Pattern Recognition and Machine Learning

(Winter 2022)

Assignment 8: ICA & Sequential Feature Selector

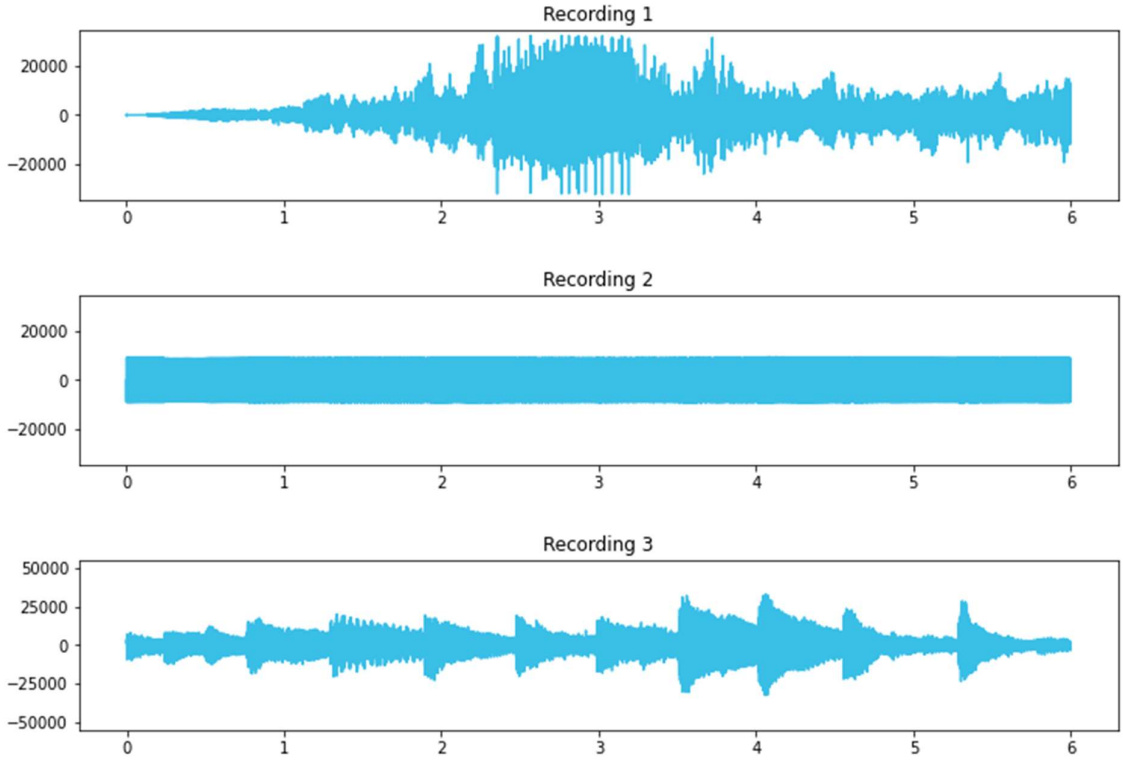
Report

Question - 1:

Task - 1: Read, Visualize and Listen the audio files.

I first used the function wave.open() that helps open a file to read/write audio data. Then, I first converted the audio signal to strings that represent bytes using wave.readframes() function. Then, I converted these strings to a 1-D array initialized from the text data in the string received before.

Then, I plotted the graphs to visualize these audio signals. I got the graphs for three signals as:



Finally, I listened the three audios using the Ipython.display.audio() function.

<u>Task - 2:</u> Extract raw audio from the three wave files and merge them to create dataset X.

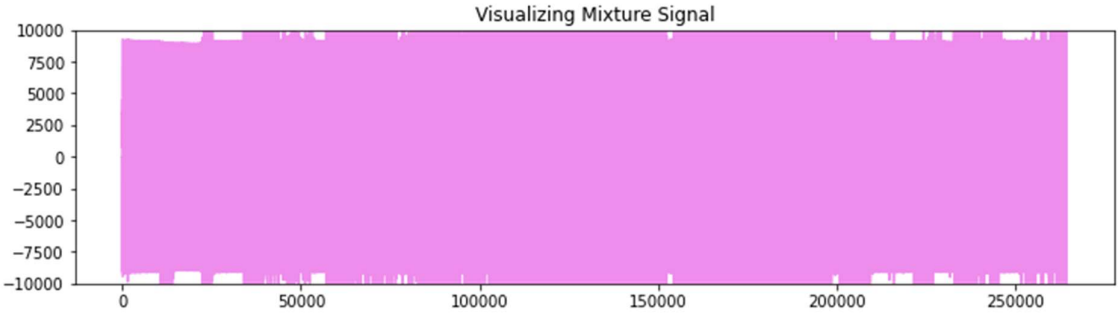
I took the extracted audio of the three waves from the previous task and zipped them to form a list of three waves.

