STAT652 Project Report

Guo Liang Gan 301311437

December 12, 2017

1 Introduction

This project aims to study the diabetes mellitus incidence among the American Pima Indian population near Phoenix, Arizona, in particular the prediction of the occurrence of diabetes [11]. This data set is consistently being studied since 1965 due to the high prevalence of diabetes in the population. Several studies were conducted on this data set, utilizing an array of different statistical learning or machine learning techniques for prediction. Such methods include neural networks, which were studied by Kayaer et al.[8] and Smith et al.[11], generalized discriminant analysis and least square SVM which were studied by Polat et al[10]. Studies on this data set also served as important references for the forecast of other medical diseases such as tuberculosis [4].

This project has two major components to solve, first the issue of missing values in the dataset and second, the prediction of diabetes. For the problem of missing values, we utilize an imputation technique, MICE(Multivariate Imputation by Chained Equations) [1]. Next, we explored different statistical learning frameworks for the prediction problem, which include logistic regression, generalized additive model (GAM) [7], gradient boosting model (GBM) [5], random forest (RF) and support vector machine (SVM) [3]. We improved each of these models by standard procedures such as statistical tests and cross validation, then choose the model that performs the best on the test set for further improvement. Experiments on predictions models are aimed to adhere to the class materials as much as possible. In other words, we do not explore methods such as deep neural networks and we try to improve the models based on knowledge learned from class.

2 Data

The data set was gathered by the National Institute of Diabetes and Digestive and Kidney Diseases and the database was received in 1990 [11].The observations were female patients who are at least 21 years old of Pima Indian heritage. The data set consists of 768 observations, 500 are positive(diabetes patient) and 268 are negative. There are 8 predictors and a binary response variable, Outcome(1: positive). The predictors are Pregnancies(number of times pregnant), Glucose(plasma glucose concentration), BloodPressure(Diastolic blood pressure (mm Hg)), SkinThickness(Triceps skin fold thickness (mm)), Insulin(2-Hour serum insulin (mu U/ml)), BMI, DPF and Age(years). In particular, Glucose refers to the plasma glucose concentration at 2 hours in an oral glucose tolerance test and the Diabetes Pedigree Function was developed by Smith et al. [11] to encode the expected genetic influence of affected and unaffected relatives on the patient's risk in developing diabetes. Some of the predictors contain missing values as it is absurd to have those measurements to be 0, such predictors are Glucose, BloodPressure, SkinThickness, Insulin and BMI.

3 Methods

Before diving into imputation and prediction, we divided the data into training and test data. We randomly choose 70 observations ($\sim 10\%$) of the data as test set.

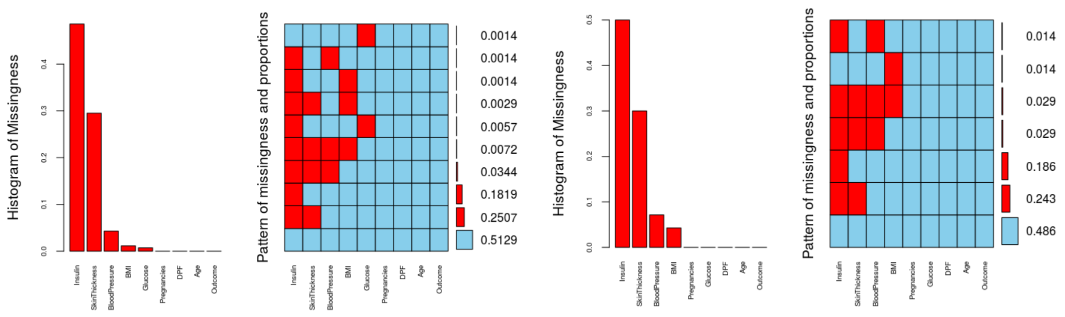

Figure 1: Missingness pattern. (Left)Training set (Right) Test set

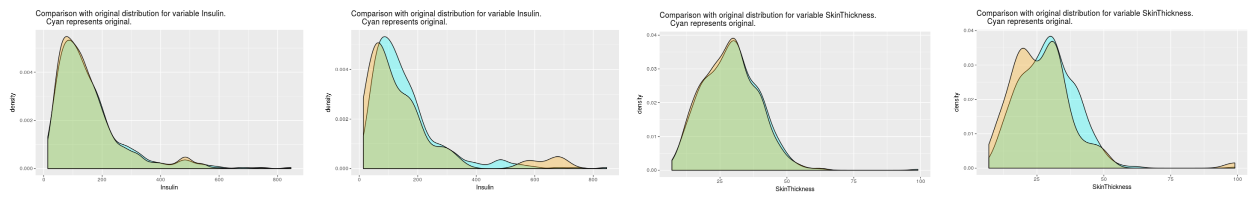

Figure 2: Comparison of the distribution of variables Insulin and SkinThickness between imputed data set and original data set without missing observations. Insulin Variable (1): Training (2): Test, SkinThickness Variable (3): Training (4): Test. Orange represents imputed, cyan represents original

