Arbeitsblatt 4

Aufgabe 1: Nächste-Nachbarn-Klassifizierer

Der Datensatz `Smarket` aus dem Paket `ISLR` enthält die Renditen (relative Änderung) des Aktienindex S&P 500 von Anfang 2001 bis Ende 2005: das sind 1250 Handelstage. An jedem Tag sind die Renditen der vorhergehenden 5 Handelstage angegeben (Variablen `Lag1`, ..., `Lag5`). Das Volumen des Handelstages ist in der Variable `Volume` enthalten (Anzahl Aktien die am Vortag gehandelt wurden in Milliarden). Die Variable `Today` enthält die Renditen des aktuellen Tages und die Variable `Direction` beschreibt, ob die aktuelle Rendite positiv oder negativ war. Hier soll versucht werden, die Richtung ("Direction") der Marktentwicklung vorherzusagen, wobei als erklärende Variablen die Renditen der 2 vorangegangenen Tage (`Lag1` und `Lag2`) benutzt werden sollen. Wir wollen die Jahre 2001-2004 zum Trainieren benutzten und dann den Klassifikator auf das Jahr 2005 anwenden. Die Daten findet man im Package `ISLR` und kann mit dem Befehl `Smarket` aufgerufen werden.

Ziel: KNN Klassifikation anhand der Aktienmarktbewegungen durchführen.

a) Verschaffen Sie sich einen Überblick über den Datensatz (R-Hinweis: `summary()`, `str()`, `dim()`, `pairs()`, `ts.plot()` ...). Datensatz auf Moodle oder direkt aus dem Paket `ISLR`.

b) Wählen Sie alle Beobachtungen in den Jahren vor 2005 mit den Variablen *Lag 1* und *Lag 2* als Trainings-Datensatz und die Beobachtungen in den restlichen Jahren als Test-Datensatz (auch mit den Variablen *Lag 1* und *Lag 2*).

c) Trainieren Sie mit der Funktion `train` aus `caret` ein knn-Modell. Starten Sie mit einem nearest neighbor (`tuneGrid=data.frame(k=1)`). Verwenden Sie das Modell für eine Vorhersage auf dem **Trainings-Datensatz** (`predict(model, newdata=train`). Quantifizieren Sie die Performance, indem Sie eine Kreuztabelle der wahren und vorhergesagten Klassen – also eine Konfusions-Matrix für den Trainings-Datensatz erstellen (Befehl `table` oder `confusionMatrix` aus 'caret). Was fällt Ihnen auf?

d) Nun wollen wir das knn-Modell mit k=1 richtig testen. Wenden Sie dazu die Prediction auf den Test-Datensatz an. Führen Sie `predict`-Funktion auch einmal mit dem Argument `type = prob"` durch. Was ist der Zweck des Types `prob`.

e) Erstellen Sie eine Konfusionsmatrix für den Test-Datensatz und beurteilen Sie damit die Performance Ihres Klassifikators anhand der Accuracy.

f) Wiederholen Sie nun die Teilaufgaben d) - e) mit der Anzahl nearest neighbors von k=2-7 (`tuneGrid=data.frame(k=2:7)`). Welches k führt zum besten Resultat?

Aufgabe 2: Kreuzvalidierung

In dieser Aufgabe wird das Churn-Verhalten (Wechseln zu einem anderen Anbieter) von Telekommunikationskunden untersucht. Der Wettbewerb unter den Telekommunikationsanbietern ist sehr gross. Viele Kunden wechseln jedes Jahr den Anbieter. Telekommunikationsunternehmen sind deshalb bestrebt, Kunden durch geeignete Marketing-Programme vom Wechseln abzuhalten. Dazu müssen jedoch die zu einem Wechsel neigenden Kunden identifiziert werden, was z.B. mit einem Klassifikationsverfahren aufgrund der Kundenmerkmale gemacht werden kann.

Ihnen steht nun ein Datensatz zur Verfügung, in dem von 3'333 Kunden eines US-Telekommunikationsunternehmens bekannt ist, ob sie den Anbieter gewechselt haben oder nicht.

Zugleich kennen wir noch folgende Kundenmerkmale:

Variablenname	Kurzbeschreibung	Typ
AccountLength	Vertragsdauer in Monaten	integer
AreaCode	Postleitzahl	*drei* Integerwerte
IntlPlan	Spezialvertrag für internationale Anrufe	binär
VMailPlan	Voice-Mail-Plan	binär
VMailMsg	Anzahl Voice-Mail-Nachrichten	integer
DayMins	Zeitdauer mit Gesprächen am Tag (in Minuten)	stetig
DayCalls	Anzahl Anrufe am Tag	integer
EveMins	Zeitdauer mit Gesprächen am Abend (in Minuten)	stetig
EveCalls	Anzahl Anrufe am Abend	integer
NightMins	Zeitdauer mit Gesprächen in der Nacht (in Minuten)	stetig
NightCalls	Anzahl Anrufe in der Nacht	integer
IntlMins	Zeitdauer mit internationalen Gesprächen (in Minuten)	stetig
IntlCalls	Anzahl internationale Anrufe	integer
CustServCalls	Anzahl Anrufe an den Kundenservice	integer
Churn	Kunde hat den Anbieter gewechselt oder nicht	binär

a) Laden Sie die Daten `churn` (`Churn1.rdata`). Die Faktorvariablen `IntlPlan` und `VMailPlan` wollen wir hier nicht berücksichtigen, obwohl das mit der caret-Funktion auch funktionieren würde. Löschen Sie diese Features (z.B. `churn$IntlPlan <- NULL`). Skalieren Sie die restlichen Daten, falls nötig (Funktion `scale()` oder `preProcess = c(center", ßcale")` in der Funktion `train`. Verwenden Sie 80 Prozent der Daten als Trainingset und 20 Prozent der Daten als Testset (z.B. mit der Funktion `createDataPartition(y=churn_sc$Churn, p=0.8, list=F)`). Wie gross ist die Performance eines k=1 Klassifikators auf dem Trainings- und Testset. Begründen Sie, wieso die Fehlerrate (`mean(res.knn != y_test)`) auf dem Trainingset kleiner ist.

b) Wiederholen Sie a) für alle Werte von $k = 1, \ldots, 20$ und plotten Sie die Fehlerrate von Trainings- und Testdaten gegenüber 1/k. Für welches k performt der knn-Klassifizierer am besten.

c) Wiederholen Sie die Analyse aus b). Führen Sie diesmal aber eine 5-fache Kreuzvalidierung auf dem gesamten Datensatz durch. Verwenden Sie dazu folgenden Code und versuchen Sie ihn nachzuvollziehen.

```
x_fold <- 5
groups <- rep(1:x_fold, length.out=dim(churn)[1])
samp <- sample(x=groups, size=dim(churn)[1], replace=FALSE)
error_test <- error_training <- rep(NA,20)
for (k in 1:20){
  error_test_fold <- rep(NA, x_fold)
  error_train_fold <- rep(NA, x_fold)
  for (i in 1:x_fold){
    churn_train <- churn[samp!=i,]
    churn_test <- churn[samp==i,]
    knn <- train(Churn ~., data = churn_train, method="knn",
                 tuneGrid=data.frame(k=k),
            preProcess = c("center", "scale"))
    pred_train <- predict(knn, newdata=churn_train)
    error_train_fold[i] <- mean(pred_train != churn_train$Churn)
```

```
    pred_test <- predict(knn, newdata=churn_test)
    error_test_fold[i] <- mean(pred_test != churn_test$Churn)
  }
  error_training[k] <- mean(error_train_fold)
  error_test[k] <- mean(error_test_fold)
}
```

d) Viel einfacher kann man die Kreuzvalidierung direkt mit dem **Paket `caret`** durchführen (Code unten). Dort kann man mit der **Funktion `trainControl()`** die Parameter für die Validierung festlegen. Wir verwenden `method = 'repeatedcv'`, bei dieser Methode wird die Kreuzvalidierung mehrmals wiederholt und der Mittelwert verwendet. Das reduziert die Schwankungen, dauert aber auch länger. Die Anzahl an Wiederholung wird mit dem Argument `repeats` festgelegt. Bei `method = cv` wird die Kreuzvalidierung nur einmal durchgeführt. Das k für die k-fache Kreuzvalidierung wird über den Parameter `number` gesteuert. `number = 10` bedeutet zum Beispiel 10-fache Kreuzvalidierung. Das Trainieren des Klassifizierer geschieht mit der **Funktion `train()`**. Der Funktion müssen die Parameter aus dem `trainControl` übergeben werden (Argument `trControl`). Die zu evaulierenden Parameter können in einem Dataframe an das Argument `tuneGrid` übergeben werden. Auch das Preprocessing der Daten ist direkt möglich. Mit `preProcess = c('center', 'scale')` werden die Daten zum Beispiel zentriert und skaliert. Mit einen Plot auf den Output, erhält man direkt eine Abbildung der Accuracy in Abhängigkeit der Anzahl Nachbarn. (Das Vorgehen kann einfach auf einen anderen Klassifizier übertragen werden, es muss nur eine andere Methode gewählt werden und allenfalls die Parameter angepasst werden).

e) Wieso empfiehlt es sich in diesem Fall, eine 10-fache Kreuzvalidation und keine Leave-one-out-Kreuzvalidierung zu verwenden?

Aufgabe 3: Diskriminanzanalyse

Verwenden Sie den Boston Datensatz `Boston.RData` um vorherzusagen, ob ein Stadtteil (Zeile im Datensatz) eine grosse Kriminalitätsrate hat. Die Zielvariable ist dabei `Boston$crime`

a) Laden Sie die Daten und unterteilen Sie den Datensatz in einen Trainings und einen Testdatensatz, wobei die Daten gleich verteilt werden sollen. Verwenden Sie dazu den Befehl `createDataPartition` aus dem Packet `caret`.

b) Verwenden Sie mit dem Befehl `train` aus dem Packet `caret` die LDA Methode und sagen Sie anhand der Trainingsdaten die Testdaten voraus. Erstellen Sie anschliessend eine Konfusionsmatrix daraus.

c) Gehen Sie gleich wie bei Teilaufgabe b) vor, dieses mal einfach mit der QDA Methode. Erstellen Sie wiederum daraus die Konfusionsmatrix.

d) Welcher Ansatz funktioniert besser?

Aufgabe 4: Leistungsnachweise

- Versuchen Sie einen unser Klassifizierer auf Ihren Datensatz anzuwenden. Beachten Sie dabei die Aspekte, welche wir heute besprochen haben.