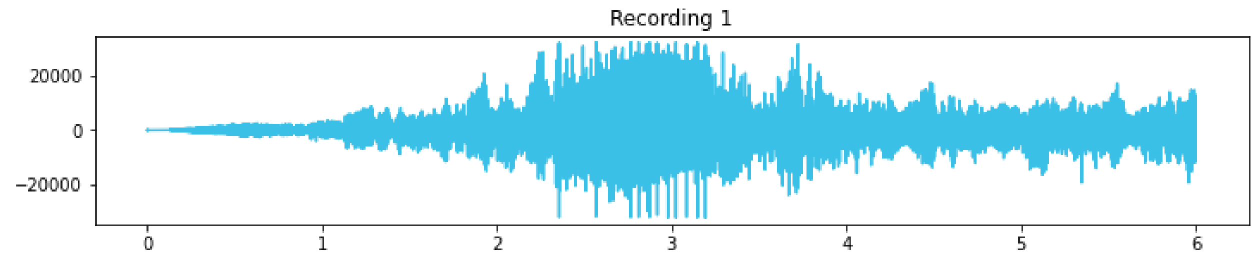
Pattern Recognition and Machine Learning
Report for : Lab 8

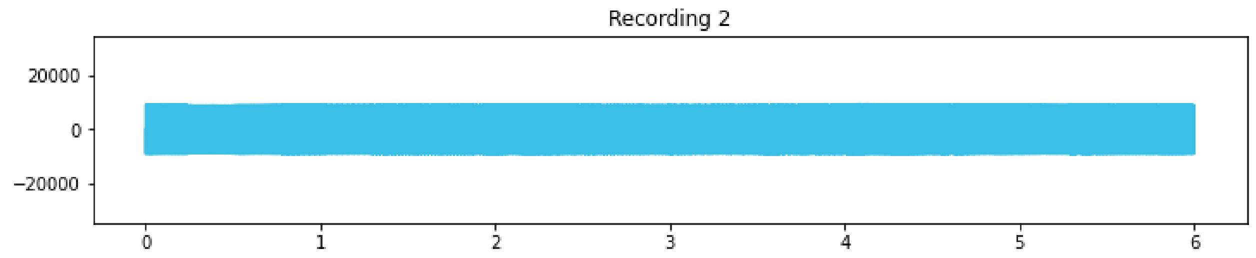
Name : Khushi Parikh

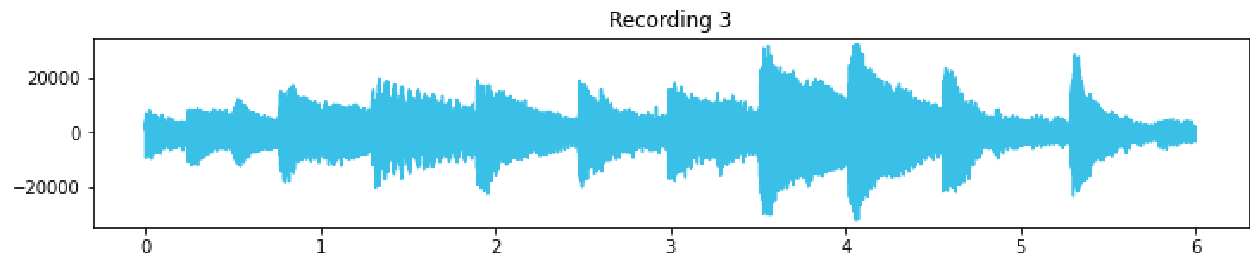
Roll No : B20EE029

Question 1:

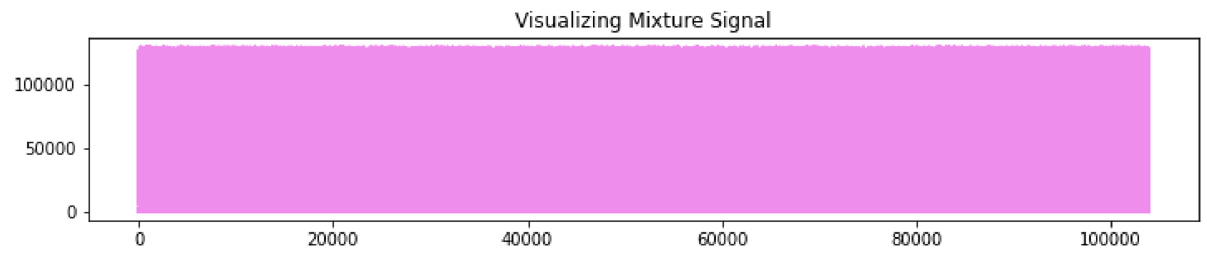
- **(1.1 + 1.2)** Three input signals were imported using a library called '**wave**'. The parameters were found and the length of audio was nframes/framerate = 5.99 seconds. The raw data was extracted and converted to a numpy array. This was then visualized using matplotlib.







- The sounds were listened to using the '**IPython**' library. An array of the three signals was constructed of size (264515,3).
- Plot of mixed signal -



- **(1.3)** ICA was implemented from scratch. The steps are -
    1) Center X by **subtracting its mean** from it.
    2) **Whiten** X. This refers to removing the correlations from the covariance matrix such that the covariance matrix becomes an identity matrix. This can be done by eigenvalue decomposition of its covariance matrix. Formula -

$$\tilde{x} = ED^{-1/2}E^T x$$

       Here, E is an orthogonal matrix of eigenvectors and D is a diagonal matrix of eigenvalues.
    3) Now, we choose a random value for the demixing matrix by using `np.random.rand` The number of components is the length of each signal.
    4) We now need to **find a new de-mixing(w) matrix** such that either it **converges** or the maximum iterations = 1000 have been reached. If it converges then the product of w and its transpose will approximately be 1. Formulae -

$$w_p = \frac{1}{n}\sum_i^n X g(W^T X) - \frac{1}{n}\sum_i^n g^{'}(W^{TX})W$$

$$w_p = w_p - \sum_{j=1}^{p-1}(w_p^T w_j)w_j$$

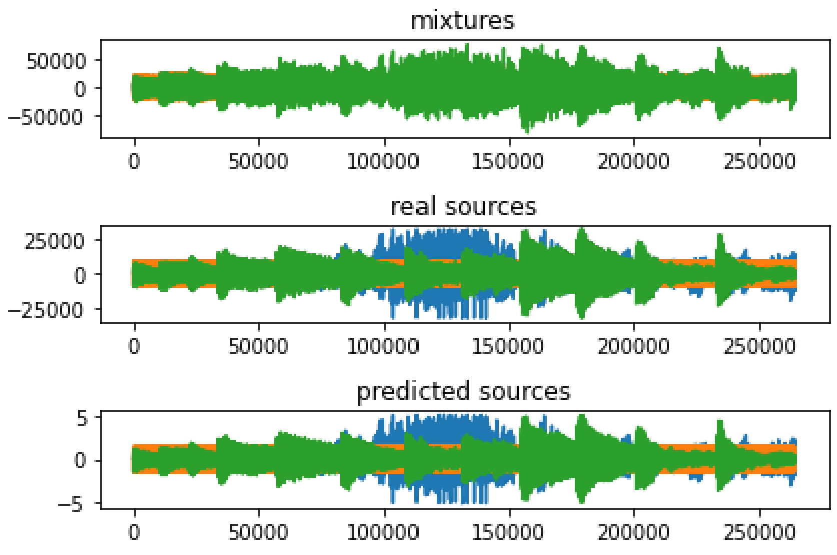
$$w_p = \frac{w_p}{||w_p||}$$

       The function g(x) here is the hyperbolic tan function.
    5) To get the individual signals, formula -

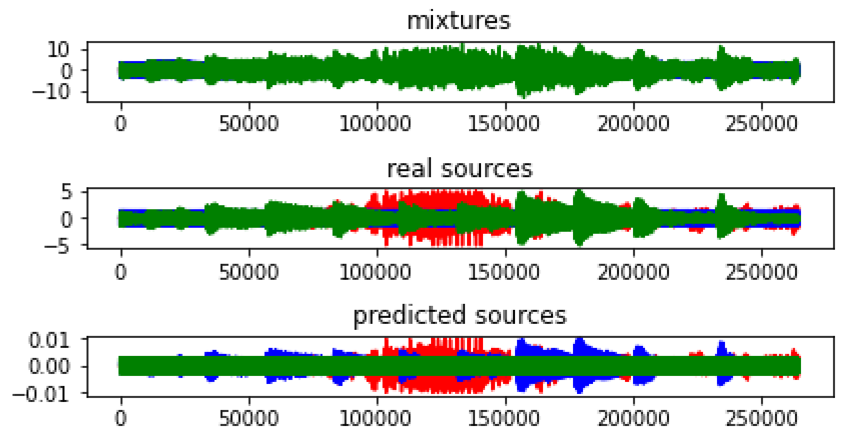
$$S = Wx$$

       Where w is a de-mixing matrix and X is the original mixture of signals.

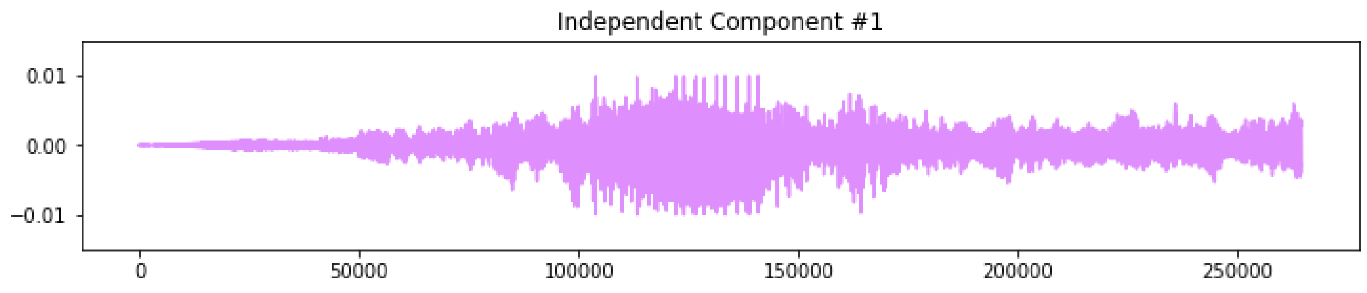
- **(1.4)** Plot of mixed,original and predicted sources **(scratch)** -

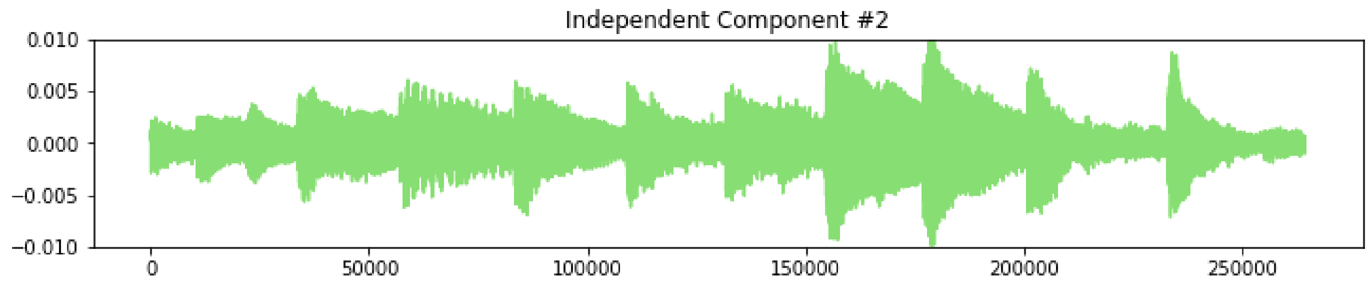


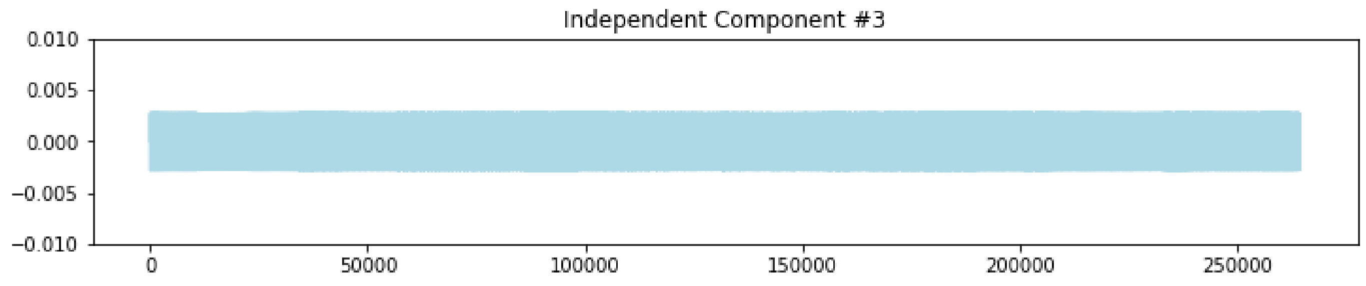
- **(1.5) FastICA** was imported from sklearn.decomposition and n_components was chosen as 3. Output -



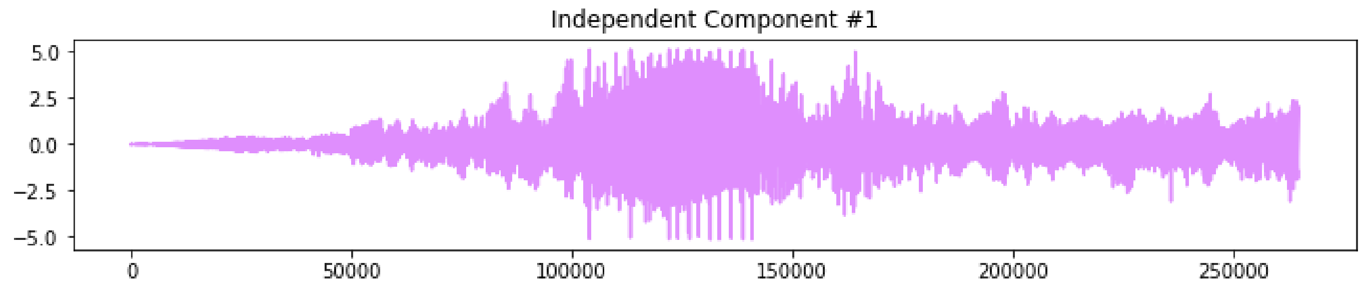
- **(1.6)** The individual signals were extracted from the final array and plotted **(FastICA)**-

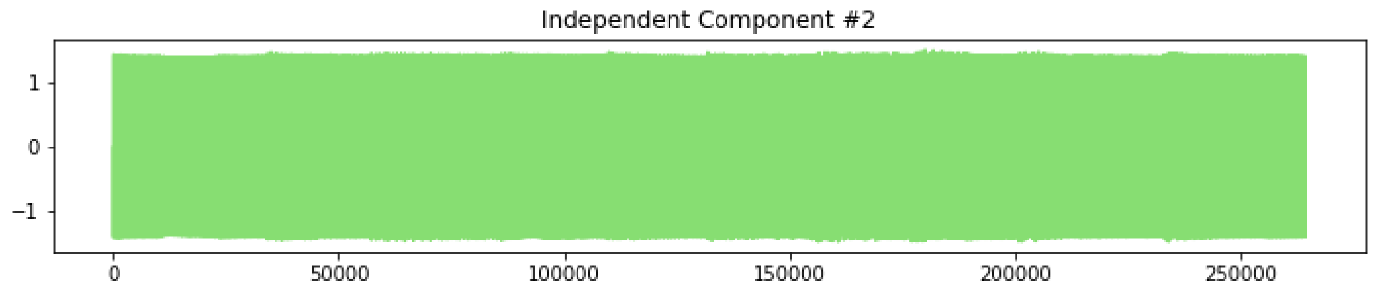


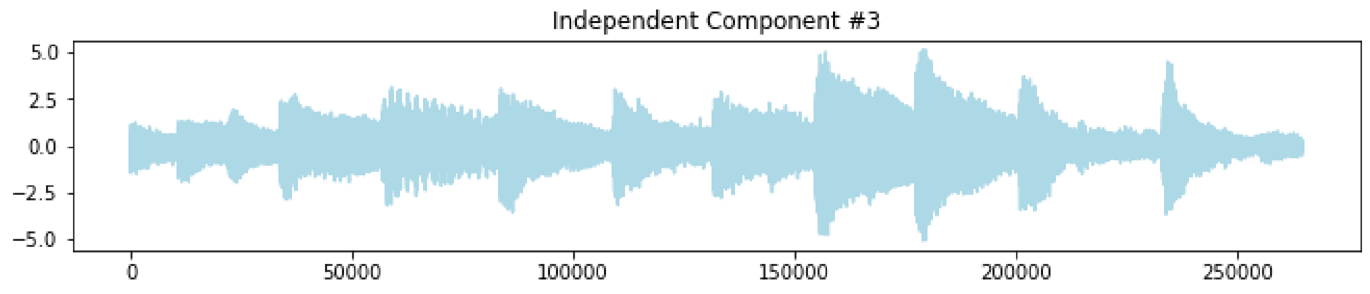




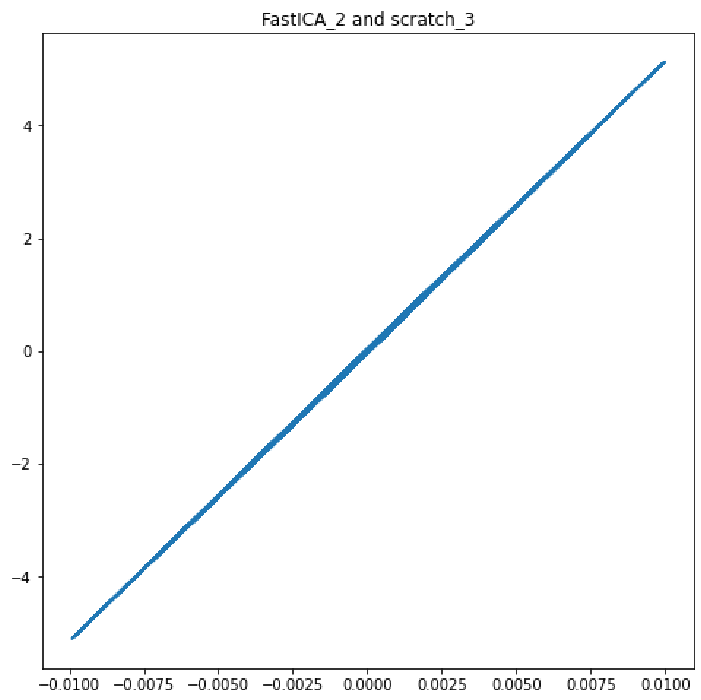
- The audio files were also listened to (part of code).
- The individual signals were extracted from the final array and plotted **(Scratch)** -

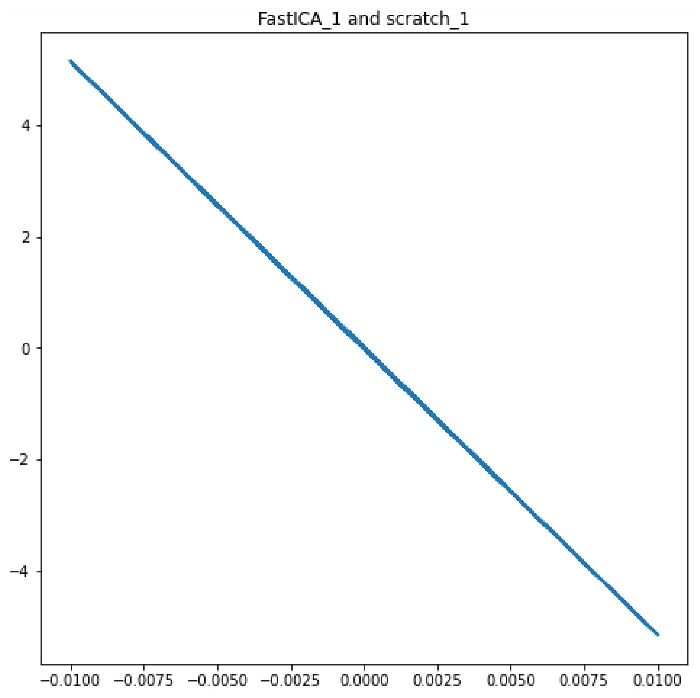


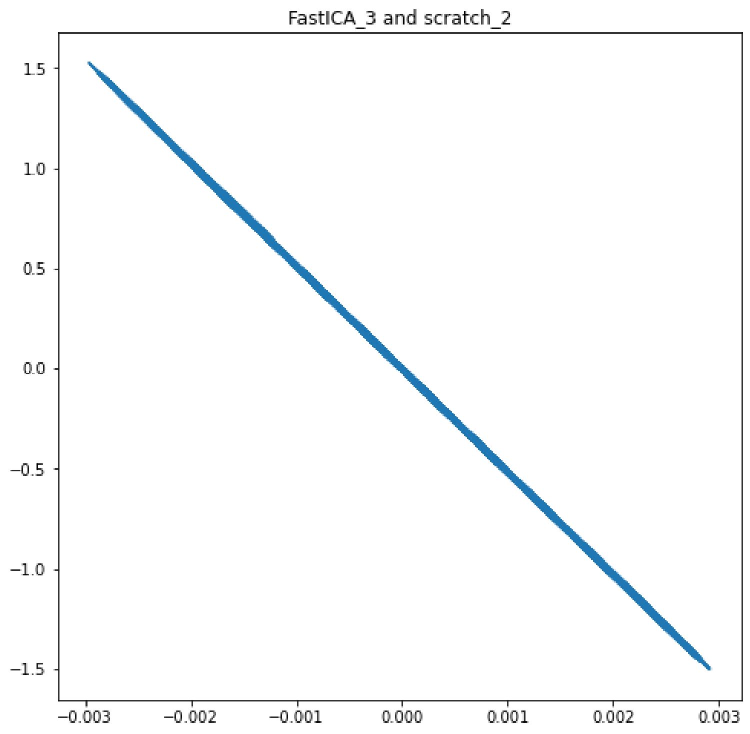




- **(1.7)** The results obtained from ICA and FastICA were very similar and had very small MSE. Graphs -





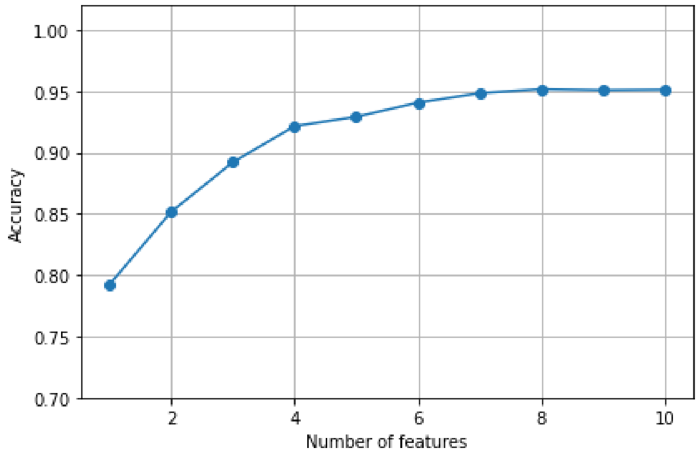


Question 2:

- **(1.1)** The data was imported and one column - `'Arrival Delay in Minutes'` had null values which were replaced by the mean. `Unnamed: 0` was dropped as it was irrelevant. The needed features were label encoded.
- **(1.2)** The required object was made :

```
SequentialFeatureSelector(estimator=DecisionTreeClassifier(random_state=0),
                          k_features=10, scoring='accuracy')
```

