ISYE 6740 Final Project

Predicting Diabetes and Prediabetes using Machine Learning Models

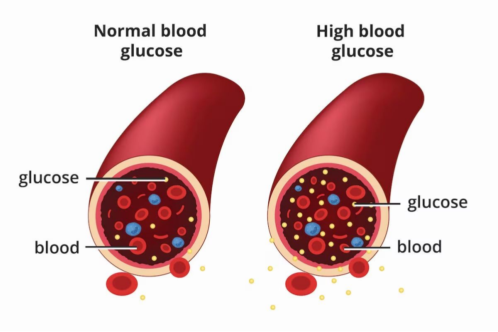
Team Nu: Abdallah Ikbarieh, Prathusha Dinesh, Saksham Sudershan

Georgia Tech

Introduction

Background

- **Diabetes** is a serious **chronic disease** in which individuals **lose the ability** to effectively **regulate levels of glucose** in the blood, and can lead to **reduced quality of life and life expectancy** (e.g., heart disease, vision loss, and kidney disease)
- Statistics in the U.S as of 2021 (NIH, 2024):
  - **38.4 million** people of all ages had **diabetes** (11.6% of the population)
  - **97.6 million** people aged 18 years or older had **prediabetes**
- There is **no cure** for diabetes, but **a better lifestyle** and physical health can help



(https://www.niddk.nih.gov)

Motivation & Problem Statement

- *Motivation:*
  - ➢ **Early detection** of such condition to **delay** or **prevent** the progression to diabetes
  - ➢ Better **patient management** and **treatment strategies**
  - ➢ Reducing **health care costs**

- *Problem statement:*
  - ➢ Classifying individuals as **diabetic**, **prediabetic**, or **non-diabetic** based on a set of **variables** related to **physical and mental health, personal lifestyle, dietary habits, and social aspects** using **machine learning models**

Project Objectives

- *Core Objective:*
  - ➢ Developing a **machine learning model** capable of classifying individuals into **diabetic, prediabetic, or non-diabetic** categories based on **21 variables**
- *Specific Objectives:*
  - ➢ Examining different **classification methods** on the diabetes data set: **AdaBoost, Random Forest, and XGBoost**
  - ➢ Comparing different **resampling methods** on the predictions performance: **undersampling, oversampling, and hybrid resampling**
  - ➢ Implementing Random Forest **Transfer Learning** using 2 different approaches to improve prediction of the **minority** (prediabetes) class

Data Preprocessing and EDA

Data Sources & Characteristics

- The diabetes dataset was obtained from **Kaggle** (https://www.kaggle.com/)
- The dataset is a **snippet** of a Behavioral Risk Factor Surveillance System **(BRFSS) survey** conducted by the Centers for Disease Control and Prevention **(CDC)** of the **year 2015**
- The dataset contains **21 variables** related to **physical and mental health, personal lifestyle, dietary habits, and social aspects of individuals**
- The output variable is the **3 classes** with **253,680 responses**:
  - ➢ **Class 0: no diabetes (213,703)**
  - ➢ **Class 1: prediabetes (4,631)**
  - ➢ **Class 2: diabetes (35,346)**

Georgia Tech

Data Sources & Characteristics

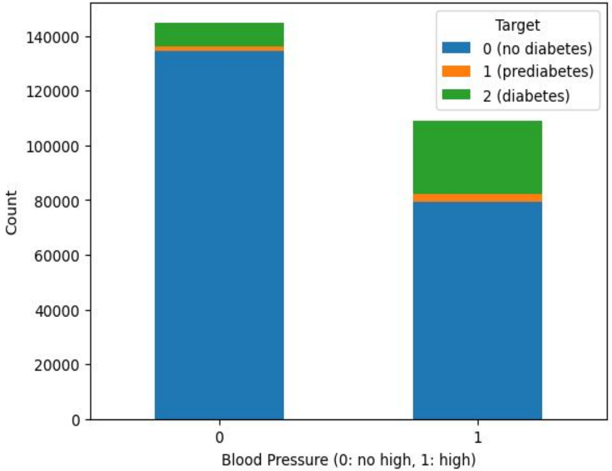
- The **X** variable consist of:
  - ➢ **Health aspects:** Blood pressure, cholesterol level, cholesterol check, body mass index, strokes, heart diseases, overall physical health, mental health, walking capability, physical illness/injury
  - ➢ **Lifestyle:** smoking, physical activity, alcohol consumption
  - ➢ **Diet:** fruits, veggies
  - ➢ **Personal background and social aspects:** sex, age, eduction, income, health care coverage, visiting the doctor (cost)
- Most of the **variables** (~14 variables) were **binary (0 or 1)** and the rest of the were **categorical groups**. So, **no data cleaning/preprocessing was applied**

