I. Introduction:

- Given the Breast Cancer Wisconsin dataset in the UCI repository, I want to use all 9 given features to predict whether the tumor is benign (noncancerous) or malignant (cancerous). The *'Class'* column indicates that 2 is for benign and 4 is malignant. The motivation for trying to determine the "best" DM classifier is because there are 2 different kinds of mistakes that we can make for breast cancer diagnosis: False Positive (when a patient's tumor is predicted to be malignant but is in fact benign) and Fall Negative (when a patient's tumor is predicted to be benign but is in fact malignant). We should prefer False Positive mistake more because it is way worse to predict a tumor to be benign while it is actually malignant. A false prediction can dramatically affect the patient. Therefore, I want to determine the best and right classifier.
- A factor that can be considered in determining how good a classifier is must be the accuracy on both training data and validation data without overfitting or underfitting. Moreover, if the classifier's training time is relatively reasonable, it can be considered good since our dataset is small (around 700 samples only) so if a classifier takes days to train, I would not consider it as a good one to use.

II. Methods:

1. Data Preprocessing:

- For me to follow the dataset easier, I added a row in the dataset file with id, all 9 features' names, and class columns and assign na_values='?'. I dropped all duplicate rows before doing anything else. It turns out there are 8 columns being duplicated with the exact same id, features, and class labels. To compute missing values, I use *KNNImputer* by scikit-learn, which is a widely used method and robust way to impute missing values. For this dataset, I decided to do imputation from a 1-nearest neighbor *(imputer = KNNImputer(n_neighbors=1))*. When split the data into training and validation test, I chose *random_state=42* because here.
- Moreover, I make sure that the class labels are -1 for benign and +1 for malignant instead of 2 and 4 respectively.

2. Packages/libraries (I used Python to utilize data science packages for this project):

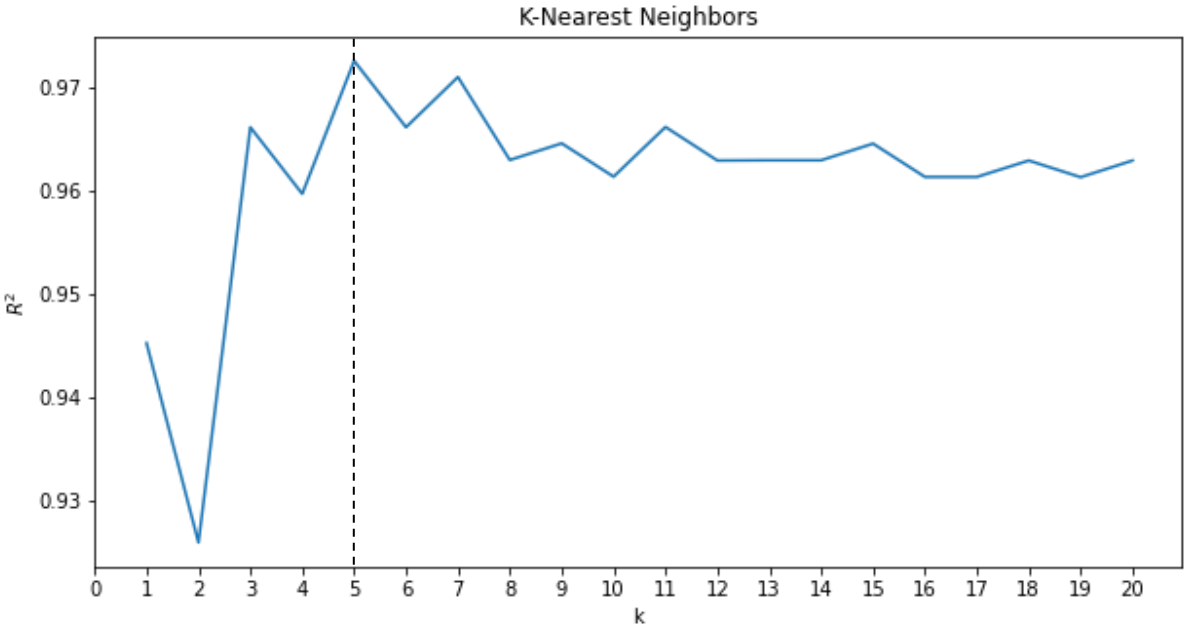
- Pandas, numpy: read in and preprocess data as well as to do computational math.
- Matplotlib: create visualization plots
- Scikit-learn: prebuilt models, classifiers.

- Tensorflow and keras library: build neural networks.

III. Results:

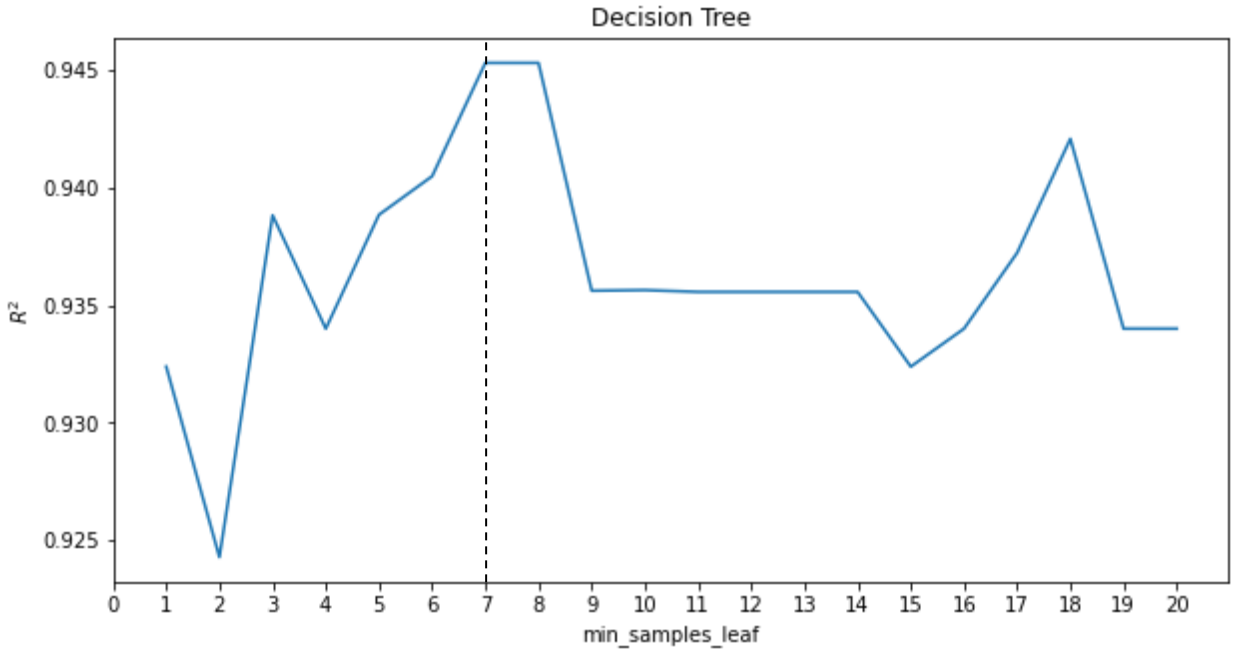
1. K-nearest Neighbors:

- K-nearest Neighbors is a supervised machine learning algorithm that can be used to solve both classification and regression problems. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Meanwhile, in k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.
- Advantages: the algorithm is simple and easy to implement. There's no need to build a model, tune parameters, or make any additional assumptions. It can be used for classification and regression.
- Disadvantages: the algorithm gets significantly slower as the number samples, predictors, variables, and k increase.
- I used KFold operation on the dataset to build and fit new k-NN models (k = [1, 20]) with each fold and computed the R^2 score to evaluate the performance of the model, where R^2 represents the proportion of variance in the dependent variable.
- I picked k = 5 because it yields the highest R^2 value among all k's. The performance of the model when k = 5 is 0.97261



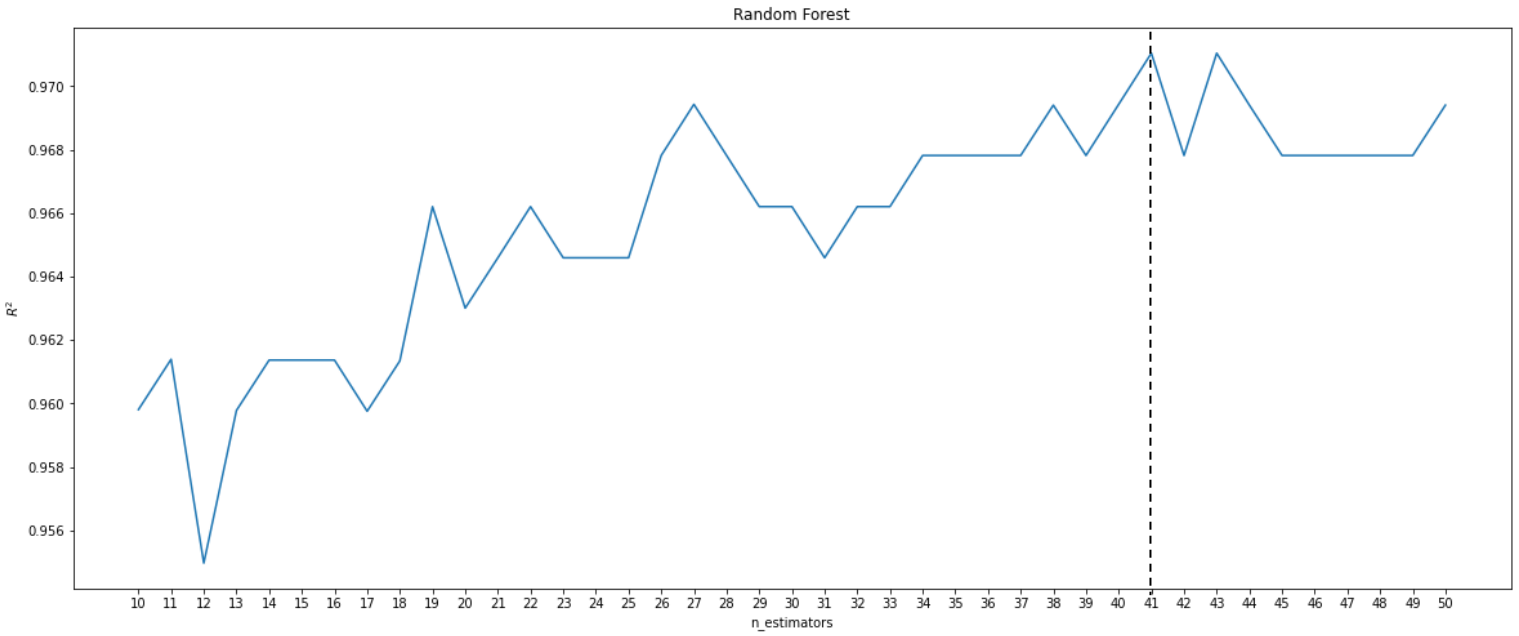
2. Decision Tree:

- A decision tree is a machine learning modeling technique for regression and classification problems. In a decision tree, a leaf node will represent class labels. Starting at the root node, we pick an attribute/feature to split the data into subsets and repeat picking splits on each child node until all data points in a subset are similar or there are no more features to split with. For a classification tree, the goal is to increase the predictability of the label as we traverse the tree (decrease entropy or Gini impurity). For a regression tree, the goal is to decrease prediction error of the response variable as we traverse the tree
- Advantages: very simple to understand due to their visual representation. Decision tree requires little data and can handle both qualitative and quantitative data. Does not require normalization or scaling of the data.
- Disadvantages: often involves higher time to train the model. Decision tree training is relatively expensive. Most importantly, a small change in the data can cause a large change in the structure of the decision tree, which leads to instability.
- Again, I computed the model scores on the minimum number of samples required to be at a leaf node ranging from 1 to 20, inclusively. I picked min_samples_leaf = 7 because the performance of the model is the highest (0.94529).



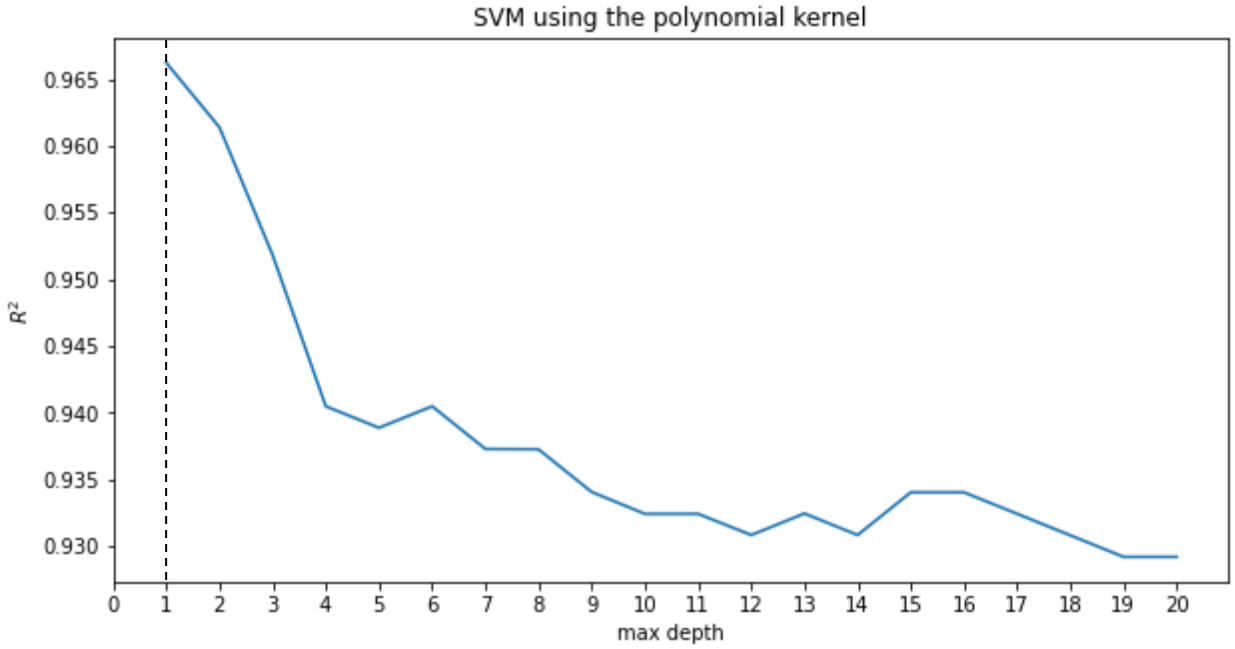
3. Random Forest:

- Random forest is based on the bagging algorithm and uses ensemble learning technique. Each individual tree in the random forests outputs a class label prediction and the class with the most votes becomes the model's prediction. Generally, we use a bootstrapped dataset and only some features to generate a single decision tree. Repeat to add more decision trees to the random forest. Class label predictions are made by evaluating all trees and returning the majority vote (for classification problem) or average value (for regression problem).
- Advantages: random forest can be used to solve both classification and regression problems. It also works well with both categorical and continuous values. No feature scaling required. Unlike a single decision tree, random forest algorithm is very stable since a new data point may impact one tree, but it is hard for it to impact all the trees.
- Disadvantages: this algorithm requires much more computational time and complexity. Moreover, random forest requires much more time to train as it generates a lot of trees.
- Again, I computed the model scores on the number of trees in the forest ranging from 10 to 50, inclusively. I picked n_estimators = 41 because the performance of the model is the highest (0.97104).



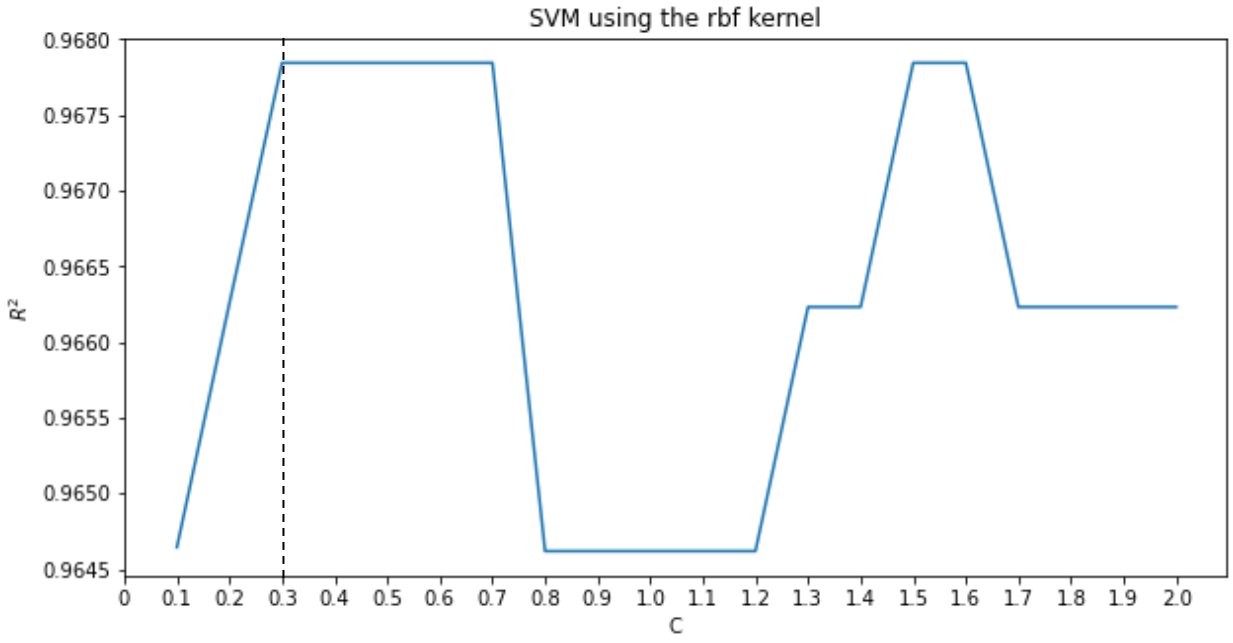
4. SVM using the polynomial kernel:

- Support Vector Machines are supervised learning models that normally produce significant accuracy with less computation power. The objective of the Support Vector Machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. While there are many possible hyperplanes, the main goal is to find a plane that has the maximum margin (maximum distance between data points of both classes). For degree-*d* polynomials, the polynomial kernel is defined as $K(x, y) = (x^T y + c)^d$, where x and y are vectors in the input space.
- Advantages: SVMs can be used for both regression and classification tasks, but it is widely used in classification objectives. Moreover, SVM is more effective in high dimensional spaces and in cases where the number of dimensions is greater than the number of samples.
- Disadvantages: SVM algorithm is not suitable for large datasets. The algorithm generally does not perform well when the dataset has a lot of noise. In cases where the number of features exceeds the number of training data samples, the SVM will underperform.
- I computed the model scores on degrees ranging from 1 to 20, inclusively. I picked degree = 1 because the performance of the model is the highest (0.96623).



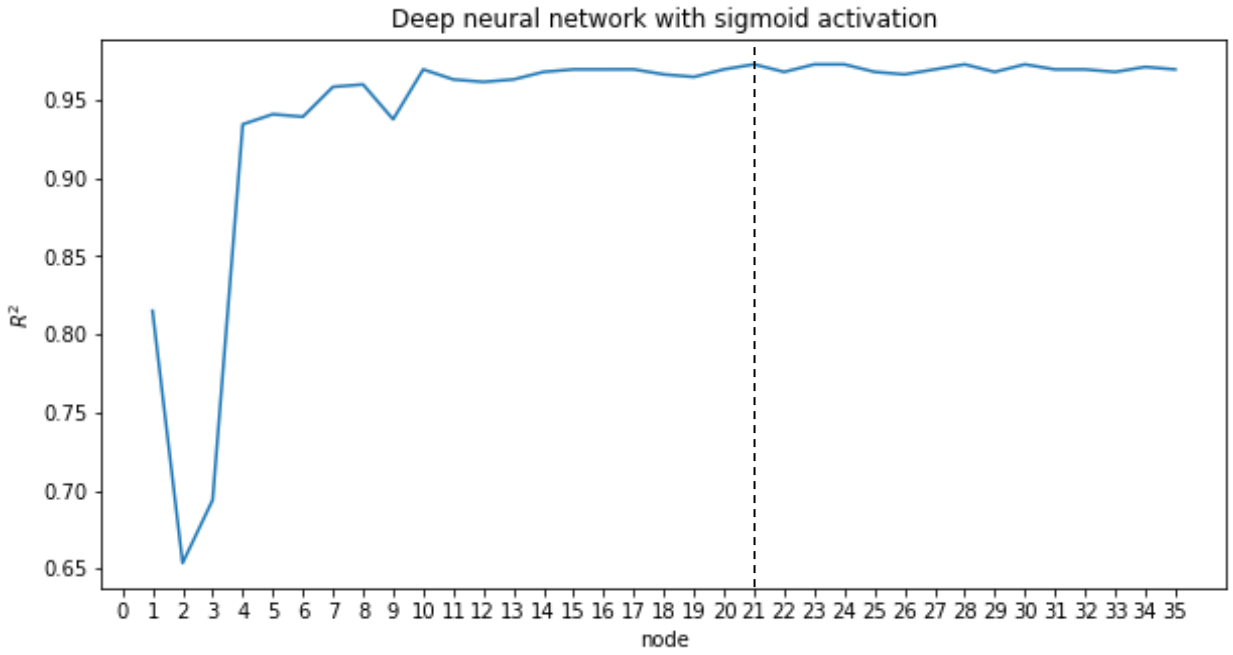
5. SVM using the Radial Basis Function (RBF) kernel:

- The RBF kernel function for two points x and y computes the similarity or how close they are to each other. This kernel can be mathematically represented as $K(x, y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ where, σ is the variance and hyperparameter and $\|x - y\|$ is the Euclidean (L_2-norm) distance between two points x and y. The RBF Kernel SVM has 2 hyperparameters associated with it in the scikit-learn library, 'C' for Regularization parameter and 'γ' fpor the RBF kernel. Here, γ is inversely proportional to σ.
- Advantages: its similarity to K-Nearest Neighbor Algorithm. RBF Kernel overcomes the space complexity as it just needs to store the support vectors during training and not the entire dataset.
- Disadvantages: We can see that as γ increase, σ reduces, the model tends to overfit for a given value of C. Therefore, it is a must to achieve the best Bias-Variance Trade off.
- I computed the model scores on regularization parameter C ranging from 0.1 to 2.0, inclusively. I picked C = 0.3 because the performance of the model is the highest (0.96784).



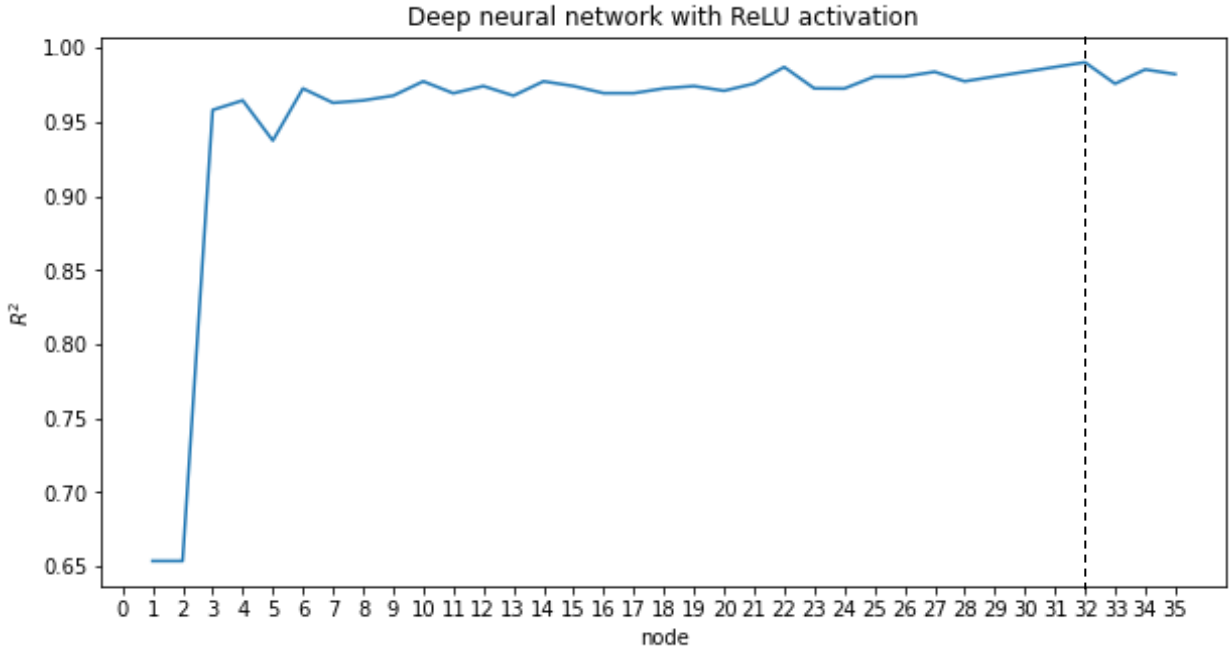
6. Deep neural network with sigmoid activation:

- Neural network is a model comprised of connected layers (an input layer, an arbitrary number of hidden layers, and an output layer) of nodes. The input nodes are predictor variable values; hidden layer nodes transform incoming values with an activation function; and finally, output nodes indicate the prediction. Sigmoid activation is denoted as $f(x) = \frac{1}{1+e^{-x}}$. My neural network consists of 3 hidden layers that transform input incoming values using the sigmoid activation function.
- Advantages: neural networks' learning methods are robust to noise in the training data, the training examples may contain errors, which do not affect the final output. Moreover, NNs can process data in parallel, meaning they can handle more than one task at the same time, hence dealing with a large dataset is easy for NNs.
- Disadvantages: NNs take a long time to train because they support parallel processing. We may not be able to determine the proper network structure of a neural network. In general, neural networks require much more data than traditional algorithms and are more complex in computing terms.
- I computed the model scores on number of nodes per a hidden layer ranging from 1 to 20, inclusively. I picked node = 21 because the performance of the model is the highest (0.97268).



7. Deep neural network with ReLU activation:

- Same as deep neural network with sigmoid function.
- ReLU activation is denoted as $f(x) = max(0, x)$.
- I computed the model scores on number of nodes per a hidden layer ranging from 1 to 20, inclusively. I picked node = 32 because the performance of the model is the highest (0.99032).



IV. Discussion:

1. Classifiers' performance comparison:

Classifier	**Final Validation Set Accuracy**
K-Nearest Neighbors	0.9714285714285714
Decision Tree	0.9285714285714286
Random Forest	0.9571428571428572
SVM using the polynomial kernel	0.9714285714285714
SVM using the RBF kernel	0.9571428571428572
Deep neural network with sigmoid activation	0.9857142567634583
Deep neural network with ReLU activation	0.9571428298950195

2. Classifiers' runtime comparison:

Classifier	Training Time (s)	Validation Time (s)
K-Nearest Neighbors	1.1730194091796875	4.184961318969727
Decision Tree	1.2729167938232422	0.32806396484375
Random Forest	71.43592834472656	4.9648284912109375
SVM using the polynomial kernel	1.7399787902832031	0.38504600524902344
SVM using the RBF kernel	2.6597976684570312	0.431060791015625
Deep NN with sigmoid activation	1098.5031127929688	160.55011749267578
Deep NN with ReLU activation	1055.476188659668	144.16098594665527

3. Lesson learned:

- While all classifiers achieve high accuracy (all higher than 90%), I would choose K-Nearest Neighbor classifier for this problem because of its accuracy on the final validation set is relatively high compared to other classifiers (97.143%). Moreover, the run time of K-NN classifier for training and for predicting on the validation set is fast enough compared to others as well. The only classifier that achieved higher accuracy is the deep neural network with sigmoid function, but its training and validation time are a lot longer than K-NN.
- If I were given this same task for a new dataset, I would increase the number of hyperparameter for all classifiers, especially for the neural network. I first tried number of nodes from 1 to 20 but the network achieved highest accuracy at 20 nodes so this could mean that the accuracies could get better if I tested out a greater number of nodes.
- The dataset for this project is relatively small so all classifiers seem to achieve high accuracies.