- **(1.3)** SFS was implemented from scratch using the following steps -
    1) A deep copy of the estimator was made. Number of features to be chosen was passed as a parameter.
    2) In the fit function, we passed each parameter to see which gave the **maximum accuracy**. The score and index was stored in an array.
    3) After this, while the number of features was less than the chosen parameter, we passed the selected feature with each of the remaining features to see which **group of features maximized the accuracy**. This was then added to the array.
- Scores of each of the 10 features :

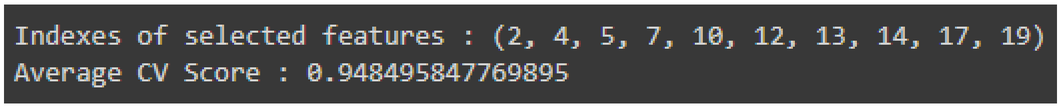


```
[0.7922494546387784,
 0.8516938277941742,
 0.8922109585525472,
 0.9216283844475812,
 0.9292313614782497,
 0.940812267419479,
 0.9484794045938663,
 0.9517515719235211,
 0.9509174900551777,
 0.9513024509174901]
```

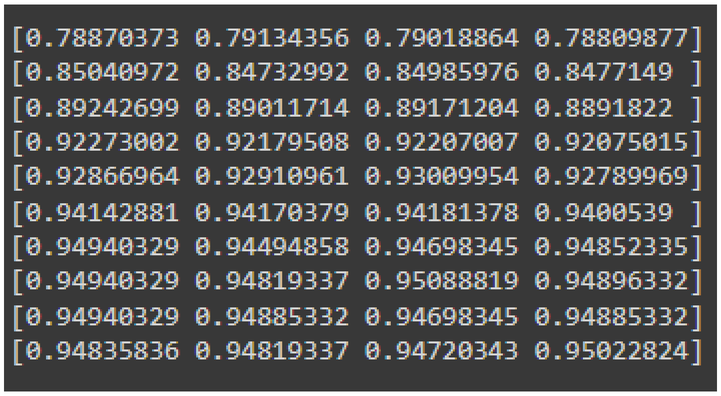
- 10 best features chosen :

```
Index 12 : Online boarding
Index 4 : Type of Travel
Index 7 : Inflight wifi service
Index 10 : Gate location
Index 19 : Inflight service
Index 2 : Customer Type
Index 5 : Class
Index 17 : Baggage handling
Index 14 : Inflight entertainment
Index 13 : Seat comfort
```

- **(1.4)** CV Scores for each of 10 parameters for cv=4 :
    1) SFS

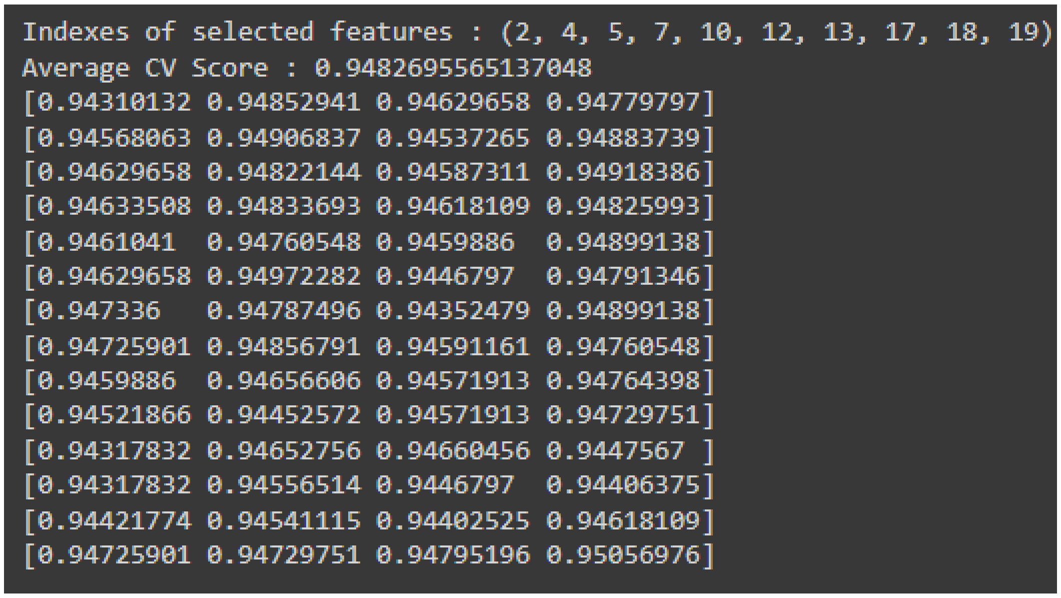


```
Indexes of selected features : (2, 4, 5, 7, 10, 12, 13, 14, 17, 19)
Average CV Score : 0.948495847769895
```



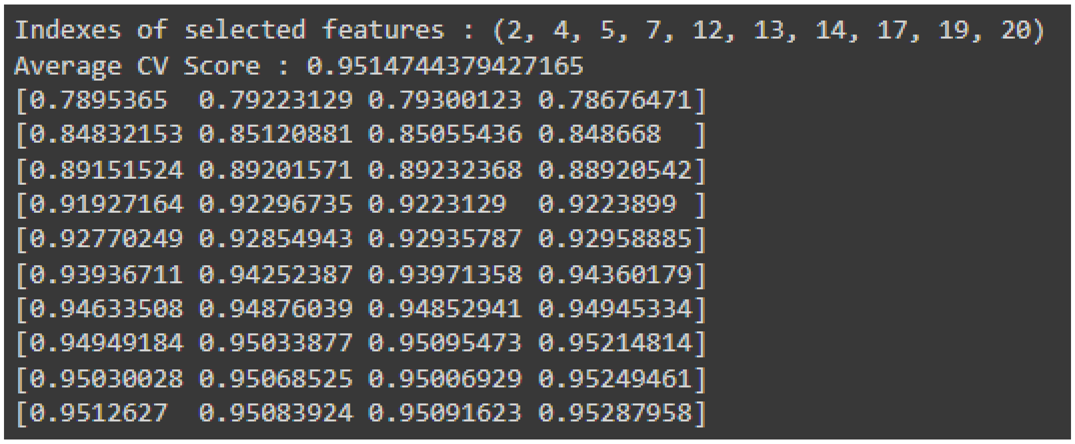
```
[0.78870373 0.79134356 0.79018864 0.78809877]
[0.85040972 0.84732992 0.84985976 0.8477149 ]
[0.89242699 0.89011714 0.89171204 0.8891822 ]
[0.92273002 0.92179508 0.92207007 0.92075015]
[0.92866964 0.92910961 0.93009954 0.92789969]
[0.94142881 0.94170379 0.94181378 0.9400539 ]
[0.94940329 0.94494858 0.94698345 0.94852335]
[0.94940329 0.94819337 0.95088819 0.94896332]
[0.94940329 0.94885332 0.94698345 0.94885332]
[0.94835836 0.94819337 0.94720343 0.95022824]
```