Georgia Tech

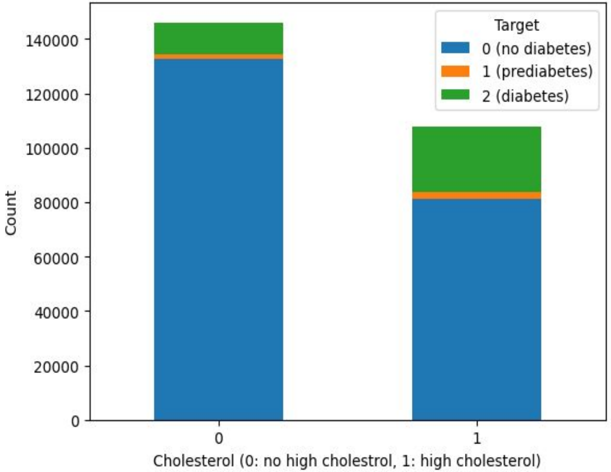
Exploratory Data Analysis

- **Health aspects:**

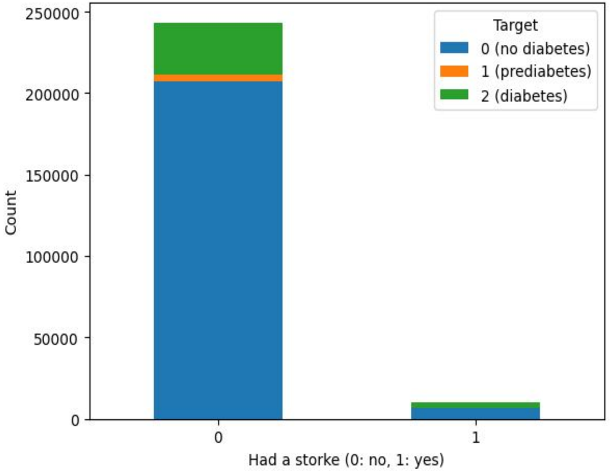
Blood pressure



Cholesterol



Stroke

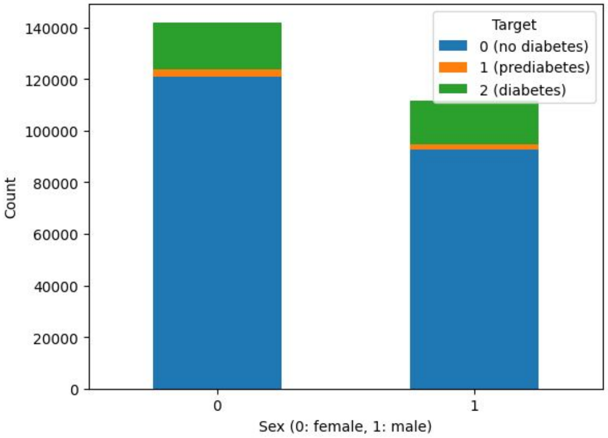


Georgia Tech

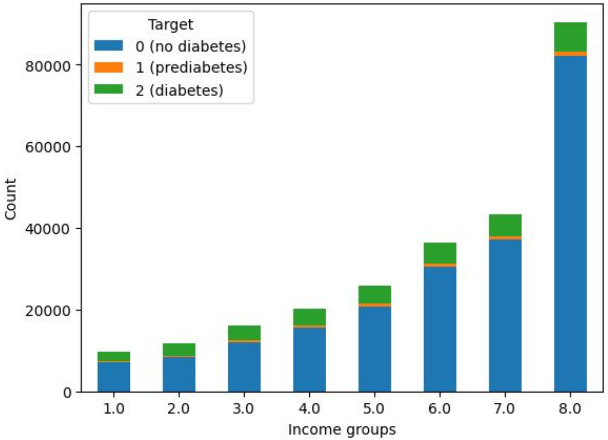
Exploratory Data Analysis

- **Personal background and social aspects:**

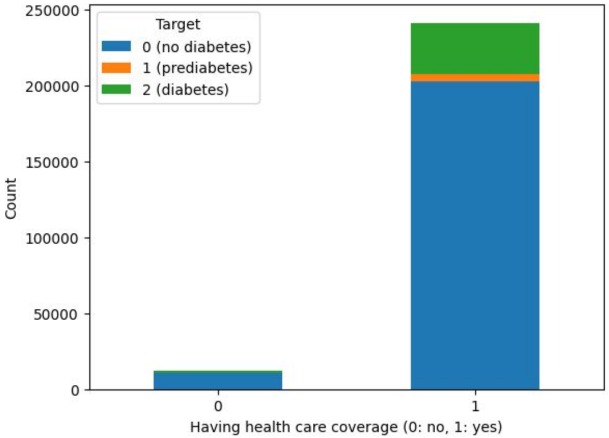
Sex



Income level



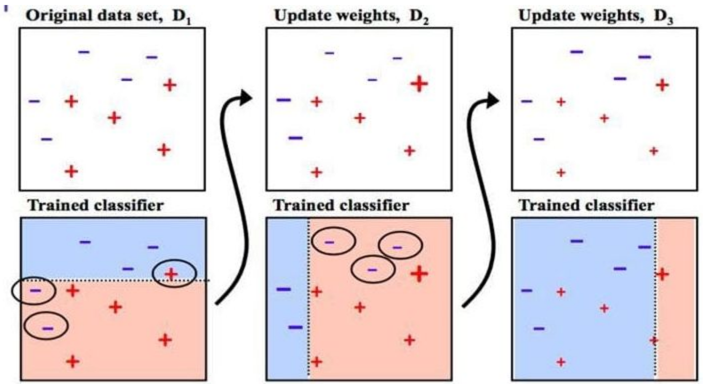
Health care coverage



Methodology

Classification Model 1: AdaBoost

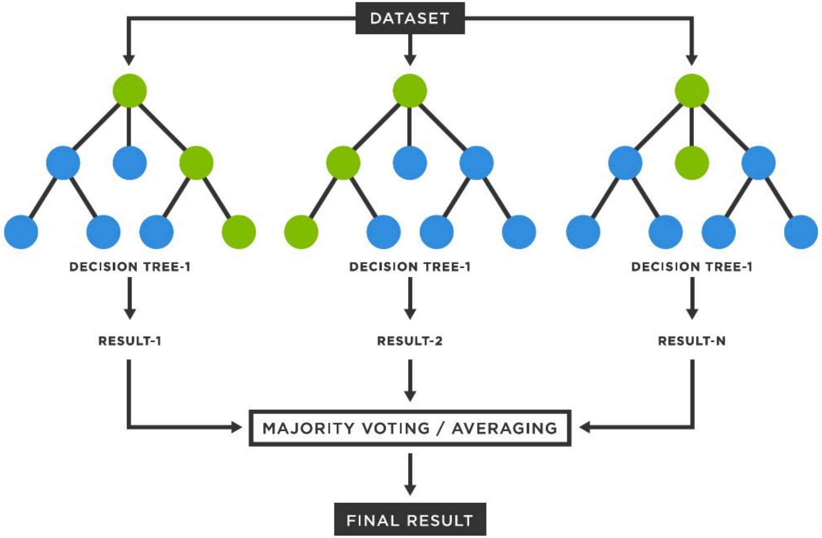
- Combines **multiple weak classifiers** into a **strong one** through **sequential training**
- Starts with a simple base learner (often a decision stump—a tree with a single split) and iteratively **adds models to the ensemble**. Each subsequent model focuses more on training instances that **were misclassified by the previous models**
- The **weights** of **incorrectly classified** instances are **increased** so that the new classifier focuses more on difficult cases
- Prone to **overfitting** when there is **noise** in the data (weak model generalization)



Georgia Tech

Classification Model 2: Random Forest

- Builds multiple r**andom decision trees** and **merges** them to get a **more stable prediction**
- Introduces **randomness** by selecting **random samples** of the features at each split point, which helps in making the model more **robust than a single decision tree**
- **Strong model generalization** even with **noisy data** (excellent due to averaging multiple trees)



Classification Model 3: XGBoost

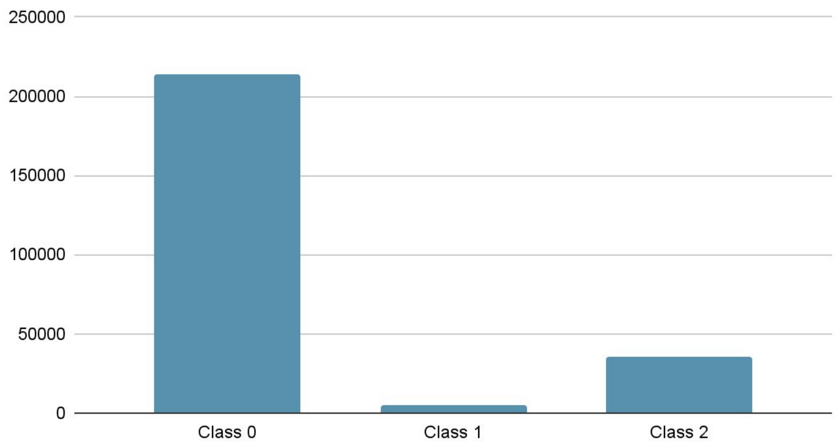
- Stands for **Extreme Gradient Boosting**, is a scalable, distributed gradient-boosted decision tree
- Builds a **strong predictive** model by **combining multiple weaker models**
- Focuses on **minimizing the residual errors** of the **previous models** (i.e., the differences between observed and predicted values)
- Allows you to **control overfitting** by introducing **penalties** on the weights and **biases** of each tree (i.e., regularization)
- Provides a good **balance** between **prediction accuracy** and **computation efficiency**, and supports parallel processing (utilize **multiple CPU cores** to expedite the training of decision trees)

Georgia Tech

Resampling Methods

Our Dataset is extremely unbalanced, with the majority class containing over 45 times as many records as the minority class. Unbalanced Data can lead to the following challenges in classification tasks:

- **Bias towards the Majority Class:** Models minimize prediction error by predicting majority class more frequently
- **Misleading evaluation metrics:** Accuracy is no longer a reliable measure of performance



Resampling Methods: Undersampling

1.1. Random undersampling: **reduces** the number of instances from the **majority class** to match the **minority class**

1.2. NearMiss undersampling: this method selects **majority class samples** based on their **distance** to the **minority class samples**. It aims to retain only those **majority samples** that help in defining **a clear decision boundary** between classes. It either keeps the **nearest** or **furthest** majority samples to enforce a better **class overlap** or **separation**

1.3. Tomek links undersampling: identifies **pairs** of **very close instances** that are of **opposite classes**, known as **Tomek Links**. By removing the **majority class members of these pairs**, the method aims to **increase the separation between classes**, enhancing the classifier's ability to make accurate predictions (**clarifying the decision boundary**)

Georgia Tech

Resampling Methods: Oversampling

2.1. Random oversampling: **duplicates examples** from the **minority class** in the dataset to match the **majority class**

2.2. Adaptive synthetic (ADASYN) oversampling: focuses on **generating synthetic samples** next to the **minority class samples** that are **harder to classify**. It adapts the **number of synthetic samples** based on the **learning difficulty** of each minority sample, thus aiming to balance the class distribution and improve classifier performance on more challenging examples

2.3. Synthetic Minority Over-sampling Technique (SMOTE): creates s**ynthetic samples** for the **minority class** by **interpolating** between existing **minority instances**. It aims to balance the class distribution by augmenting the **minority class** with **new, synthetic samples** derived from feature space similarities, thereby enhancing the **generalization** ability of classifiers.

Resampling Methods: Hybrid Sampling

3.1. SMOTETOMEK: **combines** the **SMOTE** oversampling method with **Tomek Links** undersampling. It first augments the **minority class** using **SMOTE** to create **synthetic examples** and then applies **Tomek Links** to **remove overlapping samples** between classes. The result is a more **balanced dataset** with **clearer class boundaries**, enhancing the effectiveness of classification algorithms

3.2. SMOTEENN: **combines SMOTE** oversampling with the **Edited Nearest Neighbors (ENN)** undersampling, which **removes** any **majority class samples** that are **misclassified** by their **k-nearest neighbors**. This method ensures that **synthetic minority samples** from **SMOTE** are further purified by **removing noisy and borderline majority samples**, thus refining the decision space for better model accuracy

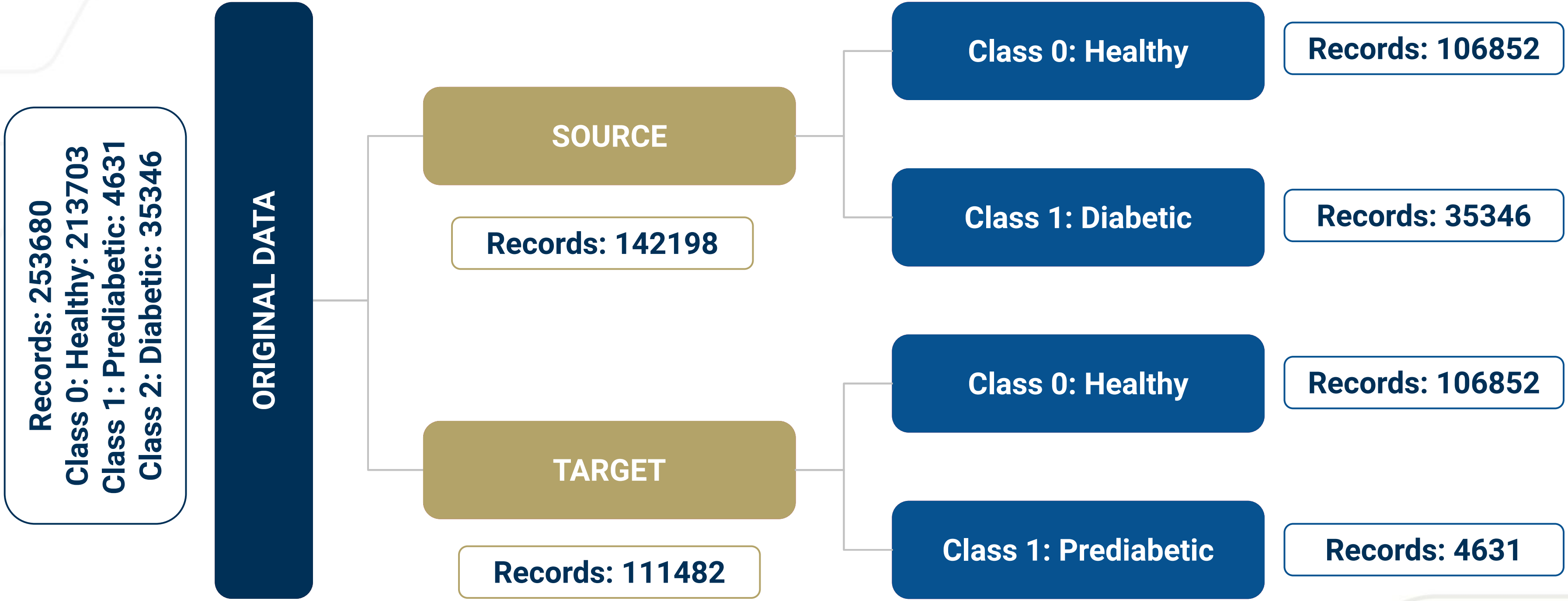
Transfer Learning

In Machine Learning, Transfer Learning is a method by which knowledge gained in one task can be reused or 'transfered' to improve performance for a slightly different task.

