Project Title: Predicting and Preventing Diabetes Using Data

Assignment: Final Written Report

Name: Regina Kang (460, rkang47@gatech.edu)

Submission Date: 04/14/2024

Course Name: ISyE 7406 – Data Mining & Statistical Learning

1. Abstract:

The purpose of this project is to predict whether someone has diabetes or prediabetes using multiple models including baseline models and ensemble methods using the Diabetes Health Indicators Dataset collected by the CDC in 2015. The baseline models used are logistic regression, linear discriminant analysis, Naïve Bayes, and a single tree, and the ensemble methods used are random forest, gradient boosting machine, and extreme gradient boosting (XGBoost). Initially, the only ensemble methods I used were random forest and gradient boosting machine, but after analysis, I decided I needed to test another model, XGBoost. The result of my analysis shows that ensemble methods generally have better predictive accuracy than baseline methods on this data set, with XGBoost performing best among models.

2. Introduction:

Ensemble methods combine predictions from many models to improve performance. They were originally developed to improve the tree based method. There are two types of ensemble methods: re-sampling (bagging, random forest) and re-weighting (boosting). Bagging and random forests create many trees in parallel, with each tree built on a bootstrap dataset, independent of the other trees. The average of these trees allows one to reduce the variance. The main idea of boosting, on the other hand, is to create trees sequentially with each new tree built to correct the misclassification error from the previous grown trees. Then, the weighted average of the trees is used to improve the training error and the hope is that this will lead to improved testing error. In this assignment, I implemented the random forest, gradient boosting machine, and XGBoost models and compared their predictive performance against baseline methods. The main data mining challenges were hyperparameter tuning for the ensemble methods, feature selection for the baseline methods, and cross validation to get a robust comparison of training error, testing error, and area under ROC curve across all models. I was able to overcome these challenges by using k-fold cross validation to choose hyperparameters, using stepwise variable selection with AIC, and effectively storing data across all Monte Carlo cross validations to compare mean metrics across models.

3. Problem Statement / Data Sources:

Diabetes is one of the most common chronic illnesses and one of the leading causes of death in the United States. In 2021, it affected around 38.4 million people or 11.6% of the US population and was also the 8th leading cause of death. There is a significant amount of publicly available data collected by the CDC that I believe we can use to gain greater insights into the factors that are most likely to cause diabetes or prediabetes as well as create models to predict whether a person has diabetes or prediabetes. This will give patients and doctors a powerful tool to predict diabetes or prediabetes early on, and help patients avoid diabetes or prediabetes by focusing on specific factors. Going forward, when I mention predicting diabetes or the factors that cause diabetes, please note that I mean diabetes or prediabetes but am using diabetes as an all-encompassing term.

I got the Diabetes Health Indicators Dataset from Kaggle and you can find the data by following this link:

https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

I used the "diabetes_binary_5050split_health_indicators_BRFSS2015.csv" data set (there are three data sets included in the link) as it is already a balanced data set. The data set contains 70,692 survey responses that the CDC collected in 2015. It has 21 feature variables, and one response variable. The response variable, *Diabetes_binary*, has a value of 0 if the person has no diabetes, and 1 if the person has prediabetes or diabetes. The predictor variables are *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Smoker*, *Stroke*, *HeartDiseaseorAttack*, *PhysActivity*, *Fruits*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *NoDocbcCost*, *GenHlth*, *MentHlth*, *PhysHlth*, *DiffWalk*, *Sex*, *Age*, *Education*, and *Income*. All of these are either binary or integer variables. In Table 1 below, I have included short descriptions of each variable.

Variable	Description
Diabetes_binary	Binary: 0 = no diabetes; 1 = diabetes
HighBP	Binary: 0 = no high BP; 1 = high BP
HighChol	Binary: 0 = no high chol; 1 = high chol
CholCheck	Binary: 0 = no chol check in 5 years; 1 = *
BMI	Integer: body mass index
Smoker	Binary: 0 = smoked ≥ 100 cigs in life; 1 = *
Stroke	Binary: 0 = never had stroke; 1 = had stroke
HeartDiseaseorAttack	Binary: 0 = no CHD or MI; 1 = yes CHD or MI
PhysActivity	Binary: 0 = no phys act in past 30d; 1 = *
Fruits	Binary: 0 = eat fruit < once daily; 1 = *
Veggies	Binary: 0 = eat veg < once daily; 1 = *
HvyAlcoholConsump	Binary: 0 = not heavy drinker; 1 = *
AnyHealthcare	Binary: 0 = no health care coverage; 1 = *
NoDocbcCost	Binary: 0 = *; 1 = no doc bc cost w/i 12mo.
GenHlth	Integer: rating of hlth 1 (excellent) – 5 (poor)
MentHlth	Integer: past 30d - how many bad mental hlth
PhysHlth	Integer: past 30d - how many bad phys hlth
DiffWalk	Binary: 0 = not hard walking/stairs; 1 = *
Sex	Binary: 0 = female; 1 = male
Age	Integer: 13 age categories; 1 = 18-24; 13 = 80+
Education	Integer: education level; 1 (no school) – 6 (col. grad)
Income	Integer: income scale; 1 (<10K) – 8 (≥75K)

Table 1: Variable Descriptions
(* indicates that a binary value of 0 or 1 represents the opposite of what is written for the other value)

You can read more about the feature variables in the link provided above. Since the data set is large, it did not make sense for me to use the entire data set given my computer's computational constraints. I had originally started the project using 0.5% or 1,269 samples of the non-balanced data set "diabetes_binary_health_indicators_BRFSS2015.csv" with 253,680 observations. Please see the Appendix for the original parameters chosen using this data set. However, as I worked through the project, I wanted to make sure I was using more observations and that I was working with a balanced data set. Therefore, I decided to create a stratified sample using 20% of the already balanced data set, which gave me a data set of 14,140 observations.

4. Exploratory Data Analysis & Methodology:

As I mentioned before, my stratified sample of data that I will use going forward for this assignment has 14,140 observations and 22 variables (1 response variable and 21 feature variables). I created a stratified sample using *CreateDataPartition()* from the *caret* library. Before doing any modeling, I wanted to do some exploratory data analysis. To figure out what some of the most important features are in the stratified sample data set (which I will just refer to as data set from now on), I performed stepwise variable selection using AIC in both directions. I found that the model with the lowest AIC included *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Stroke*, *HeartDiseaseorAttack*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *NoDocbcCost*, *GenHlth*, *PhysHlth*, *Sex*, *Age*, and *Income*. I explored the shapes of some of these features (the top four features) as well as the response variable *Diabetes_binary* using histograms (Figures 1 through 5). As you can see, the *Diabetes_binary* histogram clearly indicates that the data set is balanced. I have also created boxplots and included them in the Appendix (Figures A1 through A5).