3.1 Imputation

Almost half of the data set contains at least 1 missing value. Figure 1 refers to the distribution and patterns of missing values for the training and test set. We utilize MICE [1] to impute the missing values separately for the training and test set. MICE functions under the assumption that the missing values in the data set are assumed to be missing at random (MAR) [1]. For instance, the occurrence of missing value in Glucose is due to unobserved data but not the fact that the patient is diabetic. Formally, MAR describes the data are missing at random given the data available for used in the imputation model [1]. A brief algorithm of MICE is as follows:

Algorithm 1 MICE

```
1: procedure MICE($D$)
2:     for each variable which contains missing values do
3:         Treat missing values as placeholder and impute with mean
4:     for each variable $i$ which contains missing values do
5:         Relabel placeholder for $i$ as missing, treat $i$ as dependent variable. Fit a regression model to impute these placeholders.
```

The algorithm 1 is repeated for better estimates. We used predictive mean matching as the regression model and direct the reader to the paper [1] for more details. Also, it is advantageous to have more number of imputed data sets based on the percentage of missingness in the data as shown in [6]. However, we only considered one imputed training data set for simplicity and 5 imputed test sets for a less biased estimate of the test accuracy.

The distribution of imputed set and original data set without missing values are similar, as shown in figure 2.

3.2 Scaling and Cross Validation

Before prediction, we scaled the training and test set. We divide the training set into 5-fold cross validation sets using *caret* [9] package in R. Cross validation is used for parameters tuning and feature selection.

3.3 Prediction Models

We explored 5 different models in this project and finally chose SVM for further exploration due to its highest average accuracy on the 5 imputed test sets. We omit the explanation of RF in this section as it gave the lowest test accuracy.

3.3.1 Logistic Regression

We first fit a logistic regression with all main effects and interactions. According to p-values from the statistical tests, there is no sufficient evidence that the interaction terms have significant relationship with the response. Next, we fit a logistic regression model with only main effects and remove any variables which have high p-values in the statistical tests. After conducting feature selection based on p-value, we have a logistic regression model, lm_1 with predictors: *Pregnancies, Glucose, BMI, DPF.* However, if we consider strictly where only variables with $p < 0.001$, we have a model lm_2 with predictors *Pregnancies, Glucose, BMI.* We conducted ANOVA test on model lm_1 and lm_2, which suggested lm_1 is preferred. The average cross validation accuracy for lm_1 is 76.9% and lm_2 is 76.3%, which is not a significant difference. Being parsimonious, we choose lm_2 as a candidate for logistic regression to be tested on the test set.

3.3.2 Generalized Additive Model

Recall that GAM models the estimate of the systematic component of the prediction model as a function of flexible piecewise transformation of the feature space in an additive manner. Formally, it models f as

$$f(\mathbf{X}) = f_1(X_1) + f_2(X_2) + .. + f_p(X_p)$$

where p is the number of predictors, X_i is predictor i and $\{f_j\}$ is a set of basis functions. For our case, we first fit GAM with a natural cubic spline transformation on each predictors. Next, we performed feature selection based on statistical tests on each predictor against the null hypothesis, which assumes the predictor has no significant relationship with the response. We have a GAM model gam_1 with natural cubic spline basis function on *Glucose*, *BMI* and *Age*. Based on ANOVA test on non-parametric effects, we also consider gam_2 which is similar as gam_1 but a linear term for *Glucose* instead of a natural cubic spline transformation. Both gam_1 and gam_2 yield an average accuracy of 75.8% on cross validation. Again, we choose gam_2 as the candidate for GAM due to principle of parsimony.

3.3.3 Gradient Boosting Model

GBM is an ensemble learning (similar to random forest) which relies on a committee of weak classifiers (usually decision trees) for prediction. Each weak classifiers C_i are constructed sequentially *i.e.* C_i depends on C_{i-1}, where C_i are constructed to address problems which are "hard" for C_{i-1}. Again, we performed cross validation to choose the best number of weak classifiers (trees) and the best parameter is 12000 trees.

3.3.4 Support Vector Machine

The general idea behind SVM is to define a decision boundary which maximizes the margin between the boundary and the classes in the feature space, where such boundary is defined by some influential data points which are called support vectors. In this project, we explored soft margin SVM where an observation can be on the wrong side of the margin by at most cost c. We explored different kernels for SVM: *linear, polynomial, radial.* We again find the best parameters for cost c and γ for radial and polynomial kernels using cross validation. We found that linear kernel has the best performance, with average cross validation accuracy 77%, where radial kernel scores 73.9%. The best c parameter for linear kernel is 0.1.

Based on the average test accuracy, we decided to investigate methods to improve SVM. Firstly, we performed principal component analysis(PCA) on the training data during cross validation to choose the best number of PCs. The best number of PCs to use is 8, which is equal to the number of predictors and the average cross validation accuracy was 77%, the same as without PCA. The next approach was to perform forward subset feature selection. The best subset of predictors was all predictors excluding *Pregnancies*, which yielded 77.8% average cross validation accuracy.

3.3.5 Majority Voting

Based on the small increase in cross validation accuracy in SVM model even after PCA and feature selection, we decided to ensemble prediction models which performed sufficiently well in the test accuracy, namely GAM, SVM and GBM.

4 Results

When improving each model, we utilized cross validation for judgment. To compare different models and choosing the model to focus on, we utilize the 5 imputed test sets. Table 1 describes the details of the model and Table 2 describes the test accuracy for each model.

Model	Description
Logistic	Outcome $\sim$ Pregnancies + Glucose + BMI
GAM	Outcome $\sim$ Glucose + s(BMI,4) + s(Age, 4)
RF	Outcome $\sim$. , ntrees=900, subset size=3
GBM	Outcome $\sim$. , ntrees=6000
SVM	Outcome $\sim$. , linear kernel with cost=0.1
SVMBestFeat	Outcome $\sim$. - Pregnancies , linear kernel with cost=0.1
Majority	Ensemble of {GAM, GBM, SVM}

Table 1: Descriptions for models tested on 5 imputed test sets

Method	Test 1	Test 2	Test 3	Test 4	Test 5	Mean	Standard Deviation
Logistic	78.6%	78.6%	78.6%	78.6%	78.6%	78.6%	0
GAM	80%	80%	80%	80%	80%	80%	0
RF	77.1%	75.7%	75.7%	78.6%	75.7%	76.5%	0.0128
GBM	80%	81.4%	80%	78.6%	80 %	80%	0.0101
SVM	80%	80%	80%	80%	80%	80%	0
SVMBestFeat	80%	80%	78.6%	80%	78.6%	79.4%	0.00782
Majority	80%	80%	81.4%	80%	81.4%	**80.6%**	0.00782

Table 2: Accuracy on 5 imputed test sets for each method

4.1 Analysis

Only GAM and logistic regression models are easier to interpret and make inferences. Based on the predictors used for these 2 models, it appears that variable *Glucose* and *BMI* have a significant relationship with the response variable. For the logistic regression fitted on the training set, the coefficient of *Glucose* and *BMI* are 1.107 and 0.661 respectively. In other words, a unit increase in plasma glucose concentration, while holding other variables constant, will lead to 1.107 increase in the log-odds of success (diabetic). Similar explanation can be applied to *BMI*. In both GAM and logistic, it appears that *Glucose* has a linear relationship with the response. Also, we found that GAM model with linear terms for *Glucose* and *BMI* is better than one with natural cubic spline fitted on *BMI* and linear term on *Glucose*, in terms of average cross validation accuracy. The difference is approximately 1%. This may suggest that the transformation of BMI(linear or cubic spline) could be dependent on other variables that were used in the model *i.e.* natural cubic spline on *BMI* is encouraged when *Age* is involved, although statistical tests shown that the interactions are insignificant.

For prediction, Majority Voting scores the highest on the test sets. GAM, GBM and SVM had extremely similar accuracy for validation sets, which could suggest that an ensemble learning of these 3 methods may outperform any of the methods. We also noticed that dimensionality reduction method does not seem to be beneficial for this data set as the number of PCs which contribute to the highest validation accuracy using a SVM model is 8, which is the same as the number of predictors. This point was also complemented by the fact that the size of the best features subset for SVM model is 7, using forward selection.

5 Conclusion and Discussion

In this project, we tackled the prediction of diabetes mellitus of the Pima Indian data set by first imputing values using MICE and next, we performed prediction using different models. Considering individual models, SVM, GBM and GAM scored the highest accuracy in the test sets. However, an ensemble of these 3 models yielded a higher test accuracy. Dimensionality reduction and forward feature selection did not improve SVM, which could suggest more advanced techniques are needed such as penalizedSVM [2] which allows subset selection using LASSO, or if time permits, an exhaustive feature selection. Also, we suggest the readers to experiment using deep neural networks architecture such as multilayer perceptron, autoencoders and decoders etc. if prediction is the objective. However, inferences are also important for researchers in the medical field to draw biological insights about a particular disease.

References

[1] Melissa J. Azur, Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49, feb 2011.

[2] Natalia Becker, Wiebke Werft, Grischa Toedt, Peter Lichter, and Axel Benner. penalizedSVM: a r-package for feature selection SVM classification. *Bioinformatics*, 25(13):1711–1712, apr 2009.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] Orhan Er, Feyzullah Temurtas, and A. Çetin Tanrıkulu. Tuberculosis disease diagnosis using artificial neural networks. *Journal of Medical Systems*, 34(3):299–302, dec 2008.

