**Лабораторна робота 5**
**Метод дерева рішень на мові R**

Набір даних audit входить до бібліотеки rattle.data. набір даних аудиту – це штучно створений набір даних, що володіє певними характеристиками справжнього набору даних фінансового аудиту для моделювання продуктивних та непродуктивних аудитів фінансового звіту людини. Продуктивний аудит – це такий, що виявляє помилки та неточності у інформації, наданої клієнтом. Непродуктивний аудит – це зазвичай аудит, який виявив, що вся надана інформація у порядку.
Набір даних аудиту використовується для ілюстрації бінарної класифікації. Цільова змінна ідентифікується як TARGET_Adjusted. Набір даних не великий й складається лише з 2000 об'єктів. Його основна мета – проілюструвати моделювання у Rattle, тому підходить набір даних мінімального розміру. Сам набір отримано з загальнодоступних даних (які не мають нічого спільного з аудитами).

Структура набору даних:

ID – унікальний індикатор кожної людини;
Age – вік;
Employment – тип зайнятості;
Education – найвищий рівень освіти людини;
Marital – поточний родинний стан;
Occupation – тип заняття;
Income – заявлена сума доходу;
Gender – стать;
Deductions – загальна сума витрат, які людина відображає у своїй фінансовій звітності;
Hours – середній час роботи на тиждень, години;
IGNORE_Accounts – основна країна, у якій людина зберігає більшу частину своїх грошей (зверніть увагу, що перед ім'ям змінної стоїть IGNORE. Це визнається Rattle як роль за замовченням для цієї змінної);
RISK_Adjustment – ця змінна записує грошову суму будь-якого корегування фінансових претензій особи у результаті продуктивного аудиту, тобто, ця змінна розглядається не як вхідна змінна, а як міра ризику, пов'язаного з людиною;
TARGET_Adjusted –цільова змінна для моделювання (зазвичай для класифікаційного моделювання), це числове поле класу integer, але обмежене 0 та 1, що вказує на непродуктивний та продуктивний аудити відповідно. Продуктивні аудити – це ті, що призводять до корегування вінансової звітності клієнта.

1. Побудова моделі дерева.

Підготовка даних.
Підключіть бібліотеку rattle.data та подивіться структуру набору даних audit:

```
library(rattle.data)
str(audit)
```

Розділіть дані на тренувальну та тестову множини:

```
library(caTools)
set.seed(3000)
split<-sample.split(audit$TARGET_Adjusted, SplitRatio=0.7)
train<-subset(audit, split==T)
test<-subset(audit, split==F)
```

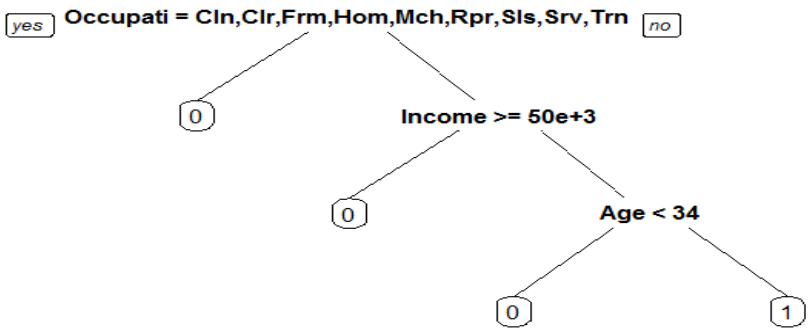
Загрузіть пакети rpart, rpart.plot, rattle, RColorBrewer.

Побудуйте модель.
Наприклад,

```
audittree<-rpart(TARGET_Adjusted~Age+Occupation+Income, data=train, method="class", control=rpart.control(minbucket=25))
```