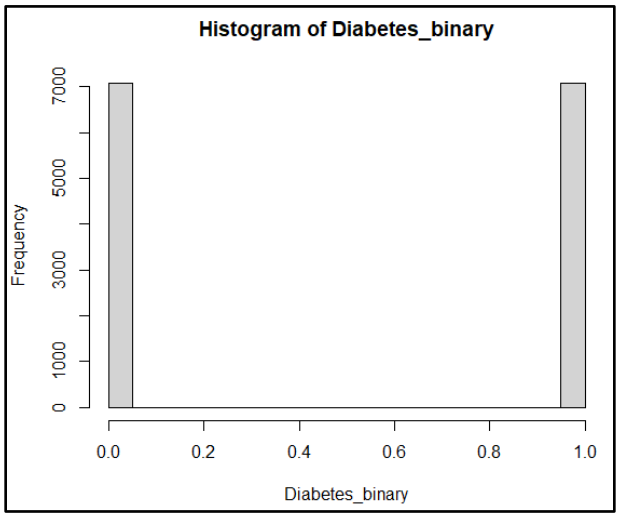


Figure 1: Histogram of *Diabetes_binary* Variable

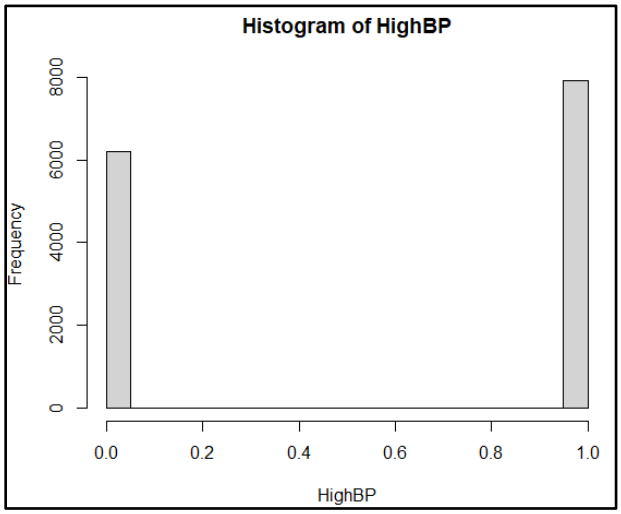


Figure 2: Histogram of *HighBP* Variable

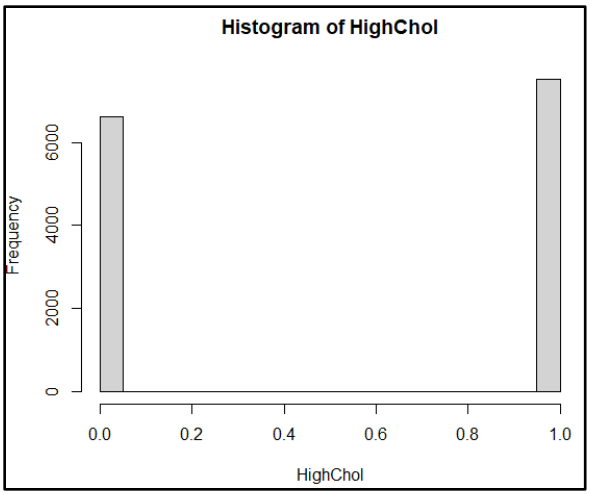


Figure 3: Histogram of *HighChol* Variable

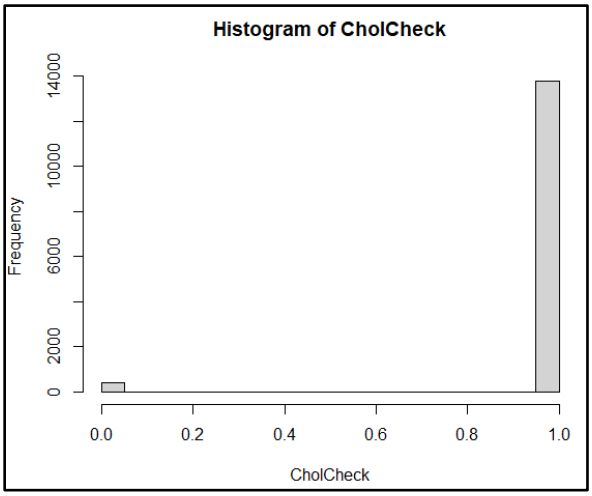


Figure 4: Histogram of *CholCheck* Variable

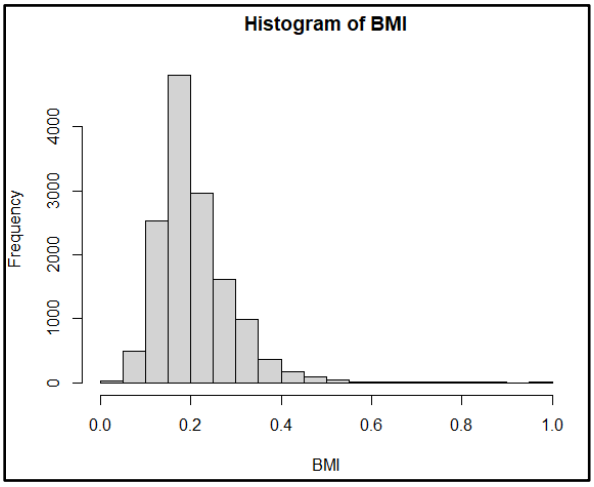


Figure 5: Histogram of *BMI* Variable

However, before deciding on the best predictor variables, I wanted to check if there was any multicollinearity between variables. Therefore, I did a VIF test, and got the results in Table 2 below. As you can see, none of the VIF values are high (greater than 5). In fact, none of the values even exceed 2. Thus, I was able to conclude my feature selection to include all the features chosen by the stepwise regression: *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Stroke*, *HeartDiseaseorAttack*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *NoDocbcCost*, *GenHlth*, *PhysHlth*, *Sex*, *Age*, and *Income*.

Predictor	VIF
HighBP	1.134355
HighChol	1.062496
CholCheck	1.018498
BMI	1.120923
Smoker	1.067143
Stroke	1.056015
HeartDiseaseorAttack	1.119279
PhysActivity	1.137263
Fruits	1.094709
Veggies	1.088437
HvyAlcoholConsump	1.017757
AnyHealthcare	1.105173
NoDocbcCost	1.145124
GenHlth	1.588281
MentHlth	1.269637
PhysHlth	1.647645
DiffWalk	1.438455
Sex	1.096890
Age	1.269804
Education	1.293144
Income	1.460135

Table 2: VIF Values for Predictor Variables

For an overall look at correlation between variables, I plotted a correlation heatmap in Figure 6 below. As you can see, most of the colors are very light, indicating that there is not a lot of correlation between variables. This confirms the results I got in the VIF analysis above. While correlation was generally low, I still wanted to see where correlation was relatively higher, so I also included a filtered correlation plot for absolute correlation values greater than 0.3 in Figure 7. The plot shows that some of the more correlated pairs of variables include *GenHlth* and *PhysHlth*, as well as *Income* and *Education*.

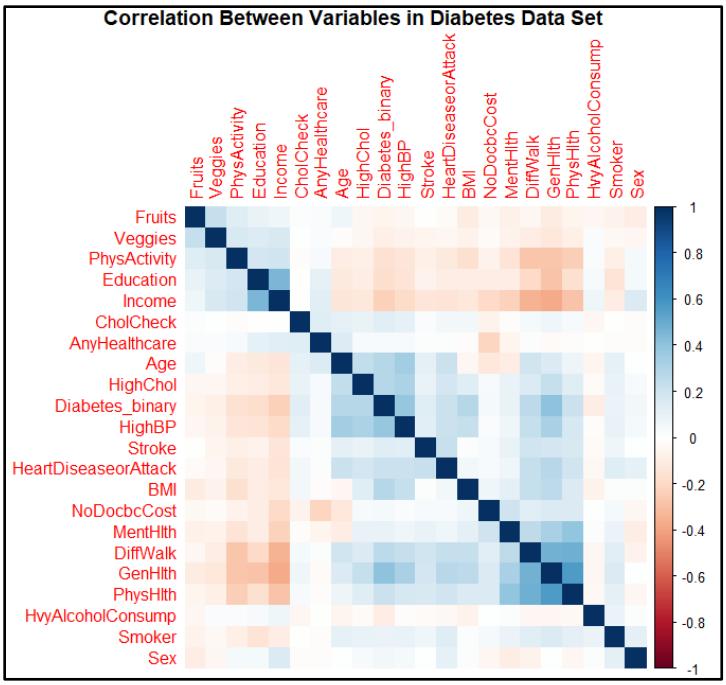


Figure 6: Correlation Between All Variables in Data Set

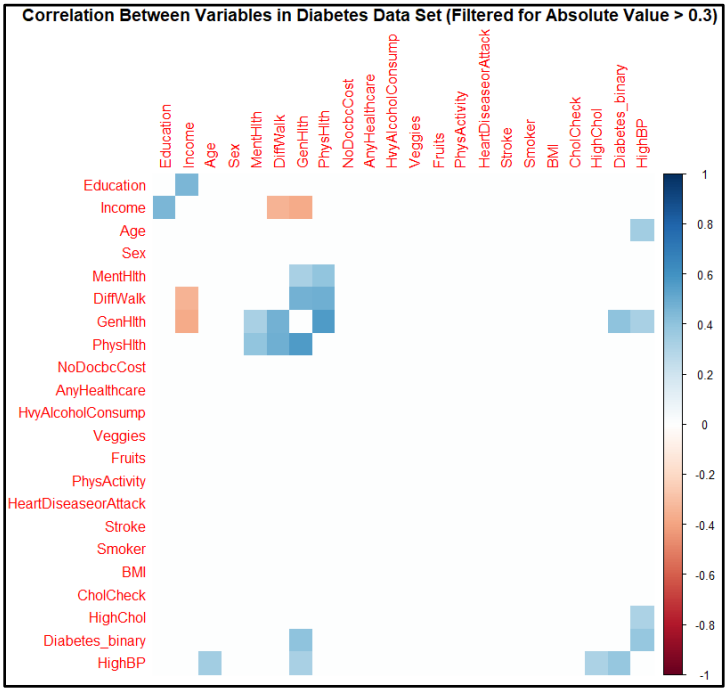


Figure 7: Correlation Plot Filtered for Absolute Correlation > 0.3

After exploring the data, it was time for me to choose which models to implement. Initially, I chose to implement the following models: random forest, gradient boosting machine, logistic regression, linear discriminant analysis, Naïve Bayes, and a single tree. Later, I added XGBoost

as well. I chose these methods because they are all known to be suitable for binary classification. Please see Table 3 for the pros and cons of each model.

Model	Pros	Cons
XGBoost	Regularization to prevent overfitting, parallel processing, fast, efficient, and accurate	Computationally expensive, hard to interpret
Random Forest	Implicit feature selection, robust to outliers, strong predictive accuracy	Black box, hard to interpret, computationally expensive
Gradient Boosting Machine	Good at handling complex data, strong predictive accuracy	Black box, computationally expensive, prone to overfitting
Logistic Regression	Easy to interpret, model coefficients can indicate feature importance	Does not perform well with complex relationships, linear decision boundaries
Single Tree	Simple, intuitive, easy to interpret, can handle non-linear problems	Prone to overfitting, sensitive to changes in data, computationally expensive
Linear Discriminant Analysis	Dimensionality reduction, simple, fast	Sensitive to outliers, prone to overfitting, assumes normal distribution of predictors
Naïve Bayes	Simple, fast, strong predictive performance when assumption of independence holds	Assumption of independence between predictors

Table 2: Pros and Cons of Each Model

I performed hyperparameter tuning on the ensemble methods (random forest, boosting), and used the features chosen from my previous feature selection for the non-ensemble methods (logistic regression, single tree, LDA, Naïve Bayes). Feature selection is done implicitly within the random forest and gradient boosting machine models, so there was no need to do this explicitly. For both ensemble methods, I used k-fold cross-validation to find the best hyperparameter combinations. I based the decision of which hyperparameters to use on only the training set (split again into training and validation sets during k-fold cross validation). I found that for the random forest, the best hyperparameters were ntree = 100, mtry = 1, and nodesize = 4. In classification, the *MeanDecreaseGini* measure is what should be considered when choosing the most important variables, so the results show that the top four most important variables are *GenHlth*, *HighBP*, *BMI*, and *HighChol*. You can find the full list of the most important variables chosen by this model in Figure A6 in the Appendix. Then, I performed hyperparameter tuning to find the optimal hyperparameters for the gradient boosting machine model, with k-fold cross validation built into the *gbm()* function. I found that the best hyperparameters were shrinkage = 0.15 and interaction.depth = 5, that the optimal M number of trees was 42, and that the feature variables with the highest relative influence were *GenHlth*, *BMI*, *HighBP*, and *Age*. Please see Figure A7 in the Appendix for the full list of variables with highest relative influence according to GBM.

