|  | **Introduction to Business Data Analytics** |
| --- | --- |
| **Homework #3 (Part B)** |  |

\_\_\_Jane Doe\_\_\_\_

(put your name above)

Note: This is an individual homework. Discussing this homework with your classmates is a **violation** of the Honor Code. If you **borrow code** from somewhere else, please add a comment in your code to **make it clear** what the source of the code is (e.g., a URL would be sufficient). If you borrow code and don’t provide the source, it is a violation of the Honor Code.

Total grade:

\_\_100\_\_ out of \_100\_ points

***ATTENTION: HW3 has two parts. Please first complete the Quiz “HW3\_Part1” on Canvas. Then, proceed with Part 2 in the following page. You will need to submit (a) a PDF file with your answers and screenshots of Python code snippets and (b) the Python code.***

**(100 points) [Mining publicly available data] Use Python for this Exercise.**

**Please use the dataset on breast cancer research from this link:** [**http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data**](http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data) **We have worked with this dataset in HW2. The description of the data and attributes can be found at this link:** [**http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names**](http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names) **. Each record of the data set represents a different case of breast cancer. Each case is described with 30 real-valued attributes: attribute 1 represents case id, attributes 3-32 represent various physiological characteristics and attribute 2 represents the type (benign or malignant). If the dataset has records with missing values, you can filter out these records using Python. Alternatively, if the data set has missing values, you could infer the missing values.**

**[We have seen this data before – No need to explore the data for this exercise]**

[Note: You did *not* have to explore the data; I am providing this for a more complete answer]

Step 1: Exploring and preparing the data

The data is first imported with updated column names, and then explored using .head(), .dtypes and .describe().

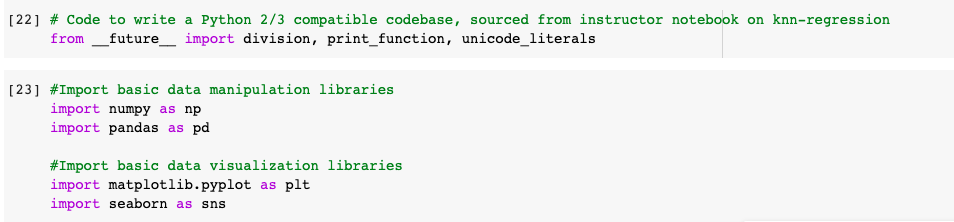
• No missing values are observed since all rows have 569 values.

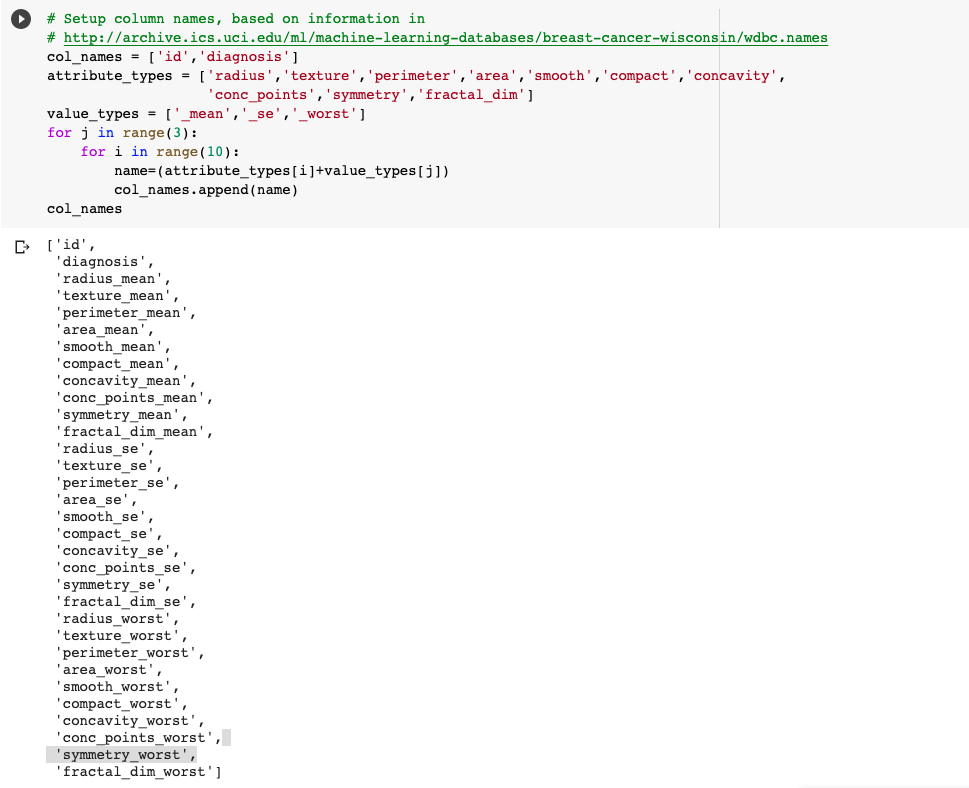
• The id variable will be excluded from our feature set since it should be relevant in making predictions

