



Fachbereich Mathematik und Naturwissenschaften
Studiengang Business Mathematics (Master of Science)

Data Mining II

Booklet

Vorgelegt von Büsra Karaoglan am 14. Juli 2018
Matrikelnummer HDA: 754331 / THM: 5056046

Referent Prof. Dr. Werner E. Helm

Inhaltsverzeichnis

1 Einleitung	2
2 Aufgabe 1	4
2.1 1 U	4
2.2 1 V	16
2.3 1 W	24
3 Aufgabe 2	33
4 Aufgabe 3	60
5 Aufgabe 4	73
5.1 4 A	73
5.2 4 B	80
5.3 4 C	88
5.4 4 D	93
6 Aufgabe 5	101

1 Einleitung

Diese Ausarbeitung dient der Vorleistung des Fachbereichs Business Mathematics an der Hochschule Darmstadt für das Modul *Data Mining 2*, gelesen von Prof. Dr. Werner Helm. Sie beinhaltet Aufgaben die mit Hilfe der Vorlesungsunterlagen und mit unterschiedlichen Softwareprogrammen bearbeitet wurden. Die lauffähigen SAS, SAS Enterprise Miner, R und Python-Dateien werden zusätzlich an Prof. Dr. Werner Helm ausgehändigt. Es werden in dieser Ausarbeitung Themenbereiche wie logistische Regression, Support Vector Machines, Entscheidungsbäume und künstliche neuronale Netze erarbeitet. Diese Themen werden auch im Folgenden kurz erläutert. Das theoretische beziehungsweise mathematische Hintergrundwissen der Themen sind Voraussetzungen für das Verständnis der Aufgaben. Allerdings werden an erforderlichen Stellen ergänzende Bemerkungen bezüglich der Thematik gemacht.

Bei vielen Fragestellungen, bei denen es um die Analyse des Einflusses einer oder mehrerer unabhängiger Variablen auf eine abhängige Variable geht, liegt als abhängige Variable keine stetige, sondern eine kategoriale oder qualitative Variable vor. Verfügt die abhängige Variable über zwei mögliche Ausprägungen (z.B. Survived = Ja oder Nein), dann spricht man von einer binären logistischen Regression. Die Ereignisse werden dabei als 0/1-Ereignisse oder auch Komplementärereignisse bezeichnet. Die abhängige Variable Y kann also die beiden Ausprägungen 0 und 1 annehmen. Die unabhängigen Variablen, oftmals auch als Kovariate oder Regressoren bezeichnet, können sowohl metrisch als auch kategorial skaliert sein. Hier in dieser Arbeit wird die binäre logistische Regression genauer dargestellt [AL05].

Eine Support Vector Machine (SVM) ist ein Verfahren bei dem ein gegebener Datensatz (welcher durch Vektoren in einem Vektor-Raum repräsentiert ist) durch eine Hyperebene in der Art zu teilen, dass den Vektoren auf derselben Seite die gleiche Klasse zugeordnet ist. Zudem wird die Größe des Randes der Hyperebene maximiert. Der Rand der Hyperebene ist durch die Vektoren gegeben, die den geringsten Abstand von dieser Ebene besitzen. Diese Vektoren werden als Support-Vektoren bezeichnet. Eine Klassifizierung mit einer Support Vector Machine ist ein überwachtes Lernverhalten, wobei die Support Vector-Klassifizierung durch das Lösen einer dualen Optimierungsprobleme erfolgt. Mit Hilfe der SVM können sowohl linear trennbare, als auch - durch eine Erweiterung der SVMs mit Kernel-Methoden - nicht-linear trennbare Daten im mehrdimensionalen Merkmalsraum separiert werden [TB14].

Die Entscheidungsbäume dienen der Aufteilung von Objekten. Dies geschieht anhand geeigneter Merkmale in Gruppen in Hinblick auf eine vorgegebene Zielgröße. Ein Entscheidungsbau hat eine baumartige Struktur mit einer Wurzel, mehreren Blattknoten, inneren Knoten und Kanten. Jedem Blattknoten ist eine Klasse zugeordnet; pro Klasse sind mehrere Blattknoten möglich. Jedem inneren Knoten ist ein Merkmal zugeordnet; pro Merkmal sind mehrere innere Knoten möglich. Die Entscheidungsbäume lassen sich zusätzlich in zwei Varianten unterteilen: Klassifikationsbäume und Regressionsbäume. Klassifikationsbäume werden bei nominal skalierten Variablen als abhängige Zielgröße eingesetzt, während bei Regressionsbäumen eine quantitative Variable als abhängige Zielgröße vorliegt [UB08].

Die künstliche neuronale Netze (KNN), oder artificial neural networks (ANN) bezeichnet, sind informationsverarbeitende Systeme die eine Analogie zum menschlichen Gehirn haben. Die neuronale Netze bestehen aus einer großen Anzahl Neuronen, die sich Informationen im Form der Aktivierung der Neuronen über gerichtete und gewichtete Verbindungen zusenden. Dabei haben sie die Fähigkeit, eine Aufgabe selbstständig, nur anhand von Trainingsbeispielen, zu lernen. Die neuronale Netze werden haben sehr unterschiedliche Einsatzmöglichkeiten, unter Anderem im Bereich der Prognose und Kategorisierung [AF11].

In dem Paper von *Tom Fawcett: An introduction to ROC analysis* [TF06], die in Data Mining 1 ausgehändigt wurde, befinden sich folgende Informationen über ROC-Kurve, AUC-Wert und Confusion Matrix. Die ROC-Kurve (ROC = Receiver Operating Characteristics) ist eine Methode für die Visualisierung und die Prüfung der Leistung eines (binären) Klassifikators. ROC-Graphen wurden lange Zeit in der Signalentdeckungstheorie verwendet, um den Kompromiss zwischen Trefferquoten und falschen Alarmraten von Klassifikatoren darzustellen. Die ROC-Kurven werden nun häufig in der medizinischen Entscheidungsfindung verwendet und wurden in den letzten Jahren zunehmend in der maschinellen Learn- und Data-Mining-Forschung eingesetzt.

Im Folgenden werden Prüfungen von Klassifizierungsproblemen nur zweier Klassen vorgeführt. Jede Instanz I wird auf ein Element $\{p, n\}$ der positiven und negativen Klassenlabels abgebildet. Um zwischen der eigentlichen Klasse und der vorhergesagten Klasse zu unterscheiden, werden $\{Y, N\}$ für die von dem Modell erzeugten Klassenvorhersagen verwendet. Somit werden die tatsächlichen Beobachtungen in die Klassen p für *positiv* und n für *negativ* und die prognostizierten Beobachtungen in die Klassen Y für *Yes* und N für *No* zugeordnet. Somit ergibt sich vier mögliche Ergebnisse:

- True Positives-TP: Beobachtung wird als positiv klassifiziert, ist tatsächlich positiv
- True Negatives-TN: Beobachtung wird als negativ klassifiziert, ist tatsächlich negativ
- False Positives-FP: Beobachtung wird als positiv klassifiziert, ist tatsächlich negativ
- False Negatives-FN: Beobachtung wird als negativ klassifiziert, ist tatsächlich positiv

Die unten abgebildete Grafik zeigt eine 2×2 *Confusion Matrix* (auch Kontingenztafel genannt).

		<u>True class</u>	
		p	n
<u>Hypothesized class</u>	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:		P	N

Es werden nun folgende Maßzahlen definiert:

$$\text{tp rate} \approx \frac{\text{Postives correctly classified}}{\text{Total positives}} = \frac{TP}{P} \quad \text{und fp rate} \approx \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}} = \frac{FP}{N}$$

Zusätzlich werden im Zusammenhang mit der ROC-Kurven-Analyse die Begriffe *Sensitivität* und *Spezifität* eingeführt. Dabei gilt

$$\text{Sensitivität} = \frac{TP}{P} \quad \text{und Spezifität} = 1 - \text{fp rate} = 1 - \frac{FP}{N}.$$

Die ROC-Kurven sind zweidimensionale Graphen, in denen tp rate von 0 bis 1 (0% bis 100%) auf der Y-Achse und die fp rate von 0 bis 1 (0% bis 100%) auf der X-Achse aufgetragen ist. Eine ROC-Grafik stellt entsprechende Kompromisse zwischen Nutzen (true positives) und Kosten (false positives) dar.

In Bezug auf die ROC-Kurve existiert noch die AUC (Area Under the Curve) Maßzahl. AUC ist definiert als die Fläche unter der ROC-Kurve. Dabei gilt ein AUC nahe 1 als zuverlässiges und AUC weniger als 0,5 als wertloser bzw. als zufälliger Klassifikator.

2 Aufgabe 1

2.1 1 U

Aufgabenstellung:

In das SAS Programm **TEST_PR_SVM_IVAN_5.sas** sind bereits die Tabellen Table 1, Table 3 und Table 5 eingearbeitet und in der VL erklärt worden. Erweitern Sie das Programm um 3 weitere Tabellen aus Ivanciu (vgl. pdf).

Für Table 3 und eine weitere frei gewählte Table n ::

Ziehen Sie die relevanten Größen aus den von SAS erzeugten Tabellen (A B C D E) (j) und aus dem SAS Output und stellen diese in einem Diagramm der Daten und des Modelles dar.

Fügen Sie 4 weitere typische neue Punkte an die Datei an (oder ggf. besser in eine separate Datei ein) und ‚scoren‘ diese Punkte manuell (mit Taschenrechner) und mit dem System. Wie ist die Übereinstimmung?

Lösung:

Die folgenden Erläuterungen stammen aus dem Handout [WISV], das in der Vorlesung von Data Mining 2 ausgeteilt wurde.

Eine Support Vector Machine bestimmt, anhand einer Menge von Trainingsdaten

$$\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) | \mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, 1\}\}$$

eine Hyperebene, die beide Klassen möglichst eindeutig voneinander trennt. Dabei soll der kleinste Abstand zur Hyperebene, dem sogenannten Margin für die Daten beider Klassen maximiert werden. Hier gibt y_i die Klassenzugehörigkeit für die Trainingsdaten \mathbf{x}_i an. Als Entscheidungsfunktion wird das sogenannte Training benutzt welche die Hyperebene berechnet, die die Trainingsdaten beider Klassen bestmöglich teilt. Die Entscheidungsfunktion ist gegeben durch einen Normalenvektor \mathbf{w} und einen sogenannten Bias b . Mit einem Trainingsdaten \mathbf{x}_i wird dabei das Vorzeichen der Entscheidungsfunktion als Klasse zugeordnet:

$$y_i = \text{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$$

Je nachdem, wo sich die Daten relativ zur Hyperebene befinden (oberhalb oder unterhalb), errechnet sich demnach ein positiver oder negativer Wert, wo nach Anwendung der Vorzeichenfunktion (sgn) nur noch ± 1 verbleibt. Für Daten, die genau auf der Trennebene liegen, wird dieser Wert zu 0.

Zunächst folgt die Erläuterung für die **lineare Trennbarkeit** der Trainingsdaten. Die SVM wählt von allen möglichen trennenden Hyperebenen diejenige mit minimaler quadratischer Norm $\|\mathbf{w}\|_2^2$ aus, so dass gleichzeitig $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ für jedes Trainingsdaten \mathbf{x}_i gilt. Dieses Vorgehen ist mit der Maximierung des kleinsten Abstands zur Hyperebene (dem Margin) äquivalent.

Das Optimierungsproblem lautet:

$$\text{minimiere bezüglich } \mathbf{w}, \mathbf{b} : \frac{1}{2} \|\mathbf{w}\|_2^2,$$

so dass die Nebenbedingung $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ für alle $1 \leq i \leq m$ gilt.

Im Allgemeinen sind viele Daten nicht streng linear separierbar. Für diesen Fall wird nun das Optimierungsproblem derart verändert, dass Verletzungen der m Nebenbedingungen möglich sind, die Verletzungen aber so klein wie möglich gehalten werden sollen. Daher wird nun eine positive Schlupfvariable ξ_i für jede Nebenbedingung eingeführt, deren Wert gerade die Verletzung der Nebenbedingungen ist. $\xi_i > 0$ bedeutet also, dass die Nebenbedingung verletzt ist. Da in der Summe die Verletzungen möglichst klein gehalten werden sollen, wird die Summe der Fehler der Zielfunktion hinzugefügt und somit ebenso minimiert. Zusätzlich wird diese Summe mit einer positiven Konstante C multipliziert, die den Ausgleich zwischen der Minimierung von $\frac{1}{2} \|\mathbf{w}\|_2^2$ und der korrekten Klassifizierung der Trainingsbeispiele regelt.

Das Optimierungsproblem für **nicht-linear separierbare** Daten besitzt somit folgende Form:

$$\text{minimiere bezüglich } \mathbf{w}, \mathbf{b} : \frac{1}{2} \|\mathbf{w}\|_2^2 + C \cdot \sum_{i=1}^m \xi_i,$$

so dass die Nebenbedingung $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ für alle $1 \leq i \leq m$ gilt.

Beide Optimierungskriterien sind konvex und werden normalerweise in der **dualen Form** gelöst. Diese Formulierung ist äquivalent zu dem primalen Problem, in dem Sinne, dass alle Lösungen des dualen auch Lösungen des primalen Problems sind - sollte aus dem Modul Operations Research im Bachelor bekannt sein.

Die Umrechnung ergibt sich dadurch, dass der Normalenvektor \mathbf{w} als Linearkombination aus Trainingsdaten geschrieben werden kann:

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Die duale Form wird dann mit Hilfe der Lagrange-Multiplikatoren und der Karush-Kuhn-Tucker-Bedingungen hergeleitet - sollte wiederum aus dem Modul Numerik 1/2 im Bachelor bekannt sein. Sie lautet:

$$\text{maximiere für } \alpha : \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

so dass die Nebenbedingungen $0 \leq \alpha_i \leq C$ und $\sum_{i=1}^m \alpha_i y_i = 0$ gelten.

Somit ergibt sich als Klassifikationsregel:

$$f(x) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b) = \text{sgn} \left(\sum_{i=1}^m \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

Der Name der SVM kommt von einer speziellen Untermenge der Trainingspunkte, deren Lagrangevariablen $\alpha_i \neq 0$ sind. Diese heißen Support-Vektoren und liegen entweder auf dem Margin (falls $y_i \cdot (\langle \mathbf{w}, \mathbf{x} \rangle + b) = 1$ oder innerhalb des Margin, also $\xi_i > 0$).

Der bisher beschriebene Algorithmus klassifiziert die Daten mit Hilfe einer linearen Funktion, welche natürlich jedoch nur dann optimal ist, wenn auch das zu Grunde liegende Klassifikationsproblem linear ist. In vielen Anwendungen ist dies aber nicht der Fall und hierfür wäre ein möglicher Ausweg die Daten in einen Raum höherer Dimension abzubilden.

Die Abbildungsfunktion mit $d_1 < d_2$ lautet:

$$\phi : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}, \mathbf{x} \mapsto \phi(\mathbf{x})$$

In das dem Algorithmus zu Grunde liegende Optimierungsproblem in der zuletzt dargestellten Formulierung gehen ja die Datenpunkte \mathbf{x}_i nur in Skalarprodukten ein. Daher ist es möglich, das Skalarprodukt $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ im Eingaberaum (*Input space*) \mathbb{R}^{d_1} durch ein Skalarprodukt im \mathbb{R}^{d_2} zu ersetzen und $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ statt dessen direkt zu berechnen. Die Kosten dieser Berechnung lassen sich sehr stark reduzieren, wenn eine positiv definite **Kernelfunktion** stattdessen benutzt wird:

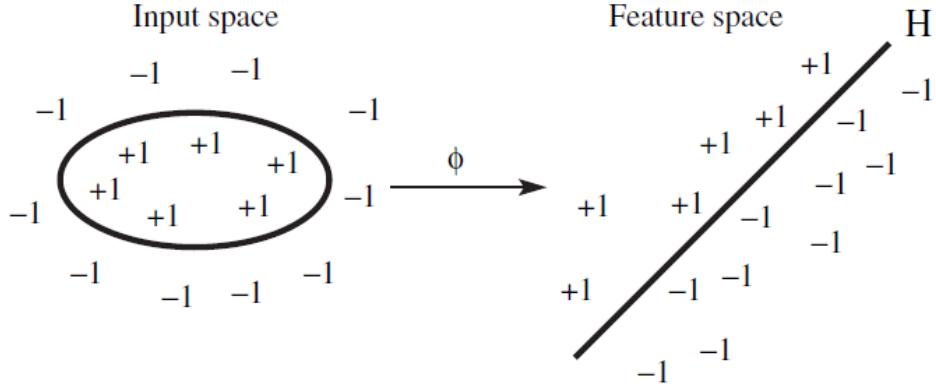
$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

Durch dieses Verfahren kann eine Hyperebene (d. h. eine lineare Funktion) in einem hochdimensionalen Raum (*Feature space*) implizit berechnet werden. Der resultierende Klassifikator hat die Form

$$f(x) = \operatorname{sgn} (\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b) = \operatorname{sgn} \left(\sum_{i=1}^m \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j) + b \right)$$

mit $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \phi(\mathbf{x}_i)$.

Dieses Vorgehen wird in Auszug von Ovidiu Ivanciu [OI07] zusätzlich mit einem Bild dargestellt:



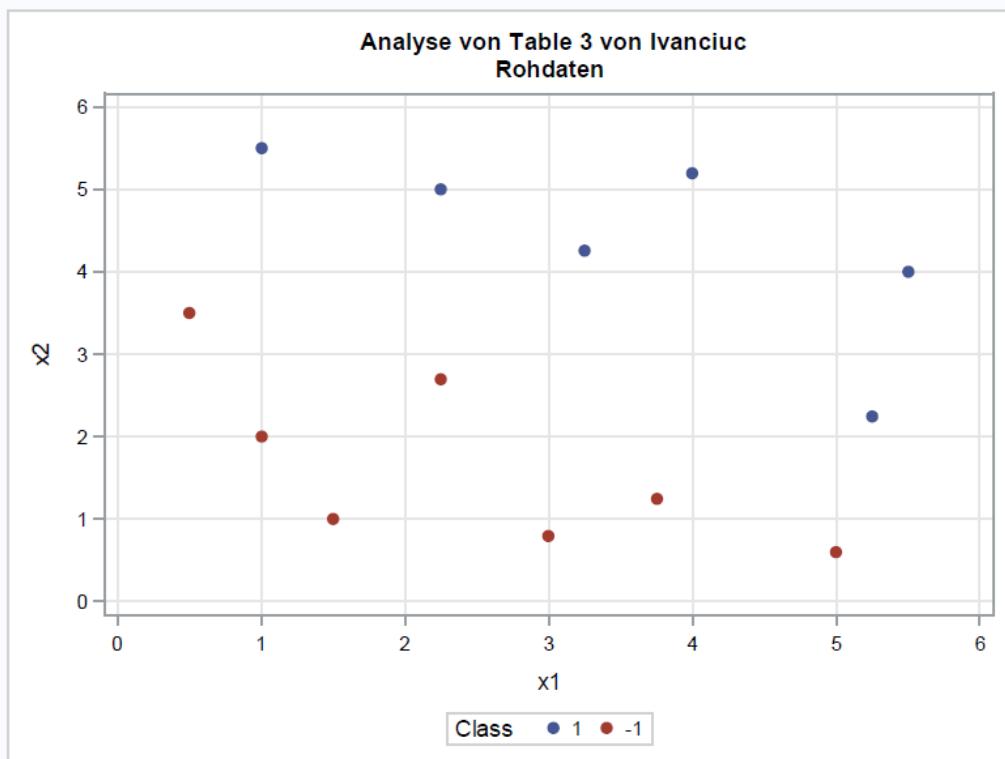
Es gibt verschiedene Arten von Kerneln, wie zum Beispiel:

- lineare Kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- polynomiale Kernel mit Grad d : $(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d$
- radiale Basisfunktion, kurz RBF-Kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$
- sigmoid Kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(p_1 \cdot \langle \mathbf{x}_i, \mathbf{x}_j \rangle + p_2)$

Nun werden die Tabellen aus [OI07] entnommen und mit Hilfe von SVM Vorhersagen für einen bestimmten Klassifizierer gemacht. Es sollte noch berücksichtigt werden, dass in der PDF [OI07] alle Daten sowohl als Training auch als Test-Daten verwendet wurden. Daher wird auch am Ende nur mit Hilfe von Confusion Matrix, auch Klassifikationstabelle genannt, die Resultate ausgewertet. Als erstes wird die Tabelle 3 als ein Beispiel einer SVM-Klassifizierung für linear trennbare Daten vorgestellt.

Pattern	x_1	x_2	Class
1	1	5.5	1
2	2.25	5	1
3	3.25	4.25	1
4	4	5.2	1
5	5.25	2.25	1
6	5.5	4	1
7	0.5	3.5	-1
8	1	2	-1
9	1.5	1	-1
10	2.25	2.7	-1
11	3	0.8	-1
12	3.75	1.25	-1
13	5	0.6	-1

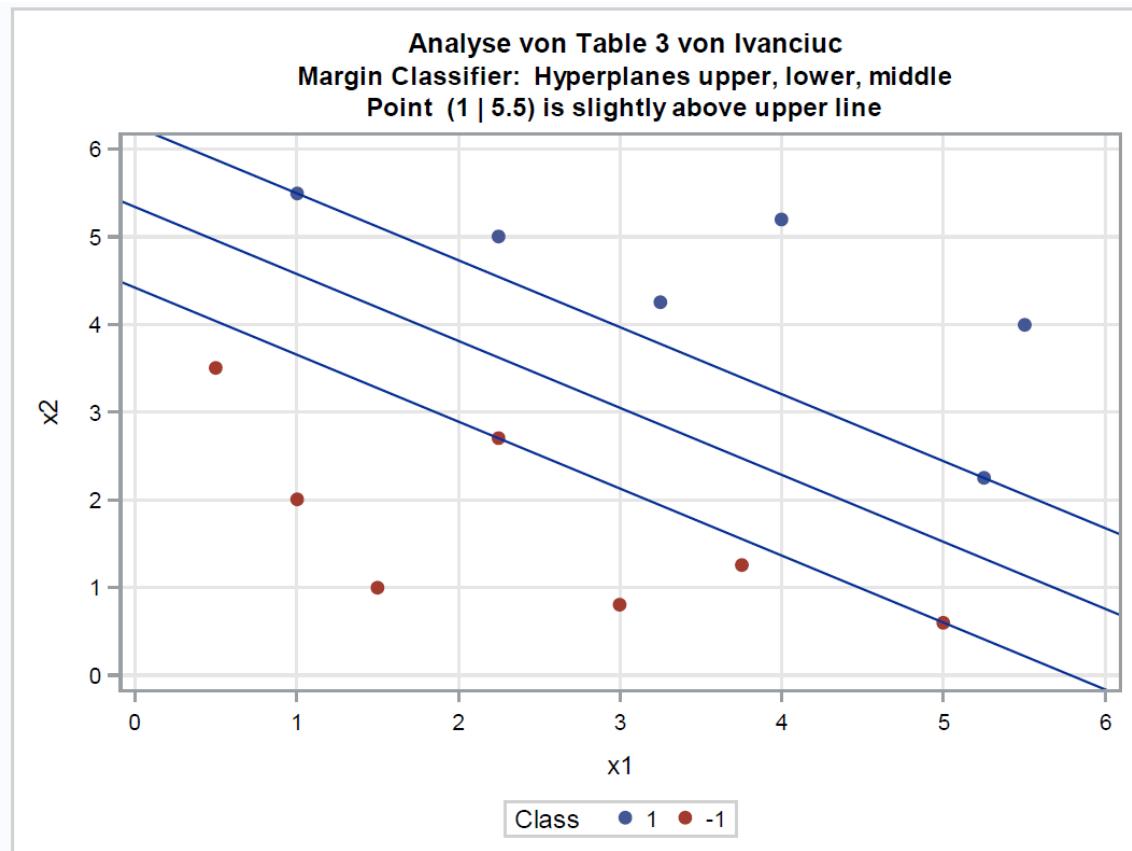
Anhand des untenstehenden Streudiagramms kann man auch bildlich sehen, dass es eine lineare Trennung der Daten möglich ist.



SAS produziert zunächst statistische Informationen über die Variablen x_1 und x_2 wie zum Beispiel Mittelwert, Standardabweichung, Schiefe und Kurtosis. Zusätzlich werden natürlich, neben Optimierungsergebnisse, zum Beispiel aktive Nebenbedingungen, auch unter anderem Anzahl der Support Vectors präsentiert. Diese Auszüge werden im kommenden Erklärungen nicht miteinbezogen und es wird nur auf die Klassifizierung und auf deren Richtigkeit eingegangen. In der untenstehenden Confusion Matrix ist es erkennbar, dass alle Klassen von SVM Prozedur richtig geschätzt wurden.

Classification Table (Training Data)			
Misclassification=0 Accuracy= 100.00			
Beobachtet	Prognostiziert		
	-1	1	Fehlend
-1	7.0000	0	0
1	0	6.0000	0
Missing	0	0	0

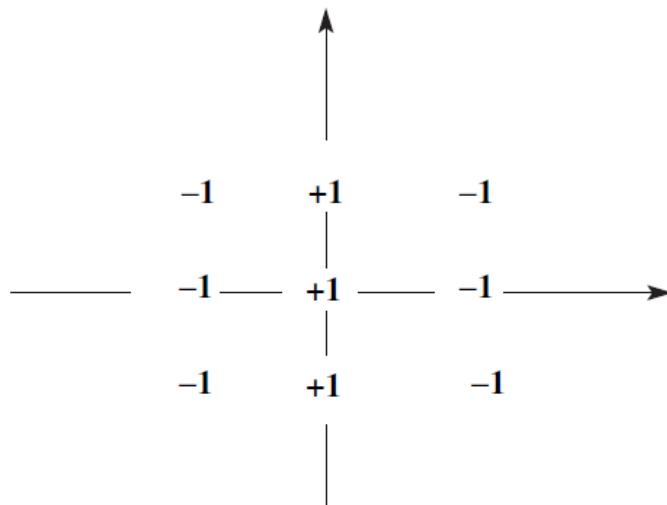
Zusätzlich wurde dann in der SAS Datei noch die Hyperebenen für Tabelle 3 eingezeichnet.



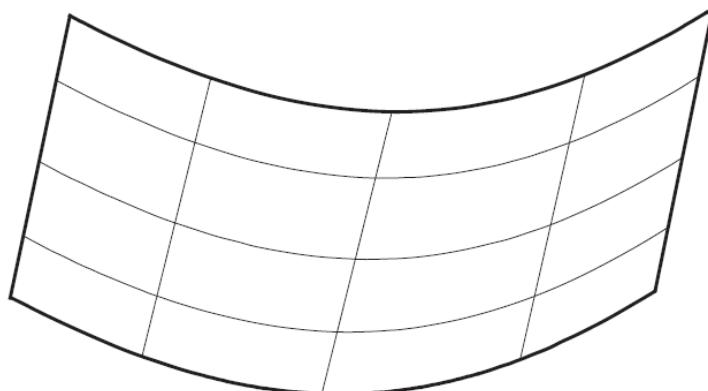
Nun wird die Tabelle 4 aus [OI07] betrachtet. Es ist ein einfaches Beispiel für zunächst linear nicht trennbare Klassen, die dann aber später im Abbildungsraum linear trennbar werden können. Als erstes stelle man die folgende Tabelle nur mit Variablen x_1 und x_2 und die Klassifizierung dazu vor.

Pattern	x_1	x_2	x_1^2	Class
1	-1	-1	+1	-1
2	-1	0	+1	-1
3	-1	+1	+1	-1
4	0	-1	0	+1
5	0	0	0	+1
6	0	+1	0	+1
7	+1	-1	+1	-1
8	+1	0	+1	-1
9	+1	+1	+1	-1

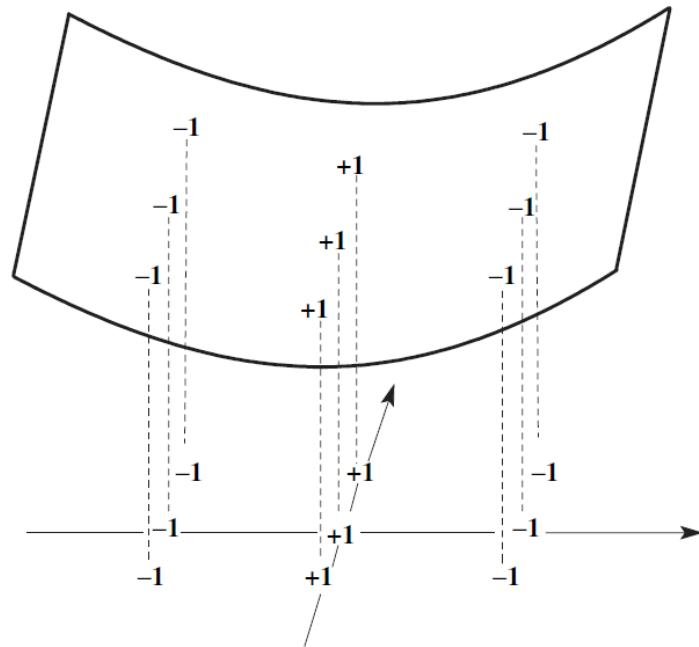
In [OI07] werde die Daten dann wie folgt dargestellt. Dabei bestehen die Daten aus dreimal +1 und sechsmal -1 Klassifizierungen.



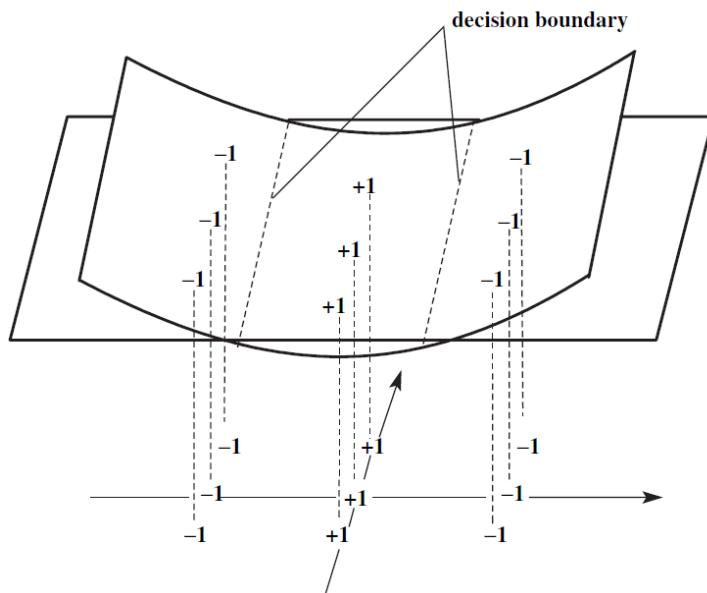
Es wird erhofft in dem höherdimensionalen Abbildungsraum die Klassen linear zu trennen. Hierfür wird nun beispielhaft x_1^2 als eine neue Dimension (in Tabelle 4, Spalte 4) eingefügt. Nach dieser Prozedur werden die Daten in einem drei-dimensionalen Abbildungsraum dargestellt. Die Oberfläche von $f(x_1, x_2) = x_1^2$ wird wie folgt dargestellt.



Nun werden die Abbildungspunkte (x_1, x_2, x_1^2) dargestellt. In der Tabelle 4 erkennt man auch, dass alle Klassen mit $+1$ $x_1^2 = 0$ haben, wobei Klassen mit -1 haben $x_1^2 = +1$.



Nun können die zwei Klassen mit einem linearen Klassifizierer, hier einer Ebene, getrennt werden. Diese Ebene ist natürlich nicht die Einzige, das heißt es gibt noch andere Möglichkeiten von Ebenen, die die Daten auch linear trennen.



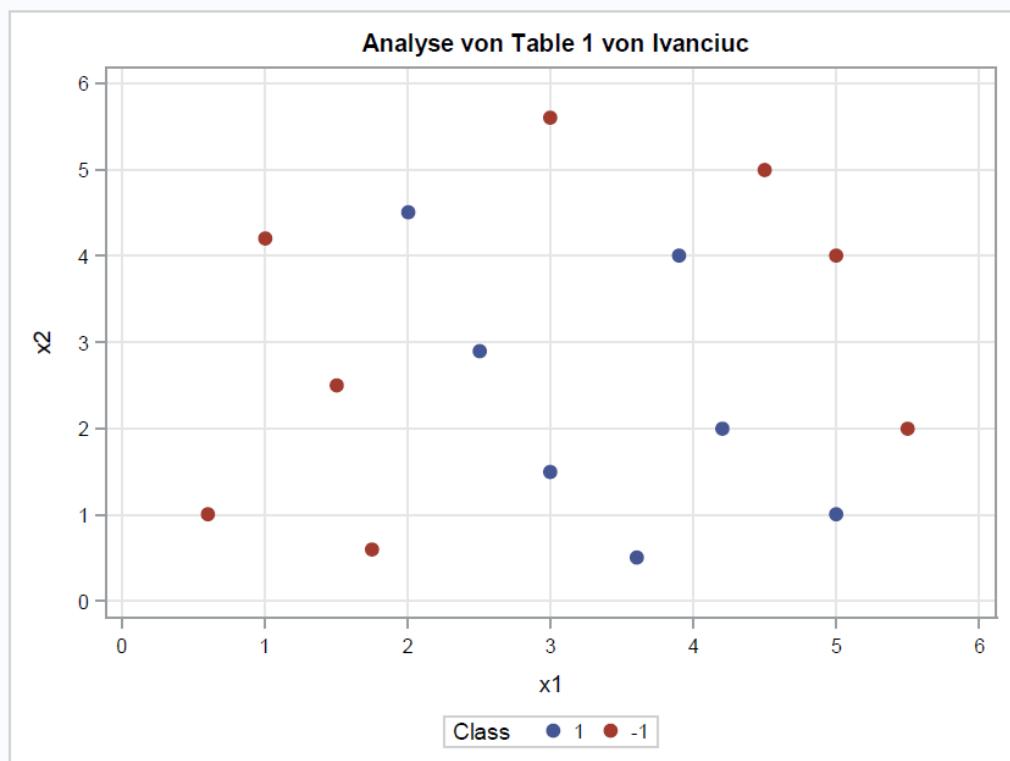
Anhand der Confusion Matrix erkennt man, dass alle Klassen richtig vorhergesagt wurden.

Classification Table (Training Data)			
Misclassification=0 Accuracy= 100.00			
Beobachtet	Prognostiziert		
	-1	1	Fehlend
-1	6.0000	0	0
1	0	3.0000	0
Missing	0	0	0

Die folgenden Tabellen, Tabelle 1, 5 und 7, beinhalten nicht linear trennbare Daten. Daher werden nun verschiedene Kerne wie POLYNOMIAL oder RADIAL verwendet. Zunächst wird die Tabelle 1 betrachtet.

Pattern	x_1	x_2	Class
1	2	4.5	1
2	2.5	2.9	1
3	3	1.5	1
4	3.6	0.5	1
5	4.2	2	1
6	3.9	4	1
7	5	1	1
8	0.6	1	-1
9	1	4.2	-1
10	1.5	2.5	-1
11	1.75	0.6	-1
12	3	5.6	-1
13	4.5	5	-1
14	5	4	-1
15	5.5	2	-1

Werden die Punkte geplottet, so ist es offensichtlich, dass eine lineare Funktion für diesen Datensatz, um eine gute Trennung der beiden Klassen zu erhalten, nicht ausreichen wird.



Würde man nun dennoch die lineare Kernfunktion nutzen, zum Beispiel mit Toleranz C = 0.1, werden folgende Ergebnisse präsentiert.

Classification Table (Training Data)				
Misclassification=6 Accuracy= 60.00				
Beobachtet	Prognostiziert			
	-1	1	Fehlend	
-1	5.0000	3.0000	0	
1	3.0000	4.0000	0	
Missing	0	0	0	

Zusätzlich kann man von der untenstehenden Tabelle entnehmen, dass SVM bei der Vorhersage der Klassifizierungen insgesamt 15 Support Vektoren benötigt. Das heißt alle Beobachtungen werden als Support Vektoren verwendet. Würde man nun dies mit Tabelle 3 vergleichen, in der von 13 Datensätze nur 3 als Support Vektoren verwendet wurden beim linearen Kern, ist es erkenntlich dass ein Vorgehen mit linearer Kernfunktion hier in Tabelle 1 nicht optimal ist.

Support Vector Classification	C_CLAS
Kernel Function	Linear
Estimation Method	FQP
Number of Observations (Train)	15
Number of Effects	2
Regularization Parameter C	0.100000
Classification Error (Training)	6.000000
Objective Function	-1.284611
Inf. Norm of Gradient	8.604228E-16
Squared Euclidean Norm of w	0.165556
Geometric Margin	4.915392
Number of Support Vectors	15
Number of S.Vector on Margin	12
Norm of Longest Vector	6.726812
Radius of Sphere Around SV	6.726812
Estimated VC Dim of Classifier	8.491389
Linear Kernel Constant (Fit)	0.166667
Linear Kernel Constant (PCE)	0.166667
Number of Kernel Calls	62

Wird die RADIALBASIS Funktion aber als Kern verwendet, so werden False Positives gänzlich verhindert.

Classification Table (Training Data)			
Misclassification=4 Accuracy=73.33			
Beobachtet	Prognostiziert		
	-1	1	Fehlend
-1	8.0000	0	0
1	4.0000	3.0000	0
Missing	0	0	0

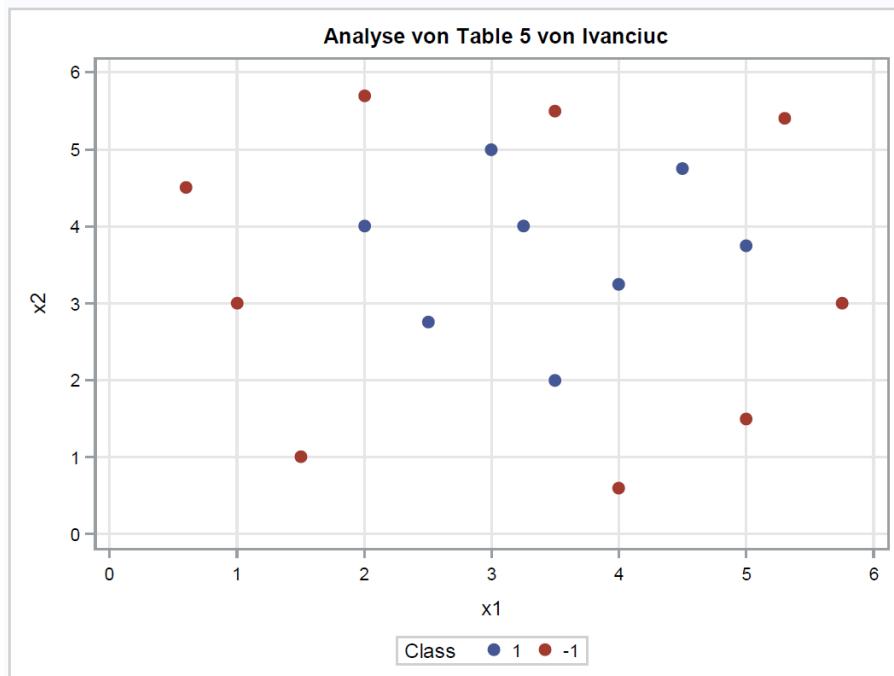
In PROC SVM können natürlich auch die Vorhersagen und die echten Werte in einer Tabelle ausgegeben. So könnte man nachvollziehen, ab wann die Prognose fehlschlägt.

Parameters, Predicted Values, and Residuals						
Nobs	Parm	Y	F_Ziel	I_Ziel	Yhat	Resid
1	0.51045	1.000000	1	-1	-0.253204	1.000000
2	0.51045	1.000000	1	-1	-0.114424	1.000000
3	0.51045	1.000000	1	1	0.165811	0
4	0.51045	1.000000	1	1	0.146507	0
5	0.51045	1.000000	1	1	0.089049	0
6	0.51045	1.000000	1	-1	-0.278115	1.000000
7	0.51045	1.000000	1	-1	-0.047714	1.000000
8	0.31089	-1.000000	-1	-1	-1.000000	0
9	0.51045	-1.000000	-1	-1	-0.926298	0
10	0.51045	-1.000000	-1	-1	-0.944043	0
11	0.40798	-1.000000	-1	-1	-1.000000	0
12	0.46894	-1.000000	-1	-1	-1.000000	0
13	0.39043	-1.000000	-1	-1	-1.000000	0
14	0.46357	-1.000000	-1	-1	-1.000000	0
15	0.51045	-1.000000	-1	-1	-0.847870	0

In Tabelle 5 wurde wieder die RADIALBASIS Kernfunktion verwendet.

Pattern	x_1	x_2	Class
1	2	4	1
2	2.5	2.75	1
3	3	5	1
4	3.5	2	1
5	4.5	4.75	1
6	5	3.75	1
7	3.25	4	1
8	4	3.25	1
9	0.6	4.5	-1
10	1	3	-1
11	1.5	1	-1
12	2	5.7	-1
13	3.5	5.5	-1
14	4	0.6	-1
15	5	1.5	-1
16	5.3	5.4	-1
17	5.75	3	-1

Anhand des Plots kann man auch direkt sehen, dass eine lineare Trennung nicht optimal wäre.



Die SVM Prozedur hat mit Hilfe der RBF Kern nur 2 Klassen nicht richtig vorhersagen können.

Classification Table (Training Data)			
Misclassification=2 Accuracy= 88.24			
Beobachtet	Prognostiziert		
	-1	1	Fehlend
-1	9.0000	0	0
1	2.0000	6.0000	0
Missing	0	0	0

In dem Paper von Ovidiu Ivanciu [OI07] geht es explizit um die Anwendung von SVM in chemischen Prozesse. Daher geht es bei der letzten Tabelle (Tabelle 7) um chemische Verbindungen.