While I will explain why in the "Analysis & Results" section, I determined that the models used thus far were not sufficiently strong models in predicting diabetes. Therefore, I decided to create another model, extreme gradient boosting (XGBoost). I performed hyperparameter tuning to find the optimal hyperparameters for the XGBoost model with k-fold cross validation built into the *xgboost()* function. The best hyperparameters were eta = 0.2 , max_depth = 7 , and subsample = 0.9. Feature selection is done implicitly within this model as well. The feature variables with the highest relative influence were *GenHlth*, *HighBP*, *BMI*, and *Age*. As you may have noticed, the most important feature variables for all the ensemble methods have significant overlap, indicating that they produce consistent results. Please see Figure A8 in the Appendix for the full list of variables with highest relative influence according to XGBoost.

To make a robust decision on which model is best, I performed Monte-Carlo cross-validation with B = 50 iterations. I allocated 70% of the data to the training set and 30% to the test set using the *sample()* function, with different random observations chosen for each set in each iteration. I chose the 70-30 split because I needed enough data in the training set to effectively train the models while leaving enough for the test set to thoroughly test them.

5. Analysis & Results:

To create each of the models, I used the following functions and libraries. I have also included the functionalities and assumptions of each model.

1. Logistic Regression: I used the *glm()* function from base R. Logistic regression uses a linear decision boundary to model the probability of a binary response. It assumes that there is no or little multicollinearity between the features.
2. Linear Discriminant Analysis: I used the *lda()* function from the *MASS* library. LDA also uses a linear decision boundary to separate the binary classes. It assumes that within each class, the feature variables are normally distributed and that each class has identical covariance matrices.
3. Naïve Bayes: I used the *naiveBayes()* function from the *e1071* library. Naïve Bayes classifies based on Bayes' theorem. It assumes that within each class, there is independence between features.
4. Single Tree: I used the *rpart()* and *prune()* functions from the *rpart* library. A single tree can create non-linear decision boundaries. It does not make significant assumptions but it is prone to overfitting.
5. Random Forest: I used the *randomForest()* function from the *randomForest* library. The random forest is an ensemble method that creates multiple trees and returns the most common result. It does not make significant assumptions.
6. Gradient Boosting Machine: I used the *gbm()* function from the *gbm* library. GBM is an ensemble method that creates trees sequentially with each new tree built to correct the misclassification error from the previous grown trees. It assumes that the features are independent.
7. XGBoost: I used the *xgboost()* function from the *xgboost* library. XGBoost is like GBM but it has better performance and efficiency. It also assumes the features are independent.

With the hyperparameters for the ensemble methods tuned, I was able to perform Monte-Carlo cross validation with B = 50 iterations to compare the performance of the following methods on this data set: logistic regression, linear discriminant analysis, Naïve Bayes, a single

tree, random forest, gradient boosting machine, and eventually, XGBoost. However, please keep in mind that I added XGBoost to the list of models after all my analysis for the first six models, so I will speak about XGBoost afterwards. For logistic regression, linear discriminant analysis, Naïve Bayes, and the single tree, I used the features I got from feature selection earlier on: *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Stroke*, *HeartDiseaseorAttack*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *NoDocbcCost*, *GenHlth*, *PhysHlth*, *Sex*, *Age*, and *Income*. As I mentioned earlier, I did not explicitly choose features for the ensemble methods as feature selection is done implicitly within them.

Within each Monte-Carlo loop, I randomly chose new training and test sets, and evaluated each model's performance on both the training and test sets for that iteration. Then, I averaged each model's training errors, testing errors and testing variances across iterations to compare performance. For another indicator of performance, I also calculated area under ROC curve for each model evaluated on the test set. Please see Table 4 below for the results for each model (mean training error, mean testing error, mean variance of testing error, and mean area under ROC curve for test set), ordered from lowest mean testing error to highest. As you can see, if you exclude XGBoost for now, gradient boosting machine and random forest had the lowest training errors. It is expected that the ensemble methods had lower mean training errors than the baseline methods because ensemble methods are known to have better predictive accuracy than standard models. However, it was interesting to see that when predicting on the test set, while gradient boosting machine performed better than all baseline models, random forest fell behind logistic regression and LDA. While random forest did have the largest increase when comparing training error to testing error, all models had fairly consistent training and testing results indicating that the models were likely not overfit.

However, I was not content in stopping my analysis here as none of the models were significantly better than the others. This is why I decided to try another ensemble method that is suitable for binary classification, XGBoost. Following the same methodology I used for the other models, I tuned the hyperparameters of this model, making sure to test reasonable values that would not cause instability or overfitting. Once again, I performed Monte-Carlo cross-validation with $B = 50$ iterations using these hyperparameters. XGBoost performed moderately better than all other models, with a mean training error of 0.2232552, a mean testing error of 0.2232720, and a mean AUC-ROC value of 0.7767122. It makes sense that two ensemble methods, GBM and XGBoost, were the strongest performing models, as ensemble methods tend to perform better than baseline methods. While random forest is also an ensemble method, it does not iteratively improve misclassification error like GBM and XGBoost do (using gradient descent). This is why I believe the random forest model did achieve as good of a test error as the other two ensemble methods. GBM and XGBoost were also a great fit for this data set as the main assumption is that the features are independent, which our VIF and correlation analysis showed held true. In addition, while XGBoost and GBM are both boosting models, it makes sense that XGBoost performed better than GBM as XGBoost offers improvements including regularization to prevent overfitting, parallel processing, speed, efficiency, and improved accuracy.

Predictor	Mean Training Error	Mean Testing Error	Mean TE Variance	Mean Area Under ROC Curve
XGBoost	**0.2232552**	**0.2232720**	**3.181151e-05**	**0.7767122**
GBM	0.2388179	0.2472560	2.252156e-05	0.7527769
Log. Regression	0.2506688	0.2509052	2.204886e-05	0.7491268
LDA	0.2509174	0.2512824	2.835416e-05	0.7487532
Random Forest	0.2408022	0.2599057	4.692628e-05	0.7401623
Naïve Bayes	0.2626834	0.2629750	3.746984e-0	0.7370416
Single Tree	0.2743140	0.2801132	8.396540e-05	0.7199460

Table 4: Mean Results for All Models Across B = 50 Monte-Carlo Iterations

6. Conclusions:

My results indicate that XGBoost is the best model to predict diabetes status on this data set. XGBoost gave me the lowest mean training error, mean testing error, and the highest mean AUC-ROC. I have confidence in my results because I used k-fold cross validation to tune my hyperparameters for the ensemble methods. I also used feature selection when creating my baseline methods. This ensured that the inputs going into each of my models were optimized for predictive power. Once I was sure of which hyperparameters to use, I also performed Monte-Carlo cross validation across 50 iterations, and was able to determine that XGBoost performed best.

In context of predicting diabetes, my XGBoost model is most likely to make accurate predictions. However, if one wants to understand what the most predictive features are, a baseline model like logistic regression might be more useful due to its interpretability. While XGBoost also gives you feature importance, it is much harder to interpret because it is built on an ensemble of trees. Therefore, if a person can answer all the questions in the CDC survey and wants to know right away if they are likely to have diabetes, they might want to run their data through my XGBoost model. However, someone who wants to know which areas of their health they should focus on to avoid getting diabetes may want to look at my logistic regression model. My logistic regression model suggests that having high BMI, poor general health, and being older have the strongest relationship with having diabetes. Therefore, to avoid getting diabetes, this person should look to improve these factors, or with unavoidable factors like aging, be conscious that it is a key factor.

If my models were to be implemented for real-world use, I would recommend that the entire dataset be used. I was unable to do so due to computational limitations. While I believe that the XGBoost model can be useful for real-life applications, I would recommend that people using it do not use it as a source of truth as it has an accuracy of around 78%. Since the goal here is to predict a serious illness, an accuracy of around 78% may indicate a need to improve upon the survey questions asked in the questionnaire. However, my models still have the potential to have a significant impact as is. They will give doctors and patients an initial tool to diagnose or self-diagnose diabetes by answering just a few questions (though an official blood test should be used to supplement the diagnosis) and give people data-driven insights into what factors to focus on to avoid diabetes.

The scientific research questions I wanted to address in this project are listed below. In the bullet points under each question, I have either answered the question or stated why the question was not addressed within this project.

1. Are there specific risk factors that are most predictive of and associated with diabetes? If so, can we shorten the survey to only gather data for those factors?
    - Yes, there are specific risk factors that are most predictive of and associated with diabetes. They include: *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Stroke*, *HeartDiseaseorAttack*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *NoDocbcCost*, *GenHlth*, *PhysHlth*, *Sex*, *Age*, and *Income*. If we only collect data for these features, we can still create the same baseline methods. However, the performance of the ensemble methods may not be as strong.
2. Do these survey questions capture enough data to be able to accurately predict if a person has diabetes? If not, do we need to add more questions to the survey? If so, what kinds of questions should be added?
    - I believe these survey questions capture enough data to provide people with an initial tool to check for diabetes, but I do not believe the accuracy is strong enough to use it as a source of truth. I believe adding more questions to the survey could be a solution. Some questions that can be added are: *How much sugar do you consume on an average day? How much salt do you consume on an average day? How much water do you drink on an average day?*
3. Can I use baseline models to accurately predict if a person has diabetes?
    - Baseline models can be used to predict if a person has diabetes, but it should be supplemented with a blood test. Baseline models are only around 75% accurate, so around 25% people will be misclassified.
4. Are there different clusters of people within the data set other than the obvious ones of people with no diabetes, prediabetes, and diabetes? If so, can we gain any significant insights from these clusters?
    - Clusters were not explored within this project due to scope. If this project were to be expanded, I would want to explore clusters as well.
5. Are there any significant differences between people who have prediabetes and diabetes? If so, are there things people with prediabetes can do to prevent getting diabetes?
    - Significant differences between people who have prediabetes and diabetes were not explored in this project because I changed the data set from the three-class data set to the binary data set.
6. Are there ways I can improve on the standard methods to predict if a person has diabetes? Are there ensemble methods that might give me better predictions?
    - Yes, ensemble methods in general give me better predictions than standard methods when predicting if a person has diabetes. While the random forest method did not outperform all standard methods, GBM and XGBoost consistently outperformed all standard methods.
7. In what ways can my findings and model(s) be applied in the real world?
    - Please see the two previous paragraphs within this "Conclusion" section.

Note: The code for this project can be found in the file named "ISyE7406_ProjectCode.R"

7. Conclusions – Lessons Learned:

In this project as well as in this course, I learned the importance of Monte Carlo cross-validation when testing model quality. It was reiterated multiple times throughout the course that model results are not reliable without testing across multiple cross-validations, and I saw this firsthand as I worked through the homeworks and the project. While one model may have performed best when I looked at just one iteration, looking at the average across 50 or 100 iterations often gave me completely different results. I also learned that hyperparameter tuning is essential, and can wildly change model results. Without hyperparameter tuning, we are just guessing the hyperparameters, and the likelihood they would be optimal is extremely low. It is also very important to make sure that you have a balanced data set. In my initial analysis for this project, I did not use a balanced data set. This gave me artificially strong accuracy values that were corrected once I used the balanced data set.

This course provided immense value to me as a data professional as it took me through the concrete steps necessary to analyze and interpret data. The homeworks consistently reiterated necessary steps and this repetition was extremely impactful. Through consistent practice throughout the course, I became much more confident in implementing the data analytics process, and writing reports interpreting my analysis. I would recommend this course to anyone who wants to improve their skills in data analytics as it helps students put into practice what was mostly theoretical in other courses throughout the OMSA program. I feel that this class has made me a much better data analyst/scientist, and I have thoroughly enjoyed this class.