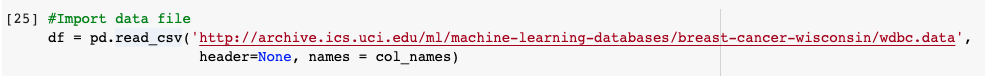
Import the data

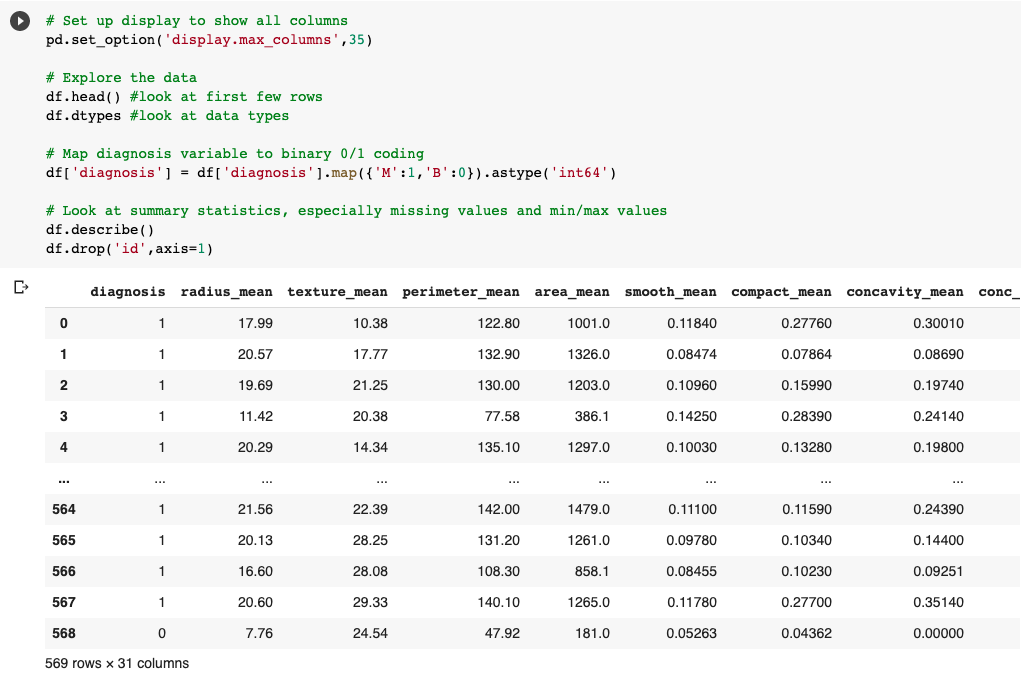
Since the dataset does not include column labels, these are generated using the information provided the wdbc.data file. The file is then imported.

Corresponding Code:

****





****

[Note: You did not have to do variable transformations; I am providing this for a more complete answer]

Next, variable distributions are examined for potential transformation. Specifically, transforming feature variables to approximately normal distributions could improve their linear predictive fit for logistic regression, and could make these relationships more readily apparent to other algorithms. The least potent transform that makes each variable approximately normal could be used.

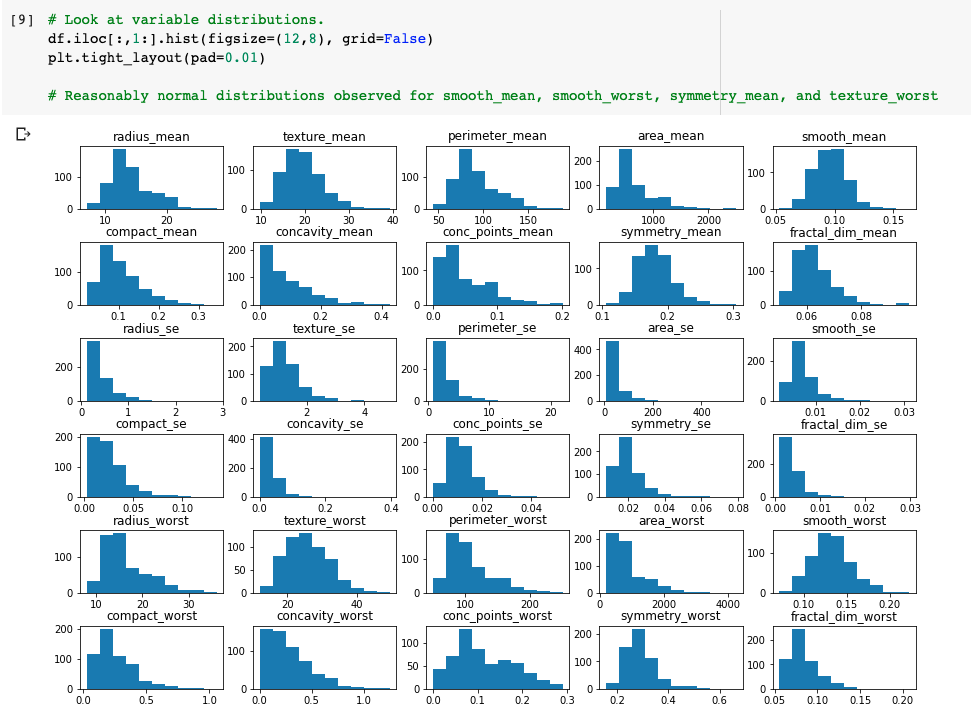
Feature histograms and normal probability plots are examined to produce the following transformations:

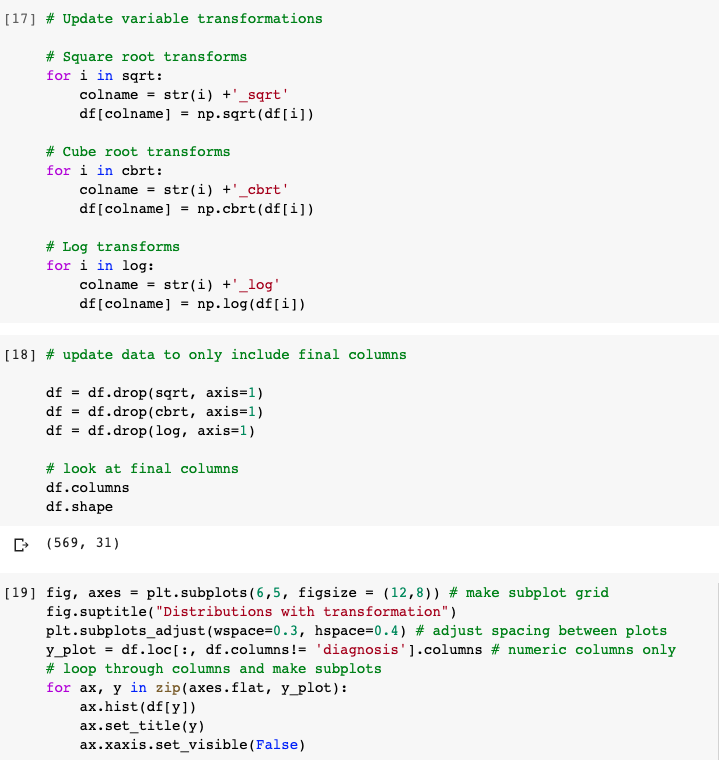
— no transformation: smooth\_mean, smooth\_worst, symmetry\_mean, texture\_worst

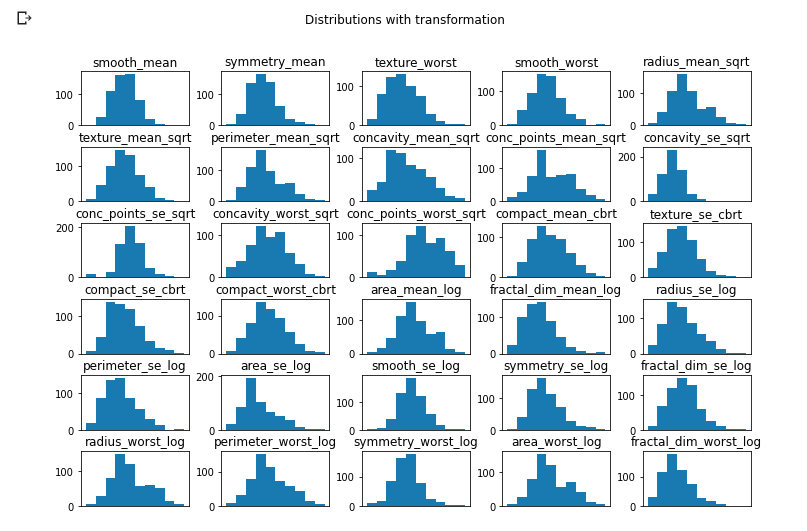
— square root transformation: radius\_mean, texture\_mean, perimeter\_mean, concavity\_mean, conc\_points\_mean, concavity\_se, conc\_points\_se, concavity\_worst, conc\_points\_worst

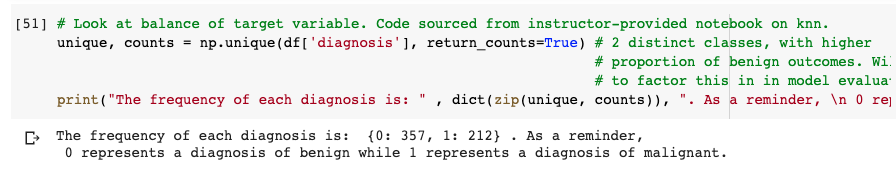
— cube root transformation: compact\_mean, texture\_se, compact\_se, compact\_worst

— log transformation: area\_mean, fractal\_dim\_mean, radius\_se, perimeter\_se, area\_se, smooth\_se, symmetry\_se, fractal\_dim\_se, radius\_worst, perimeter\_worst, symmetry\_worst, area\_worst, fractal\_dim\_worst







Examining the values of the diagnosis variable (shown below), it is noted that there is a slight unbalance present in the classes. Roughly 62% of observations fall in the benign outcome (value = 0). This indicates data stratification could be used when creating data subsets and examining model scoring measurements beyond just accuracy may be important.

**a)** **We would like to perform a predictive modeling analysis on this same dataset using the a) decision tree, b) the k-NN technique and c) the logistic regression technique. Using the nested cross-validation technique, try to optimize the parameters of your classifiers in order to improve the performance of your classifiers (i.e., f1-score) as much as possible. Please make sure to always use a random state of “42” whenever applicable. What are your optimal parameters and what is the corresponding performance of these classifiers? Please provide screenshots of your code and explain the process you have followed.**

Step 2: Model Building and Hyerparameter tuning

Part (a), Model 1: Decision Tree

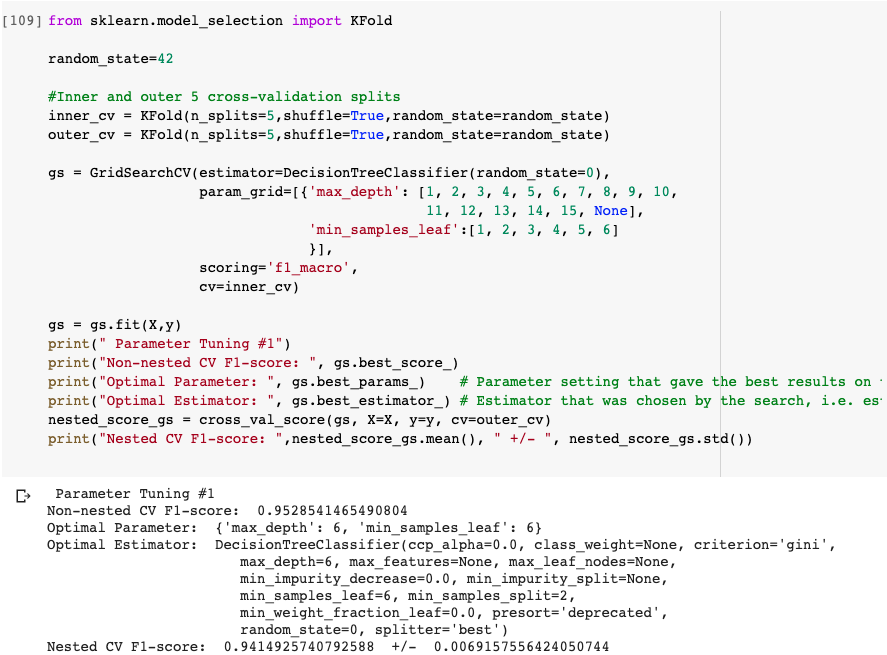
Nested cross-validation is used to optimize parameter values based on f1 score, and estimate overall generalization performance for each classification technique. Five-fold cross-validation is used across both inner and outer cross-validation layers due to the relatively small size of the data set (569 rows).

For the decision tree classifier, parameter tuning is used to find the optimal value from the following parameter entries:

* maximum tree depth: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, None
* minimum samples in leaf node: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

While this list is non-comprehensive, these constraints were felt to represent a reasonable array of choices that wouldn't lead to underfitting. A greater number of parameters and value choices can be evaluated in situations where more powerful processing resources are available.

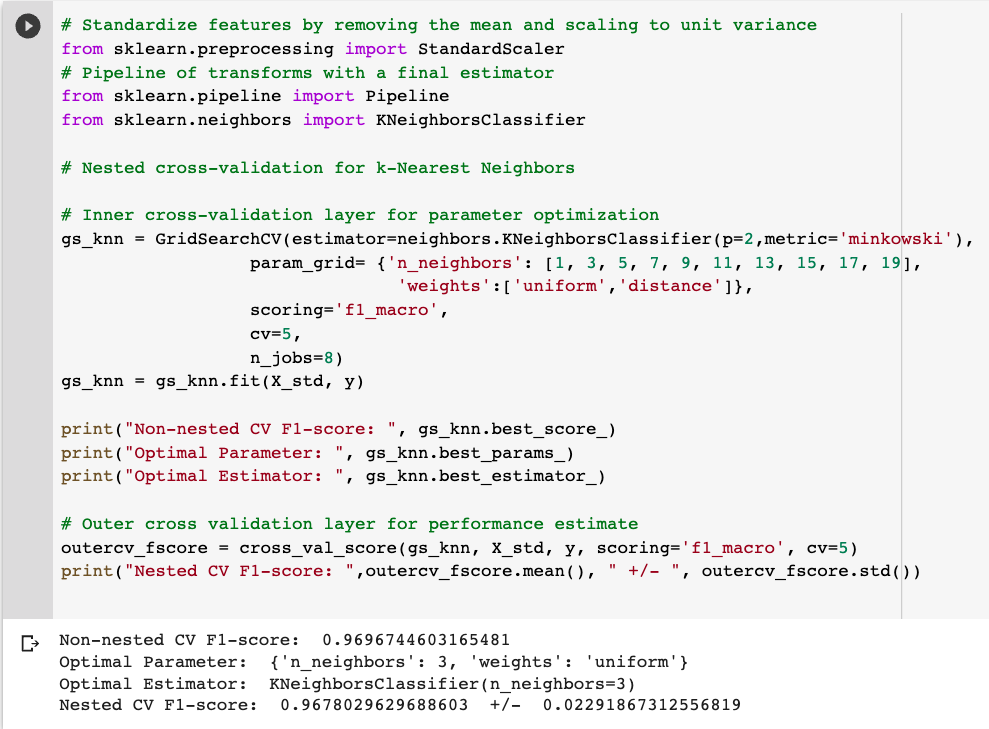
Corresponding Code:

