Georgia Institute of Technology

ISYE/CSE/MGT 6748 Applied Analytics Practicum

Patient Health Risk Prediction

Michael Guo, Vazirani Yogesh & Sarfroz Nawaz Katiyar Hyder Ali

Data Wrangling and Cleaning

Developed on Windows 10 with Python 3.10.8

- Three Jupyter notebooks are provided: **1 - Create Edges File.ipynb**, **2 - One Hot Encoding.ipynb**, and **3 - Final Dataset Changes.ipynb.**
    - **1 - Create Edges File.ipynb**
        - Creates the edges dataset used in the various modeling functions
        - Required packages: pandas and numpy
        - Inputs: **med_full_final_melted.csv (Formatted Medical Claims File)** and **code_description_pairs.txt (ICD10 Base Code – Description Key File)** - obtained from this repository: https://github.com/StefanoTrv/simple_icd_10
            - Note: the **med_full_final_melted.csv** is the medical claims dataset with all ICD10 Base Codes melted into a single column and the "–1" Member Life IDs removed.
        - Outputs: **edges.csv (Edges File)**
        - Instructions: Run through each of the cells to generate an **edges.csv** file. You can adjust the path file name/path to write in the last cell.
            - Note: This notebook creates an **edges.csv** file that has the same values as the **edges.csv** file in the Data folder. However, the order of the rows are slightly different, because the original file was not sorted by the Source and Target Node names.
        - This notebook takes a significant amount of time (~30 minutes) to run.
    - **2 - One Hot Encoding.ipynb**
        - Performs one hot encoding of the ICD10 Base Codes contained in a medical claims dataset where the "-1" Member Life ID rows have been removed and the Primary, Secondary, and Tertiary ICD10 Base Codes have been melted into one column.
        - Required packages: pandas and numpy
        - Inputs: **med_full_final_melted.csv (Formatted Medical Claims File)** and **rx_full_final.csv (Formatted Pharmacy Claims File)**
        - Outputs: **formatted_data_all_codes.csv**
        - Instructions: Run through each of the cells to generate a **formatted_data_all_codes.csv** file. You can adjust the path file name/path to write in the last cell.

        - This notebook takes a significant amount of time (~30 minutes) to run.
    - **3 - Final Dataset Changes.ipynb**
        - Performs final data cleaning tasks on the **formatted_data_all_codes.csv** data and outputs the **all_data_with_ages.csv** dataset that our non-Timebound models are trained on.
        - Required packages: pandas and numpy
        - Inputs: **formatted_data_all_codes.csv (Data from the One Hot Encoding notebook)**
        - Outputs: **all_data_with_ages.csv**
        - Instructions: Run through each of the cells to generate an **all_data_with_ages.csv** file. You can adjust the path file name/path to write in the last cell.

Logistic Regression

Developed on Windows 10 with Python 3.10.8

- The functions for auto-generating both the non-Timebound and Timebound versions of the Logistic Regression Models are contained within the **logistic_regression_functions.py** file.
    - **4 – Autogenerate a Timebound Logistic Regression Model (gen_log_reg_model)**
        - Required packages: pandas, scikit-learn, and numpy
        - Inputs
            - Required
                - input_dataset - Dataframe with one-hot encoded ICD10 Base Codes, Biological Gender, and Age
                - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
                - edges - data frame containing information on the edges between nodes of the undirected graph generated for this model (generated in the Create Edges File.ipynb notebook)
                - exclusions list - List of ICD10 Base Codes to exclude
            - Optional
                - limit – number of predictor ICD10 Base Codes to select. Default: 20
                - random_seed – random seed to be used when performing the test/train split and training using the LogisticRegressionCV function. Default: None
                - test_size – the proportion of the input dataset that should separated from the training dataset. Default: 0.20
                - Num_folds – the number of folds to be used in the LogisticRegressionCV function. Default: 10
                - Max_iter –the maximum number of iterations of the LogisticRegressionCV optimization algorithm. Default: 250
        - Outputs (In a tuple)
            - target_model – the model that was autogenerated

- model_accuracy – the validation balanced accuracy of the model
- predictor_codes_df – a dataframe that contains the predictor ICD10 Base Codes chosen, their Descriptions and weights
- test_train_dict – a dictionary that contains the X_train, X_test, y_train, and y_test datasets

▪ Instructions:
- Load all the functions from the logistic_regression_functions.py file.
- Use the gen_log_reg_model function with appropriate inputs to automatically generate a logistic regression model.
- If you wish to exclude an ICD10 Base Code from being included as a predictor in the auto-generated Logistic Regression, add it to the exclusions.csv file and run the **gen_log_reg_model** function.
- There is an example for how to run this function in the **4 - Logistic Regression Function.ipynb** notebook.