8. Appendix

The random forest parameters tested were ntree values of 100, 200, 300, 400, and 500, mtry values of 1, 2, 3, 4, 5, 6, and 7, and nodesize value of 3, 4, 5, 6, and 7. When I was using 0.5% of the non-balanced data set, the best parameters were ntree = 500, mtry = 5, and nodesize = 7. However, when using the balanced data set, I decided to only allow mtry to be 1 as larger values were causing overfitting. The boosting parameters tested were shrinkage values of 0.05, 0.1, 0.15, and 0.2 and interaction.depth values of 1, 2, 3, 4, and 5. When I was using 0.5% of the non-balanced data set, the best parameters were shrinkage = 0.05 and interaction.depth = 5, and the M number of trees was 70. The XGBoost parameters tested were eta values of 0.01, 0.05, 0.1, and 0.2, max_depth values of 3, 4, 5, 6, and 7, and subsample values of 0.5, 0.6, 0.7, 0.8, and 0.9. I made sure to use eta values that were not too big as big eta values can lead to unstable model behavior. I also made sure the depths were not too large as this can lead to overfitting. When I was using 0.5% of the non-balanced data set, the best parameters were eta = 0.2, max_depth = 7, and subsample = 0.9. When I was using 0.5% of the non-balanced data set, the feature selection chose the following features: *HighBP*, *HighChol*, *BMI*, *HeartDiseaseorAttack*, *GenHlth*, *MentHlth*, and *Age*. The features selected changed significantly when I adjusted the project to look at the balanced data set (20% stratified sample).