****

Using F1-score as our performance metric, we optimize with a nested-cross validation procedure three parameters: the maximum depth of the decision tree and the minimum number of samples required to split an internal node. The optimal values are shown above and the generalization performance of the optimal model is F1-score: 0.9415 which is very close to the optimal value of 1. We could have further optimized the decision tree model by either exploring additional alternative values for the two parameters we optimized or optimizing for additional parameters too (e.g., splitting criterion).

Part (a), Model 2: k-Nearest Neighbors

Prior to model building for k-NN classification, the input values are first standardized. This is to avoid undue influence from features with larger measurement scales. Standardization is performed as shown below. Different parameter values for the number of neighbors k, weights, and distance metric p are then evaluated on the validation data. Note that when p = 1, this is equivalent to using Manhattan distance, and Euclidean distance for p = 2.

Corresponding Code:

An alternative implementation would be to use the pipeline function in order to use the same inner\_cv and outer\_cv splits as before.

Corresponding Code:

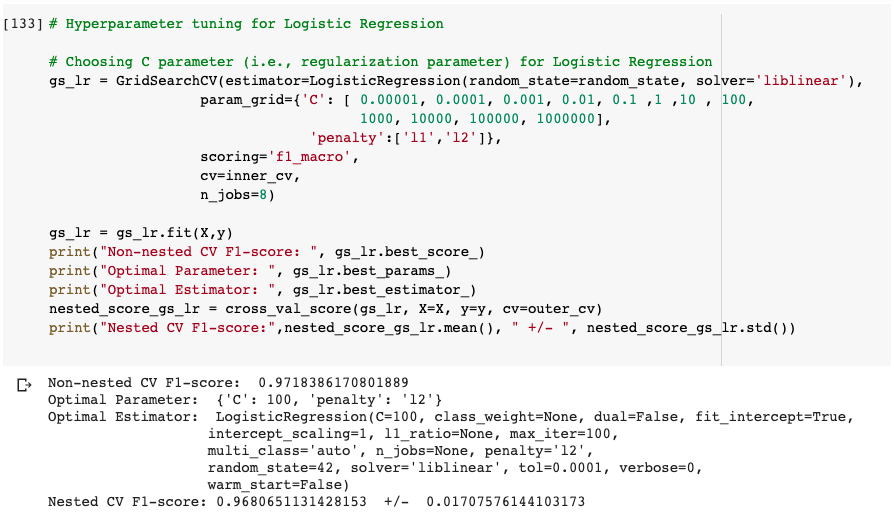


Using F1-score as our performance metric, we optimize with a nested-cross validation procedure three parameters: the distance metric, the weights, and the number of nearest neighbors. The optimal values are shown above and the generalization performance of the optimal model is F1-score: 0.9678 (0.9560 in the second alternative implementation) which is very close to the optimal value of 1. We could have further optimized the kNN model by exploring additional alternative values of the aforementioned parameters or optimizing for additional parameters too (e.g., distance metric).

Part (a), Model 3: Logistic Regression

For logistic regression, the following parameter values are evaluated using parameter tuning:

* C i.e. inverse of regularization strength: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 5, 10, 15, 50, 100, 500.
* Type of penalty: L1 Norma, L2 Norm



Using F1-score as our performance metric, we optimize with a nested-cross validation procedure two parameters: the regularization strength and the type of penalty. The optimal values are shown above (i.e., {u'penalty': u'l2', u'C': 100}}) and the generalization performance of the optimal model is F1-score: 0.9681 which is very close to the optimal value of 1. We could have further optimized the logistic regression model by exploring additional alternative values of the aforementioned parameters.