No	Compound	E_{HOMO}	E_{LUMO}	Q^{-1}	MOA	Class
1	tetrachloroethene	-9.902	-0.4367	-0.0372	1	+1
2	1,2-dichloroethane	-11.417	0.6838	-0.1151	1	+1
3	1,3-dichloropropane	-11.372	1.0193	-0.1625	1	+1
4	dichloromethane	-11.390	0.5946	-0.1854	1	+1
5	1,2,4-trimethylbenzene	-8.972	0.5030	-0.2105	1	+1
6	1,1,2,2-tetrachloroethane	-11.655	-0.0738	-0.2785	1	+1
7	2,4-dichloroacetophenone	-9.890	-0.5146	-0.4423	1	+1
8	4-methyl-2-pantanone	-10.493	0.8962	-0.4713	1	+1
9	ethyl acetate	-11.006	1.1370	-0.5045	1	+1
10	cyclohexanone	-10.616	3.3960	-0.5584	1	+1
11	2,4,6-trimethylphenol	-8.691	0.4322	-0.4750	2	-1
12	3-chloronitrobenzene	-10.367	-1.2855	-0.4842	2	-1
13	4-ethylphenol	-8.912	0.4334	-0.4931	2	-1
14	2,4-dimethylphenol	-8.784	0.3979	-0.4980	2	-1
15	4-nitrotoluene	-10.305	-1.0449	-0.5017	2	-1
16	2-chloro-4-nitroaniline	-9.256	-0.9066	-0.6434	2	-1
17	2-chloroaniline	-8.376	0.3928	-0.6743	2	-1
18	pentafluoroaniline	-9.272	-1.0127	-0.8360	2	-1
19	4-methylaniline	-8.356	0.6156	-0.9429	2	-1
20	4-ethylaniline	-8.379	0.6219	-0.9589	2	-1

Zunächst wurde die Deskriptorauswahlprozedur für SVM-Modell durchgeführt und die drei wichtigen Descriptoren E_{HOMO} , E_{LUMO} und Q^{-1} gefunden. Dann wurden 20 Verbindungen (Compound) als Datenset ausgewählt. Es wurde dann unterschieden, ob sich bei Mechanism of Toxic Action, also Mechanismus der toxischen Wirkung, nonpolar bzw. class +1 oder polar bzw. class -1 ergibt.

Würde man nun den linearen Kern mit C=100 oder den RBF Kern wählen, so ergibt sich die folgende Confusion Matrix. Zusätzlich sollte noch erwähnt werden, dass bei allen KERNEL RBF Befehlen K_PAR=1 gesetzt wurde. Dies bedeutet dass $\sigma = 1$ ist.

Classification Table (Training Data)			
Misclassification=0 Accuracy= 100.00			
Beobachtet	Prognostiziert		
	-1	1	Fehlend
-1	10.0000	0	0
1	0	10.0000	0
Missing	0	0	0

2.2 1 V

Aufgabenstellung:

In das SAS Programm **SVM_IRIS_ABC.sas** sind bereits die Fisher Iris Daten mit erster grafischer Darstellung eingearbeitet.

Untersuchen Sie durch Einführen von neuen Variablen und Weglassen von je einer Gruppe die zwei ZWEI-GRUPPEN Fälle

- V1 : Zwei gut getrennte Gruppen.
- V2: Zwei sich überlappende Gruppen.

Stellen Sie die Ergebnisse wie bei A1 U dar und erklären Sie ‚was herauskommt‘.

Lösung:

Der Iris-Datensatz ist einer der bekanntesten historischen Referenz-Datensätze und wurde 1935 von dem amerikanischen Botaniker Edgar Anderson erstellt, der die geographischen Unterschiede der Schwertlilien (Iris) untersuchte. Die von Anderson erhobenen Daten wurden erstmals von Sir Ronald Aylmer Fisher - daher auch der Name Fisher Iris Daten - im Jahre 1936 als Beispiel für die Diskriminanzanalyse multivariater Daten verwendet. Der Datensatz umfasst Messdaten von 150 Irispflanzen, davon jeweils 50 Borsten-Schwertlilien (Iris Setosa), Virginia-Schwertlilien (Iris Virginica) und verschiedenfarbige Schwertlilien (Iris Versicolor). Der Datensatz enthält von jeder dieser 150 Pflanzen die Länge und Breite eines Kelchblatts - *Sepalum* - sowie die Länge und Breite eines Blütenblatts - *Petalum* - in Millimetern [TR15].

Variablen in der Reihenfolge ihrer Erstellung				
#	Variable	Typ	Län	Etikett
1	Species	Char	10	Iris Species
2	SepalLength	Num	8	Sepal Length (mm)
3	SepalWidth	Num	8	Sepal Width (mm)
4	PetalLength	Num	8	Petal Length (mm)
5	PetalWidth	Num	8	Petal Width (mm)

Zusätzlich kann man auch im SAS die Beobachtungen ausgeben lassen. Hier ist ein Ausschnitt von den ersten zehn.

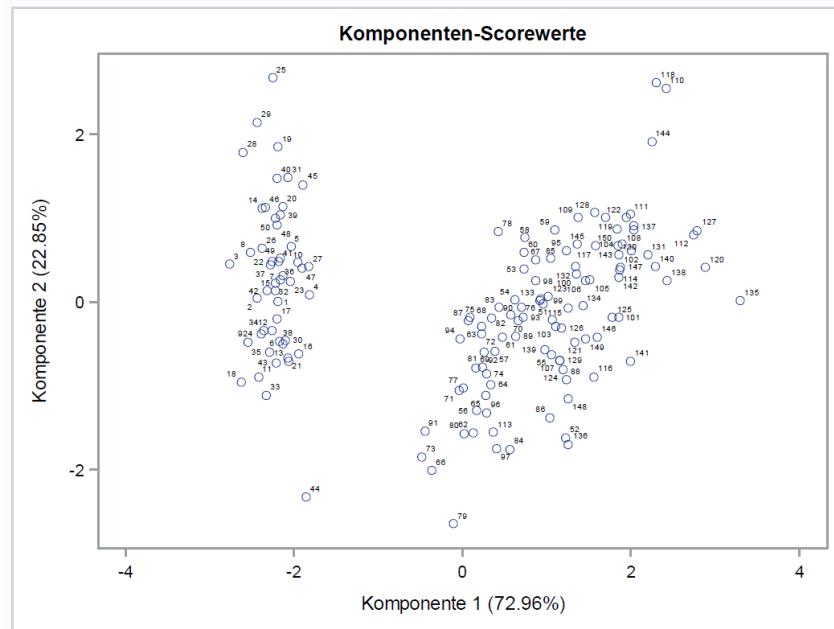
Die ersten zehn Beobachtungen von 150

Beob.	Species	SepalLength	SepalWidth	PetalLength	PetalWidth
1	Setosa	50	33	14	2
2	Setosa	46	34	14	3
3	Setosa	46	36	10	2
4	Setosa	51	33	17	5
5	Setosa	55	35	13	2
6	Setosa	48	31	16	2
7	Setosa	52	34	14	2
8	Setosa	49	36	14	1
9	Setosa	44	32	13	2
10	Setosa	50	35	16	6

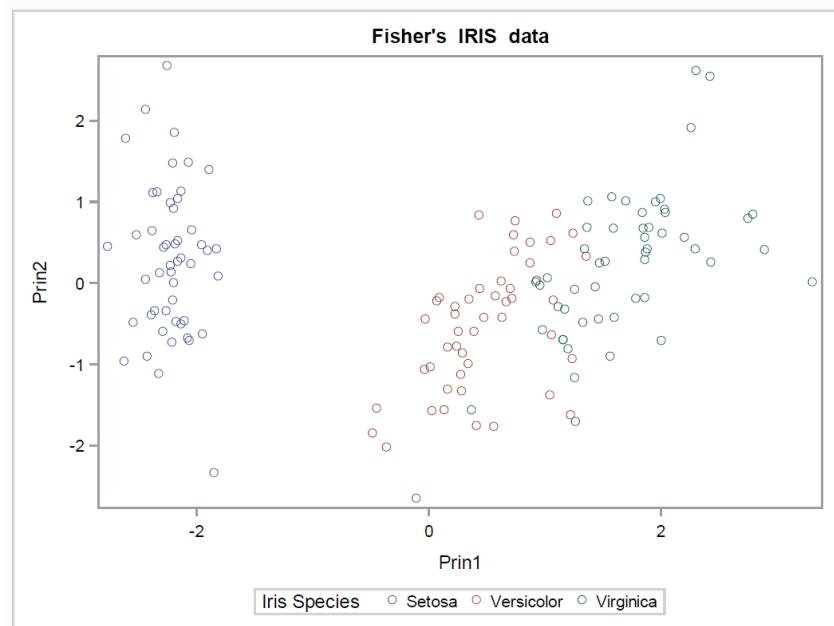
Zunächst folgt eine Hauptkomponentenanalyse (PCA - Principal Component Analysis) des Fisher Datensatzes. Es ist ein Verfahren, das umfangreiche Datensätze strukturiert, vereinfacht und zu veranschaulicht. Dabei werden die ursprünglichen beobachteten Variablen, hier *SepalLength*, *SepalWidth*, *PetalLength* und *PetalWidth*, durch eine geringere Zahl von möglichst aussagekräftigen Linearkombinationen, also den Hauptkomponenten, dargestellt. Es wird hier in dieser Ausarbeitung nicht mehr weiter auf PCA eingegangen, da dies nur eine Hilfestellung für die primäre Aufgabe, SVM-Modell Anwendung auf Iris-Daten, ist.



Eines der Vorteile von PCA ist, dass es Komponenten Plot erstellt werden kann.



Diese wird nun mit Farbe erneut produziert.



Hierbei ist es erkenntlich, dass wenn später anhand von SVM die Klassen *Setosa*, *Versicolor* und *Virginica* vorhergesagt werden sollen, diese dann beim *Versicolor* und *Virginica* Schwierigkeiten bei der Unterscheidung beider Klassen haben wird. Daher werden auch später noch einmal die Daten in EASY und HARD GROUPS unterschieden.

Nun wird das SVM-Modell mit verschiedenen Kernfunktionen angewendet. Es wird die lineare, radialbasis mit $\sigma = 1$, polynomiale mit Grad 3 und sigmoide mit $p_1 = p_2 = 1$ Kernfunktionen verwendet und mit Hilfe der Cross Validation werden die Vorhersagen gemacht.

Classification Table (Cross Validation)				
Observed	SETOSA	VERSICOLOR	VIRGINICA	Missing
SETOSA	50	0	0	0
VERSICOLOR	0	47	3	0
VIRGINICA	0	4	46	0
Missing	0	0	0	0

Misclassification=7 Accuracy= 95.33

Classification Table (Cross Validation)				
Observed	SETOSA	VERSICOLOR	VIRGINICA	Missing
SETOSA	50	0	0	0
VERSICOLOR	0	47	3	0
VIRGINICA	0	3	47	0
Missing	0	0	0	0

Misclassification=6 Accuracy= 96.00

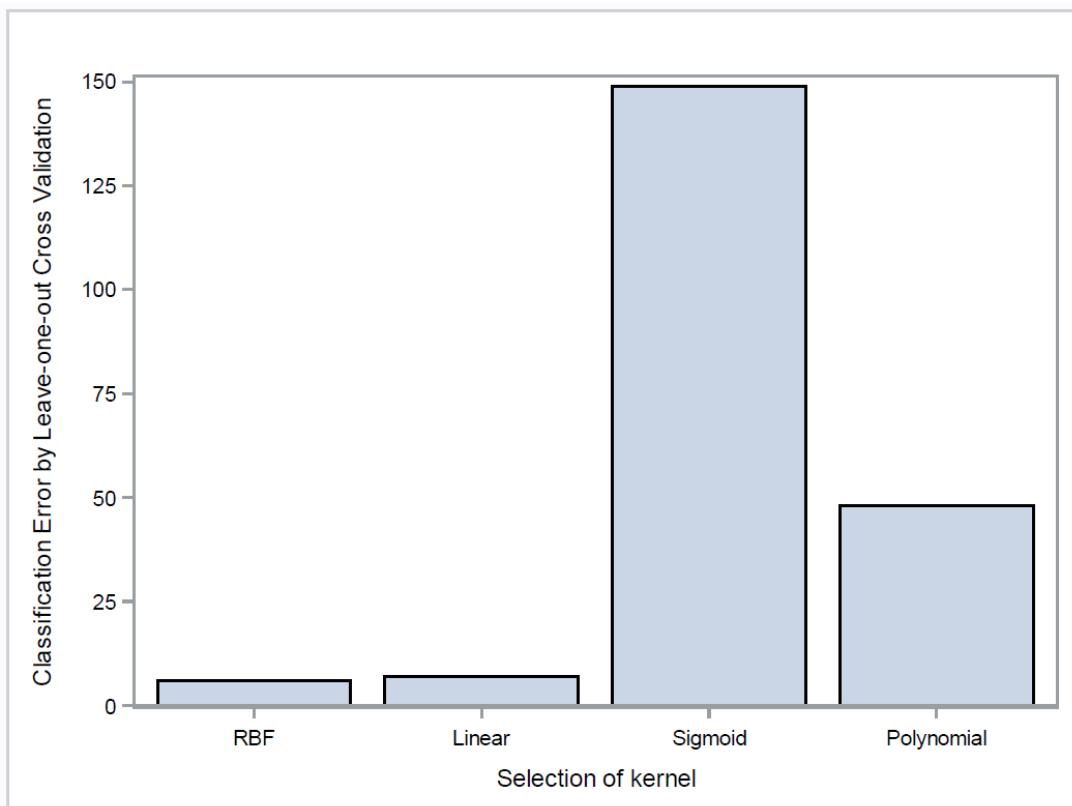
Classification Table (Cross Validation)				
Observed	SETOSA	VERSICOLOR	VIRGINICA	Missing
SETOSA	50	0	0	0
VERSICOLOR	5	2	43	0
VIRGINICA	0	0	50	0
Missing	0	0	0	0

Misclassification=48 Accuracy= 68.00

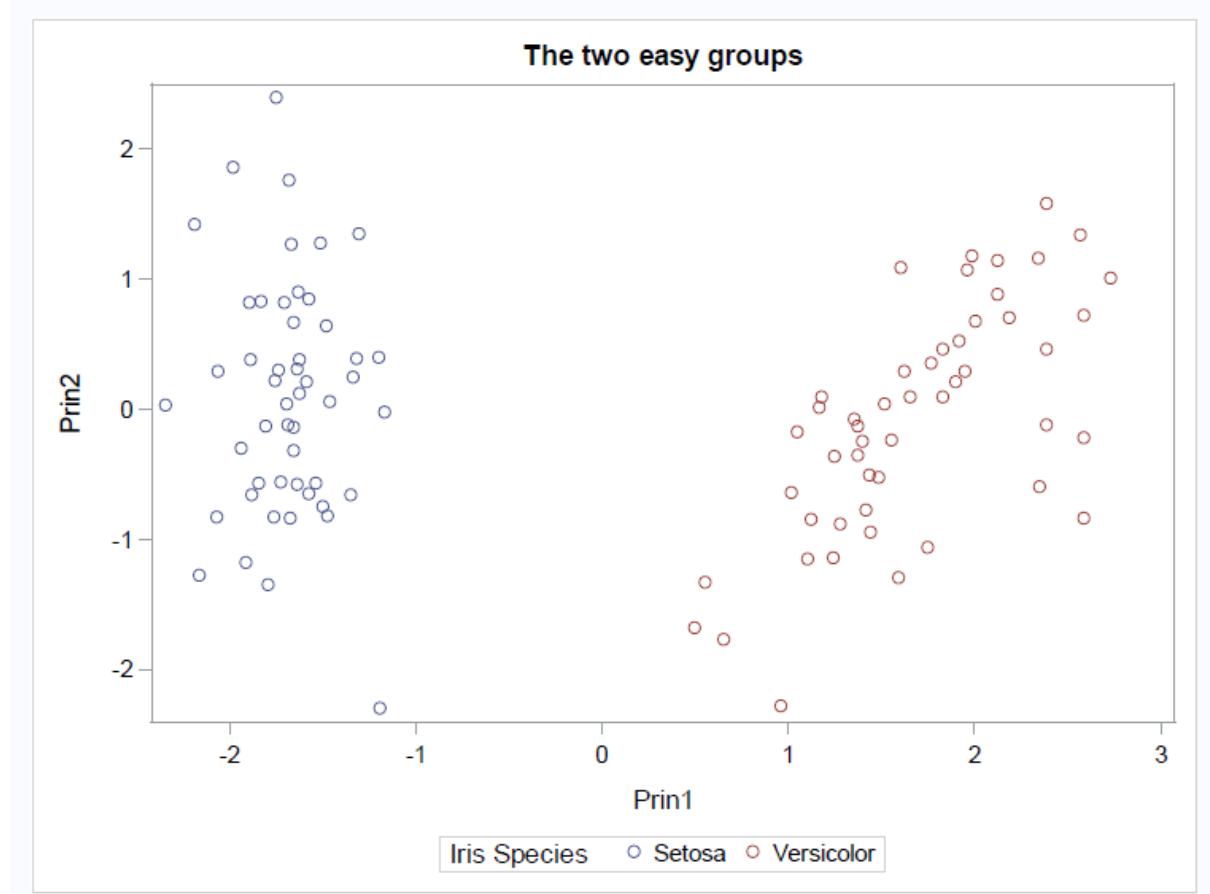
Classification Table (Cross Validation)				
Observed	SETOSA	VERSICOLOR	VIRGINICA	Missing
SETOSA	1	0	49	0
VERSICOLOR	0	0	50	0
VIRGINICA	0	50	0	0
Missing	0	0	0	0

Misclassification=149 Accuracy= 0.67

Werden die verschiedene Kernfunktionen miteinander verglichen, so erkennt man, dass RBF die besten und SIGMOID die schlechtesten Resultate liefert.



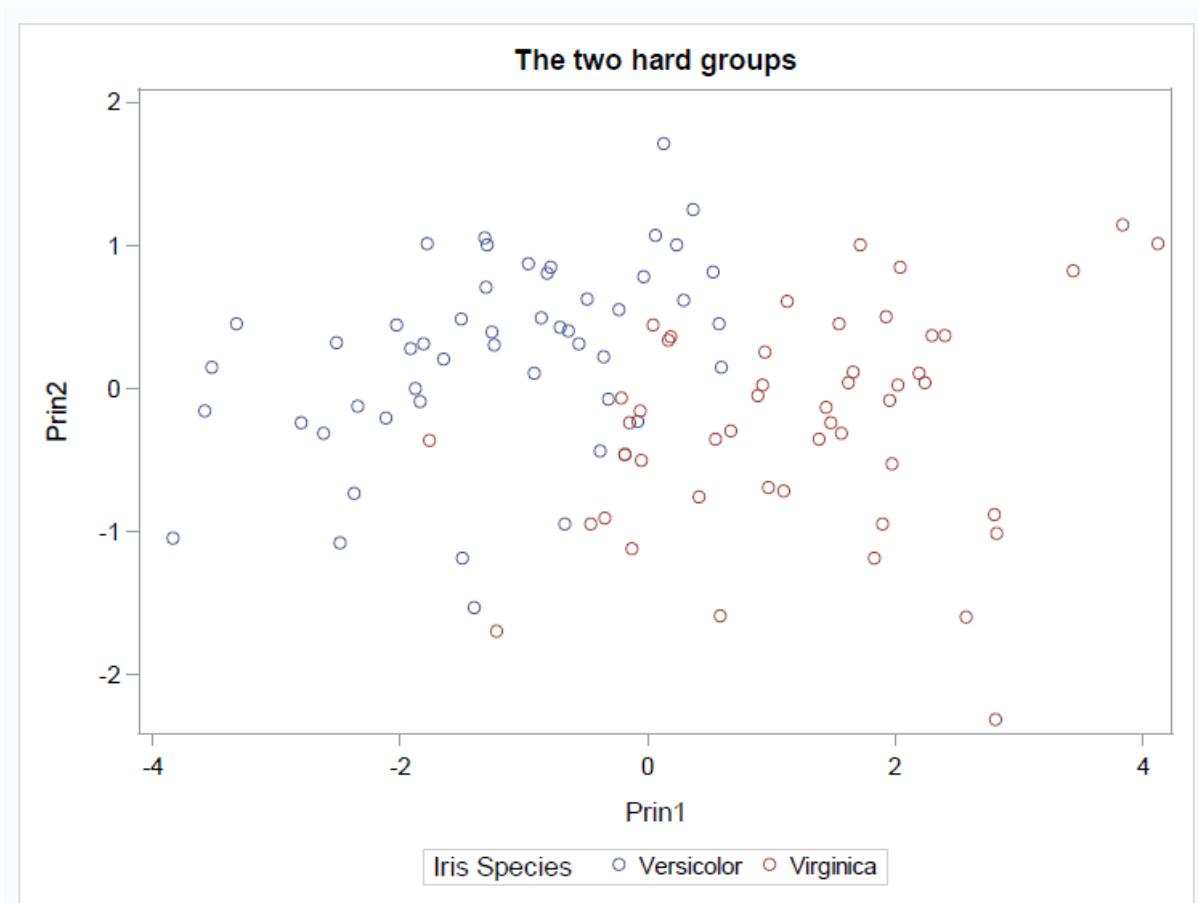
Nun werden, wie oben schon kurz erläutert, die Daten in EASY und HARD GROUPS unterscheidet und so probiert bessere Vorhersagen zu treffen. Zunächst hat man oben im bunten Bild der Komponenten gesehen, dass es beim *Versicolor* und *Virginica* Schwierigkeiten bei der Unterscheidung geben wird. Daher nimmt man nun bei der EASY GROUP einer dieser Klassen, hier alles außer *Virginica*, das heißt die Klassen *Setosa* und *Versicolor* werden vorhergesagt. Man hätte hier auch *Setosa* und *Virginica* als EASY GROUP nehmen können.



Hierbei erkennt man, dass es eine lineare Kernfunktion optimaler Trennung hervorrufen kann. Dies wurde auch für SVM mit $C=1$ und $C=0.001$ und Cross Validation durchgeführt. Bei beiden Toleranzen wurde eine perfekte Trennung durchgeführt und die Klassen wurden immer richtig vorhergesagt.

Classification Table (Cross Validation)			
Misclassification=0 Accuracy= 100.00			
Beobachtet	Prognostiziert		
	SETOSA	VERSICOLOR	Fehlend
SETOSA	50.0000	0	0
VERSICOLOR	0	50.0000	0
Missing OR	0	0	0

Nun wird HARD GROUP analysiert. Hierbei möchte man *Virginica* und *Versicolor* bestmöglich klassifizieren.



Hier wurden wieder verschiedene C Werte, C=1 und C=0.001, ausgetestet.

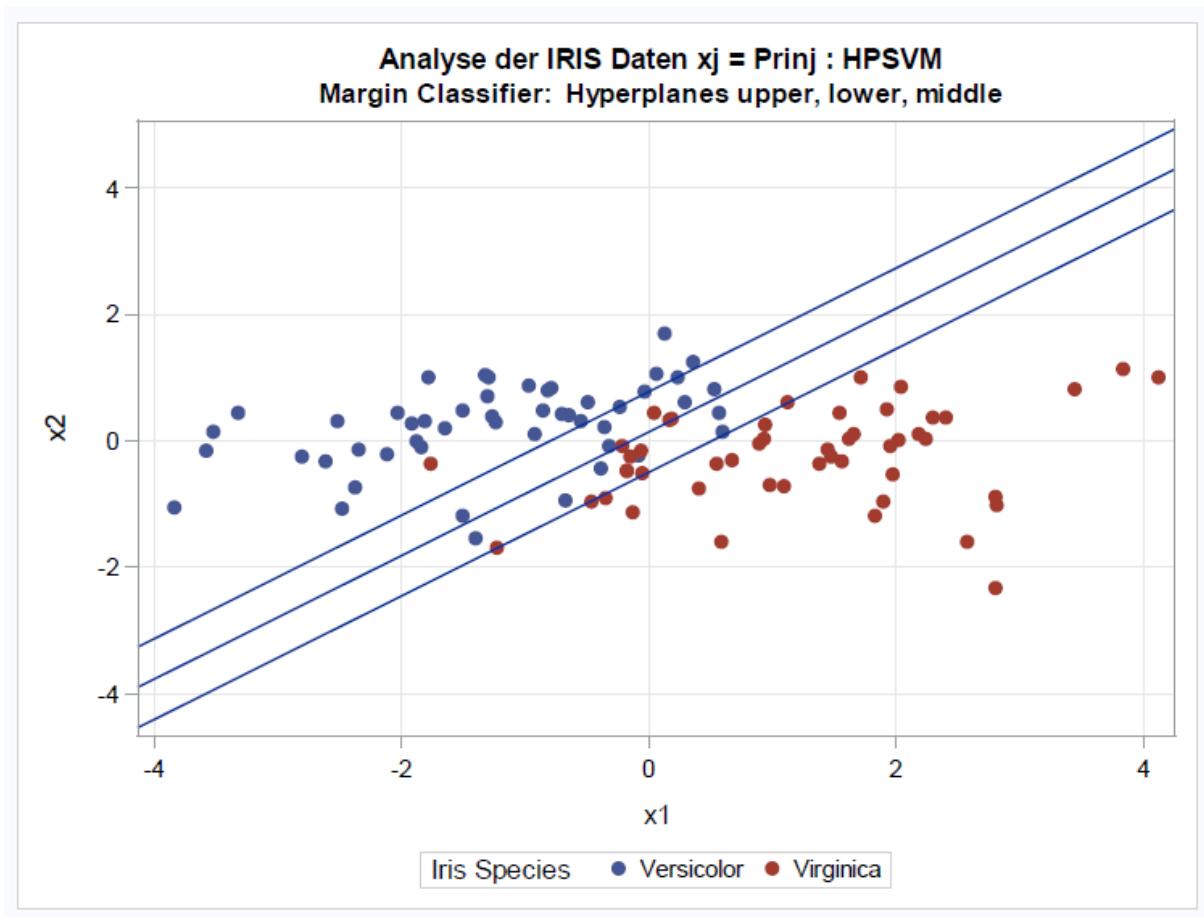
Classification Table (Cross Validation)			
Misclassification=5 Accuracy= 95.00			
Beobachtet	Prognostiziert		
	VERSICOLOR	VIRGINICA	Fehlend
VERSICOLOR	47.0000	3.0000	0
VIRGINICA	2.0000	48.0000	0
Missing A	0	0	0

Classification Table (Cross Validation)			
Misclassification=5 Accuracy= 95.00			
Beobachtet	Prognostiziert		
	VERSICOLOR	VIRGINICA	Fehlend
VERSICOLOR	48.0000	2.0000	0
VIRGINICA	3.0000	47.0000	0
Missing A	0	0	0

Anstatt normale SVM kann auch HPSVM für die lineare Trennung der HARD GROUP verwendet werden.

Die Prozedur HPSVM				
Beobachtet	Klassifizierungsmatrix			Summe
	Versicolor	Virginica	Training-Prognose	
Versicolor	44	6	50	
Virginica	4	46	50	
Total	48	52	100	

Anpassungsstatistiken	
Statistik	Training
Accuracy	0.9000
Error	0.1000
Sensitivity	0.8800
Specificity	0.9200



2.3 1 W

Aufgabenstellung:

In DM 1 wurden für Aufgaben/Projekte mit Kurznamen DONOR (Spendensammlung) | ORGANICS (Bio-Lebensmittel) | CREDIT | INQ2005 ENROLLMENT (Einschreibung) Data Mining Projekte und DM-Flüsse im SAS Enterprise Miner angelegt.

Fügen Sie HP SVM Knoten hinzu. Variieren Sie einige Einstell-Parameter. Bewerten Sie die Performance im Vergleich, aber speziell der SVM-Methoden. Wenn möglich, führen Sie den Effekt der Imbalance Trap (Falle, in die man geraten kann bei großer Unebenheit der binären Ausprägungen – vgl. VL) einmal absichtlich herbei (und vermeiden ihn dann für den Rest Ihres Lebens als Data Scientist!)

Alternativ: Wenn Ihnen dieser SAS EMiner Background fehlt, können Sie die Aufgabe auch wahlweise via R oder später via Python anpacken.

Lösung:

Für diese Aufgabenstellung wurde beispielhaft die Daten von CREDIT ausgearbeitet. Die folgende Erläuterungen wurden aus Data Mining I Booklet, die in Wintersemester 2017/2018 von mir erstellt wurde, entnommen. Hierbei geht es im Prinzip die Erkennung der Wichtigkeit von SMOTE-Methode.

Eine Bank möchte ein neues Kreditprodukt anbieten. Um zukünftige Kreditentscheidungen zu treffen wird ein Risikomodell erstellt. Dabei werden eine Stichprobe von Bewerbern für das ursprüngliche Kreditprodukt als Daten ausgewählt. Es wird nun ein Enterprise Miner Project erstellt, um die besten Vorhersagemodelle zu finden. Zunächst wird die Aufgabe nach den Schritten der in der Vorlesung ausgeteilten PDF ausgearbeitet.

Es folgt ein Ausschnitt der Variablen. Hierbei werden 28 Regressoren für die Vorhersage der abhängigen Variable TARGET verwendet.

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
BanruptcyInd	Input	Binary	No		No	.	.
CollectCnt	Input	Interval	No		No	.	.
DeroqCnt	Input	Interval	No		No	.	.
ID	ID	Nominal	No		No	.	.
InqCnt06	Input	Interval	No		No	.	.
InqFinanceCr	Input	Interval	No		No	.	.
InqTimeLast	Input	Interval	No		No	.	.
TARGET	Target	Binary	No		No	.	.
TL50UtilCnt	Input	Interval	No		No	.	.
TL75UtilCnt	Input	Interval	No		No	.	.
TLBadCnt24	Input	Interval	No		No	.	.
TLBadDeroq	Input	Interval	No		No	.	.
TLBalHCPct	Input	Interval	No		No	.	.
TLCnt	Input	Interval	No		No	.	.
TLCnt03	Input	Interval	No		No	.	.
TLCnt12	Input	Interval	No		No	.	.
TLCnt24	Input	Interval	No		No	.	.
TLDel3060Cr	Input	Interval	No		No	.	.
TLDel60Cnt	Input	Interval	No		No	.	.
TLDel60Cnt2	Input	Interval	No		No	.	.
TLDel60CntA	Input	Interval	No		No	.	.
TLDel90Cnt2	Input	Interval	No		No	.	.
TLMaxSum	Input	Interval	No		No	.	.
TLOpen24Pct	Input	Interval	No		No	.	.
TLOpenPct	Input	Interval	No		No	.	.
TLSatCnt	Input	Interval	No		No	.	.
TLSatPct	Input	Interval	No		No	.	.
TLSum	Input	Interval	No		No	.	.
TLTimeFirst	Input	Interval	No		No	.	.
TLTimeLast	Input	Interval	No		No	.	.

Die ADVANCED OPTIONS werden wie folgt geändert:

The screenshot shows the properties of a 'Data Mining' node. On the left, the 'Advanced Options' section is expanded, showing settings like 'Output Type' (View), 'Role' (Raw), and 'Drop Map Variable' (Yes). In the center, a 'Data Partition' node is connected to a 'CREDIT' node. On the right, the 'Advanced Options' dialog is open, displaying various detection thresholds and behaviors. The 'Class Levels Count Threshold' is set to 2. Below it, a detailed description explains that if 'Detect class levels'=Yes, interval variables with less than the specified number of levels will be marked as NOMINAL. The default value is 20.

Property	Value
General	
Node ID	Ids
Imported Data	
Exported Data	
Notes	
Train	
Output Type	View
Role	Raw
Rerun	No
Summarize	No
Drop Map Variable	Yes
Columns	
Variables	
Decisions	
Refresh Metadata	
Advisor	Advanced
Advanced Options	
Data	
Data Selection	Data Source
Sample	Default
Advisor	
Use the basic setting to set the initial measurement levels and roles based on the variable attributes. Use the advanced setting to set the initial measurement levels and roles based on both the variable attributes and distributions.	

Property	Value
Detect Class Levels	Yes
Class Levels Count Threshold	2
Reject Vars with Excessive Miss	Yes
Missing Percentage Threshold	50
Reject Vars with Excessive Clas	Yes
Reject Levels Count Threshold	20
Identify Empty Columns	Yes
Database Pass-Through	Yes

Class Levels Count Threshold

If "Detect class levels"=Yes, interval variables with less than the number specified for this property will be marked as NOMINAL. The default value is 20.

OK **Cancel**

Der StatExplore-Knoten wurde verwendet, um vorläufige Statistiken über die Zielvariable bereitzustellen. Die folgenden zwei Abbildungen und deren Ergebnisse sind auch in der PDF, welche in Data Mining 1 verteilt wurde, zu finden.

```
Class Variable Summary Statistics
(maximum 500 observations printed)
```

Data Role=TRAIN

Data Role	Variable Name	Role	Number of Levels					Mode Percentage		Mode2 Percentage	
			Levels	Missing	Mode	Percentage	Mode2	Percentage	Mode2	Percentage	
TRAIN	BanruptcyInd	INPUT	2	0	0	84.67	1	15.33			
TRAIN	TARGET	TARGET	2	0	0	83.33	1	16.67			

Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing		Median	Maximum	Skewness	Kurtosis
				Missing	Missing				
CollectCnt	INPUT	0.857	2.161352	3000	0	0	50	7.556541	111.8365
DerogCnt	INPUT	1.43	2.731469	3000	0	0	51	5.045122	50.93801
InqCnt06	INPUT	3.108333	3.479171	3000	0	0	2	2.580016	12.82077
InqFinanceCnt24	INPUT	3.555	4.477536	3000	0	0	2	2.806893	13.05141
InqTimeLast	INPUT	3.108108	4.637831	2812	188	0	24	2.386563	5.626803
TL5UtilCnt	INPUT	4.077904	3.108076	2901	99	0	23	1.443077	3.350659
TL75UtilCnt	INPUT	3.121682	2.605435	2901	99	0	20	1.50789	3.686636
TLBadCnt24	INPUT	0.567	1.324423	3000	0	0	16	4.376858	28.58301
TLBADerogCnt	INPUT	1.409	2.460434	3000	0	0	47	4.580204	48.24276
TLBalHCPct	INPUT	0.648178	0.266486	2959	41	0	0.6955	3.3613	-0.18073
TLCnt	INPUT	7.879546	5.421595	2997	3	0	7	40	1.235579
TLCnt03	INPUT	0.275	0.582084	3000	0	0	0	7	2.805575
TLCnt12	INPUT	1.821333	1.925265	3000	0	0	1	15	1.623636
TLCnt24	INPUT	3.882333	3.396714	3000	0	0	3	28	1.60771
TLDe13060Cnt24	INPUT	0.726	1.163633	3000	0	0	0	8	1.381942
TLDe160Cnt	INPUT	1.522	2.809653	3000	0	0	0	38	3.30846
TLDe160Cnt24	INPUT	1.068333	1.806124	3000	0	0	0	20	3.080191
TLDe160CntAll	INPUT	2.522	3.407255	3000	0	0	1	45	2.564126
TLDe190Cnt24	INPUT	0.814667	1.609508	3000	0	0	0	19	3.623972
TLMaxSum	INPUT	31205.9	29092.91	2960	40	0	24187	271036	2.061138
TLOpen24Pct	INPUT	0.564219	0.480105	2997	3	0	0.5	6	2.779055
TLOpenPct	INPUT	0.496168	0.206722	2997	3	0	0.5	1	0.379339
TLSatCnt	INPUT	13.51168	8.931769	2996	4	0	12	57	0.851193
TLSatPct	INPUT	0.518331	0.234759	2996	4	0	0.5263	1	-0.12407
TLSum	INPUT	20151.1	19682.09	2960	40	0	15546	210612	2.276832
TLTimeFirst	INPUT	170.1137	92.8137	3000	0	6	151	933	1.031307
TLTimeLast	INPUT	11.87367	16.32141	3000	0	0	7	342	6.447907
									80.31043

Als erstes wird eine schrittweise logistische Regressionsanalyse durchgeführt. Hierbei werden 50% der Daten für das Training und 50% für die Validierung verwendet.

Übersicht schrittweise Auswahl

Schritt	Eingegeben	Effekt		Anzahl ein	Chi-Quadrat	Waldsches Chi-Quadrat			Pr > ChiSq
		DF							
1	TLDe160Cnt24	1		1	87.6784				<.0001
2	IMP_TLBalHCPct	1		2	47.3054				<.0001
3	TLDe13060Cnt24	1		3	46.1590				<.0001
4	IMP_TLSatPct	1		4	18.2931				<.0001
5	InqFinanceCnt24	1		5	17.0563				<.0001
6	TLOpenPct	1		6	15.6759				<.0001
7	TLTimeFirst	1		7	7.6235				0.0058
8	IMP_TL75UtilCnt	1		8	6.6041				0.0102
9	BanruptcyInd	1		9	4.4758				0.0344
10	TLCnt03	1		10	4.5268				0.0334
11	TLOpen24Pct	1		11	4.9244				0.0265

Analyse Maximum-Likelihood-Schätzer

Parameter	DF	Schätzung	Standard	Waldsches	Pr > ChiSq	Standardisierter	Exp(Est)
			Fehler	Chi-Quadrat		Schätzer	
Intercept	1	-2.7983	0.4470	39.19	<.0001		0.061
BanruptcyInd 0	1	0.2656	0.1172	5.13	0.0235	0.1420	1.304
IMP_TL75UtilCnt	1	0.0942	0.0318	8.79	0.0030	0.2340	1.099
IMP_TLBalHCPct	1	1.5836	0.3456	21.00	<.0001	-0.4316	4.872
IMP_TLSatPct	1	-3.3490	0.4992	45.01	<.0001	0.1441	0.035
InqFinanceCnt24	1	0.0576	0.0157	13.38	0.0003	0.1656	1.059
TLCnt03	1	-0.3939	0.1613	5.96	0.0146	-0.1163	0.674
TLDe13060Cnt24	1	0.2639	0.0653	16.31	<.0001	0.1143	1.302
TLDe160Cnt24	1	0.1167	0.0426	7.51	0.0061	0.1009	1.124
TLOpen24Pct	1	0.3747	0.1697	4.87	0.0273	0.1887	5.217
TLOpenPct	1	1.6519	0.4983	10.99	0.0009	-0.1343	0.997
TLTimeFirst	1	-0.00259	0.000975	7.05	0.0079		

Odds-Ratio-Schätzer

Effekt	Punktschätzwert
BanruptcyInd 0 vs 1	1.701
IMP_TL75UtilCnt	1.099
IMP_TLBalHCPct	4.872
IMP_TLSatPct	0.035
InqFinanceCnt24	1.059
TLCnt03	0.674
TLDe13060Cnt24	1.302
TLDe160Cnt24	1.124
TLOpen24Pct	1.454
TLOpenPct	5.217
TLTimeFirst	0.997

Es stellt sich heraus, dass nur 11 der Regressoren einen signifikanten Einfluss auf die abhängige Variable TARGET haben.

Nun wird die Vorhersage mit Hilfe der Support Vector Machine (SAS Enterprise Miner: HP SVM) ermittelt. Support Vector Machines sind, wie schon in der Einleitung erwähnt und in der Aufgabe 1 näher erläutert, leistungsstarke maschinelle Lerntechniken für die Klassifizierung und Regression. Hierbei kann man mit INTERIOR POINT und mit ACTIVE SET arbeiten. Näheres kann in [SA10] nachgelesen werden. Nachdem HPSVM-Knoten mit verschiedene Möglichkeiten (INTERIOR POINT und ACTIVE SET mit verschiedene Kernfunktionen) ergab ACTIVE SET mit radialer Basisfunktion das bestmögliche Vorhersagemodell. Daher werden auch nur die Ergebnisse der SVM mit radialer Basisfunktion als Kern untersucht.

The screenshot shows two windows from SAS Enterprise Miner. On the left is the 'Properties' window for a node named 'HPSVM'. The 'General' tab is selected, showing details like Node ID, Imported Data, Exported Data, Notes, Train settings (Variables, Maximum Iteration 25, Use Missing as LevNo, Tolerance 1.0E-6, Penalty 1.0), Optimization Method (Active Set), and Status (Create Time 20.09.17 23:40, Run ID 229566f9-5ffa-43, Last Error, Last Status Complete). The 'Active Set Options' section is expanded, describing options specific to the active set optimization method. On the right is the 'Active Set Options' dialog box. It contains a table with properties and values:

. Property	Value
Kernel	Radial Basis Function
Polynomial Degree	2
RBF Parameter	1.0
Sigmoid Parameter 1	1.0
Sigmoid Parameter 2	-1.0

Below the table, the 'Kernel' section is expanded, stating: 'Specifies the kernel type that the support vector machine uses.' At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Zunächst wird ein Klassifizierungsmatrix angegeben. Hierbei sollte dies mit Vorsicht betrachtet werden, da die Spalten und Zeilen (Beobachtet = True Class und Training-Prognose = Hypothesized Class) vertauscht sind. Zudem werden viele Statistiken wie zum Beispiel AVERAGE SQUARED ERROR aus-

gegeben.

Klassifizierungsmatrix

Beobachtet	Training-Prognose			Validierungs-Prognose		
	1	0	Summe	1	0	Summe
1	0	250	250	0	249	249
0	0	1250	1250	0	1248	1248
Total	0	1500	1500	0	1497	1497

Fit Statistics

Target=TARGET Target Label=' '

Statistics	Statistics Label	Fit	
		Train	Validation
ASE	Average Squared Error	0.14	0.15
DIV	Divisor for ASE	3000.00	3000.00
MAX	Maximum Absolute Error	0.98	1.00
NOBS	Sum of Frequencies	1500.00	1500.00
RASE	Root Average Squared Error	0.38	0.38
SSE	Sum of Squared Errors	428.99	440.69
DISF	Frequency of Classified Cases	1500.00	1500.00
MISC	Misclassification Rate	0.17	0.17
WRONG	Number of Wrong Classifications	250.00	250.00

Nun werden die beiden Modelle miteinander verglichen.