o **4 – Autogenerate a Timebound Logistic Regression Model (gen_log_reg_model_timebound)**

▪ Required packages: pandas, scikit-learn, and numpy

▪ Inputs
- Required
  - medical_claims – the medical claims dataset with all ICD10 Base Codes melted into a single column and the "–1" Member Life IDs removed.
  - rx_claims – the pharmacy claims dataset
  - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
  - edges - data frame containing information on the edges between nodes of the undirected graph generated for this model (generated in the Create Edges File.ipynb notebook)
  - exclusion_list - List of ICD10 Base Codes to exclude
- Optional
  - limit – number of predictor ICD10 Base Codes to select. Default: 20
  - random_seed – random seed to be used when performing the test/train split and training using the LogisticRegressionCV function. Default: None
  - test_size – the proportion of the input dataset that should separated from the training dataset: Default: 0.20
  - num_folds – the number of folds to be used in the LogisticRegressionCV function. Default: 10
  - max_iter – the maximum number of iterations of the LogisticRegressionCV optimization algorithm. Default: 250

▪ Outputs (In a tuple)
- target_model – the model that was autogenerated
- model_accuracy – the validation balanced accuracy of the model

- predictor_codes_df – a dataframe that contains the predictor ICD10 Base Codes chosen, their Descriptions and weights
- test_train_dict – a dictionary that contains the X_train, X_test, y_train, and y_test datasets

- Instructions:
  - Load all the functions from the logistic_regression_functions.py file.
  - Use the gen_log_reg_model_timebound function with appropriate inputs to automatically generate a logistic regression model.
  - If you wish to exclude an ICD10 Base Code from being included as a predictor in the auto-generated Logistic Regression, add it to the exclusions.csv file and run the **gen_log_reg_model** function.
  - There is an example for how to run this function in the **4 - Logistic Regression Function (Timebound).ipynb** notebook.

Random Forest

This is developed on Windows using python 3.10X

The functions needed to auto-generate both the non-Timebound and Timebound versions of the Random Forest Classifier Models are contained within the **random_forest_functions.py** file.

random_forest_functions.py has 11 functions supporting the Random Forest Classifier Model

get_predictor_codes:

Function finds and returns top 'N' predictor ICD10 Base Codes (based on weights), that has strong relation with the target ICD10 Base Codes. By default, the function returns the top 20.

create_time_bound_claim:

Function creates a Timebound medical claims dataset for training and prediction of the model

gen_one_hot_data_input_base_codes:

Function to One-Hot encode the Timebound/Non-Timebound Dataset for training and prediction of the model

create_final_dataset:

Function creates the final modeling dataset for training and prediction of the model

gen_rf_model:

Function uses the input dataset, splits into train and test. Trains the RF Classifier model, predicts with test dataset and returns the target model with predict probabilities

gen_rf_model_timebound

Function specific to Timebound model, uses other above functions to create the timebound dataset and return the target RFClassifier model with predict probabilities

hyperparameter_tuning

Function to find the Hyper parameters specific to the input dataset and ICD10 Base code to predict. Returns best parameters of n_estimators and max_depth for each target ICD10 Base Code.

hyperparameter_tuning_timebound

Function uses the Timebound dataset and uses hyperparameter_tuning to find the optimal Parameters for the target ICD10 Base Codes.

get_accuracy_sensitivity_for_all_threshold

Function returns accuracy and sensitivity of the input RFClassifier for different threshold values (.1, .3 and .5)

get_accuracy_sensitivity

Function returns the accuracy and sensitivity of the input RFClassifier given threshold value.

gen_rf_plot

Function to plot the confusion matrix, AUC/ROC Curve of the given RFClassifier model.

There are 3 Python Jupyter notebook files '**5 - Hyper Tuning Tree Model**', '**6 - Training and Evaluating Tree Model With HyperParam**' and '**7 - Evaluation Tree Model With HyperParam With Difference threshold**' that use the above functions to train, predict, evaluate and plot the models.

Step 1: Run the Below Jupyter notebook to find the Hyper Parameters for the target ICD10 Base Codes – E11, C18, C50, I10, I25 and N18

Note: This notebook runs almost **~30min**, so please run only if needed to find the Hyper Parameters

- '5 - Hyper Tuning Tree Model.ipynb' (hyperparameter_tuning & hyperparameter_tuning_timebound) –
  - Required packages: pandas, scikit-learn, and numpy

  Non-Timebound (hyperparameter_tuning)

  - Inputs:
    - Required
      - all_data_with_ages_without_timebound – One-hot encoded dataset with all ICD10 Base Codes included without timebound
      - ICD10 Base Code – Target ICD10 Base Code to find Hyper Parameters for
      - predictor_codes – Top 20 Predictor Codes to use to find the Hyper Parameters
    - Optional
      - random_seed – random seed to be used when performing the test/train split and training using the RFClassifier function.
      - test_size – the proportion of the input dataset that should be separated from the training dataset
  - Outputs:
      - hyper_params_with_ages_without_timebound – Dictionary with Hyper parameters for n_estimators and max_depth

Timebound(hyperparameter_tuning_timebound)

- Inputs:
    - Required
        - medical_claims – the medical claims dataset with all ICD10 Base Codes melted into a single column
        - rx_claims – the pharmacy claims dataset
        - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
        - edges - dataframe containing information on the edges between nodes of the undirected graph generated for this model (generated in the Create Edges File.ipynb notebook)
        - exclusion_list - List of ICD10 Base Codes to exclude
    - Optional
        - random_seed – random seed to be used when performing the test/train split and training using the RFClassifier function.
        - test_size – the proportion of the input dataset that should separated from the training dataset
- Outputs:
        - hyper_params_with_ages_without_timebound – Dictionary with Hyper parameters for n_estimators and max_depth
- Instructions:
    - Load all the functions from random_forest_functions.py
    - Read the input medical/RX claims, edges and exclusion list
    - Run the hyperparameter_tuning and hyperparameter_tuning_timebound function to find the optimal parameters for the model to use.

Step 2: Run the below Jupyter notebook to use the Hyper Parameters for the target ICD10 Base Codes – E11, C18, C50, I10, I25 and N18 and evaluate the model's performance based on Accuracy, Sensitivity & AUC/ROC. Also plot the model' ROC curve.

- '6 - Training and Evaluating Tree Model With HyperParam' (gen_rf_model & gen_rf_model_timebound)
    - Required packages: pandas, scikit-learn, and numpy

Non-Timebound

gen_rf_model

- Inputs:
    - Required
        - input_dataset - Dataframe with one-hot encoded ICD10 Base Codes, Biological Gender, and Age
        - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for

- edges - data frame containing information on the edges between nodes of the undirected graph generated for this model (generated in the Create Edges File.ipynb notebook)
- exclusions list - List of ICD10 Base Codes to exclude
- best_params – Optimal Parameters from Step 1

- Optional
  - random_seed – random seed to be used when performing the test/train split and training using the RFClassifier function.
  - test_size – the proportion of the input dataset that should separate from the training dataset
- Outputs (tuples)
  - target_model – the model that was autogenerated
  - pred_prob – models predicted probabilities
  - predictor_codes_df – a dataframe that contains the predictor ICD10 Base Codes chosen, their Descriptions and weights
  - test_train_dict – a dictionary that contains the X_train, X_test, y_train, and y_test datasets

get_accuracy_sensitivity:

- Inputs:
  - Required
    - Target_model – Target model from gen_rf_model
    - Pred_prob – Predicted Probabilities from gen_rf_model
    - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
    - threshold – Threshold to use to find the accuracy, sensitivity, confusion matrix and model prediction
  - Outputs(tuples)
    - accuracy, sensitivity, confusion matrix, prediction of the target model.

gen_rf_plot

- Inputs:
  - Required
    - Target_model – Target model from gen_rf_model
    - test_train_dict – a dictionary that contains the X_train, X_test, y_train, and y_test datasets
    - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
    - model_prediction – prediction for the target model
    - accuracy – accuracy of the target model
    - sensitivity – sensitivity of the target model
    - confusion_matrix – confusion matrix of the target model
    - threshold – Threshold to use to find the accuracy, sensitivity, confusion matrix and model prediction

- ❖ Optional
  - type – string contains whether it is for with and without timebound
- Outputs (Only Plot)
  - No outputs only plot of Confusion matrix and AUC/ROC Curve.

Timebound

- Same as Non-Timebound except it uses gen_rf_model_timebound to build the model

- Instructions:
  - Load all the functions from random_forest_functions
  - Read the input medical/RX claims, data with all ages, edges, and exclusion list
  - Run the gen_rf_model and gen_rf_model_timebound to build the RFClassifier model with hyperparameters and for all the 6 ICD10 Target Base Codes.

Step 3: Run the below Jupyter notebook to use the Hyper Parameters for the target ICD10 Base Codes – E11, C18, C50, I10, I25 and N18 and evaluate the model's performance based on Accuracy, Sensitivity & AUC/ROC. Also plot the model

- '7 - Evaluation Tree Model With HyperParam With Difference threshold' (get_accuracy_sensitivity_for_all_threshold)
  - ❖ This Jupyter notebook also uses gen_rf_model and gen_rf_model_timebound as given in the Step 2 to build the model for Timebound and Non-Timebound and in addition uses the function get_accuracy_sensitivity_for_all_threshold to find the accuracy and sensitivity of the model for evaluation
    - Required packages: pandas, scikit-learn, and numpy

get_accuracy_sensitivity_for_all_threshold

- **Inputs:**
  - Required
    - Target_model – Target model from gen_rf_model or gen_rf_model_timebound
    - Pred_prob – Predicted Probabilities from gen_rf_model_timebound
    - input_code – a string denoting the ICD10 Base code the user wants to predict the probability for
  - Optional:
    - Input_type – String to differentiate whether it is for Timebound and Non-Timebound model
- **Outputs:**
  - No outputs only print statement for accuracy and sensitivity of the model
- **Instructions:**
  - Load all the functions from random_forest_functions
  - Read the input medical/RX claims, data with all ages, edges and exclusion list
  - From Step 1, get the hyperparameters for the target models
  - Run gen_rf_model /gen_rf_model_timebound and get_accuracy_sensitivity_for_all_threshold to get the accuracy and sensitivity of the Timebound and Non-Timebound models for all the ICD codes

Neural Network

This is developed on MAC using python 3.7X

The Jupyter notebook for running this model is "**Create Neural Network Classifier.ipynb**" This notebook generates three different types of models. All models use "Age" as a predictor. The models are: Using All ICD10 Base codes, using a subset of the ICD10 Base Codes, Using Timebound datasets. We ran all models using six ICD10 Base Codes as targets. They are - E11, C18, C50, I10, I25 and N18

Required libraries to run these models: numpy pandas, scikit-learn, matplotlib, tensorflow, keras

1. **Model with All Predictors** – This model takes in one ICD10 Base Code as the target code and all the remaining ICD10 Base codes as predictors

   Inputs

   - One –hot encoded dataset with age, using all ICD10 Base Codes (all_data_with_ages.csv)

   Output:

   - Confusion Matrix, ROC curve plot, Sensitivity, AUC score and Accuracy

   Instructions:

   - Run the function create_neural_network_model(dataset, target_code, dataset_name, use_subset, threshold) where:

     Dataset – imported one-hot encoded file

     Target_code = ICD10 Base Code for Prediction

     Dataset_Name = Just a name to store output variables

     Use_subset = FALSE

     Threshold value = setting a threshold value for predictions

2. **Model with a subset of predictors** – This model takes in one ICD10 Base Code as the target code and a list of the pre-identified predictors (top 20). For running this model

   Inputs:

   - One –hot encoded dataset with age, using all ICD10 Base Codes (all_data_with_ages.csv)
   - Subset of predictors, Use the code block to generate the "Predictors Used.csv"

   Output:

   - Confusion Matrix, ROC curve plot, Sensitivity, AUC score and Accuracy

   Instructions:

- Run the function is create_neural_network_model(dataset, target_code, dataset_name, use_subset, threshold) where:

  Dataset – imported one-hot encoded file

  Target_code = ICD10 Base Code for Prediction

  Dataset_Name = Just a name to store output variables

  Use_subset = TRUE

  Threshold value = setting a threshold value for predictions

3. **Model with Timebound dataset** - This model takes in one ICD10 Base Code as the target code and all other ICD10 Base codes as predictors. For this model we generated six different datasets, one for each of the target codes identified above. These are available for use in the Data folder. If you would like to generate new timebound dataset for any other target ICD10 Base Code, please use the gen_timebound_training_data function from the timebound_functions.py file to generate a new csv file as input to the NN model. The gen_timebound_training_data function takes medical_claims data (with a melted ICD10 Base Code column), pharmacy_claims data, and input_code (target ICD10 Base Code) as inputs and writes a modified one-hot encoded training dataset for that specific target code.

   Input

   - One –hot encoded dataset with age, using all Timebound ICD10 Base Codes (<TimeBound-ICD10 Target Base Code.csv>)

   Output:

   - Confusion Matrix, ROC curve plot, Sensitivity, AUC score and Accuracy

   Instructions:

   - Run the function is create_neural_network_model(dataset, target_code, dataset_name, use_subset, threshold) where:

     Dataset – imported one-hot encoded file

     Target_code = ICD10 Base Code for Prediction

     Dataset_Name = Just a name to store output variables

     Use_subset = FALSE

     Threshold value = setting a threshold value for predictions

   Tuning the Neural Network

   You can modify the train-test split to test the model. We have used an 80:20 split for all our models. There are several tuning parameters that you can modify to tune the NN Model. The values for these parameters can be changed in the “create_neural_network_model” function

In the model creation method, you can modify:

- Number of hidden layers
- Number of neurons
- Activation function

In the model compile method, you can modify:

- Loss
- Optimizer
- Metrics to be measured

In the fit method, you can modify:

- Epochs

You can set different threshold values to compare the performance of the model with regards to sensitivity, AUC and Accuracy.