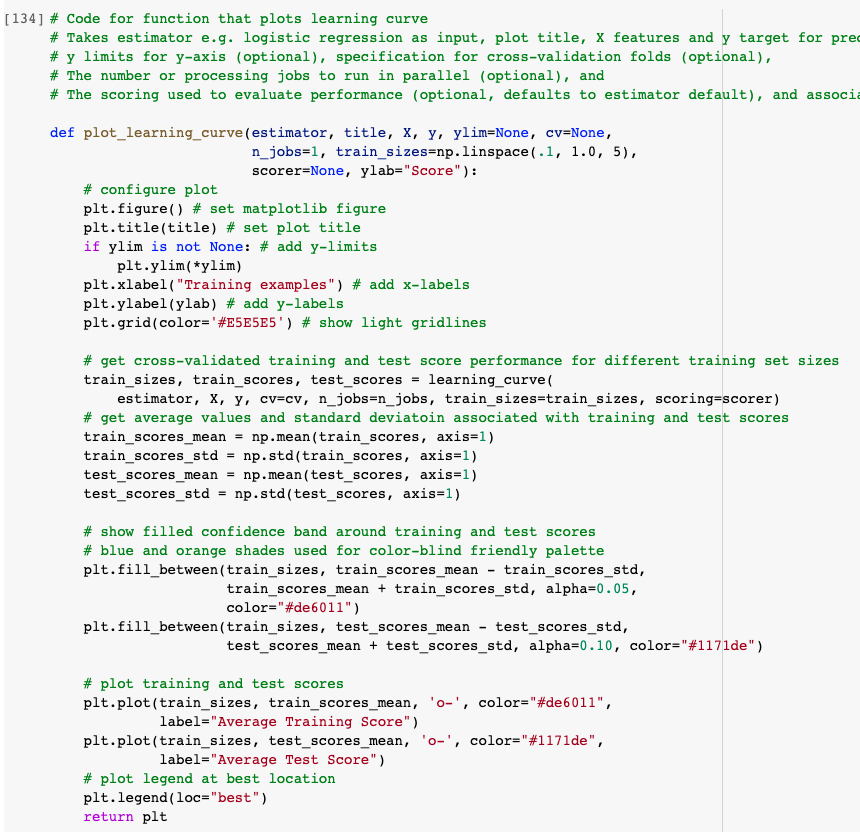
Contrasting the performance of the three classifiers, we notice that logistic regression (F1-score: 0.9681) outperforms the other two classifiers (F1-score: 0.9678 for kNN, F1-score: 0.9415 for Decision Tree).

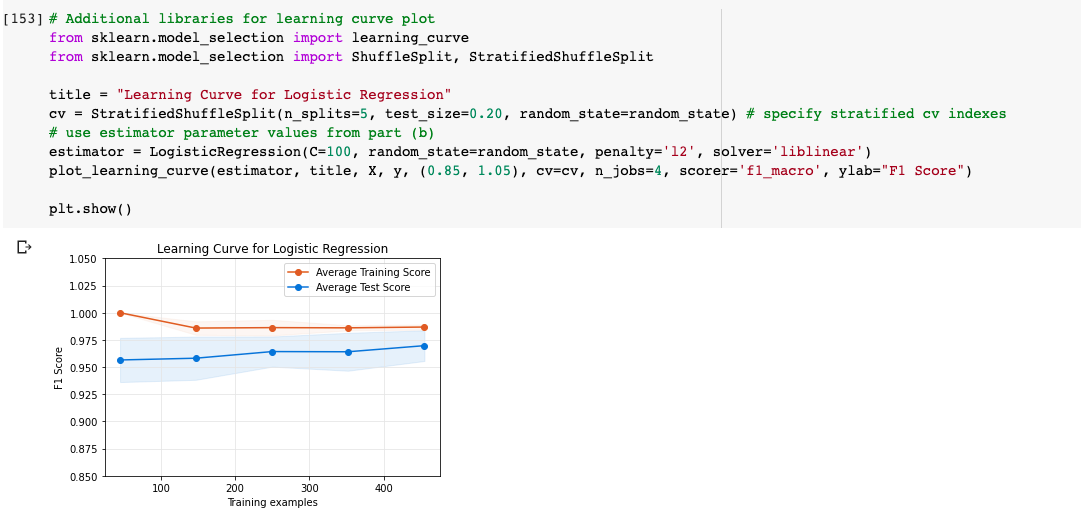
**b)** **Build and visualize a learning curve for the logistic regression technique (visualize the performance for both training and test data in the same plot). Please provide screenshots of your code and explain the process you have followed.**

The learning curve function is called with a custom test vs. training indices using StratifiedShuffleSplit and a test size of 20% as the cross-validation parameter. Moreover, the estimator for logistic regression uses parameter values based on the optimal values found in the previous part, i.e. with C=100 and l2 as the default penalty type.

The learning curve is plotted using F1 score as the scoring parameter as shown above.

