Heinrich Heine
Universität
Düsseldorf

INSTITUT FÜR INFORMATIK
Softwaretechnik und Programmiersprachen
Universitätsstr. 1 D–40225 Düsseldorf

Leistungsvorhersage von Studenten mittels maschinellen Lernens

von
Luca Kutzner

Bachelorarbeit

im Studiengang Informatik
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Beginn der Arbeit:	09. Juli 2020
Abgabe der Arbeit:	30. September 2020
Gutachter:	Dr. Jens Bendisposto
	Prof. Dr. Gunnar Klau

Selbstständigkeitserklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Velbert, den 30. September 2020

_____

Luca Kutzner

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit der Frage, wie die erwarteten, nächsten Leistungen von Studenten aussehen werden. Darunter fallen die Leistungen in der Klausur, die Punktzahl auf dem nächsten Übungsblatt und ob ein Student Hilfe benötigt oder nicht. Alle Daten dafür sollen aus dem **Punkteverhalten der Studenten auf den bisherigen Übungsblättern** entnommen werden und die Prognosen mithilfe von Methoden aus dem maschinellen Lernen erstellt werden.

Dafür wird ein Datensatz aus dem Fach „Informatik I“ der Heinrich-Heine-Universität Düsseldorf verwendet, der, unter anderem, sowohl die erreichten Übungspunkte von 12 Übungsblättern, als auch die erzielte Punktzahl und Note aus der Klausur enthält.

Für die Vorhersage der Leistungen in der Klausur werden die **Summe aller bisherigen Übungspunkte** und die **Übungspunkte pro bearbeitetes Blatt** berechnet. Diesen *Features* wird dann das jeweilige Quadrat, aber auch die paarweise Multiplikation beider hinzugefügt.
Es wurden verschiedene Vorhersagemethoden in einer *GridSearch* ausprobiert. Die lineare Regression hat sich hierbei, von allen getesteten, als durchschnittlich genauste bewiesen. Sie erreichte einen R^2-Wert von 0.4576, eine Notenschnittabweichung von 0.023 und eine durchschnittliche, absolute Abweichung von 14.419 Klausurpunkten, was etwa drei Notenschritten entspricht, wenn für jeden Studenten einmal die Vorhersage gebildet und der Schätzer mit allen anderen trainiert wird.
Die Vorhersage kann auch laufend, während des Semesters gebildet werden, fängt beim ersten Blatt mit einem R^2-Wert von 0.0809 an und wird für jedes neue Blatt immer genauer.

Für die Vorhersage der Leistungen auf dem nächsten Übungsblatt hingegen werden die **Punktzahlen der zwei zuletzt korrigierten Übungsblättern** genutzt, welchen, genau wie den anderen *Features*, wieder das jeweilige Quadrat und die paarweise Multiplikation angehängt wird. Im Durchschnitt aller Blattvorhersagen wird ein R^2-Wert von 0.5327 und eine durchschnittliche, absolute Abweichung von 18.243% der Punkte erreicht.

Ob ein Student Hilfe benötigt, wird daran festgelegt, ob er auf dem **aktuellen Blatt unter dem Durchschnitt abzüglich der doppelten Standardabweichung** fällt und auf dem **nächsten Blatt weniger als die Hälfte der Punkte** erreicht.
Damit die Hilfe auch einen Nutzen hat, ist es verpflichtend, dass die ausgewählten Studenten eine **Eigenleistung** zeigen.

Zum Schluss konnte eine **Serveranwendung mit einer *HTML-GUI*** entwickelt werden, in der alle genannten Vorhersagen bestimmt werden können und durch Hinzufügen von Daten mit Übungspunkten und Klausurergebnissen die Schätzer für die Vorhersagen verbessert werden können.

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Jedes Jahr werden von den Studenten der Mathematisch-Naturwissenschaftlichen Fakultät viele Übungsaufgaben bearbeitet. Die dort erhaltenden Punkte lassen vermutlich darauf schließen, ob und mit welcher Note die Klausur bestanden wird. Dieser Vermutung soll in dieser Arbeit nachgegangen werden.
Seitens der Studenten kann die voraussichtliche Note dazu animieren, noch mehr zu üben, oder ihnen die Angst vor der Klausur nehmen.
Die Organisatoren der Veranstaltungen erhalten dafür einen guten Überblick darüber, wie gut ihre Studenten in der Klausur, verglichen mit schon vergangenen Veranstaltungen, abschneiden werden, um gegebenenfalls Übungsaufgaben, Vorlesungs- und Klausurinhalte abzuändern.