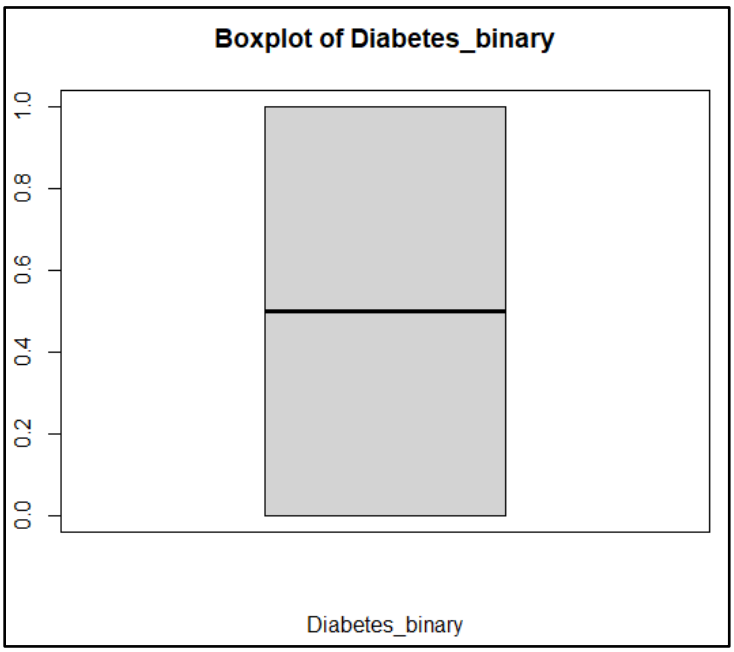


Figure A1: Boxplot of *Diabetes_binary* Variable

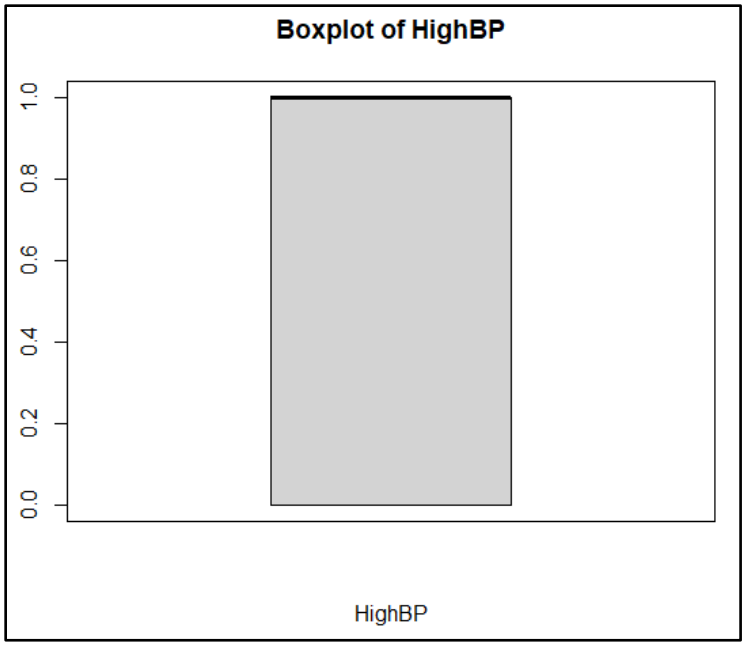


Figure A2: Boxplot of *HighBP* Variable

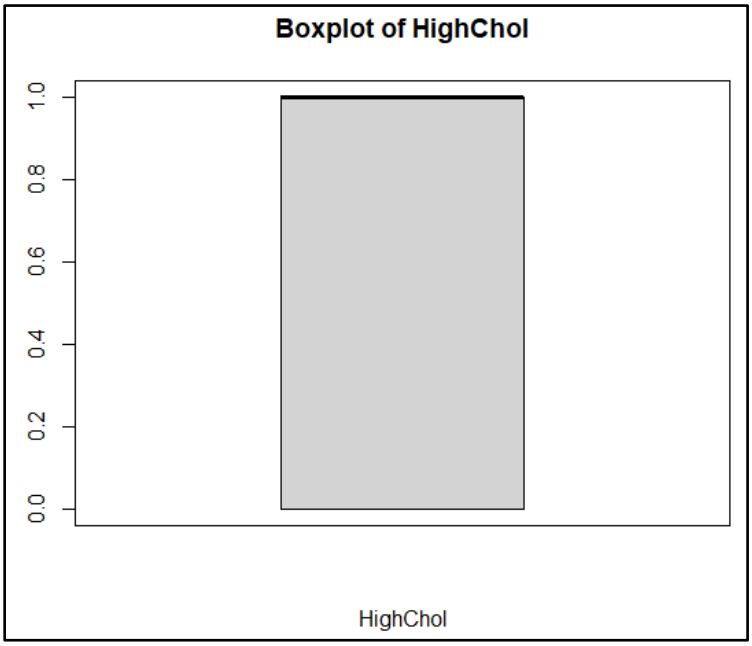


Figure A3: Boxplot of *HighChol* Variable

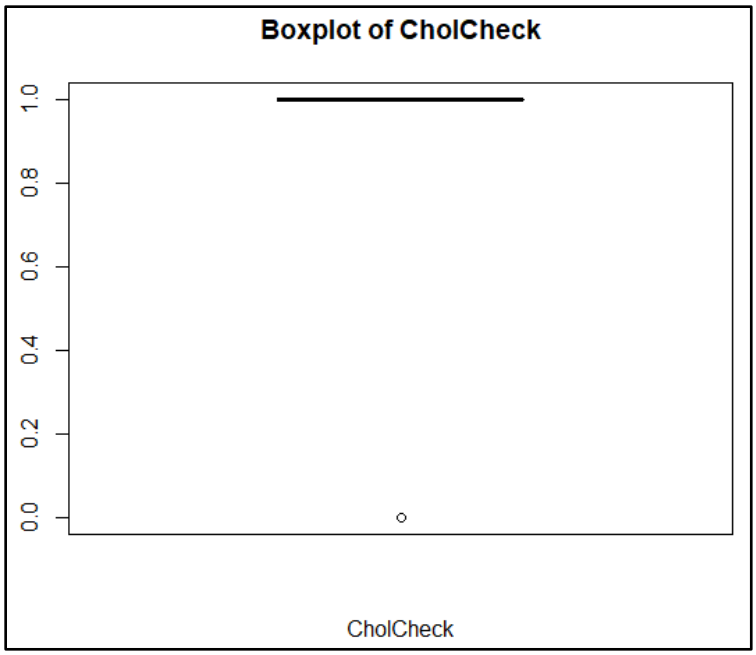


Figure A4: Boxplot of *CholCheck* Variable

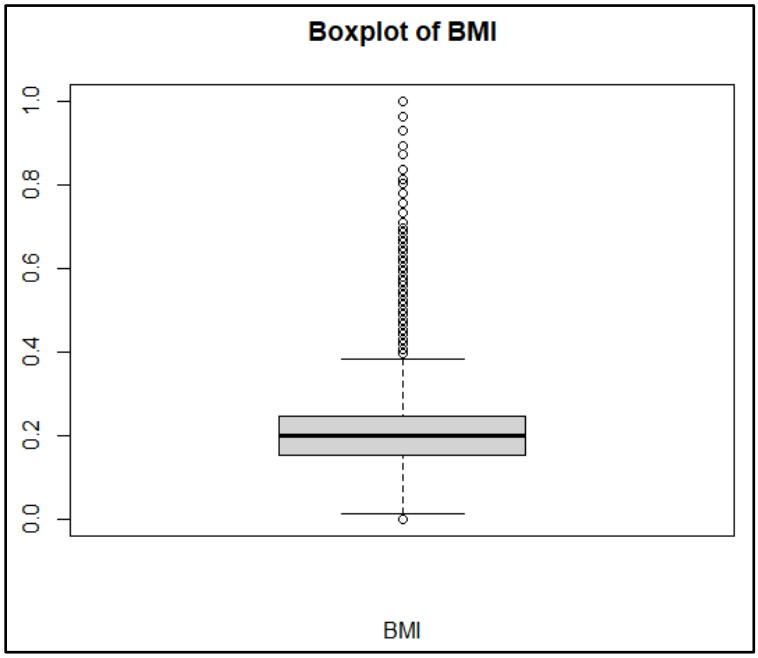


Figure A5: Boxplot of *BMI* Variable

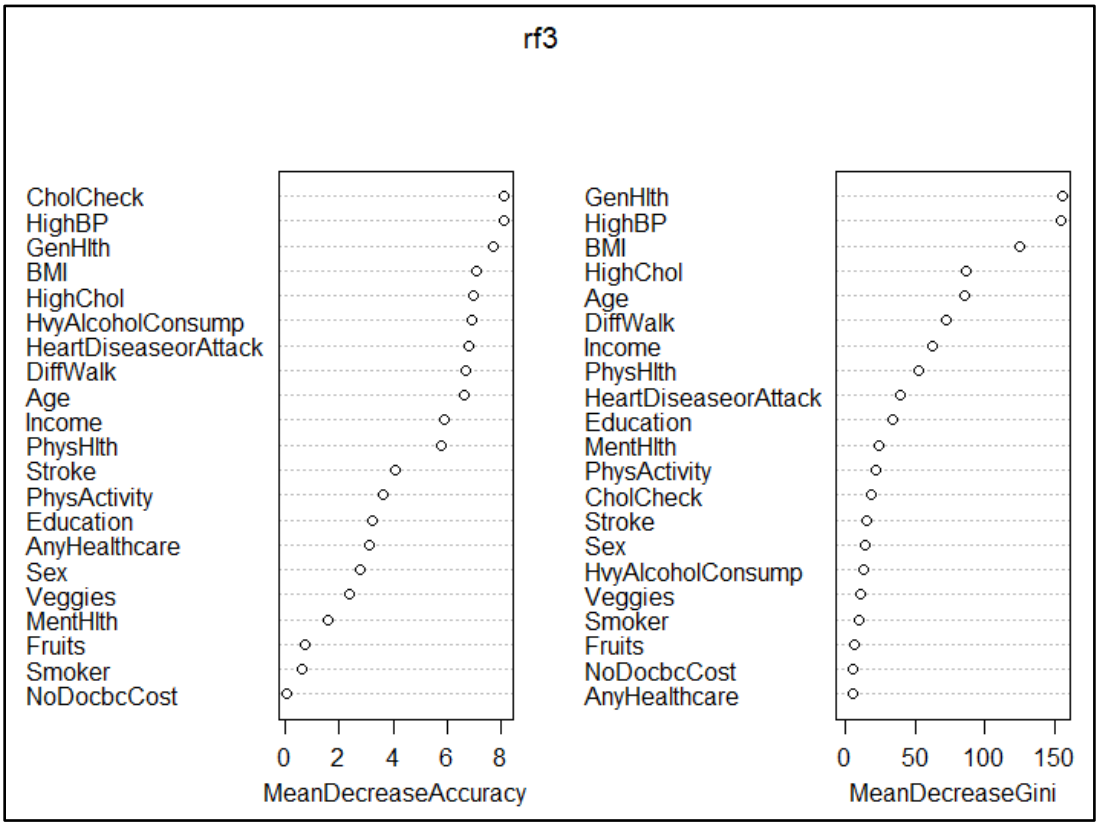
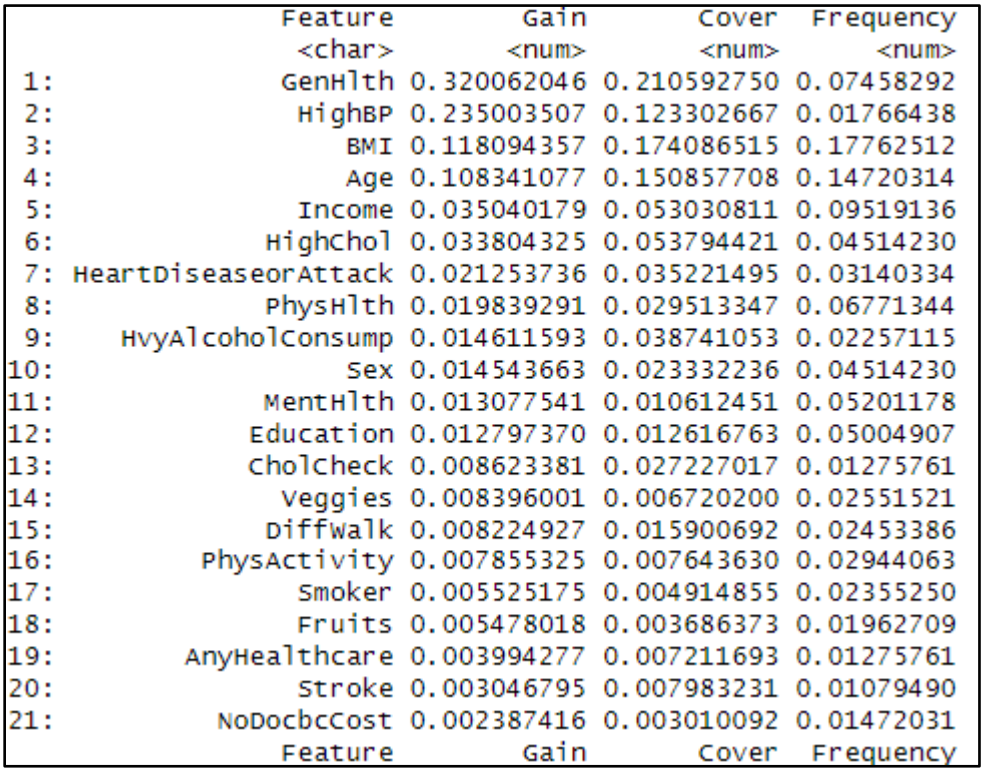