We apply Transfer Learning by:

- Splitting the original dataset into 'Source' and 'Target' dataset, where:
  - Source contains only 'Healthy' and 'Diabetic' classes
  - Target contains only 'Healthy' and 'Pre-diabetic' classes
- Training a Random Forest model on the 'Source' Data
- Applying the above model to the 'Target' Data using two different transfer learning methods

Building Source and Target Data for TL



TL Method 1 : Adding Trees to the Forest

For the first Transfer Learning Method, we train a Random Forest on the source data, and add to the forest additional trees trained on the Target data. The steps are as follows:

1. Train a Random Forest Classifier (rf_clf) on the Source train set. Use Hyperparameter Tuning to find the optimal depth and number of estimators to maximize accuracy on the Source test set.
2. Set rf_clf += extra_estimators to add more unbuilt trees to the classifier, that can be fit to the Target set.
3. Train this larger model with extra decision trees on the Target train set. Use this model to make predictions on the Target test set.

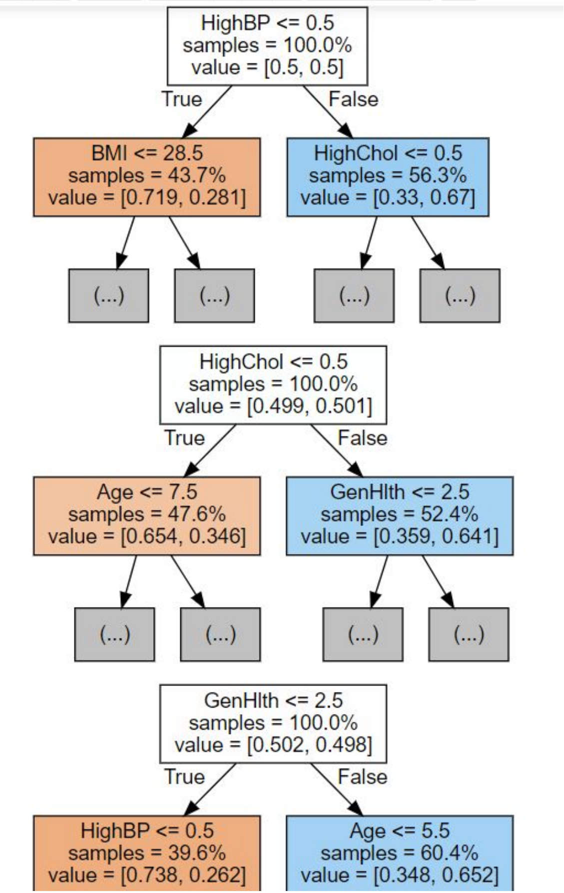
TL Method 2 : Retaining the Forest Structure

For the second Transfer Learning Method, we retain the structure of the Random Forest trained on the Source data and apply it to the Target data. The steps are as follows:

1. Train a Random Forest Classifier (rf_Source) on the Source train set. Use Hyperparameter Tuning to find the optimal depth and number of estimators to maximize accuracy on the Source test set.
2. Initialize a new instance of a Random Forest Classifier (rf_Transfer) with n_estimators, max_depth and seed equal to that of the Random Forest trained on the Source data (rf_Source).
3. Set rf_Transfer.estimators_ = rf_Source.estimators_. rf_Transfer now has the structure of rf_Source.
4. Train the new model (rf_Transfer) on the Target train set. Use this model to make predictions on the Target test set.

Georgia Tech

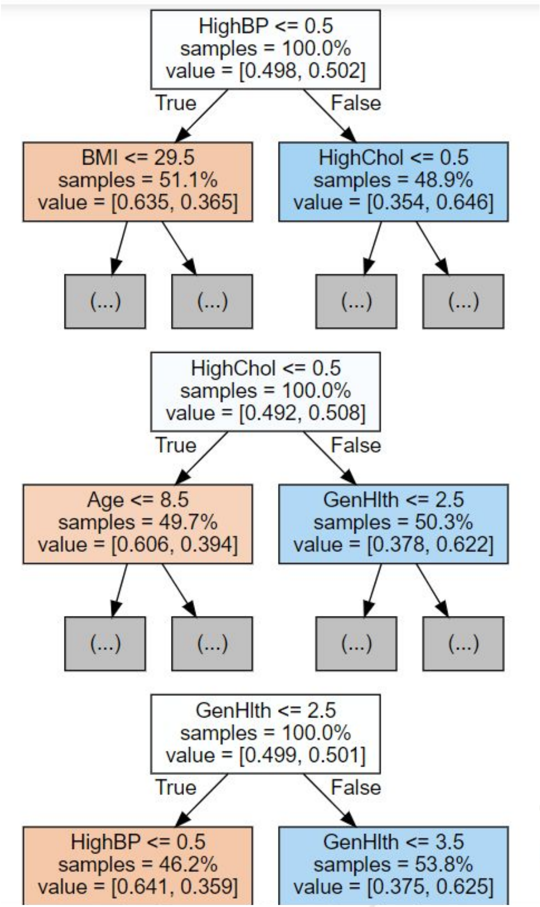
TL Method 2 : Retaining Forest Structure