[5] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, oct 2001.

[6] John W. Graham, Allison E. Olchowski, and Tamika D. Gilreath. How many imputations are really needed? some practical clarifications of multiple imputation theory. *Prevention Science*, 8(3):206–213, jun 2007.

[7] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3), 1986.

[8] Kamer Kayaer and Tulay Yıldırım. Using the adap learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP)*, pages 181–184, 2003.

[9] Max Kuhn. Building predictive models inRUsing thecaretPackage. *Journal of Statistical Software*, 28(5), 2008.

[10] Kemal Polat, Salih Güneş, and Ahmet Arslan. A cascade learning system for classification of diabetes disease: Generalized discriminant analysis and least square support vector machine. *Expert Systems with Applications*, 34(1):482–487, jan 2008.

[11] Smith J. W., Everhart J. E., Dickson W. C., Knowler W. C., and Johannes R.S. Medical diagnosis on pima indian diabetes using general regression neural networks. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 261–265, 1988.

Project652

Guo Liang Gan

November 14, 2017

Libraries and reading data

```
library(ggplot2)
library(mice)
library(dplyr)
library(VIM)
library(gam)
library(caret)
library(randomForest)
library(gbm)
library(e1071)

pima <- read.csv("pima-diabetes.csv")
#summary(pima)
names(pima)[names(pima)=="DiabetesPedigreeFunction"] <- "DPF"
pima$Outcome <- as.factor(pima$Outcome)

#Plausible missing values in Glucose, BloodPressure, SkinThickness, Insulin, BMI,
#as it will be absurd to have value 0 for these variables
```

Visualizing missing patterns for whole data set

```
#Replace 0 with N/A for Glucose, BloodPressure, SkinThickness, INsulin and BMI
missing <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")
pima <- pima %>%
  mutate_at(.vars=missing, funs(replace(.,.==0, NA)))

#missing data patterns using mice
md.pattern(pima)

#Visualize using VIM
miss.plot <- aggr(pima, numbers=TRUE, sortVars=TRUE, cex.axis=.6, labels=names(pima), ylab=
                    c("Histogram of Missingness", "Pattern of missingness and proportions"))
summary(miss.plot)
```

Splitting into training and test sets, visualize missing pattern for test and training set

```
#Split data into training and test set. Test set will not be used until the final stage.
set.seed(19)
```

```r
testSeq <- sample(1:nrow(pima), 70)  #10%
train <- pima[-testSeq,]
test <- pima[testSeq,]

#Distribution of missingness for test and train approx same as whole data set
aggr(train, numbers=TRUE, sortVars=TRUE, cex.axis=.6, labels=names(train), ylab=
       c("Histogram of Missingness", "Pattern of missingness and proportions"))
aggr(test, numbers=TRUE, sortVars=TRUE, cex.axis=.6, labels=names(test), ylab=
       c("Histogram of Missingness", "Pattern of missingness and proportions"))
```

Imputation of missing data using MICE

Retrieve imputed data set and visualize the imputed distribution vs original

Training

```r
#Imputed training data
imp1 <- complete(train.impute, action=1)  #Use only one imputed data set for training

#imp1
#Insulin
ggplot(data=subset(pima, !is.na(Insulin)), aes(x=Insulin)) +
  geom_density(fill="cyan", alpha=0.3) +
  geom_density(data=imp1,fill="orange", alpha=0.3) +
  labs(title="Comparison with original distribution for variable Insulin.
       \n     Cyan represents original.")

#SkinThickness
ggplot(data=subset(pima, !is.na(SkinThickness)), aes(x=SkinThickness)) +
  geom_density(fill="cyan", alpha=0.3) +
  geom_density(data=imp1,fill="orange", alpha=0.3) +
  labs(title="Comparison with original distribution for variable SkinThickness.
       \n     Cyan represents original.")
```

Test

```r
#Imputed test data. Utilize multiple imputed test set for choosing models
imp.test1 <- complete(test.impute, action=1)
imp.test2 <- complete(test.impute, action=2)
imp.test3 <- complete(test.impute, action=3)
imp.test4 <- complete(test.impute, action=4)
imp.test5 <- complete(test.impute, action=5)

#imp.test1
#Insulin
ggplot(data=subset(pima, !is.na(Insulin)), aes(x=Insulin)) +
  geom_density(fill="cyan", alpha=0.3) +
  geom_density(data=imp.test1,fill="orange", alpha=0.3) +
  labs(title="Comparison with original distribution for variable Insulin.
       \n     Cyan represents original.")

#SkinThickness
ggplot(data=subset(pima, !is.na(SkinThickness)), aes(x=SkinThickness)) +
  geom_density(fill="cyan", alpha=0.3) +
  geom_density(data=imp.test1,fill="orange", alpha=0.3) +
  labs(title="Comparison with original distribution for variable SkinThickness.
       \n     Cyan represents original.")
```