Fit Statistics Table	Reg	HPSVM
Target: TARGET		
Data Role=Train		
Statistics		
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.24	0.05
Train: Kolmogorov-Smirnov Statistic	0.43	0.57
Train: Akaike's Information Criterion	1134.91	
Train: Average Squared Error	0.11	0.14
Train: Roc Index	0.79	0.82
Train: Average Error Function	0.37	.
Train: Cumulative Percent Captured Response	34.00	42.00
Train: Percent Captured Response	15.60	18.80
Selection Criterion: Valid: Misclassification Rate	0.16	0.17
Train: Degrees of Freedom for Error	1488.00	.
Train: Model Degrees of Freedom	12.00	.
Train: Total Degrees of Freedom	1500.00	.
Train: Frequency of Classified Cases	.	1500.00
Train: Divisor for ASE	3000.00	3000.00
Train: Error Function	1110.91	.
Train: Final Prediction Error	0.12	.
Train: Gain	240.00	320.00
Train: Gini Coefficient	0.58	0.64
Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.43	0.58
Train: Kolmogorov-Smirnov Probability Cutoff	0.22	0.05
Train: Cumulative Lift	3.40	4.20
Train: Lift	3.12	3.76
Train: Maximum Absolute Error	0.98	0.98
Train: Misclassification Rate	0.15	0.17
Train: Mean Square Error	0.11	.
Train: Sum of Frequencies	1500.00	1500.00
Train: Number of Estimate Weights	12.00	.
Train: Root Average Sum of Squares	0.34	0.38
Train: Cumulative Percent Response	56.67	70.00
Train: Percent Response	52.00	62.67
Train: Root Final Prediction Error	0.34	.
Train: Root Mean Squared Error	0.34	.
Train: Schwarz's Bayesian Criterion	1198.67	.
Train: Sum of Squared Errors	341.41	428.99
Train: Sum of Case Weights Times Freq	3000.00	.
Train: Number of Wrong Classifications	.	250.00

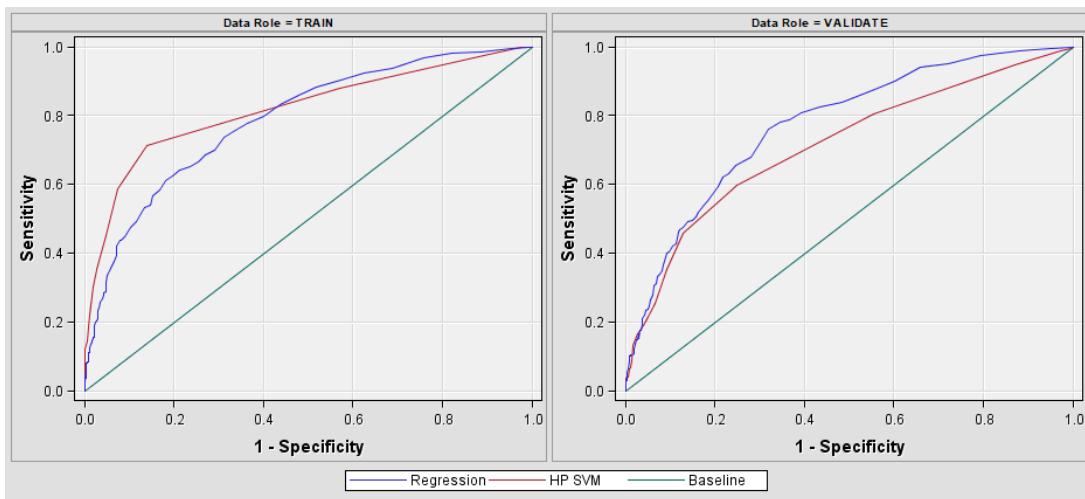
Data Role=Valid	Reg	HPSVM
Statistics		
Valid: Kolmogorov-Smirnov Statistic	0.44	0.35
Valid: Average Squared Error	0.12	0.15
Valid: Roc Index	0.77	0.71
Valid: Average Error Function	0.38	.
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.15	0.05
Valid: Cumulative Percent Captured Response	28.40	25.20
Valid: Percent Captured Response	13.60	8.80
Valid: Frequency of Classified Cases	.	1500.00
Valid: Divisor for VASE	3000.00	3000.00
Valid: Error Function	1143.41	.
Valid: Gain	184.00	152.00
Valid: Gini Coefficient	0.55	0.43
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.44	0.36
Valid: Kolmogorov-Smirnov Probability Cutoff	0.14	0.05
Valid: Cumulative Lift	2.84	2.52
Valid: Lift	2.72	1.76
Valid: Maximum Absolute Error	0.98	1.00
Valid: Misclassification Rate	0.16	0.17
Valid: Mean Square Error	0.12	.
Valid: Sum of Frequencies	1500.00	1500.00
Valid: Root Average Squared Error	0.34	0.38
Valid: Cumulative Percent Response	47.33	42.00
Valid: Percent Response	45.33	29.33
Valid: Root Mean Square Error	0.34	.
Valid: Sum of Square Errors	355.85	440.69
Valid: Sum of Case Weights Times Freq	3000.00	.
Valid: Number of Wrong Classifications	.	250.00

Unten ist das CLASSIFICATION TABLE abgebildet. Dabei werden direkt die True Negativ, True Positiv, False Positive und False Negative ausgegeben. Hierbei sticht eine Unregelmäßigkeit ins Auge, und zwar dass die SVM-Modell für TP und FP nichts vorhergesagt hat. Dieses Verhalten könnte mit Hilfe von SMOTE-Methode vermieden werden. Näheres dazu wird ab Aufgabe 3 erläutert.

Event Classification Table
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Model Node	Model Description	Data Role	Target	Target Label	False Negative	True Negative	False Positive	True Positive
Reg	Regression	TRAIN	TARGET		200	1219	31	50
Reg	Regression	VALIDATE	TARGET		206	1210	40	44
HPSVM	HP SVM	TRAIN	TARGET		250	1250	0	0
HPSVM	HP SVM	VALIDATE	TARGET		250	1250	0	0

Ferner werden die ROC-Kurven beider Modelle abgebildet.

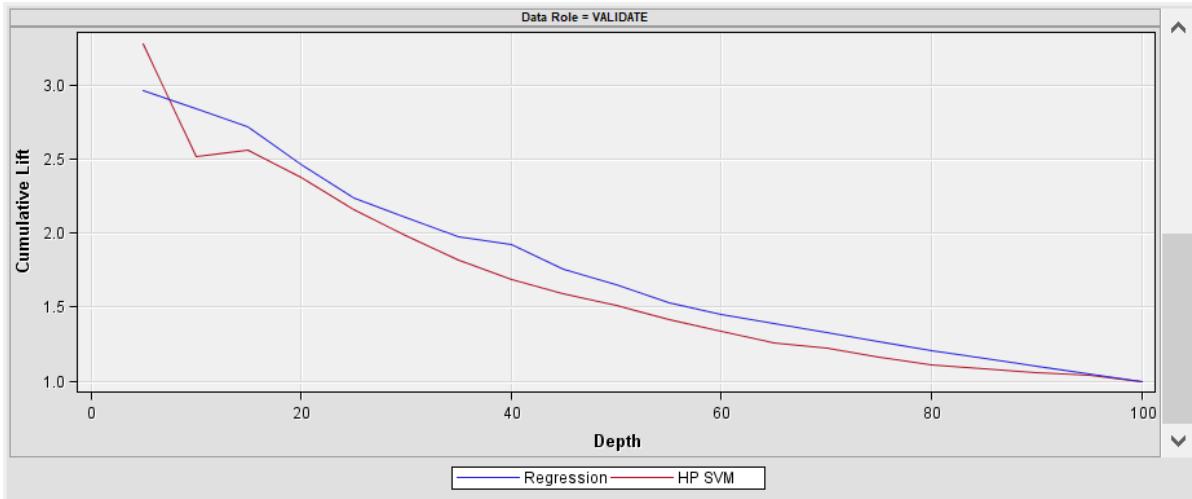


Laut der Aufgabenstellung soll mit der Validierungsdaten argumentiert werden. Hierbei ergab, dass der Support Vector Machines einen höheren AVERAGE SQUARED ERROR hatte als bei der schrittweisen logistischen Regression. Zusätzlich erkennt man anhand der ROC-Index und ROC-Kurve, dass der schrittweise logistische Regression besser als Vorhersagemodell für die Kreditwürdigkeit passt als SVM.

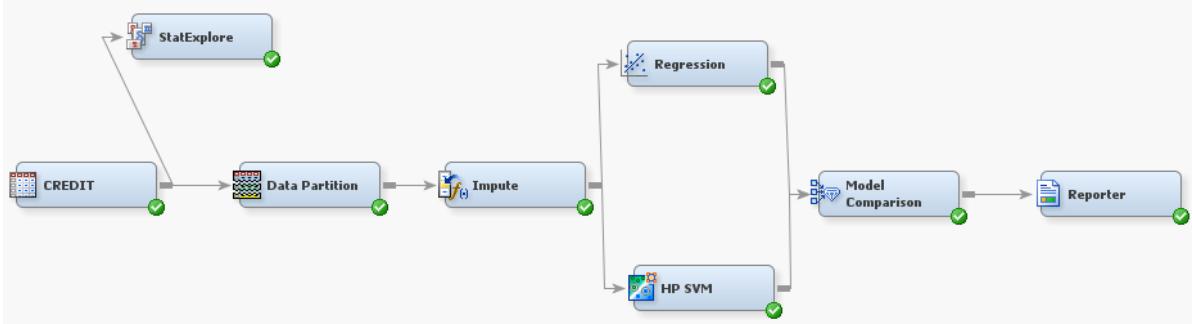
Lift und Gain Charts visualisieren wie gut ein Vorhersagemodell ist. Da in der Vorlesung vermehrt die kumulativen Lift Charts betrachtet wurde, werden auch hier nur diese interpretiert. Ferner werden auch hier nur die Validierungsergebnisse der Modelle abgebildet. Die Frage ist wie viele Kunden müssten mit der neuen Kreditprodukt bekannt gemacht werden, damit diese sich dafür interessieren. Der kumulative Lift Chart zeigt visuell den Vorteil der Verwendung eines der Vorhersagemodele und hilft bei der Frage, wie viel wahrscheinlicher kann ein Kunde auf das neue Kreditprodukt aufmerksam gemacht werden im Vergleich einer zufällige Kontaktaufnahme der Kunden.

Es ist hier zu erkennen, dass bei beiden Modellen etwa 50% der möglichen Neukunden erreicht wird, wenn die Top 20% der Kunden kontaktiert werden - da ein Lift von etwa 2,5 herrscht.

Des Weiteren kann anhand der Grafik auch gesagt werden, dass bei der schrittweisen logistischen Regression mehr möglichen Neukunden erreicht werden können.



Zum Schluss folgt die Abbildung der Aufgabe 10 im Ganzen.



3 Aufgabe 2

Aufgabenstellung:

KAGGLE oder kaggle ist die Kurzform für die am Kopf des u.a. Programmes genannte Web Seite, die Einführungen, Datensammlungen und fortlaufende Wettbewerbe von DM- und DS-Aufgaben anbietet.

- A Schauen Sie sich Details von kaggle an. Melden Sie sich an.
- B Lesen Sie die Beschreibung der Datei/Aufgabe TITANIC (Merkmale/Bedeutung auf kaggle).

Insbesondere ist Y == Survived bzw. train|test\$Survived die primäre binäre TARGET Variable.

Y == Survived = 1 bedeutet, dass die(se) Person überlebt hat.

Sie erhalten ein R-Skript **KAGGLE_TITANIC_LOGREG_A1.R**.

Verstehen Sie die einzelnen Schritte dieses Skriptes. Zum besseren Nachvollziehen können Sie einige Ausgaben via print(), summary() und durch das Erzeugen von *.csv Dateien einfügen.

Klarerweise müssen Sie die Pfade auf eigene Pfade anpassen!

Beachten Sie die gewählten Dezimaltrenner und Separatoren (für deutsches MS-Excel). Diese können leicht umgestellt werden auf internationale Form.

Kommandos vom Typ install.packages(„name“) sind ggf. auskommentiert um häufiges Nachladen zu reduzieren. Ggf. müssen diese von Ihnen einmalig aktiviert werden.

Vorgestellt wird eine einfache, aber gute Methode: Bin. Logistische Regression – mit R.

Prüfen Sie die Abfolge Ihrer eigenen Dateien (z.B. in Excel anschauen ...)

Wenn Sie ein Problem feststellen, kann es sein, dass an einigen Stellen noch nachgebessert werden muss.

Verwendet wird eine brutal simple Methode: die Beobachtungen Nr. 1-800 gehen in eine Trainingsmenge; die Beobachtungen Nr. 801-889 gehen in eine Validierungsmenge, die hier aber als Test-Menge bezeichnet wird.

Wenn Sie sicher sind, dass alles OK und verstanden ist, reduzieren Sie die Outputs, speziell die summary().-Aufrufe und manche write.table(). Checken Sie gegenüber der Vorlage.

Erstellen Sie eine B-Version des Skriptes: Verwenden Sie die Startzufallszahl von <12345> und erzeugen Sie neue <Titanic_train_nr> und <Titanic_test_nr> Dateien, indem Sie die Aufteilung randomisieren (zufällig machen). Protokollieren, erläutern und bewerten Sie die neuen Ergebnisse. 60:40 oder 70:30 oder 80:20 sind gängige Verhältnisse train:test bzw. train:valid.

Die Bewertungen natürlich auf der test == valid(at)ion)-Datei

Ab hier gabelt sich die Aufgabe ::

Sie können/sollten die Daten nach SAS und Python bringen und mit den dortigen Möglichkeiten analysieren und Unterschiede/Übereinstimmungen festhalten und bewerten.

Sie können weitere alternative Modelle neben die Logistische Regression setzen, etc. pp.

Checken Sie, „was kaggle sagt“.

Ein möglicher weiterer Weg besteht darin, NACH! Bearbeitung der Aufgabe 3 den SMOTE-Algorithmus einzusetzen. Also dem nachstehenden Vorschlag von Aufgabe 3 zu folgen: Zitat :

Variieren Sie die Prozentsätze in SMOTE und protokollieren die Unterschiede ...

Gibt es so etwas wie ein bestes Ergebnis ?

- C Eigene freie Ideen der Weiterentwicklung.
- D Exportieren Sie die relevanten Dateien nach SAS (*.sas7bdat) und bauen Sie echt vergleichbare SAS EMiner Modelle auf, dokumentieren in Kurzform die Ergebnisse.

Bewerten und vergleichen Sie die R-Ergebnisse mit den SAS-Ergebnissen.

- E Versuchen Sie dasselbe mit/über/via PYTHON. Dazu gibt es verschiedene Vorgehensweisen. Werden Sie kreativ. Dokumentieren Sie stets Ihre Ergebnisse.

Bewerten und vergleichen Sie die R-Ergebnisse mit den SAS-Ergebnissen und den PYTHON-Ergebnissen (Ergebnisse schließt Bewertung der Programme/Skripte und des Outputs plus Interpretation ein).

Lösung:

Ein sehr häufig angewandter Demonstrationsdatensatz für Klassifikationen ist eine Studie über das Sinken der Titanic, für 1309 Passagiere jeweils angegeben ist, ob dieser überlebt hat oder nicht, sowie zusätzlichen Informationen über den Passagieren. Auf der Webseite <https://www.kaggle.com/c/titanic/data> (Kurzform KAGGLE oder kaggle) Einführungen, Datensammlungen und fortlaufende Wettbewerbe von DataMining- und DataScience-Aufgaben angeboten.

Hier ab Aufgabe 2 werden die Datensätze in Test- und Trainingsdaten unterteilt. Mit Hilfe des Trainingsdatensatzes soll ein Machine Learning Modell erstellt werden. Durch den Testdatensatz wird dann überprüft, wie gut das Modell mit neuen Daten funktioniert bzw. ist das Ziel, eine Vorhersage über das Überleben der Personen im Testdatensatz zu treffen. Folgende Variablen sind im Datensatz vorhanden:

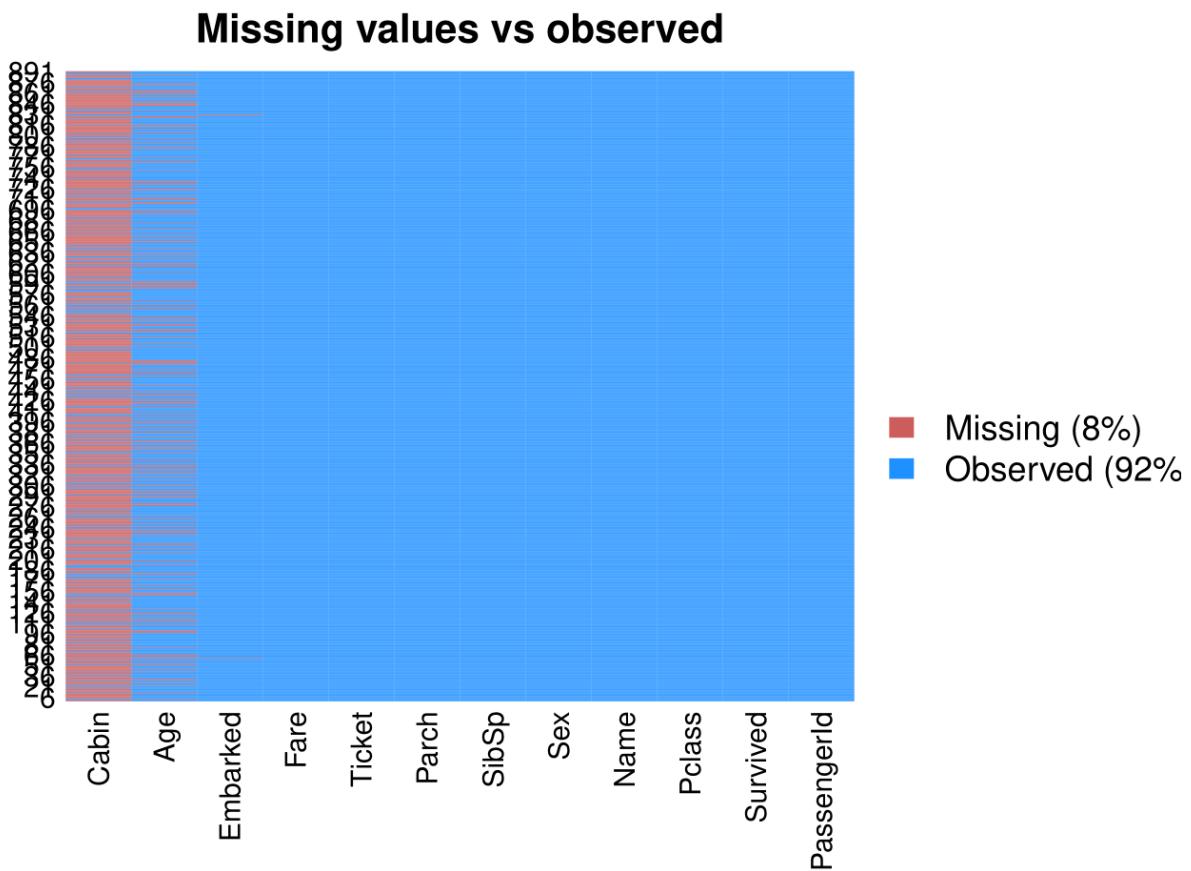
- PASSENGERID: Laufende Nummer
- SURVIVED: Überlebensvariable mit 0=Nein und 1=Ja
- PCLASS: Klasse auf dem Schiff 1=Erste, 2=Zweite und 3=Dritte
- NAME: Nachname, Titel, Vorname(n)
- SEX: Geschlecht mit male=männlich und female=weiblich
- AGE: Alter in Jahren
- SIBSP: Anzahl der Geschwister bzw. Partner, die mit an Bord waren
- PARCH: Anzahl der Eltern bzw. Kinder, die mit an Bord waren
- TICKET: ID Nummer des Tickets
- FARE: Preis des Tickets
- CABIN: Kabinennummer
- EMBARKED: Einstiegshafen mit C=Cherbourg, Q=Queenstown und S=Southampton

In der PDF, TITANIC_BESCHREIBUNG_A, wurden leider die Trainingsdaten von 891 Beobachtungen als Rohdaten genommen und später zum Testen wird dieser Datensatz dann in TRAINING und TEST Datensätze unterteilt. Das heißt die 418 Beobachtungen die eigentlich als Testen gedacht sind werden hier nicht verwendet!

Zu Beginn müssen erstmal alle Daten in R eingelesen werden und zur Analyse bereinigt werden. Das heißt der Datensatz muss für die Analyse vorbereitet werden, falls irgendwelche fehlende Werte oder als NA kodierte Zellen auftauchen. Somit ist es wichtig, sich anzuschauen wie viele Ausprägungen es pro Variable im Datensatz gibt und wie viele Werte pro Variable fehlen:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	0	0	0	0	177	0
Parch	Ticket	Fare	Cabin	Embarked		
0	0	0	687	2		

Für die Variable Age fehlen 177 und für die Variable Cabin 687 Werte. Dies kann zudem durch folgende Grafik veranschaulicht werden:



Die Variable Cabin hat eindeutig zu viele fehlende Werte, wodurch diese Variable nicht mehr benutzt werden kann. Zusätzlich werden auch die Variablen PassengerId und Ticket vernachlässigt, da es sich bei denen nur um einen Index handelt.

Unter Verwendung der Funktion `subset()` werden daher nur die relevanten Spalten ausgewählt, das heißt die nicht-relevanten Variablen Cabin, PassengerID, Ticket und auch Name, werden aus dem Datensatz entfernt. Somit sehen die Daten so wie im untenstehenden Excel-Sheet aus.

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22	1	0	7,25	S
1	1	female	38	1	0	71,2833	C
1	3	female	26	0	0	7,925	S
1	1	female	35	1	0	53,1	S
0	3	male	35	0	0	8,05	S
0	3	male	NA	0	0	8,4583	Q
0	1	male	54	0	0	51,8625	S
0	3	male	2	3	1	21,075	S

Im nächsten Schritt möchte man nun die fehlenden Werte (NA) ersetzen. Dabei gibt es verschiedene Methoden hierzu, aber hier wird der Mittelwert verwendet. Abschließend werden die Zeilen entfernt in denen für die Variable Embarked die Werte, diese waren 2 Zeilen, gefehlt haben.

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22	1	0	7,25	S
1	1	female	38	1	0	71,2833	C
1	3	female	26	0	0	7,925	S
1	1	female	35	1	0	53,1	S
0	3	male	35	0	0	8,05	S
0	3	male	29,6991176	0	0	8,4583	Q
0	1	male	54	0	0	51,8625	S
0	3	male	2	3	1	21,075	S

Es wird also geprüft, ob ein Zusammenhang zwischen einer abhängigen binären Variable (hier SURVIVED) und mehreren unabhängigen Variablen (die unten aufgeführten sieben Regressoren) besteht. Als Regressoren fungieren dann nach Datenbereinigung:

- $x_1 = \text{PCLASS}$
- $x_2 = \text{SEX}$
- $x_3 = \text{AGE}$
- $x_4 = \text{SIBSP}$
- $x_5 = \text{PARCH}$
- $x_6 = \text{FARE}$
- $x_7 = \text{EMBARKED}$

Die dichotome (binäre) Variable SURVIVED = Y nimmt nur die Werte 0 und 1 an. Man möchte nun beim logistischen Regressionsmodell mithilfe der logistischen Verteilungsfunktion den Effekt der erklärenden Variablen x_1, \dots, x_k (hier $k = 7$ Regressoren) auf die Wahrscheinlichkeit für $Y_i = 1$ bzw. $Y_i = 0$ bestimmen. Hierbei wird der Erwartungswert von Y auf das Intervall $[0, 1]$ beschränkt, durch das Aufsetzen einer Response-Funktion $F(x)$, $x \in \mathbb{R}$ wird hier die Linearkombination $\eta = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_7 x_7$ der Regressoren ebenfalls auf dieses Intervall begrenzt. Die logistische Regression benutzt als Responsefunktion die logistische Funktion

$$\Pi_i = P(Y_i = 1 | x_1, \dots, x_k) = F(\eta) = \frac{e^\eta}{1 + e^\eta} = \frac{1}{1 + e^{-\eta}} = \frac{1}{1 + e^{-(\alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_7 x_7)}}.$$

Im Vergleich zur linearen Regression wird also hier die Wahrscheinlichkeit für $Y = 1$ nicht direkt aus den erklärenden Variablen modelliert, sondern indirekt über das sogenannte Logit-Modell. Das Logit ist die logarithmierte Chance für das Auftreten von $Y = 1$ und ist wie folgt definiert:

$$\eta = \text{Logit}(Y_i = 1|x_1, \dots, x_k) = \ln \left(\frac{\Pi_i}{1 - \Pi_i} \right) = \alpha + \beta_1 x_1 + \dots + \beta_k x_k$$

Zusätzlich wird die Chance $\frac{\Pi_i}{1 - \Pi_i} = \frac{P(Y_i=1)}{P(Y_i=0)}$ auch als Odds bezeichnet und wird unten an der entsprechenden Stelle weiter ausgeführt.

Nun wird der Datensatz - was eigentlich Titanic Training ist - manuell in Trainings- und Testmengen aufgeteilt, d.h. die Beobachtungen Nr. 1-800 werden als Trainingsmenge verwendet und die Beobachtungen nur. 801-889 als Testmenge. Das Verhältnis von Nicht-Überlebt und Überlebt für beide Datensätze lässt sich mithilfe des `table()` Befehls ausgeben.

```
table(train$Survived)

##
##   0   1
## 493 307

table(test$Survived)

##
##   0   1
## 56 33
```

Somit kann nun die Trainingsphase beginnen. Es wird hierbei ein den Trainingsdaten entsprechendes Modell angewendet. Da eine Klassifikation stattfinden soll, eignen sich als Modell zum Beispiel Entscheidungsbäume, Neuronale Netze, Random Forest oder Support Vector Machines. Hierzu findet man online viele R Codes, oder auch Python Codes, in denen Titanic Daten mit verschiedenen Modell durchgetestet wurde. In dieser Ausarbeitung wird nur die binäre logistische Regression als Modell angewendet. Das Ziel ist es, wie oben schon erläutert, ein Modell mit möglichst guter Vorhersagekraft zu finden.

```
```{r}
Model fitting --- logistische Regression
model <- glm(Survived ~ ., family=binomial(link='logit'), data=train)
summary(model)
```

Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
     data = train)

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-2.6064 -0.5954 -0.4254  0.6220  2.4165 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 5.137627  0.594998  8.635 < 2e-16 ***
Pclass      -1.087156  0.151168 -7.192 6.40e-13 ***
Sexmale     -2.756819  0.212026 -13.002 < 2e-16 ***
Age        -0.037267  0.008195 -4.547 5.43e-06 ***
SibSp       -0.292920  0.114642 -2.555  0.0106 *  
Parch      -0.116576  0.128127 -0.910  0.3629  
Fare        0.001528  0.002353  0.649  0.5160  
EmbarkedQ  -0.002656  0.400882 -0.007  0.9947  
EmbarkedS  -0.318786  0.252960 -1.260  0.2076  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1065.39  on 799  degrees of freedom
Residual deviance: 709.39  on 791  degrees of freedom
AIC: 727.39

Number of Fisher Scoring iterations: 5
```

Die Anpassung einer logistischen Regression geschieht mit der `glm()` Funktion, mit der allgemein GLM-Modelle spezifiziert werden können. Wichtig dabei ist, dass als Familie binomial angegeben wird, da es sich um eine dichotome Abhängige Variable (`TARGET = Survived`) handelt. Der Standardwert für den Link bei binomial ist `logit`, die Angabe kann daher auch vernachlässigt werden. Zusätzlich sollte noch erwähnt werden, dass R `SEXMALE` als Dummy Variable intern aufnimmt.

Zusätzlich wurde noch mit Cross Validation probiert ein besseres Modell anzupassen. Es wurden mehrere verschiedene Parameter `k` ausprobiert und es ergab jeweils dieselben Resultate wie ohne CV. Unten ist ein Ausschnitt von logistische Regression mit Leave-One-Out CV.

```
```{r}
logistische Regression mit Cross-Validation
library(caret)

definiere training control
train_control<- trainControl(method="cv", number=800)

trainiere das Modell
modelloo<- train(as.factor(Survived) ~., data=train, trControl=train_control, method="glm",
family=binomial(link='logit'))

summary(modelloo)
```

There were missing values in resampled performance measures.
Call:
NULL

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.6064 -0.5954 -0.4254  0.6220  2.4165 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 5.137627  0.594998  8.635 < 2e-16 ***
Pclass      -1.087156  0.151168 -7.192 6.40e-13 ***
Sexmale     -2.756819  0.212026 -13.002 < 2e-16 ***
Age        -0.037267  0.008195 -4.547 5.43e-06 ***
SibSp       -0.292920  0.114642 -2.555  0.0106 *  
Parch       -0.116576  0.128127 -0.910  0.3629  
Fare        0.001528  0.002353  0.649  0.5160  
EmbarkedQ  -0.002656  0.400882 -0.007  0.9947  
EmbarkedS  -0.318786  0.252960 -1.260  0.2076  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1065.39 on 799 degrees of freedom
Residual deviance: 709.39 on 791 degrees of freedom
```

Die Wald-Statistik testet die Nullhypothese, dass der jeweilige Regressionskoeffizient β in der Grundgesamtheit 0 ist. Somit wird nun überprüft, ob die unabhängigen Variablen einen Einfluss haben oder nicht. Falls die p-Werte des Wald-Tests kleiner als die konventionelle Signifikanzgrenze von 5% sind, wird es als signifikant eingeschätzt und die Regressoren leisten einen Beitrag zur Erklärung der abhängigen Variablen. Die Hypothesen lauten:

$$\begin{aligned} H_0 &: \text{Der Regressionskoeffizient } \beta_i \text{ ist Null.} \\ H_1 &: \text{Der Regressionskoeffizient } \beta_i \text{ ist ungleich Null.} \end{aligned}$$

Dies kann man auch wie folgt verstehen:

$$\begin{aligned} H_0 &: \beta_1 = \beta_2 = \dots = \beta_k = 0 \\ H_1 &: \text{Mindestens ein } \beta_i \neq 0, 1 \leq i \leq k \end{aligned}$$

Die Wald-Statistik wird gegen die Chi-Quadrat-Verteilung getestet und Die Ergebnisse des Wald-Tests und deren Signifikanz können dann unten in der Abbildung entnommen werden. In R werden auch mit Hilfe von Sternchen die Regressoren gekennzeichnet, die einen signifikanten Einfluss auf die abhängigen Variablen, hier SURVIVED, haben.

Es ist oben aus dem Koeffizienten zu erkennen, dass nicht alle Regressoren einen signifikanten Einfluss auf die Erklärung der abhängigen Variable SURVIVED ausüben. Dies wird später noch einmal aufgegriffen und es wird ein neues logistisches Modell erstellt. Zum Beispiel sieht man, dass Sex den niedrigsten p-Wert hat, was auf eine starke Assoziation des Geschlechts des Passagiers mit der Wahrscheinlichkeit des Überlebens schließen lässt. Der negative Koeffizient legt hier unter anderem nahe, dass bei allen anderen Variablen der männliche Passagier weniger wahrscheinlich überlebt hat.

Zusätzlich werden nun die Schätzer des Quotenverhältnis, auch Odds-Ratio (kurz OR) genannt, betrachtet. Dabei drücken die Chance, das Eintreten eines Ereignisses, im Verhältnis zu dem Nicht-Eintreten des Ereignisses auf. In der Regel werden zu der Odds-Ratio-Schätzern die 95% Konfidenzintervalle angegeben. Das bedeutet, dass der gesuchte Parameter mit einer Wahrscheinlichkeit von 95% im Konfidenzintervall liegt. Überdies erlaubt der Konfidenzintervall Schlüsse bezüglich der statistischen Signifikanz zu ziehen. Beinhaltet der 95%-ige Konfidenzintervall nicht den Wert der Nullhypothese, welche bei Odds Ratio 1 wäre (also $H_0: OR = 1$), dann bedeutet das ein signifikantes Ergebnis zum Niveau $\alpha = 5\%$ für den Ausschluss eines Nulleffekts ist. Zusammenfassend kann also die statistische Signifikanz durch Konfidenzintervallen geprüft werden. Dabei müssen die komplett oberhalb oder unterhalb der 1 liegen.

```

```{r}
nur odds ratios
exp(coef(model))
```


	(Intercept)	Pclass	Sexmale	Age	SibSp	Parch	Fare
(Intercept)	170.31116493	0.33717391	0.06349342	0.96341847	0.74608157	0.88996209	1.00152965
EmbarkedQ							
EmbarkedS	0.99734737	0.72703115					



```{r}
odds ratios und 95% Konfidenzeintervall
exp(cbind("Odds ratio" = coef(model), confint.default(model, level = 0.95)))
```


	Odds ratio	2.5 %	97.5 %
(Intercept)	170.31116493	53.06157608	546.64589790
Pclass	0.33717391	0.25071463	0.45344878
Sexmale	0.06349342	0.04190364	0.09620678
Age	0.96341847	0.94806687	0.97901864
SibSp	0.74608157	0.59594081	0.93404865
Parch	0.88996209	0.69232428	1.14401956
Fare	1.00152965	0.99692057	1.00616004
EmbarkedQ	0.99734737	0.45458465	2.18815521
EmbarkedS	0.72703115	0.44282640	1.19363772


```

Dies wird später noch einmal aufgegriffen und es wird ein neues logistisches Modell erstellt. Es lässt sich also hier dank der Konfidenzintervalle ablesen, dass nur die Regressoren PCLASS, SEX, AGE und SIBSP statistisch signifikant sind.

Die Differenz zwischen der Nullabweichung und der Restabweichung zeigt, wie das Modell gegen das Nullmodell, das heißt ein Modell mit nur Intercept, sich schlägt. Hier gilt je größer diese Lücke, desto besser ist es. Somit ist es deutlich, dass durch das Hinzufügen von Pclass, Sex und Age die Restabweichung deutlich reduziert wird. Die anderen Variablen scheinen das Modell weniger zu verbessern. Hier gilt wieder, dass obwohl SibSp einen niedrigen p-Wert hat kann man diesen nicht wirklich als signifikanten Einfluss sicherstellen. Ein großer p-Wert deutet darauf hin, dass das Modell ohne die Variable mehr oder weniger die gleiche Variationsbreite erklärt.

```
```{r}
Analysis of deviance
anova(model, test="Chisq")
```
Analysis of Deviance Table

Model: binomial, link: logit

Response: Survived

Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL 799 1065.39
Pclass 1 83.607 798 981.79 < 2.2e-16 ***
Sex 1 240.014 797 741.77 < 2.2e-16 ***
Age 1 17.495 796 724.28 2.881e-05 ***
SibSp 1 10.842 795 713.43 0.000992 ***
Parch 1 0.863 794 712.57 0.352873
Fare 1 0.994 793 711.58 0.318717
Embarked 2 2.187 791 709.39 0.334990
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Es existieren Maße für die Güte der Modellpassung, als pseudo- R^2 -Koeffizienten, die an den R^2 in der linearen Regression angelehnt sind. Es gibt mehrere Varianten wie nach McFadden, Cox & Snell oder Nagelkerke. Hier wird die Anpassungsgüte nach McFadden ausgewertet.

Für McFadden R^2 gilt als Daumenregel:

Bereits $0,2 < R^2_{\text{McFadden}} < 0,4$ stellt eine besonders gute Anpassung des Modells dar.

```
```{r}
McFadden R^2
#install.packages("pscl")
library(pscl)
pR2(model)
```
```
 llh llhNull G2 McFadden r2ML r2CU
-354.6950111 -532.6961008 356.0021794 0.3341513 0.3591775 0.4880244
```

Es können nun mit Hilfe von `predict()` Befehl das Modell auf die Testdaten (Beobachtungen von 801 bis 889) anwenden. Anbei ist ein Ausschnitt von den p-Wahrscheinlichkeiten ob ein Passagier überleben wird oder nicht.

801	802	803	804	805	806	807	808
0.753963539	0.555339553	0.268966522	0.100199823	0.087635519	0.382591374	0.710629291	0.175686502
809	810	811	812	813	814	815	816
0.908053970	0.103739874	0.068118069	0.197720714	0.494184371	0.089171014	0.467061844	0.670916792
817	818	819	820	821	822	823	824
0.213873799	0.057753258	0.066521634	0.821595593	0.100432614	0.391432165	0.611505040	0.075747618
825	826	827	828	829	830	831	832
0.121371738	0.097977308	0.498158813	0.121502197	0.740067460	0.371903262	0.121700886	0.114599892
833	834	835	836	837	838	839	840
0.134986038	0.910083153	0.122515421	0.091625068	0.090655969	0.557799989	0.126453986	0.333472642
841	842	843	844	845	846	847	848
0.951654603	0.103720965	0.139463537	0.059909635	0.008355321	0.102205818	0.227599634	0.941965726
849	850	851	852	853	854	855	856
0.062646787	0.018977299	0.760342331	0.956020337	0.679523342	0.686600366	0.869568623	0.292103154
857	858	859	860	861	862	863	864
0.659585257	0.121700886	0.035857454	0.236809924	0.878945394	0.117155477	0.271536987	0.750104004
865	866	867	868	869	870	871	872
0.843612428	0.474212324	0.091809697	0.149186396	0.103741054	0.839019072	0.438484070	0.050342131
873	874	875	876	877	878	879	880
0.840719304	0.790541057	0.126778652	0.130623230	0.091605453	0.878042770	0.836990876	0.081870226
881	882	883	884	885	886	887	888
0.679954623	0.242370054	0.107133934	0.470461277	0.249994665	0.955615201	0.490084348	0.591592661
889							
0.112642486							

Diese können auch direkt als SURVIVED=1 oder SURVIVED=0 ausgegeben werden. Anbei ist auch noch die Tabelle mit den echten Klassen ob jemand überlebt hat oder nicht.

Eine Genauigkeit von 0,8427 am Testset zwar ein ziemlich gutes Ergebnis, jedoch sollte noch beachtet werden, dass dieses Ergebnis von der manuellen Aufteilung der Daten abhängt, die am Anfang gemacht wurde. Das heißt für ein genaueres Ergebnis, sollte man die Trainings- und Testdaten anders aufteilen.

```
Confusion Matrix and Statistics

 reference
data 0 1
 0 51 9
 1 5 24

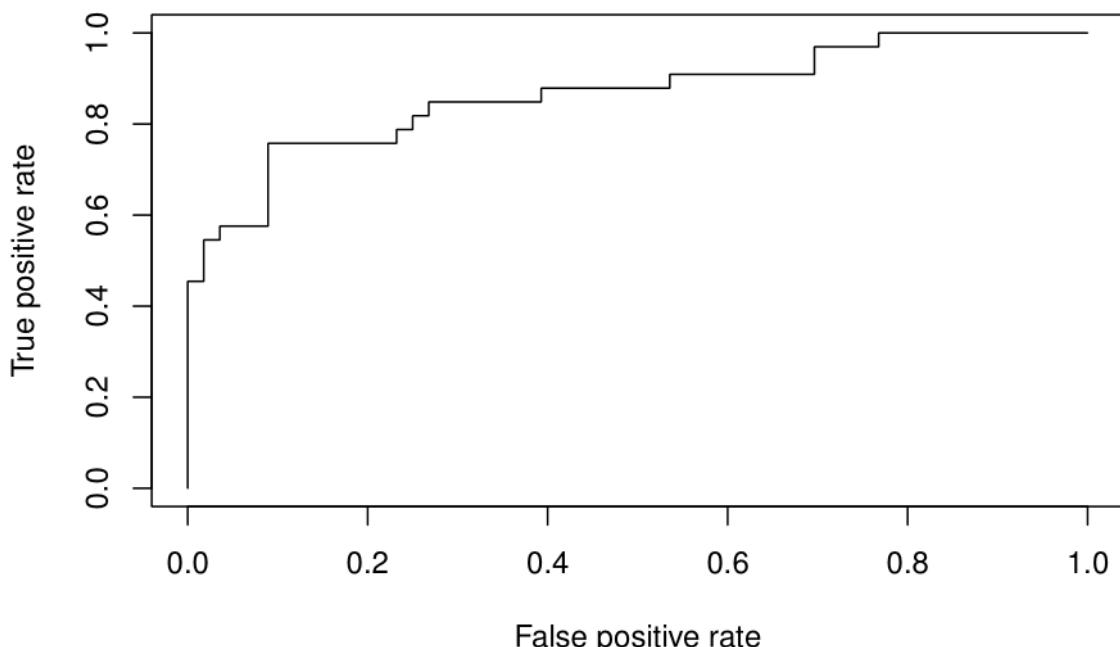
 Accuracy : 0.8427
 95% CI : (0.7502, 0.9112)
No Information Rate : 0.6292
P-Value [Acc > NIR] : 8.248e-06

 Kappa : 0.6543
McNemar's Test P-Value : 0.4227

Sensitivity : 0.9107
Specificity : 0.7273
Pos Pred Value : 0.8500
Neg Pred Value : 0.8276
Prevalence : 0.6292
Detection Rate : 0.5730
Detection Prevalence : 0.6742
Balanced Accuracy : 0.8190

'Positive' Class : 0
```

Die, dem Modell zugehörige ROC-Kurve hat einen sehr guten AUC-Wert.



```
```{r}
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```
[1] 0.8647186
```

Aufgrund der p-Werte kann zusätzlich behauptet werden, dass nur die Regressoren PCLASS, SEX und AGE einen signifikanten Einfluss auf die Abhängige Variable haben und ein neues Model kann aufgebaut werden. Auch wenn die Variable SIBSP noch knapp das Signifikanzniveau von 5% erreicht, wird diese Variable nicht in das neue Model 2 aufgenommen, da es nicht viel neue Informationen dazu beiträgt. Die restlichen Regressoren haben keinen signifikanten Einfluss und können aus der obigen Modellgleichung entfernt werden. Somit folgt

$$F(\eta) = \frac{1}{1 + e^{-\eta}} = \frac{1}{1 + e^{-(\alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)}}$$

mit  $x_1 = \text{PCLASS}$ ,  $x_2 = \text{SEX}$  und  $x_3 = \text{AGE}$ .

```
#bei model2 nur die signifkanten Einflussfaktoren mit in Log. Reg. genommen!
model2 <- glm(Survived ~ Pclass + Sex + Age, family=binomial(link='logit'), data=train)
...
Call:
glm(formula = Survived ~ Pclass + Sex + Age, family = binomial(link = "logit"),
 data = train)

Deviance Residuals:
 Min 1Q Median 3Q Max
-2.6320 -0.6570 -0.4239 0.6420 2.4093

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
(Intercept) 4.633935 0.472667 9.804 < 2e-16 ***
Pclass -1.139026 0.125153 -9.101 < 2e-16 ***
Sexmale -2.646164 0.197657 -13.388 < 2e-16 ***
Age -0.031477 0.007724 -4.075 4.6e-05 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Dispersion parameter for binomial family taken to be 1

Null deviance: 1065.39 on 799 degrees of freedom
Residual deviance: 724.28 on 796 degrees of freedom
AIC: 732.28

Number of Fisher Scoring iterations: 4
```

Anhand der dazugehörigen Confusion Matrix und der ACCURACY (Korrektklassifikationsrate) erkennt man, dass zwar nicht mehr so gut die Target Variable vorhergesagt wird, nichtsdestotrotz dieses Model beim Wunsch nach einer Dimensionsreduktion verwendet werden kann.

```
Confusion Matrix and Statistics

 reference
data 0 1
 0 48 10
 1 8 23

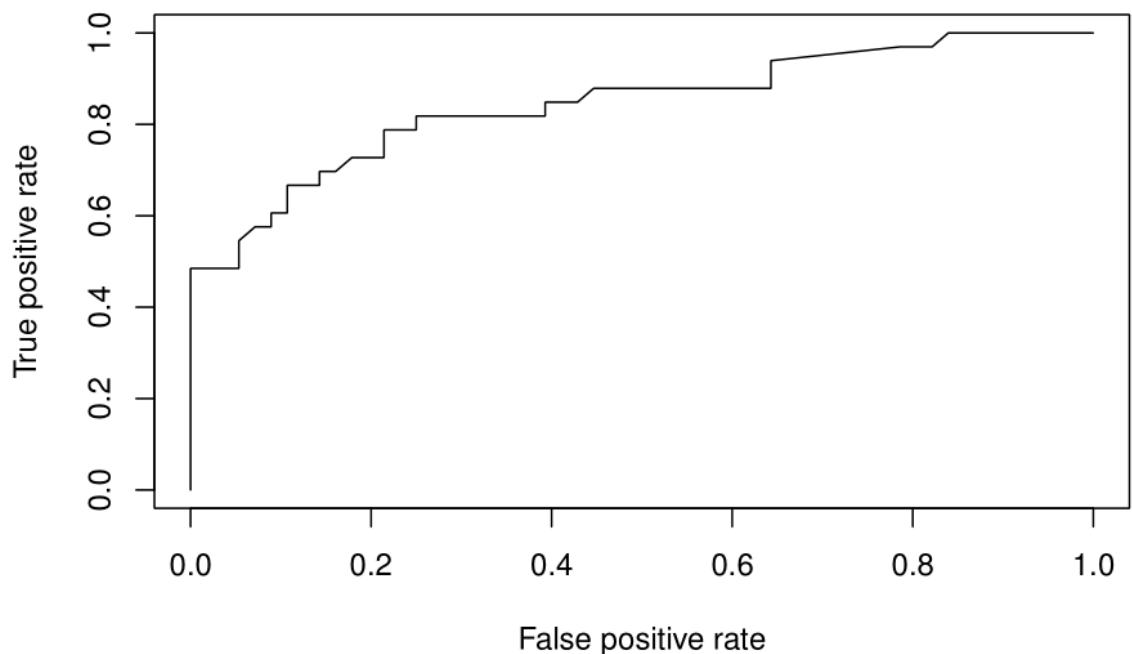
Accuracy : 0.7978
95% CI : (0.6993, 0.8755)
No Information Rate : 0.6292
P-Value [Acc > NIR] : 0.0004642

Kappa : 0.5611
McNemar's Test P-Value : 0.8136637

Sensitivity : 0.8571
Specificity : 0.6970
Pos Pred Value : 0.8276
Neg Pred Value : 0.7419
Prevalence : 0.6292
Detection Rate : 0.5393
Detection Prevalence : 0.6517
Balanced Accuracy : 0.7771

'Positive' Class : 0
```

Am Schluss kann noch die ROC-Kurve betrachtet werden. Hier hat die ROC-Kurve einen AUC Wert von 0,844697. Trotz weniger Regressoren liefert das neue logistische Regressionsmodell somit ein gutes Ergebnis.



```
auc2 <- performance(pr2, measure = "auc")
auc2 <- auc2@y.values[[1]]
auc2
[1] 0.844697
```

Nun werden die Daten nicht mehr manuell, sondern zufällig mit set.seed(12345), aufgeteilt. Es werden Modell A mit 60:40, Modell B mit 70:30 und Modell C mit 80:20 Aufteilungen vorgestellt. Zunächst einmal Modell A mit einem AUC-Wert am Ende von 0.8460718.

```

Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
 data = traina)

Deviance Residuals:
 Min 1Q Median 3Q Max
-2.7070 -0.6171 -0.4032 0.6431 2.5176

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
(Intercept) 5.307468 0.713954 7.434 1.05e-13 ***
Pclass -1.164082 0.184945 -6.294 3.09e-10 ***
Sexmale -2.685712 0.260642 -10.304 < 2e-16 ***
Age -0.043100 0.010101 -4.267 1.98e-05 ***
SibSp -0.195717 0.131931 -1.483 0.138
Parch -0.133584 0.146807 -0.910 0.363
Fare 0.002509 0.002807 0.894 0.372
EmbarkedQ -0.203258 0.484099 -0.420 0.675
EmbarkedS -0.336471 0.297581 -1.131 0.258

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 702.22 on 532 degrees of freedom
Residual deviance: 464.50 on 524 degrees of freedom
AIC: 482.5

Number of Fisher Scoring iterations: 5

Confusion Matrix and Statistics

 reference
data 0 1
 0 185 46
 1 28 97

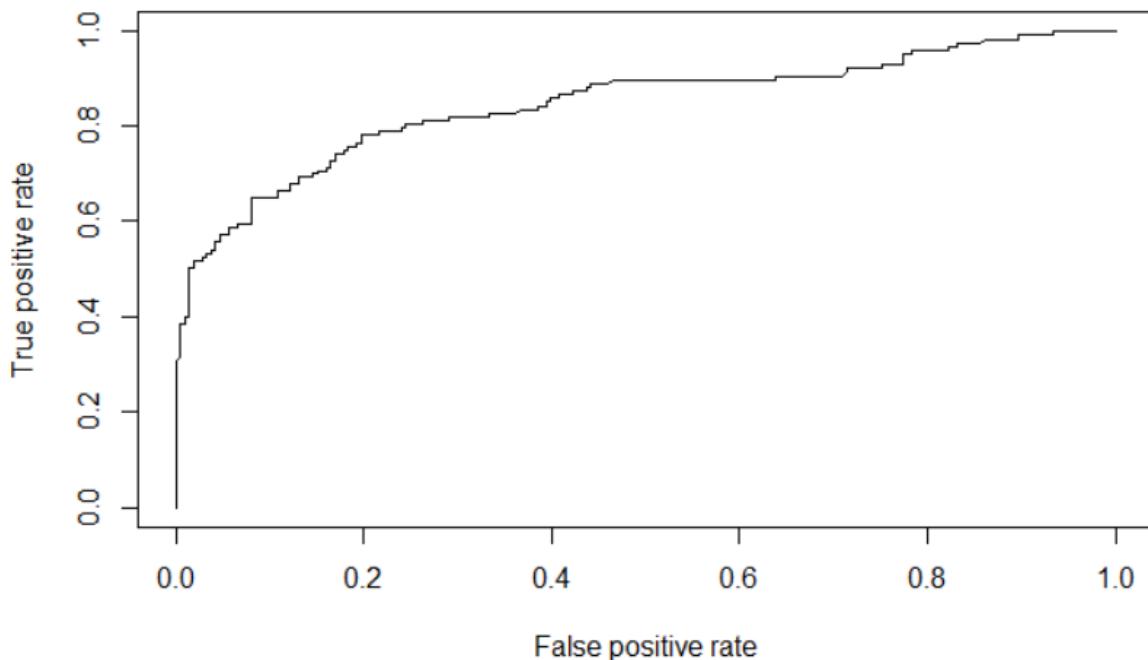
Accuracy : 0.7921
95% CI : (0.7462, 0.8331)
No Information Rate : 0.5983
P-Value [Acc > NIR] : 5.399e-15

Kappa : 0.5584
McNemar's Test P-Value : 0.04813

Sensitivity : 0.8685
Specificity : 0.6783
Pos Pred Value : 0.8009
Neg Pred Value : 0.7760
Prevalence : 0.5983
Detection Rate : 0.5197
Detection Prevalence : 0.6489
Balanced Accuracy : 0.7734

'Positive' Class : 0

```



Dann folgt Modell B mit einem AUC Wert von 0.8650841.

```

Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
 data = trainb)

Deviance Residuals:
 Min 1Q Median 3Q Max
-2.6233 -0.6260 -0.4320 0.6927 2.4253

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
(Intercept) 5.043751 0.649223 7.769 7.92e-15 ***
Pclass -1.092431 0.167947 -6.505 7.79e-11 ***
Sexmale -2.623022 0.235964 -11.116 < 2e-16 ***
Age -0.038604 0.009129 -4.229 2.35e-05 ***
SibSp -0.275967 0.130381 -2.117 0.0343 *
Parch -0.106149 0.136387 -0.778 0.4364
Fare 0.002198 0.002681 0.820 0.4124
EmbarkedQ -0.083677 0.435170 -0.192 0.8475
EmbarkedS -0.310637 0.276428 -1.124 0.2611

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 826.72 on 621 degrees of freedom
Residual deviance: 561.73 on 613 degrees of freedom
AIC: 579.73

Number of Fisher Scoring iterations: 5

```

### Confusion Matrix and Statistics

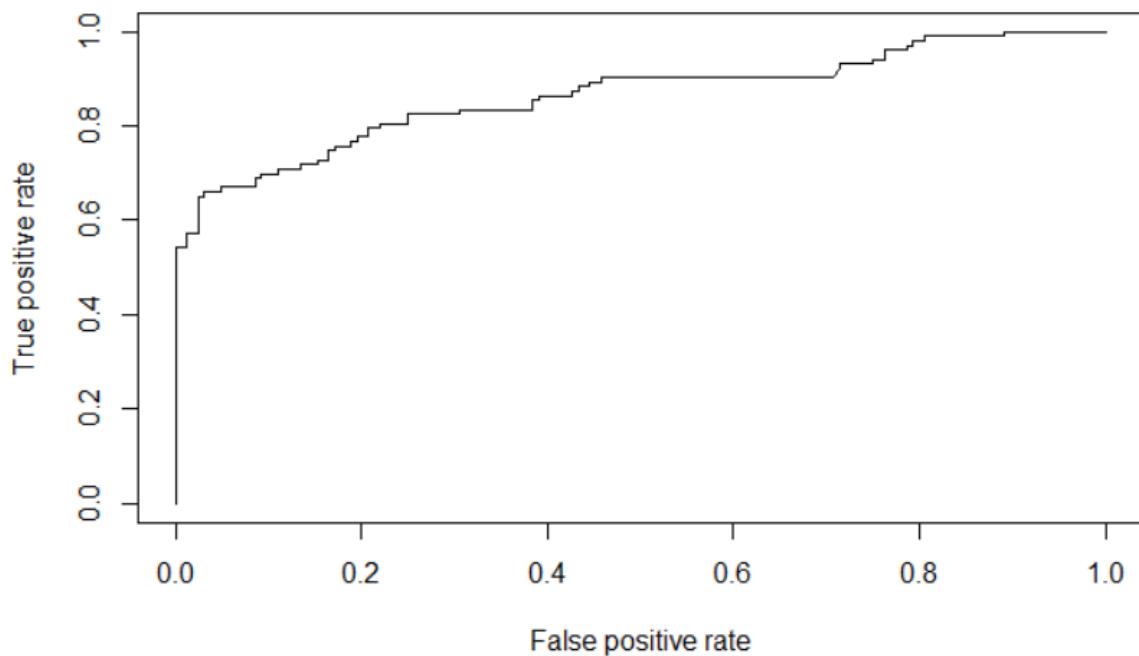
| reference | 0   | 1   |
|-----------|-----|-----|
| data      | 0   | 149 |
| 0         | 149 | 32  |
| 1         | 15  | 71  |

Accuracy : 0.824  
95% CI : (0.7729, 0.8677)  
No Information Rate : 0.6142  
P-Value [Acc > NIR] : 8.517e-14

Kappa : 0.6168  
McNemar's Test P-Value : 0.0196

Sensitivity : 0.9085  
Specificity : 0.6893  
Pos Pred Value : 0.8232  
Neg Pred Value : 0.8256  
Prevalence : 0.6142  
Detection Rate : 0.5581  
Detection Prevalence : 0.6779  
Balanced Accuracy : 0.7989

'Positive' Class : 0



Und zuletzt folgen die Resultate von Modell C mit 0.8501819 als AUC Wert.

```
Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
 data = trainc)

Deviance Residuals:
 Min 1Q Median 3Q Max
-2.6844 -0.5859 -0.4271 0.6282 2.4529

Coefficients:
 Estimate Std. Error z value Pr(>|z|)
(Intercept) 5.124317 0.630578 8.126 4.42e-16 ***
Pclass -1.078624 0.160038 -6.740 1.59e-11 ***
Sexmale -2.747935 0.225889 -12.165 < 2e-16 ***
Age -0.041516 0.008892 -4.669 3.03e-06 ***
SibSp -0.285191 0.125258 -2.277 0.0228 *
Parch -0.135109 0.129054 -1.047 0.2951
Fare 0.002774 0.002672 1.038 0.2993
EmbarkedQ -0.072494 0.424569 -0.171 0.8644
EmbarkedS -0.252285 0.264601 -0.953 0.3404

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 940.10 on 710 degrees of freedom
Residual deviance: 622.21 on 702 degrees of freedom
AIC: 640.21

Number of Fisher Scoring iterations: 5
```

#### Confusion Matrix and Statistics

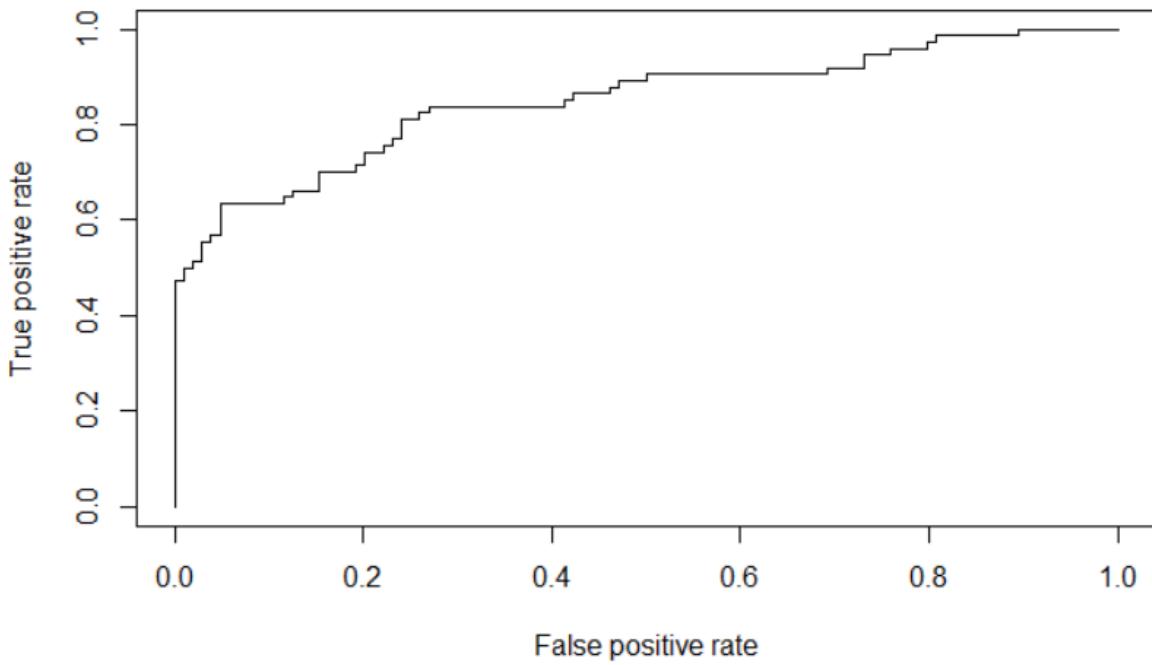
```
reference
data 0 1
 0 88 24
 1 16 50

Accuracy : 0.7753
95% CI : (0.7068, 0.8343)
No Information Rate : 0.5843
P-Value [Acc > NIR] : 6.42e-08

Kappa : 0.5301
McNemar's Test P-Value : 0.2684

Sensitivity : 0.8462
Specificity : 0.6757
Pos Pred Value : 0.7857
Neg Pred Value : 0.7576
Prevalence : 0.5843
Detection Rate : 0.4944
Detection Prevalence : 0.6292
Balanced Accuracy : 0.7609

'Positive' Class : 0
```



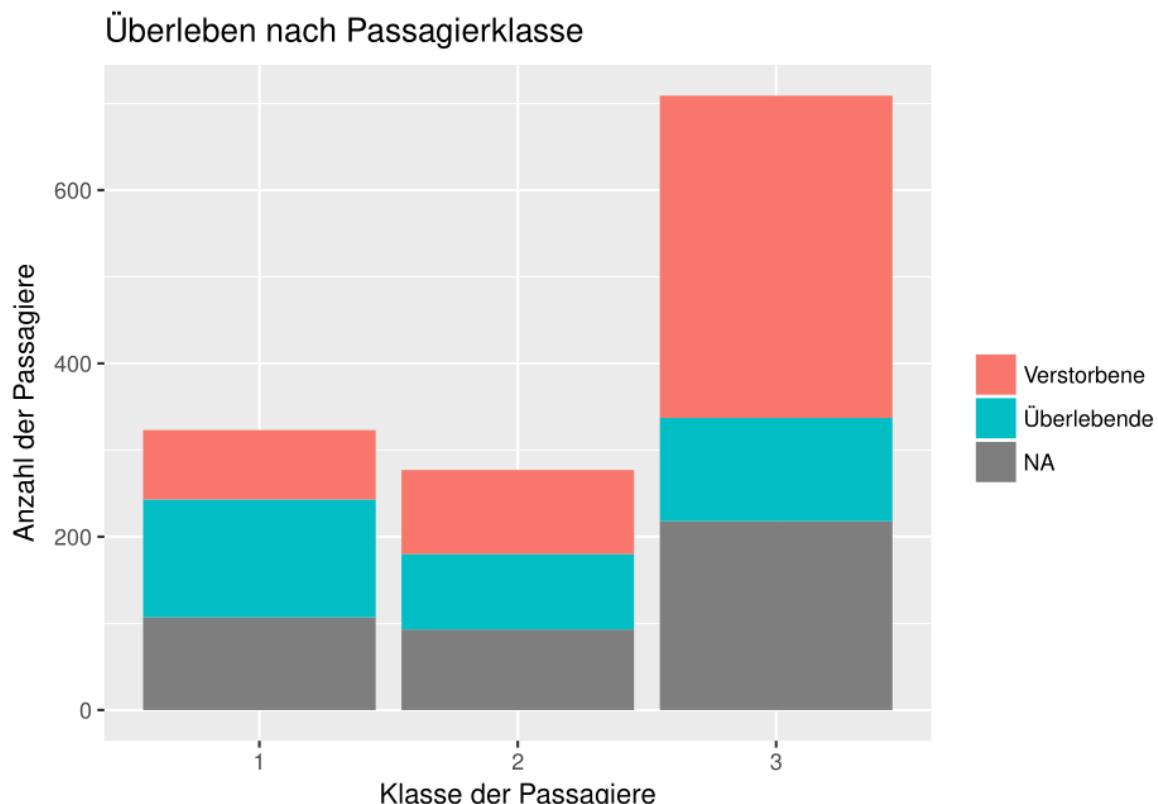
Zusätzlich sollte noch einmal erwähnt werden, dass die Titanic eigentlich Daten von insgesamt 1309 Beobachtungen hat.

```
```{r}
library('dplyr') # data manipulation
library(ggplot2)
#####
trainspater <- read.csv('C:/BÜSRA/Uni/Master/B Fächer/Data Mining
2/HELM_2018/KAGGLE/KAGGLE_TITANIC/train.csv', stringsAsFactors = F)
testspater <- read.csv('C:/BÜSRA/Uni/Master/B Fächer/Data Mining
2/HELM_2018/KAGGLE/KAGGLE_TITANIC/test.csv', stringsAsFactors = F)

# binde training & test data für später eigene Datapartition
full <- bind_rows(trainspater, testspater)
```
```
# check data
str(full)
```
```
'data.frame': 1309 obs. of 12 variables:
 $ PassengerId: int 1 2 3 4 5 6 7 8 9 10 ...
 $ Survived    : int 0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass      : int 3 1 3 1 3 3 1 3 3 2 ...
 $ Name        : chr "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Brig
"Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
 $ Sex         : chr "male" "female" "female" "female" ...
 $ Age         : num 22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp       : int 1 1 0 1 0 0 0 3 0 1 ...
 $ Parch       : int 0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket      : chr "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
 $ Fare        : num 7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin       : chr "" "C85" "" "C123" ...
 $ Embarked    : chr "S" "C" "S" "S" ...
```

```

Hierfür wurde ein Trainings- und ein Test-Datensatz als CSV Datei zur Verfügung gestellt. Das heißt eigentlich sollten die Modelle in der Trainingsdatei antrainiert und später in Testdatei auf deren Richtigkeit getestet werden. Diese Vorgehen und viele Weitere können online sehr schnell nachgelesen werden. Zusätzlich könnte man das logistische Regressionsmodell noch durch ein anderes Modells ersetzen oder auch mit einem anderen Modell verglichen und mit Hilfe der Anpassungsgüte argumentieren welche der Modelle am besten die Vorhersage, überleben oder nicht überleben, trifft. Zudem können auch von vornherein auf spezielle Annahmen geprüft werden. Zum Beispiel mit Hilfe des untenstehenden Plots könnte man auf die Idee kommen, dass die Passagiere mit wenig Geld, somit auch Passagiere in der dritten Klasse, eher das Unglück nicht überlebt haben.

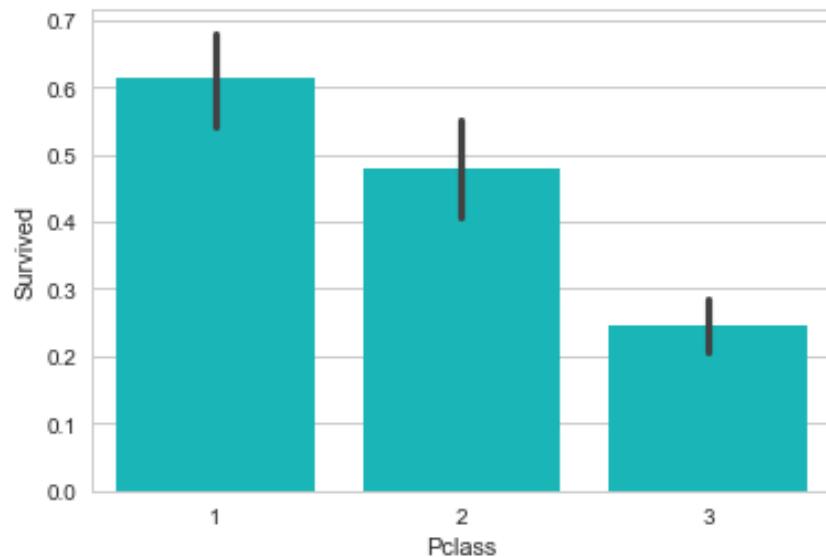


Die schon in R aufbereitete Trainings- und Testdatensätze werden nun in Python eingelesen.

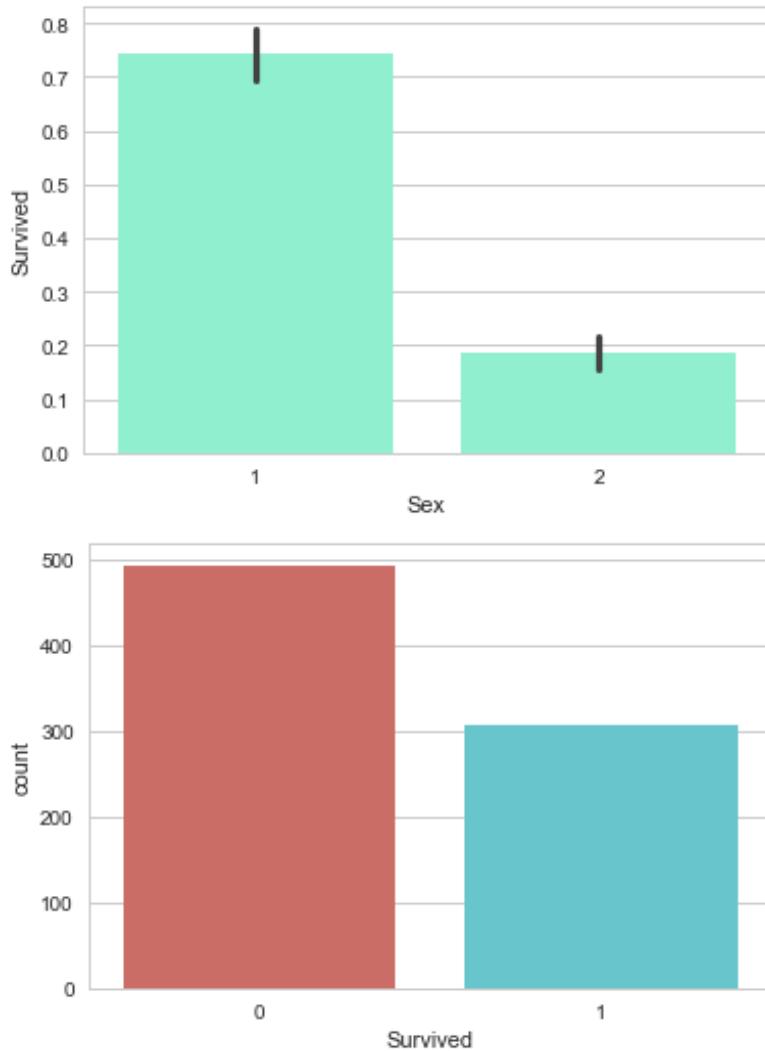
```
train.head()
```

|   | Survived | Pclass | Sex | Age  | SibSp | Parch | Fare    | Embarked |
|---|----------|--------|-----|------|-------|-------|---------|----------|
| 0 | 0        | 3      | 2   | 22.0 | 1     | 0     | 7.2500  | 3        |
| 1 | 1        | 1      | 1   | 38.0 | 1     | 0     | 71.2833 | 1        |
| 2 | 1        | 3      | 1   | 26.0 | 0     | 0     | 7.9250  | 3        |
| 3 | 1        | 1      | 1   | 35.0 | 1     | 0     | 53.1000 | 3        |
| 4 | 0        | 3      | 2   | 35.0 | 0     | 0     | 8.0500  | 3        |

Es können wieder anhand unterschiedlicher Plots neue Annahmen für Modelle gemacht werden.



Hier sei noch darauf hingewiesen, dass in Python auch eine SEXMALE Dummy Variable zwar eingefügt werden konnte, nur diese hier unerheblich war, da SEX=1=FEMALE und SEX=2=MALE intern deklariert wurde.



Hier wird nun wieder die logistische Regression angewendet und die Koeffizienten der hier in Python ähneln sehr die von R.

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression()
LogReg.fit(train_x, train_Y)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False)

Check trained model intercept
print(LogReg.intercept_)

Check trained model coefficients
print(LogReg.coef_)

[5.2995254]
[[-0.67452012 -2.27628786 -0.021383 -0.24520561 -0.0893804 0.00553232]]
```

Python Analyse ergab einen Accuracy von 0.8089 und einen AUC Wert von 0.862 - wobei bei R 0.8427 als Accuracy und 0.8647 als AUC hatte.

```
Make predictions
y_pred = LogReg.predict(test_x)
```

```
Generate table of predictions vs actual
pd.crosstab(y_pred,test_Y)
```

Survived 0 1

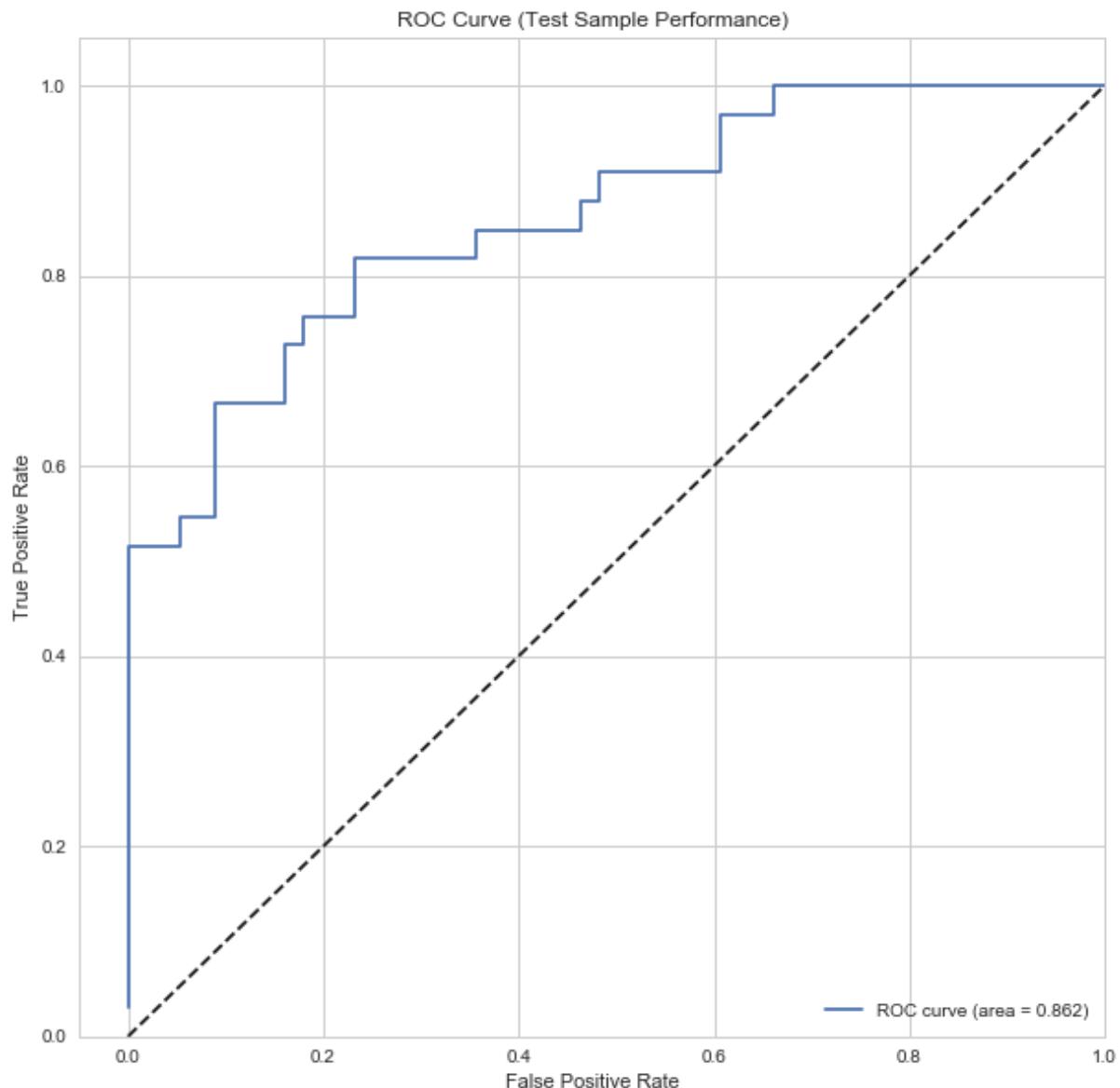
row\_0

|  | 0 | 50 | 11 |
|--|---|----|----|
|--|---|----|----|

|   |    |    |
|---|----|----|
| 0 | 50 | 11 |
| 1 | 6  | 22 |

```
LogReg.score(test_x, test_Y) ### auch durch (50+22)/89
```

0.8089887640449438



Auch hier werden die Daten, also das Trainings-csv.Datei, zufällig mit den Prozentsätzen für Modell A mit 60:40, für Modell B mit 70:30 und für Modell C mit 80:20 gesplittet. Dabei wird wieder seed(12345), hier als random\_state deklariert, verwendet.

```
data = pd.read_table("C:\\\\BÜSRA\\\\Uni\\\\Master\\\\B Fächer\\\\Data Mining 2\\\\HELM_2018\\\\KAGGLE\\\\Titanic_data.txt", sep=",")
data.columns = ['Survived','Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']

define 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked' as the features and Survived as the response
feature_cols = ['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']
X = data[feature_cols]
y = data.Survived

split the data into training and testing sets
from sklearn.cross_validation import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X, y, test_size=0.4, random_state=12345)
X2_train, X2_test, y2_train, y2_test = train_test_split(X, y, test_size=0.3, random_state=12345)
X3_train, X3_test, y3_train, y3_test = train_test_split(X, y, test_size=0.2, random_state=12345)
```

```
LogReg1 = LogisticRegression()
LogReg1.fit(X1_train, y1_train)

print(LogReg1.intercept_)
print(LogReg1.coef_)

[4.94811492]
[[-0.69863862 -2.05953991 -0.02505105 -0.29327041 -0.12687918 0.00963864
 0.0497222]]
```

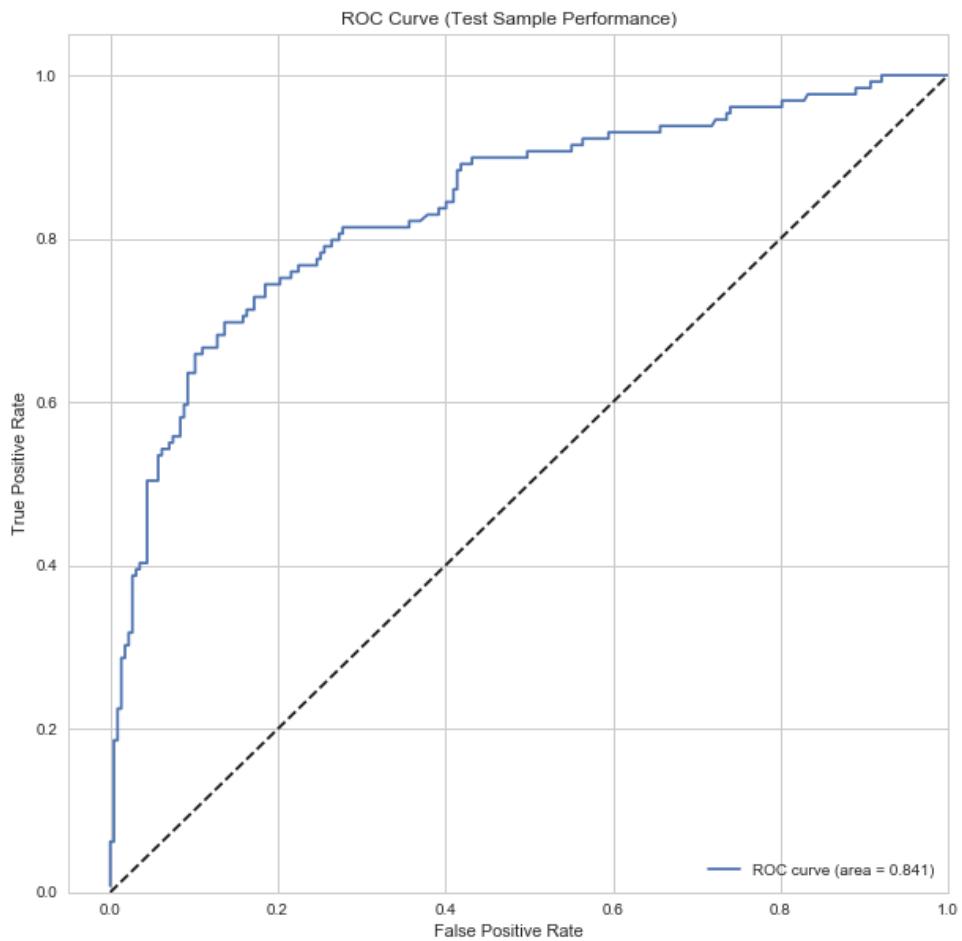
```
y_pred1 = LogReg1.predict(X1_test)

pd.crosstab(y_pred1,y1_test)
```

| Survived | 0   | 1  |
|----------|-----|----|
| row_0    |     |    |
| 0        | 201 | 43 |
| 1        | 26  | 86 |

```
LogReg1.score(X1_test, y1_test)

0.8061797752808989
```



Somit gilt für Modell A in Python 0.8062 Accuracy und 0.841 AUC Wert, wobei Modell A in R einen Accuracy von 0.7921 und einen AUC Wert von 0.845 hatte.

```
LogReg2 = LogisticRegression()
LogReg2.fit(X2_train, y2_train)

print(LogReg2.intercept_)
print(LogReg2.coef_)

[5.15783485]
[[-7.21679436e-01 -2.13894790e+00 -2.28319475e-02 -2.60991131e-01
 -6.53604832e-02 7.32351504e-03 -1.30232295e-03]]
```

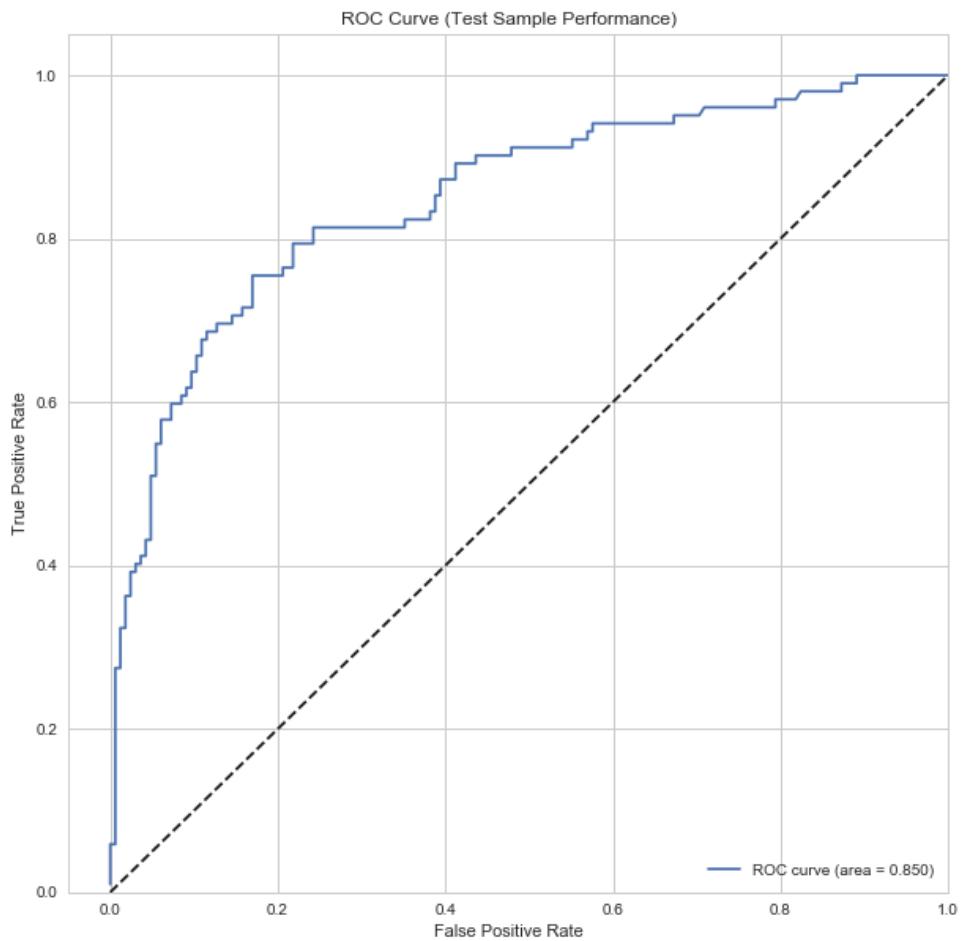
```
y_pred2 = LogReg2.predict(X2_test)

pd.crosstab(y_pred2,y2_test)
```

|       | Survived | 0  | 1 |
|-------|----------|----|---|
| row_0 |          |    |   |
| 0     | 147      | 34 |   |
| 1     | 18       | 68 |   |

```
LogReg2.score(X2_test, y2_test)
```

0.8052434456928839



Modell B hat nun in Python für Accuracy 0.805, in R war es 0.824 und mit einem AUC Wert von 0.850 in Python und 0.865 in R.

```
LogReg3 = LogisticRegression()
LogReg3.fit(X3_train, y3_train)

print(LogReg3.intercept_)
print(LogReg3.coef_)

[5.31975409]
[[-0.67977128 -2.18891008 -0.02467515 -0.28382837 -0.04399703 0.00842633
 -0.05146575]]
```

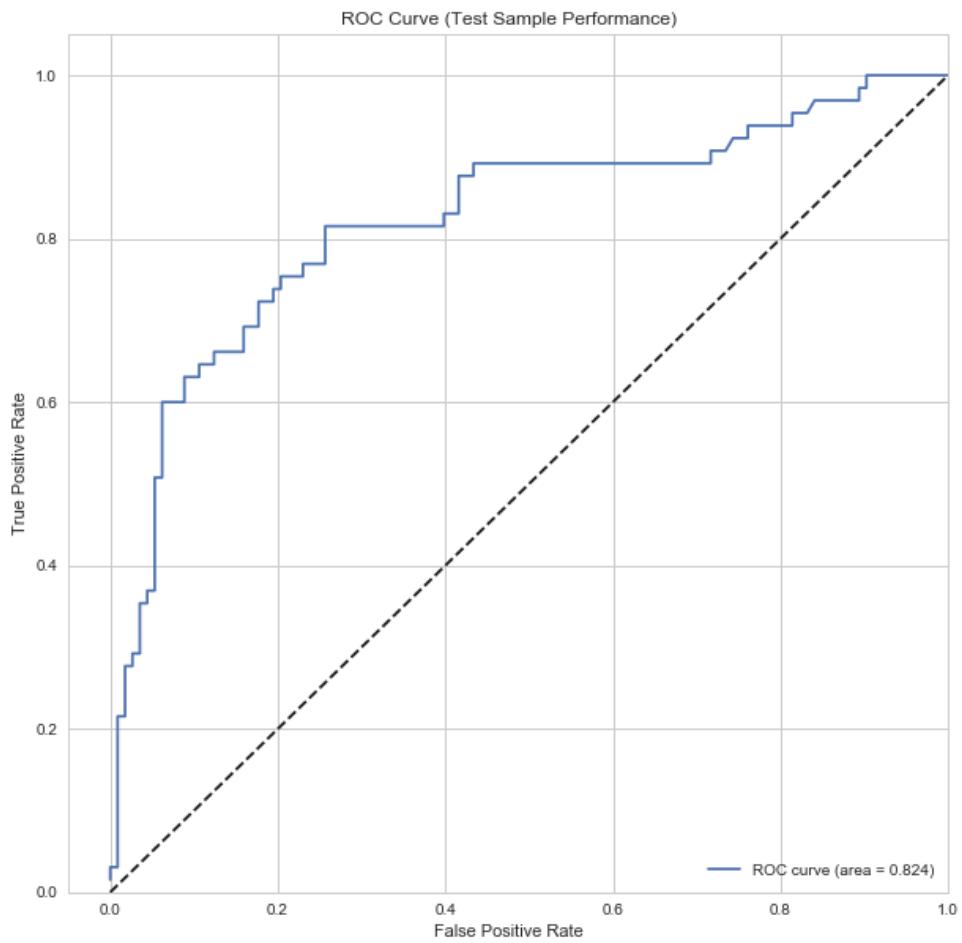
```
y_pred3 = LogReg3.predict(X3_test)

pd.crosstab(y_pred3,y3_test)
```

|       | Survived | 0  | 1  |
|-------|----------|----|----|
| row_0 | 0        | 99 | 23 |
|       | 1        | 14 | 42 |

```
LogReg3.score(X3_test, y3_test)
```

```
0.7921348314606742
```



Beim letzten Modell hat Python die Werte 0.7921 für Accuracy und 0.824 für AUC ausgegeben, wobei es in R 0.7753 und 0.850 waren.