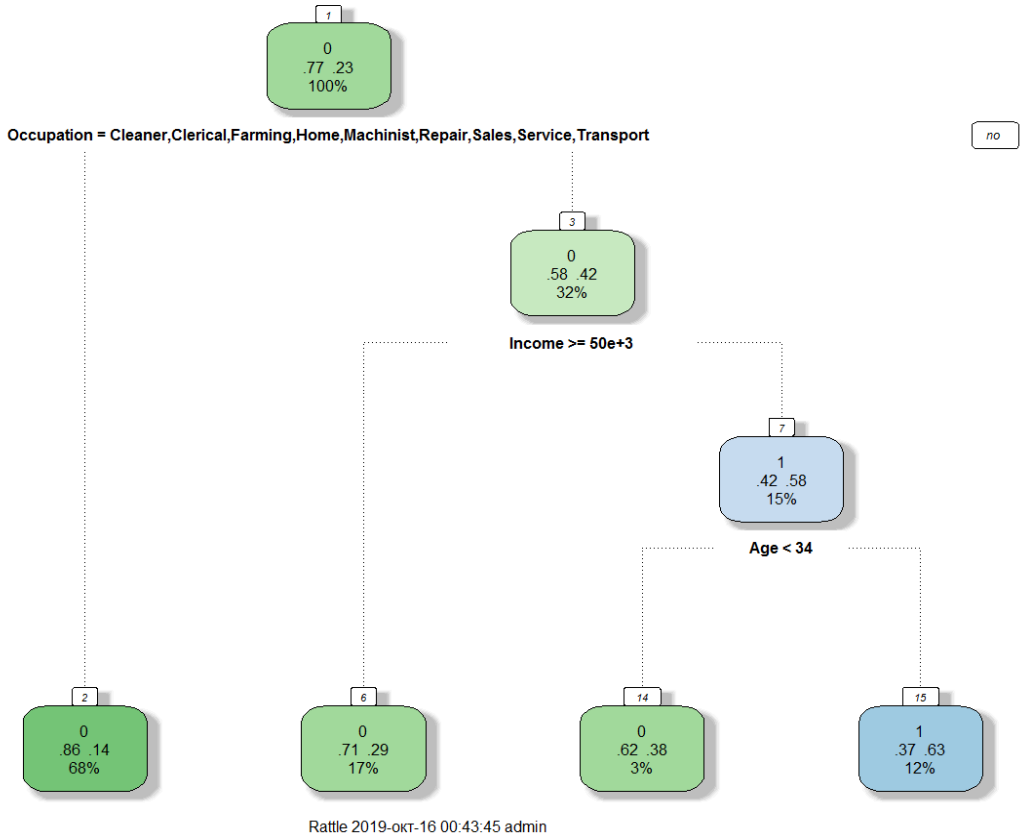
Зобразіть дерево:

```
prp(audittree)
```



Або у більш наглядному вигляді:

fancyRpartPlot(audittree)




Отримайте передбачення:

PredictCART<-predict(audittree, newdata=test, type="class")

Визначте точність:

```
conf.matr<-table(test$TARGET_Adjusted, PredictCART)
conf.matr
```

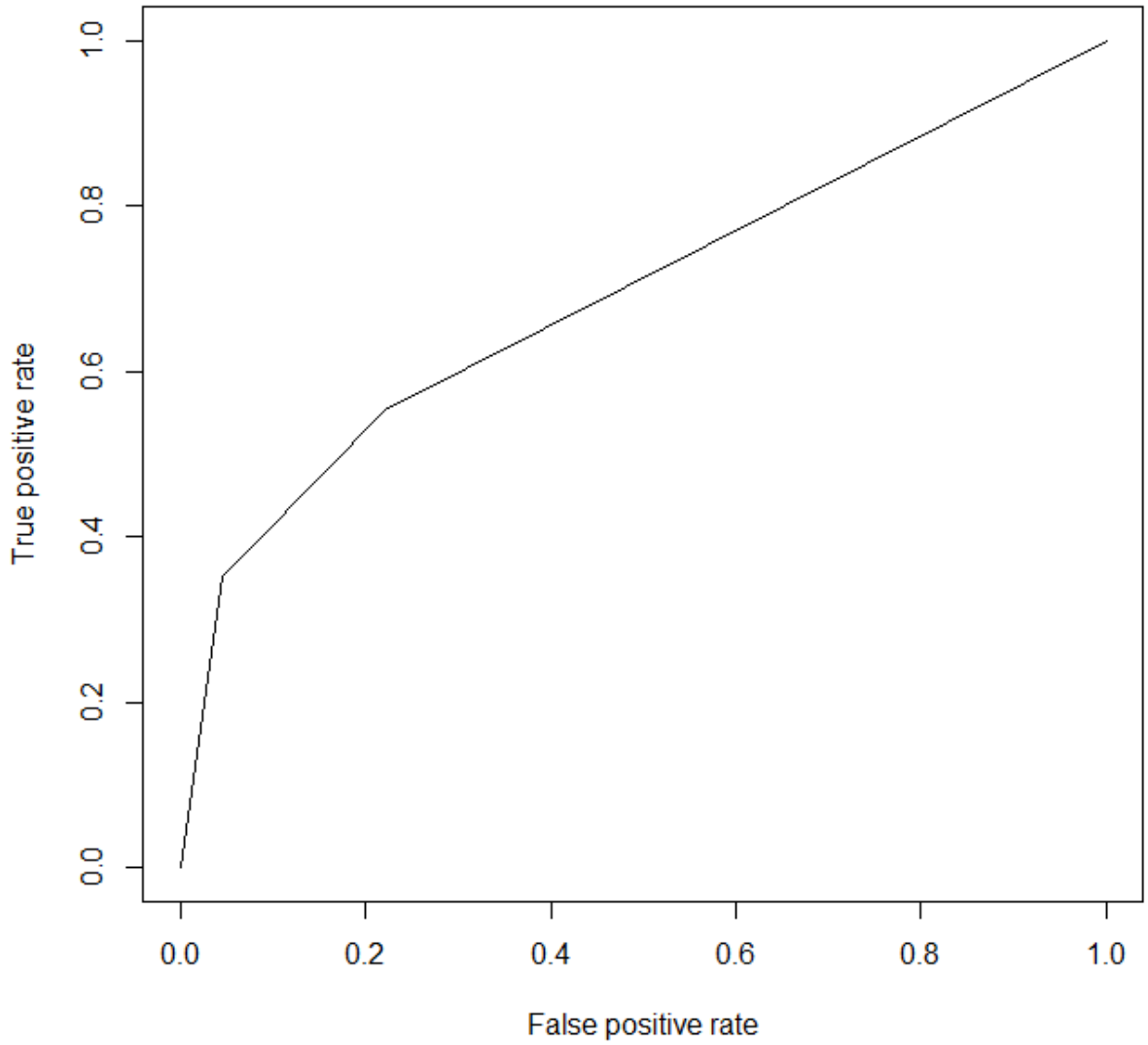
```
   PredictCART
      0   1
  0 440  21
  1  90  49
```

```
accuracy<-(conf.matr['0','0']+conf.matr['1','1'])/sum(conf.matr[])
paste("Точність:", accuracy)
```

```
[1] "Точність: 0.815"
```

Побудова кривої похибок або ROC-кривої:

```
library(ROCR)
predictROC<-predict(audittree, newdata=test)
pref<-prediction(predictROC[,2],test$TARGET_Adjusted)
perf<-performance(pref, "tpr", "fpr")
plot(perf)
```




2. Метод крос-валідації

Встановіть необхідні пакети caret, e1071.

Розбийте тренувальні дані на частини (folds).

Наприклад,
fitControl<-trainControl(method="cv", number=30)

Тут, method="cv" означає, що використовувати крос-валідацію, а number=30 – що використовувати 30 частин.

Задайте значення cp.
Наприклад від 0,001 до 2

```
cartGrid<-expand.grid(.cp=(1:200)*0.001)
```

Увага: залежна змінна повинна мати тип factor

```
train(as.factor(TARGET_Adjusted) ~ Age+Occupation+Income, data = train, method = "rpart", trControl = fitControl, tuneGrid = cartGrid, na.action=na.exclude)
```

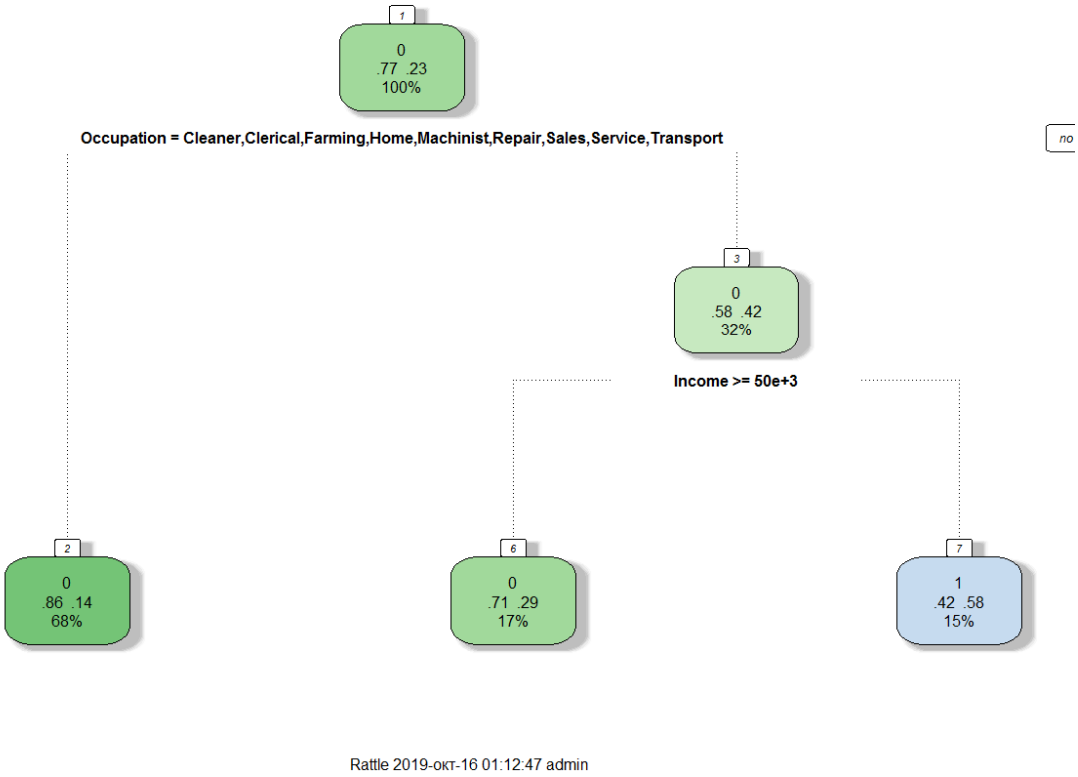
Результатом буде таблиця, яка містить різні значення точності для різних cp. Потрібно значення cp, яке максимізує точність, воно вказано в кінці виведення функції:

```
cp     Accuracy   Kappa
  0.001  0.7397389   0.1932924176
  0.002  0.7533247   0.2098812703
  0.003  0.7555974   0.1980251487
  0.004  0.7646065   0.2148368580
  0.005  0.7721486   0.2411609837
  0.006  0.7668624   0.2104484664
  0.007  0.7690839   0.2226016774
  0.008  0.7713566   0.2301501457
  0.009  0.7758852   0.2497323309
  0.010  0.7766596   0.2570468149
...
0.198  0.7654972   0.0000000000
  0.199  0.7654972   0.0000000000
  0.200  0.7654972   0.0000000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.036.
```

Отримане cp необхідно використовувати у функції rpart як параметр control=rpart.control(cp=)

```
audittree_1<-rpart(TARGET_Adjusted~Age+Occupation+Income, data=train, method="class", control=rpart.control(cp=0.036))
```