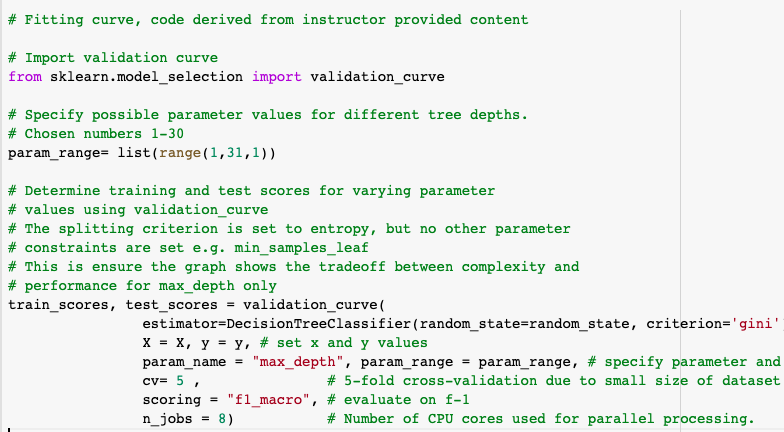
With a very small number of training examples e.g. 50, the gap between average training and test performance (based on cross-validation) is higher, but the gap is smaller at 350+ examples. Moreover, training data performance varies little across folds, but a large amount of variability is observed for test fold performance (indicated by the blue filled area). Overall, training/test performance looks stable between 350 and 450 examples suggesting little improvement from additional data, although it is still possible f1 scores may improve with more training examples.

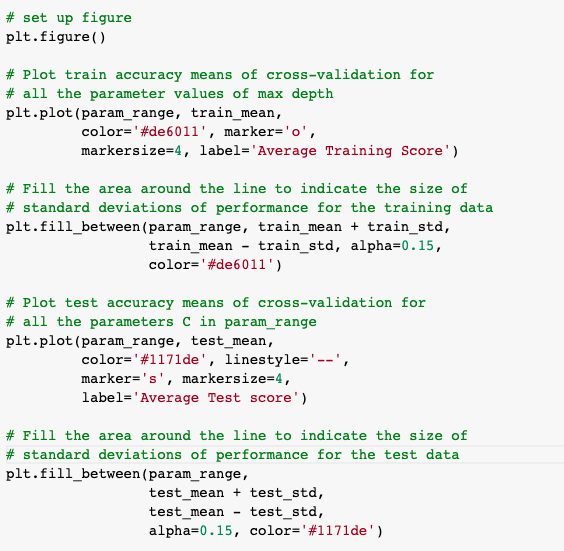
****

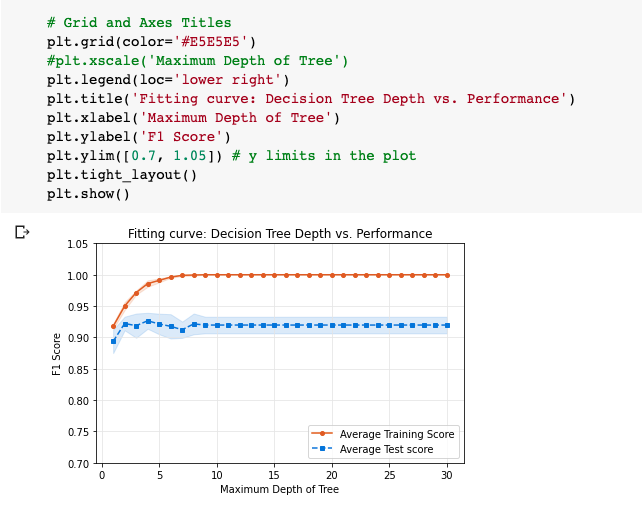


**c)** **Build a fitting graph for different depths of the decision tree (visualize the performance for both training and test data in the same plot). Please provide screenshots of your code and explain the process you have followed.**

For the fitting graph, various tree depths are evaluated as shown below.