Random Forest trained on source data

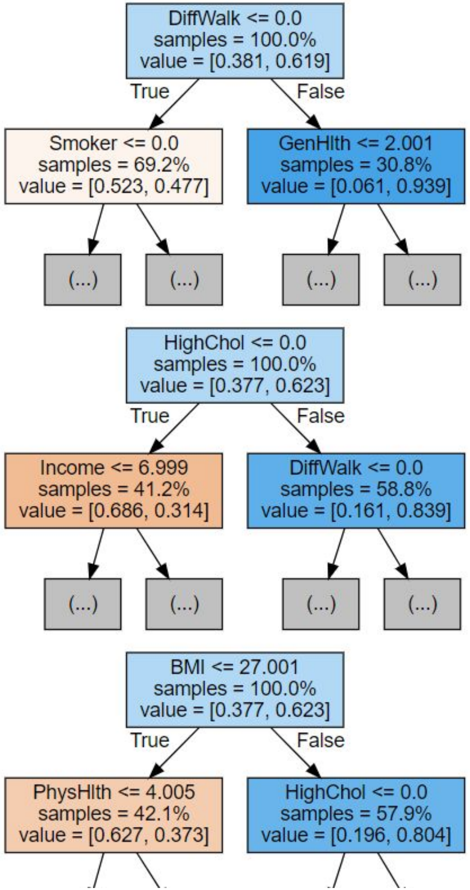
Retain Structure

Train on Target Data



Georgia Tech

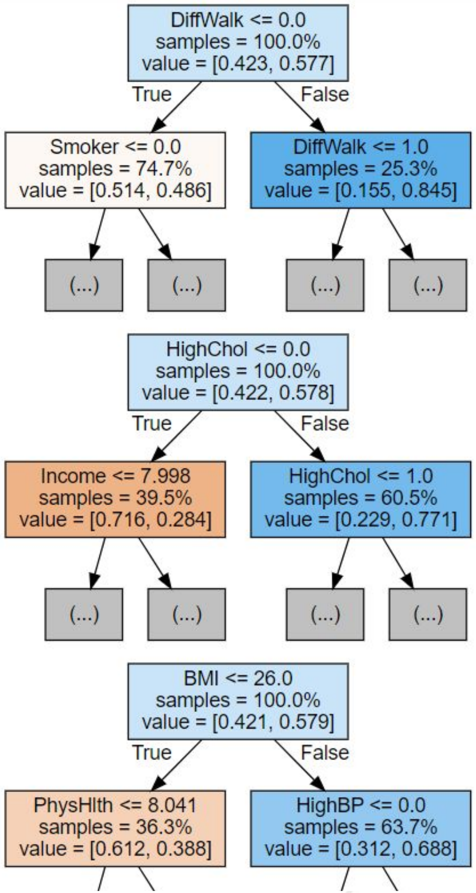
TL Method 2 : Retaining Forest Structure



Random Forest trained on source data

Retain Structure

Train on Target Data



Georgia Tech

Results and Analysis: Classification Methods & Resampling

AdaBoost, Random Forest, and XGBoost

- Imbalanced dataset (with 3 classes)

Method	Class	Training Accuracy	Training Recall	Testing Accuracy	Testing Recall
AdaBoost	Class 0 (no diabetes)	0.85	0.97	0.85	0.97
	Class 1 (prediabetes)		0.00		0.00
	Class 2 (diabetes)		0.21		0.21
Random Forest	Class 0 (no diabetes)	0.99	1.00	0.84	0.97
	Class 1 (prediabetes)		0.94		0.00
	Class 2 (diabetes)		0.96		0.20
XGBoost	Class 0 (no diabetes)	0.86	0.98	0.85	0.98
	Class 1 (prediabetes)		0.01		0.00
	Class 2 (diabetes)		0.24		0.20

Georgia Tech

Resampling Methods

- Balanced dataset (with 3 classes)

Method	Class	Testing Accuracy	Testing Recall
Random Undersampling	Class 0 (no diabetes)	0.58	0.59
	Class 1 (prediabetes)		0.37
	Class 2 (diabetes)		0.54
Adaptive Synthetic (ADASYN)	Class 0 (no diabetes)	0.83	0.93
	Class 1 (prediabetes)		0.00
	Class 2 (diabetes)		0.33
SmoteENN	Class 0 (no diabetes)	0.75	0.77
	Class 1 (prediabetes)		0.01
	Class 2 (diabetes)		0.70

Results and Analysis: Transfer Learning

Classifying Source with RF Trained on Source Data

- Balanced Datasets (2 Classes Each)
  - Source Data - Diabetes vs No Diabetes
  - All Testing Metrics from Source's Test Set

Method	Class	Testing Accuracy	Testing Precision	Testing Recall	Testing F1-Score
Training on RUS Source Data	Class 0 (no diabetes)	0.73	0.91	0.71	0.80
	Class 1 (diabetes)		0.48	0.79	0.59
Training on SmoteENN Source Data	Class 0 (no diabetes)	0.71	0.91	0.69	0.79
	Class 1 (diabetes)		0.46	0.80	0.59

Classifying Target with RF Trained on Source Data

- Balanced Datasets (2 Classes Each)
  - Source Data - Diabetes vs No Diabetes
  - Target Data - Prediabetes vs No Diabetes
  - All Testing Metrics from Target's Test Set

Method	Class	Testing Accuracy	Testing Precision	Testing Recall	Testing F1-Score
Training on RUS Source Data	Class 0 (no diabetes)	0.70	0.98	0.70	0.82
	Class 1 (prediabetes)		0.08	0.64	0.15
Training on SmoteENN Source Data	Class 0 (no diabetes)	0.72	0.98	0.72	0.83
	Class 1 (prediabetes)		0.09	0.67	0.16

Georgia Tech

Classifying Target with RF Trained on Target Data

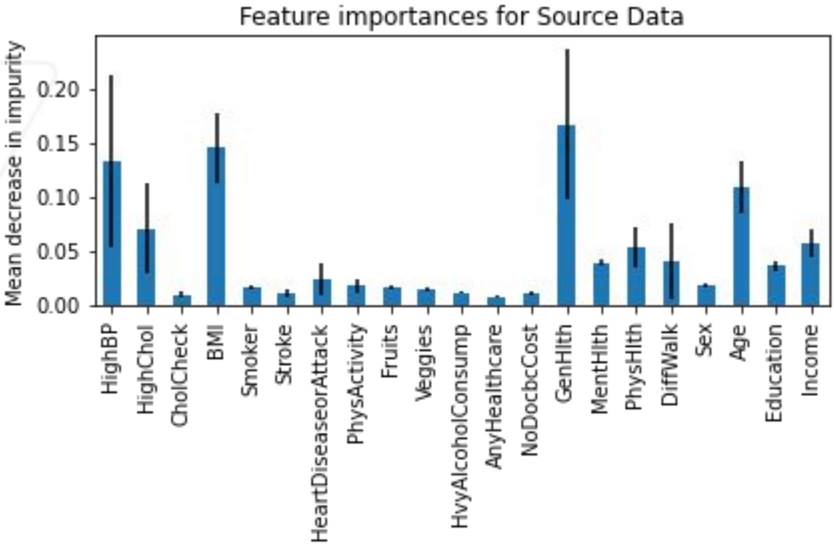
- Balanced Datasets (2 Classes Each)
  - Source Data - Diabetes vs No Diabetes
  - Target Data - Prediabetes vs No Diabetes
  - All Testing Metrics from Target's Test Set

Method	Class	Testing Accuracy	Testing Precision	Testing Recall	Testing F1-Score
Training on RUS Target Data	Class 0 (no diabetes)	0.64	0.98	0.64	0.77
	Class 1 (prediabetes)		0.08	0.76	0.15
Training on SmoteENN Target Data	Class 0 (no diabetes)	0.91	0.96	0.95	0.96
	Class 1 (prediabetes)		0.12	0.16	0.14

