Task -2

This task is about the detailed analysis of task 1.

Firstly, in task 1 we perform an overfitting analysis to understand what is overfitting. An undesirable behaviour of a machine learning system used for predictive modeling is known as overfitting.

Overfitting:

Overfitting occur due to

1. Data used for training has noise (junk values) in it since it has not been cleansed.
2. The model's variance is substantial.
3. The used training dataset's size is insufficient.
4. The model is overly intricate.

For overfitting to not occur we need to decrease the capacity of model to memorize the train data.

Underfitting occur due to high bias and low variance. This can be resolved by increasing the number of features, by performing feature engineering, by removing noise from the data

In this task we took an example of an overfitting ML model to a training dataset. Initially we define a synthetic dataset and make\_classification function is used to define a binary classification prediction. Next, we split the data by importing train\_test\_split into 7:3 ratio. Now we use decision tree classifier and test on diff. tree depths from 1 to 30. Each tree depth will be listed, a specific tree will be fitted to the training dataset, and the tree will then be assessed on both the train and test sets.  Performance on both train and test should get better as the tree's depth rises, but at a certain point, it should start overfitting the trained dataset at the cost of worse performance on the hold test set. Now we will plot the accuracy of train and test scores.

Now we took a counterexample of overfitting a scikit\_learn here we vary the no. of neighbours from 1 to 50 to see the difference

Cross-Validation

Cross-validation is a statistical technique for assessing and contrasting learning algorithms that involves splitting the data into two sections: one for learning or training a model and the other for model validation.

Now we perform Cross-validation. It uses a statistical technique to divide data into groups in order to assess and compare learning algorithms. The train test split helper function in scikit-learn can be used to quickly compute a random split into training and test sets. Loading the iris data set will allow us to apply a linear support vector machine on it. We split the data into 6:4 ratio. Now we estimate the accuracy by splitting the data and computing the score, 5 consecutive times with different split each time. Now we use the cross\_validate function for multiple metric evaluation. A list, tuple, or set of names for predefined scorers can be used to specify the multiple metrics. By passing a cross validation iterator in its place, you may potentially utilize different cross validation techniques we used shuffle split in the task. And also used iterable yielding splits as arrays of indices.

Different Cross-Validation iterators used:

1. K-fold: It separates all samples into k folds
2. Repeated K-Fold: repeats K-Fold n times
3. Leave One Out: It creates learning set by taking all the samples except for one

Validation Curve

Model accuracy (scores) for training and test data are represented by the validation curve in relation to the inverse regularization parameter.

Here we import the modules required and setup some plots. Then we read the data and combine both the train and test sets. Now we built up various functions to encrypt the data in the set and convert it to a machine-readable format. In this instance, we convert gender to a numerical function and set the family sizes to 1, 2, 3, or 4. The family size variable's dispersion is narrowed and its significance is raised by treating all families with more than three members as four and setup dataframe containing faces and ages and add to another dataframe. Now we set up random Forest Model and use utility to plot the validation curve

Plotting ROC curve:

Firstly, we import all the necessary packages fit the model and plot the ROC curve. Graph displaying a model's performance across all categorization thresholds