EGN 5442 Project 3

Introduction

The dataset has 31 columns and 104721 rows that are mostly numeric, with some exceptions. The columns are labeled as "x [i]", where [i] is an integer from 2 to 31. The exception to the column naming scheme is the "y" column that was used for the y_train and y_test variables. The rest of the data was used for the x_train and x_test variables. After cleaning and preparation, I tested two predictive models: Logistic Regression and Random Forest Classification. Accuracy and

Data Cleaning and Preparation

Initially, the dataset contained several columns with strings, non-numerical symbols, and too many missing values. For the columns with Female/Male and "D_C" / "L_C", I assigned column values to 0 and 1, respectively, for both groups. Then, I removed the following characters: '()&#' from any point in the dataset. I renames the x3 and x5 columns to 'gender' and 'age', respectively. I dropped column x31 because it had too many missing or inf values. Any numerical values that were of string type were converted to either float or int.

The data split was 80%/20% train/test, and was completed with the "train_test_split" function from Scikit-Learn Model Selection.

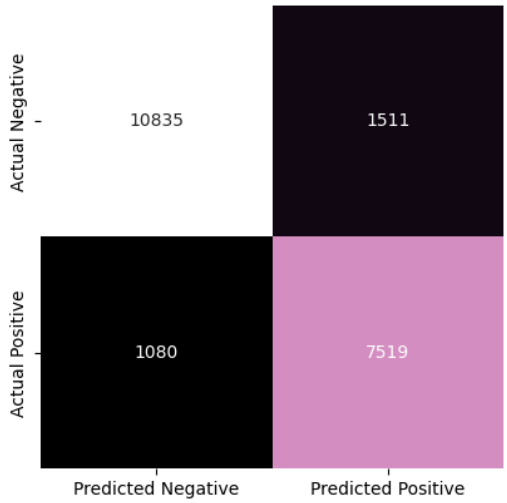
Part A – Logistic Regression

Logistic regression (LR) is classified as a supervised machine learning algorithm that attempts to predict a binary dependent variable from an independent variable. LR works extremely well for binary datasets, but lacks when considering continuous dataset. Additionally, there cannot exist collinearity between independent variables or the model breaks down.

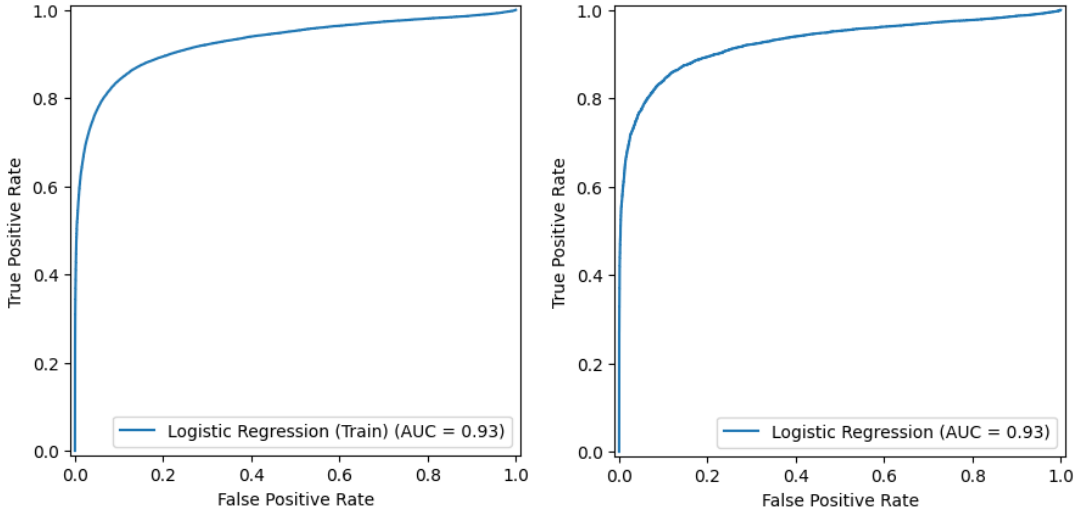
I did not perform any feature selection or analysis, but I did consider hyperparameter tuning. Hyperparameter tuning took about 10 minutes to complete with the RandomizedSearchCV package from scikit-learn. The parameter grid for tuning is as follows:

- Solver: 'saga', 'liblinear'
- Penalty: 'l1', 'l2'
- C: 0.001, 0.01, 0.1, 1, 10, 1000
- Max_iter: 10, 100, 500, 1000

The optimal parameters were{"Solver": "liblinear", "Penalty": "l1", "C": 0.01, "max_iter":100}. The confusion matrix is as follows:



Considering the ROC-AUC curves, the left figure is the LR for the training set, while the right figure is the test set. The AUC values are the same for both sets.