****

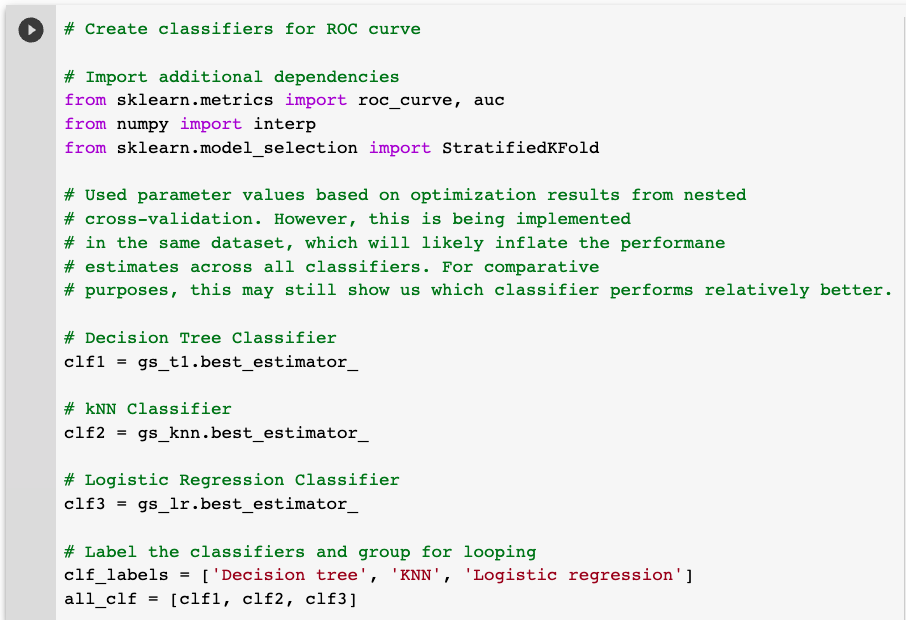
****

****

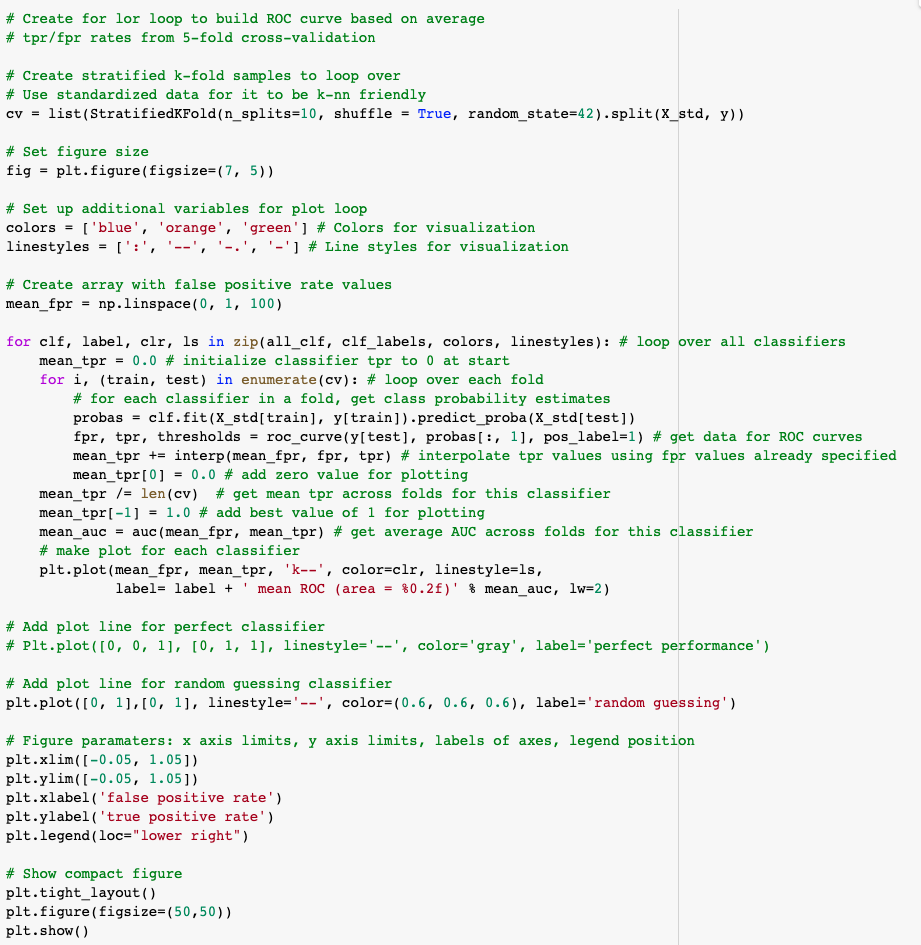
The fitting graph relies on 5-fold cross-validation to produce training and test fold performance estimates on average f1 score. The final graph is shown below, and indicates that if the only parameter being set is tree\_depth, a depth of 4 seems to be best, since this produces the highest average test score and the smallest gap between training and test performance (i.e. suggests there is less overfitting).

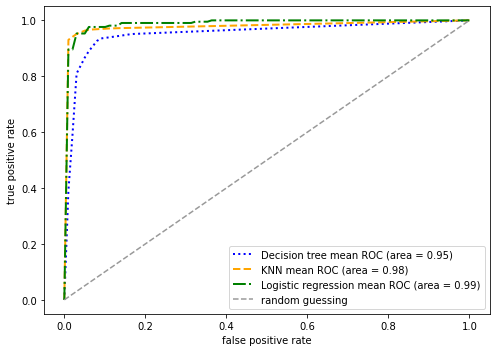
**d)** **Create an ROC curve for k-NN, decision tree, and logistic regression. Discuss the results. Which classifier would you prefer to choose? Please provide screenshots of your code and explain the process you have followed.**

The optimal parameter values for each classifier identified in part (a) using nested cross-validation are used to create models for the ROC curve, as shown below:



Moreover, 10-fold cross-validation is used to get average true positive and false positive rates for each classifier, using the code shown below:





From the curve, we can see that both k-nn and logistic regression have relatively strong performance, with high true positive rates at relatively low false positive rates. Decision tree emerges as the weakest performer here.

Overall, the logistic regression tree can be seen as the best performing classifier (consistent with the results from nested cross-validation on f1 score). This is because the highest the classifier has an overall higher area under the ROC curve at 0.99 vs. kNN’s 0.98 although the two have almost identical performance.