Als letztes wurde hier zusätzlich noch SMOTE Algorithmus verwendet. Näheres dazu kann in Aufgabe 3 nachgelesen werden. Zunächst schaut man sich die Anzahl der Klassenverteilung an. Hierbei ist es deutlich, dass die Klasse 1 immer unterrepräsentiert ist.

```
import collections

counter_train=collections.Counter(y1_train)
print(counter_train)

Counter({0: 322, 1: 211})

counter_train2=collections.Counter(y2_train)
print(counter_train2)

Counter({0: 384, 1: 238})

counter_train3=collections.Counter(y3_train)
print(counter_train3)

Counter({0: 436, 1: 275})
```

Für SMOTE Methode nimmt man hier eine der Train- und Testmengen und baut wieder eine logistische Regression auf. Hier wurde die erste Trainingsmenge gewählt und SMOTE Methode angewendet.

```
from imblearn.over_sampling import SMOTE
X1_resampled, y1_resampled = SMOTE().fit_sample(X1_train, y1_train)

counter_y1=collections.Counter(y1_resampled)
print(counter_y1)

Counter({0: 322, 1: 322})

LogRegSmote = LogisticRegression()
LogRegSmote.fit(X1_resampled, y1_resampled)

print(LogRegSmote.intercept_)
print(LogRegSmote.coef_)

[5.92210561]
[[-0.77082428 -2.18225719 -0.02908607 -0.2900307 -0.16628743 0.00984382
 0.00947524]]
```

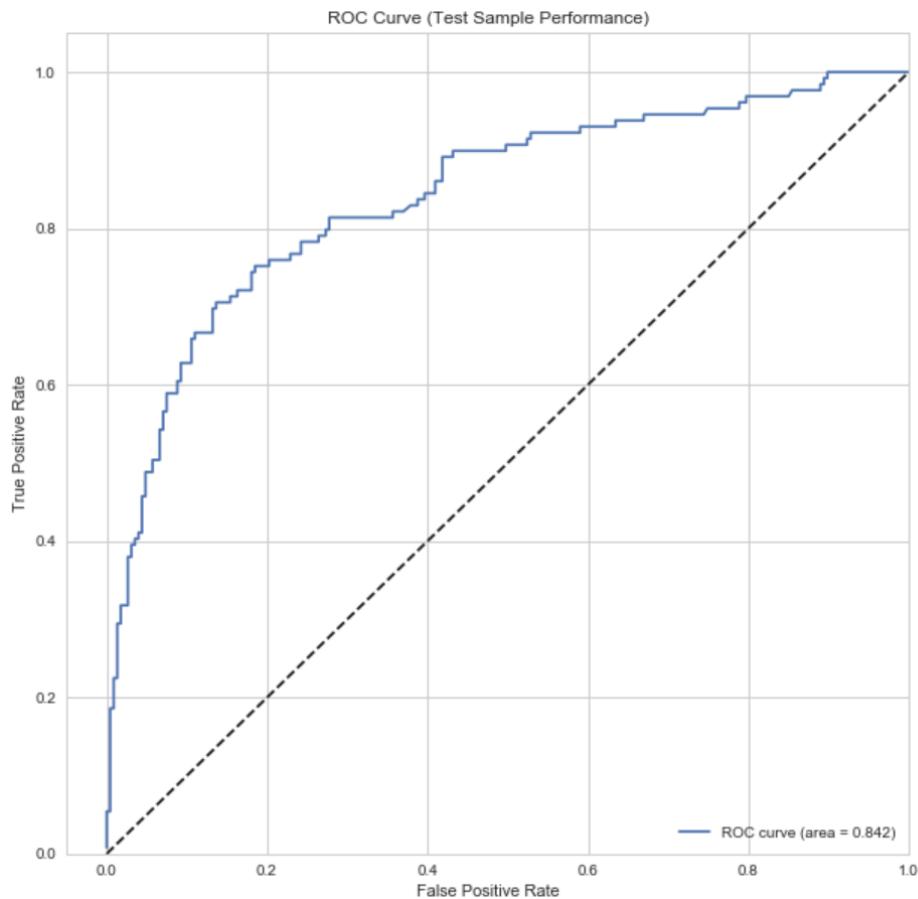
Verglichen mit der eigentlichen Modell, also LogReg1 Modell ohne SMOTE, wurde hier sogar weniger Accuracy erzielt.

```
y_pred1Smote = LogRegSmote.predict(X1_test)
pd.crosstab(y_pred1Smote,y1_test)
```

| Survived | 0   | 1  |
|----------|-----|----|
| row_0    |     |    |
| 0        | 183 | 32 |
| 1        | 44  | 97 |

```
LogRegSmote.score(X1_test, y1_test)
```

0.7865168539325843



## 4 Aufgabe 3

---

*Aufgabenstellung:*

HYPOTHYROID ist der englische Name für Erkrankungen bei Schilddrüsenunterfunktion (hypo == unter/niedrig | hyper == über/hoch).

UCI ist die Kurzform für die am Kopf des u.a. Programmes genannte Datensammlung. Github ist die Kurzform für <http://abc.github.io/def/> (abc und def variabel).

Lesen Sie die Beschreibung der Datei (Merkmale/Bedeutung auf UCI). Insbesondere ist Y == target bzw. hyper\$target die primäre Variable, die fast nur aus Missing-values besteht. Wenn Sie jemanden in der Familie mit Schilddrüsenproblemen haben, befragen Sie diese Person nach den aufgeführten Laborwerten. Y == target == 1 bedeutet Diagnose: Unterfunktion, bzw. Unterfunktion positiv.

Sie erhalten ein R-Skript **SMOTE\_Beispiel\_DST.R**

Verstehen Sie die einzelnen Schritte dieses Skriptes. Zum besseren Nachvollziehen sind vielfach Ausgaben via print(), summary() und durch das Erzeugen von \*.csv. Dateien eingefügt.

Klarerweise müssen Sie die Pfade auf eine Pfade anpassen!

Beachten Sie die gewählten Dezimaltrenner und Separatoren (für deutsches MS-Excel). Diese können leicht umgestellt werden auf internationale Form.

Kommandos vom Typ install.packages("name") sind auskommentiert um häufiges Nachladen zu reduzieren. Ggf. müssen diese von Ihnen einmalig aktiviert werden.

Hinterfragen Sie die (zu guten) Ergebnisse::

Wenn z.B. die Ärzte eine sehr starre Regel verwenden::

Unterfunktion < -> best\_TSH\_Wert über einem fixen Limit, so läge ein ähnlicher Effekt vor wie bei den DEPRESSionsdaten aus DM1!

Prüfen Sie die Abfolge der Dateien (z.B. in Excel anschauen ...).

Wenn Sie ein Problem feststellen, kann es sein, dass an einigen Stellen noch nachgebessert werden muss.

Wenn Sie sicher sind, dass alles OK ist, reduzieren Sie die Outputs, speziell die vielen summary()-Aufrufe und manche write.table(). Checken Sie gegenüber der Vorlage.

Ändern Sie die Startzufallszahl auf <1234> verändern Sie die o.a. Kennung \_01\_ auf \_02\_ und protokollieren, erläutern und bewerten Sie die neuen Ergebnisse.

**Die Bewertung natürlich auf der test == valid(at)ion)-Datei!**

Variieren Sie die Prozentsätze in SMOTE und protokollieren die Unterschiede.

Die Grundzüge des SMOTE-Algorithmus werden in der Vorlesung erklärt.

Ergänzen Sie das R-Skript um die Erstellung und Ausgabe der sogenannten **Confusion Matrix**.

Ergänzen Sie das R-Skript um einige weitere Features und Modelle, die Sie am Ende im ROC-Chart mit anzeigen lassen.

Ggf. variieren Sie die steuernden Kennzahlen (loss|metrics|cv|u.a.) und protokollieren die Auswirkungen.

Gibt es so etwas wie ein bestes Ergebnis?

- F **Eigene freie Ideen der Weiterentwicklung.** Z.B. finden Sie ein anderes Paket mit SMOTE mit mehr/besseren Möglichkeiten?
- G **Exportieren Sie die relevanten Dateien nach SAS** (\*.sas7bdat) und bauen Sie echt vergleichbare SAS EMiner Modelle auf, dokumentieren in Kurzform die Ergebnisse.

Bewerten und vergleichen Sie die R-Ergebnisse mit den SAS-Ergebnissen.

- Zunächst nur ohne SMOTE (sollte sehr schnell gehen - in Analogie zu DM 1).
- Versuchen Sie etwas mit SMOTE.
- H **Versuchen Sie dasselbe mit(über/via PYTHON).** Dazu gibt es verschiedene Vorgehensweisen. Werden Sie kreativ. Dokumentieren Sie stets ihre Ergebnisse.

Bewerten und vergleichen Sie die R-Ergebnisse mit den SAS-Ergebnissen und den PYTHON-Ergebnissen (Ergebnisse schließt Bewertung der Programme/Skripte und des Outputs plus Interpretation ein).

Lösung:

Als erstes werden die Daten aufbereitet. Dabei werden zunächst die Daten eingelesen und die Spalten erhalten die dazugehörigen Namen zugewiesen. Das Excell Sheet dazu sieht wie folgt aus.

| A           | B    | C   | D            | E                | F              | G               | H                 | I                  | J        | K    |
|-------------|------|-----|--------------|------------------|----------------|-----------------|-------------------|--------------------|----------|------|
| target      | age  | sex | on_thyroxine | query_on_thyroid | on_antithyroid | thyroid_surgery | query_hypothyroid | query_hyperthyroid | pregnant | sick |
| hypothyroid | 72 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 15 F | t   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 24 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 24 F | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 77 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 85 F | f   | f            | f                | f              | t               | f                 | f                  | f        | f    |
| hypothyroid | 64 F | f   | f            | f                | t              | f               | f                 | f                  | f        | f    |
| hypothyroid | 72 F | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| hypothyroid | 20 F | f   | f            | f                | f              | t               | f                 | f                  | f        | f    |

Anschließend wird die TARGET Variable in hypothyroid=1 und negative=0 umcodiert.

| A      | B    | C   | D            | E                | F              | G               | H                 | I                  | J        | K    |
|--------|------|-----|--------------|------------------|----------------|-----------------|-------------------|--------------------|----------|------|
| target | age  | sex | on_thyroxine | query_on_thyroid | on_antithyroid | thyroid_surgery | query_hypothyroid | query_hyperthyroid | pregnant | sick |
| 1      | 72 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 15 F | t   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 24 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 24 F | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 77 M | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 85 F | f   | f            | f                | f              | t               | f                 | f                  | f        | f    |
| 1      | 64 F | f   | f            | f                | t              | f               | f                 | f                  | f        | f    |
| 1      | 72 F | f   | f            | f                | f              | f               | f                 | f                  | f        | f    |
| 1      | 20 F | f   | f            | f                | f              | t               | f                 | f                  | f        | f    |

Zusätzlich werden alle anderen Zellen in binäre Variablen übersetzt.

- Unbekannt: ? = NA
- False: f = 0
- True: t = 1
- No: n = 0
- Yes: y = 1
- Male: M = 0
- Female: F = 1

| A      | B   | C   | D            | E                | F              | G               | H                 | I                  | J        | K    |
|--------|-----|-----|--------------|------------------|----------------|-----------------|-------------------|--------------------|----------|------|
| target | age | sex | on_thyroxine | query_on_thyroid | on_antithyroid | thyroid_surgery | query_hypothyroid | query_hyperthyroid | pregnant | sick |
| 1      | 72  | 0   | 0            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 15  | 1   | 1            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 24  | 0   | 0            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 24  | 1   | 0            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 77  | 0   | 0            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 85  | 1   | 0            | 0                | 0              | 0               | 1                 | 0                  | 0        | 0    |
| 1      | 64  | 1   | 0            | 0                | 0              | 1               | 0                 | 0                  | 0        | 0    |
| 1      | 72  | 1   | 0            | 0                | 0              | 0               | 0                 | 0                  | 0        | 0    |
| 1      | 20  | 1   | 0            | 0                | 0              | 0               | 1                 | 0                  | 0        | 0    |

Schaut man nun die absolute bzw. auch die relative Häufigkeit der Verteilung von 0 und 1 von TARGET Variable, so erkennt man dass es sich hierbei um eine unbalancierte Datenmenge handelt. Bei 5% handelt es sich eindeutig um einen verzerrten Datensatz, auch bekannt als seltenes Ereignis, englisch: rare event.

```
```{r}
# check balance of outcome variable
print(table(hyper$target))
print(prop.table(table(hyper$target)))
```

```

|            | 0          | 1 |
|------------|------------|---|
| 3012       | 151        |   |
|            | 0          | 1 |
| 0.95226051 | 0.04773949 |   |

Die fehlenden Beobachtungen, also NA Werte, bei den Variablen AGE, TSH, T3, TT4, T4U und FTI werden durch deren Mittelwerte ersetzt. Natürlich wird bei der binären Variable SEX nicht mit der Mittelwert, sondern es wird dazu die passende Funktion geschrieben und damit ersetzt. Zusätzlich wird noch bei der Variable TBG alle Zeilen mit 0 aufgefüllt, da nur wenige Zeilen mit Zahlen gefüllt waren. Anschließend werden die Daten in Trainings- und Testmengen mit einer 50 prozentigen Wahrscheinlichkeit aufgeteilt. Somit werden die Daten von 3163 Beobachtungen in 1582 Trainings- und in 1581 Testdatensätze unterteilt. Die dazugehörige absolute und relative Häufigkeiten für TARGET Variable kann dann in der untenstehenden Abbildung erkannt werden.

```
```{r}
library(caret)
set.seed(1234)
splitIndex <- createDataPartition(hyper$target, p = .50,
                                  list = FALSE,
                                  times = 1)
trainSplit <- hyper[ splitIndex, ]
testSplit <- hyper[-splitIndex, ]
```

```

```
```{r}
table(trainSplit$target)
table(testSplit$target)
```

```

|      | 0  | 1 |
|------|----|---|
| 1503 | 79 |   |
| 1509 | 72 |   |

```
```{r}
prop.table(table(trainSplit$target))
prop.table(table(testSplit$target))
```

```

|            | 0          | 1 |
|------------|------------|---|
| 0.95006321 | 0.04993679 |   |
| 0.9544592  | 0.0455408  |   |

Das Ergebnis Balance der Daten liegt bei beiden Teilen immer noch bei etwa 5%.

In dieser Aufgabenstellung soll eine binäre TARGET Variable vorausgesagt werden. Möchte man nun ein Entscheidungsbaums-Modell hierfür nehmen, so verwendet man den CART-Verfahren. Der CART-Algorithmus (Classification and Regression Trees) dient Entscheidungsfindung und wird auch daher bei Entscheidungsbäumen eingesetzt. CART wurde erstmals 1984 von Leo Breiman publiziert. Ein bedeutendes Merkmal des CART-Algorithmus ist, dass CART-Verfahren nur rein binäre Entscheidungsbäume unterstützt, das heißt, bei jedem Schritt erfolgt die Aufteilung in 2 Teilmengen und somit sind in jeder Verzweigung immer genau zwei Äste vorhanden. Das zentrale Element dieses Algorithmus ist somit das Finden einer optimalen binären Trennung [WICA]. Zudem gibt es Ensemble-Methoden, die multiple Modelle zu einem komplexen Gesamtmodell kombinieren, um bessere Prognosegüte zu erreichen. Hier in diesem Fall werden also mehr als nur einen Entscheidungsbaum konstruiert und gehofft ein besseres Vorhersagemodell zu erhalten als mit der einfachen Entscheidungsbaum Variante. Es gibt zwei sehr wichtigste Ensemble-Methoden, und zwar die Bagging und Boosting Methoden. Hier in der Aufgabe wird mit Bagging gearbeitet. Bagging (Bootstrap aggregating) ist wieder eine Methode von Leo Breiman, die im Jahre 1996 vorgestellt wurde. Bagging wendet Bootstrap-Prinzip an, das heißt es findet eine zufällige Auswahl des Trainingsets mit Zurücklegen statt. Somit liefert Bootstrap-Algorithmus Trainingsmengen aus denen dann verschiedene Prädiktoren generiert werden. Da es hier sich um einem Klassifikationsproblem handelt, wird am Ende die Klasse als beste vorausgesagt die am häufigsten prognostiziert wurde [LB96].

Nun wird in R mit Hilfe von METHODE TREEBAG diese CART Algorithmus mit Bagging Methode angewendet. Dabei kann man unten auch die am Ende vorausgesagten Wahrscheinlichkeiten betrachten. Diese wurden für alle 1581 Beobachtungen in der Testmenge generiert. Man kann die Wahrscheinlichkeiten aber auch als Summe darstellen.

```
addmargins(table(pred)) # ist binär (0 / 1), wenn testSplit$target binär (factor) ist
ist metrisch, wenn testSplit$target nicht binär (factor) ist
...
pred
0.00163331139489972 0.00555095037911386 0.0148833270018522 0.016606211123897
 1462 5 1 4
0.0166316388199091 0.0271022360525626 0.0318641408144673 0.0343569615773087
 3 2 8 9
0.0358641408144673 0.0398118199443312 0.0496488560991826 0.052982189432516
 1 2 2 2
0.0569821894325159 0.0937821894325159 0.132756548406875 0.138300870751197
 1 1 1 1
0.159100870751197 0.175218086868413 0.22195801360834 0.239151661802112
 1 1 1 1
0.273818328468779 0.689900741027155 0.846653447413623 0.90475855518761
 1 1 2 3
0.917422678182854 0.929131538823267 0.945107993061901 0.946133634087542
 3 3 2 2
0.948551828678339 0.94925901510766 0.949866697931221 0.950269541423449
 1 1 1 1
0.954431468975766 0.955441995291555 0.961907993061901 0.961923107771752
 2 2 34 2
0.962933634087542 Sum
 11 1581
```
{r}
pred # das heißt 0.001633311 wurde 1462 mal als wkeit berechnet
[1] 0.961907993 0.917422678 0.962933634 0.001633311 0.961907993 0.001633311 0.961907993
[8] 0.961907993 0.962933634 0.961907993 0.929131539 0.962933634 0.961907993 0.904758586
[15] 0.961907993 0.961907993 0.961907993 0.950269541 0.961907993 0.961907993 0.961907993
[22] 0.001633311 0.961907993 0.001633311 0.962933634 0.961907993 0.056982189 0.929131539
[29] 0.961907993 0.961907993 0.962933634 0.961907993 0.948551829 0.961907993 0.961907993
[36] 0.955441995 0.954431469 0.239151662 0.962933634 0.929131539 0.946133634 0.273818328
[43] 0.961907993 0.961907993 0.961907993 0.917422678 0.949866698 0.946133634 0.961907993
...
```

Natürlich werden hierzu die Confusion Matrix mit der Accuracy und zum Modell dazugehörige ROC Kurve und AUC Wert abgebildet. Hier fällt insgesamt auf, dass bei 1581 Testdaten nur 71 mal die 1, also TARGET=1=Patient hat Unterfunktion, prognostiziert wurde.

```
Confusion Matrix and Statistics

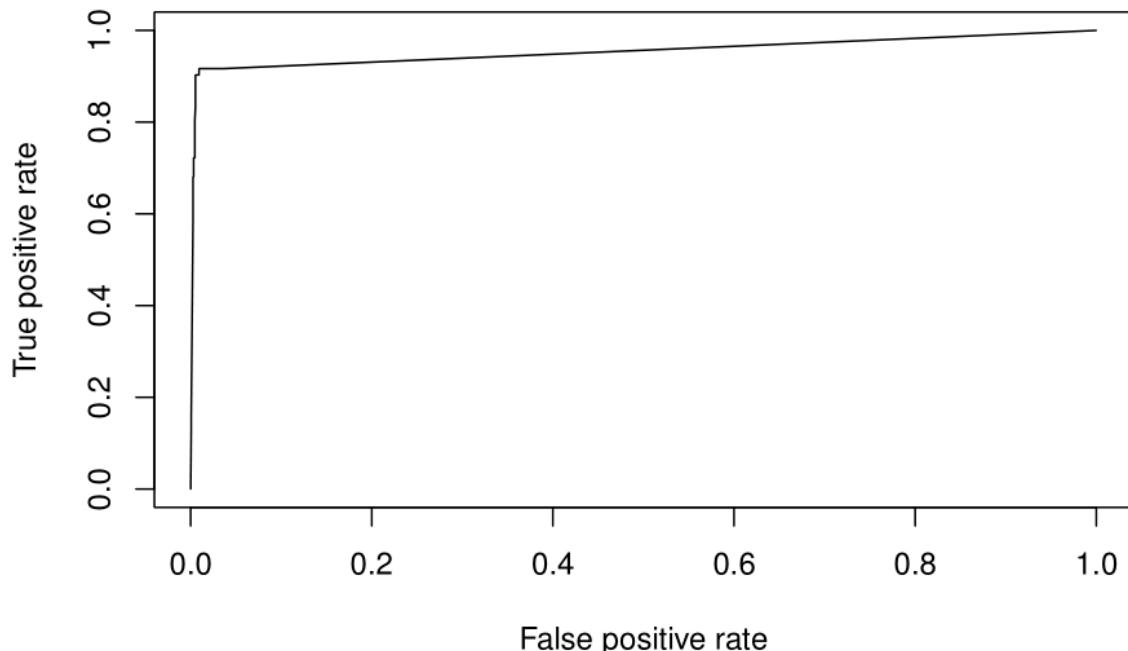
           reference
data      0      1
0  1501     9
1       8    63

Accuracy : 0.9892
95% CI  : (0.9828, 0.9937)
No Information Rate : 0.9545
P-Value [Acc > NIR] : 2.733e-15

Kappa : 0.8755
McNemar's Test P-Value : 1

Sensitivity : 0.9947
Specificity : 0.8750
Pos Pred Value : 0.9940
Neg Pred Value : 0.8873
Prevalence : 0.9545
Detection Rate : 0.9494
Detection Prevalence : 0.9551
Balanced Accuracy : 0.9348

'Positive' Class : 0
```



```
auctree <- performance(prtree, measure = "auc")
auctree <- auctree@y.values[[1]]
auctree

## [1] 0.9547115
```

Nun wird die Unbalanciertheit der Daten näher betrachtet. Ein Datensatz ist unbalanciert, wenn die Klassen nicht gleich groß repräsentiert sind. Im HYPOTHYROID Datensatz ist es durch das Vorkommen von 0 in 3012 Beobachtungen für TARGET Variable, wobei die 1 nur in 151 Fälle auftritt, erkennbar. Damit würde man 0 als majority class und 1 als minority class deklarieren. Es gibt unterschiedliche mögliche Ansätze die man bei so einer Problemstellung verwenden kann. Hier wird mit SMOTE (Synthetic Minority Over-sampling TEchnique) gearbeitet. Dieses Verfahren produziert, wie der Name schon sagt, synthetische Datenpunkte und zieht diese als Stichproben anstatt einfach von der minority class überrepräsentierte Stichproben zu ziehen. Somit werden also synthetische Stichproben erstellt anstatt einfach die aus dem minority class zu kopieren. Diese Stichproben mit Hilfe der k nächsten Nachbarn (nearest neighbors) der minority class erstellt und aus der majority class werden unterrepräsentierte Stichproben entnommen, was dann am Ende zu einem gut balancierten Datensatz führen wird [CH02]. Es sollte noch erwähnt werden, dass für R und Python k=5 als default Wert für nächste Nachbarn implementiert.

```
```{r}
library(DMwR)

testSplit$target <- as.factor(testSplit$target)
trainSplit$target <- as.factor(trainSplit$target)

#trainSplit$target <- as.factor(trainSplit$target)
trainSplit <- SMOTE(target ~ ., trainSplit, perc.over = 100, perc.under=200)
trainSplit$target <- as.numeric(trainSplit$target)

...```

```

```
```{r}

write.table(x = trainSplit, file = paste(Pfad,"hyper_01_bin_mean_trainsplit_SMOTE.
= ",", sep = ";", row.names = FALSE)
...```

```

```
```{r}
table(trainSplit$target)

prop.table(table(trainSplit$target))
##die WKEITEN die 1 und 0 aufgeteilt sind hier das idealfall--also 50/50
...```

```

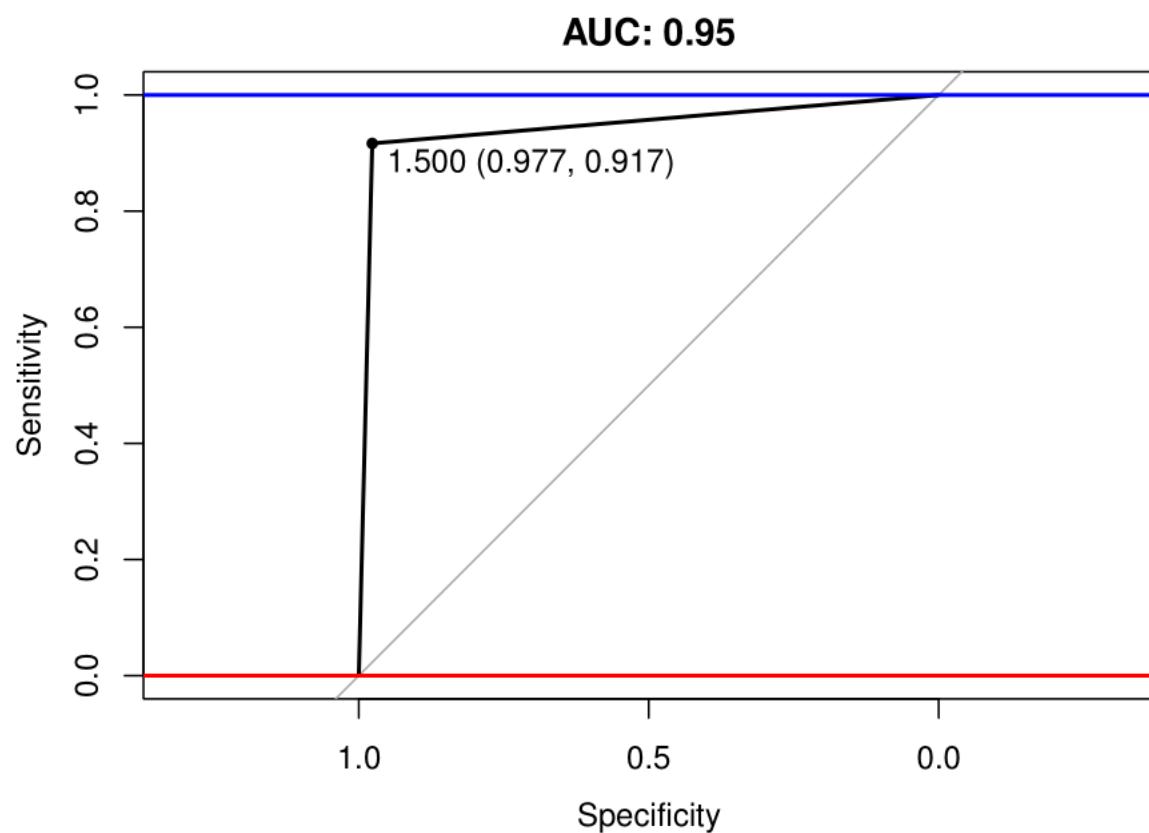
0	1
158	158

0	1
0.5	0.5

Mit Hilfe von perc.over und perc.under wurden jetzt von der minority class und von der majority class entnommen.

Im nächsten Schritt wird nun wieder auf die Testdaten das CART Algorithmus mit Bagging Methode - also treebag Modell aus R - angewendet und die Klassen prognostiziert.

Zudem hat die neue ROC Kurve einen AUC Wert von 0.9467.



Das ganze wird natürlich noch einmal in Python implementiert. Hierbei wurden einfacheheitshalber die bereinigten Daten als csv Datei eingelesen und mit denen weiter gearbeitet.

```
hyper.head()
```

	target	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	thyroid_surgery	query_hypothyroid	...
0	1	72.0	0	0	0	0	0	0	0
1	1	15.0	1	1	0	0	0	0	0
2	1	24.0	0	0	0	0	0	0	0
3	1	24.0	1	0	0	0	0	0	0
4	1	77.0	0	0	0	0	0	0	0

5 rows × 25 columns

```
feature_cols = ['age', 'sex', 'on_thyroxine', 'query_on_thyroxine', 'on_antithyroid_medication',
X = hyper[feature_cols]
y = hyper.target
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=1234)
```

Nun lässt man wie in R ausgeben wie die Klassen 1 und 0 verteilt sind. In R waren 1582 Beobachtungen als Trainingsdatensatz und 1581 als Testdatensatz aufgeteilt worden. Hier in Python ist das genau umgekehrt, also 1581 als Training- und 1582 als Testdatenmenge. Dies kann an verschiedene seed() Einstellungen der Programme liegen. Zudem gilt bei Python, dass die Zufälligkeit an der jeweiligen Definition innerhalb der Packages geknüpft ist. Hier in Aufteilung der Daten wurde zum Beispiel sklearn verwendet. Es gibt aber noch andere Packages wie pandas oder direkt random bei denen dann seed unterschiedlich definiert ist.

```
import collections

counter_train=collections.Counter(y_train)
print(counter_train)

Counter({0: 1503, 1: 78})
```

```
counter_test=collections.Counter(y_test)
print(counter_test)

Counter({0: 1509, 1: 73})
```

Zudem konnte hier nicht genau "treebag" Methode aus R implementiert werden. Es ist zwar ganz ähnlich - der einzige Unterschied hier ist dass 100 Bäume für den Bagging Part erstellt werden. Daher kann auch am Ende nicht genau gesagt werden welche der Programme und dazugehörigen Modelle besser in der Vorhersage waren.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

tree = DecisionTreeClassifier()
bag = BaggingClassifier(tree, n_estimators=100, random_state=1234)
bag.fit(X_train, y_train)

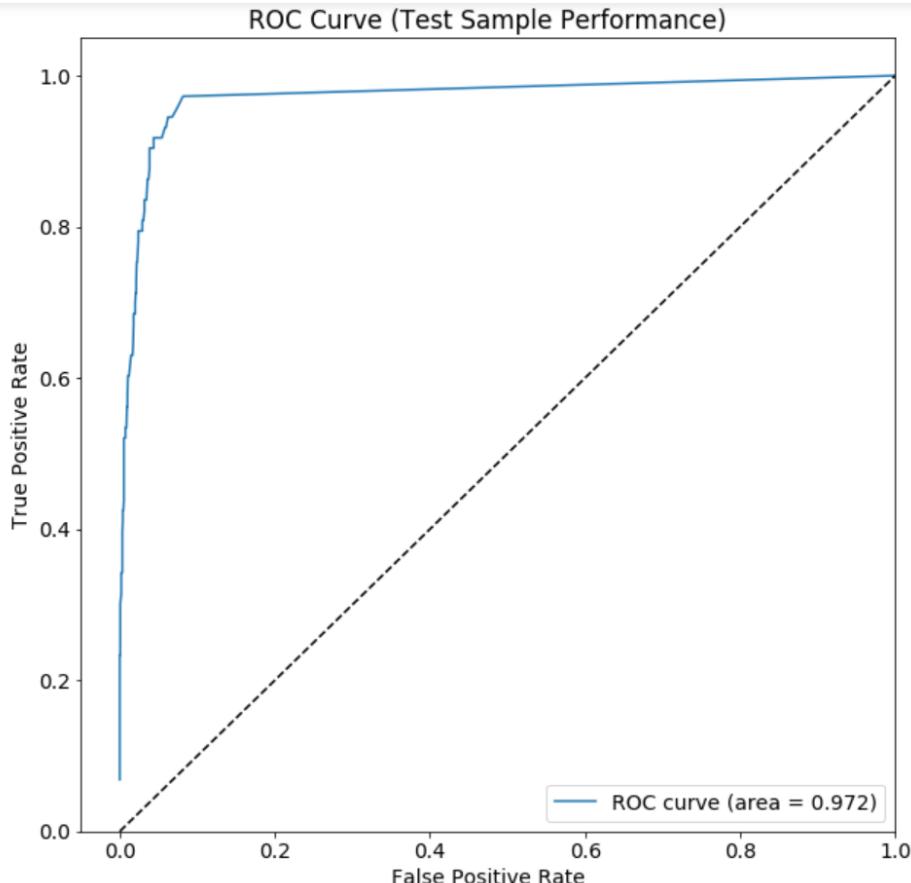
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None, c
 max_features=None, max_leaf_nodes=None,
 min_impurity_decrease=0.0, min_impurity_split=None,
 min_samples_leaf=1, min_samples_split=2,
 min_weight_fraction_leaf=0.0, presort=False, random_state=None,
 splitter='best'),
 bootstrap=True, bootstrap_features=False, max_features=1.0,
 max_samples=1.0, n_estimators=100, n_jobs=1, oob_score=False,
 random_state=1234, verbose=0, warm_start=False)

ypred = bag.predict(X_test)
pd.crosstab(ypred,y_test)
```

target	0	1
row_0		
0	1479	23
1	30	50

```
bag.score(X_test, y_test)
```

0.9664981036662452



Nun kann man, wie bei R Teil, perfekt balancierte Datensätze mit Hilfe von SMOTE Algorithmus erzeugen.

```
from imblearn.over_sampling import SMOTE # über Terminal: pip install imblearn
X_resampled, y_resampled = SMOTE().fit_sample(X_train, y_train)
#print(sorted(counter(y_resampled).items()))
```

```
counter_y=collections.Counter(y_resampled)
print(counter_y)
```

```
Counter({0: 1503, 1: 1503})
```

```
treeSmote = DecisionTreeClassifier()
bagSmote = BaggingClassifier(treeSmote, n_estimators=100,
 random_state=1234)
```

```
bagSmote.fit(X_resampled, y_resampled)
```

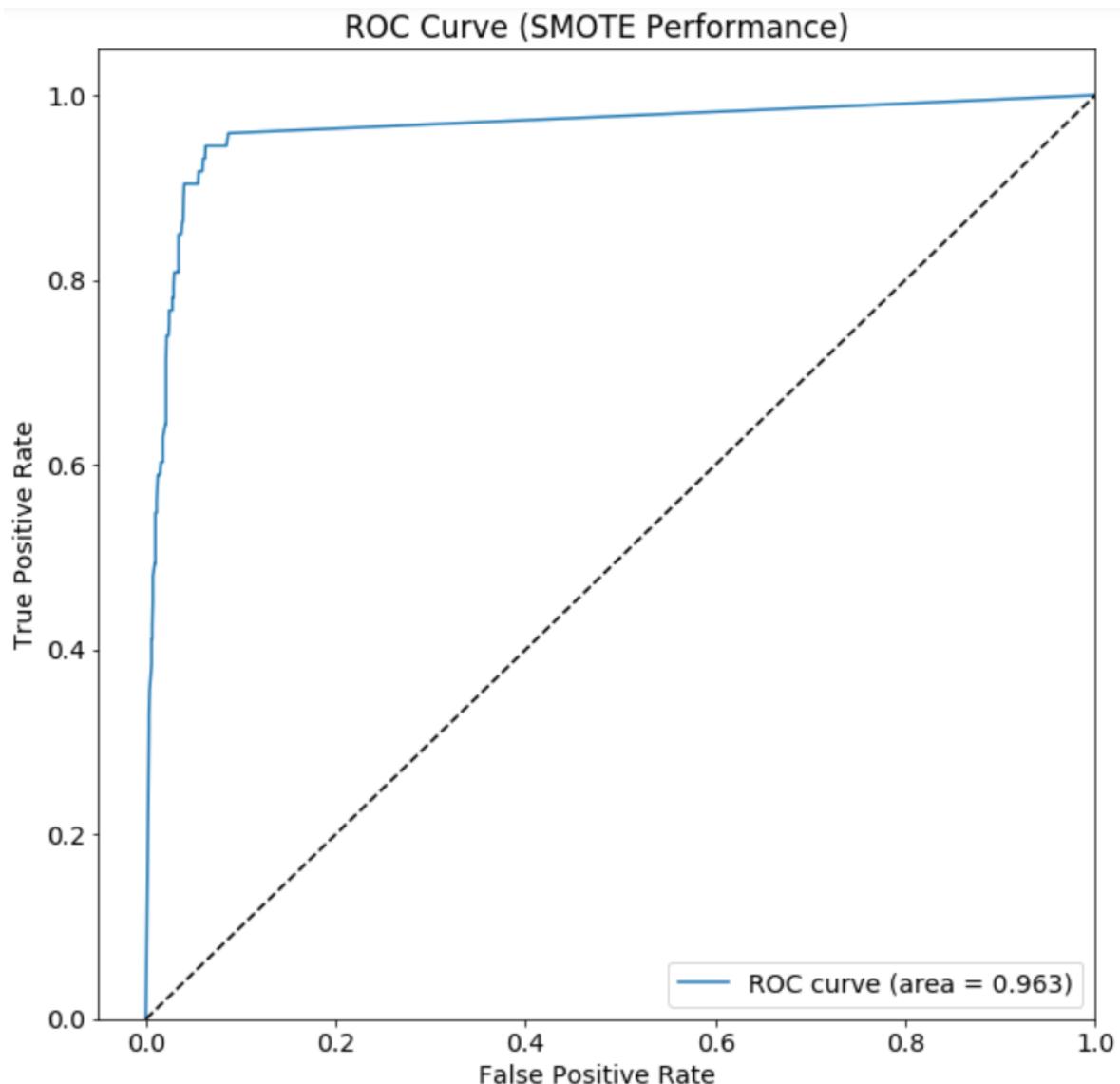
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None,
 max_features=None, max_leaf_nodes=None,
 min_impurity_decrease=0.0, min_impurity_split=None,
 min_samples_leaf=1, min_samples_split=2,
 min_weight_fraction_leaf=0.0, presort=False, random_state=None,
 splitter='best'),
 bootstrap=True, bootstrap_features=False, max_features=1,
 max_samples=1.0, n_estimators=100, n_jobs=1, oob_score=False,
 random_state=1234, verbose=0, warm_start=False)
```

```
ypredSmote = bagSmote.predict(X_test)
pd.crosstab(ypredSmote,y_test)
```

target	0	1
row_0		
0	1472	18
1	37	55

```
bagSmote.score(X_test, y_test)
```

```
0.9652338811630847
```



Somit ergab für das Entscheidungsbaum Modell mit Bagging Verfahren vor SMOTE Anwendung einen Accuracy von 0.966 und einen AUC Wert von 0.972 und nach SMOTE Methode ergaben dann 0.965 und 0.963. In R ergaben vor SMOTE die Werte 0.989 für Accuracy und 0.955 für AUC, während es nach SMOTE 0.974 und 0.947 berechnet wurden.

Insgesamt könnte man sagen, dass die SMOTE Methode eventuell nicht viel dazu beigetragen hat. Obwohl in den Daten die Klasse 1 deutlich unter-, mit 5 %, und die Klasse 0 über-, mit 95%, -repräsentiert sind, ergaben beim Entscheidungsbaum mit Bagging nach SMOTE Anwendung nicht sehr große Unterschiede bei der Verbesserung. Dies kann sich für andere Modelle, wie zum Beispiel Support Vector Machines oder logistische Regression, natürlich ganz anders aussehen. Um den Rahmen der Ausarbeitung nicht zu überstrapazieren wurden weitere Modelle nicht mehr aufgenommen.

## 5 Aufgabe 4

---

### 5.1 4 A

*Aufgabenstellung:*

Bauen Sie das Modell und die Analyse von **Comparing Architectures** aus Abschnitt 2-59 nach.

Bewerten Sie die Ergebnisse kritisch.

Nehmen Sie einige (kleine) Ergänzungen und Erweiterungen vor.