    2) SBS



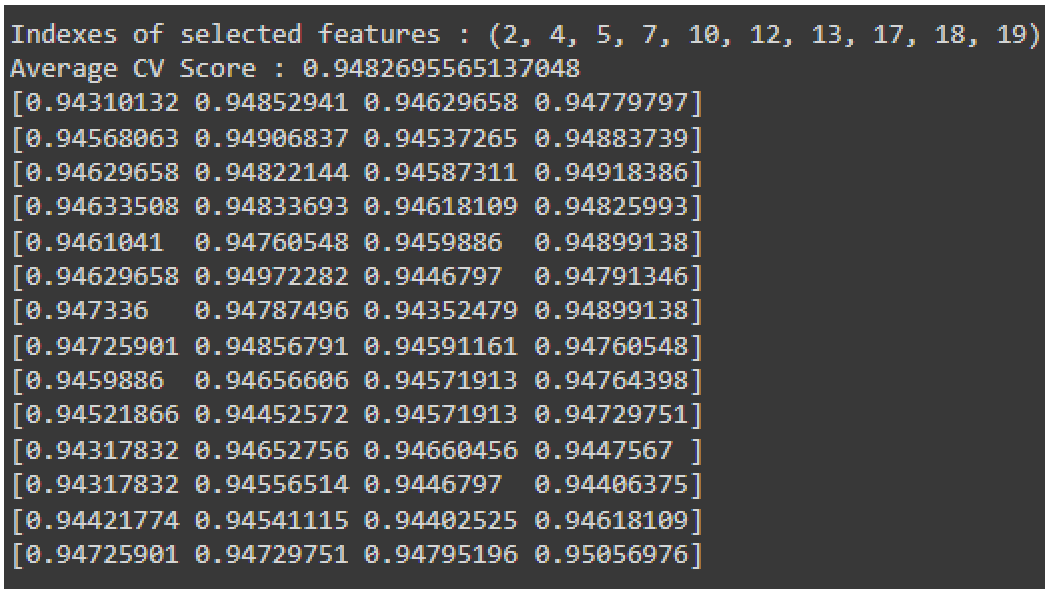
```
Indexes of selected features : (2, 4, 5, 7, 10, 12, 13, 17, 18, 19)
Average CV Score : 0.9482695565137048
[0.94310132 0.94852941 0.94629658 0.94779797]
[0.94568063 0.94906837 0.94537265 0.94883739]
[0.94629658 0.94822144 0.94587311 0.94918386]
[0.94633508 0.94833693 0.94618109 0.94825993]
[0.9461041  0.94760548 0.9459886  0.94899138]
[0.94629658 0.94972282 0.9446797  0.94791346]
[0.947336   0.94787496 0.94352479 0.94899138]
[0.94725901 0.94856791 0.94591161 0.94760548]
[0.9459886  0.94656606 0.94571913 0.94764398]
[0.94521866 0.94452572 0.94571913 0.94729751]
[0.94317832 0.94652756 0.94660456 0.9447567 ]
[0.94317832 0.94556514 0.9446797  0.94406375]
[0.94421774 0.94541115 0.94402525 0.94618109]
[0.94725901 0.94729751 0.94795196 0.95056976]
```

    3) SFFS



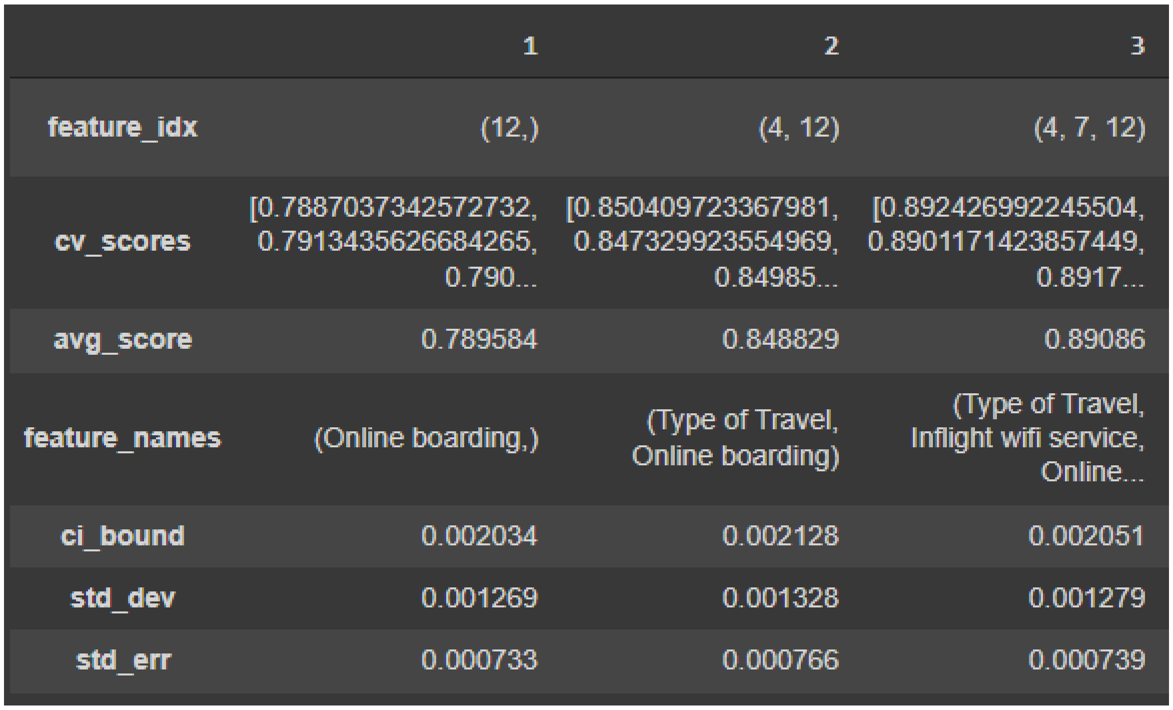
```
Indexes of selected features : (2, 4, 5, 7, 12, 13, 14, 17, 19, 20)
Average CV Score : 0.9514744379427165
[0.7895365  0.79223129 0.79300123 0.78676471]
[0.84832153 0.85120881 0.85055436 0.848668  ]
[0.89151524 0.89201571 0.89232368 0.88920542]
[0.91927164 0.92296735 0.9223129  0.9223899 ]
[0.92770249 0.92854943 0.92935787 0.92958885]
[0.93936711 0.94252387 0.93971358 0.94360179]
[0.94633508 0.94876039 0.94852941 0.94945334]
[0.94949184 0.95033877 0.95095473 0.95214814]
[0.95030028 0.95068525 0.95006929 0.95249461]
[0.9512627  0.95083924 0.95091623 0.95287958]
```

