PATTERN RECOGNITION AND MACHINE LEARNING

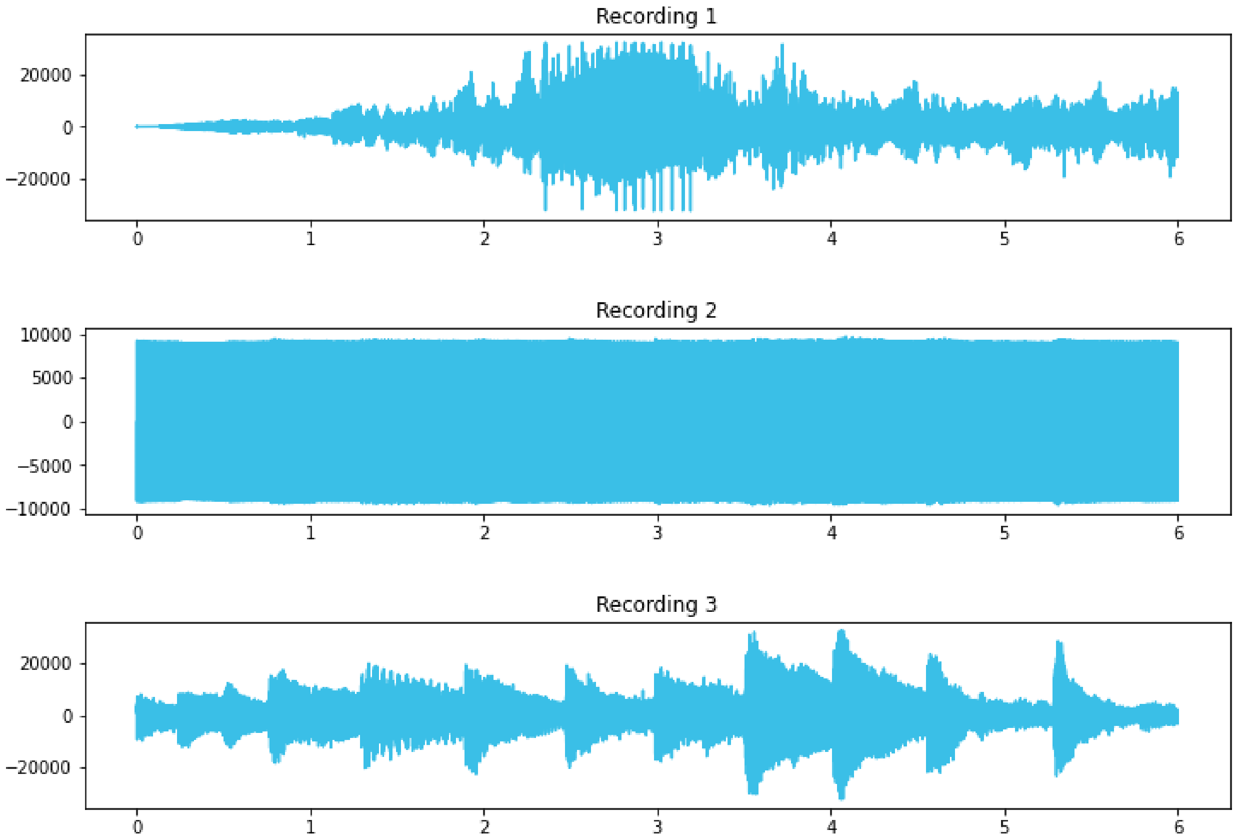
LAB 08

LAB REPORT

QUESTION 1.

1)

- Data is provided in .wav format, which can be worked with using python libraries.
- The visualization of the sounds is as follows:



2)

- Wave library is used to extract raw audio from the input.
- Numpy c_ is used to combine the audio and mix them.

3)

- ICA: It is an algorithm to separate a multivariate signal into its underlying components.
- The algorithm relies on three assumptions given below:

    1.) The Sources are statistically independent.

    2.) Each independent component has a non-gaussian distribution.

    3.) There are an equal number of observations and sources.

- Implementation:

    • ICA Pre-processing:

    1.) Centering: Subtracting the sample mean which makes our statistical model (ICA) zero-mean.

