In [1]:

```
##Import necessary modules

import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
import tkinter as tk
import math
from matplotlib import pyplot as plt
```

In [2]:

```
##Import Data and build numpy arrays to manipulate
dataset = pd.read_csv('PressureSensorData.csv')
currentData = dataset
transPressure = np.array(currentData['pt_psia'])
anaPressure = np.array(currentData['pa_psia'])
voltage = np.array(currentData['pd_volts'])

#Count how many entries are made
entryCount = 0
```

In [3]:

```
##Create function to add entries to our dataset

def addDataEntry(event1,event2,event3,existingData):
    colNames = existingData.columns.tolist()

    try:
        #Format entries into dataframe and merge it with existing data
        newEntry = pd.DataFrame([["Custom Entry",float(truePEntry.get()),float(anaPEntry.get()),float(voltEntry.get()
        global currentData
        currentData = existingData.append(newEntry)

        #Clear entry fields after adding data entry
        truePEntry.delete(0, 'end')
        anaPEntry.delete(0, 'end')
        voltEntry.delete(0, 'end')

        #Let the user know their entry went through
        manipLabel.configure(text="Your data entry has been added")
```

```
        global entryCount
        entryCount += 1
    except:
        manipLabel.configure(text="Please only input numbers and make sure every field has an entry")
```

In [4]:

```
##Create function to remove the last data entry


######### Having alot of issues here, errors if I remove the -1 when I try to drop the last entry show the row count
######### but when it doesn't error out it drops the last entry of the original dataset along with every added entry.
######### Will come back to this.



#def remEntry(modified):
#    global entryCount
#    if entryCount == 0:
#        manipLabel.configure(text="There are no custom data entries" + str(entryCount))
#    else:
#        global currentData
#        currentData = modified.drop(modified.index[len(modified)]-1)
#        manipLabel.configure(text="Last data entry has been dropped")
#        entryCount -= 1
```

In [5]:

```
##Create function to remove all custom data entries

def clearEntries():
    global currentData
    global dataset
    global entryCount

    #Set our working dataset to be equal to our original dataset and clear our entryCount
    currentData = dataset
    entryCount = 0
    manipLabel.configure(text="All Custom Data Entries Have Been Cleared")
```

In [16]:

```
##Create function to get estimated Analogue Pressure for input voltage
```

```
def getAnaP(event):
    try:
        volts = float(userVoltIn.get())
        model = smf.ols('pa_psia ~ pd_volts', data=currentData)
        model = model.fit()
        result = model.params[0] + volts * model.params[1]
        anaRes.configure(text = "Your Estimated Analogue Readout is: " + str(result) + " (PSIA)")
    except:
        anaRes.configure(text = "That's not a number")
```

In [14]:

```
##Create function to get estimated True Pressure for input voltage

def getTrueP(event):
    try:
        volts = float(userVoltIn.get())
        model = smf.ols('pt_psia ~ pd_volts', data=currentData)
        model = model.fit()
        result = model.params[0] + volts * model.params[1]
        trueRes.configure(text = "Your Estimated True Pressure is: " + str(result) + " (PSIA)")

        #Establish variables for calculating confidence interval
        sampSize = len(transPressure)                       #Sample size
        sampStd = np.std(transPressure)                     #Standard deviation (σ)

        #Find Margin of error
        margin = zValue*(sampStd/math.sqrt(sampSize))

        #Calculate our interval
        lowEnd = result-margin
        highEnd = result+margin

        confRes.configure(text = "With a Confidence Interval of (" + str(lowEnd) + "," + str(highEnd) + ")" + " (PSIA
    except:
        trueRes.configure(text = " ")
        confRes.configure(text = " ")
```

In [8]:

```
##Create function to find confidence interval
```

```
def findConf():

    #Select a z-value based on the selected radio-button
    confChoice = int(var.get())
    if(confChoice == 1):
        z = 1.645                       #The z value associated with 90% Confidence Interval
    elif(confChoice == 2):
        z = 1.960                       #The z value associated with 95% Confidence Interval
    elif(confChoice == 3):
        z = 2.576                       #The z value associated with 90% Confidence Interval
    global zValue
    zValue = z
    result = z
```

In [26]:

```
#### User Interface


#Initialize GUI
gui = tk.Tk()
gui.geometry("1000x500")

#Ask for user input
inputLabel = tk.Label(gui, text="Input Voltage:").pack()
userVoltIn = tk.Entry(gui)
userVoltIn.pack()

#Create radio buttons to select confidence interval
confidenceInput = tk.Label(gui,text="Select your desired Confidence Level")
confidenceInput.pack()
var = tk.StringVar(gui, "1")
values = {"90%" : "1","95%" : "2","99%" : "3"}
for (text, value) in values.items():
    tk.Radiobutton(gui, text = text, variable = var, value = value,command=findConf).pack()

#Bind return key to our functions
userVoltIn.bind("<Return>", getAnaP)
userVoltIn.bind("<Return>", getTrueP, add="+")

#Create a button to run our functions
bigOlButton = tk.Button(gui, text="Calculate",command=(lambda e=userVoltIn:[getAnaP(e),getTrueP(e)])).pack()
```

```
#Display our results
anaRes = tk.Label(gui)
trueRes = tk.Label(gui)
confRes = tk.Label(gui)
anaRes.pack()
trueRes.pack()
confRes.pack()




##Interface to add data entries
addDataEntriesLabel = tk.Label(gui,text="Add your own data entries below").pack()
truePInputLabel = tk.Label(gui, text="New True Pressure Entry:").pack()
truePEntry = tk.Entry(gui)
truePEntry.pack()
anaPInputLabel = tk.Label(gui, text="New Analogue Pressure Entry:").pack()
anaPEntry = tk.Entry(gui)
anaPEntry.pack()
voltInputLabel = tk.Label(gui, text="New Voltage Entry:").pack()
voltEntry = tk.Entry(gui)
voltEntry.pack()

#Button to add the data and clear entry fields
dataEntryButton = tk.Button(gui,text="Add Data Entry",command=(lambda a=truePEntry,b=anaPEntry,c=voltEntry:[addDataEr
manipLabel = tk.Label(gui)

########## See above, this thing is giving me problems
#Button to remove the last data entry
#remEntryButton = tk.Button(gui,text="Remove Last Data Entry",command=(lambda a=currentData:remEntry(a))).pack()

#Button to remove all additional data entries
clearEntriesButton = tk.Button(gui,text="Clear Data Entries",command=clearEntries).pack()


manipLabel.pack()




gui.mainloop()
```

In [ ]: