I. Introduction and Objectives of the Experiment

The objectives of the assignment, in brief is to design 2 classification problems, experiment with multiple Machine Learning algorithms listed, and generate classification Models. Also, explain, how the model performance varies with hyper parameters and how do we choose the Best Model among all.

II. Datasets

For this Assignment, I have used the following two Datasets:

1. Wine Quality (Link); also referred to as Dataset 1 later.
   Description: The features for this Dataset are related to the features of the wine for. E.g. Acidity, residual sugar, density, pH, sulphates, etc. The task is to classify – given the features / attributes of the wine, the quality of wine is either a 'White' wine or 'Red'.
2. Adult Income Dataset (Link) ; also referred to as Dataset 2 later.
   Description: The features of this Dataset contain information related to US Individuals like Education, Age, Marital-status, Occupation, etc. and a Target variable 'Yes' or 'No' – 'Yes' if income if greater than $50000, else 'No'

<u>What makes the Dataset interesting:</u>

I tried to look at the following indicators that can help measure the interesting-ness of a given dataset:

1. Presence of Categorical vs Numerical Attributes and the Dataset size.

	Number of Categorical Columns	**Number of Numerical Columns**	**Number of Rows**	**Type of Problem**
Wine Dataset	1	11	6497	Multi-class Classification
Adult Income Dataset	8	6	48842	Binary Classification

2. Balance of the Dataset:

   Wine Dataset:

Class A Proportion	**Class B Proportion**	**Class C Proportion**
60%	36.6%	3.4%

   Adult Income Dataset:

Class A	**Class B**
76%	24%

3. Relationship between the input variable and output variable – One of the factors considered here was, to check the relationship between an input feature and an output variable i.e. Linear vs Non-Linear Relation. The end goal was – for Dataset with Linear Relationships, see how the Linear Models perform as compared to Non-Linear Dataset.

III. Training Curves with changing Training sizes:

For this experiment, I varied the Training Dataset proportion starting from 10% till 90%. The training performance (F1-score here) has been captured as the training data varies, on a fixed test Dataset. The left figure indicates the curve for Dataset 1 and the right one for Dataset 2.

1. Decision Trees

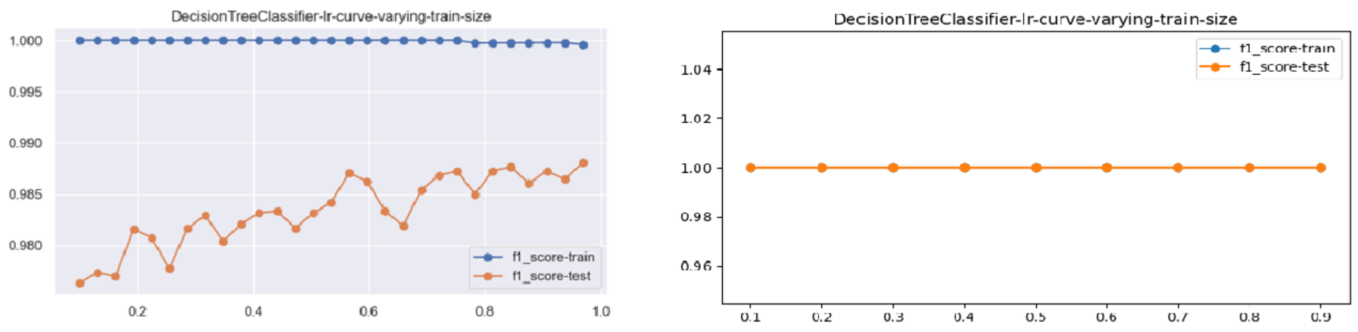


Fig 1: Decision Trees F1-score after varying Training Data size.

2. LinearSVC

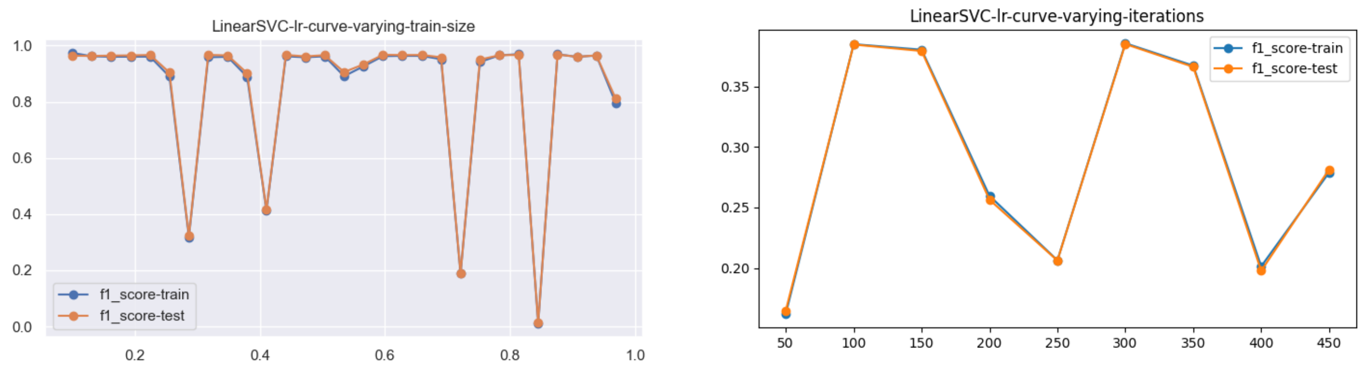


Fig 2: Linear SVC F1-score after varying Training Data size.

3. SVC (with rbf kernel)

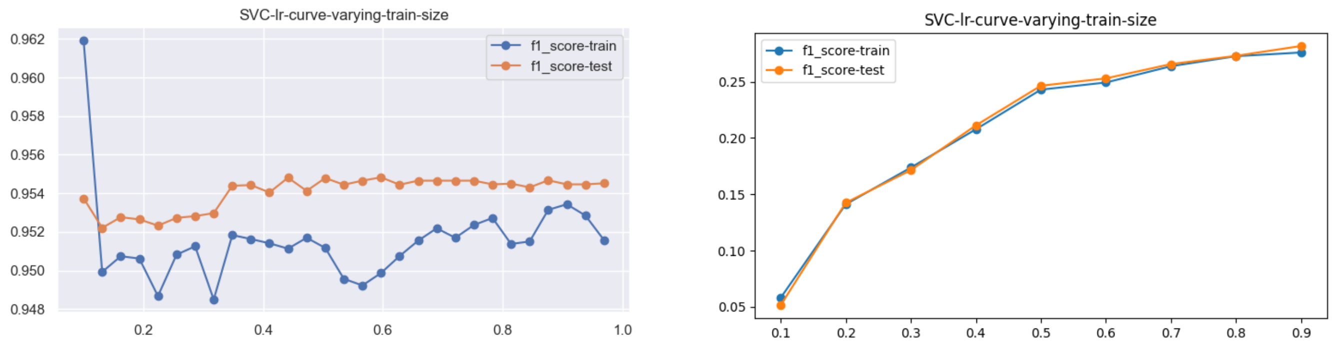


Fig 3: SVC (rbf) F1-score after varying Training Data size.

4. K-Nearest Neighbors

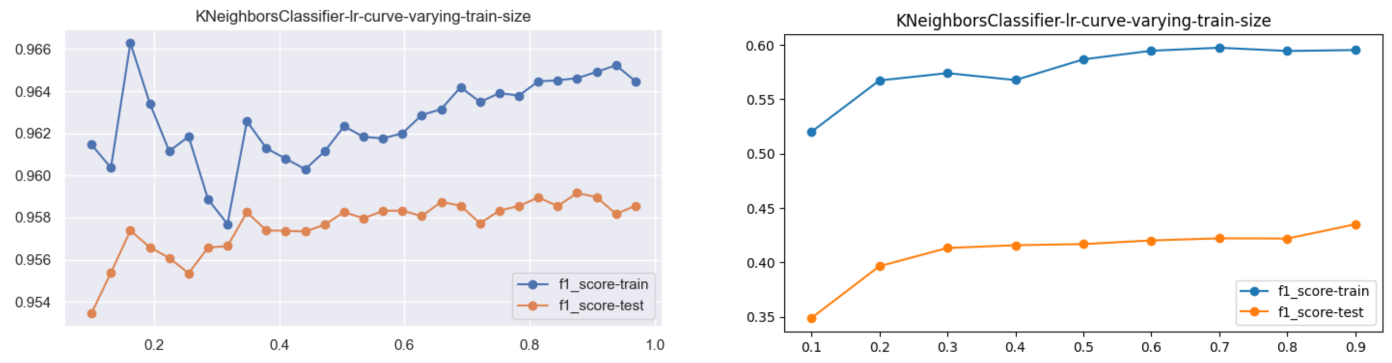


Fig 4: K-Nearest Neighbors F1-score after varying Training Data size.

5. Neural Networks

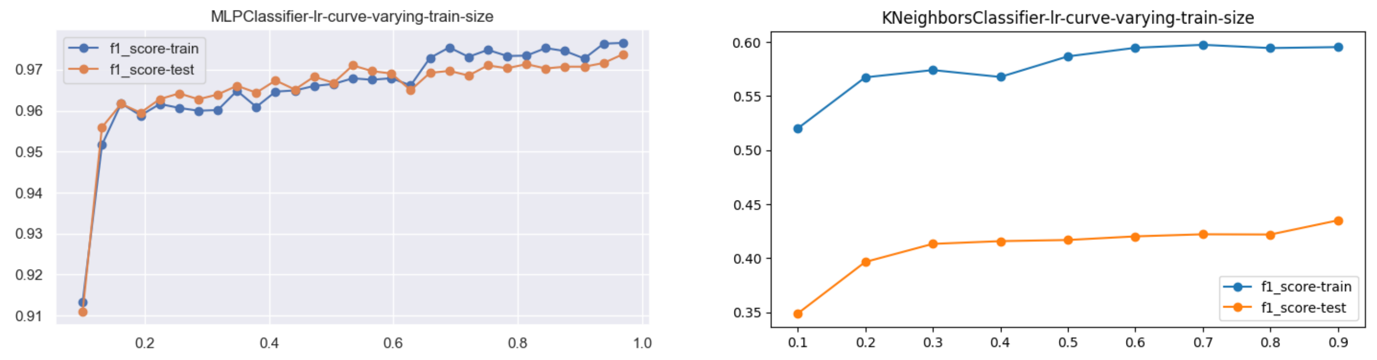


Fig 5: MLPClassifier F1-score after varying Training Data size.

6. AdaBoost Classifier

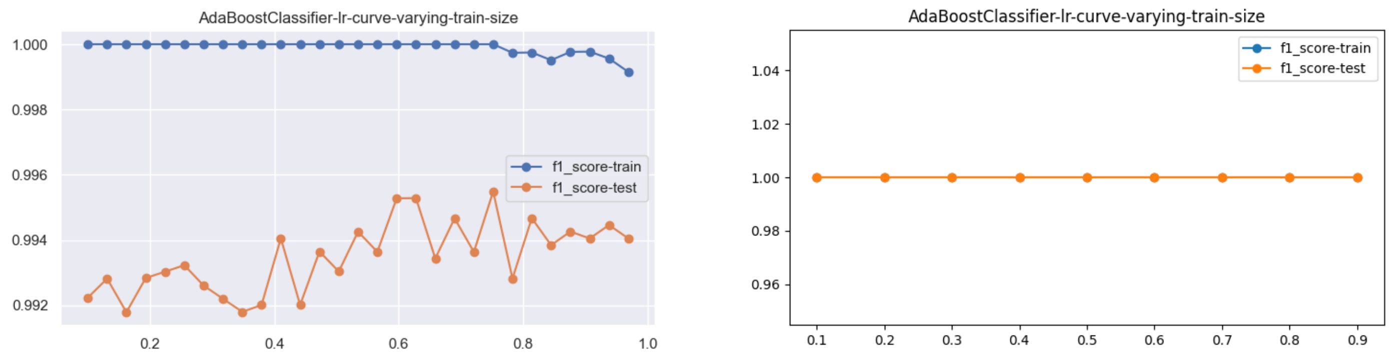


Fig 6: Adaboost Classifier F1-score after varying Training Data size

Observations from the Learning Curves:

1. As the Training Data increases, performance for some models increases. This is true for all the Models.
2. For Dataset 1 - For most of the Models, test performance starts at 90% F1-score at about 10 to 20% of the Training Data, indicating, the problem is not that complex.
3. For Dataset 2 – AdaBoostClassifier seems to outperform all the Models and can learn everything from the Data as compared to other models. The data was not scaled here, as we needed to understand the effect of the Training Size.

4. Models performing good on even lower proportions could mean that the classification boundary is easier to model, and models can do good job with fewer data.

IV. Learning Curves with Changing Number of iterations / estimators:

1. Adaboost – varying the number of estimators.

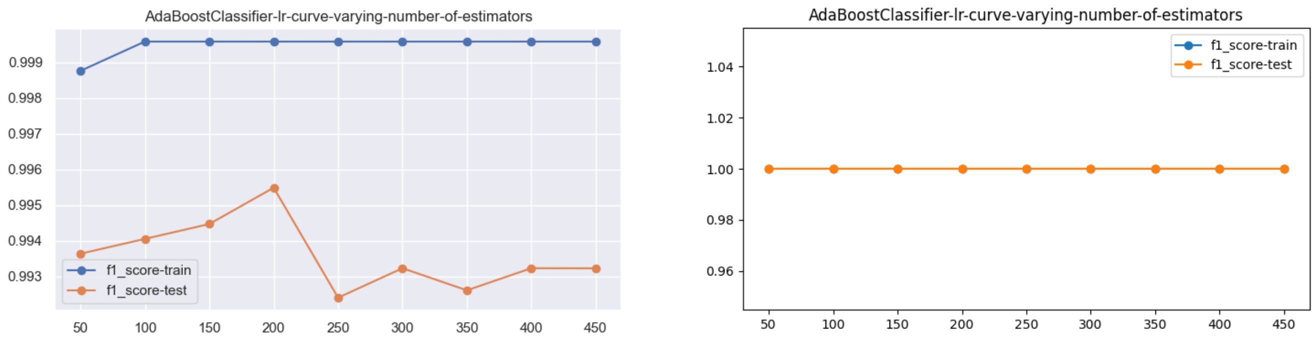


Fig 7: AdaBoost Classifier F1-score after varying number of estimators.

2. Neural Network – varying the number of hidden layers.

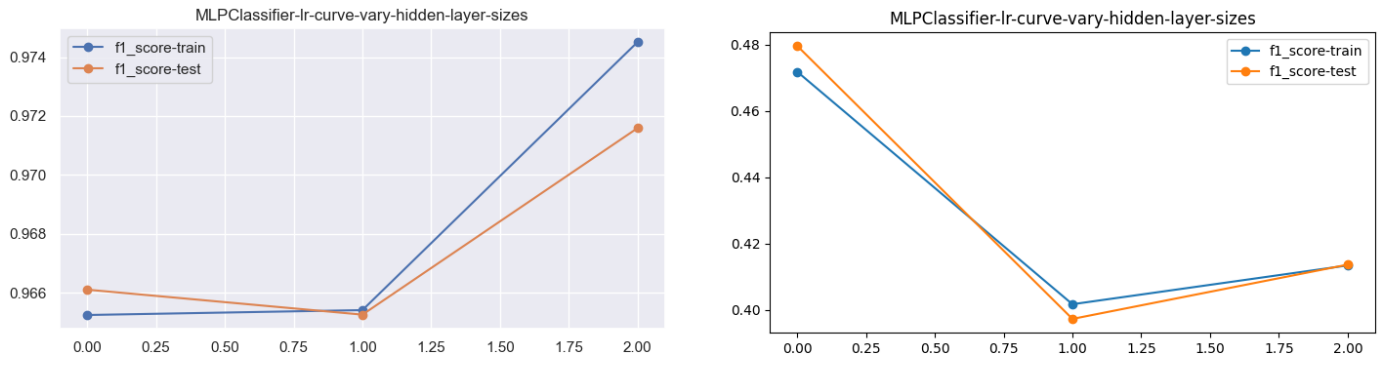


Fig 8: MLP Classifier F1-score after varying number of iterations.

3. LinearSVC – varying the number of iterations.

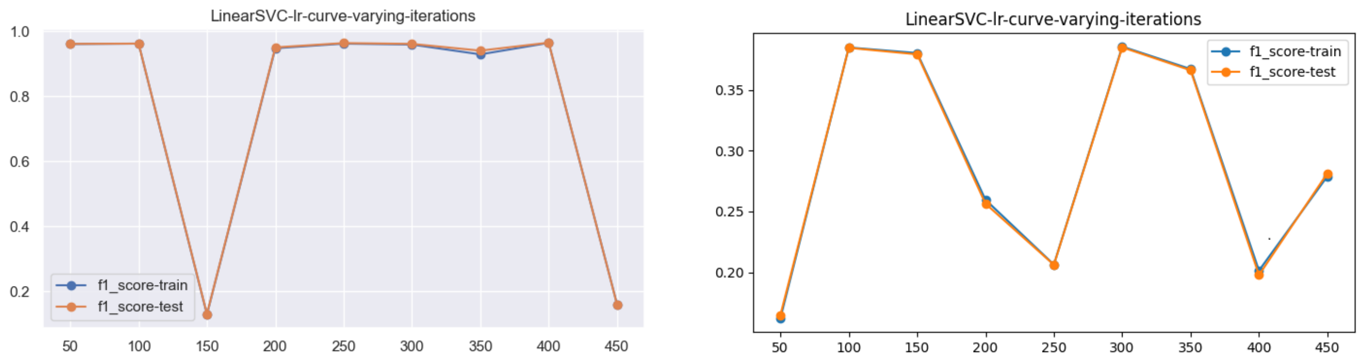
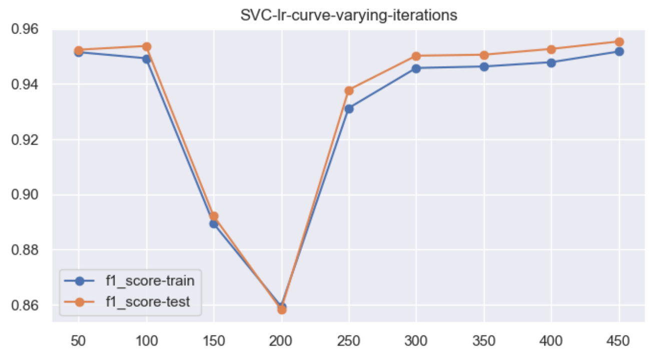


Fig 9: Linear SVC Classifier F1-score after varying number of iterations.

4. SVC – varying the number of iterations.



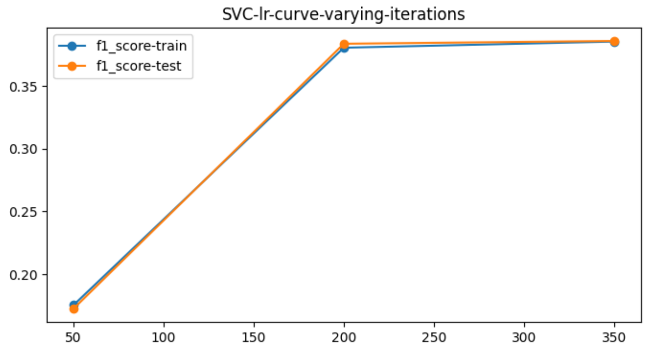


Fig 10: SVC (rbf) Classifier F1-score after varying number of estimators.

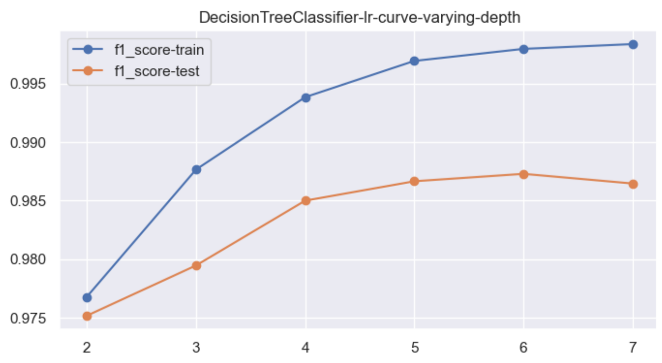
<u>Observations from the Learning Curves:</u>

1. In general, as we increase the number of Iterations or estimators, it seems that the performance keeps on increase.
   For e.g.:
   - SVC classifier in Fig 10 I both Dataset 1 and 2.
   - Neural Network in Fig 8 in Dataset 1
2. For few cases, the training performance decreases after certain number of Estimators / Iterations
   for e.g.
   - Adaboost (Fig 7) – Dataset 1 – the performance dips after crossing 150 estimators.
   - Linear SVC (Fig 9) – Dataset 2 – the performance dips after 350 iterations.
   - Neural Networks (Fig 8) – Dataset 2
   This could indicate that model is overfitting .

V. Graphs for changing selected Hyperparameter Ranges (Model Complexity Graph or Validation Curves)

Selected Hyperparameters that could provide a higher impact were varied and the curves were plotted to check the respective changes.

1. Decision Trees
   a. Max-depth



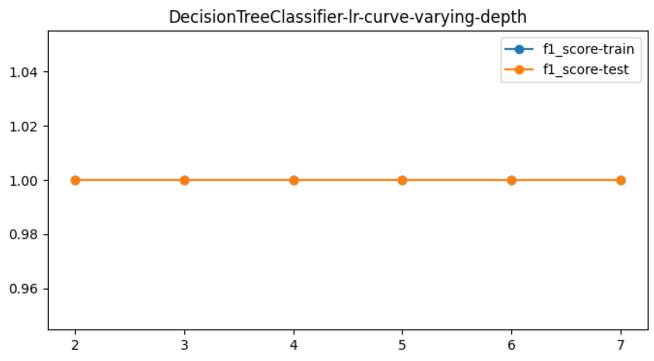


Fig 11: Decision Tree Classifier F1-score after varying Tree depth.

b. Max-Features

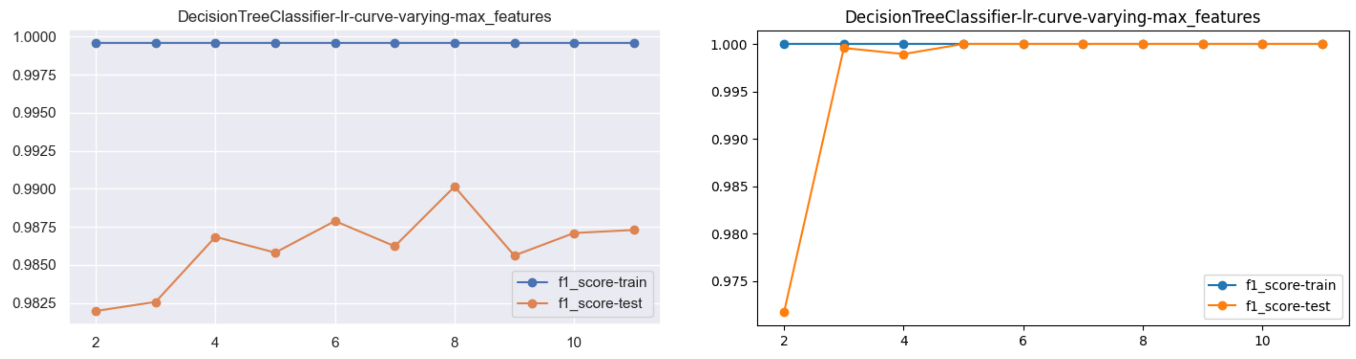


Fig 12: Decision Tree Classifier F1-score after varying max_features

c. Min-samples-leaf

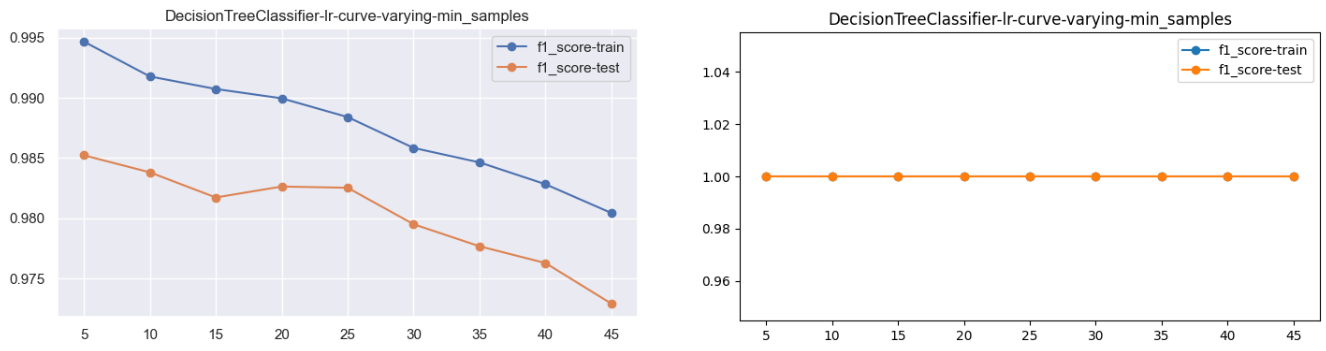


Fig 13: Decision Tree Classifier F1-score after varying min_samples

2. Linear SVC

a. Class_weight

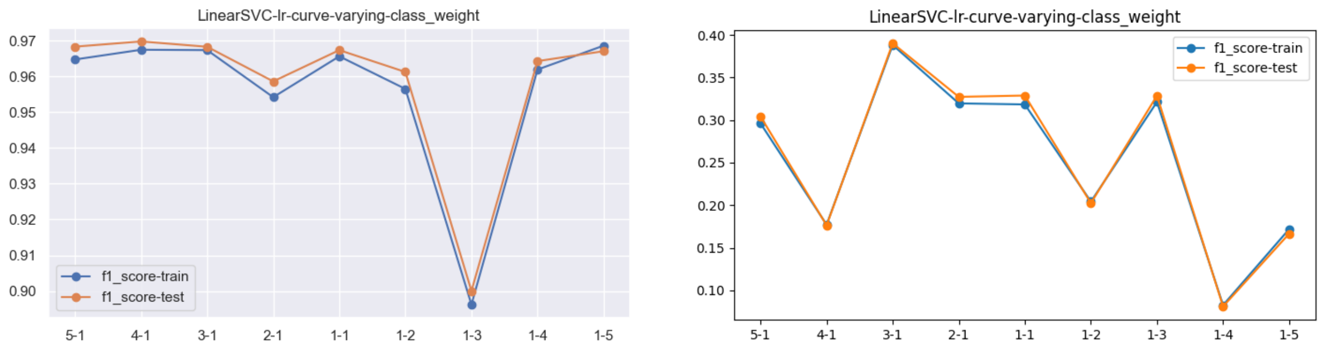
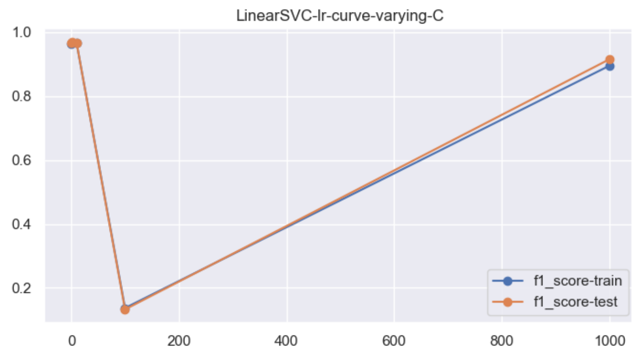


Fig 14: Linear SVC Classifier F1-score after varying class_weight

b. C



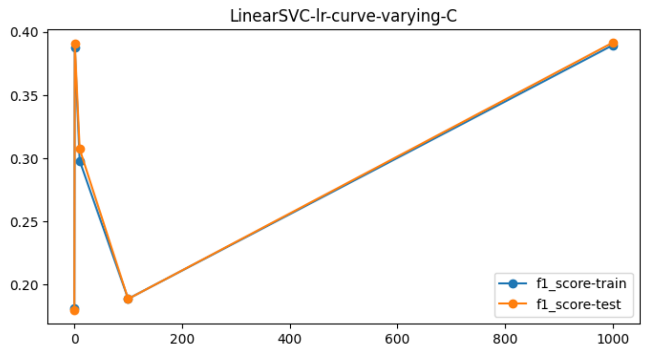
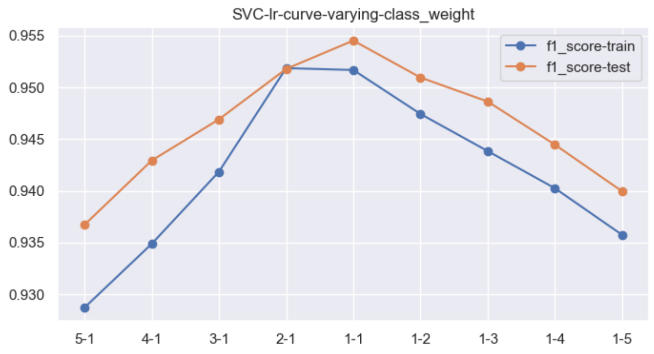


Fig 15: Linear SVC Classifier F1-score after varying C

3. SVC

a. Class_weight



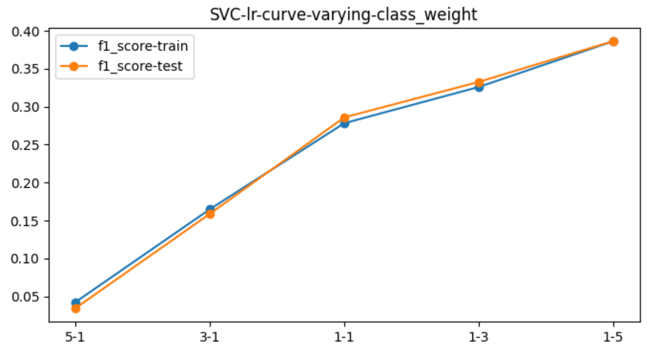
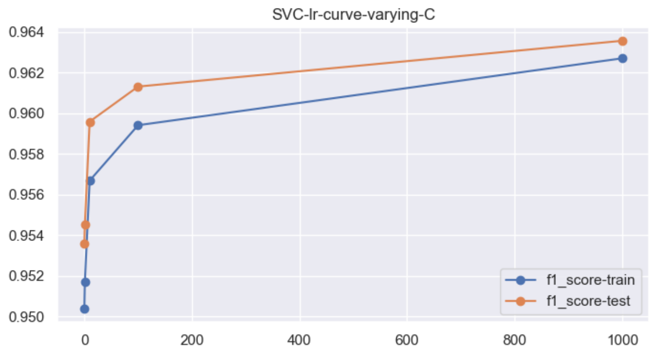


Fig 16: SVC (rbf) Classifier F1-score after varying class_weight

b. C



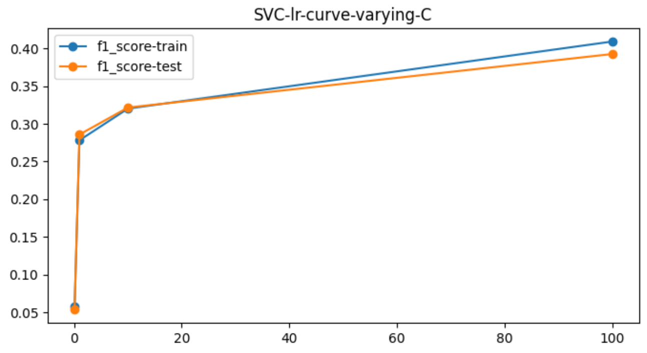


Fig 17: SVC (rbf) Classifier F1-score after varying max_features

4. K-Nearest-Neighbors

a. N_neighbors

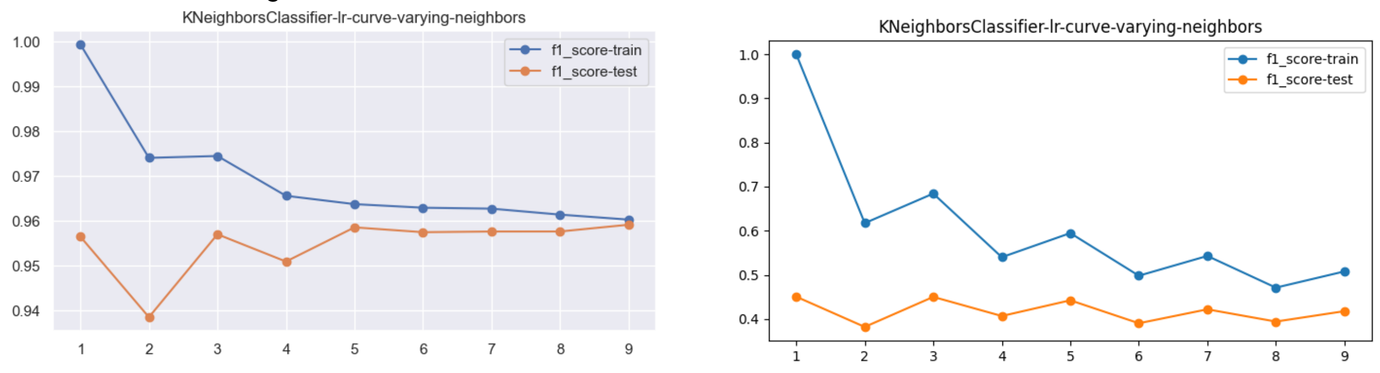


Fig 18: KNN Classifier F1-score after varying n_neighbors

b. Distance

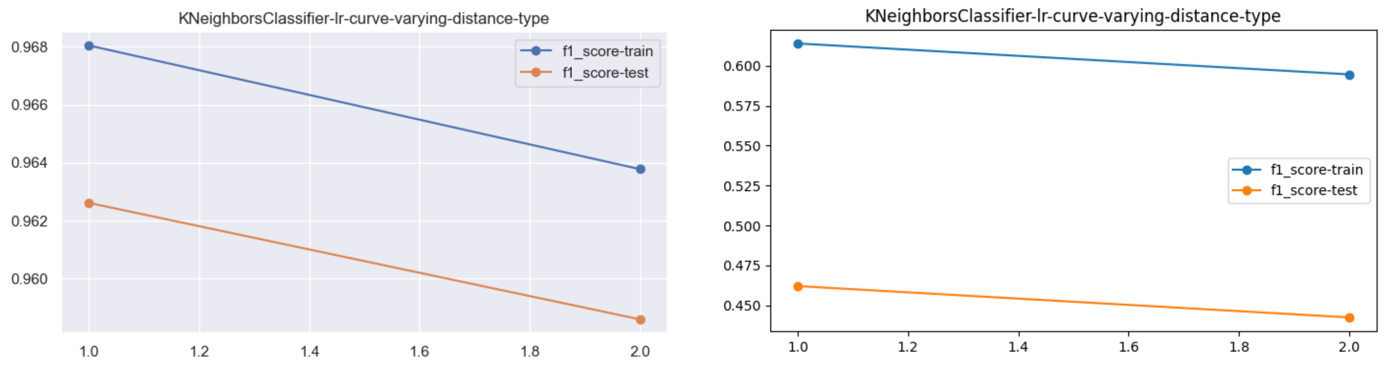


Fig 18: KNN Classifier F1-score after varying distance_type

5. Neural Networks

a. Activation

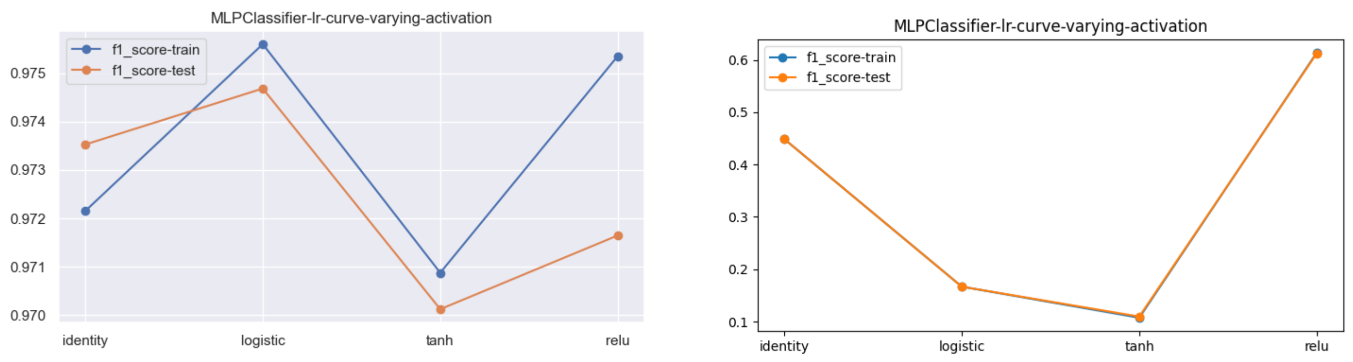
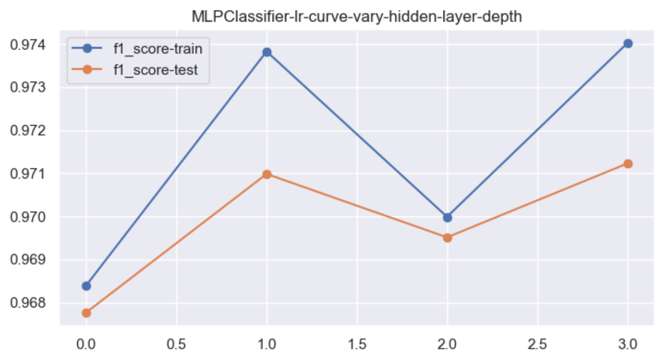


Fig 19: MLP Classifier F1-score after varying activation function.

b. Hidden-layer-depth



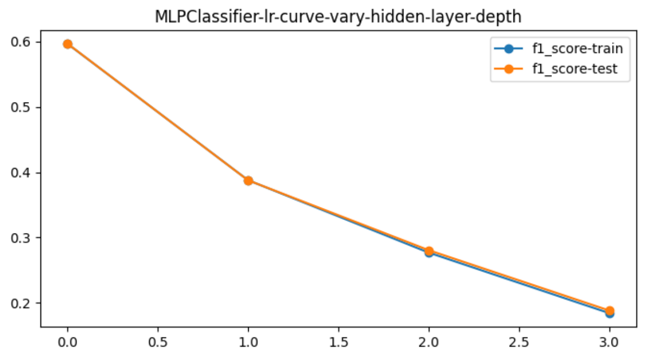


Fig 20: MLP Classifier F1-score after varying hidden layer Depth.

Observations from Hyperparameter-based experiments:

1. Few parameters provide a clear increase in performance as the parameter value is changed:
   For e.g.
   - Depth, max_features for Decision Trees (Fig 11)
   - Depth of hidden layers in Neural Networks (Fig 20) etc
   - Choice of activation function (relu) (Fig 19)
   - C in SVC Classifier (Fig 17)
2. Some of the parameters indicate that increasing the value makes the model overfit:
   For e.g.
   - N_neighbors in KNN (Fig 18) – lower k values indicate high overfitting
   - Max_features in DecisionTrees when max_features > 8
   - Class_weight in SVC (Fig 16)

VI. Wall-Clock Time for Models:

Model Type	**Dataset 1**		**Dataset 2**	
	Training Time	**Inference Time**	**Training Time**	**Inference Time**
Decision Trees	0.014	0.001	0.019	0.001
Linear SVC	0.06	0.001	2.16	0.002
SVC (rbf)	0.053	0.0019	13.17	2.061
K-Nearest Neighbors	0.006	0.069	0.126	0.24
Neural Networks	2.4	0.002	13.57	0.005
AdaBoost	0.15	0.016	0.036	0.002

VII. Test Performance of Models – result of Grid Search:

	Dataset 1	**Dataset 2**
Decision Trees	98.63%	54%
Linear SVC	99.50%	99.9%
SVC (rbf)	99.71%	99.9%
K-Nearest Neighbors	99.54%	97.28%
Neural Networks	99.65%	98.55%
AdaBoost	99.2%	99.9%

VIII. Conclusions and Observations based on all the Data above:

1. In general, all the times for Dataset 1 are relatively very low as compared to the times in Dataset 2. This comes from the fact that Dataset 2 is very high in terms of its size.
2. Time-wise performance:
   a. Models like Decision Trees, and AdaBoost are very quick in terms of Training and Inference.
   b. For Models like Neural Networks, and SVC – Linear and Non-Linear – the training time and the inference times are very high.
   c. For K-Nearest Neighbors, the inference time is very high as compared to its training time.
3. Metric-wise performance
   a. Choice of Metric – Why F1-score: Given that both the Datasets were highly imbalanced, F1-score gives a balanced performance w.r.t all the Target classes. However, the choice depends on the type of the Problem. For.eg. in a prediction problem if we are detecting Cancer or any other disease – we cannot afford to give any True Negatives. In such cases, we will try to optimize for Recall.
4. Best-Model
   a. How to choose Best-Model: Best Model depends on the following criteria:
      i. Best Wall-clock times
      ii. Best Metric (F1-score here)
      iii. Interpretability
   b. The choice of the best model depends on which situation, the model is going to be used.
   c. In terms of Best Clock times – Decision Trees and Adaboost are better – Adaboost better than Decision Trees as it has a better metric as well as compared to Decision Trees overall
   d. In terms of Interpretability – Models like Decision Trees and K-Nearest Neighbors are clear winners as it is very easy to understand how a model has made any prediction. Neural Networks have the least interpretability of all.
   e. All the numbers have been derived on the Infrastructure of 1 core of CPU – no GPU / TPU used for benchmarking.