Preparation for cross validation sets

```
#scale and center
imp1.scale <- data.frame(scale(imp1[,-9], center=TRUE, scale = TRUE))
imp1.scale$Outcome <- imp1$Outcome

#Cross validation set
#Split to 5-folds for cross validation using caret.
imp1.fold <- createFolds(imp1.scale$Outcome, k=5)

#scale and center
imp.test1scale <- data.frame(scale(imp.test1[,-9], center=TRUE, scale = TRUE))
imp.test1scale$Outcome <- imp.test1$Outcome
imp.test2scale <- data.frame(scale(imp.test2[,-9], center=TRUE, scale = TRUE))
imp.test2scale$Outcome <- imp.test2$Outcome
imp.test3scale <- data.frame(scale(imp.test3[,-9], center=TRUE, scale = TRUE))
imp.test3scale$Outcome <- imp.test3$Outcome
imp.test4scale <- data.frame(scale(imp.test4[,-9], center=TRUE, scale = TRUE))
imp.test4scale$Outcome <- imp.test4$Outcome
imp.test5scale <- data.frame(scale(imp.test5[,-9], center=TRUE, scale = TRUE))
imp.test5scale$Outcome <- imp.test5$Outcome
imp.testScale <- list(imp.test1scale, imp.test2scale,
                      imp.test3scale, imp.test4scale, imp.test5scale)
```

Prediction starts here

Logistic Regression

```
#Compare 5 models: Logistic, GAM, random forest, gbm, svm
#Not much interactions, stick to singletons
lm.fit1 <- glm(data=imp1.scale, Outcome ~ .*., family=binomial())
summary(lm.fit1)

lm.fit2 <- glm(data=imp1.scale, Outcome ~ ., family=binomial())
summary(lm.fit2)

#Remove high p values
lm.fit3 <- glm(data=imp1.scale, Outcome ~ Pregnancies + Glucose + BMI + DPF,
               family=binomial())
summary(lm.fit3)
lm.fit4 <- glm(data=imp1.scale, Outcome ~ Pregnancies + Glucose + BMI ,
               family=binomial())
summary(lm.fit4)
anova(lm.fit4, lm.fit3, test="LRT")

lm3.valacc <- rep(NA,5)
lm4.valacc <- rep(NA,5)

#Cross validate to compare lm3 and lm4
for (i in 1:5){
  val.fit1 <- glm(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~ Pregnancies
                  + Glucose + BMI + DPF, family=binomial())
  val.fit2 <- glm(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~ Pregnancies
                  + Glucose + BMI, family=binomial())
  val.pred1 <- predict(val.fit1, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred2 <- predict(val.fit2, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred1 <- ifelse(val.pred1 > 0.5, 1, 0)
  val.pred2 <- ifelse(val.pred2 > 0.5, 1, 0)
  lm3.valacc[i]<-sum(val.pred1 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred1)
  lm4.valacc[i]<-sum(val.pred2 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred2)
  }
mean(lm3.valacc)
mean(lm4.valacc)  #parsimonious
```

GAM

```
#GAM
gam.fit1 <- gam(data=imp1.scale, Outcome ~ s(Pregnancies,4) +
                  s(Glucose,4) + s(BloodPressure, 4) + s(Insulin,4) +
                  s(SkinThickness,4) + s(BMI,4) + s(DPF,4) + s(Age,4),
                family=binomial())
summary(gam.fit1)

#Remove variables with high p-value
gam.fit1a <- gam(data=imp1.scale, Outcome ~ Pregnancies + s(Glucose,4) +
                   s(SkinThickness, 4) + s(BMI,4) + s(DPF,4) + s(Age, 4),
                 family=binomial())
summary(gam.fit1a)
anova(gam.fit1, gam.fit1a)  #Reject null, prefer smaller model

#Remove next highest p-value
gam.fit1c <- gam(data=imp1.scale, Outcome ~ s(Glucose,4) +
                   s(BMI,4) + s(Age, 4), family=binomial())
summary(gam.fit1c)

#Anova for non-parametric effects suggest linear terms for Glucose
gam.fit1d <-  gam(data=imp1.scale, Outcome ~ Glucose + s(BMI,4) +
                    s(Age, 4), family=binomial())
summary(gam.fit1d)
anova(gam.fit1d, gam.fit1c) #suggest using linear term for Glucose
```

```r
gam1c.valacc <- rep(NA,5)
gam1d.valacc <- rep(NA, 5)

#Cross validate to compare gam.fit1d and gam.fit1c
for (i in 1:5){
  val.fit1 <- gam(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~
                    s(Glucose,4) + s(BMI,4) + s(Age, 4),
                  family=binomial())
  val.fit2 <- gam(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~ Glucose +
                    s(BMI,4) + s(Age, 4), family=binomial())
  val.pred1 <- predict(val.fit1, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred2 <- predict(val.fit2, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred1 <- ifelse(val.pred1 > 0.5, 1, 0)
  val.pred2 <- ifelse(val.pred2 > 0.5, 1, 0)
  gam1c.valacc[i]<-sum(val.pred1 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred1)
  gam1d.valacc[i]<-sum(val.pred2 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred2)
  }

mean(gam1c.valacc)
mean(gam1d.valacc)  #parsimonious

#Linear BMI vs splined BMI
linearBMI.valacc <- rep(NA,5)
splineBMI.valacc <- rep(NA, 5)

for (i in 1:5){
  val.fit1 <- gam(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~
                    Glucose + BMI, family=binomial())
  val.fit2 <- gam(data=imp1.scale[-imp1.fold[[i]], ], Outcome ~
                    Glucose + s(BMI,4), family=binomial())
  val.pred1 <- predict(val.fit1, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred2 <- predict(val.fit2, newdata=imp1.scale[imp1.fold[[i]], ])
  val.pred1 <- ifelse(val.pred1 > 0.5, 1, 0)
  val.pred2 <- ifelse(val.pred2 > 0.5, 1, 0)
  linearBMI.valacc[i]<-sum(val.pred1 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred1)
  splineBMI.valacc[i]<-sum(val.pred2 == imp1.scale[imp1.fold[[i]], "Outcome"])/length(val.pred2)
  }

mean(linearBMI.valacc)
mean(splineBMI.valacc)
```

Random Forest

```r
set.seed(19)
ntrees <- seq(from=100, to=1000, by=50)

rf.valacc <- rep(NA, length(ntrees))

#Choose number of trees based on CV
for (j in 1:length(ntrees)){
  temp <- rep(NA,5)
  for (i in 1:5){
    rf.fit <- randomForest(Outcome ~ ., data=imp1.scale[-imp1.fold[[i]], ],
                           mtry=sqrt(8), ntree=ntrees[j], importance=TRUE)
    rf.pred <- predict(rf.fit, newdata=imp1.scale[imp1.fold[[i]], ])
    temp[i]<-sum(rf.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(rf.pred)
  }
  rf.valacc[j] <- mean(temp)
}

rf.dfvalacc <- as.data.frame(ntrees)
colnames(rf.dfvalacc) <- "NumOfTrees"
rf.dfvalacc$Accuracy <- rf.valacc
ggplot(data=rf.dfvalacc, aes(x=NumOfTrees, y=Accuracy)) +
  geom_point()

#Tried up to 8000 trees, accuracy oscillates around the same region.
#So just focus on smaller number of trees.
#Max
max(rf.valacc)
max.tree <- ntrees[which.max(rf.valacc)]
```

GBM

```r
set.seed(19)
ntrees <- seq(from=5000, to=12000, by=1000)

boost.valacc <- rep(NA, length(ntrees))
#Choose number of trees based on CV
for (j in 1:length(ntrees)){
  temp <- rep(NA,5)
  for (i in 1:5){
    boost.fit <- gbm(as.character(Outcome) ~ .,
                     data=imp1.scale[-imp1.fold[[i]], ], n.trees=ntrees[j],
                     distribution = "bernoulli")
    boost.pred <- predict(boost.fit, newdata=imp1.scale[imp1.fold[[i]], ],
                          n.trees=ntrees[j], type="response")
    boost.pred <- ifelse(boost.pred > 0.5, 1, 0)
    temp[i]<-sum(boost.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(boost.pred)
  }
  boost.valacc[j] <- mean(temp)
}

boost.dfvalacc <- as.data.frame(ntrees)
colnames(boost.dfvalacc) <- "NumOfTrees"
boost.dfvalacc$Accuracy <- boost.valacc
ggplot(data=boost.dfvalacc, aes(x=NumOfTrees, y=Accuracy)) +
  geom_point()

max(boost.valacc)
boost.maxtree <- ntrees[which.max(boost.valacc)]
```

SVM

```r
#Linear
set.seed(19)
costs <- c(10^{-3:1})
svm.linearValAcc <- rep(NA, length(costs))

for (j in 1:length(costs)){
  temp <- rep(NA,5)
  for (i in 1:5){
    svm.fit <- svm(Outcome ~ ., type="C-classification", cost=costs[j],
                   kernel="linear", data=imp1.scale[-imp1.fold[[i]], ])
    svm.pred <- predict(svm.fit, newdata=imp1.scale[imp1.fold[[i]], ])
    temp[i]<-sum(svm.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(svm.pred)
  }
  svm.linearValAcc[j] <- mean(temp)
}

max(svm.linearValAcc)
max.svmcost <- costs[which.max(svm.linearValAcc)]

#radial
costs <- c(10^{-3:3})
gammas <- seq(from=0.5, to=4, by=0.5)
svm.radialValAcc <- matrix(NA, nrow=length(costs), ncol = length(gammas))
for (j in 1:length(costs)){
  for (g in 1:length(gammas)){
    temp <- rep(NA,5)
    for (i in 1:5){
      svm.fit <- svm(Outcome ~ ., type="C-classification", cost=costs[j],
                     gamma=gammas[g], kernel="radial",
                     data=imp1.scale[-imp1.fold[[i]], ])
      svm.pred <- predict(svm.fit, newdata=imp1.scale[imp1.fold[[i]], ])
      temp[i]<-sum(svm.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(svm.pred)
    }
    svm.radialValAcc[j, g] <- mean(temp)
  }

}
max(svm.radialValAcc) #Linear kernel is better
max.radial <- which(svm.radialValAcc == max(svm.radialValAcc), arr.ind = TRUE)
max.radialcost <- costs[max.radial[1]]
max.radialgamma <- gammas[max.radial[2]]
```

Compare all methods on test set

```r
set.seed(19)
best.lm <- glm(data=imp1.scale, Outcome ~ Pregnancies +
                 Glucose + BMI , family=binomial())
best.gam <- gam(data=imp1.scale, Outcome ~ Glucose + s(BMI,4) +
                  s(Age, 4), family=binomial())
```

```r
best.rf <- randomForest(Outcome ~ ., data=imp1.scale, mtry=sqrt(8),
                        ntree=max.tree, importance=TRUE)
best.gbm <- gbm(as.character(Outcome) ~ ., data=imp1.scale, n.trees=boost.maxtree,
                distribution = "bernoulli")
best.svm <- svm(Outcome ~ ., type="C-classification", cost=max.svmcost,
                kernel="linear", data=imp1.scale)

acc <- function(pred, label){
  return(sum(pred == label)/length(pred))
}

test.acc <- matrix(NA, nrow=5, ncol = 5)
for (i in 1:5){
  lm.pred <- predict(best.lm, newdata=imp.testScale[[i]])
  lm.pred <- ifelse(lm.pred >0.5, 1, 0)
  gam.pred <- predict(best.gam, newdata=imp.testScale[[i]])
  gam.pred <- ifelse(gam.pred >0.5, 1, 0)
  rf.pred <- predict(best.rf, newdata=imp.testScale[[i]])
  gbm.pred <- predict(best.gbm, newdata=imp.testScale[[i]],
                      n.trees=boost.maxtree, type="response")
  gbm.pred <- ifelse(gbm.pred >0.5, 1, 0)
  svm.pred <- predict(best.svm, newdata=imp.testScale[[i]])

  test.label = imp.testScale[[i]]$Outcome
  test.acc[1,i] <- acc(lm.pred, test.label)
  test.acc[2,i] <- acc(gam.pred, test.label)
  test.acc[3,i] <- acc(rf.pred, test.label)
  test.acc[4,i] <- acc(gbm.pred, test.label)
  test.acc[5,i] <- acc(svm.pred, test.label)
}

#1:gam, rf, gbm, svm
#2:gam, gbm, svm
#3:gam, gbm, svm
#4:gam, svm
#5:gam, gbm, svm
test.acc <- data.frame(test.acc)
transform(test.acc, SD=apply(test.acc, 1, sd))
transform(test.acc, Mean=apply(test.acc,1, mean))
```

Techniques to improve SVM

PCA

```r
#Summarize PCA information for whole training data
imp1.scalePCA <- prcomp(imp1.scale[,-9])
plot(imp1.scalePCA)
summary(imp1.scalePCA)
biplot(imp1.scalePCA, cex=0.5, scale=0)

#Cross Validate number of PCs and cost
```

```r
costs <- c(10^{-3:3})
numPca <- 1:8
svm.pcaLinear <- matrix(NA, nrow=length(costs), ncol = length(numPca))
for (c in 1:length(costs)){
  for (p in numPca){
    temp <- rep(NA,5)
    for (i in 1:5){
      mat.impVar <- imp1.scale[-imp1.fold[[i]], -9]
      pca <- prcomp(mat.impVar)

      #use p pcs for prediction
      load <- pca$rotation[,1:p]
      load <- as.matrix(load)
      train <- as.matrix(mat.impVar) %*% load
      train <- as.data.frame(train)
      train$Outcome <- imp1.scale[-imp1.fold[[i]], 9]
      mat.valImpVar<- imp1.scale[imp1.fold[[i]], -9]
      val <- as.matrix(mat.valImpVar) %*% load
      val <- as.data.frame(val)
      val$Outcome <- imp1.scale[imp1.fold[[i]], 9]
      svm.fit <- svm(Outcome ~ ., type="C-classification",
                     kernel="linear", cost=costs[c], data=train)
      svm.pred <- predict(svm.fit, newdata=val)
      temp[i] <- sum(svm.pred == val$Outcome)/length(svm.pred)
    }
  svm.pcaLinear[c,p] <- mean(temp)
  }
}
max(svm.pcaLinear)
max.svmpca <- which(svm.pcaLinear == max(svm.pcaLinear),
                    arr.ind = TRUE)  #8 pcs is the best
max.svmpcaCost <- costs[max.svmpca[1]]
max.svmpcaPC <- numPca[max.svmpca[2]]
```

Feature Selection for SVM (Using Forward Selection)

```r
comb2 <- combn(names(imp1.scale),2)
svm.featSelValAcc2 <- rep(NA, dim(comb2)[2])

#Forward selection to choose variables for svm
for (j in 1:dim(comb2)[2]){
  temp <- rep(NA,5)
  for (i in 1:5){
    data <- imp1.scale[-imp1.fold[[i]], comb2[,j]]
    data$Outcome <- imp1.scale[-imp1.fold[[i]], 9]
    svm.fit <- svm(Outcome ~ ., type="C-classification", cost=0.1,
                   kernel="linear", data=data)
    test <- imp1.scale[imp1.fold[[i]], comb2[,j]]
    test$Outcome <- imp1.scale[imp1.fold[[i]], 9]
    svm.pred <- predict(svm.fit, newdata=test)
    temp[i]<-sum(svm.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(svm.pred)
  }
  svm.featSelValAcc2[j] <- mean(temp)
}

best.featSel <- rep(NA,7)
best.featSelAcc <- rep(NA,7)
best.featSel[1] <- list(comb2[,which.max(svm.featSelValAcc2)])
best.featSelAcc[1] <- max(svm.featSelValAcc2)
remain <- setdiff(names(imp1.scale)[-9], best.featSel[[1]])
curr <- best.featSel[[1]]

count <- 2
while (length(remain) != 0){
  svm.acc <- rep(NA, length(remain))
  for(k in 1:length(remain)){
    temp <- rep(NA,5)
    for (i in 1:5){
      data <- imp1.scale[-imp1.fold[[i]], c(curr, remain[k])]
      data$Outcome <- imp1.scale[-imp1.fold[[i]], 9]
      svm.fit <- svm(Outcome ~ ., type="C-classification", cost=0.1,
                     kernel="linear", data=data)
      test <- imp1.scale[imp1.fold[[i]], c(curr,remain[k])]
      test$Outcome <- imp1.scale[imp1.fold[[i]], 9]
      svm.pred <- predict(svm.fit, newdata=test)
      temp[i]<-sum(svm.pred == imp1.scale[imp1.fold[[i]], "Outcome"])/length(svm.pred)
    }
    svm.acc[k] <- mean(temp)
  }

  max.feat <- remain[which.max(svm.acc)]
  curr <- c(curr, max.feat)
  remain <- setdiff(names(imp1.scale)[-9], curr)
  best.featSel[count] <- list(curr)
  best.featSelAcc[count] <- max(svm.acc)
  count <- count + 1
}

max(best.featSelAcc)
bestFeat <- best.featSel[which.max(best.featSelAcc)][[1]]
```

SVM with bestFeat on test sets

```
test.acc <- rep(NA,5)
svm.bestFeat <- svm(Outcome ~ . - Pregnancies, type="C-classification",
                    cost=0.1, kernel="linear", data=imp1.scale)

for (i in 1:5){
  svm.pred <- predict(svm.bestFeat, newdata=imp.testScale[[i]])

  test.label = imp.testScale[[i]]$Outcome
  test.acc[i] <- acc(svm.pred, test.label)
}
```

```r
test.acc
mean(test.acc)
sd(test.acc)
```

Majority voting using GAM, GBM and SVM

```r
test.acc <- rep(NA,5)

for (i in 1:5){
  gam.pred <- predict(best.gam, newdata=imp.testScale[[i]])
  gam.pred <- ifelse(gam.pred >0.5, 1, 0)
  gbm.pred <- predict(best.gbm, newdata=imp.testScale[[i]],
                      n.trees=boost.maxtree, type="response")
  gbm.pred <- ifelse(gbm.pred >0.5, 1, 0)
  svm.pred <- predict(best.svm, newdata=imp.testScale[[i]])

  temp <- data.frame(gam = gam.pred, gbm=gbm.pred, svm=svm.pred)
  maj.pred <- apply(temp, 1, function(x) { sum(sum(x == 1) > sum(x == 0)) })

  test.label = imp.testScale[[i]]$Outcome
  test.acc[i] <- acc(maj.pred, test.label)
}
test.acc
mean(test.acc)
sd(test.acc)
```