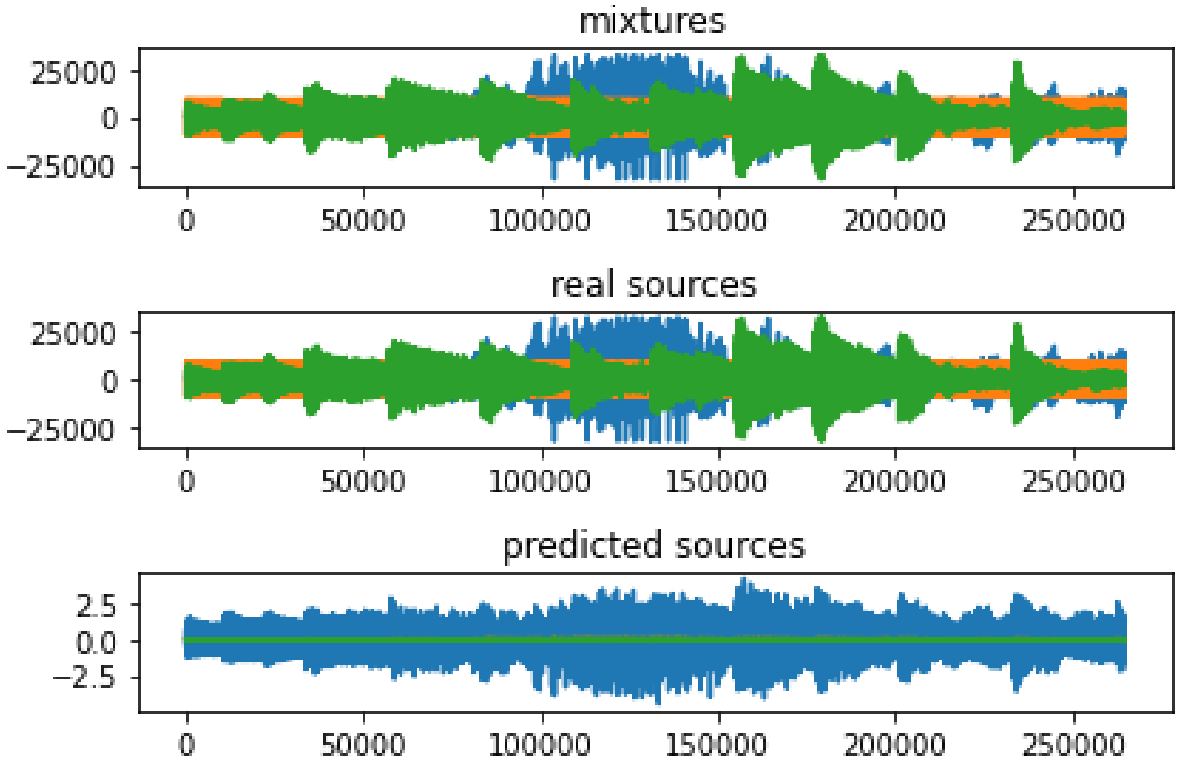
    2.) Whitening: To remove the potential correlations between the components. Whitening ensures that all dimensions are treated equally a priori before the algorithm is run.

- ICA Estimation: $x = As$

    The above equation is the matrix representation of mixing model, here x represents the observed Sources where A is the mixing matrix and s are the sources now we need to estimate the demixing matrix W to find the sources

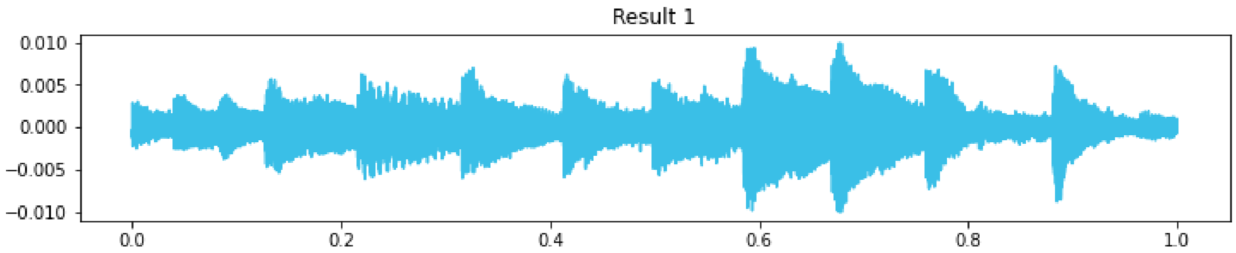
- $s = Wx$ We randomly initialize the demixing matrix and run a iterative algorithm where in each iteration the values of our demixing matrix are updated until convergence is achieved , Convergence is said to be attained when the following condition is fulfilled: Dot product of w and w_transpose is roughly equal to 1.

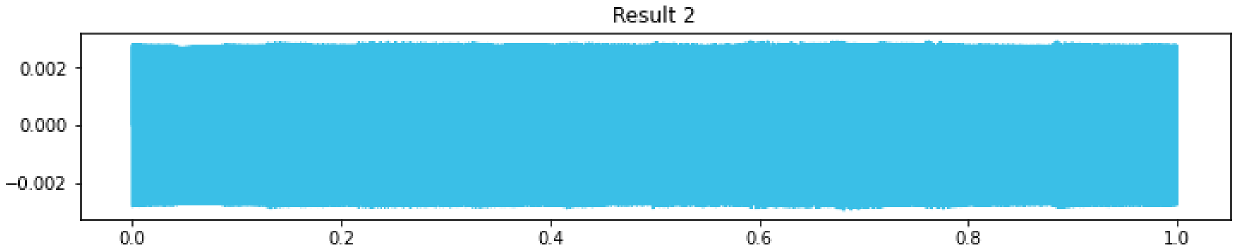
4)

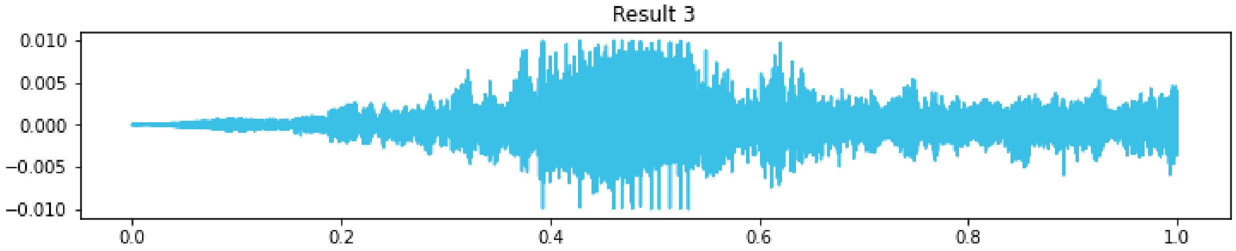
- 

5) 6)

- SKLearn is used to implement FastICA.
- VIsualisaiton of the reported signals is shown below:







7)

- Fast ICA is more robust to noise in the mixed signal when compared to typical implementation of ICA.
- Fast ICA is computationally efficient and requires less memory.

QUESTION 2.

1)

- The data is categorically encoded.
- Columns with little to no significance are dropped.
- Predefined python libraries are used to create a SFS object and embed Decision Tree Classifier into it.

2) 3)

- 10 best features: `'Customer Type',`
- `'Type of Travel',`
- `'Class',`
- `'Inflight wifi service',`
- `'Gate location',`
- `'Online boarding',`
- `'Seat comfort',`
- `'Inflight entertainment',`
- `'Baggage handling',`
- `'Inflight service'`
- Accuracy: 95.07 %
- CV scores : `[0.94990106, 0.9505285 , 0.94888749, 0.95274868, 0.95144319]`

4)