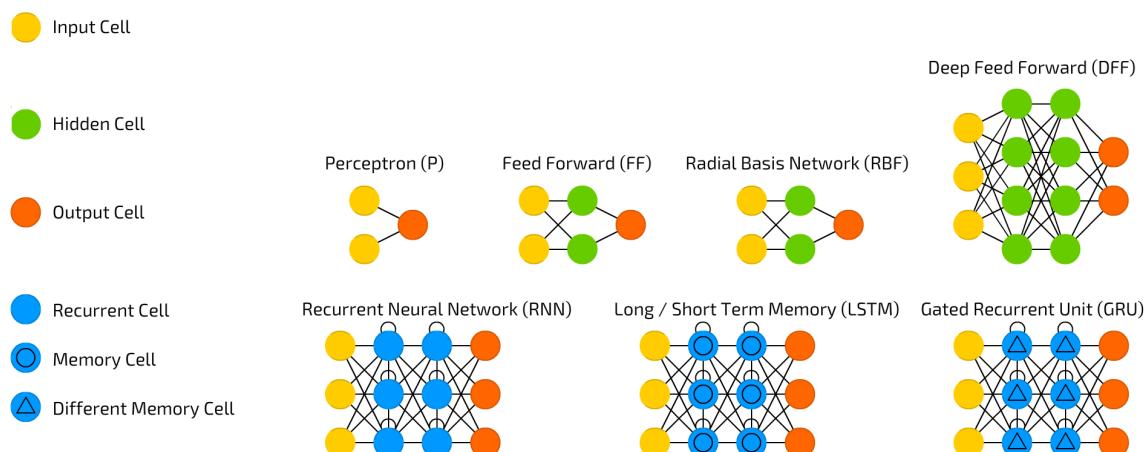
Fügen Sie danach Knoten mit HPSVM und HPFOREST ggf. andere hinzu.

Wer ist der Sieger (Champion Modell) nach welchem Kriterium?

*Lösung:*

Die Funktionsweise und der Aufbau künstlicher neuronaler Netze wird hier im Rahmen der Ausarbeitung nur kurz erläutert, da es in der Vorlesung schon sehr ausführlich besprochen und verständlich vermittelt wurde. Die künstlichen neuronalen Netze sind schichtweise in sogenannten Layern (Schichten) angeordnet und in einer festen Hierarchie miteinander verbunden. Mit der *Input Layer* fließen Informationen über eine oder mehrere *Hidden Layer* bis hin zur *Output Layer*. Natürlich kann auch dann der Output des einen Neurons der Input des nächsten sein [NN18].

Unten befindet sich eine kurze übersichtliche Grafik mit einigen Versionen neuronaler Netze die später in dieser Ausarbeitung noch als Modelle verwendet werden. Die gesamte Überblick kann auf der Webseite [NN18] betrachtet werden.



Es werden hier nur die drei Arten der neuronalen Netze kurz beschrieben, welche später eine hohe Relevanz für die kommenden Aufgaben haben.

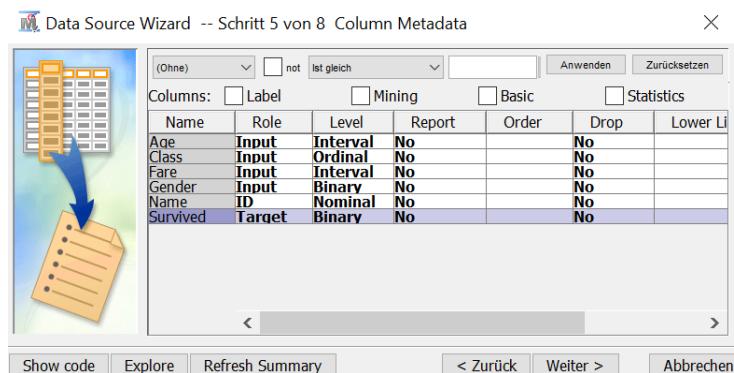
Bei *Feedforward Netze* werden Informationen von der Input Layer über die Hidden Layer bis hin zur Output Layer in eine Richtung, forward also vorwärts, weitergereicht.

*Rekurrente Netze*, auch rückgekoppelte oder Feedback Netze genannt, sind wie Feedforward-Netze, jedoch existieren zusätzliche Verbindungen durch die dann Informationen bestimmte Bereiche des Netzwerkes auch rückwärts beziehungsweise erneut durchlaufen können.

*Long short-term memory* (auf deutsch langes Kurzzeitgedächtnis) ist so zusagen angepasster RNN die aber so konstruiert sind, dass sie sich Informationen über einen größeren Zeitraum merken können. RNN hat ein *vanishing exploding gradients* Problem haben, die dann durch LSTM mit Hilfe von input and forget gates gelöst wurde [NN18].

Zusätzlich gibt es verschiedene Lernstrategien für diverse Netzwerktypen, die hier nicht näher erläutert wird. Wie schon in der Einleitung erwähnt, die neuronale Netze sind massiv lernfähige Systeme die anhand von Trainingsbeispielen lernen. Beim Training entsteht dann eine Differenz zwischen der tatsächlichen Ausgabe des neuronalen Netzes und einer gewünschten Ausgabe, welche in den Trainingsdaten bekannt ist. Somit entstehen also Trainings- oder auch Netzwerkfehler, wobei man am Ende diesen Fehler minimieren möchte [AF11].

Die Bearbeitung der Aufgabe 4 wird mit Hilfe von der SAS PDF Datei über Neurale Netze Modellierung [DY17] erfolgen. Für die erste Aufgabeteil A wird wieder der **titanic** Datensatz verwendet. Nur hier werden weniger Variablen, im Vergleich zu Aufgabe 2, aufgelistet und mehr Datensätze - hier 1309 - verwendet. Als erster Schritt werden Regressoren bzw. unabhängige Variablen in die richtige Metrik eingeordnet. Dann wird als Target, wie bei Aufgabe 2 auch, wieder SURVIVED genommen, das heißt man möchte am Ende vorhersagen können ob ein Passagier überlebt hat oder nicht. Hier sollte noch erwähnt werden, dass die Daten nicht in TRAIN und VALIDATION Sets aufgeteilt werden. Das bedeutet alle Resultate werden auf Trainingsdatensatz bewertet.

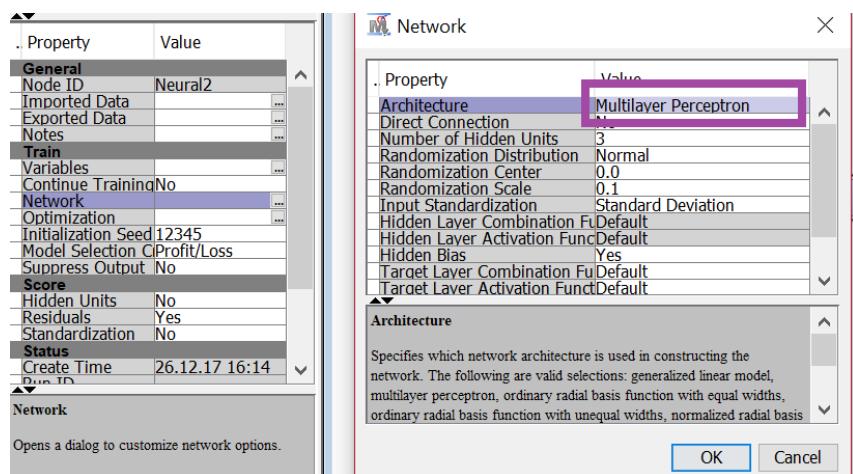
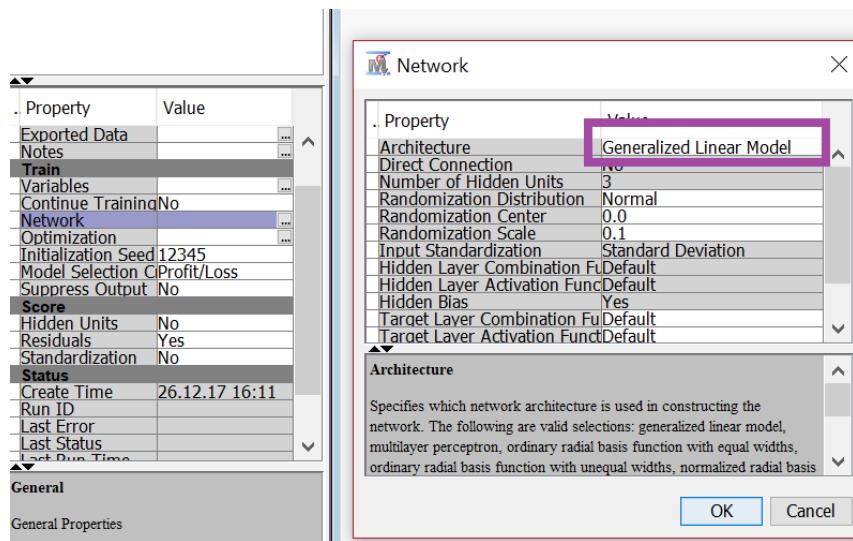


Es werden gleich verschiedene neuronale Netze miteinander verglichen. Diese sind:

- GLM: Generalisierte Lineare Modelle - hat keine hidden layers
- MLP: Multi-layer perceptron - feedforward-Netze und bei Enterprise Miner ist dies die Default Einstellung, wobei keine hidden layers und keine hidden units. Zudem sind die Neuronen der einzelnen Schichten vollverknüpft.
- ORBF: Ordinary Radial Basis Function - feedforward-Netze mit single hidden layer. Die hidden units haben glockenförmige-, normal- oder Gauß-Verteilung. Es verwendet die Exponential-Aktivierungsfunktion um den Gaußschen zu erzeugen, die in den Gewichten der hidden layer lokalisiert ist.
- NRBF: Normalized Radial Basis Functions - die Aktivierung von jeweiligen hidden unit ist ein Produkt der kooperativen Aktivierung aller beteiligten hidden units in der layer.

Es gibt noch Kombinationsfunktionen, die beim Erstellen eines Radial Basis Function geeignet sind. Es sind Kombinationen von equal/unequal mit Breite/Höhe/Volumen.

Es werden nun also insgesamt neun Neuronale Netze mit unterschiedlicher Architekturkonzepten eingebaut.



<b>General</b>	Node ID	Neural3	<b>Property</b>	Value
General	Imported Data	...	Architecture	Ordinary Radial - Equal Width
General	Exported Data	...	Direct Connection	No
General	Notes	...	Number of Hidden Units	3
<b>General</b>	Node ID	Neural4	<b>Property</b>	Value
General	Imported Data	...	Architecture	Ordinary Radial - Unequal Width
General	Exported Data	...	Direct Connection	No
General	Notes	...	Number of Hidden Units	3
<b>General</b>	Node ID	Neural5	<b>Property</b>	Value
General	Imported Data	...	Architecture	Normalized Radial - Equal Height
General	Exported Data	...	Direct Connection	No
General	Notes	...		
<b>Train</b>				
<b>General</b>	Node ID	Neural6	<b>Property</b>	Value
General	Imported Data	...	Architecture	Normalized Radial - Equal Volumes
General	Exported Data	...	Direct Connection	No
General	Notes	...	Number of Hidden Units	3
<b>General</b>	Node ID	Neural7	<b>Property</b>	Value
General	Imported Data	...	Architecture	Normalized Radial - Equal Width
General	Exported Data	...	Direct Connection	No
General	Notes	...	Number of Hidden Units	3
<b>General</b>	Node ID	Neural8	<b>Property</b>	Value
General	Imported Data	...	Architecture	Normalized Radial - Equal Width and Height
General	Exported Data	...	Direct Connection	No
General	Notes	...		
<b>General</b>	Node ID	Neural9	<b>Property</b>	Value
General	Imported Data	...	Architecture	Normalized Radial - Unequal Width and Height
General	Exported Data	...	Direct Connection	No
General	Notes	...	Number of Hidden Units	3

Zu diesen Neuronalen Netzen werden noch zusätzlich Knoten mit HPSVM - für HPSVM wurden mehrere Kerne ausprobiert und das beste, hier POLYNOMIAL mit DEGREE=3, genommen - und HPFOREST hinzugefügt. Schließlich werden die Modelle mit Hilfe der MODEL COMPARISON miteinander verglichen.

Fit Statistics Model Selection based on Train: Misclassification Rate (_MISC_)							
Selected Model	Model Node	Model Description	Train:	Average	Train:	Train: Gini Coefficient	
			Misclassification Rate	Squared Error	Roc Index		
Y	HPDMForest	HP Forest	0.13216	0.09700	0.947	0.894	
	HPSVM2	HP SVM-best	0.20015	0.16608	0.834	0.669	
	Neural3	ORBFEQ	0.20168	0.15298	0.828	0.655	
	Neural9	NRBFUN	0.20474	0.15416	0.808	0.615	
	Neural8	NRBFEQ	0.20550	0.15502	0.812	0.623	
	Neural7	NRBFEW	0.20626	0.15233	0.823	0.647	
	Neural6	NRBFEV	0.20932	0.15230	0.824	0.648	
	Neural5	NRBFEH	0.21085	0.15121	0.826	0.652	
	Neural4	ORBFUN	0.21390	0.15230	0.829	0.658	
	Neural2	MLP	0.21543	0.15169	0.824	0.648	
	Neural	GLM	0.23606	0.17629	0.783	0.567	

Fit Statistics Table Target: Survived		
Data Role=Train		
Statistics	HPDMForest	HPSVM2
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.42	0.42
Train: Kolmogorov-Smirnov Statistic	0.72	0.56
Train: Akaike's Information Criterion	.	.
Train: Average Squared Error	0.10	0.17
Train: Roc Index	0.95	0.83
Train: Average Error Function	.	.
Train: Cumulative Percent Captured Response	26.20	24.40
Train: Percent Captured Response	13.00	12.20
Selection Criterion: Train: Misclassification Rate	0.13	0.20
Train: Degrees of Freedom for Error	.	.
Train: Model Degrees of Freedom	.	.
Train: Total Degrees of Freedom	.	.
Train: Frequency of Classified Cases	1309.00	1309.00
Train: Divisor for ASE	2618.00	2618.00
Train: Error Function	.	.
Train: Final Prediction Error	.	.
Train: Gain	161.80	143.81
Train: Gini Coefficient	0.89	0.67
Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.72	0.56
Train: Kolmogorov-Smirnov Probability Cutoff	0.37	0.41
Train: Cumulative Lift	2.62	2.44
Train: Lift	2.62	2.46
Train: Maximum Absolute Error	0.90	0.96
Train: Misclassification Rate	0.13	0.20
Train: Mean Squared Error	.	.
Train: Sum of Frequencies	1309.00	1309.00
Train: Number of Estimated Weights	.	.
Train: Root Average Squared Error	0.31	0.41
Train: Cumulative Percent Response	100.00	93.13
Train: Percent Response	100.00	93.85
Train: Root Final Prediction Error	.	.
Train: Root Mean Squared Error	.	.
Train: Schwarz's Bayesian Criterion	.	.
Train: Sum of Squared Errors	253.96	434.80
Train: Sum of Case Weights Times Freq	.	.
Train: Number of Wrong Classifications	173.00	262.00

Neural3	Neural9	Neural8	Neural7	Neural6	Neural5	Neural4	Neural2	Neural
0.44	0.61	0.48	0.46	0.49	0.61	0.43	0.45	0.67
0.53	0.51	0.51	0.52	0.52	0.53	0.53	0.51	0.47
1278.69	1297.22	1289.54	1274.85	1273.20	1267.10	1269.24	1263.64	1405.16
0.15	0.15	0.16	0.15	0.15	0.15	0.15	0.15	0.18
0.83	0.81	0.81	0.82	0.82	0.83	0.83	0.82	0.78
0.47	0.48	0.48	0.47	0.47	0.47	0.47	0.47	0.53
25.30	24.60	25.40	24.60	25.20	24.80	26.00	25.60	23.40
12.90	12.20	12.80	12.00	12.60	12.00	12.80	12.80	10.80
0.20	0.20	0.21	0.21	0.21	0.21	0.21	0.22	0.24
1289.00	1285.00	1290.00	1287.00	1288.00	1288.00	1287.00	1287.00	1303.00
20.00	24.00	19.00	22.00	21.00	21.00	22.00	22.00	6.00
1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00
.	.	.	.	.	.	.	.	.
2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00
1238.69	1249.22	1251.54	1230.85	1231.20	1225.10	1225.24	1219.64	1393.16
0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.18
152.81	145.81	153.81	145.81	151.81	147.81	159.80	155.80	133.82
0.66	0.62	0.62	0.65	0.65	0.65	0.66	0.65	0.57
0.52	0.51	0.51	0.50	0.51	0.52	0.52	0.50	0.47
0.41	0.48	0.49	0.43	0.41	0.41	0.43	0.44	0.63
2.53	2.46	2.54	2.46	2.52	2.48	2.60	2.56	2.34
2.60	2.46	2.58	2.42	2.54	2.42	2.58	2.58	2.17
0.98	0.95	0.96	0.97	0.97	0.95	0.98	0.98	0.93
0.20	0.20	0.21	0.21	0.21	0.21	0.21	0.22	0.24
0.16	0.16	0.16	0.15	0.15	0.15	0.15	0.15	0.18
1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00	1309.00
20.00	24.00	19.00	22.00	21.00	21.00	22.00	22.00	6.00
0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.42
96.56	93.89	96.95	93.89	96.18	94.66	99.24	97.71	89.31
99.23	93.85	98.46	92.31	96.92	92.31	98.46	98.46	83.08
0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.42
0.39	0.40	0.40	0.39	0.39	0.39	0.39	0.39	0.42
1382.23	1421.47	1387.90	1388.75	1381.92	1375.81	1383.13	1377.53	1436.23
400.49	403.59	405.85	398.81	398.73	395.86	398.72	397.12	461.53
2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00	2618.00
264.00	268.00	269.00	270.00	274.00	276.00	280.00	282.00	309.00

Event Classification Table

Model Selection based on Train: Misclassification Rate (\_MISC\_)

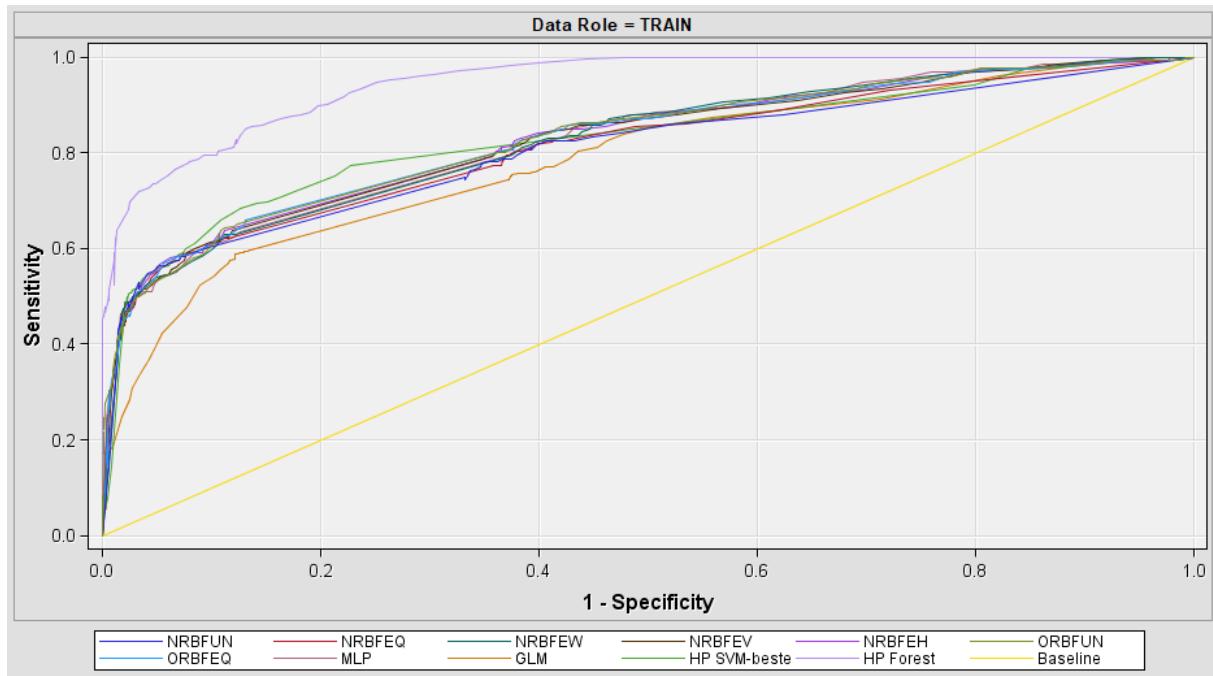
Model Node	Model Description	Data Role	Target	Target Label	False Negative	True Negative	False Positive	True Positive
HPDMForest	HP Forest	TRAIN	Survived		124	760	49	376
Neural9	NRBFUN	TRAIN	Survived		213	754	55	287
Neural8	NRBFEQ	TRAIN	Survived		212	752	57	288
Neural7	NRBFEW	TRAIN	Survived		230	769	40	270
Neural6	NRBFEV	TRAIN	Survived		217	752	57	283
Neural5	NRBFEH	TRAIN	Survived		200	733	76	300
Neural4	ORBFUN	TRAIN	Survived		208	737	72	292
Neural3	ORBFEQ	TRAIN	Survived		208	753	56	292
Neural2	MLP	TRAIN	Survived		245	772	37	255
Neural	GLM	TRAIN	Survived		204	704	105	296
HPSVM2	HP SVM-best	TRAIN	Survived		226	773	36	274

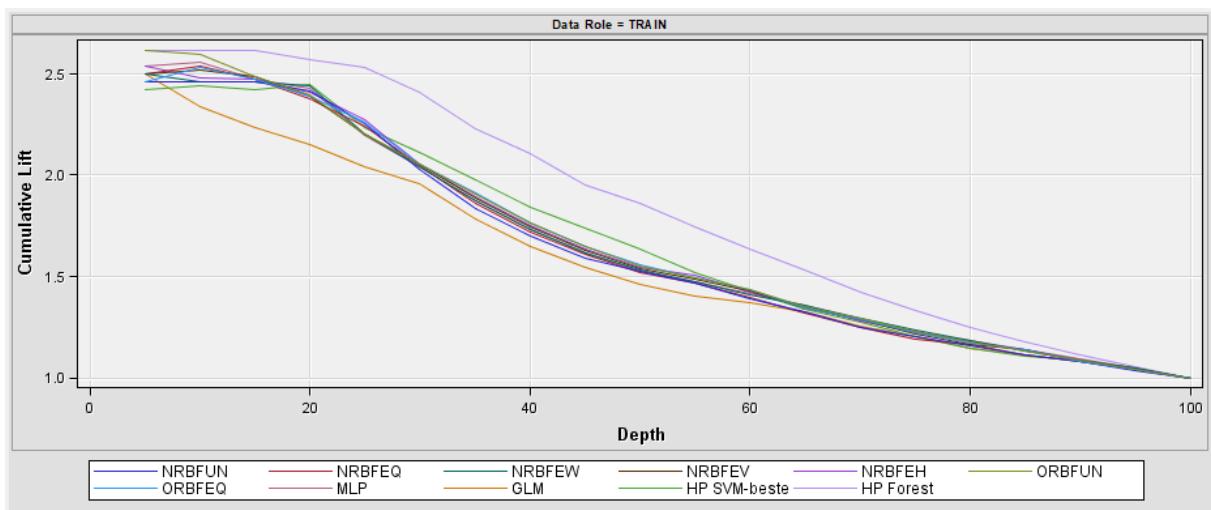
Laut Classification Table und Fit Statistics ist das Champion Modell HPFOREST. Da HPFOREST sehr gut abgeschnitten hat wurde noch im EMiner Datei zusätzlich DECISION TREE eingefügt und dennoch blieb HPFOREST als bestes Modell erhalten. Ein Ausschnitt davon ist das untenstehende Bild.

**Fit Statistics**  
Model Selection based on Train: Misclassification Rate (\_MISC\_)

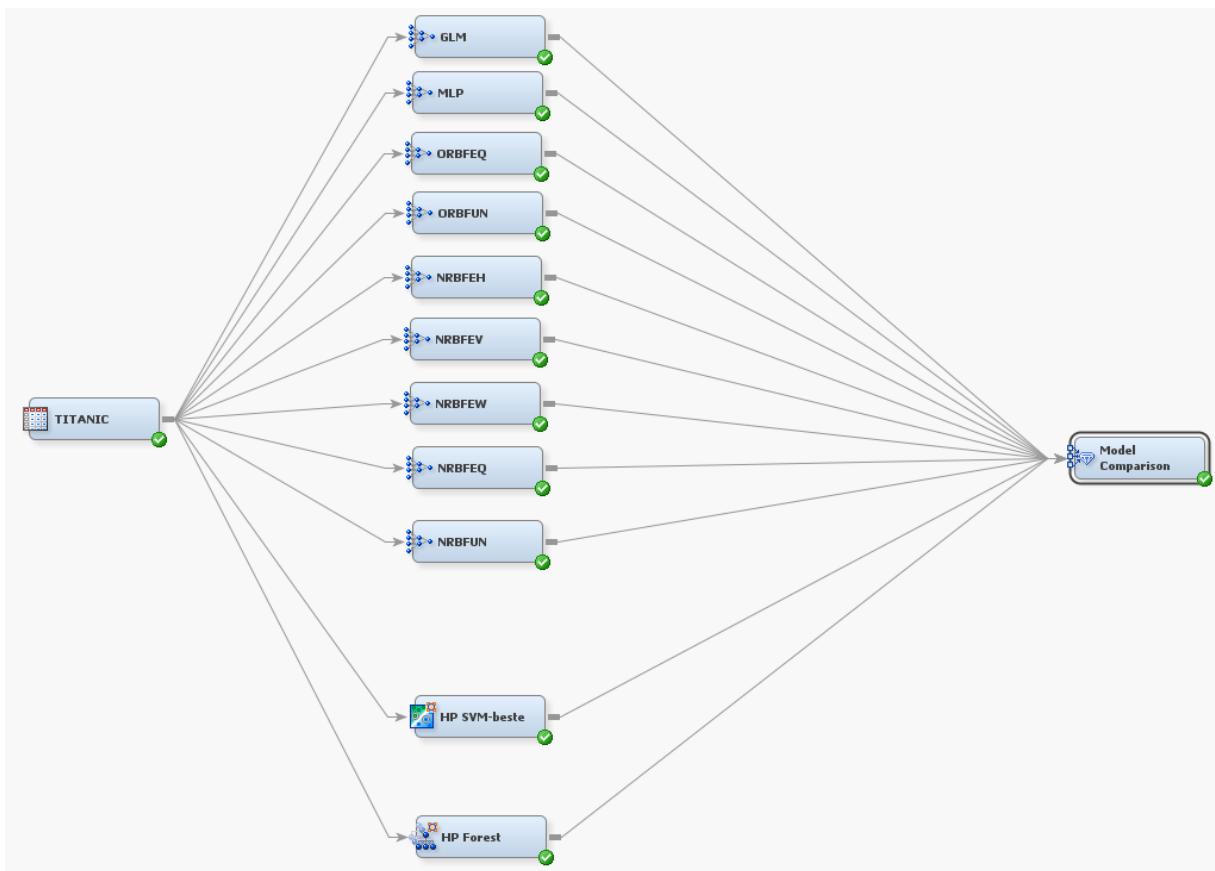
Selected Model	Model Node	Description	Train:	Average	Train:	
			Misclassification Rate	Squared Error	Roc Index	Train: Gini Coefficient
Y	HPDMForest	HP Forest	0.13216	0.09700	0.947	0.894
	Tree	Decision Tree	0.18335	0.13730	0.826	0.653
	HPSVM2	HP SVM-best	0.20015	0.16608	0.834	0.669
	Neural3	ORBF EQ	0.20168	0.15298	0.828	0.655
	Neural9	NRBF UN	0.20474	0.15416	0.808	0.615
	Neural8	NRBF EW	0.20550	0.15502	0.812	0.623
	Neural7	NRBF EW	0.20626	0.15233	0.823	0.647
	Neural6	NRBF EV	0.20932	0.15230	0.824	0.648
	Neural5	NRBF EH	0.21085	0.15121	0.826	0.652
	Neural4	ORBF UN	0.21390	0.15230	0.829	0.658
	Neural2	MLP	0.21543	0.15169	0.824	0.648
	Neural	GLM	0.23606	0.17629	0.783	0.567

Ferner werden natürlich die ROC-Kurven aller Modelle abgebildet.





Zum Schluss folgt die Abbildung der Aufgabe 4 a im Ganzen.



## 5.2 4 B

*Aufgabenstellung:*

Bauen Sie das Modell und die Analyse von **Mini-Batch Gradient Descent** aus Abschnitt 4-77 nach.

Bewerten Sie die Ergebnisse kritisch.

Nehmen Sie einige (kleine) Ergänzungen und Erweiterungen vor.

*Lösung:*

Es wurde schon einmal erwähnt, dass beim Training der künstlichen neuronalen Netze Trainingsfehler entstehen. Ziel ist es diesen Fehler zu minimieren. Somit hat man ein Optimierungsproblem ohne Nebenbedingungen, also unrestrictierte Optimierungsproblem. Hierfür gibt es mehrere Algorithmen die zur Verfügung stehen. Eines davon wäre die Gradientenabstiegsmethode [AF11].

Es gibt Batch und Stochastische Methoden für den Gradientenabstiegsverfahren. Einen Kompromiss aus beiden Methoden stellen Mini-Batches dar. Dabei werden die Trainingsdaten in kleinere Einheiten aufgeteilt und nach jeder dieser Einheiten die Gewichtungen aktualisiert. Näheres dazu kann in [DY17] nachgelesen werden.

In diesem Datensatz, genannt **develop**, geht es um eine Bank ein Versicherungsprodukt bei den Kunden vorstellen möchte. Dabei möchte die Bank ein zielgerichtetes Marketingkampagne starten und würde gerne vorher wissen welche der Kunden an diesem Produkt überhaupt interessiert sein würden. Der Datensatz hat 32.264 Kunden und somit auch Beobachtungen mit 47 Unabhängige Variablen. Diese Variablen beinhalten Informationen wie Wohnort oder auch andere Produkte die der Kunde noch benutzt oder benutzt hat. Die binäre Zielvariable heißt **Ins**. Diese soll als Indikator dafür dienen, ob ein Kunde dieses Renten Versicherungsprodukt kaufen würde oder nicht (1=ja und 0=nein). Die Daten werden zunächst bereinigt. Dabei werden falsch klassifizierten Variablen in richtige Merkmale eingeordnet und zwei der Variablen nicht berücksichtigt, da diese als nominale Variablen sind und als irrelevant deklariert wurden. Ein Ausschnitt der Daten sieht wie folgt aus.

The screenshot shows the 'Data Source Wizard -- Schritt 5 von 8 Column Metadata' window. On the left, there's a preview icon showing a database table with several rows and columns. On the right, a detailed table lists 47 columns with their properties:

Name	Role	Level	Report	Order	Drop	Lower Li
ATM	Input	Binary	No		No	
ATMamt	Input	Interval	No		No	
AcctAge	Input	Interval	No		No	
Age	Input	Interval	No		No	
Branch	Rejected	Nominal	No		No	
CC	Input	Binary	No		No	
CCBal	Input	Interval	No		No	
CCPurc	Input	Interval	No		No	
CD	Input	Binary	No		No	
CDBal	Input	Interval	No		No	
CRScore	Input	Interval	No		No	
CashBk	Input	Interval	No		No	
Checks	Input	Interval	No		No	
DDA	Input	Binary	No		No	
DDABal	Input	Interval	No		No	
Dep	Input	Interval	No		No	
DepAmt	Input	Interval	No		No	
DirDep	Input	Binary	No		No	
HMOwn	Input	Binary	No		No	
HMVal	Input	Interval	No		No	
ILS	Input	Binary	No		No	
ILSBal	Input	Interval	No		No	
IRA	Input	Binary	No		No	
IRABal	Input	Interval	No		No	
InArea	Input	Binary	No		No	
Income	Input	Interval	No		No	
Ins	Target	Binary	No		No	
Inv	Input	Binary	No		No	
InvBal	Input	Interval	No		No	
LOC	Input	Binary	No		No	
LOCBal	Input	Interval	No		No	
LORes	Input	Interval	No		No	
MM	Input	Binary	No		No	
MMBal	Input	Interval	No		No	
MMCred	Input	Interval	No		No	
MTG	Input	Binary	No		No	
MTGBal	Input	Interval	No		No	
Moved	Input	Binary	No		No	
NSF	Input	Binary	No		No	
NSFAmt	Input	Interval	No		No	
POS	Input	Interval	No		No	
POSAmt	Input	Interval	No		No	
Phone	Input	Interval	No		No	
Res	Rejected	Nominal	No		No	
SDB	Input	Binary	No		No	
Sav	Input	Binary	No		No	
SavBal	Input	Interval	No		No	
Teller	Input	Interval	No		No	

Um die genaue Schätzungen der Reaktionswahrscheinlichkeit zu erhalten wird nun die Gesamtheit der vorherigen Wahrscheinlichkeiten eingeben. Somit wird die positive Antwortrate von der vorherigen Kampagne, nur 2%, eingetippt.

### Data Source Wizard -- Schritt 6 von 10 Decision Configuration



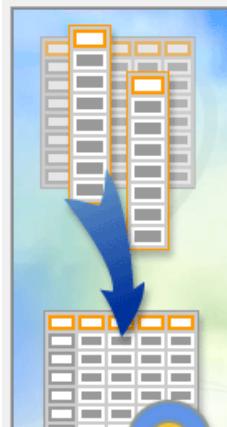
Decision Processing

Do you want to build models based on the values of the decisions ?

If you answer yes, you may enter information about the cost or profit of each possible decision, prior probability and cost function. The data will be scanned for the distributions of the target variables.

No       Yes

### Data Source Wizard -- Schritt 7 von 10 Decision Configuration



Prior Probabilities			
Targets	Prior Probabilities	Decisions	Decision Weights
Do you want to enter new prior probabilities?			
<input checked="" type="radio"/> Yes <input type="radio"/> No		Set Equal Prior	
Level	Count	Prior	Adjusted Prior
1	11175	0.3464	0.02
0	21089	0.6536	0.98

Insgesamt gibt es also 45 Unabhängige Variablen bzw. Regressoren und eine Abhängige Variable. Zusätzlich wurde mit Hilfe der Data Partition die Daten in 70% Trainings- und in 30% Validations- bzw. Testmengen eingeteilt.

### Data Source Wizard -- Schritt 10 von 10 Summary



Metadata Completed.

**Library:** EMLIB2  
**Data Source:** DEVELOP  
**Role:** Raw

Role	Level	Count
Input	Binary	17
Input	Interval	28
Rejected	Nominal	2
Target	Binary	1

Unten ist ein kleiner Ausschnitt von der SAS Code abgebildet. Dabei wird zunächst der Einfachheit halber Zielvariable und die intervallskalierten Regressoren als Makrovariablen gespeichert. Danach wurde das Mini Batch Verfahren mit Anzahl der hidden layer, batches und Epochen definiert. Diese wurde dann mit PROC NEURAL verknüpft und als letztes einmal laufen lassen.

```

Training Code
/*
 Program : c4s5dl
 Description: Mini-Batch Gradient Descent

options nodate nonumber nonotes;

%let target=ins;
/* generate interval input variable list */
proc contents data=&EM_IMPORT_DATA(drop=&target)
 out=temp(keep=name type) noplay;
run;
proc sql noplay;
 select name into: inputs separated by " "
 from temp
 where type=1;
quit;
%macro mini_batch(dsn=,nhidden=,nbatches=,nepochs=);
ods _all_ close;
data temp;
 set &dsn;
 random=ranuni(12345);
run;

```

Als Ergebnis liefert der SAS Code folgende fit statistics. Hierbei ist es erkennbar, dass es nur 17% Average Squared Error und 25% Misclassification Rate ergab.

#### Variable Summary

Role	Measurement	Frequency
	Level	Count
ID	INTERVAL	1
INPUT	BINARY	17
INPUT	INTERVAL	28
REJECTED	NOMINAL	2
TARGET	BINARY	1

#### Mini-Batch

Beob.	_ASE_	_AIC_	_SBC_	_MISC_
1	0.17106	23536.38	24700.00	0.25589

Unten ist zusätzlich die zugehörige Confusion Matrix für die Trainingsdaten abgebildet. Würde man nun die Accuracy berechnen, so ergibt es einen Wert von nur 0.7353. Dies bedeutet, dass hier 73,53% alle Interesse bzw. keine Interesse der Kunden richtig klassifiziert wurden.

### Mini-Batch

#### Die Prozedur FREQ

##### Table of F\_Ins by I\_Ins

F\_Ins(Von: Ins)      I\_Ins(Bis: Ins)

Häufigkeit				Summe
Prozent Zeile	0	1		
0	12385   2376   14761			
	83.90   16.10			
1	3403   4420   7823			
	43.50   56.50			
Summe	15788	6796	22584	

##### Statistiken für Tabelle von F\_Ins nach I\_Ins

Statistik	DF	Wert	Prob
Chi-Quadrat	1	3967.7973	<.0001
Likelihood-Ratio Chi-Quadrat	1	3887.2222	<.0001
Kontinuitätskorr. Chi-Quad.	1	3965.8769	<.0001
Mantel-Haenszel Chi-Quadrat	1	3967.6216	<.0001
Phi-Koeffizient		0.4192	
Kontingenzkoeffizient		0.3866	
Cramers V		0.4192	

##### Exakter Test von Fisher

Zelle (1,1) Häufigkeit (F)	12385
Linksseitige Pr <= F	1.0000
Rechtsseitige Pr >= F	<.0001
Tabellenwahrscheinlichkeit (P)	<.0001
Zweiseitige Pr <= P	<.0001

Stichprobengröße = 22584

Nun erkennt man Confusion Matrix für die Validierungsdaten. Die dazugehörige Accuracy beträgt 0.7375. Hier sollte noch kurz darauf aufmerksam gemacht werden, dass das Modell also beim 44.42% der Fälle wo es eine 1, also der Kunde würde den Renten Versicherungsprodukt kaufen, vorhersagt dieser Fall aber eigentlich nicht zutrifft, also INS=0=nein. Diese Zahl ist viel zu hoch wenn man wirklich an einer zielgerichtete Marketingkampagne interessiert ist bzw. durchführen möchte.

#### Mini-Batch

#### Die Prozedur FREQ

#### Table of F\_Ins by I\_Ins

F_Ins(Von: Ins)	I_Ins(Bis: Ins)

Häufigkeit				Summe
Prozent Zeile 0	1			
0   5276   1052   6328				
	83.38   16.62			
1   1489   1863   3352				
	44.42   55.58			
Summe   6765   2915   9680				

#### Statistiken für Tabelle von F\_Ins nach I\_Ins

Statistik	DF	Wert	Prob
<hr/>			
Chi-Quadrat	1	1579.9698	<.0001
Likelihood-Ratio Chi-Quadrat	1	1546.1605	<.0001
Kontinuitätskorr. Chi-Quad.	1	1578.1194	<.0001
Mantel-Haenszel Chi-Quadrat	1	1579.8066	<.0001
Phi-Koeffizient		0.4040	
Kontingenzkoeffizient		0.3746	
Cramers V		0.4040	

#### Exakter Test von Fisher

Zelle (1,1) Häufigkeit (F)	5276
Linksseitige Pr <= F	1.0000
Rechtsseitige Pr >= F	<.0001
Tabellenwahrscheinlichkeit (P)	<.0001
Zweiseitige Pr <= P	<.0001

Stichprobengröße = 9680

Dieses Modell wurde noch mit einem anderen neuronalen Netz, hier das MLP, verglichen. Dabei hat anscheinend die MLP Variante eine bessere Prognosen erstellt.

Fit Statistics														
Selected Model	Predessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion : Valid: Average Profit	Train: Average Squared Error	Train: Divisor for ASE	Train: Maximum Absolute Error	Train: Sum of Frequencies	Train: Root Average Squared Error	Train: Sum of Squared Errors	Train: Frequency of Classified Cases	Train: Misclassification Rate
Y	Neural MdlImp	Neural MdlImp	Neural Model Import	...Ins	...Ins	0.98003 0.98	0.16809 0.1710...	45168 45168	0.9913... 0.9776...	22584 22584	0.4099... 0.41359	7592.2... 7726.3...	22584 22584	0.34582 0.3463...

#### Fit Statistics

Model Selection based on Valid: Average Profit (\_VAPROF\_)

Selected Model	Model	Train:			Valid:			
		Valid: Average		Average	Train:	Average	Valid:	
		Model	Node	Description	Profit	Error	Rate	
Y	Neural MdlImp	Neural Model Import	Network	...Ins	0.98003 0.98000	0.16809 0.17106	0.34582 0.34640	0.17054 0.17391
								0.34576 0.34628

Data Role=Valid

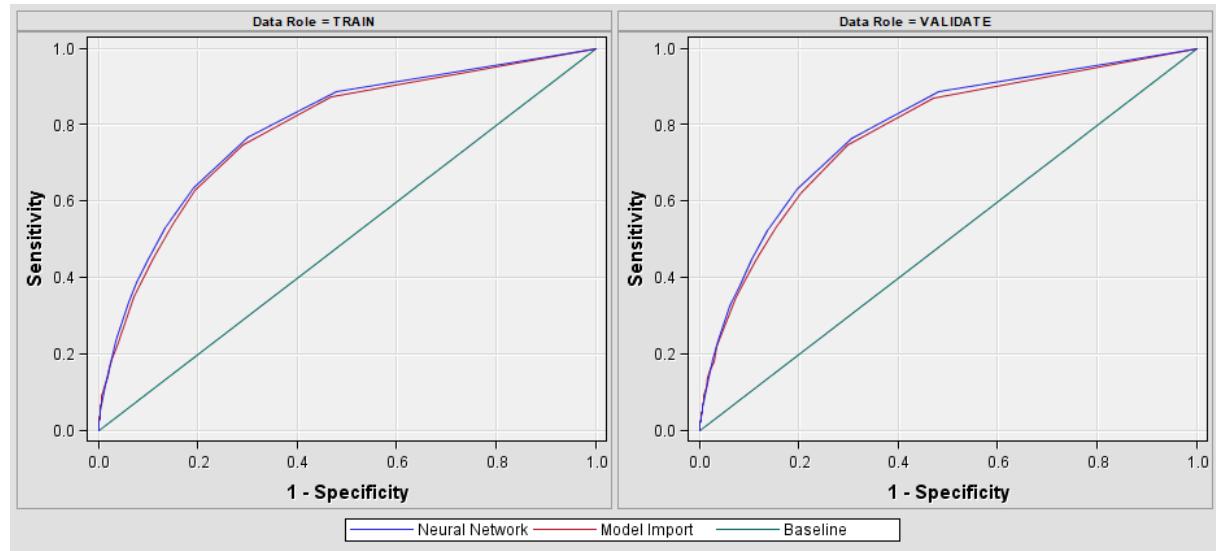
Statistics	Neural	MdlImp
Valid: Kolmogorov-Smirnov Statistic	0.46	0.45
Valid: Average Profit	0.98	0.98
Valid: Average Squared Error	0.17	0.17
Valid: Roc Index	0.80	0.79
Valid: Average Error Function	0.51	.
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.02	0.02
Valid: Cumulative Percent Captured Response	41.62	39.23
Valid: Percent Captured Response	14.92	12.98
Valid: Frequency of Classified Cases	.	9680.00
Valid: Divisor for ASE	19360.00	19360.00
Valid: Error Function	9952.25	.
Valid: Gain	316.15	292.13
Valid: Gini Coefficient	0.59	0.58
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.46	0.44
Valid: Kolmogorov-Smirnov Probability Cutoff	0.02	0.02
Valid: Cumulative Lift	4.16	3.92
Valid: Lift	2.99	2.59
Valid: Maximum Absolute Error	0.98	0.98
Valid: Misclassification Rate	0.35	0.35
Valid: Mean Squared Error	0.17	.
Valid: Sum of Frequencies	9680.00	9680.00
Valid: Total Profit	9486.69	9486.40
Valid: Root Average Squared Error	0.41	0.42
Valid: Cumulative Percent Response	8.32	7.84
Valid: Percent Response	5.98	5.19
Valid: Root Mean Squared Error	0.41	.
Valid: Sum of Squared Errors	3301.62	3366.87
Valid: Sum of Case Weights Times Freq	19360.00	.
Valid: Number of Wrong Classifications	3347.00	3352.00

Zusätzlich wird noch Classification Table mit ausgegeben. Hier wird nicht erklärbarerweise die Randsummen als FALSE NEGATIVE und als TRUE NEGATIVE vorgestellt. Zummindest von dem importierten Modell, also von Mini Batch Modell, wissen wir dass die Classification anders aussehen sollte! Eventuell ist diese Ausgabe nicht richtig.

