print(f'Result: mean(col2) is NOT > mean(col1).')

else:

print(f'Result: mean(col1) is NOT > mean(col2).')

else:

print(f'The p-value is less than alpha; the experiment rejects the null hypothesis.')

if orig\_diff < 0:

print(f'Result: mean(col2) > mean(col1) at this significance.')

else:

print(f'Result: mean(col1) > mean(col2) at this significance.')

# ## Question 2: For data in `hospital.txt`:

# - Divide subjects based on smoker (0: non-smoker, 1: smoker). Then, check whether weight has a significant difference between groups.

# - Design a regression model to explore the relationship between blood pressure and smoke. Report B, P, AIC, and BIC.

# In[5]:

# Use student's t-test to determine the difference between the means of the groups.

# Null hypothesis: Weight has no significant difference between the groups

df = pd.read\_csv('hospital.txt', sep='\t')

df\_smoker = df['Weight'][df['Smoker'] == 1]

df\_non\_smoker = df['Weight'][df['Smoker'] == 0]

ttest = stats.ttest\_ind(df\_smoker, df\_non\_smoker)

# The t value with alpha = 0.05 and degrees of freedom = 98 is about 1.984

alpha = 0.05

degrees\_freedom = len(df\_smoker) + len(df\_non\_smoker) - 2

t\_const = stats.t.ppf(1 - (alpha / 2), degrees\_freedom)

print(f'T-test statistic = {ttest.statistic} and p-value = {ttest.pvalue}.')

print(f'Constant from t-value lookup table = {t\_const}.')

if ttest.statistic > t\_const:

print('Able to reject null hypothesis: there is a significant difference between the means of the two groups.')

else:

print('Fail to reject null hyopthesis: there is no significant difference between the means of the two groups.')

# In[6]:

# Use two GLMs to examine the effect that smoking has on blood pressure.

X = df['Smoker']

y = df['BloodPressure\_high']

z = df['BloodPressure\_Low']

model = sm.GLM(y, X, family=sm.families.Gaussian())

results = model.fit()

model2 = sm.GLM(z, X, family=sm.families.Gaussian())

results2 = model2.fit()

print(f'These models examine the effect that smoking has on blood pressure - both high and low.')

print(f'Note: in the summary, the beta values are listed under the coefficient column and the p-values are under the P>|z| column.\n')

print(results.summary())

print(f'AIC: {results.aic}.')

print(f'BIC: {results.bic}.')

print('\n\n')

print(results2.summary())

print(f'AIC: {results2.aic}.')

print(f'BIC: {results2.bic}.')

# In[7]:

# Use another GLM to examine the opposite: the likelihood that an individual is a smoker based on their blood pressure.

X = df[['BloodPressure\_high', 'BloodPressure\_Low']]

y = df['Smoker']

model = sm.GLM(y, X, family=sm.families.Gaussian())

results = model.fit()

print(f'This model measures the likelihood that an individual is a smoker based on their blood pressure.')

print(f'Note: in the summary, the beta values are listed under the coefficient column and the p-values are under the P>|z| column.\n')

print(results.summary())

print(f'AIC: {results.aic}.')

print(f'BIC: {results.bic}.')