Розгляньте отримане дерево



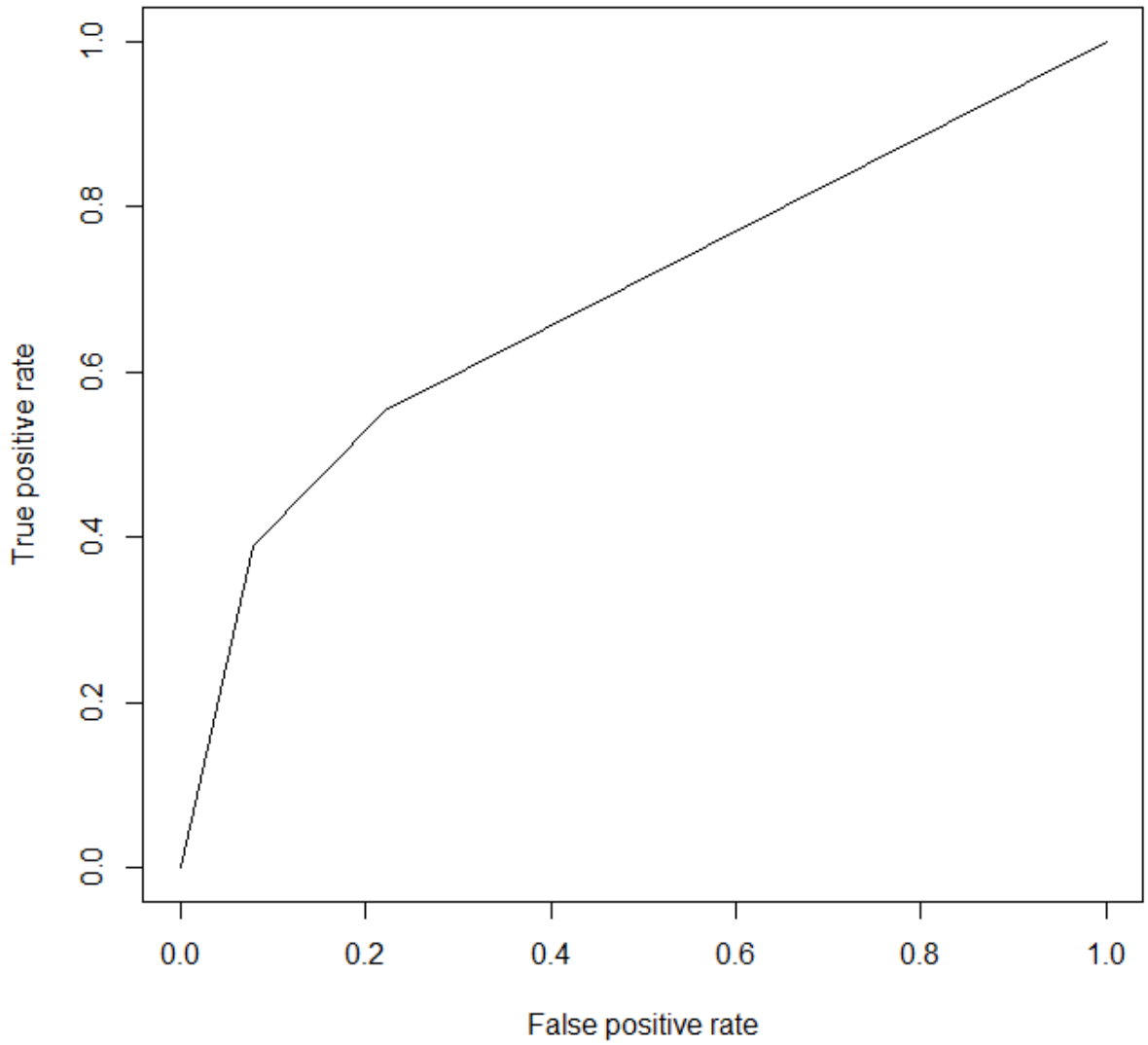

Отримайте передбачення та визначте точність:

```
PredictCART<-predict(audittree_1, newdata=test, type="class")

conf.matr<-table(test$TARGET_Adjusted, PredictCART)
conf.matr

   PredictCART
     0   1
 0 425  36
 1  85  54

accuracy<-(conf.matr['0','0']+conf.matr['1','1'])/sum(conf.matr[])
paste("Точність:", accuracy)

[1] "Точність: 0.798333333333333"
```

Побудуйте криву похибок:

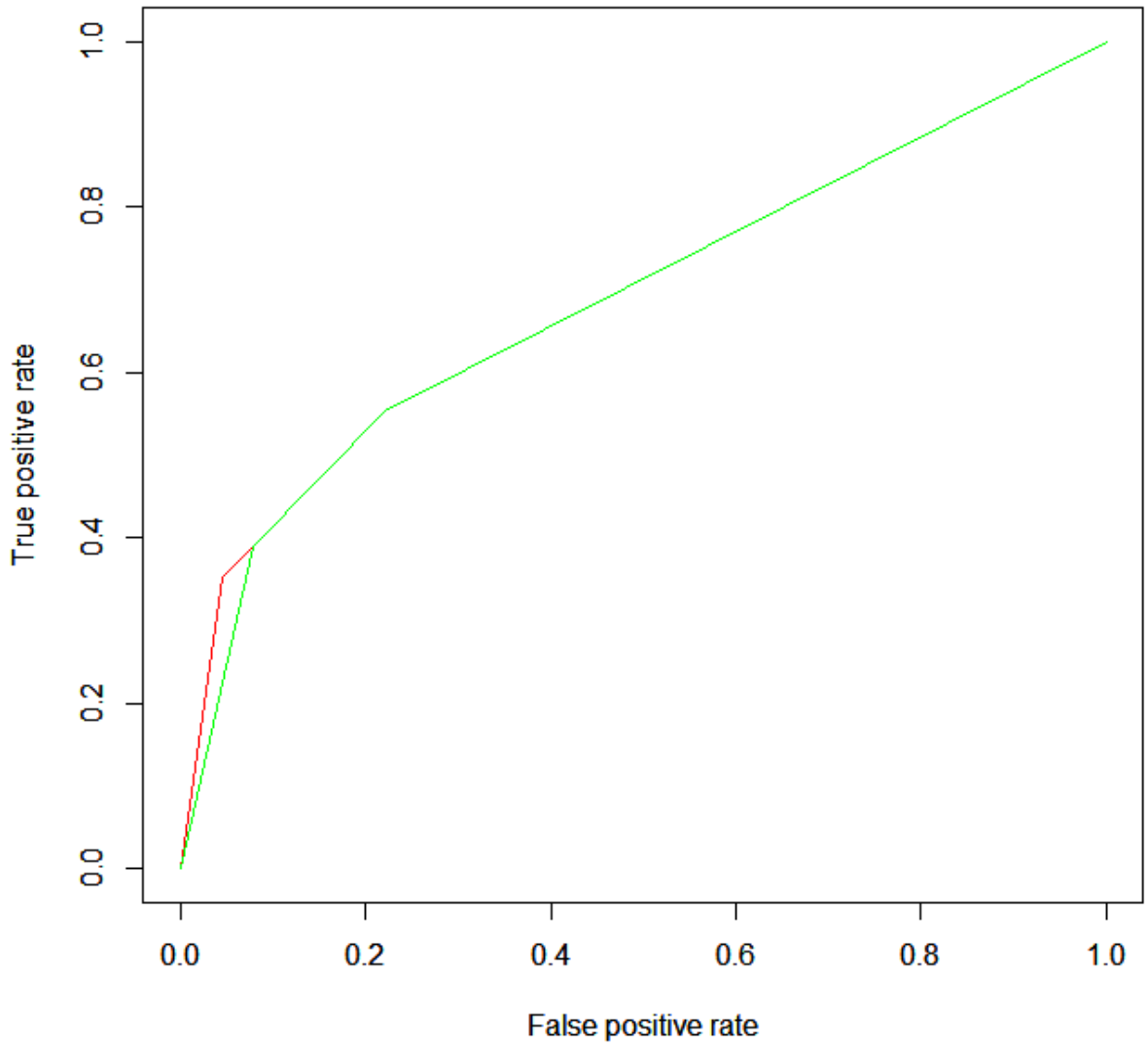
```
predictROC1<-predict(audittree_1, newdata=test)
> pref1<-prediction(predictROC1[,2],test$TARGET_Adjusted)
> perf1<-performance(pref1, "tpr", "fpr")
```

> plot(perf1)




Порівняйте криві похибок після застосування двох методів:

```
plot(perf, col="red")
par(new=T)
plot(perf1, col="green")
```




3. Алгоритм Випадковий ліс (Random forest)

Підключіть бібліотеку randomForest.

Увага: залежна змінна повинна мати тип factor

Побудуйте випадковий ліс:

trainforest<-randomForest(as.factor(TARGET_Adjusted) ~ Age+Occupation+Income, data=train, nodesize=25, ntree=200, na.action=na.exclude)

Отримайте передбачення та точність моделі:

predictforest<-predict(trainforest, newdata=test)

```
conf.matr<-table(test$TARGET_Adjusted, predictforest)
conf.matr

predictforest
     0   1
  0 405  29
  1  75  60

accuracy<-(conf.matr['0','0']+conf.matr['1','1'])/sum(conf.matr[])
paste("Точність:", accuracy)

[1] "Точність: 0.817223198594025"
```

Отриманий ліс має точність на тестовому наборі даних приблизно 81,7% та є кращим за дерево прийняття рішень.

Крос-валідацію для випадкового лісу можна не застосовувати, оскільки ліс стійкий до незначних атрибутів та менше піддається перенавчанню.