Figure A6: Hyperparameter Tuned RF Features by Importance (Left: Accuracy; Right: Gini)

	var	rel.inf
GenHlth	GenHlth	20.6292076
BMI	BMI	16.2006629
HighBP	HighBP	13.0313213
Age	Age	12.2111150
Income	Income	6.2147670
PhysHlth	PhysHlth	4.9922112
MentHlth	MentHlth	4.6068504
HighChol	HighChol	4.1599132
Education	Education	2.8488365
DiffWalk	DiffWalk	2.3915611
HeartDiseaseorAttack	HeartDiseaseorAttack	2.3863835
Sex	Sex	1.7272573
Veggies	Veggies	1.3170885
HvyAlcoholConsump	HvyAlcoholConsump	1.1689821
Smoker	Smoker	1.0959176
Fruits	Fruits	1.0541734
PhysActivity	PhysActivity	0.9553965
CholCheck	CholCheck	0.8230787
Stroke	Stroke	0.7830484
AnyHealthcare	AnyHealthcare	0.7069764
NoDocbcCost	NoDocbcCost	0.6952514

Figure A7: Feature Variables by Relative Importance for Hyperparameter Tuned GBM



	Feature	Gain	Cover	Frequency
	<char>	<num>	<num>	<num>
1:	GenHlth	0.320062046	0.210592750	0.07458292
2:	HighBP	0.235003507	0.123302667	0.01766438
3:	BMI	0.118094357	0.174086515	0.17762512
4:	Age	0.108341077	0.150857708	0.14720314
5:	Income	0.035040179	0.053030811	0.09519136
6:	HighChol	0.033804325	0.053794421	0.04514230
7:	HeartDiseaseorAttack	0.021253736	0.035221495	0.03140334
8:	PhysHlth	0.019839291	0.029513347	0.06771344
9:	HvyAlcoholConsump	0.014611593	0.038741053	0.02257115
10:	Sex	0.014543663	0.023332236	0.04514230
11:	MentHlth	0.013077541	0.010612451	0.05201178
12:	Education	0.012797370	0.012616763	0.05004907
13:	CholCheck	0.008623381	0.027227017	0.01275761
14:	Veggies	0.008396001	0.006720200	0.02551521
15:	DiffWalk	0.008224927	0.015900692	0.02453386
16:	PhysActivity	0.007855325	0.007643630	0.02944063
17:	Smoker	0.005525175	0.004914855	0.02355250
18:	Fruits	0.005478018	0.003686373	0.01962709
19:	AnyHealthcare	0.003994277	0.007211693	0.01275761
20:	Stroke	0.003046795	0.007983231	0.01079490
21:	NoDocbcCost	0.002387416	0.003010092	0.01472031
	Feature	Gain	Cover	Frequency

Figure A8: Feature Variables by Relative Importance for Hyperparameter Tuned XGBoost

9. Bibliography & Credits

"Diabetes Statistics - NIDDK." *National Institute of Diabetes and Digestive and Kidney Diseases*, U.S. Department of Health and Human Services, www.niddk.nih.gov/health-information/health-statistics/diabetes-statistics). Accessed 25 Mar. 2024.

Mei, Yajun. "Week 1 Module 1 Topics 1 & 2 Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

Mei, Yajun. "Week 4 Module 3 Topics 1 & 2 Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

Mei, Yajun. "Week 5 Module 3 Topics 3 & 4 Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

Mei, Yajun. "Week 8 Module 5 Topic 1 Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

Mei, Yajun. "Week 9 Module 5 Topic 2 (L1 - L6) Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

Mei, Yajun. "Week 10 Module 5 Topic 2 (L7 - L12) Videos." Data Mining & Statistical Learning. Georgia Tech. Atlanta. Lecture.

"Statistics about Diabetes." *Statistics About Diabetes | ADA*, diabetes.org/about-diabetes/statistics/about-diabetes. Accessed 25 Mar. 2024.