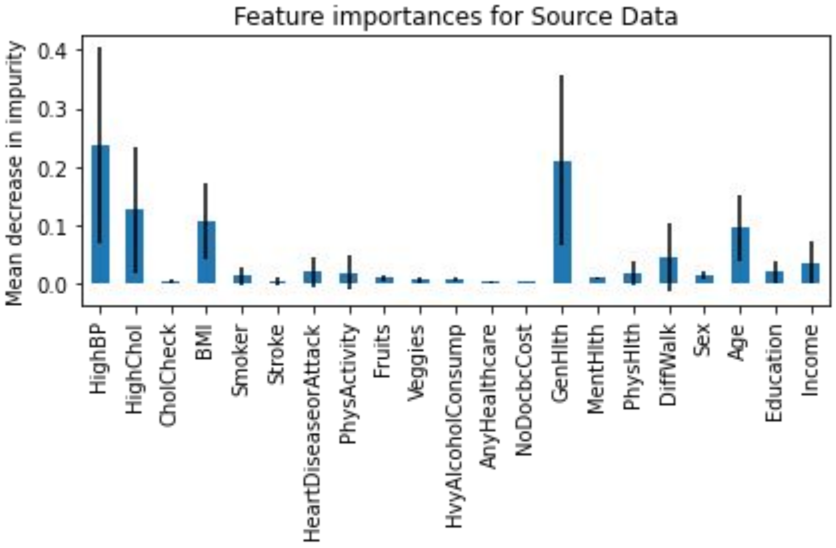
Georgia Tech

Feature Importances for Source Data

With Random Under Sampling



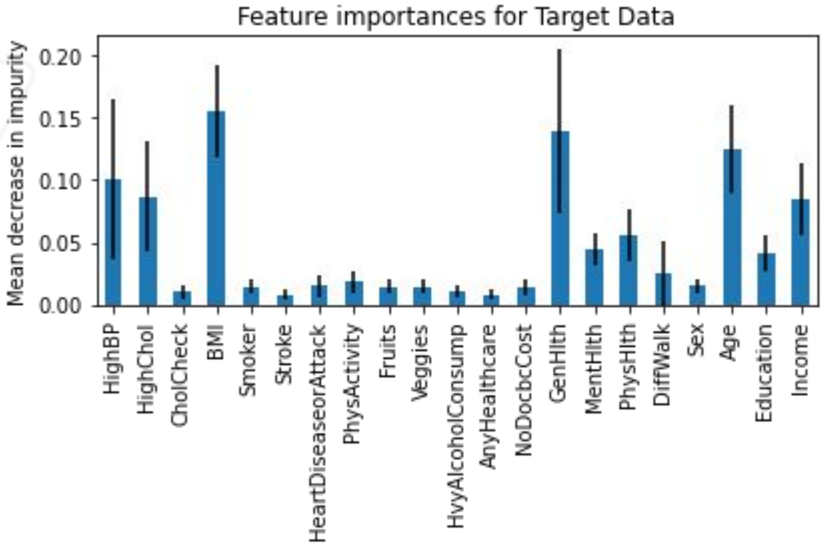
With SmotENN



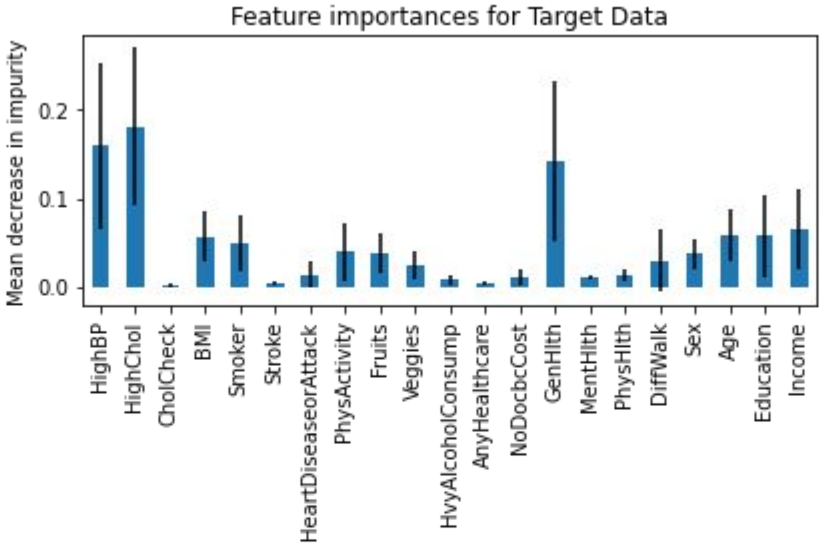
Georgia Tech

Feature Importances for Target Data

With Random Under Sampling



With SmotENN



Transfer Learning with Random UnderSampling

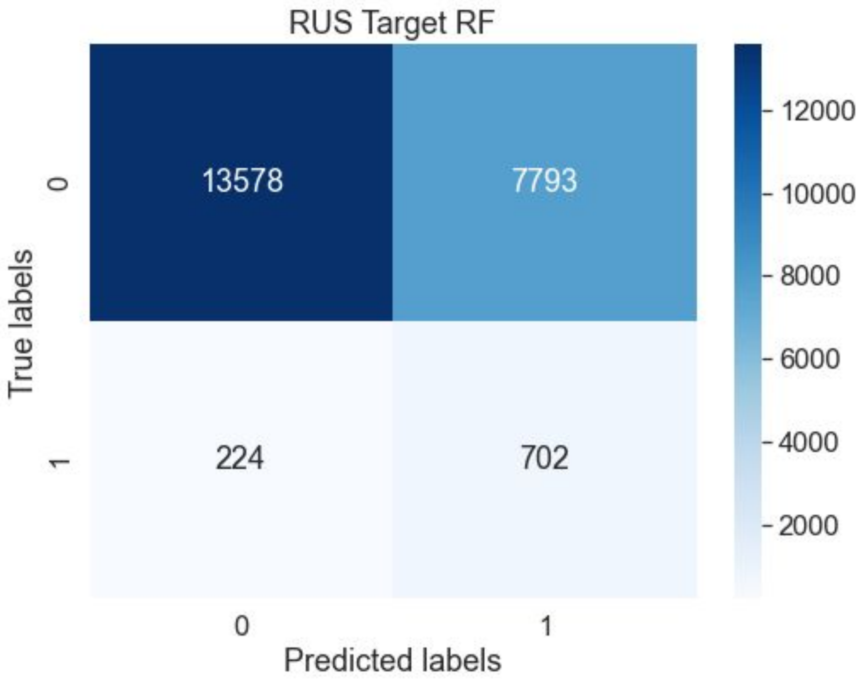
- Balanced Datasets (2 Classes Each)
  - Source Data - Diabetes vs No Diabetes
  - Target Data - Prediabetes vs No Diabetes
  - All Testing Metrics from Target's Test Set