I also plotted the mixture signal using this list.

Then, I used this formed list to form a dataset.

```
1 X.head()
```

	signal1	signal2	signal3
0	-23	-52	2142
1	-16	-624	2328
2	-28	-1178	2010
3	-11	-1740	1989
4	-25	-2282	1671



<u>Task - 3:</u> Implement ICA from scratch. For Convergence select either 1000 iterations, or when the dot product of w (demixing matrix) and its transpose is roughly equal to 1.

For this task, I kept the points discussed in the lab session while implementing the ICA from scratch i.e., at a high-level ICA can be broken down into following steps:

1) Centre x by subtracting the mean.

2) Whiten x.

3) Choose a random initial value for the de-mixing matrix w.

4) Calculate the new value for w.

5) Normalize w.

6) Check whether algorithm has converged and if it hasn't, return to step 4.

7) Take the dot product of w and x to get the independent source signals.

So, following these points I first implemented the function that helps centre the x (dataset) by subtracting the mean so as to make x a zero-mean variable.

Then, I implemented the function whiten() that helps to whiten the matrix i.e., to transform it in such a way that potential correlations between its components are

removed (covariance equal to 0) and the variance of each component is equal to 1. So, I calculated the covariance matrix first then calculated its eigenvalues and eigenvectors. Then, substituted the values into the formula;