Event Classification Table  
Model Selection based on Valid: Average Profit (\_VAPROF\_)

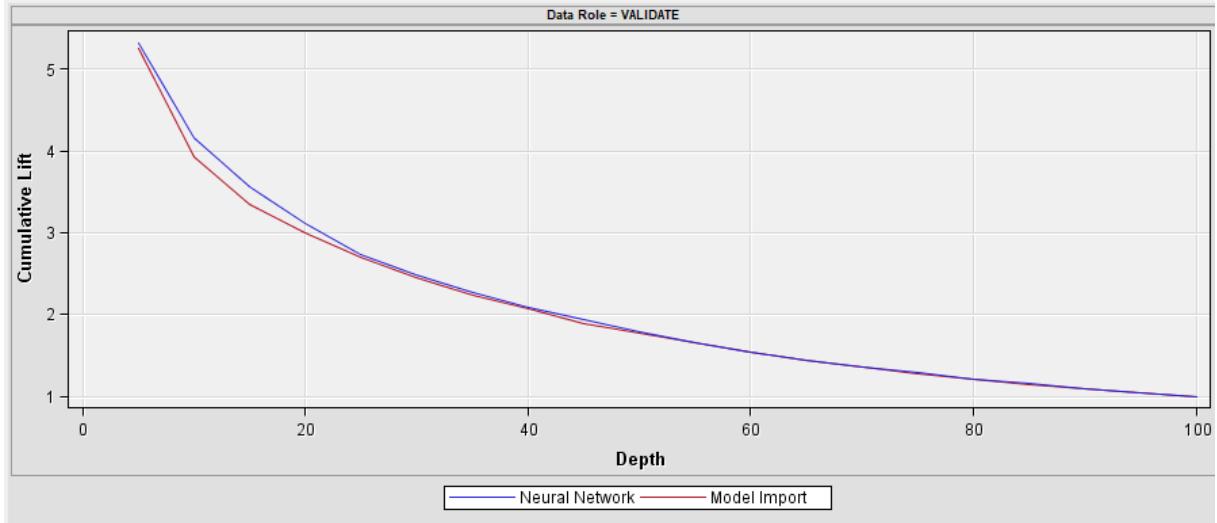
Model Node	Model Description	Data Role	Target	Label	False Negative	True Negative	False Positive	True Positive
MdlImp	Model Import	TRAIN	Ins		7823	14761	0	0
MdlImp	Model Import	VALIDATE	Ins		3352	6328	0	0
Neural	Neural Network	TRAIN	Ins		7809	14760	1	14
Neural	Neural Network	VALIDATE	Ins		3347	6328	.	5

Ferner werden die ROC-Kurven beider Modelle abgebildet.

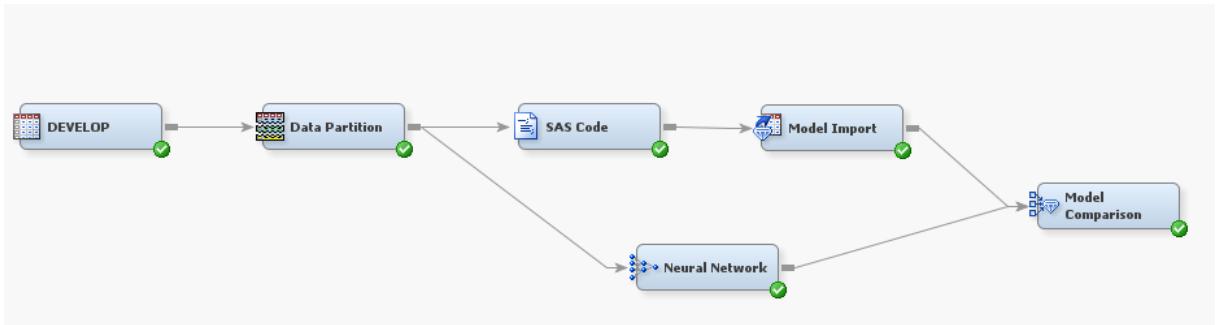


Laut der Aufgabenstellung soll mit den Validierungsdaten argumentiert werden. Hierbei ergab, dass der Mini Batch einen höheren Average Squared Error hat als bei der MLP neuronale Netze. Zusätzlich erkennt man anhand der ROC-Kurve, dass das MLP Modell besser als Vorhersagemodell dient als Mini Batch.

Anhand der kumulativen Lift Chart kann man nun sagen wie viele Kunden mit dem neuen Versicherungsprodukt bekannt gemacht werden müssen, damit diese sich dafür interessieren. Dabei zeigt der Chart visuell den Vorteil der Verwendung eines der Vorhersagemodelle und hilft bei der Frage, wie viel wahrscheinlicher kann ein Kunde auf das neue Produkt aufmerksam gemacht werden im Vergleich einer zufälligen Kontaktaufnahme der Kunden. Es ist hier zu erkennen, dass bei beiden Modellen etwa 60% der möglichen Neukunden erreicht wird, wenn die Top 20% der Kunden kontaktiert werden - da ein Lift von etwa 3 herrscht.



Abschließend folgt das Diagramm Abbildung der Aufgabe 4 B.



## 5.3 4 C

*Aufgabenstellung:*

Bauen Sie das Modell und die Analyse von **Implementing a Time Delay Neural Network** aus Abschnitt 5-13 nach.

Bewerten Sie die Ergebnisse kritisch.

Nehmen Sie einige (kleine) Ergänzungen und Erweiterungen vor.

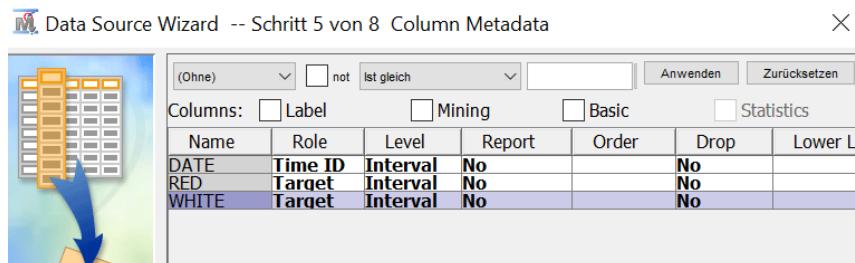
Zum Beispiel erscheinen mir die Kennzahlen (p-Werte) für <RED> nach der Transformation schlechter als vor der Transformation. Entweder ist die Transformation überflüssig oder eine andere wäre angebracht. Ich würde keine riesigen Effekte auf die finalen Ergebnisse der Analyse erwarten, aber <you never know>.

Optional: Analysieren Sie die Daten ggf. mit einem "besseren" Modell via R oder Python.

*Lösung:*

In diesem Teil wird nun ein Time Delay Neural Network (TDNN, zu deutsch Zeitverzögertes Neuronales Netz) konstruiert. Es ist ein mehrschichtiges künstliches neuronales Netz, das durch die Verwendung von Zeitfenstern über mehrere Eingaben in der Lage ist, zeitliche Abhängigkeiten von Eingaben zu verarbeiten. Dabei röhrt die Benennung aus der Verwendung von Verzögerungselementen (Delays) her, mit denen mehrere Zeitschritte parallel angelegt werden [WITD].

Es werden hier die Zeitreihendaten **wine**, welche Informationen über Verbrauchsstufen von Rot- und Weißwein in Australien zwischen Januar 1981 bis zum Dezember 1994 beinhaltet. Somit sind sowohl der Zeitraum, also DATE, als auch 2 Variablen, und zwar RED und WHITE, welche auch als TARGET deklariert werden, in den Daten vorhanden. Hierbei werden also Vorhersagen einmal über Rotwein und einmal über Weißwein einzeln gemacht.



Hier sollte noch darauf aufmerksam gemacht werden, dass die TARGET Variablen nicht binär sind, sondern dass es sich um kontinuierliche Variablen handelt. Da aber das default neuronale Netz, also MLP, die Annahme hat dass bei metrischen TARGET Variablen diese normalverteilt sind, muss nun diese Annahme durch den SAS CODE (später als Univariate unbenannt) PROC UNIVARIATE für die beiden Variablen überprüft werden.

```
Training Code
/*
Program : c5sldla
Description: assess the normality of the target variables
*/
proc univariate data=&EM_IMPORT_DATA normal;
 var %EM_TARGET;
run;
```

Der folgende Test auf Normalverteilung für das Rot- und Weißwein weißen darauf hin, dass weder RED noch WHITE normalverteilt sind. Dabei wurde mit mehreren Teststatistiken gearbeitet, bei denen es mit der Hypothese  $H_0$  auf die Normalverteilung mit einem Signifikanzniveau  $\alpha = 5\%$  getestet wurde.

```
Die Prozedur UNIVARIATE
Variable: RED

Tests auf Normalverteilung

Test --Statistik--- -----p-Wert-----
Shapiro-Wilk W 0.973467 Pr < W 0.0016
Kolmogorov-Smirnov D 0.089131 Pr > D <0.0100
Cramer-von Mises W-Sq 0.243287 Pr > W-Sq <0.0050
Anderson-Darling A-Sq 1.459864 Pr > A-Sq <0.0050
```

```
Die Prozedur UNIVARIATE
Variable: WHITE

Tests auf Normalverteilung

Test --Statistik--- -----p-Wert-----
Shapiro-Wilk W 0.956433 Pr < W <0.0001
Kolmogorov-Smirnov D 0.091811 Pr > D <0.0100
Cramer-von Mises W-Sq 0.251973 Pr > W-Sq <0.0050
Anderson-Darling A-Sq 1.640071 Pr > A-Sq <0.0050
```

Zusätzlich werden nun durch den SAS CODE (später als Tap Delay unbenannt) die Variablen zunächst erst mit Hilfe der Logarithmus transformiert - da ja die Normalverteilung Annahme nicht erfüllt war - und dann werden die lagged (Lags = die Zeitdifferenzen, mehr Informationen dazu konnte man sich bei Herrn Becker in der Vorlesung zur Zeitreihenanalyse aneignen) inputs generiert. Zudem wurden die Daten durch CUTOFF in Trainings- und Validationsmengen aufgeteilt. Somit gehören von Januar 1981 bis zum letzten Tag von Dezember 1991 der Trainings- und ab ersten Januar 1992 bis zum Dezember 1994 der Validationsdatensätze an.

```
Training Code
Description: Generate NTAPS time lags and data split.
----- */

%macro tap_delay(ntaps=,cutoff=);
 data &EM_EXPORT_TRAIN &EM_EXPORT_VALIDATE;
 set &EM_IMPORT_DATA;
 %do i=1 %to &ntaps;
 if red ge 0 then red=log(red+1);
 if white ge 0 then white=log(white+1);
 rai=lag&i(red);
 wai=lag&i(white);
 %end;
 if _n_ > &ntaps then do;
 if date < &cutoff then output &EM_EXPORT_TRAIN;
 else output &EM_EXPORT_VALIDATE;
 end;
 run;
%end tap_delay;

*tap_delay(ntaps=12,cutoff='1jan92'd)
```

Somit werden nun 12 Zeitdifferenz Werte (lagged values) für Rot- und für Weißwein als Inputs erstellt. Da all diese Variablen in das Modell mit eingebaut werden um eine Vorhersage zu machen, wird das neuronale Netz nun eine vector autoregression (Vektorautoregressive Modelle auch kurz VAR-Modelle) durchgeführt.

**M Variables - Neural**

Name	Use	Report	Role	Level
RED	<b>Yes</b>	No	Target	Interval
WHITE	<b>Yes</b>	No	Target	Interval
r1	<b>Default</b>	No	Input	Interval
r10	<b>Default</b>	No	Input	Interval
r11	<b>Default</b>	No	Input	Interval
r12	<b>Default</b>	No	Input	Interval
r2	<b>Default</b>	No	Input	Interval
r3	<b>Default</b>	No	Input	Interval
r4	<b>Default</b>	No	Input	Interval
r5	<b>Default</b>	No	Input	Interval
r6	<b>Default</b>	No	Input	Interval
r7	<b>Default</b>	No	Input	Interval
r8	<b>Default</b>	No	Input	Interval
r9	<b>Default</b>	No	Input	Interval
w1	<b>Default</b>	No	Input	Interval
w10	<b>Default</b>	No	Input	Interval
w11	<b>Default</b>	No	Input	Interval
w12	<b>Default</b>	No	Input	Interval
w2	<b>Default</b>	No	Input	Interval
w3	<b>Default</b>	No	Input	Interval
w4	<b>Default</b>	No	Input	Interval
w5	<b>Default</b>	No	Input	Interval
w6	<b>Default</b>	No	Input	Interval
w7	<b>Default</b>	No	Input	Interval
w8	<b>Default</b>	No	Input	Interval
w9	<b>Default</b>	No	Input	Interval

Es wird ein MLP Netz mit nur einem hidden unit durchgeführt.

**M Network**

Property	Value
<b>General</b>	
Node ID	Neural
Imported Data	...
Exported Data	...
Notes	...
<b>Train</b>	
Variables	...
Continue Training	No
Network	...
Optimization	...
Initialization Seed	12345
Model Selection	CProfit/Loss
Suppress Output	No
<b>Score</b>	
Hidden Units	No
Residuals	Yes
Standardization	No

Property	Value
Architecture	Multilayer Perceptron
Direct Connection	Yes
Number of Hidden Units	1
Randomization Distribution	Normal
Randomization Center	0.0
Randomization Scale	0.1
Input Standardization	Standard Deviation
Hidden Layer Combination Func	Default
Hidden Layer Activation Func	Default
Hidden Bias	Yes
Target Layer Combination Func	Default
Target Layer Activation Func	Default

**Number of Hidden Units**

Man möchte nun die Vorhersage der Zeitreihe anhand einer Plot veranschaulichen. Hierfür wird nun wieder ein SAS CODE (später in Plot unbenannt) verwendet. Es wurde auch noch ein Gütemaß, und zwar MAPE = mean absolute percentage error, als Ausgabe gefordert. Die mittlere absolute prozentuale Abweichung wird mit  $a_t$  echte Wert und  $f_t$  die vorhergesagte Wert im Zeitpunkt  $t$  definiert als:

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^n \left| \frac{a_t - f_t}{a_t} \right|$$

**Training Code**

```
Description: plot the training and validation time series estimates
 and calculate the MAPE.

----- */

options nodate nonumber;
options reset=all device=actximg;

*macro plotseries(dsn=,target=,id=);
 data _null_;
 call symputx('plot',trim(scan("&dsn",2,'_')));
 call symputx('predicted',%str("p_&target"));
 call symputx('residual',%str("r_&target"));
 run;
 title1 h=1.5 f=swissb "splot Data Set";
 proc gplot data=&dsn;
 symbol1 c=bl i=join v=none;
 symbol2 c=r i=join v=none l=3;
 legend1 label=none mode=share position=(top center inside);
 axis1 label=none;
 format date monyy6.;
 plot &target*time=1 &predicted*time=2/overlay frame legend=legend1
 vaxis=axis1 description=&id;
 run;
 quit;
 data stats;

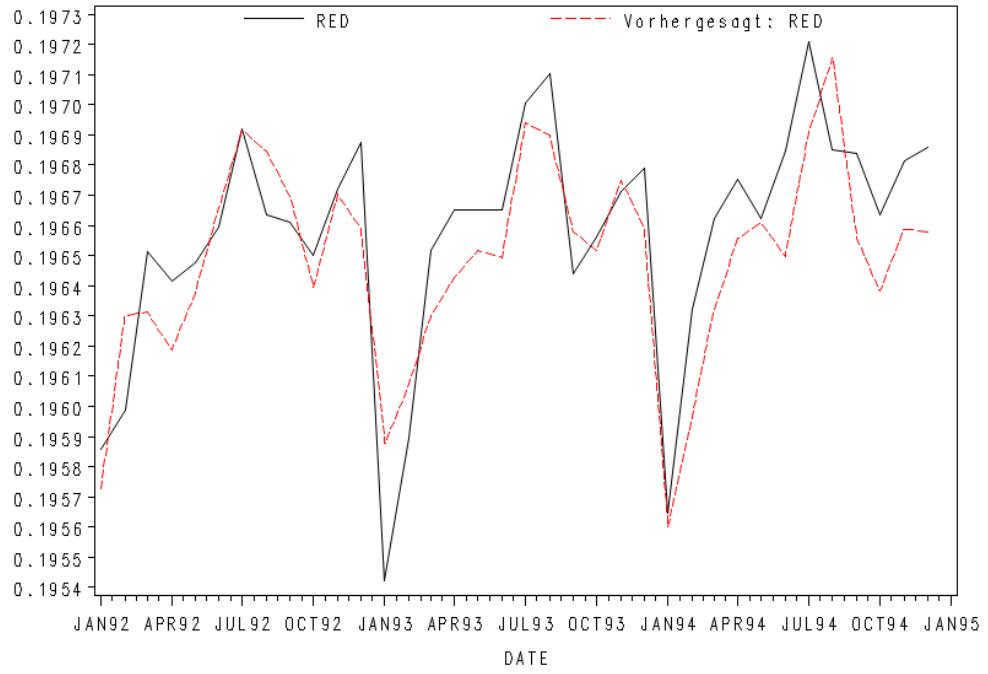

```

MAPE misst die Größe der Abweichung im Vergleich zur Größe der Daten als Prozentwert. Hier ist es erkennbar, dass beim Training- und bei Validationsmengen die Prognosen fast perfekt waren, da hier sogar ein MAPE weniger als 1%, und das dass die Vorhersage beim Weißwein besser war als beim Rotwein.

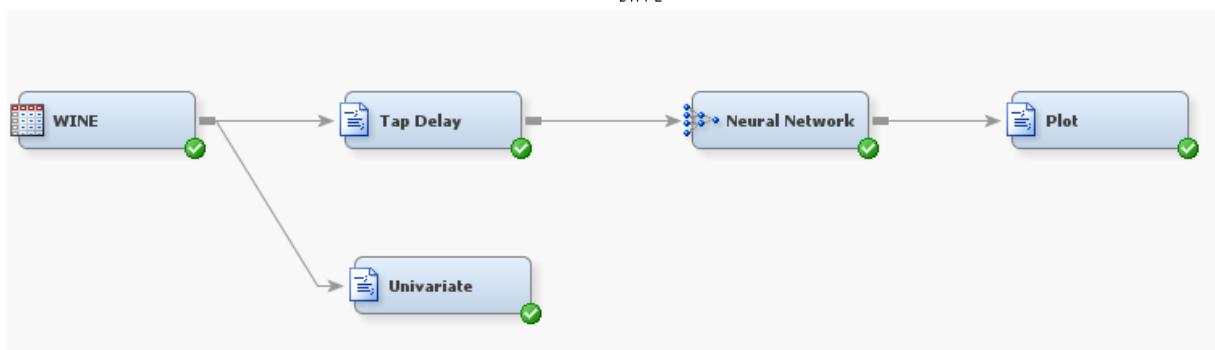
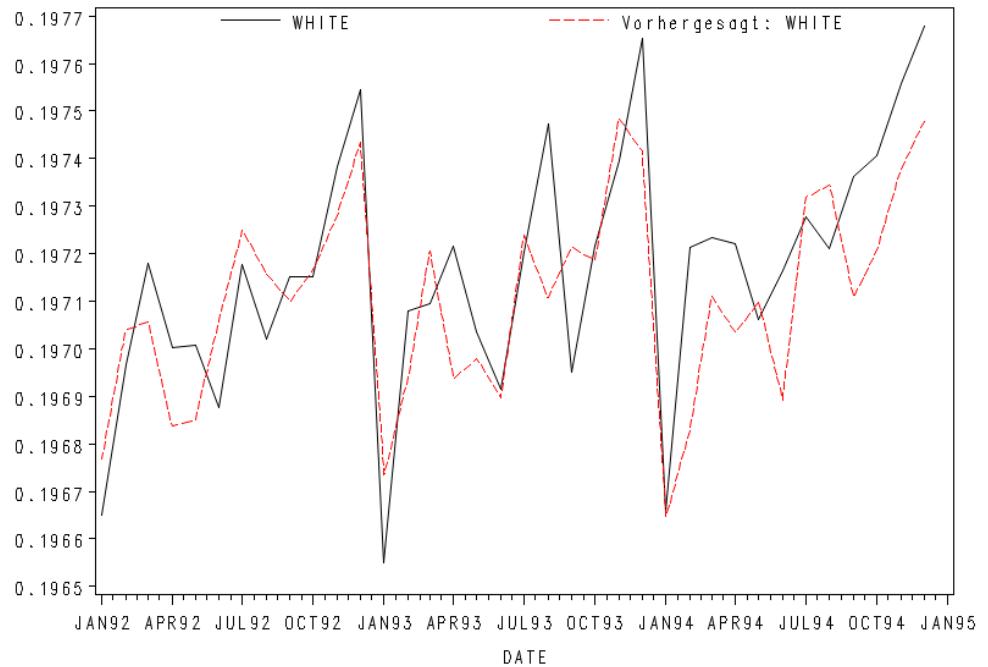
TRAIN Data Set	TRAIN Data Set
mape	mape
0.0517	0.0423
VALIDATE Data Set	VALIDATE Data Set
mape	mape
0.0949	0.0724

Anhand der Plots für RED und WHITE erkennt man, dass die Vorhersagen in den späteren Jahren, ab circa Januar 1994, nicht so gut wie am Anfang der Validation getroffen wurden. Dies führt daher, da es probiert wurde mit den Daten aus der Vergangenheit (von 1981 bis 1992) nicht nur die nähre Zukunft zu vorherzusagen.

**VALIDATE Data Set**



**VALIDATE Data Set**



## 5.4 4 D

Aufgabenstellung:

Wie die Aufgabe 4 A - 4 C für Comparing One- and Two-Layer Networks in 6-16.  
Verwenden Sie insbesondere den sogenannten LINK Graph.

Wenn Sie an das Ende des Miner Flusses einen Reporter Knoten setzen erhalten Sie alles/vieles wahlweise als pdf oder rtf. Daraus können Auszüge verwendet werden. Bitte nach Möglichkeit keinen vollständigen Report in die Ausarbeitung (Explosion der Seitenzahl).

Lösung:

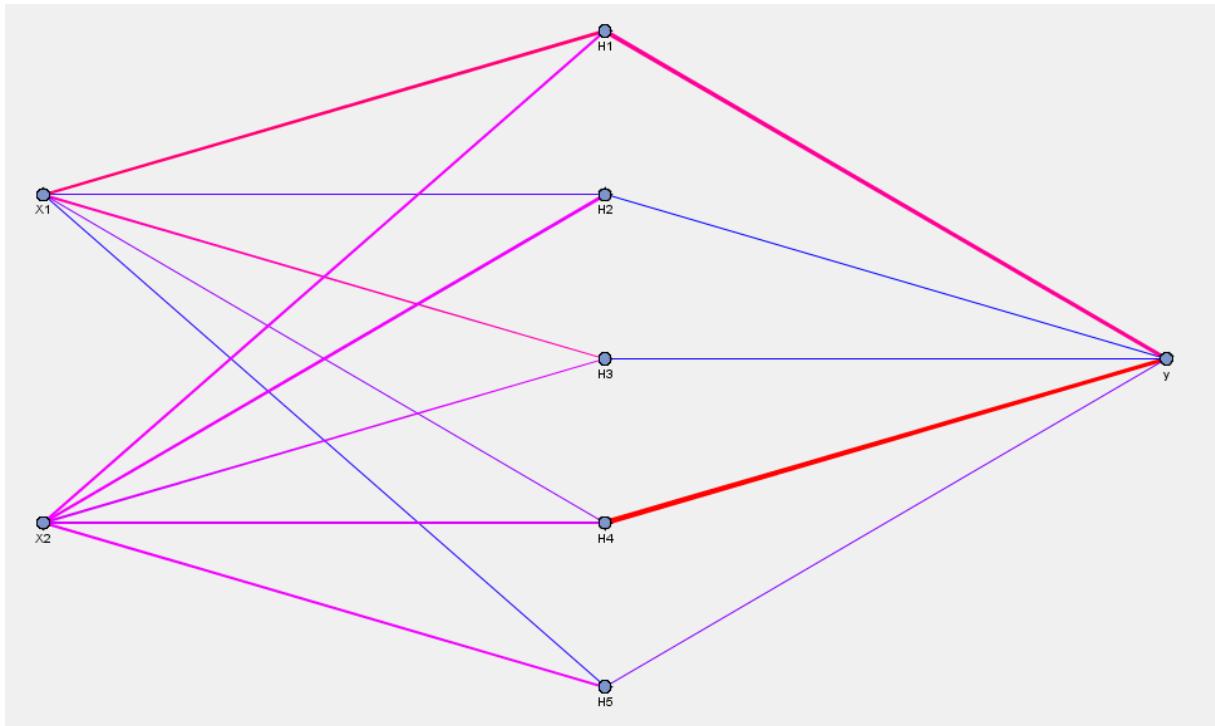
In diesem Aufgabenteil möchte man nun mit kontinuierlichen abhängige (Y) und unabhängigen (X1 und X2) Variablen arbeiten. Hierfür werden die Daten **hill\_valey** verwendet. Es werden im Folgenden 1-schichtige und 2-schichtige neuronale Netze an die Daten als Prognose Modell angepasst. Man geht davon aus, dass die 2-schichtigen Netze bessere Vorhersagen produzieren werden.

Name	Role	Level	Report	Order	Drop	Low
X1	Input	Interval	No		No	
X2	Input	Interval	No		No	
Y	Target	Interval	No		No	

Zudem werden nun nicht die normale neuronale Netze sondern die high performance neuronale Netze der SAS EMiner verwendet und wieder mit MLP Netze gearbeitet. Es werden noch die folgende Angaben bei der HPNN gemacht.

Property	Value
<b>General</b>	
Node ID	HPNNA
Imported Data	...
Exported Data	...
Notes	...
<b>Train</b>	
Variables	...
Use Inverse Priors	No
Create Validation	No
<b>Network Options</b>	
Input Standardization	Range
Architecture	One Layer
Number of Hidden Neurons	5
Number of Hidden Layers	3
Hidden Layer Options	...
Direct Connections	No
Target Standardization	Range
Target Activation Function	Identity
Target Error Function	Normal
Number of Tries	5
Maximum Iterations	1000
Use Missing as Level	No
Report	

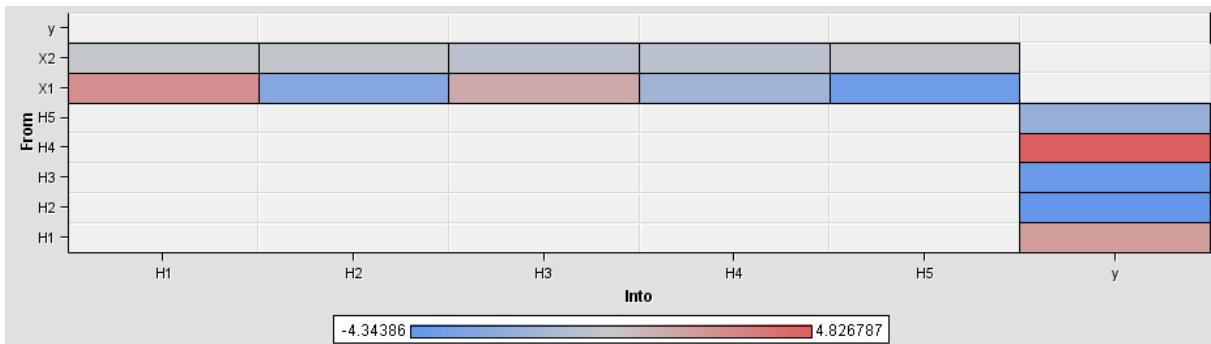
Man kann sogar den Graph dazu in EMiner ausgeben lassen. Wie schon bei den Angaben sieht man dass es 2 input Variablen, X1 und X2, gibt und fünf hidden units in single hidden layer, da beim ersten das 1-schichtige Netz dargestellt werden soll, und ein target unit (die Y Variable). Es sollte noch angemerkt werden, dass die Breite und die Farbe der Linien die resultierende Größe des Gewichts für jeweilige Verbindung angibt.



Zudem können natürlich noch die Fit Statistics hierzu ausgegeben werden. Es fällt auf, dass die Average Squared Error ein Wert von 0.001703 hat, was sehr gering und somit auch sehr gut ist.

Fit Statistics						
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
V	Target	ASE	Average Squa...	0.001703	.	.
V	Target	DIV	Divisor for ASE	2046	.	.
V	Target	MAX	Maximum Abs...	0.136712	.	.
V	Target	NOBS	Sum of Frequ...	2046	.	.
V	Target	RASE	Root Average ...	0.041264	.	.
V	Target	SSE	Sum of Squar...	3.483809	.	.

Man kann auch die Gewichte als Abbildung ausgeben lassen. Man erkennt zum Beispiel beim H4 (also FROM) geht es eine sehr hohe Gewichtung (4.826787) zu den target output Y (durch INTO).



Es können noch andere Plots wie zum Beispiel Mittelwert der Prognosewerte oder auch Iterationen und dazugehörige Average Training Error ausgeben lassen. Auf diese Plots wird hier nicht weiter drauf eingegangen.

Man kann auch die Informationen, die in den Plots grafisch dargestellt wurden, als Output ausgeben lassen. Hierbei verwendet HPNEURAL als Optimierungstechnik den BFGS-Algorithmus mit begrenztem Speicher. Das Broyden-Fletcher-Goldfarb-Shanno (BFGS) Verfahren ist ein numerisches Verfahren zur Lösung von nichtlinearen Optimierungsproblemen. Mit begrenztem Speicher approximiert die BFGS, unter Verwendung von nur wenigen Vektoren, das inverse Hesse Matrix.

#### Die Prozedur HPNEURAL

##### Performance-Informationen

Execution Mode	Single-Machine
Number of Threads	4

##### Datenzugriffsinformationen

Daten	Engine	Rolle	Pfad
EMWS4.IDS_DATA	V9	Input	Auf Client

##### Modellinformationen

Data Source	EMWS4.IDS_DATA
Architecture	MLP
Number of Input Variables	2
Number of Hidden Layers	1
Number of Hidden Neurons	5
Number of Target Variables	1
Number of Weights	21
Optimization Technique	Limited Memory BFGS

Number of Observations Read	2046
Number of Observations Used	2046
Number Used for Training	2046
Number Used for Validation	0

Als Ausgabe erhält man auch die Trainingseinzelheiten, die hier aber nicht weiter beachtet werden. Betrachtet man die Fit Statistics und den Average Squared Error (weniger als 0.01), so ist es erkennbar dass die HPNEURAL Netze enorme Leistungen erbringen.

##### Fit Statistics

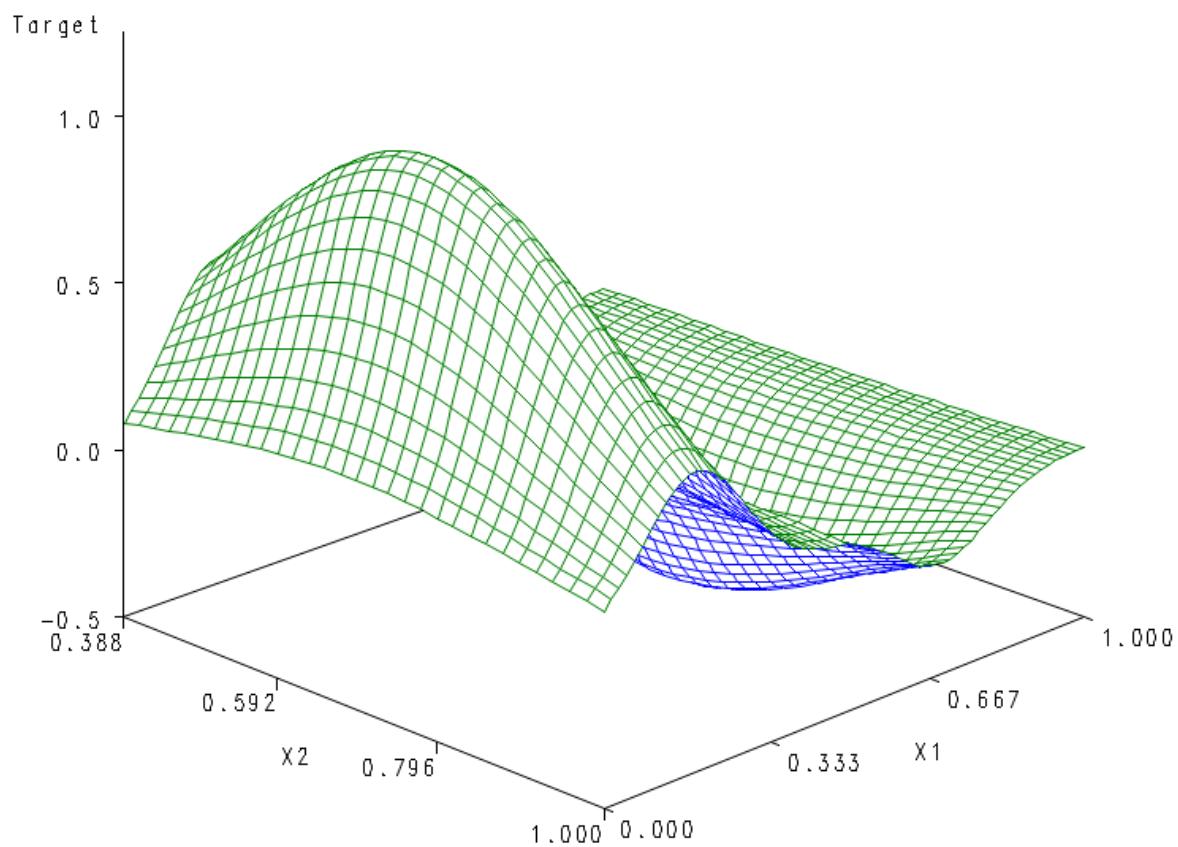
Target=y Target Label=Target

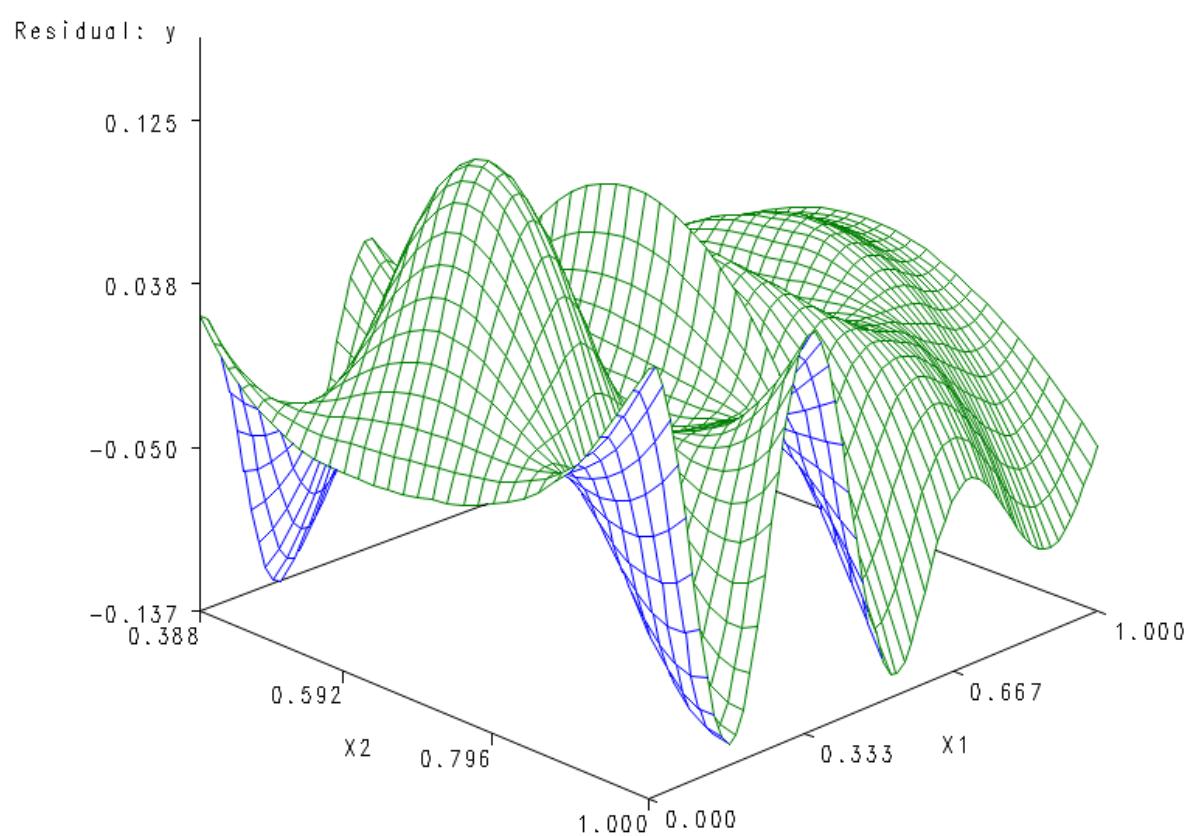
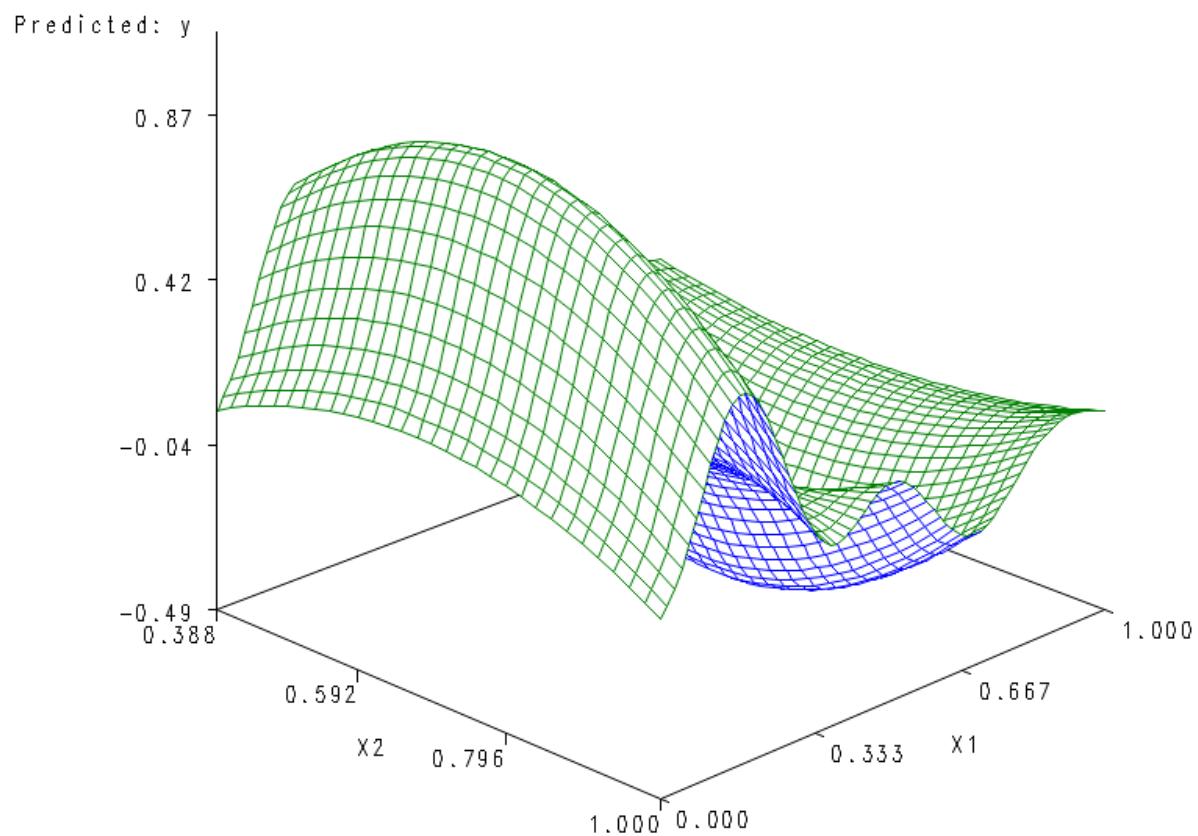
Fit Statistics	Statistics Label	Train
_ASE_	Average Squared Error	0.00
_DIV_	Divisor for ASE	2046.00
_MAX_	Maximum Absolute Error	0.14
_NOBS_	Sum of Frequencies	2046.00
_RASE_	Root Average Squared Error	0.04
_SSE_	Sum of Squared Errors	3.48

Es können natürlich noch weitere Ausgaben wie Score Rankings oder Score Distribution betrachtet werden, welche hier nicht weiter ausgeführt werden. Nun werden mit Hilfe der SAS Code zunächst die Verteilung der echten Werte als Plot 1, dann die Prognose als Plot 2 und zuletzt die Residuen als Plot 3 dargestellt.

```
Training Code
options nodate nonumber;
goptions reset=all device=actximg;

proc g3d data=&em_import_data;
plot x1*x2= y/rotate=315 description="plot 1";
plot x1*x2=p_y/rotate=315 description="plot 2";
plot x1*x2=r_y/rotate=315 description="plot 3";
run;
quit;
```

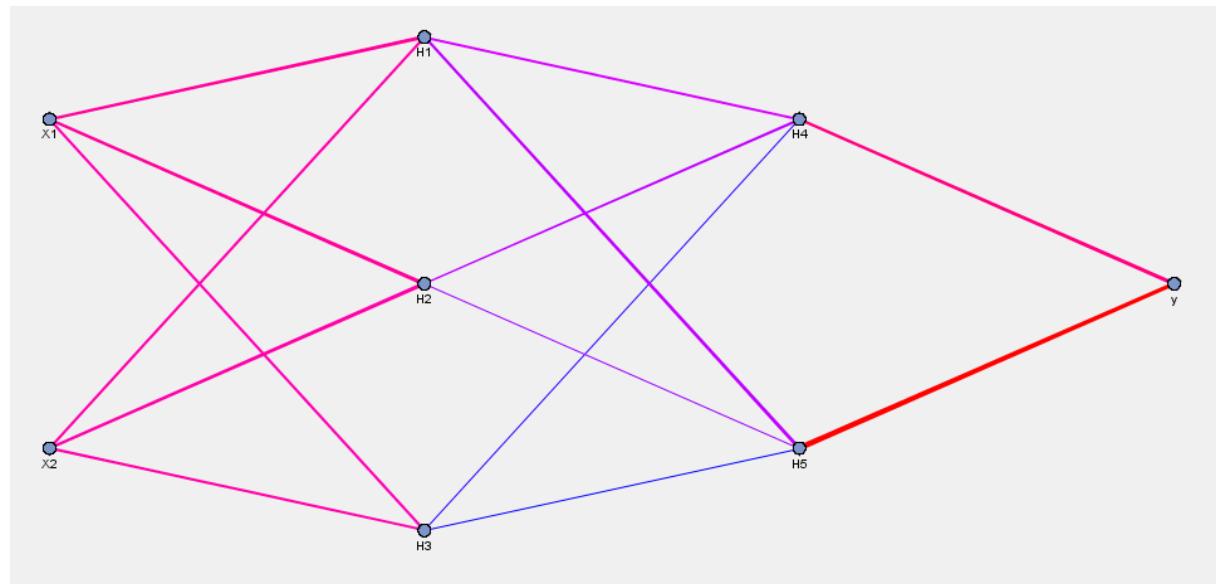




Zwar gibt die single-hidden-layer Netz sehr gute Prognose und erfasst auch genau den Hügel und den angrenzenden Tal, aber bei der Prognose Plot sieht man einen Hügel bzw. Falte der nicht in den eigentlichen Daten ist. Daher wird nun das ganze mit two-layer wiederholt und es wird erhofft bessere Prognosen zu erhalten.