- SFS: `'cv_scores': array([0.94891695, 0.94984362, 0.9498803 , 0.95111592]),`

  `'avg_score': 0.9499391964575459`

- SBS: `'cv_scores': array([0.95115642, 0.95115642, 0.9507684 , 0.95300795]),`

  `'avg_score': 0.9515222959993955`

- SFFS: `'cv_scores': array([0.95088613, 0.95127225, 0.95042088, 0.95296934]),`

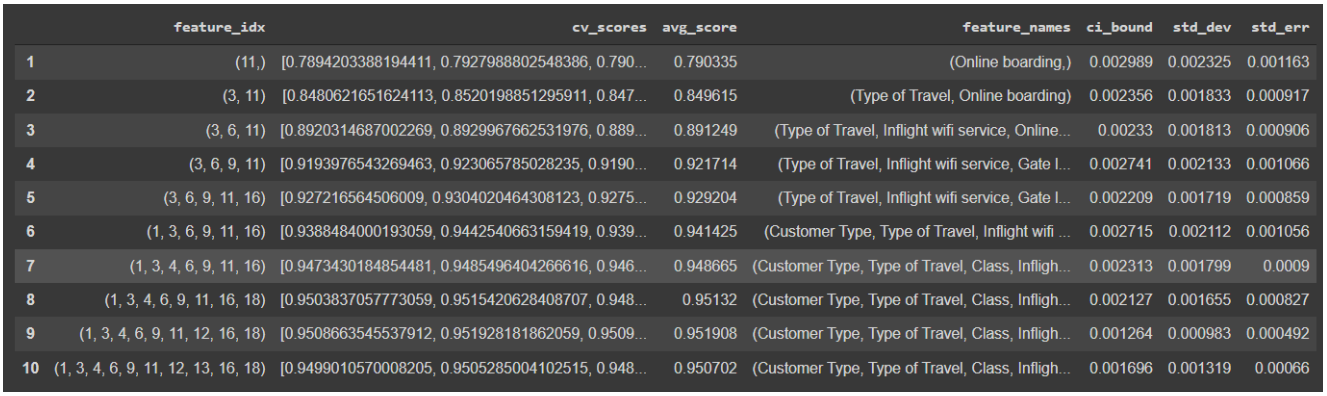
  `'avg_score': 0.9513871519192167`

- SFS: `'cv_scores': array([0.94802888, 0.94992085, 0.94999614, 0.95042088]),`

  `'avg_score': 0.9495916871020177`

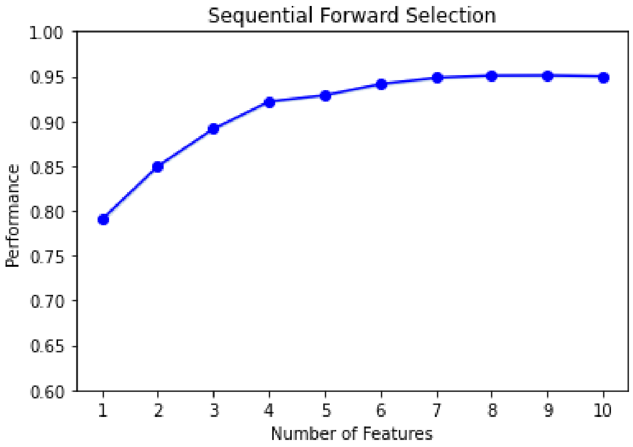
5)

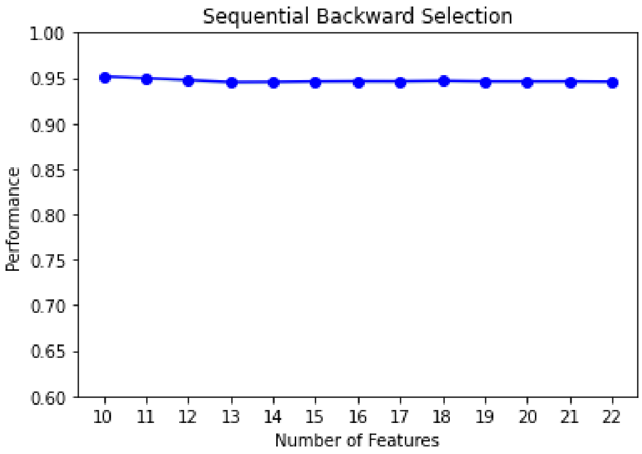
- get_metric_dict is used for all 4 configurations from the pandas Dataframe to visualize the output.
- An example is pasted:

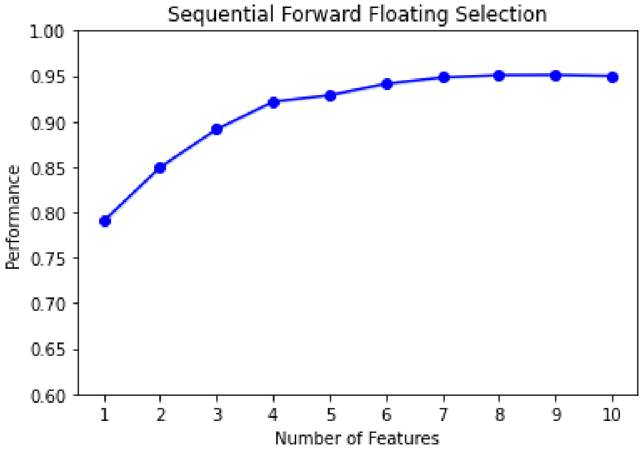


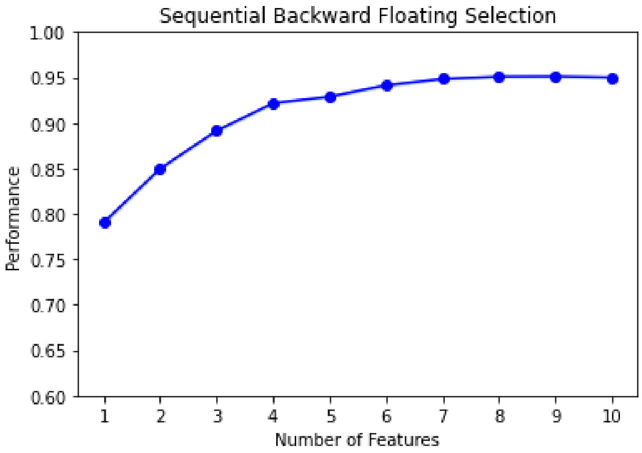
	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
1	(11,)	[0.7894203388194411, 0.7927988802548386, 0.790...	0.790335	(Online boarding,)	0.002989	0.002325	0.001163
2	(3, 11)	[0.8480621651624113, 0.8520198851295911, 0.847...	0.849615	(Type of Travel, Online boarding)	0.002356	0.001833	0.000917
3	(3, 6, 11)	[0.8920314687002269, 0.8929967662531976, 0.889...	0.891249	(Type of Travel, Inflight wifi service, Online...	0.00233	0.001813	0.000906
4	(3, 6, 9, 11)	[0.9193976543269463, 0.923065785028235, 0.9190...	0.921714	(Type of Travel, Inflight wifi service, Gate l...	0.002741	0.002133	0.001066
5	(3, 6, 9, 11, 16)	[0.927216564506009, 0.9304020464308123, 0.9275...	0.929204	(Type of Travel, Inflight wifi service, Gate l...	0.002209	0.001719	0.000859
6	(1, 3, 6, 9, 11, 16)	[0.9388484000193059, 0.9442540663159419, 0.939...	0.941425	(Customer Type, Type of Travel, Inflight wifi ...	0.002715	0.002112	0.001056
7	(1, 3, 4, 6, 9, 11, 16)	[0.9473430184854481, 0.9485496404266616, 0.946...	0.948665	(Customer Type, Type of Travel, Class, Infligh...	0.002313	0.001799	0.0009
8	(1, 3, 4, 6, 9, 11, 16, 18)	[0.9503837057773059, 0.9515420628408707, 0.948...	0.95132	(Customer Type, Type of Travel, Class, Infligh...	0.002127	0.001655	0.000827
9	(1, 3, 4, 6, 9, 11, 12, 16, 18)	[0.9508663545537912, 0.951928181862059, 0.9509...	0.951908	(Customer Type, Type of Travel, Class, Infligh...	0.001264	0.000983	0.000492
10	(1, 3, 4, 6, 9, 11, 12, 13, 16, 18)	[0.9499010570008205, 0.9505285004102515, 0.948...	0.950702	(Customer Type, Type of Travel, Class, Infligh...	0.001696	0.001319	0.00066

6)


Sequential Forward Selection

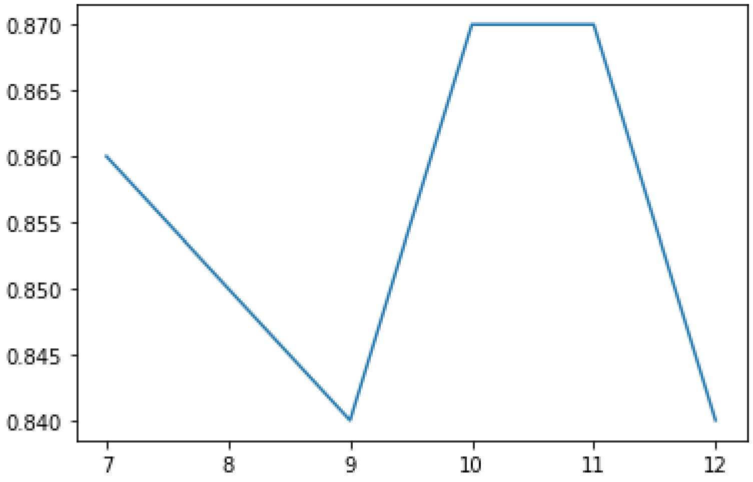

Sequential Backward Selection


Sequential Forward Floating Selection


Sequential Backward Floating Selection

7)

- A graph was created by considering the first 100 rows, since it took a large amount of time.


0.870
0.865
0.860
0.855
0.850
0.845
0.840
7
8
9
10
11
12