4) SBFS



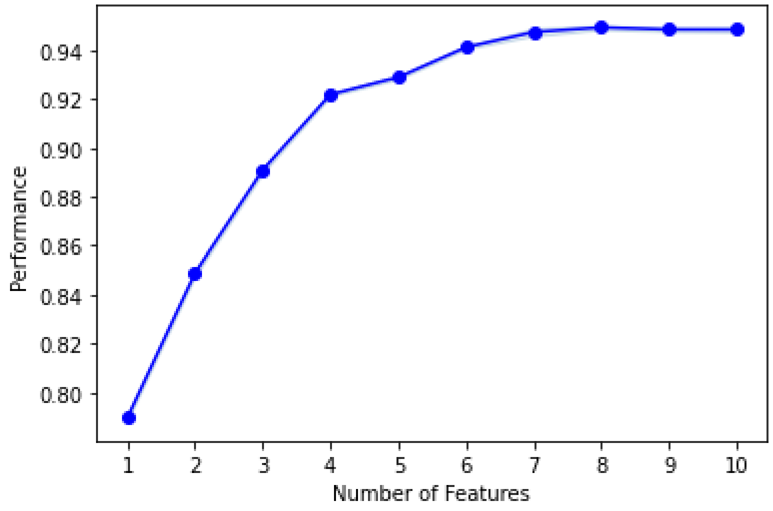
```
Indexes of selected features : (2, 4, 5, 7, 10, 12, 13, 17, 18, 19)
Average CV Score : 0.9482695565137048
[0.94310132 0.94852941 0.94629658 0.94779797]
[0.94568063 0.94906837 0.94537265 0.94883739]
[0.94629658 0.94822144 0.94587311 0.94918386]
[0.94633508 0.94833693 0.94618109 0.94825993]
[0.9461041  0.94760548 0.9459886  0.94899138]
[0.94629658 0.94972282 0.9446797  0.94791346]
[0.947336   0.94787496 0.94352479 0.94899138]
[0.94725901 0.94856791 0.94591161 0.94760548]
[0.9459886  0.94656606 0.94571913 0.94764398]
[0.94521866 0.94452572 0.94571913 0.94729751]
[0.94317832 0.94652756 0.94660456 0.9447567 ]
[0.94317832 0.94556514 0.9446797  0.94406375]
[0.94421774 0.94541115 0.94402525 0.94618109]
[0.94725901 0.94729751 0.94795196 0.95056976]
```

● **(1.5)** Output was visualized for all four models as follows (3 out of 10 features selected is shown here for SFS) -

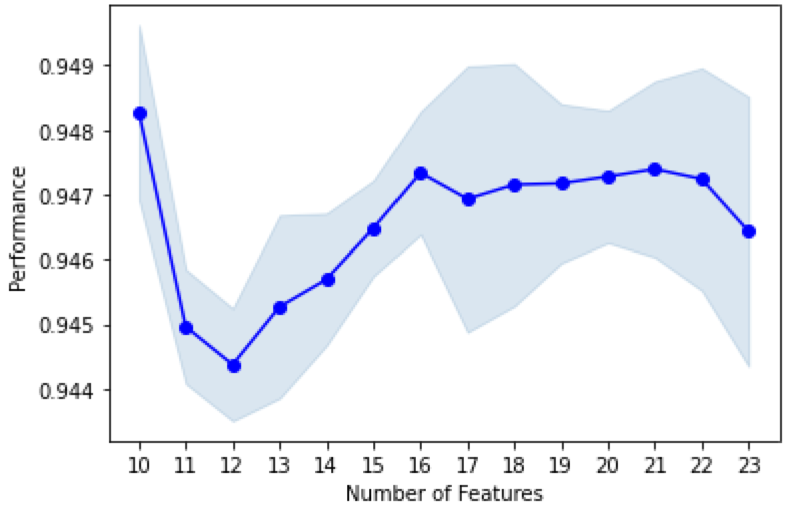


	1	2	3
feature_idx	(12,)	(4, 12)	(4, 7, 12)
cv_scores	[0.7887037342572732, 0.7913435626684265, 0.790...	[0.850409723367981, 0.847329923554969, 0.84985...	[0.892426992245504, 0.8901171423857449, 0.8917...
avg_score	0.789584	0.848829	0.89086
feature_names	(Online boarding,)	(Type of Travel, Online boarding)	(Type of Travel, Inflight wifi service, Online...
ci_bound	0.002034	0.002128	0.002051
std_dev	0.001269	0.001328	0.001279
std_err	0.000733	0.000766	0.000739