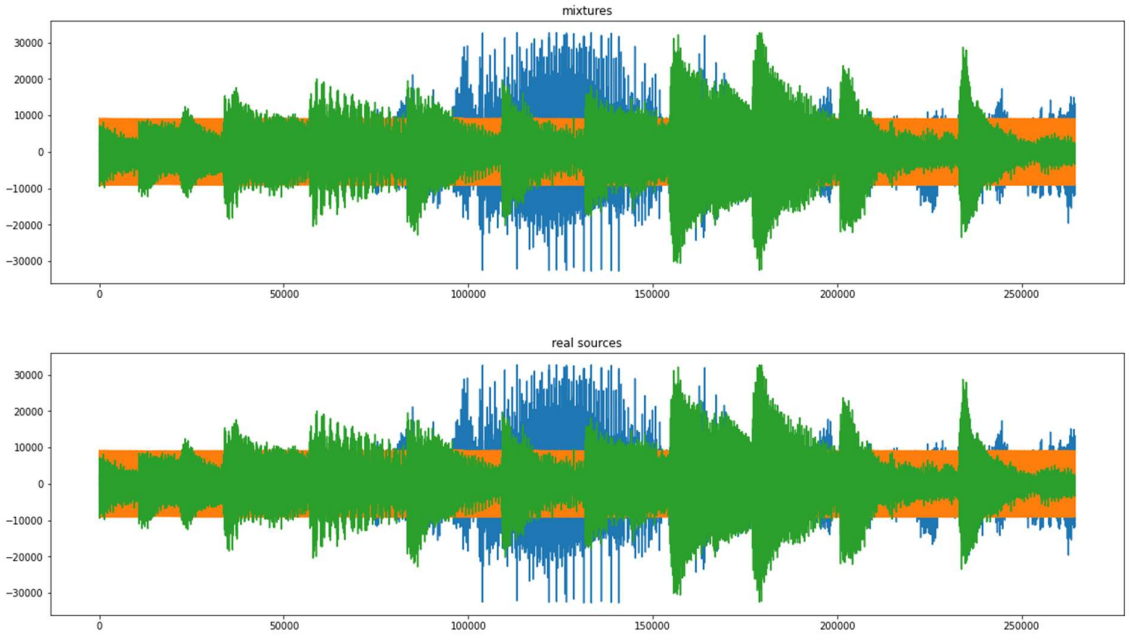
$$\tilde{x} = ED^{-1/2}E^{T}x$$

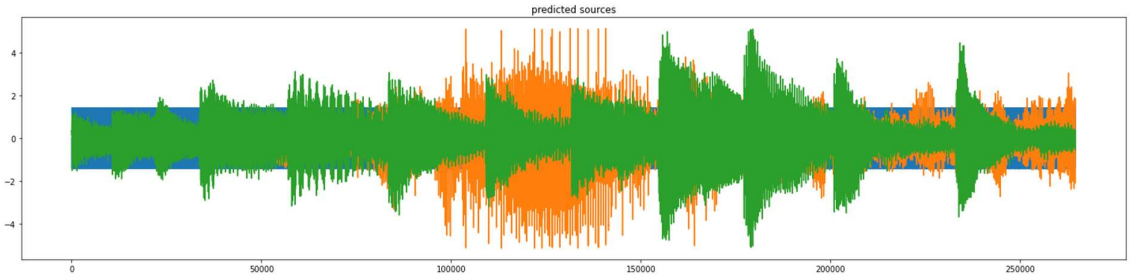
Then, I implemented a function "calculateNewW()" that updates the value of w (de-mixing matrix) until the ICA algorithm is converged or the maximum number of iterations has been reached. So, by this function I returned updated normalized de-mixing matrix.

Finally, I implemented the function "ica()" that calls all the function implemented above i.e., centre(), whiten(), calculateNewW(). In this function we calculate the de-mixing matrix in the range of number of iterations or when the dot product of w and its transpose is roughly equal to 1. And we repeat this process for each component of the matrix.

Task - 4: Plot mixture, real source and predicted source from the output of ICA.

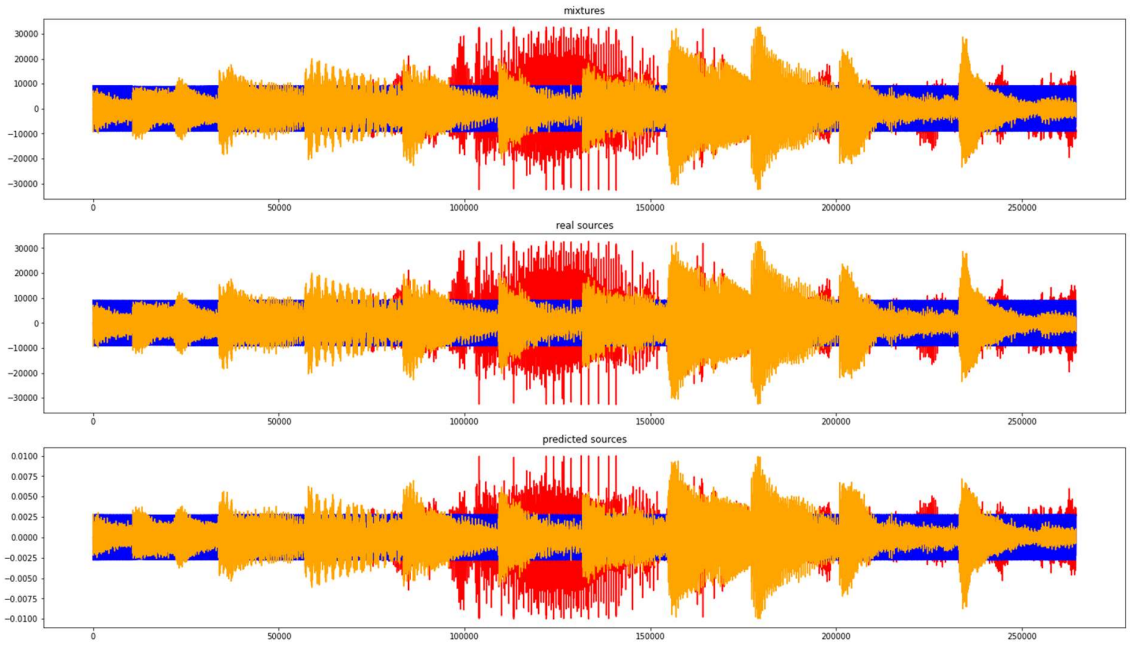
The plot that I got for the mixture, real source and predicted source from the output of ICA are:





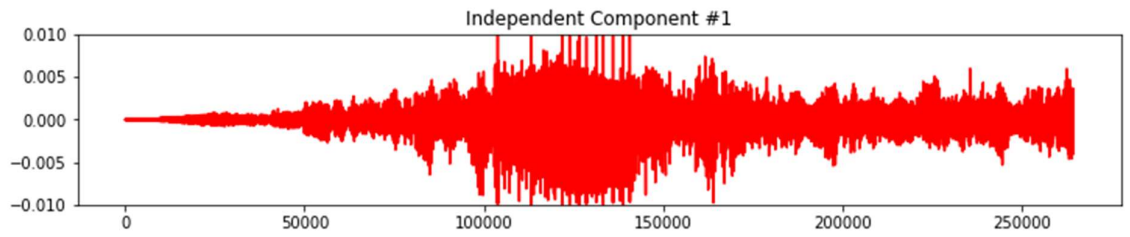
<u>Task - 5:</u> Implement Fast ICA (import from sklearn.decomposition) selecting num_components = 3.

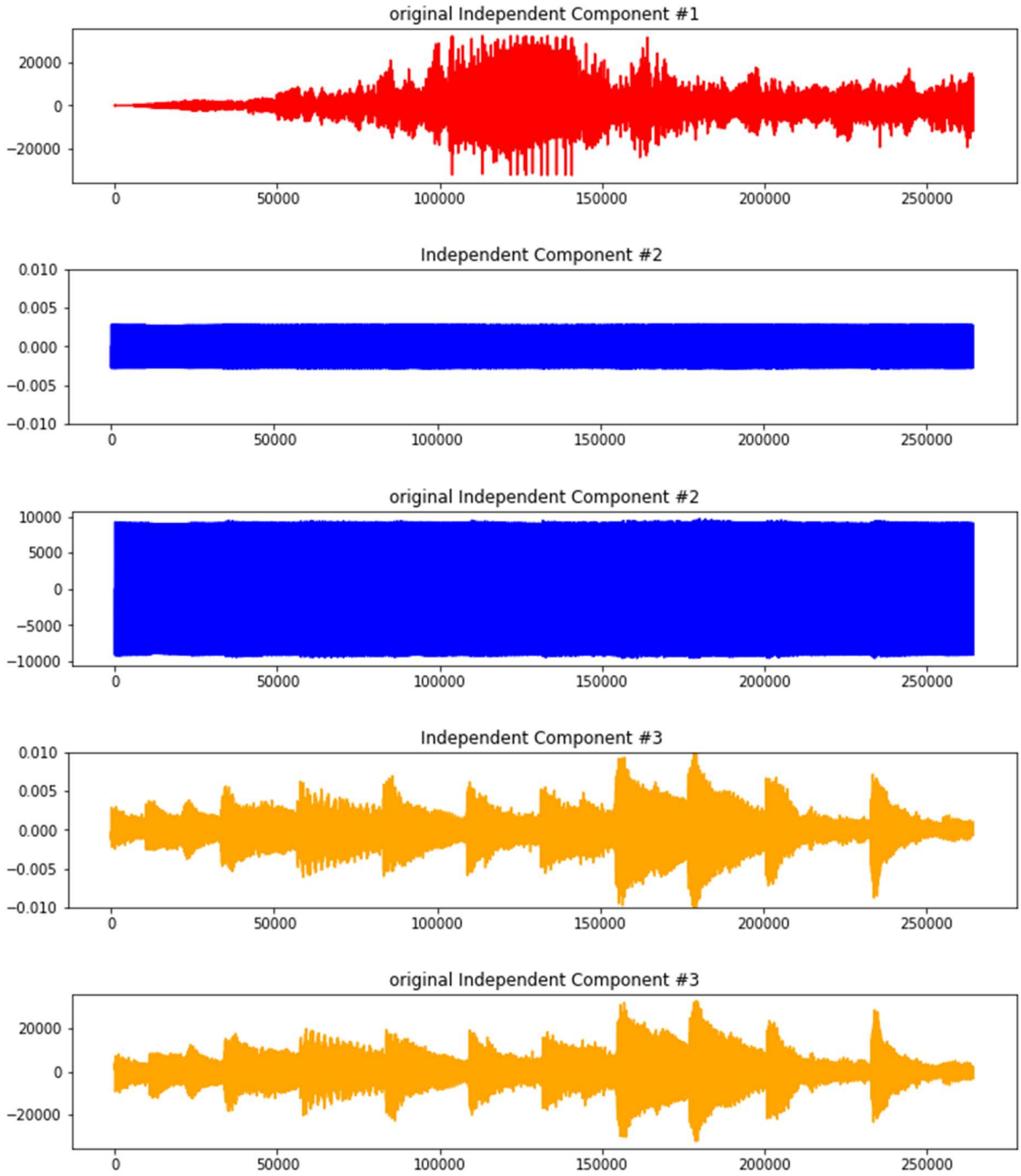
For this task, I implemented fast ICA for that I imported the FastICA from the sklearn.decomposition library and first instantiated the FastICA and passed the hyperparameter num_components = 3 as given in the task.



<u>Task - 6:</u> Separate, Visualize and Listen the independent component obtained from task 5.

Then, I separated the three components from the result I got from Fast ICA and stored them, Then, I plotted the graphs to visualize the mixture, real source and predicted source by the Fast ICA implementation. I also plotted the independent components that I got after applying the Fast ICA. The plots that I got are:





Then, finally I listened to audio that I got after applying Fast ICA. On my observation the audio was similar to the original/real audio sources.

<u>Task - 7:</u> Comment on the results obtained from ICA and Fast ICA and report observations in the report.

The result obtained from ICA and Fast ICA are similar. There are some instances where fluctuations are there in the predicted sources as compared to real sources. Also, there are some instances where Fast ICA performed better than ICA as in ICA some sources were predicted as upside down. Overall, predicted sources were similar to the real sources in both the cases.

Question - 2:

Task - 1: Pre-process, clean and prepare the dataset based on the previous lab experience. Separate features and labels as X and Y respectively.

For this task, I first took the input of the training dataset. I checked for any nan/missing values in the dataset. I found that there are 310 missing values in the "Arrival Delay in Minutes" feature. So, I dropped all the rows containing the nan values. The number of rows before and after dropping the rows are:

```
Arrival Delay in Minutes          310
```

```
The original number Of rows in dataset:  103904
No Of rows in Dataset after droping NAN values:  103594
```

Then, I label encoded all the categorical column present in the dataset. Hence, I separated the features and label as X and Y.

I applied these pre-processing steps on testing dataset as well.

Task - 2: Create an object of SFS by embedding Decision Tree classifier object, providing 10 features, forward as True, floating as False and scoring = accuracy.

First, I imported the DecisionTreeClassifier from the sklearn.tree library and instantiated the Decision Tree Classifier and set the random state as 0.

Then, I imported the SequentialFeatureSelector from the mlxtend.feature_selection of mlxtend library and first instantiated the SequentialFeatureSelector and passed the hyperparameters as given in the task i.e., k_features = 10, forward = True, floating = False, scoring = 'accuracy'.

Task - 3: Train SFS and report accuracy for all 10 features. Also list the names of the 10 best features selected by SFS.

I trained the SFS using the training data. I reported the accuracies for all the 10 best features, I reported the accuracies for the 10 features and also listed those 10 best features and return their indexes.

The 10 best features and their accuracy that I got is:

```
10: {'avg_score': 0.9506631738158859,
 'cv_scores': array([0.95009412, 0.95062503, 0.94898402, 0.95212124, 0.95149146]),
 'feature_idx': (3, 5, 6, 8, 11, 13, 14, 15, 18, 20),
 'feature_names': ('Customer Type',
  'Type of Travel',
  'Class',
  'Inflight wifi service',
  'Gate location',
  'Online boarding',
  'Seat comfort',
  'Inflight entertainment',
  'Baggage handling',
  'Inflight service')}}
```

<u>Task - 4:</u> Using the forward and Floating parameter toggle between SFS(forward True, floating False), SBS (forward False, floating False), SFFS (forward True, floating True), SBFS (forward False, floating True), and choose cross validation = 4 for each configuration. Also report cv scores for each configuration.

For this task, I trained 4 different SequentialFeatureSelector by setting the hyperparameter as given in this task.

