Klausur zu Healthcare Data Analytics

Dr. Michael Strobel

15.07.2022

Organisatorisches

Vorname:

Nachname:

Matrikelnummer:

Rahmenbedingungen

- Die Klausur dauert 90 Minuten.
- Die Klausur enthält 5 Aufgaben auf 18 Seiten. Bitte überprüfen Sie ob der Klausurbogen vollständig ist.
- Erlaubte Hilfsmittel: ein doppelseitig beschriebenes Din-A4 Blatt und ein (nicht programmierbarer) Taschenrechner.
- Weitere Hilfsmittel sind nicht erlaubt.

Aufgabe 1: Allgemeines Verständnis

1. Was ist der Unterschied zwischen Trainings Set und Test Set?
2. Wofür dienen Label bei Supervised Learning?
3. Nennen Sie ein Beispiel für ein Unsupervised Learning.
4. Wie viel Prozent der Daten sollten in etwa für das Testset beiseite gelegt werden?
5. Was ist der Unterschied zwischen Klassifikation und Regression?
6. Was verstehen wir unter Underfitting?
7. Was was verstehen wir unter Overfitting?
8. Was verstehen wir unter Regularisierung eines Modells?
9. Was ist Semi-Supervised Learning?
10. Was ist ein Data Warehouse?

Aufgabe 2: Python Basics

a) Python Pakete

In der Vorlesung haben wir zahlreiche Software Pakete kennengelernt. Beschreibungen Sie kurz die **Verwendungen** der unten stehen Pakete in der Vorlesung an. Desweiteren interpretieren Sie die **Kommandos** und deren **Output**.

Numpy

Code Beispiel

```
>>> import numpy as np
>>> age_in_years = np.array([10, 13, 18, 7, 19, 33, 45, 81, 60])
>>> np.where(age_in_years < 20, age_in_years, 0)
array([10, 13, 18,  7, 19,  0,  0,  0,  0])
```

Verwendung von Numpy:

Interpretation des Code Beispiels:

Pandas

Code Beispiel

```
>>> import numpy as np
>>> import pandas as pd

>>> df = pd.DataFrame(np.arange(20).reshape(5,4), columns=["A","B","C","D"])
>>> df.loc[[0,1],"B"]
0    1
1    5
Name: B, dtype: int32
```

Verwendung von Pandas:

Interpretation des Code Beispiels:

b) Quicksort

Implementierung Sie Quicksort in Python 3

```
def quicksort(arr):


    pivot =
    left =
    middle =
    right =
    return


>>> print(quicksort([3, 6, 8, 10, 1, 2, 1]))
[1, 1, 2, 3, 6, 8, 10]
```

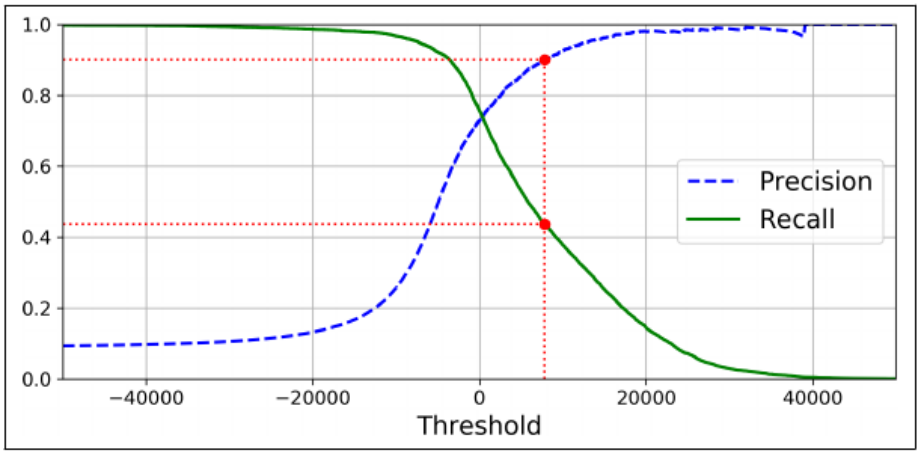
Hinweis: der rekursive Quicksort Algorithmus in Haskell kann folgendermaßen implementiert werden

```
-- Die leere liste ist bereits sortiert
quickSort []     = []
-- x:xs nimmst das erste Element aus der Liste und nennt es x,
-- xs ist dann der Rest der Liste ohne x
quickSort (x:xs) =

quickSort [a | a <- xs, a < x]    -- Sortiere die linke Seite
++ [x] ++                         -- Sortiere x in die Mitte
quickSort [a | a <- xs, a >= x]   -- Sortiere die rechte Seite
```

Aufgabe 3: Precision und Recall

a) Precision / Recall Tradeoff

1. Wie ist Precision definiert? Wie kann Precision interpretiert werden?
2. Wie ist Recall definiert? Wie kann Recall interpretiert werden?
3. Was versteht man unter dem Precision- und Recall Tradeoff? Interpretieren Sie hiefür das Schaubild aus der Vorlesung.



4. Sie haben einen Binären Klassifikator trainiert, der 1 ausgibt wenn eine Person krank ist und 0 wenn die Person gesund ist. In $y_{predicted}$ finden Sie die Vorhersagen des Algorithmus und in y_{true} finden Sie den korrekten Wert.

$y_{predicted} = [0, 1, 1, 0, 1, 1]$

$y_{true} = [0, 1, 0, 0, 1, 0]$

Berechnen Sie folgende Werte

- True Positive (TP)
- False Positive (FP)
- False Negative (FN)
- Precision
- Recall

Aufgabe 4: Pipelines

a) Verständnis

1. Was verstehen wir unter einer Data Pipeline?
2. Was ist der Unterschied zwischen einer Kategorischen und einer Numerischen Pipeline?
3. Nennen Sie 3 Schritte die Sie in einer Numerischen Data Pipeline ausführen sollten. Begründen Sie Ihre Antwort.
4. Nennen Sie 3 Schritte die Sie in einer Kategorischen Data Pipeline ausführen sollten. Begründen Sie Ihre Antwort.

b) Pipeline in Python

Gegeben seien folgen Daten die wir in einen Pandas Dataframe mit dem Variablennamen `heart_data` laden:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40.0	M	ATA	140.0	289.0	0.0	Normal	172.0	N	0.0	Up	0
1	49.0	F	NAP	160.0	180.0	0.0	Normal	156.0	N	1.0	Flat	1
2	37.0	M	ATA	130.0	283.0	0.0	ST	98.0	N	0.0	Up	0
3	48.0	F	ASY	138.0	214.0	0.0	Normal	108.0	Y	1.5	NaN	1
4	54.0	M	NAP	150.0	NaN	0.0	Normal	122.0	N	0.0	Up	0

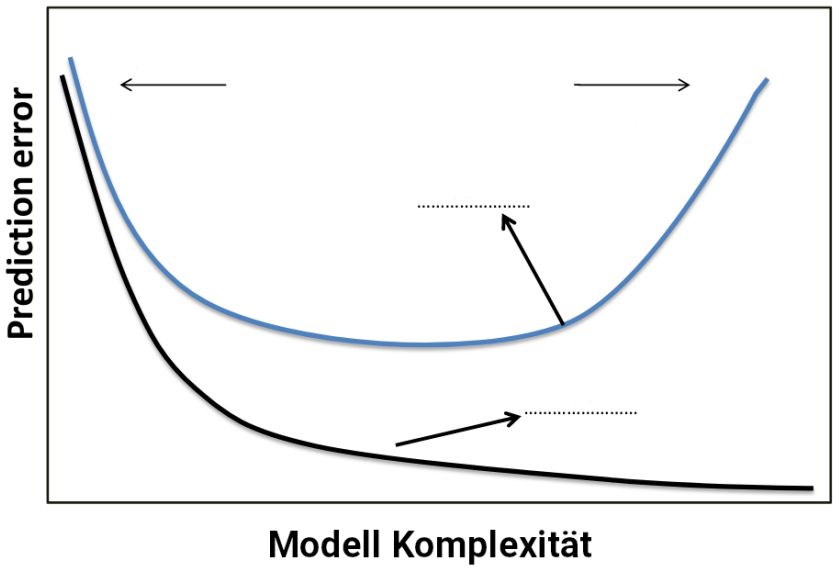
Ergänzen Sie den unten stehenden Python 3 Code um sowohl eine kategorische als auch eine numerische Pipeline zu bauen und diese dann zu einer Gesamtpipeline zusammenzusetzen.

```
import numpy as np
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import (
    LabelEncoder,
    OneHotEncoder,
    OrdinalEncoder,
    StandardScaler,
)
from sklearn.impute import SimpleImputer

heart_data = pd.read_csv("data/heart_unclean.csv")

ordinal_features =
nominal_features =
numeric_features =

numeric_transformer = Pipeline(
    steps=[

    ]
)
nominal_transformer = Pipeline(
    steps=[


    ]
)
ordinal_transfomer = Pipeline(
    steps=[
```

```
    ]
)

full_pipeline = ColumnTransformer(
    transformers=[



    ]
)
```

Aufgabe 5: Neuronale Netze

a) Verständnis



- Welche der Kurven ist wahrscheinlicher der Trainingsfehler und welche ist der Validierungsfehler? Geben Sie dies im Diagramm an, indem Sie die gepunkteten Linien ausfüllen.
- In welchen Bereichen des Diagramms sind Bias und Varianz niedrig bzw. hoch? Geben Sie deutlich an auf dem Diagramm mit vier Beschriftungen: “geringe Varianz”, “hohe Varianz”, “geringer bias”, “hoher bias”.
- In welchen Bereich underfittet bzw. overfitted das Modell? Geben Sie dies im Diagramm mit “Overfitting” und “Underfitting” an.

b) Keras

Auf dem Bild ist die Architektur von eines Convolutional Neural Network zu sehen. Ergänzen Sie den unten stehenden TensorFlow Code um das CNN in Keras zu implementieren.

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully connected	–	1,000	–	–	–	Softmax
F10	Fully connected	–	4,096	–	–	–	ReLU
F9	Fully connected	–	4,096	–	–	–	ReLU
S8	Max pooling	256		3×3	2	`valid`	–
C7	Convolution	256		3×3	1	`same`	ReLU
C6	Convolution	384		3×3	1	`same`	ReLU
C5	Convolution	384		3×3	1	`same`	ReLU
S4	Max pooling	256		3×3	2	`valid`	–
C3	Convolution	256		5×5	1	`same`	ReLU
S2	Max pooling	96		3×3	2	`valid`	–
C1	Convolution	96		11×11	4	`valid`	ReLU
In	Input	3 (RGB)	227×227	–	–	–	–

```
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense

model = keras.models.Sequential([
  Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu',\
  input_shape=(227,227,3)),









  Dense(1000,activation='softmax')
])
```

Aufgabe 6: Decision Tree und Random Forrest

1. Nennen Sie vier Eigenschaften von Decision Trees.
2. Was verstehen wir unter dem Gini Koeffizient? Wie interpretieren Sie Gini 0.1 und Gini 0.5?
3. Wie kann ein Decision Tree Regularisiert werden?
4. Erklären Sie das Konzept von Bagging und Boosting. Was ist der Unterschied?
5. Wie werden Decision Trees konstruiert?
6. Betrachten Sie das Schaubild A. Welches Bilder zeigt einen Decision Tree und welches ein Random Forrest? Begründen Sie Ihre Antwort.

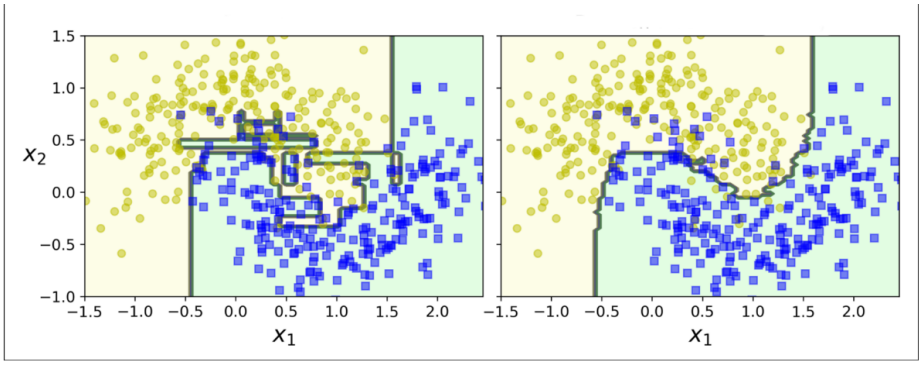


Figure 1: Schaubild A