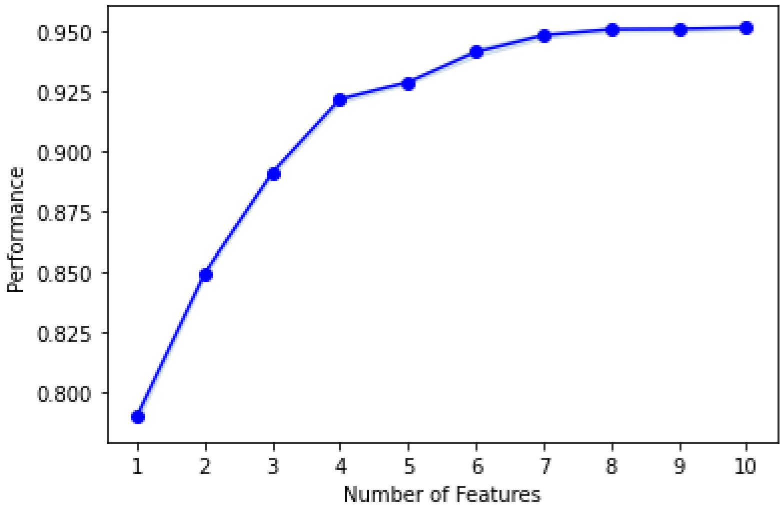
- **(1.6)** Output plots -
    1) SFS



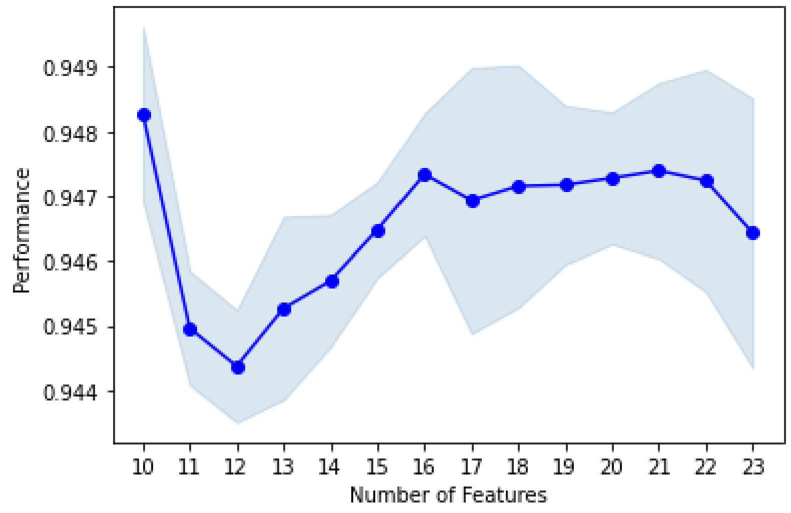
    2) SBS



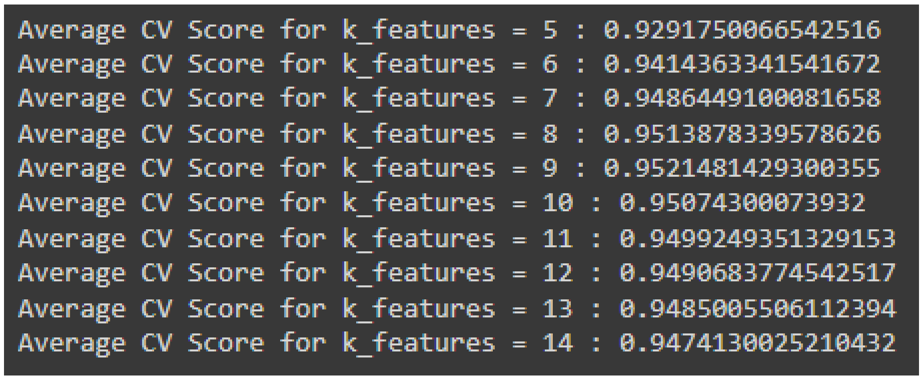
    3) SFFS



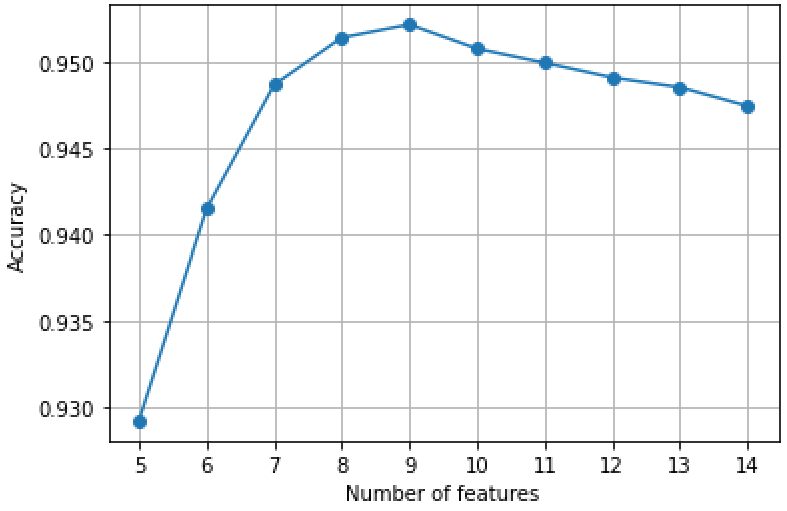
4) SBFS



- **(1.6)** Changing the number of features, we get -



```
Average CV Score for k_features = 5 : 0.9291750066542516
Average CV Score for k_features = 6 : 0.9414363341541672
Average CV Score for k_features = 7 : 0.9486449100081658
Average CV Score for k_features = 8 : 0.9513878339578626
Average CV Score for k_features = 9 : 0.9521481429300355
Average CV Score for k_features = 10 : 0.95074300073932
Average CV Score for k_features = 11 : 0.9499249351329153
Average CV Score for k_features = 12 : 0.9490683774542517
Average CV Score for k_features = 13 : 0.9485005506112394
Average CV Score for k_features = 14 : 0.9474130025210432
```



- We see that we get **maximum accuracy** when the **number of features is taken as 9** and accuracy **decreases on either side**.