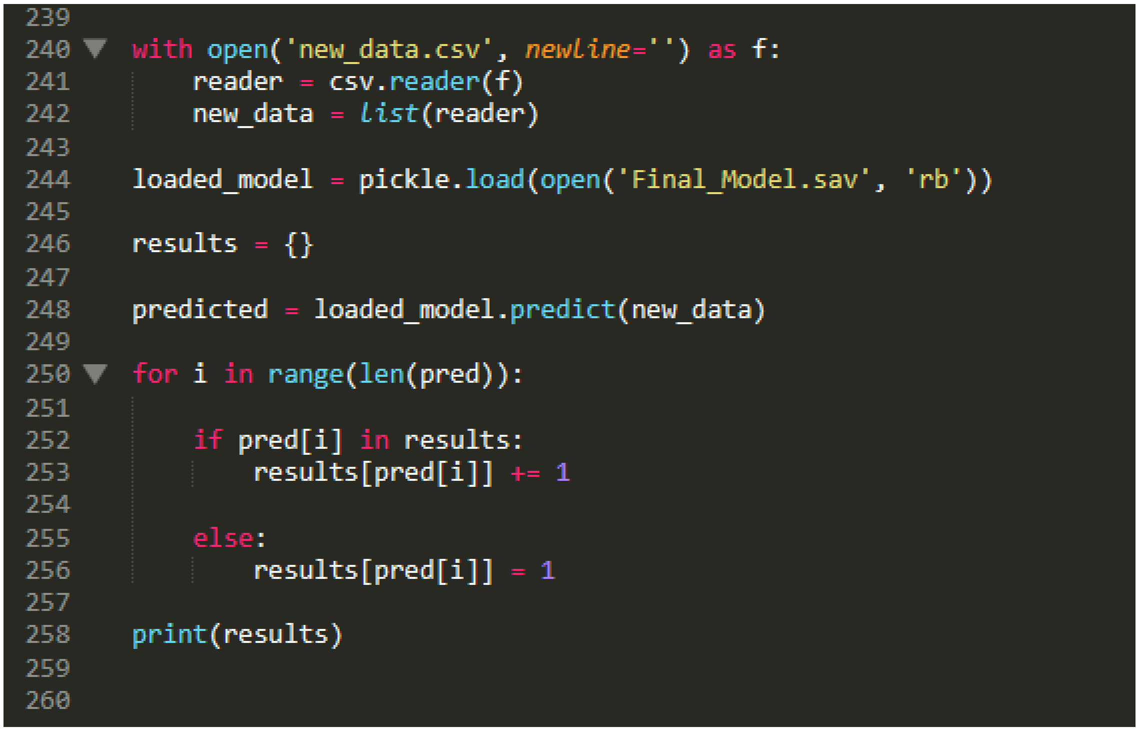
Overview and Instructions for Running the Fault Detection Program

1. Overview of Python Scripts

In our deliverables we included the python scripts for cleaning the data, binary model, the building of each Uncertainty Class, and the final model. The only script you need to run the model is the final model script. The others are just to show how the data was cleaned and how we completed each step described in the explanation above.

2. Process to Test New Data

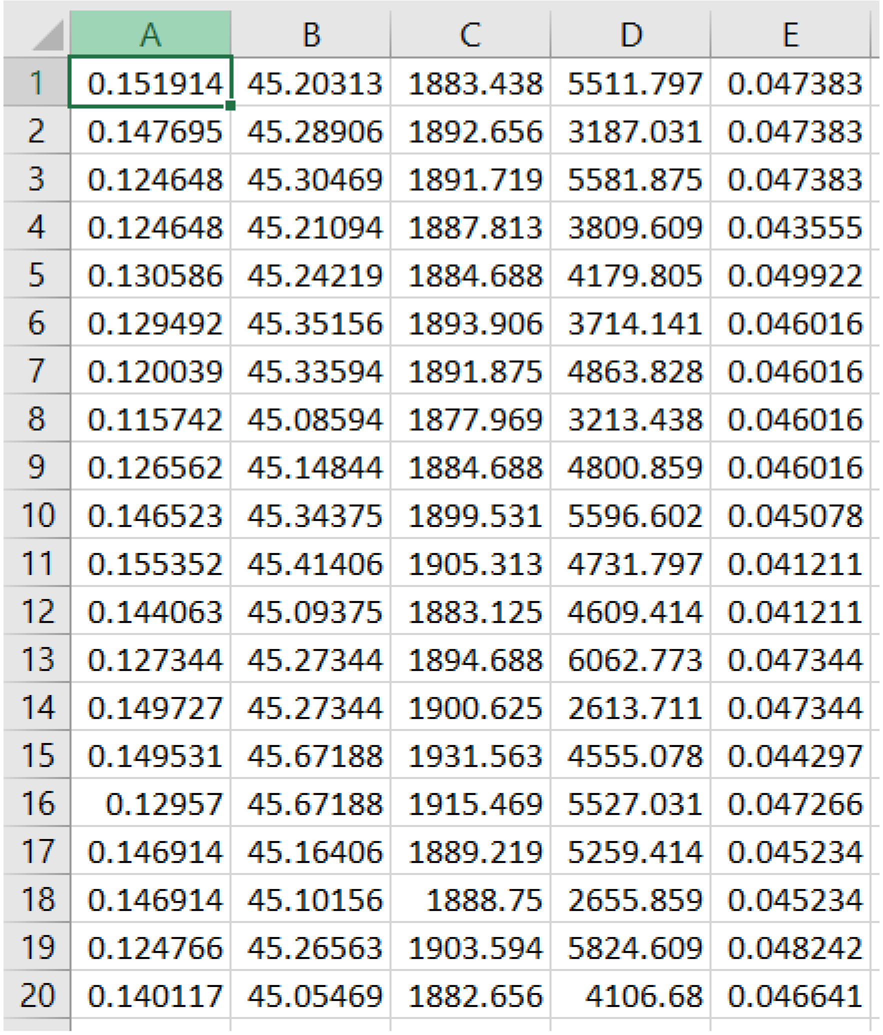
The only python script you need to test new data is the Final_Model.py script, and most of it is commented out. The only lines of code you need are lines 240 – 258 shown below.



```
239
240  with open('new_data.csv', newline='') as f:
241      reader = csv.reader(f)
242      new_data = list(reader)
243
244  loaded_model = pickle.load(open('Final_Model.sav', 'rb'))
245
246  results = {}
247
248  predicted = loaded_model.predict(new_data)
249
250  for i in range(len(pred)):
251
252      if pred[i] in results:
253          results[pred[i]] += 1
254
255      else:
256          results[pred[i]] = 1
257
258  print(results)
259
260
```

Figure 1. Code for Testing New Data

The first step to test data is to format a csv file in a way that the model can understand it. An example of how that would look like is below.



	A	B	C	D	E
1	0.151914	45.20313	1883.438	5511.797	0.047383
2	0.147695	45.28906	1892.656	3187.031	0.047383
3	0.124648	45.30469	1891.719	5581.875	0.047383
4	0.124648	45.21094	1887.813	3809.609	0.043555
5	0.130586	45.24219	1884.688	4179.805	0.049922
6	0.129492	45.35156	1893.906	3714.141	0.046016
7	0.120039	45.33594	1891.875	4863.828	0.046016
8	0.115742	45.08594	1877.969	3213.438	0.046016
9	0.126562	45.14844	1884.688	4800.859	0.046016
10	0.146523	45.34375	1899.531	5596.602	0.045078
11	0.155352	45.41406	1905.313	4731.797	0.041211
12	0.144063	45.09375	1883.125	4609.414	0.041211
13	0.127344	45.27344	1894.688	6062.773	0.047344
14	0.149727	45.27344	1900.625	2613.711	0.047344
15	0.149531	45.67188	1931.563	4555.078	0.044297
16	0.12957	45.67188	1915.469	5527.031	0.047266
17	0.146914	45.16406	1889.219	5259.414	0.045234
18	0.146914	45.10156	1888.75	2655.859	0.045234
19	0.124766	45.26563	1903.594	5824.609	0.048242
20	0.140117	45.05469	1882.656	4106.68	0.046641

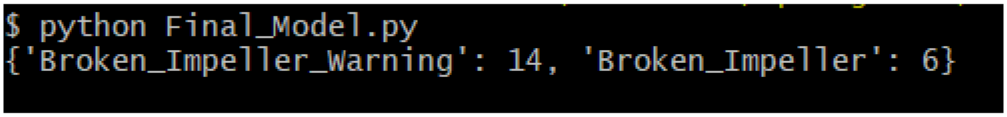
Figure 2. Example csv File

The order of the columns is x vibration, suction pressure, discharge pressure, flow rate, and y vibration from left to right. It is important that there are no headers in this file because the program will error. Also, the csv file name and the file name referenced in line 240 of the script need to be the same name.

In line 244, the model is imported. It is important that the Final_Model.sav file is in the same folder as the Final_Model.py script or it will not be able to access the model. Next, in line 246 a dictionary is initialized that will be the object the model stores the predictions in. In line 248 the model predicts the class of the data points from the csv file, and in lines 250-256 the predictions are loaded into the dictionary so that there is a count of each class that was predicted.

Finally, in line 258 the dictionary holding the counts of each predicted class is printed. This can be changed to plot a graph of your choosing using a python toolbox called matplotlib.

An example of the returned results for the data points from Figure 2 is shown below.



```
$ python Final_Model.py
{'Broken_Impeller_Warning': 14, 'Broken_Impeller': 6}
```

Figure 3. Example Results

The example data points are taken from a couple days before the first Broken_Impeller labeled data point so these results make sense. This is also a good example of how this model can be used as a condition health index. From these results we can assume the pump will need repairs soon but will not reach a critical failure for at least a few days.