WATER LEAK DETECTION SYSTEM

COMPUTING FOR AI & MACHINE LEARNING

CCE AUG – DEC 2022

SRIDEVI NARAYAN

Problem Statement

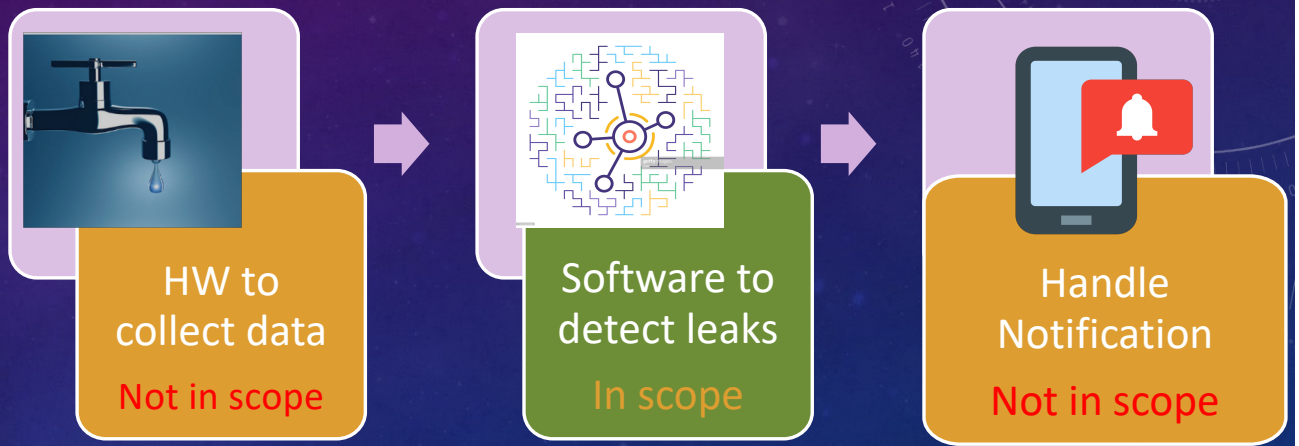
At homes, we find that water is wasted. Mainly attributable to

- Water leaks due wear and tear of the plumbing
- Or taps are not closed properly.
- Or overflow

Proposed Solution

Water Leak Detection System

A non-invasive system which can provide notifications on detecting a leak

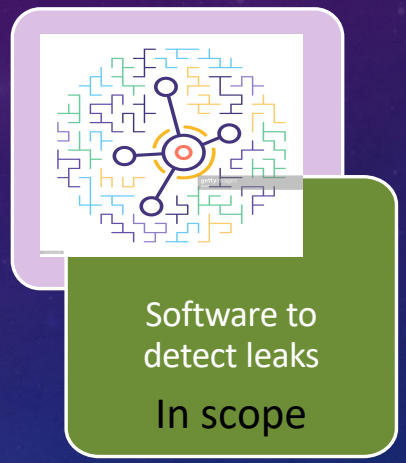


Sridevi Narayan

PROJECT SCOPE

Steps

1. Collect data manually (water drip, running water, white noise)
2. Convert into .wav files
3. Extract features
4. Build and compile a model
5. Train the model (incl. experiments with
    1. Test split percentage change
    2. #epochs
    3. Normalise data
6. Evaluate and predict

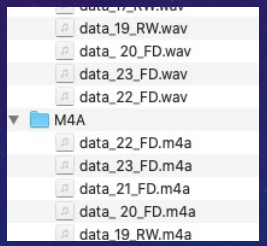


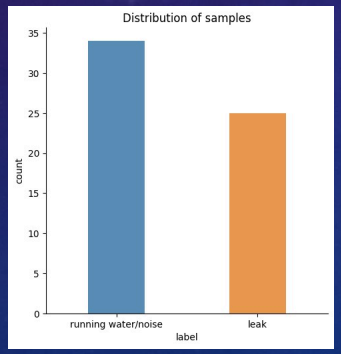
Leak present or not
1 or 0

COLLECT DATA AND FEATURE EXTRACTION

1. Audio files

- Used iphone to record sounds in .m4a format
  - Drip, Running Water and whitenoise
  - Differing heights and differing flows
- Labelled the files with a naming convention
- Used a third-part tool (ConverterLite) to convert to .WAV files



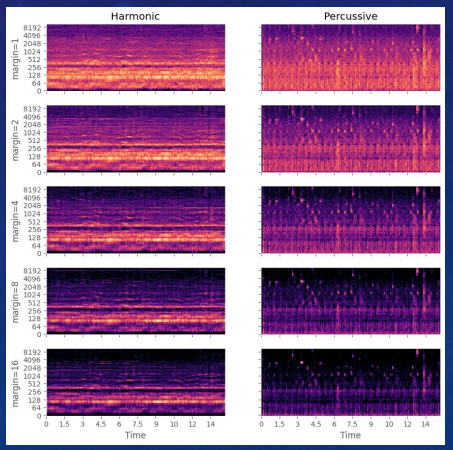


Sridevi Narayan

2. Feature Extraction

- Use librosa library for extracting features
- Around 8-10 features were looked at, and looking at the plots, chose the following features. Alongwith it, the sample name and label were also stored

‘ spectral_centroid',
'spectral_bandwidth',
'spectral_rolloff',
'onset_strength',
'mfcc',
'zero_crossing_rate',
'percussion'



FEATURE EXTRACTION

Sample#	spectral_centroid	spectral_bandwidth	spectral_rolloff	onset_strength	mfcc	zero_crossing_rate	percussion	label
0	2511.3862502250300	2729.875911571640	5842.98490636489	1.0450059	-9.416307	0.11145429851628200	0.00043898445	running water/noise
1	4404.511193339160	2927.3963228064900	8127.7172191723000	1.1537287	-6.2394376	0.3326127885698200	0.001990815	running water/noise
2	1493.9859350581200	2298.5555997648000	3552.2040758634900	0.96454114	-7.512391	0.03440583881578950	0.00031832934	running water/noise
3	6176.509395713940	2558.9978452407700	9052.090669014080	1.0142285	-9.270225	0.5861673949949700	0.0047904216	running water/noise
4	3261.1774459395600	2767.3630921233400	6846.577785326090	1.0439733	-12.088974	0.20098788496376800	0.0019871066	running water/noise
5	3827.0907340096800	2897.3937667993000	7378.245206643610	1.8293152	-8.2832155	0.19734389909351100	0.0066237394	running water/noise
6	2159.6084465598300	2528.3402639510600	4900.780391693120	1.0256083	-6.2748113	0.0838460922241211	0.0007355827	running water/noise
7	2457.0649969961400	2855.9773280872200	6046.647832498130	1.0226448	-5.8739867	0.073238296812749	0.0012670269	leak
8	2991.407623938850	2866.1983307544800	6351.338902285940	1.335028	-6.9454026	0.13139823929704000	0.0011241431	running water/noise
9	1942.2015958817400	2440.676151564850	4443.727401540250	1.1578088	-7.0780997	0.05965375296677220	0.00056884665	running water/noise
10	1674.740328857120	2252.133133573060	3710.8027109842800	1.0940149	-14.702501	0.05152780376746510	0.0003706649	leak
11	1955.6517669938600	2398.773748252550	4323.337493733290	0.96410817	-16.73592	0.0337384065842246	5.8112168E-05	running water/noise
12	2568.95052810829	2762.7852079044100	5791.420412853940	1.62908	-8.389852	0.07530456880179560	0.0064972	leak

1. Spectral Centroid:The **spectral centroid** indicates at which frequency the energy of a spectrum is centered upon or in other words It indicates where the ” center of mass” for a sound is located. This is like a weighted mean:
2. Spectral_bandwidth : It is the spectral range of interest around the centroid, that is, the variance from the spectral centroid
3. spectral_rolloff It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0.
4. MFCC coefficients model the spectral energy distribution in a persistent and meaningful way, thus they are most widely feature for audio classification
5. zero_crossing_rate: is simply the number of times a waveform crosses the horizontal time axis.
6. Percussive part – Separated the harmonic and percussive parts.
7. Onset strength/Spectral flux: : Spectral flux measures the spectral change between two successive frames

NN ARCHITECTURE

```
model = Sequential([
        layers.Dense(100, activation='relu'),
        layers.Dense(100, activation="relu"),
        layers.Dense(2, activation="softmax")
                ])

model.compile(loss='categorical_crossentropy',
          metrics=['accuracy'], optimizer='Adam')
```

- #Nodes chosen based on literature and some experiments by tinkering with the parameter
- Categorical Crossentropy:
  - Categorical – Classification problem
  - Cross Entropy
    - "distance" between two probability distributions,
    - the network would be learning to minimize the "distance" between two probability distributions:
      - the first is the output vector which has been scaled by a softmax (all values sum to 1) and
      - the ground truth vector, which is just a probability of 1 for our target class, and a probability of 0 elsewhere.

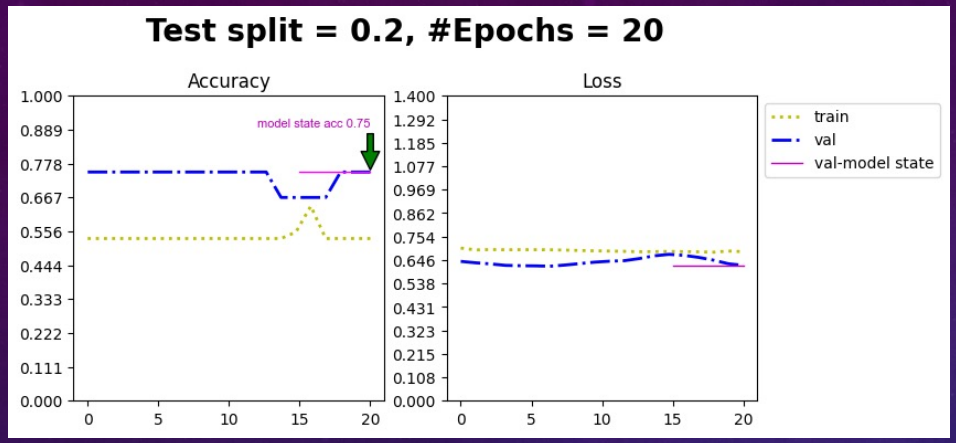
TRAINING RESULTS

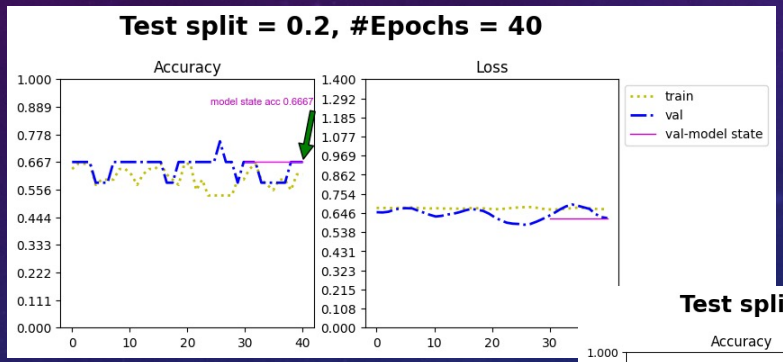
Sridevi Narayan

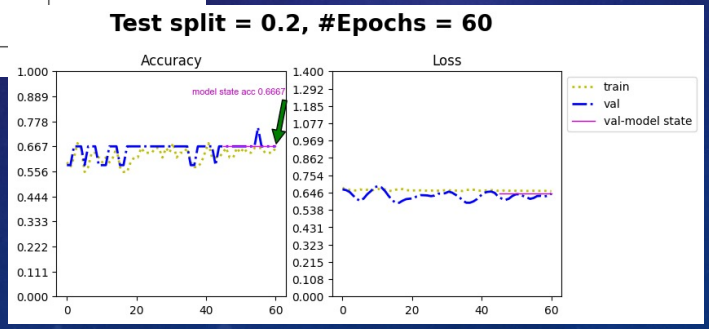
TRAINING, VALIDATION AND PREDICTION

- Training was carried out with different combinations of training /test split and #epochs
  - split_set = [0.2, 0.3, 0.4]
  - num_epochs = [20, 30, 40, 50, 60, 70, 80, 90, 100]
- This was done for both normalized data (features) and un-normalized.
- Post-training, the test score was evaluated
- The loss and accuracy metrics were looked at during training and for the test set. Across epochs
- The metrics were looked at for the prediction (using the test set only)

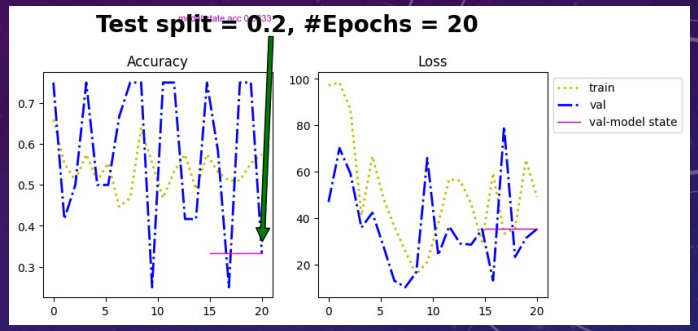
NORMALIZED

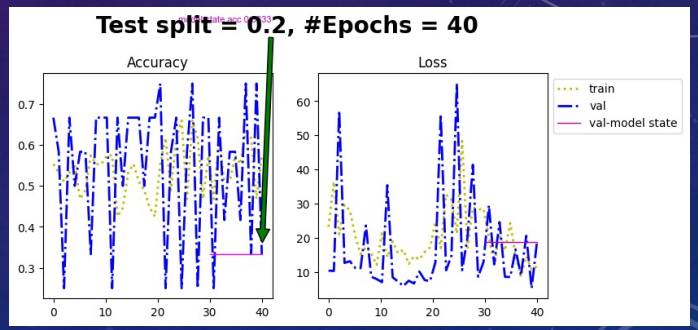


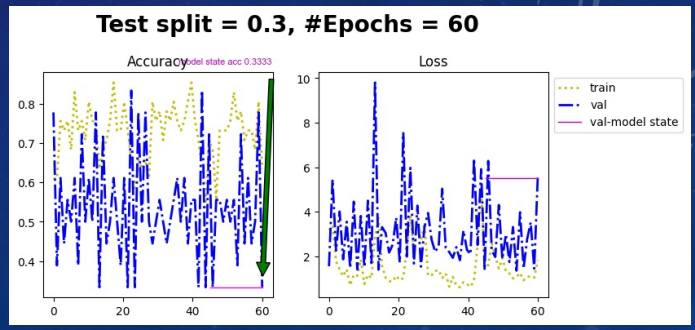




UN NORMALIZED

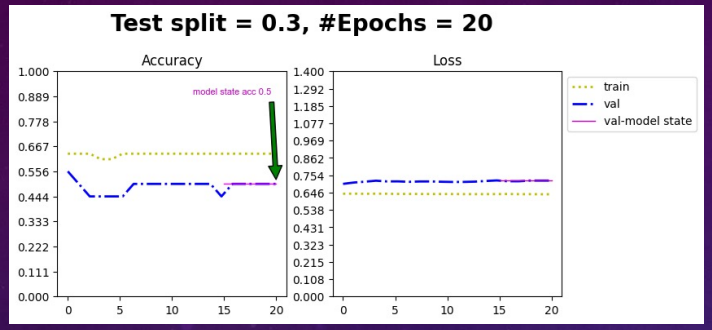


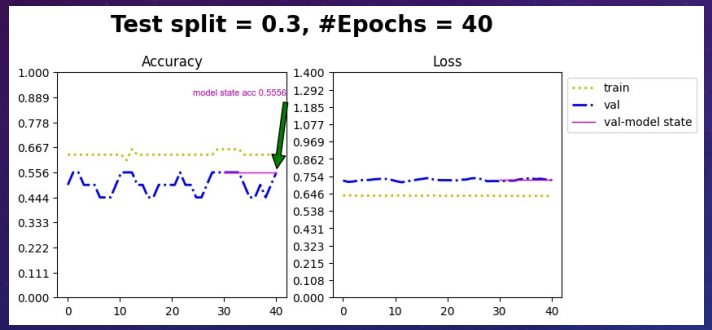


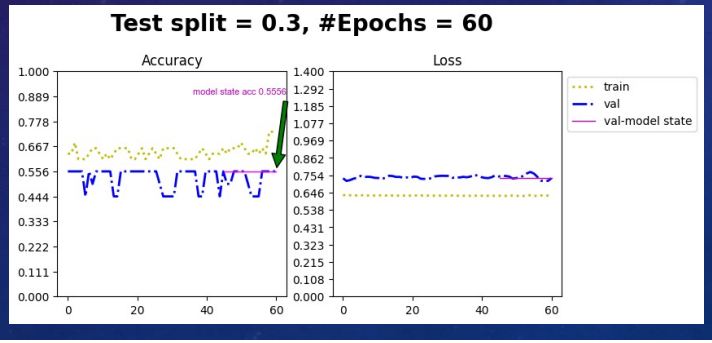


Sridevi Narayan

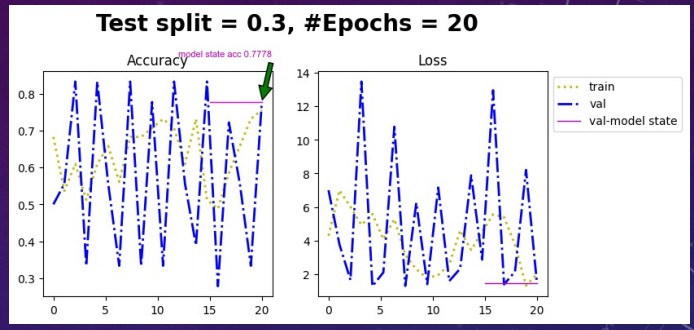
NORMALIZED

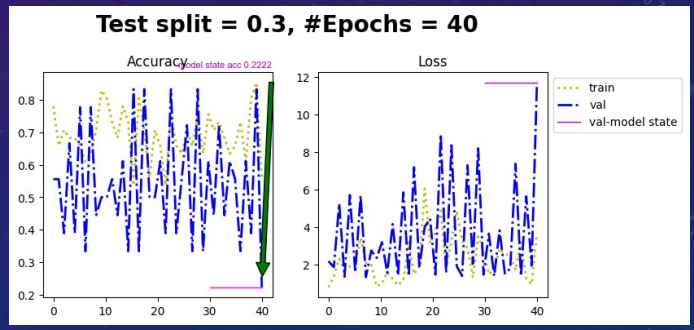


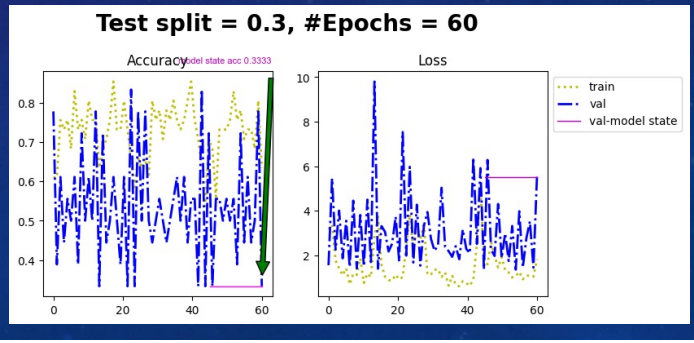




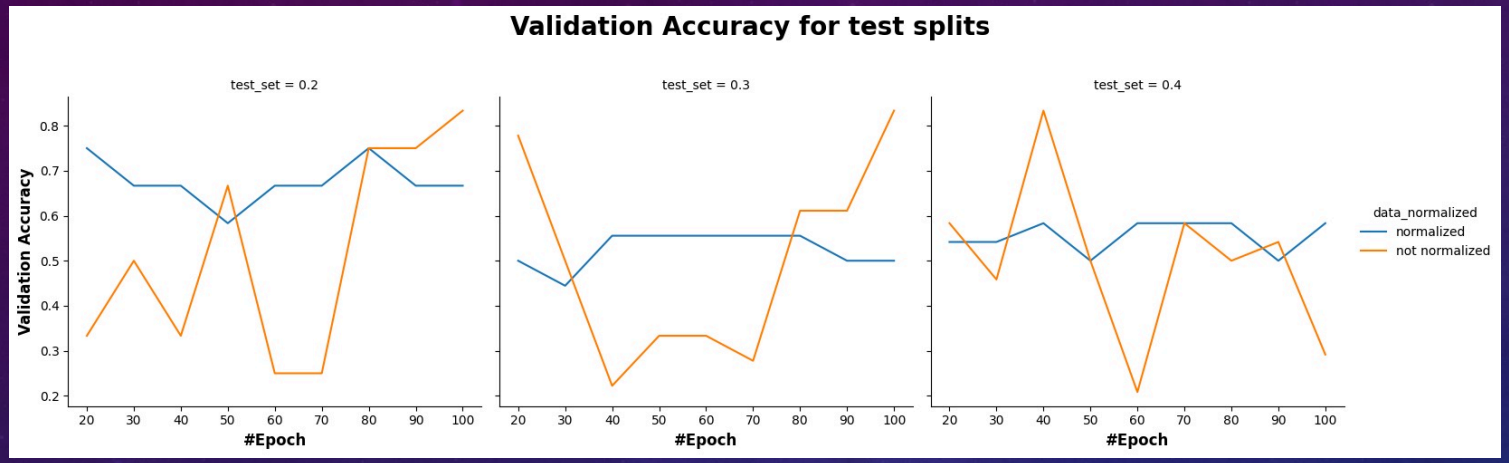
UN NORMALIZED





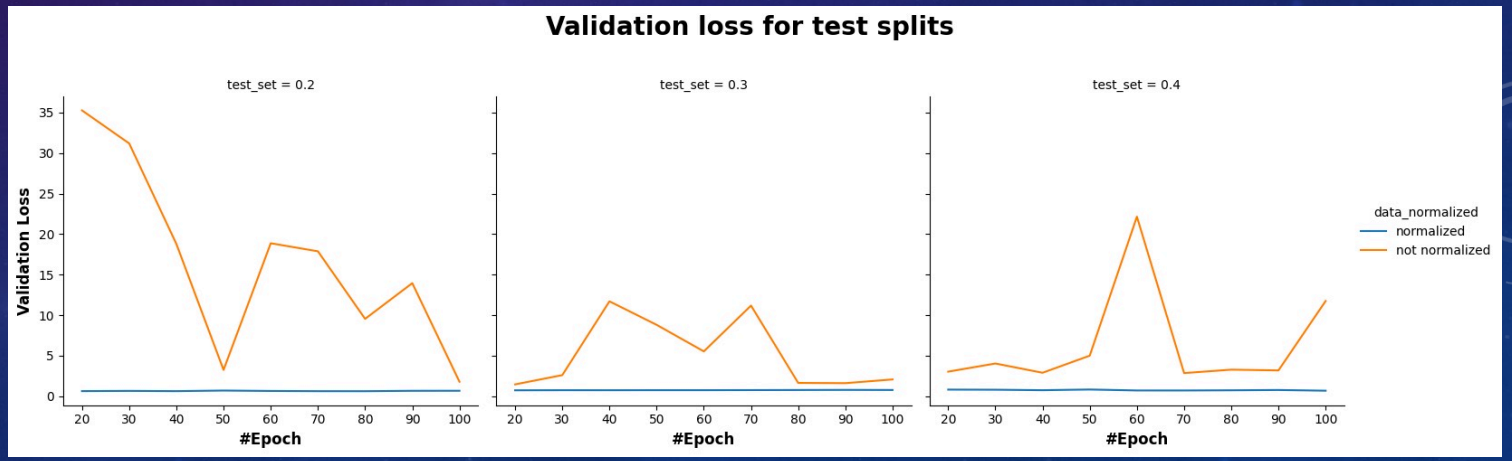


SUMMARY – ACCURACY AND LOSS



Results are stable for Normalized data
Best:
Accuracy 75%
Loss

Results fluctuating for
Un normalized data



Sridevi Narayan

PREDICTION Best Results: 75% Accuracy

consolidated_metrics

test_set	tr_epoch	pred	label	test_score
0.2	20	[1 1 1 1 1 1 1 1 1 1 1 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6213282942771912, 0.75]
0.2	30	[1 1 1 1 1 0 1 0 1 1 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6468111872673035, 0.6666666865348816]
0.2	40	[1 1 1 1 1 0 1 0 1 1 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6175420880317688, 0.6666666865348816]
0.2	50	[0 1 0 1 0 0 0 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6891528964042664, 0.5833333134651184]
0.2	60	[0 1 1 1 1 0 1 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6394063830375671, 0.6666666865348816]
0.2	70	[0 1 1 1 1 0 1 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6148672103881836, 0.6666666865348816]
0.2	80	[1 1 1 1 1 0 1 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6112706065177917, 0.75]
0.2	90	[0 1 1 1 1 0 1 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6573533415794373, 0.6666666865348816]
0.2	100	[0 1 1 1 1 0 1 0 1 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1]	[0.6625798344612122, 0.6666666865348816]
0.3	20	[0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7204411029815674, 0.5]
0.3	30	[0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7339358329772949, 0.4444444477558136]
0.3	40	[0 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7303650379180908, 0.5555555820465088]
0.3	50	[0 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7364193797111511, 0.5555555820465088]
0.3	60	[0 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7352089881896973, 0.5555555820465088]
0.3	70	[0 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7462310791015625, 0.5555555820465088]
0.3	80	[0 1 1 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7575891613960266, 0.5555555820465088]
0.3	90	[0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.771270215511322, 0.5]
0.3	100	[0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1]	[0.7590370774269104, 0.5]
0.4	20	[0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1]	[0.8074951171875, 0.5416666865348816]
0.4	30	[0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1]	[1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1]	[0.7925899624824524, 0.5416666865348816]

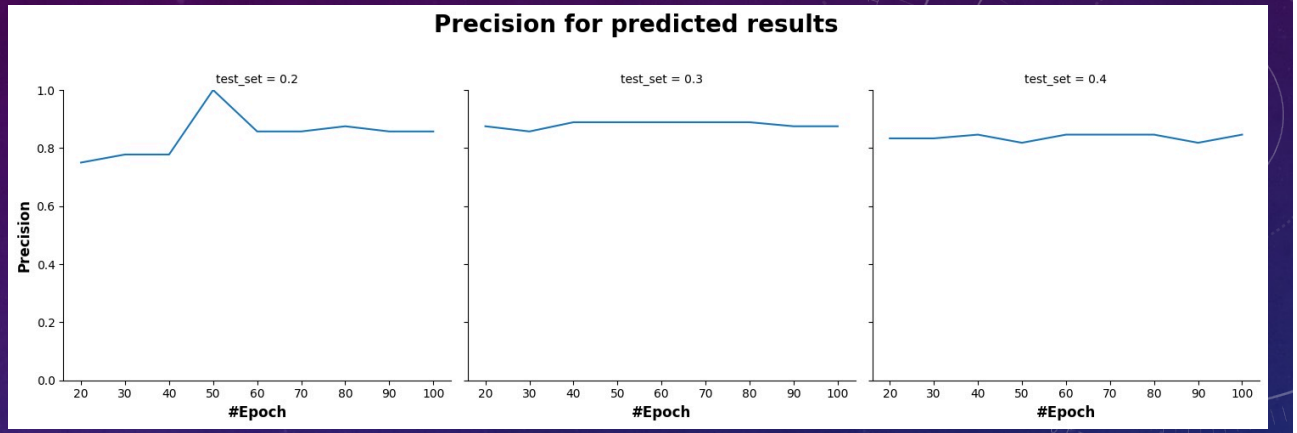
Sridevi Narayan

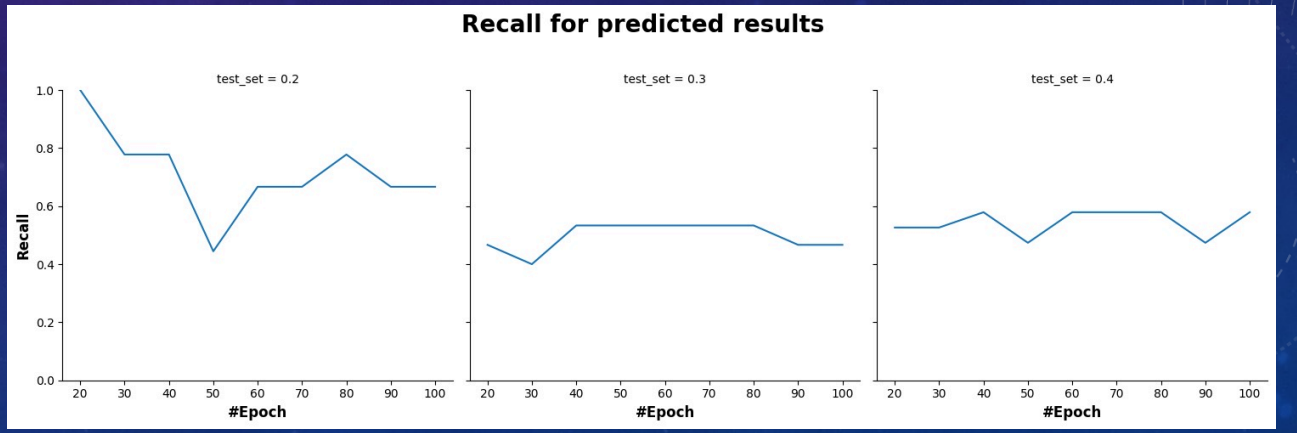
All sources, raw data, processed data, plots and results are available in github
git@github.com:sridevinarayan/git-github.com-sridevinarayan-WLDS2.git

PREDICTION – PRECISION AND RECALL

- Precision is high. (78%-90%)
- Recall is lower - (45% -50%)

- This implies that on the average,
  - True Positives are high (Detected many leaks)
  - False Negatives are also hight (Did not detect some leaks)





NEXT STEPS

- Analyse Accuracy in the metrics against precision and recall
- More sample collection – automated
- Model experiments- both algorithms and parameters. Focus on improving accuracy and recall
- Complete system with HW and notifications included.
- Check with image classification for mfcc instead of all features used

THANK YOU

Sridevi Narayan

BACKUP

Sridevi Narayan

DEVELOPMENT ENVIRONMENT

- PYTHON - Jupiter NB
- LIBROSA
- PANDAS
- MATPLOTLIB
- NUMPY
- TENSORFLOW/ KERAS
- SEABORN
- GITHUB

All sources, raw data, processed data, plots and results are available in github
git@github.com:sridevinarayan/git-github.com-sridevinarayan-WLDS2.git