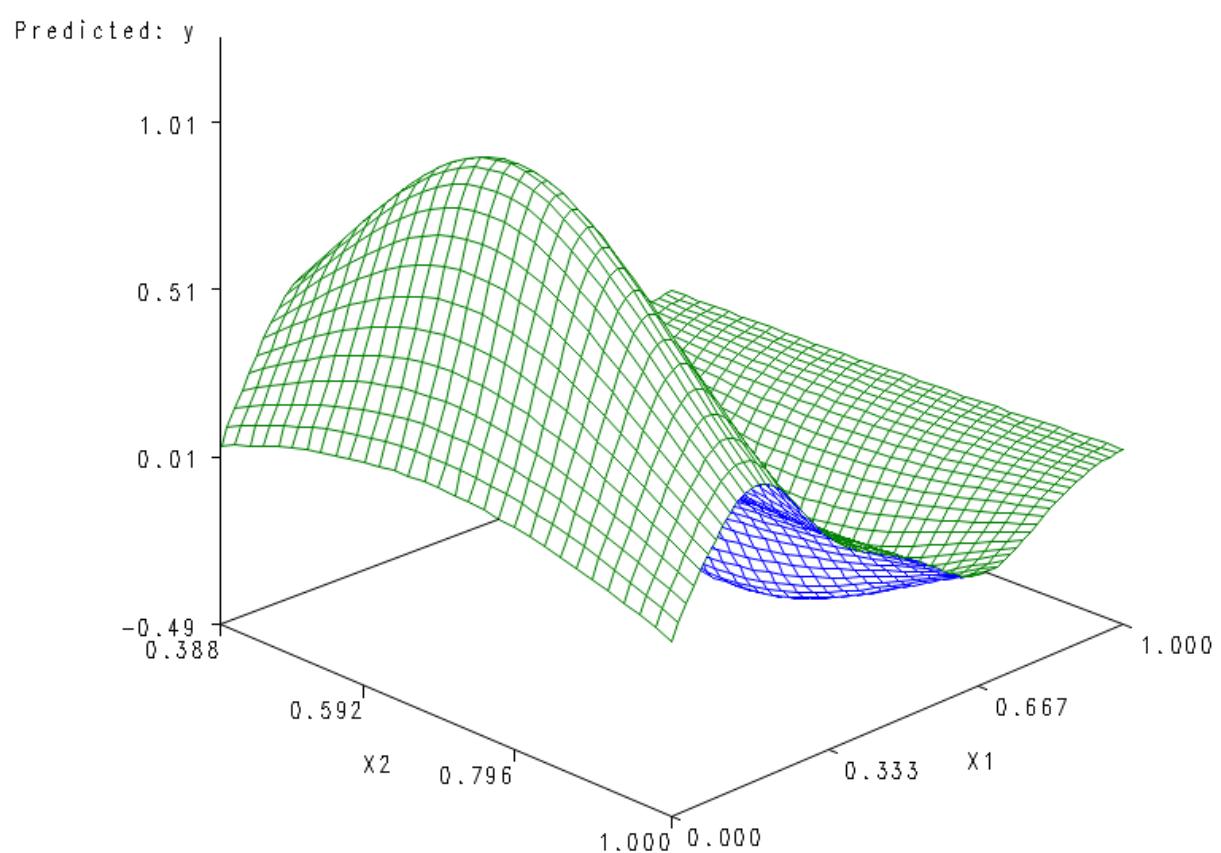
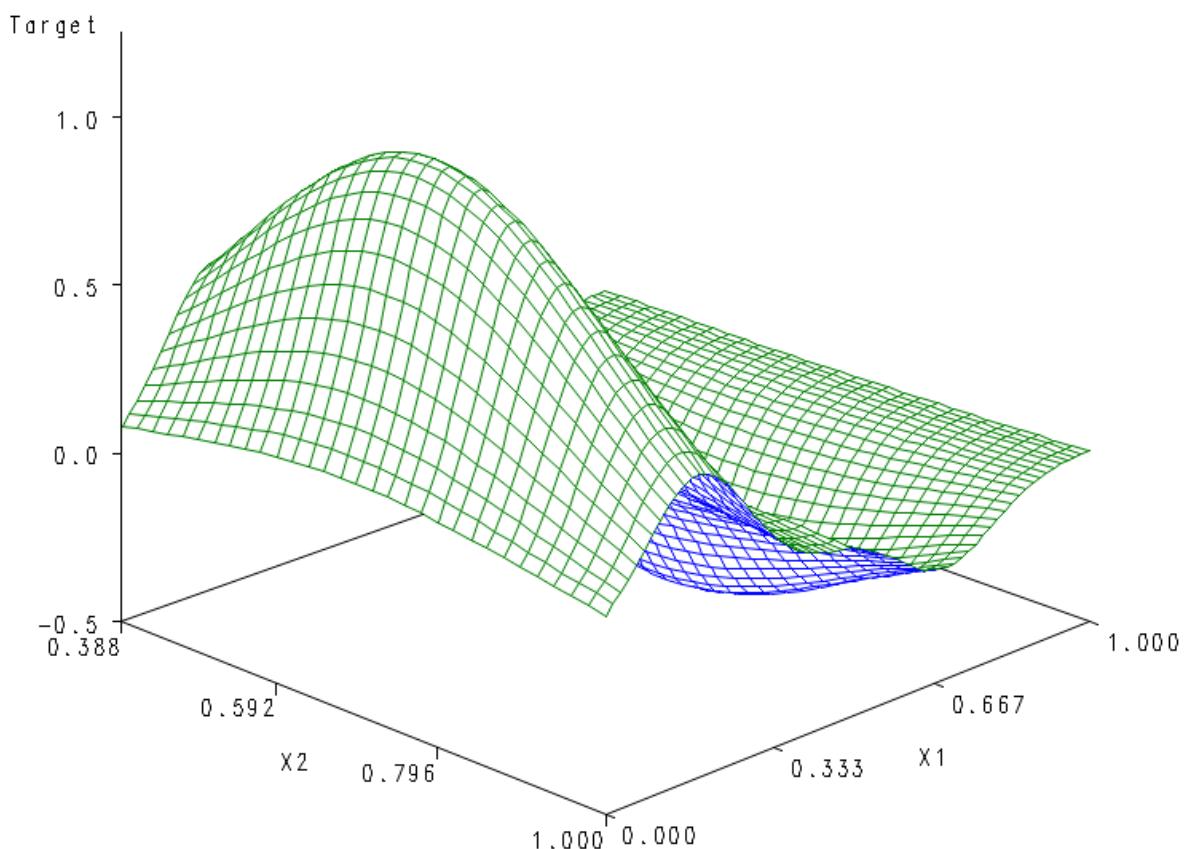
Property	Value
<b>General</b>	
Node ID	HPNNA
Imported Data	...
Exported Data	...
Notes	...
<b>Train</b>	
Variables	
Use Inverse Priors	No
Create Validation	No
<b>Network Options</b>	
Input Standardization	Range
Architecture	One Layer
Number of Hidden Neurons	User-Defined
Number of Hidden Layers	Logistic
Hidden Layer Options	One Layer
Direct Connections	One Layer with Direct
Target Standardization	One Layer with Direct
Target Activation Function	Two Layers
Target Error Function	Two Layers with Direct
Number of Tries	Two Layers with Direct
Maximum Iterations	1000

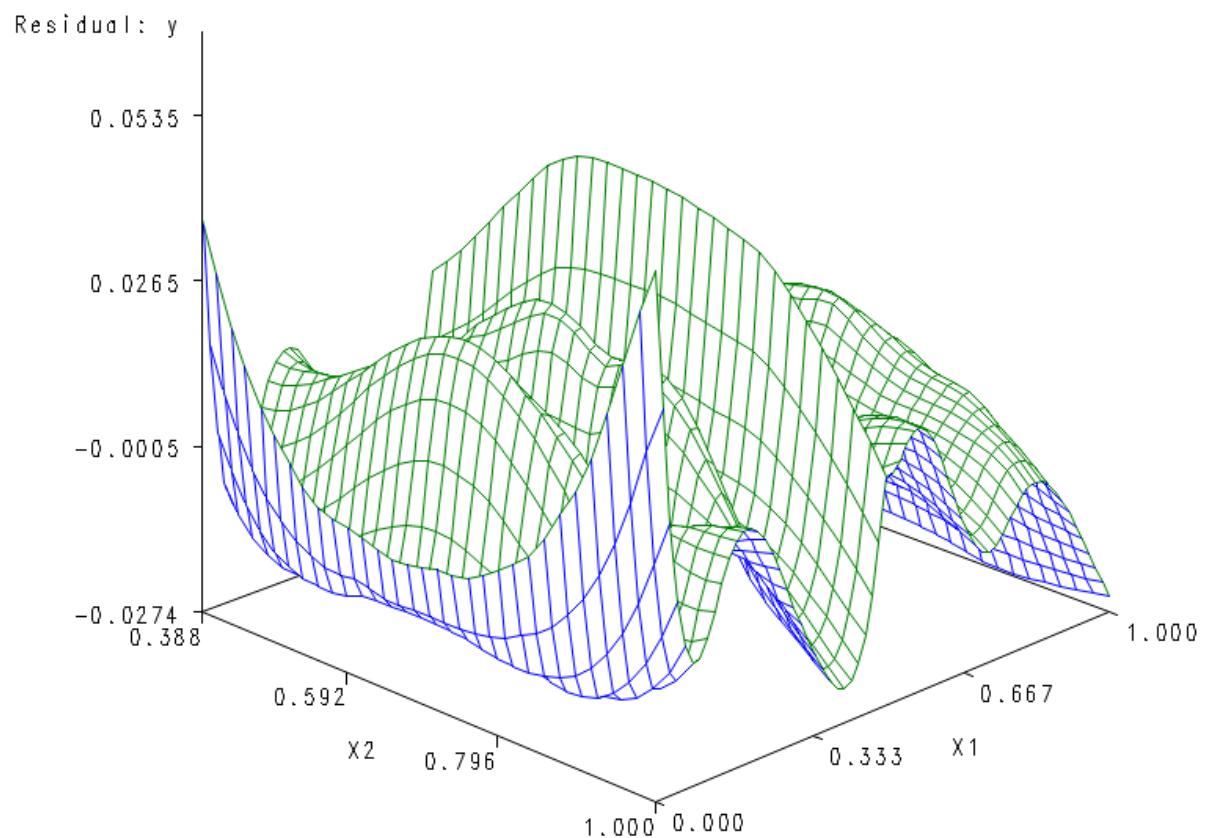
Die hidden units wurden auf zwei Schichten aufgeteilt.



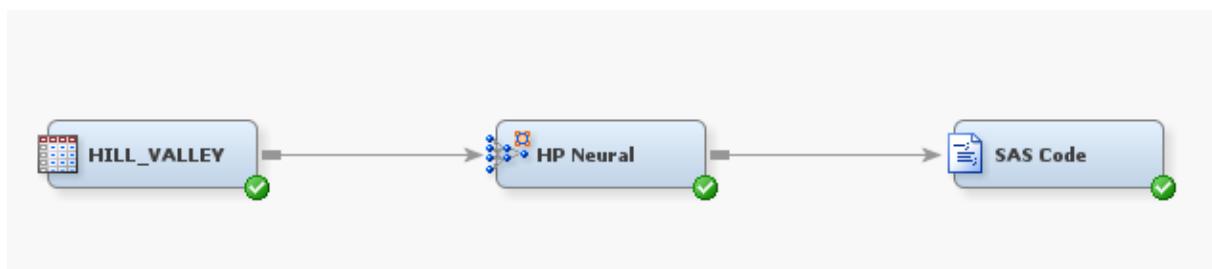
Ein ASE Wert von 0.0001414 ist bei weitem besser als beim single-layer mit 0.001703 ASE.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
V	Target	ASE	Average Squa...	.0001414	.	.
V	Target	DIV	Divisor for ASE	2046	.	.
V	Target	MAX	Maximum Abs...	0.053493	.	.
V	Target	NOBS	Sum of Frequ...	2046	.	.
V	Target	RASE	Root Average ...	0.01189	.	.
V	Target	SSE	Sum of Squar...	0.289247	.	.





Nun kann man sehen, dass die Falte die vorher bei single-layer vorhanden war, nun vollständig verschwunden ist. Es sollte beachtet werden, dass dies erreicht wurde obwohl weder die Anzahl der verwendeten hidden units noch die Anzahl der angeforderten Trainings-Iterationen erhöht wurden.



## 6 Aufgabe 5

---

*Aufgabenstellung:*

- A Ignorieren Sie zunächst die \*.ipynb Versionen. Diese sind noch nicht rundum aktualisiert. Alle \*.py Versionen sind geprüft und laufen ohne Fehler unter Keras 2.\* und TensorFlow. Ergänzen Sie die Skripte dahingehend, dass die wichtigsten Kennzahlen und Grafiken (für \_train und \_test Confusion Matrix, ROC-Chart, verschiedene Metrics (Accuracy, ...), sowie Outputs zur Kontrolle bei simulierten Daten **bei allen** erzeugt und ausgegeben werden.
- B Erweitern Sie die Skripte dahingehend, dass, wo sinnvoll, mehrere Modelle verwendet und verglichen werden, also z.B. durch ROC-Charts mit mehreren ROC-Kurven (entsprechend den Modellen). Dies geht in Richtung der Funktionalität des ASSESSMENT- Model Comparison Knoten aus dem SAS Enterprise Miner.
- C Erstellen Sie Versionen mit Rekurrenten Neuronalen Netzen (RNN) an Stelle von LSTM-Netzen, ggf. andere Varianten, die über model.add(XXX(...)) leicht zugänglich sind.
- D Variieren Sie Größe der Netze (Anzahl Schichten|Anzahl Neuronen), sowie Parameter wie activation, loss, optimizer, learning rate, epochs, batch\_size, etc pp ... und schauen bzw. dokumentieren Sie, was die Auswirkungen sind.

**Schnell (Rechenzeit/Laufzeit) stabile Ergebnisse mit sehr guten Kennzahlen zu erzielen ( \_test == \_valid geht über \_train ) ist in der Regel das primäre ZIEL.**

Wählen Sie **mindestens eines** der oben genannten größeren Daten/Probleme aus und versuchen Sie auf jeweils identischem Datenbestand.

— die Ergebnisse von SAS 2017 (d.h. ohne SAS Deep Learning Toolkit), also am leichtesten und schnellsten die besten Ergebnisse, die über verschiedene Modelle via Enterprise Miner erzielt und via ASSESSMENT- Model Comparison dargestellt werden können, mit PYTHON, genauer mit PYTHON (Keras, TensorFlow) zu verbessern, also z.B. eine bessere Accuracy oder eine bessere Confusion Matrix auf \_test zu produzieren z.B. mit einem Neuronalen Netz vom Typ Deep Learning und das zu erklären.

*Lösung:*

Diese Aufgabe beschränkt sich nur auf die **pima-indians-diabetes** Daten. Dieser Datensatz stammt ursprünglich vom National Institute of Diabetes and Digestive and Kidney Disease, wobei die Patienten nach den Kriterien der Weltgesundheitsorganisation auf Diabetes getestet wurden. Ziel ist es, den Ausbruch von Diabetes, also OUTCOME, bei Pima-Indianern durch die Verwendung von neuronalen Netzen vorherzusagen. Ein Modell, welches sehr gut könnte, könnte dazu beitragen, Präventionsmaßnahmen für die Betroffenen auszurichten. Hierbei sind alle Patienten weiblich, mindestens 21 Jahre alt und lebten in der Nähe von Phoenix, Arizona. Die Daten bestehen somit aus mehreren medizinischen Prädiktorvariablen und einer Zielvariable (OUTCOME). OUTCOME weist einen Klassenwert auf, der angibt, ob es bei der Patientin innerhalb von 5 Jahren nach der Messung einen Diabetes ausbrach (1) oder nicht (0). Prädiktorvariablen beinhalten die Anzahl der Schwangerschaften, die der Patientin hatte, ihren BMI, Insulinspiegel, Alter und so weiter. Dies ist ein beliebtes Datensatz, der in der machine learning häufig untersucht wurde. Eine gute Vorhersagegenauigkeit, also Accuracy, liegt laut der Data-Science-Portal Kaggle bei 70% - 76%.

Um das neuronale Netz zu bauen, braucht man zum einen TensorFlow und zum anderen Keras. Durch TensorFlow findet der Lernprozess, bei dem das neuronale Netz optimiert wird, statt und Keras benötigt diese Bibliothek als Backend. Dank Keras werden Modelle definiert und dann an TensorFlow übergeben. Außerdem wird noch Sklearn verwendet, um die Rohdaten in Trainings- und Testdaten zu splitten. Der Schritt ist zwar am Anfang bei der ersten Modell nicht dabei, wird aber später in Einsatz gebracht, da es ja bekanntlich der Test eines statistischen Modells an neuen Daten aussagekräftiger ist als an Daten, mit denen der Algorithmus bzw. hier die neuronale Netze zuvor schon trainiert wurden.

Die Daten werden zunächst eingelesen und als acht Regressoren (X) und als eine Zielvariable (Y) deklariert.

```
dataset = pd.read_csv("C:\\\\B\\\\SRA\\\\Uni\\\\Master\\\\B Fächer\\\\Data Mining 2\\\\HELM_2018\\\\PYTHON_KERAS")

dataset.columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

split into input (X) and output (Y) variables
X = dataset.iloc[:,0:8]
Y = dataset.iloc[:,8]

dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	
0	6	148	72	35	0	33.6		0.627	50	1
1	1	85	66	29	0	26.6		0.351	31	0
2	8	183	64	0	0	23.3		0.672	32	1
3	1	89	66	23	94	28.1		0.167	21	0
4	0	137	40	35	168	43.1		2.288	33	1

Die Keras Python-Bibliothek für Deep Learning konzentriert sich auf die Erstellung von Modellen als eine Reihe von Schichten (sequence of layers). Daher werden im Folgenden mit den Sequential aus Keras Bibliothek gearbeitet. Mit der Funktion Sequential() baut man ein Modell, mit der man manuell Schicht für Schicht zum neuronalen Netz hinzufügen kann. Dabei wird die Methode add() verwendet, in der man die Art der Schicht definiert. Mit Dense() fügt man eine reguläre Schicht hinzu, bei der jeder Knoten dieser Schicht mit jedem Knoten der nächsten Schicht verbunden ist. Nur bei der ersten Schicht wird die Anzahl der Inputvariablen notwendig eingetragen (hier 8 Regressoren, also 8 inputs). Zudem wird natürlich hier mit dieser Funktion auch die Anzahl der Knoten, hier 12, definiert. Mit dem Argument kernel\_initializer wird der Anfangszustand aller Gewichtungen zwischen den Knoten, welche während des Trainings optimiert werden, bestimmt. Die Initialisierung kann im einfachsten Fall gleichverteilt, also uniform, erfolgen. Danach wird noch mit Hilfe des Arguments activation zuletzt noch bestimmt, wie jeder Knoten der Schicht auf den jeweiligen Input reagiert soll. Es werden im Folgenden die Aktivierungsfunktionen relu und sigmoid genutzt. Mit Rectified Linear Unit (kurz ReLU) wird die Funktion  $ReLU(x) := \max(0, x) = x^+$  und mit sigmoid die logistische Funktion, also  $\text{sigmoid}(x) := \frac{1}{1+\exp(-x)}$  gemeint.

Es wird hier also ein neuronales Netz mit zwei versteckten Schichten (two hidden layer) gebaut. Hierbei sind alle Inputvariablen mit allen Knoten der versteckten Schicht verbunden. Bei der ersten versteckten Schicht gibt es 12 Knoten die mit 8 Inputvariablen verbunden werden. Diese hidden layer wird dann wieder mit einem anderen hidden layer mit 8 Knoten verbunden. Zu der letzte hidden layer fügt man nun noch eine weitere Schicht hinzu. Sie entspricht dem Output und weil man nur an einer Zahl (OUTCOME=0 oder 1) interessiert ist, besitzt diese Schicht auch nur einen Knoten.

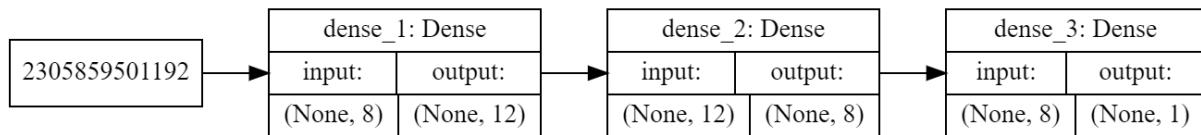
```

model = Sequential()

model.add(Dense(12, input_dim=8, kernel_initializer ='uniform', activation='relu'))
model.add(Dense(8, kernel_initializer ='uniform', activation='relu'))
model.add(Dense(1, kernel_initializer ='uniform', activation='sigmoid'))

```

Unten sieht man die schematische Darstellung des neuronalen Netzes. Diese wurde mit Hilfe der IPython.display Bibliothek und dazugehörige SVG Modul abgebildet.



Bevor das Modell trainiert wird, muss es erst kompiliert werden. Hier wird definiert wie der Lernprozess ablaufen soll. Es wird dem Algorithmus gesagt, welche Fehlerfunktion auf welche Art und Weise minimiert werden soll und welche Kennzahl während der Optimierung ausgegeben werden soll. Danach können wir das Modell auch schon trainieren.

Cross entropy oder auf Deutsch Kreuzentropie ist eine häufig verwendete loss function in neuronalen Netzen. Das Ziel muss es sein, die Kreuzentropie möglichst nahe an 0 zu bringen, dann ist der Fehler am kleinsten und das neuronale Netz sagt die Ziffer, also OUTCOME, korrekt voraus. Die Neuronen wurden, wie oben schon erläutert, im ersten Schritt mit zufälligen Aktivierungen initialisiert. Das Netzwerk berechnet nun die Ausgabegröße und wird von der Zielfunktion (Loss) bestraft. Der Fehler den es gemacht hat, wird über so genannte Backpropagation auf die jeweiligen Neuronen zurück verteilt, die ihn verursacht haben.

Es wird geschaut ob der Fehler größer oder kleiner wird wenn man die Aktivierung erhöht. Mathematisch gesehen wird also die Steigung der Fehlerfunktion bestimmt. Idealer Weise gibt es eine Richtung in die man optimieren kann, sodass die Fehlerfunktion minimiert wird. Somit kommt man auch bei der Fehlerminimierung an. Diese Suche nach der idealen Aktivierung bedeutet mathematisch das Finden von Minimalwerten im Fehlerraum. Es gibt viele Verfahren wie Stochastic Gradient Descent (SGD), Nesterov Momentum, Adagrad und Adadelta oder Rmsprop. Hier wird ADAM verwendet, der im Moment einer der am häufigsten angewendeten Optimierer ist. Adam steht für Adaptive moment estimation und ist eine methodische Kombination von den Optimierungstechniken AdaGrad und RMSProp [PN18].

Die kompletten Daten werden insgesamt 40-mal durch das neuronale Netz geschoben. In jedem dieser Durchgänge werden die Daten in Batches von je 10 Instanzen – in diesem Fall 10 Patienten – in das Netz eingespeist. Nach jedem Batch wird eine Fehlerfunktion berechnet und die Gewichte zwischen den Knoten via Backpropagation werden entsprechend angepasst, sodass der Schätzfehler kleiner wird. Während der Algorithmus arbeitet, wird der Fortschritt vom Programm ausgegeben. Rechts sieht man zum Beispiel in jeder Zeile den momentanen Accuracy Score des Modells.

```

Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Fit the model
model.fit(X, Y, epochs=40, batch_size=10)

Epoch 1/40
768/768 [=====] - 1s 862us/step - loss: 0.6761 - acc: 0.6510
Epoch 2/40
768/768 [=====] - 0s 189us/step - loss: 0.6622 - acc: 0.6510
Epoch 3/40
768/768 [=====] - 0s 194us/step - loss: 0.6524 - acc: 0.6523
Epoch 4/40
768/768 [=====] - 0s 181us/step - loss: 0.6432 - acc: 0.6602
Epoch 5/40
768/768 [=====] - 0s 191us/step - loss: 0.6264 - acc: 0.6706

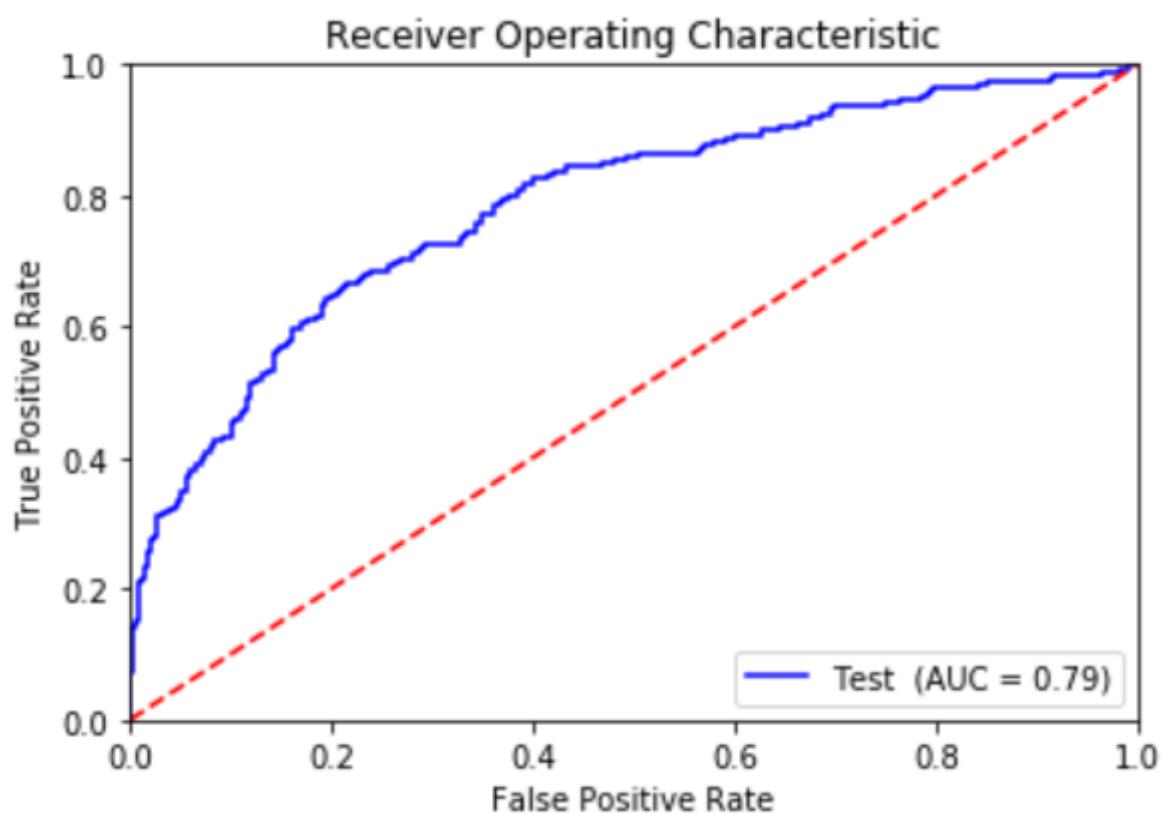
```

Auch auf den Testdaten, was eigentlich die Daten sind die auch für Trainieren verwendet wurde, performt das neuronale Netz mit einer Präzision von 74.87%.

```
evaluate the model
scores = model.evaluate(X, Y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
768/768 [=====] - 0s 141us/step
acc: 74.87%
```

Die dazugehörige ROC-Kurve und die Confusion Matrix werden auch noch abgebildet.



```
y_pred1 = model.predict_classes(X)
pd.crosstab(y_pred1[:,0],Y, colnames=
```

True = reference	0	1
Predicted = data		
0	403	96
1	97	172

Das ganze Modell wird nun auf wirkliche Trainings- und Testdaten angewendet. Dabei bleiben die definierte Angaben alle gleich.

```
split the data into training and testing sets
from sklearn.cross_validation import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4, random_state=12345)

#print(X_train)
#print(X_test)

model2 = Sequential()

model2.add(Dense(12, input_dim=8, kernel_initializer ='uniform', activation='relu'))
model2.add(Dense(8, kernel_initializer ='uniform', activation='relu'))
model2.add(Dense(1, kernel_initializer ='uniform', activation='sigmoid'))

Compile model
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Fit the model
model2.fit(X_train, Y_train, epochs=40, batch_size=10)

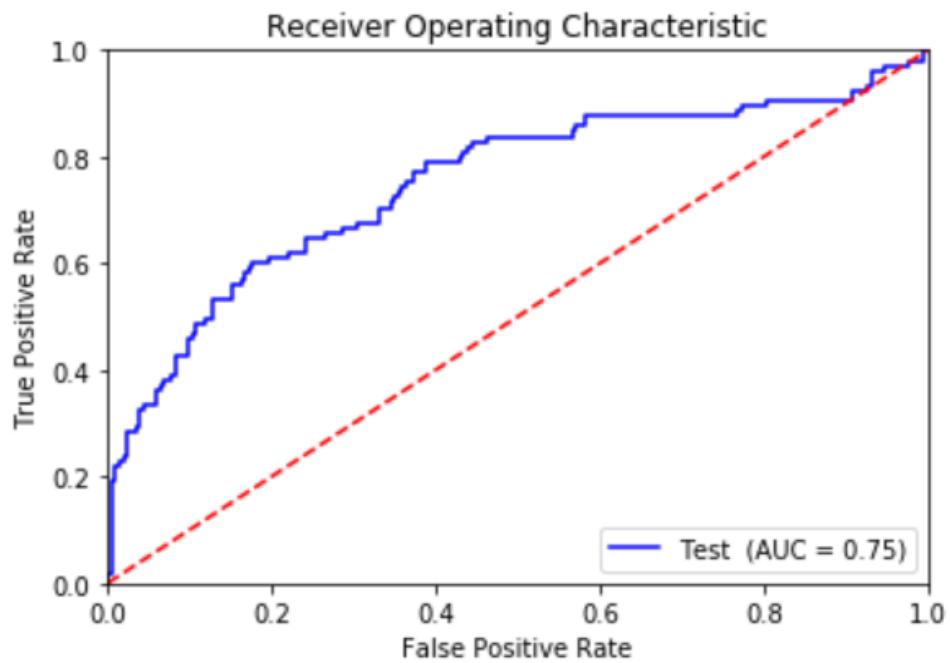
Epoch 1/40
460/460 [=====] - 0s 678us/step - loss: 0.6876 - acc: 0.6435
Epoch 2/40
460/460 [=====] - 0s 105us/step - loss: 0.6752 - acc: 0.6457
Epoch 3/40
460/460 [=====] - 0s 113us/step - loss: 0.6694 - acc: 0.6457
```

Hier ist es erkennbar, dass sich die Accuracy nicht verschlechtert sondern ganz im Gegenteil sich sogar verbessert.

```
evaluate the model
scores = model2.evaluate(X_test, Y_test)
print("%s: %.2f%" % (model2.metrics_names[1], scores[1]*100))

308/308 [=====] - 0s 253us/step
acc: 75.00%
```

True = reference		0	1
Predicted = data			
		0	1
	0	179	53
	1	24	52



Nun wird noch ein Schicht, also noch ein hidden layer, eingefügt. Zudem wird das neuronale Netz hier mit 1000 Epochen und mit einem Batch Menge von 40 lernen.

```

model3 = Sequential()

model3.add(Dense(12, input_dim=8, kernel_initializer ='uniform', activation='relu'))
model3.add(Dense(8, kernel_initializer ='uniform', activation='relu'))
model3.add(Dense(8, kernel_initializer ='uniform', activation='relu'))
model3.add(Dense(1, kernel_initializer ='uniform', activation='sigmoid'))
```

---

```
Compile model
model3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

---

```
Fit the model
model3.fit(X_train, Y_train, epochs=1000, batch_size=40)
```

```

Epoch 1/1000
460/460 [=====] - 0s 43us/step - loss: 0.5044 - acc: 0.7565
Epoch 2/1000
460/460 [=====] - 0s 31us/step - loss: 0.5032 - acc: 0.7500
Epoch 3/1000
460/460 [=====] - 0s 28us/step - loss: 0.4974 - acc: 0.7696
```

Die dazugehörige Accuracy ist leider sogar weniger als Modell 2.

```
evaluate the model
scores = model3.evaluate(X_test, Y_test)
print("%s: %.2f%%" % (model3.metrics_names[1], scores[1]*100))

308/308 [=====] - 0s 274us/step
acc: 74.35%
```

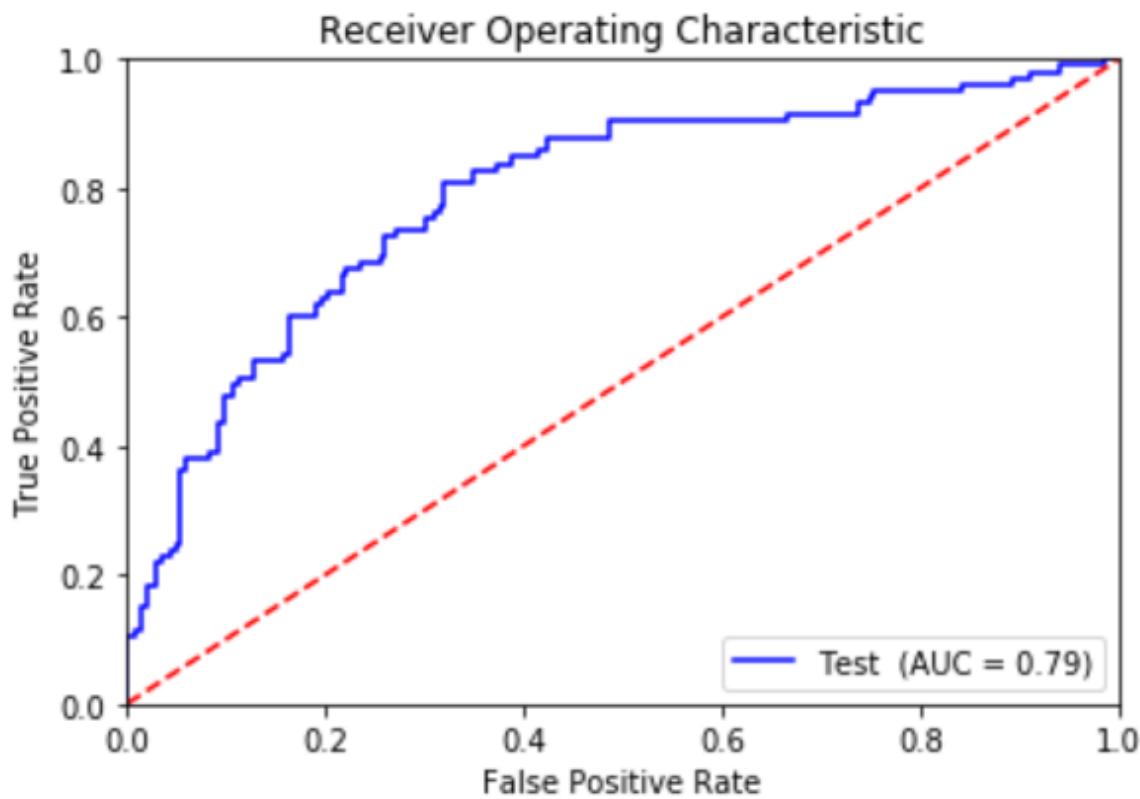
```
y_pred3 = model3.predict_classes(X_test)

pd.crosstab(y_pred3[:,0],Y_test, colnames=['True = reference'],
```

True = reference 0 1

Predicted = data

	0	1
0	158	34
1	45	71



Je nach Initialisierung, Loss Funktion und Aktivierungsfunktion lernt das Netzwerk schneller oder langsamer. Daher können natürlich noch weitere Modelle aufgebaut und eventuell bessere Ergebnisse erzielt werden.

# Literaturverzeichnis

---

- [AF11] Andreas Friedrich: Neuronale Netze - Theoretische Grundlagen und Anwendung in der Verkehrserkennung, GRIN Verlag, 2011
- [AL05] Alexander Linder: Web Mining - Die Fallstudie Swarovski: Theoretische Grundlagen und praktische Anwendung, Springer-Verlag, 2005
- [CH02] Chawla, Bowyer, Hall, Kegelmeyer: SMOTE - Synthetic Minority Over-sampling Technique, AI Access Foundation and Morgan Kaufmann Publishers, 2002
- [DY17] David Yeo: Neural Network Modeling - Course Notes, SAS Institute Inc. Cary, 2017
- [HB06] J.F. Hair, W.C. Black: Multivariate data analysis, Prentice Hall, 2006
- [HP06] Helmut Pruscha: Statistisches Methodenbuch - Verfahren, Fallstudien, Programmcodes, Springer-Verlag, 2006
- [KB00] Klaus Backhaus, Bernd Erichson, Wulff Plinke und Rolf Weiber: Multivariate Analysemethoden, Springer-Verlag, 2000
- [LB96] Leo Breiman: Bagging Predictors - Machine Learning, 24, Seite 123-140, 1996
- [NN18] <https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/>, abgerufen am 02.07.2018
- [OI07] Ovidiu Ivanciu: Applications of Support Vector Machines in Chemistry, Wiley-VCH, John Wiley & Sons, 2007.
- [PN18] <http://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert>, abgerufen am 08.07.2018
- [SA10] Shigeo Abe: Support Vector Machines for Pattern Classification, Springer Science & Business Media, 2010
- [TB14] Tim vor der Brück: Wissensakquisition mithilfe maschineller Lernverfahren auf tiefen semantischen Repräsentationen, Springer-Verlag, 2014
- [TF06] Tom Fawcett: An introduction to ROC analysis, Pattern Recognition Letters 27, 2006
- [TR15] Thomas A. Runkler: Data Mining - Modelle und Algorithmen intelligenter Datenanalyse, Springer VS, 2015
- [UB08] Udo Bankhofer und Jürgen Vogel: Datenanalyse und Statistik: Eine Einführung für Ökonomen im Bachelor, Springer-Verlag, 2008
- [WICA] [https://de.wikipedia.org/wiki/CART\\_\(Algorithmus\)](https://de.wikipedia.org/wiki/CART_(Algorithmus)), abgerufen am 11.06.2018
- [WISV] [https://de.wikipedia.org/wiki/Support\\_Vector\\_Machine](https://de.wikipedia.org/wiki/Support_Vector_Machine), abgerufen am 28.06.2018
- [WITD] [https://de.wikipedia.org/wiki/Time\\_Delay\\_Neural\\_Network](https://de.wikipedia.org/wiki/Time_Delay_Neural_Network), abgerufen am 04.07.2018

## **Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Soweit ich auf fremde Materialien, Texte oder Gedankengänge zurückgegriffen habe, enthalten meine Ausführungen vollständige und eindeutige Verweise auf die Urheber und Quellen. Alle weiteren Inhalte der vorgelegten Arbeit stammen von mir im urheberrechtlichen Sinn, sowie keine Verweise und Zitate erfolgen. Mir ist bekannt, dass ein Täuschungsversuch vorliegt, wenn die vorstehende Erklärung sich als unrichtig erweist.

Darmstadt, den 14. Juli 2018

---

Unterschriften