For 1 model SFS (forward True, floating False)

```
(3, 5, 6, 8, 11, 13, 14, 15, 18, 20)
CV Score:
0.9499198910655715
```

For 2 model SBS (forward False, floating False)

```
(3, 5, 6, 8, 13, 14, 15, 18, 20, 21)
CV Score:
0.9514257638213464
```

For 3 model SFFS (forward True, floating True)

```
(3, 5, 6, 8, 13, 14, 15, 18, 20, 21)
CV Score:
0.9514257638213464
```

For 4 model SBFS (forward False, floating True)

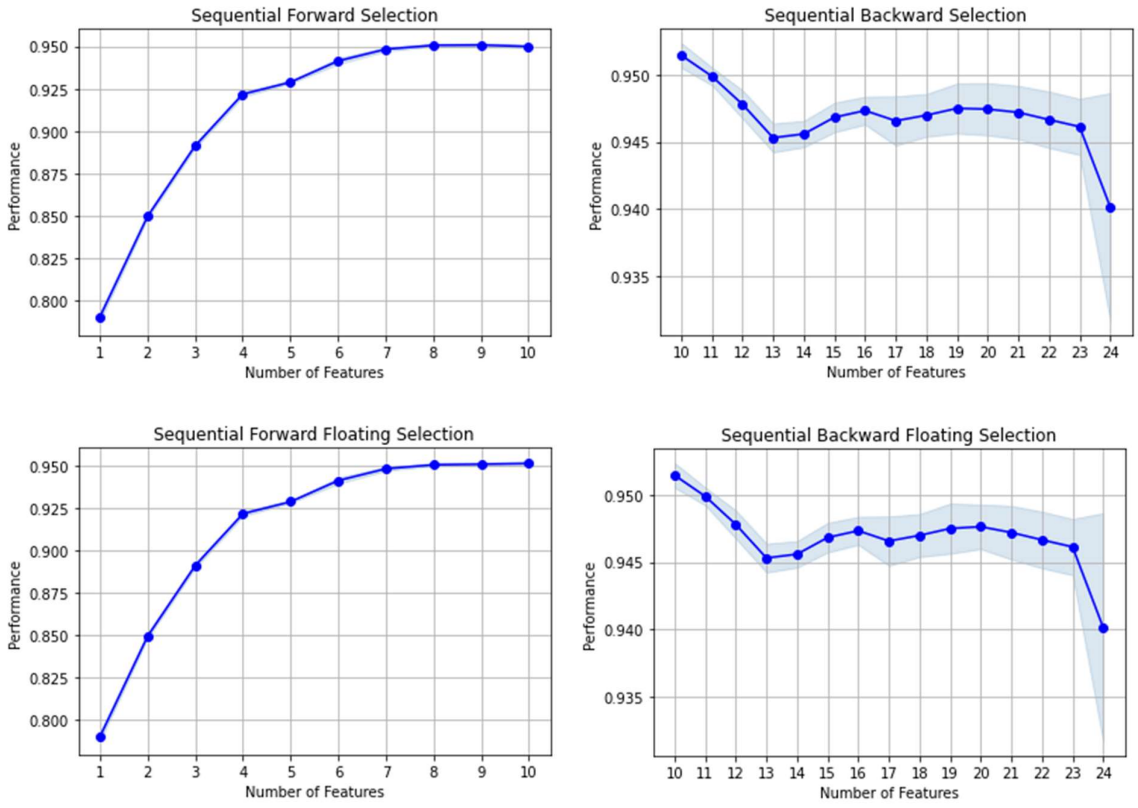
```
(3, 5, 6, 8, 13, 14, 15, 18, 20, 21)
CV Score:
0.9514257638213464
```

<u>Task - 5:</u> Visualize the output from the feature selection in a pandas DataFrame format using the get_metric_dict for all four configurations.

For visualizing the output from the feature selection in a pandas DataFrame format, I used the get_metric_dict for all the four configurations and reported the DataFrame for each configuration in my colab file.

<u>Task - 6:</u> Finally plot the results for each configuration (from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs).

For this task, I plotted the result for each configuration by using the imported plot_sequential_feature_selection from mlxtend.plotting library. The plots that I got for each configuration are:



<u>Task - 7:</u> For task 2 vary the features by increasing or decreasing, observe and report the results.

For this task, I varied the features value from 1 to 23 and I calculated the average scores, cv_scores and hence evaluated the best number of features for the given SequentialFeatureSelector model. I also plotted the graph to visualize that how the performance varies on increasing and decreasing the number of features. From the plot it is visible that for number of features = 9 performance was best. It also signifies that for a smaller number of features there will be less computation. Hence, for 9 features it performs best. The plot that I got is: