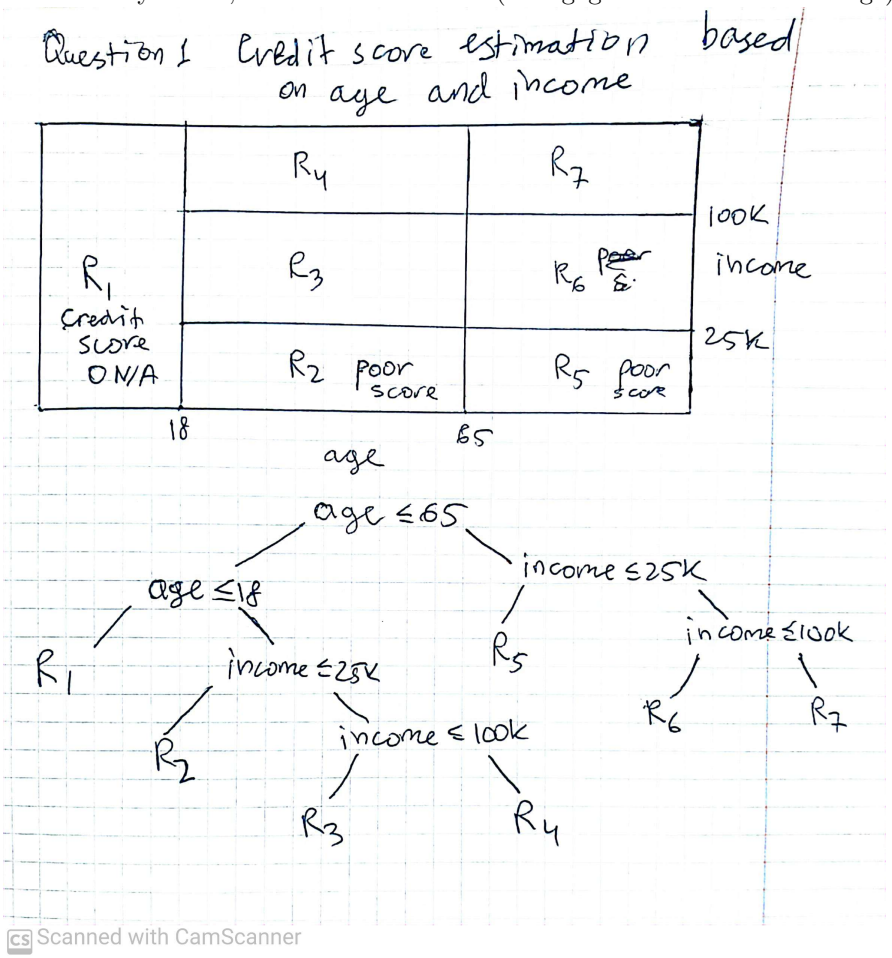
MATH80619A Assignment 2 - Chapter 3 Report

Madi Kassymbekov

Question 1

This is a map of income and age relationship on estimating the credit score. Income cut points are 25k and 100k dollars to segment people into poor, average and wealthy people, while cut points for age are 18 and 65 to segment people into children, adults and senior people. Regions mapped from 1 to 7 mean the following:
R1 - N/A credit score (not applicable for children)
R2 - low income adult, poor credit score
R3 - average income adult, average credit score
R4 - wealthy adult, excellent credit score
R5 - low income senior, poor credit score
R6 - average income senior, average credit score (mortgage not allowed due to age)
R7 - wealthy senior, excellent credit score (mortgage not allowed due to age)



Question 2

Majority vote technique

Most common class given the cutoff value will be selected which in this case is 1 as there are 6 probabilities above 0.5 and 4 under 0.5.

Average probability technique

Under this technique average probability from bootstrapped sample is calculated. In this case it is 0.45. Under the cutoff value of 0.5, the target predicted variable will be classified as 0.

Question 3 R Code tree models comparison

Compute the global error rate, false positive rate and false negative rate on the test set.

```
library(rpart)
library(randomForest)
library(party)
trainset <- read.csv("data/pimadiabetes_train.csv")
testset <- read.csv("data/pimadiabetes_test.csv")
set.seed(37569)
#Find optimal cut-off for binary classification function from notes
bestcutp=function(predcv,y,gainmat=diag(2),cutp=seq(0,1,.02),plotit=FALSE)
{
  nc=length(cutp) ; gain=rep(0,nc)
  for(i in 1:nc)
  {
  pred=as.numeric(predcv>cutp[i])
  gain[i]=mean(gainmat[1,1]*(pred==0)*(y==0)+gainmat[1,2]*(pred==0)*(y==1)+
  gainmat[2,1]*(pred==1)*(y==0)+gainmat[2,2]*(pred==1)*(y==1))
  }
  if(plotit){plot(cutp,gain,type="l",xlab="threshold",ylab="gain")}
  out=list(NULL,NULL)
  out[[1]]=cbind(cutp,gain)
  out[[2]]=out[[1]][which.max(gain),]
  out
}
```

a) Grow a single tree using the CART algorithm based on a cutoff of 0.5 for prediction

```
cart.fit <- rpart(diabetes~., data=trainset, method="class")
cart.predict <- predict(cart.fit, newdata=testset, type="class")
confmatrix <- table(cart.predict, testset$diabetes)
knitr::kable(confmatrix, caption="CART confusion matrix")
```

Table 1: CART confusion matrix

	0	1
0	107	29
1	23	41

```
error_rate <- mean(cart.predict!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results <- data.frame("CART", mean(cart.predict!=testset$diabetes),
```

```
                    confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
                    confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
names(results) <- c("Model", "Error_Rate", "FalsePosRate", "FalseNegRate")
knitr::kable(results, caption="CART Model Performance")
```

Table 2: CART Model Performance

Model	Error_Rate	FalsePosRate	FalseNegRate
CART	0.26	0.1769231	0.4142857

b)Repeat question a) using an optimal cutoff minimizing the global error rate assuming an equal cost between false positive and false negative. In general, can you compare the error rates computed using different gain matrices (answer yes or no, and explain)?

```
predcartprob <- predict(cart.fit, newdata=testset, type="prob")
optcutoff <- bestcutp(predcartprob[,2], testset$diabetes, gainmat=diag(2),
cutp = seq(0,1,.02, plotit=FALSE))
optcutoff[[2]] #Optimal cut-off value
```

```
##  cutp  gain
## 0.200 0.755
```

```
predcartoptimal <- as.numeric(predcartprob[,2]>optcutoff[[2]][1])
confmatrix <- table(predcartoptimal, testset$diabetes)
knitr::kable(confmatrix, caption="CART optimal cut-off confusion matrix")
```

Table 3: CART optimal cut-off confusion matrix

	0	1
0	97	16
1	33	54

```
error_rate <- mean(predcartoptimal!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="CART optimal cut-off",
                                   Error_Rate=mean(predcartoptimal!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[2,], caption="CART optimal cut-off Model Performance")
```

Table 4: CART optimal cut-off Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
2	CART optimal cut-off	0.245	0.2538462	0.2285714

Yes, we can compare error rates of different gain matrices as long as the dataset used to calculate it is the same as well as as the target dependent variable. Otherwise, we can't compare the gain matrices' error rate.

c) Repeat questions a) and b) using a conditional tree

```
ct.fit <- ctree(as.factor(diabetes)~., data=trainset)
ct.predict <- predict(ct.fit, newdata=testset, type="response")
confmatrix <- table(ct.predict, testset$diabetes)
knitr::kable(confmatrix, caption="Conditional Tree confusion matrix")
```

Table 5: Conditional Tree confusion matrix

	0	1
0	102	27
1	28	43

```
error_rate <- mean(ct.predict!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Conditional Tree",
                                   Error_Rate=mean(ct.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[3,], caption="Conditional Tree Model Performance")
```

Table 6: Conditional Tree Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
3	Conditional Tree	0.275	0.2153846	0.3857143

```
#Optimal cutoff conditional tree
predctprob <- predict(ct.fit, newdata=testset, type="prob")
predctprob <- matrix(unlist(predctprob), ncol=2, byrow=TRUE)
optcutoff <- bestcutp(predctprob[,2], testset$diabetes, gainmat=diag(2),
cutp = seq(0,1,.02, plotit=FALSE))
optcutoff[[2]] #Optimal cut-off value
```

```
##  cutp  gain
## 0.600 0.735
```

```
predctoptimal <- as.numeric(predctprob[,2]>optcutoff[[2]][1])
confmatrix <- table(predctoptimal, testset$diabetes)
knitr::kable(confmatrix, caption="Conditional Tree optimal cut-off confusion matrix")
```

Table 7: Conditional Tree optimal cut-off confusion matrix

	0	1
0	122	45
1	8	25

```
error_rate <- mean(predctoptimal!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Conditional Tree optimal cut-off",
                                   Error_Rate=mean(predctoptimal!=testset$diabetes),
```

```
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[4,], caption="Conditional Tree Optimal cutoff Model Performance")
```

Table 8: Conditional Tree Optimal cutoff Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
4	Conditional Tree optimal cut-off	0.265	0.0615385	0.6428571

d) Grow a random forest with B=500 trees using the CART algorithm and a cutoff of 0.5

```
rf.fit <- randomForest(as.factor(diabetes)~., data=trainset, ntree=500)
rf.predict<- predict(rf.fit, newdata=testset, type="response")
confmatrix <- table(rf.predict, testset$diabetes)
knitr::kable(confmatrix, caption="Random Forest CART based confusion matrix")
```

Table 9: Random Forest CART based confusion matrix

	0	1
0	111	34
1	19	36

```
error_rate <- mean(rf.predict!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Random Forest CART",
                                   Error_Rate=mean(rf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[5,], caption="Random Forest CART based Model Performance")
```

Table 10: Random Forest CART based Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
5	Random Forest CART	0.265	0.1461538	0.4857143

e) Grow a random forest with B=500 trees using the conditional tree algorithm and a cutoff of 0.5

```
cf.fit <- cforest(as.factor(diabetes)~., data=trainset,
controls = cforest_control(ntree = 500))
cf.predict<- predict(cf.fit, newdata=testset, type="response")
confmatrix <- table(cf.predict, testset$diabetes)
knitr::kable(confmatrix, caption="Random Forest Conditional Tree based confusion matrix")
```

Table 11: Random Forest Conditional Tree based confusion matrix

	0	1
0	110	32
1	20	38

```
error_rate <- mean(cf.predict!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Random Forest CTree",
                                   Error_Rate=mean(cf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[6,], caption="Random Forest Conditional Tree Model Performance")
```

Table 12: Random Forest Conditional Tree Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
6	Random Forest CTree	0.26	0.1538462	0.4571429

f) Repeat questions d) and e) using the bagging approach

```
bag.rf.fit <- randomForest(as.factor(diabetes)~., data=trainset,
mtry=8,
ntree=500)
bag.rf.predict <- predict(bag.rf.fit, newdata=testset)
confmatrix <- table(bag.rf.predict, testset$diabetes)
knitr::kable(confmatrix, caption="Random Forest CART Bagging based confusion matrix")
```

Table 13: Random Forest CART Bagging based confusion matrix

	0	1
0	108	30
1	22	40

```
error_rate <- mean(predcartoptimal!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Random Forest CART Bagging",
                                   Error_Rate=mean(bag.rf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[7,], caption="Random Forest CART Bagging based Model Performance")
```

Table 14: Random Forest CART Bagging based Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
7	Random Forest CART Bagging	0.26	0.1692308	0.4285714

```
#CTree Based RF Bagging
bag.cf.fit <- cforest(as.factor(diabetes)~., data=trainset,
                          controls = cforest_control(ntree = 500,mtry=8))
bag.cf.predict <- predict(bag.cf.fit, newdata=testset)
confmatrix <- table(bag.cf.predict, testset$diabetes)
knitr::kable(confmatrix, caption="Random Forest CTree Bagging based confusion matrix")
```

Table 15: Random Forest CTree Bagging based confusion matrix

	0	1
0	109	34
1	21	36

```
error_rate <- mean(bag.cf.predict!=testset$diabetes)
fpr <- confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"])
fnr <- confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"])
results[nrow(results) + 1,] = list(Model="Random Forest CTree Bagging",
                                   Error_Rate=mean(bag.cf.predict!=testset$diabetes),
FalsePosRate=confmatrix["1","0"]/(confmatrix["1","0"]+confmatrix["0","0"]),
FalseNegRate=confmatrix["0","1"]/(confmatrix["0","1"]+confmatrix["1","1"]))
knitr::kable(results[8,], caption="Random Forest CTree Bagging based Model Performance")
```

Table 16: Random Forest CTree Bagging based Model Performance

	Model	Error_Rate	FalsePosRate	FalseNegRate
8	Random Forest CTree Bagging	0.275	0.1615385	0.4857143

g) Compare the results. Compute the variable importance measures based on the best model.

```
knitr::kable(results, caption="Models Performance Comparison")
```

Table 17: Models Performance Comparison

Model	Error_Rate	FalsePosRate	FalseNegRate
CART	0.260	0.1769231	0.4142857
CART optimal cut-off	0.245	0.2538462	0.2285714
Conditional Tree	0.275	0.2153846	0.3857143
Conditional Tree optimal cut-off	0.265	0.0615385	0.6428571
Random Forest CART	0.265	0.1461538	0.4857143
Random Forest CTree	0.260	0.1538462	0.4571429
Random Forest CART Bagging	0.260	0.1692308	0.4285714
Random Forest CTree Bagging	0.275	0.1615385	0.4857143

Based on the results of all the models tested, CART tree with optimal cut-off on average performed the best across all metrics such as best error rate, best false negative rate but worst false positive rate. In my opinion, it is due to small sized dataset where single tree could perform similar to more advanced methods like random forest and more prone to overfitting. With addition of finding optimal cutoff value, CART with optimal cutoff benefitted a lot decreasing error rate by 1.5% and outperforming other methods. Random forest generally outperforms other CART and conditional tree in general but in this case, it could not benefit from ensemble methods by using specified hyperparameters due to small dataset. Potentially, with tuning hyperparameters it could at least reach the performance of single tree with optimal cutoff. In terms of variable importance for best model, glucose predictor was the by far the most important in fitting this tree followed by mass and age to define whether diabetes is present.

```
knitr::kable(cart.fit$variable.importance, caption="Variable Importance")
```

Table 18: Variable Importance

	x
glucose	66.58684
mass	27.30498
age	24.95488
pressure	19.61302
insulin	16.25810
pregnant	13.47010
pedigree	11.39796
triceps	8.54755

Question 4 Multiple-Choice Questions

1. Which of the following statements is true about tree-based methods?

A. Given the same dataset for both bagged decision tree and random forest, the number of predictors used at a single split will be equal between the two.

Answer: False. Random Forest uses a subset of predictors for model building, whereas bagged trees use all the features at once.

B. Generally on average, random forests perform better than bagged trees and single trees.

Answer: True. Random forests perform better in practice due to the fact that forest has less correlated trees on average compared to bagging trees due to random feature selection for each bootstrapped sample and existence of Out-Of-Bag sample that allows to measure each bootstrapped fited tree's performance. It's also true that ensemble method which is random forest on average perform better than single trees due to the weighted averaging of end prediction of bootstrapped results which will have better generalization performance.

C. Decision trees' performance is affected by multicollinearity.

Answer: False. If, for example, 10 features are highly correlated, decision tree will pick only one of these features to be used in splitting.

D. Conditional inference trees have the same bias for picking features for splitting compared to CART decision trees.

Answer: False. Conditional inference trees at each split calculate significance test statistics for each feature and feature with minimum p-value is chosen for splitting. CART prefers features with higher number of distinct values (especially fallible when a variable consists of both informative and noisy values) which leads to a feature selection bias problem. The whole purpose of conditional inference trees was to correct this selection bias and have completely unbiased feature selection for splitting.

2. What can be concluded as true about Random forest tree algorithm?

A. Random forest tree results can be interpreted as well as CART.

Answer: False. Random forest is a black box model and can't be used for interpretation due to the large number of trees and bigger depth than cart. Even if taking into account only one tree, it can be very deep with thousands of nodes which is unbearable for human to interpret and use as an explanatory model.

B. Feature Normalization improves the prediction performance of random forest.

Answer: False. Normalizing features does not improve performance because each node in the random forest tree is simply splitting based on criterion and there is no comparison between features' values. It is not only applicable to random forest but also to all decision trees in general.

C. Variable importance of random forest highlights significant features to be used in predicting target variable of the dataset in general regardless of the model used.

Answer: False. Variable importance only showcases importance of features used in fitting this specific random forest model for the given dataset. It should not be interpreted as a global significance of feature like in linear regression.

D. One of the random forest model's advantages is that it has low bias while also maintaining lower variance compared to other decision tree models.

Answer: True. As low bias is the feature of tree based models, high variance is one of its drawbacks dependent on feature number that leads to larger trees and potentially vulnerability to overfitting. However, compared to other tree models like CART and conditional inference tree, random forest model use bootstrap aggregation technique with random features to do prediction and afterwards uses weighted averaging technique which lowers the variance of predictions compared to classical decision tree models.