Method	Class	Testing Accuracy	Testing Precision	Testing Recall	Testing F1-Score
Adding 100 Extra Decision Trees	Class 0 (no diabetes)	0.69	0.98	0.69	0.81
	Class 1 (prediabetes)		0.09	0.71	0.15
Retaining Original Forest Structure	Class 0 (no diabetes)	0.63	0.98	0.63	0.77
	Class 1 (prediabetes)		0.08	0.75	0.15

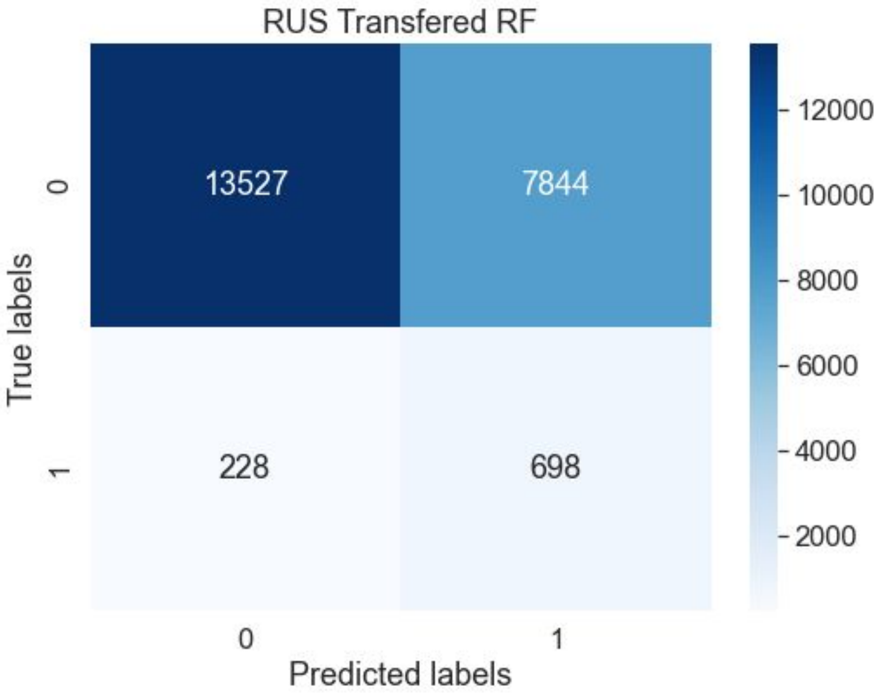
Georgia Tech

Confusion Matrices: Classifying Target Test Set

Traditional Random Forest



Transfer Learning



Transfer Learning with SmoteENN

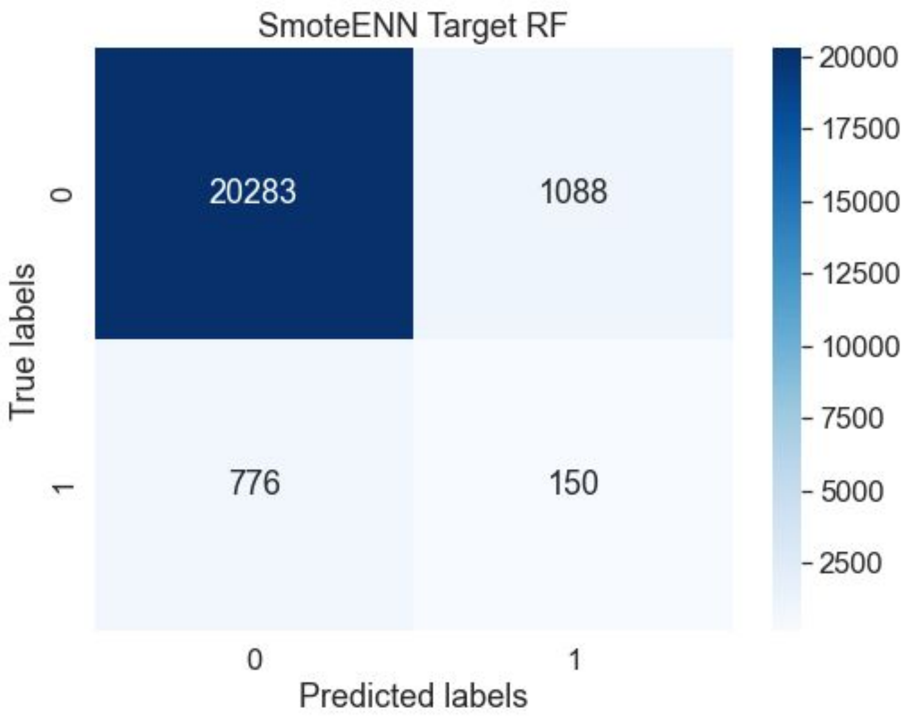
- Balanced Datasets (2 Classes Each)
  - Source Data - Diabetes vs No Diabetes
  - Target Data - Prediabetes vs No Diabetes
  - All Testing Metrics from Target's Test Set

Method	Class	Testing Accuracy	Testing Precision	Testing Recall	Testing F1-Score
Adding 100 Extra Decision Trees	**Class 0 (no diabetes)**	0.85	0.97	0.87	0.92
	Class 1 (prediabetes)		0.11	0.39	0.18
Retaining Original Forest Structure	**Class 0 (no diabetes)**	0.91	0.96	0.95	0.96
	Class 1 (prediabetes)		0.12	0.16	0.14