The accuracy, precision, recall, and F1-score are 0.8763, 0.8744, 0.8327, and 0.8530, respectively.

Part B – Random Forest Classification

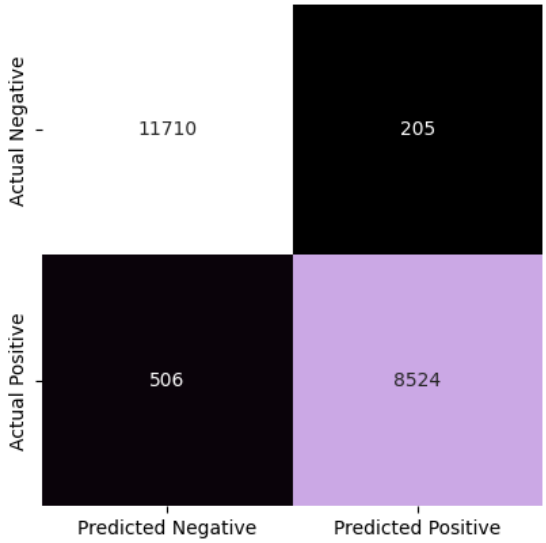
The Random Forest Classification (RFC) model is a tree-based model that takes multiple decision trees from the inputs and outputs the most common label from all trees. Advantages of RFC include its ability to handle large datasets without overfitting.

Hyperparameter tuning took about 30 minutes to complete with the RandomizedSearchCV package from scikit-learn. The parameter grid for tuning is as follows:

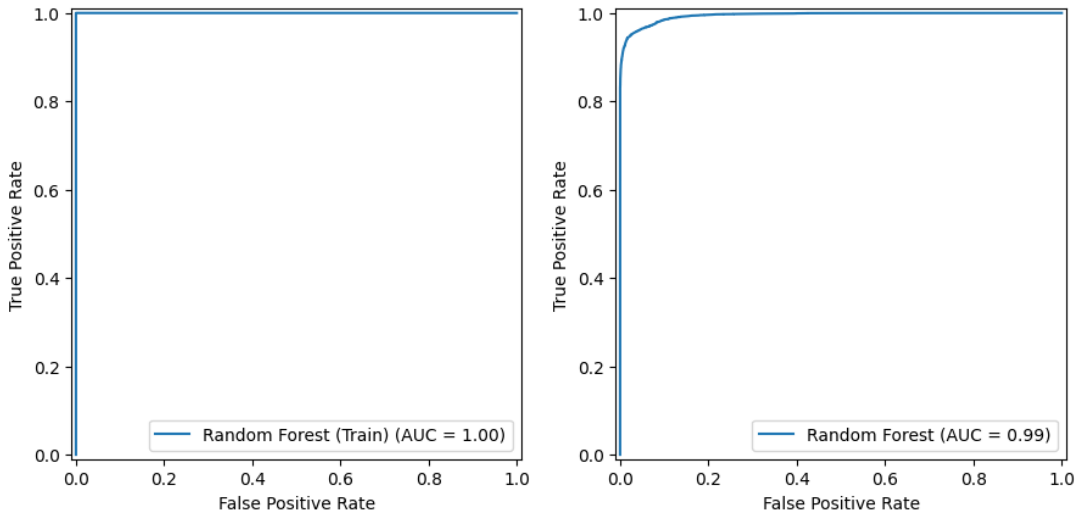
- N_estimators: 50, 100, 250
- Max_depth: None, 10, 20, 30, 40

- Min_samples_split: 2, 5, 10
- Max_features: ‘auto’, ‘sqrt’, ‘log2’

From this hyperparameter tuning, the best parameters were {‘n_estimators’: 250, ‘max_depth’: 30, ‘min_samples_split’: 2, ‘max_features’: ‘sqrt’}. The confusion matrix is as follows:



The ROC-AUC curves are as follows, with the training curve on the left and the test curve on the right. The AUC for the training curve is minimally better than the test curve.



The accuracy, precision, recall, and F1-score are 0.9661, 0.9765, 0.9440, 0.9600, respectively.

Discussion

Comparing the two models from Parts A and B, the Random Forest Classification model appears to be better at fitting the provided data. The accuracy, precision, and AUC for the RFC model are better than the LR model. If I were describing this to a business partner, I would support my

argument with the accuracy and precision metrics, and potentially with a test vs. predicted test variable graph to show an R^2-value. The closer the R^2-value is to 1, the better the fit of the model.

	Logistic Regression Model	Random Forest Classification
AUC for training [%]	93	99
AUC for testing [%]	93	100
Accuracy [%]	87.63	96.61
Precision [%]	87.44	97.65