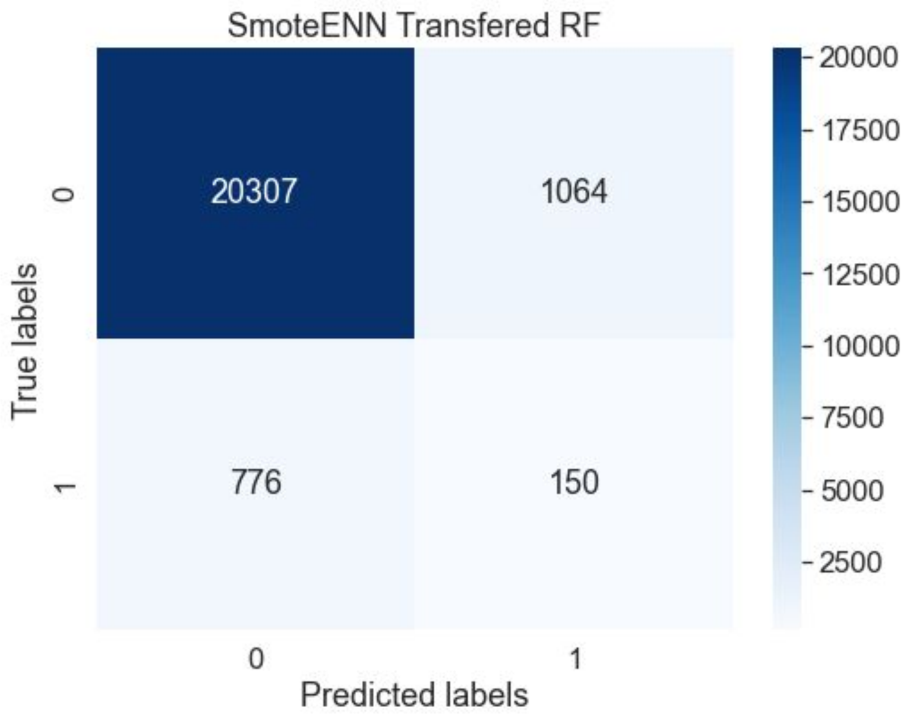
Georgia Tech

Confusion Matrices: Classifying Target Test Set

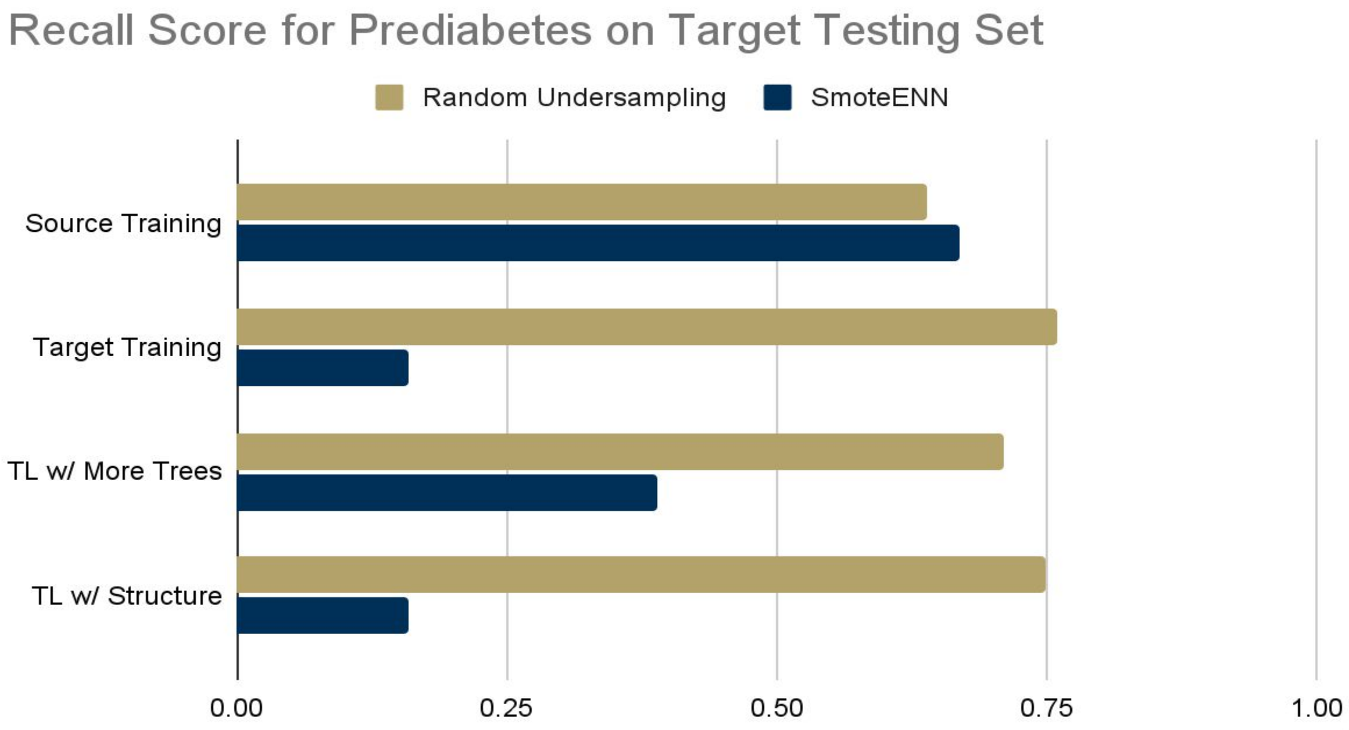
Traditional Random Forest



Transfer Learning



Learnings (Beyond Predictions)



Learnings

Impact of Resampling

- Without resampling, all methods used are unable to classify the minority class on previously unseen data, even after good performance on the training set.
- Oversampling methods were ineffective in improving minority class prediction
- Only Random Undersampling proved to improve minority class prediction

Impact of Transfer Learning

- Transfer Learning's effectiveness differs with the resampling method used.
- With SmoteENN resampled data, both TL methods are able to equal or improve on RF trained on just the Target data
- With RUS data, TL by retaining the structure is almost as effective as an RF trained on the Target data

Conclusions

Main Contributions

Resampling

- We implement 8 different resampling methods and evaluate them using 3 different prediction models.
- We find that Undersampling methods are best for improving the detection of the minority class

Transfer Learning

- In this study, we demonstrate and compare two different methods of Transfer Learning with Random Forests.
- We show that while information gained by training on the Source data is transferable to the Target data, it does not necessarily improve performance

Takeaways

Resampling

- Resampling methods vary in effectiveness based on datasets and tasks. For example, the SmoteENN method works well for balancing the source training set and predicting on the source testing set. However, it does not do well on the target set.
- Resampling after splitting the data gives us poor performance. However, resampling before splitting can lead to data leakage and low generalizability

Transfer Learning

The effectiveness of Transfer Learning is not always guaranteed, but it can be applied in tasks where the basic assumptions are met (Zhang et al., 2021):

- the learning tasks in the two domains are related/similar;
- the source domain and target domain data distributions are not too different;
- a suitable model can be applied to both domains.

Georgia Tech

Broader Impacts

Potential Real-World Applications

- Diagnosing Prediabetes

  Prediabetes is said to usually have no signs or symptoms (Mayo Clinic, 2023). Being able to correctly flag 80% of prediabetic cases can be extremely important!
- Proof-of-work for Transfer Learning in healthcare applications, especially when data sample is small
- Predicting other Chronic Diseases (Heart Disease, Alzheimer's, and Cancer, etc.)

Future Work

- We would propose instance-based approaches to transfer learning such as TrAdaBoost, which reweigh the source training dataset.
- Such models essentially discard the source data points that do not train the model to perform the target task, and thus the effect of negative transfer is minimized and that of positive transfer maximized.