Des Weiteren gibt es noch das Problem, dass es zu viele Studenten gibt, die nicht einmal die Klausurzulassung erreichen. In den meisten Fällen wenden sich diese Studenten aber leider nicht an das *Mentoring*-Programm, obwohl es genau für diese Studenten eingerichtet wurde.
Ziel dieses Systems ist es also auch schon früh mangelhafte Leistungen zu erkennen und die betroffenen Studenten zu markieren, damit sie dem *Mentoring*-Programm zugeführt werden können.

Insgesamt soll eine Anwendung einwickelt werden, die für diese Probleme eine Vorhersage bildet und dessen Schätzer sich, basierend auf neuen Daten, weiterentwickeln.

1.2 Verwandte Arbeiten

Es gibt bereits andere Arbeiten, die in eine ähnliche Richtung gehen, wie diese.

Askinadze, hat in seiner Arbeit [Ask16] die *Support Vector Machines* (*SVM*) benutzt, um eine Klassifikation nach „bestanden“ oder „nicht bestanden“ zu entwickeln. Dafür nutzte er einen 395 Stichproben großen Datensatz einer Mathematik-Klasse einer portugiesischen Schule. Die Daten enthielten nur privaten Informationen, wie unter anderem, Wohnort, Job der Eltern, Familiengröße und Beziehungsstatus.
Im Falle dieser Arbeit sollen solche Informationen aber von den Leistungen innerhalb des jeweiligen Kurses abhängig sein und wenn möglich sogar eine spezifische Klausurnote vorhergesagt werden.

Ramesh, Parkavi, Ramar haben in ihrem Papier [RPR13] zumindest die Klausurnote vor-

hergesagt und abhängig dieser Note ausgefiltert, wer hilfsbedürftig ist und wer nicht. Auch sie haben ausschließlich private Informationen, wie zum Beispiel Geschlecht, Status der Eltern, Wohnort, Schulart oder die Familiengröße für die Vorhersage verwendet, aber keine Übungspunkte oder sonstige Leistungen innerhalb eines Kurses. Wie sich herausstellt, sorgt für ihre Daten der *Multi Layer Perception*-Klassifizierer die genauste Aufteilung nach „bestanden“ oder „nicht bestanden“.

Ya-Han Hu, Chia-Lun Lo und Sheng-Phao Shih hingegen nutzen in [HLS14] das Verhalten von Studenten in einem e-learning-System, um riskant-schlechte Studenten noch während des Semesters zu identifiziert. Das Ergebnis ist ein *AdaBoost*-Klassifizierer um eben jene zu bestimmen.
Leider sind in dieser Arbeit dafür zu wenig Daten über das Onlineverhalten der Studenten vorhanden, sodass in diese Richtung fast nicht geforscht werden kann.

Die Publikation [MM17] von Mwalumbwe und Mtebe, bezieht die Daten aus *Moodle*, einer Onlineplattform, welche auch hier an der HHU verwendet wird, um den Zusammenhang mit den Klausurnoten zu bestimmen. Das Ergebnis ist, dass Diskussionsbeiträge, Interaktionen mit anderen Studenten und Übungen die wichtigsten Faktoren dafür sind.

Dambolena verwendete in seinem Papier [Dam00] eine simple, eindimensionale, lineare Regression ersten Grades, um von den Halbjahresklausurergebnissen eine Prognose für die Abschlussklausur zu bestimmen.
Da das Fach des analysierten Datensatzes keine Halbjahresklausur anbietet, kann in diese Idee nicht überprüft werden.

Es gibt natürlich auch noch viele weitere Arbeiten, in denen verschiedene, erwartete nächste Leistungen von Schülern oder Studenten vorhergesagt werden, aber da, außer Müllers Arbeit [Mü18], keine andere Arbeit gefunden wurde in der, in Abhängigkeit von bereits erreichten Übungspunkten, die Leistung in der Klausur vorhergesagt, die Leistungen auf dem nächsten Übungsblatt geschätzt oder hilfsbedürftige Studenten ausgewählt werden, wird sich hier diesen Problemen angenommen.

In der Bachelorarbeit von Johannes Müller [Mü18] wurde bereits eine Vorhersage entwickelt, mit der die Klausurnote vorhergesagt werden kann. Er verwendete dazu den selben Datensatz, wie jener, der in dieser Arbeit genutzt wird, kommt aber zu einem etwas anderen Ergebnis, was an einem anderen Zweck der Vorhersage liegt.
Seine Vorhersage soll allen Studenten, die unter die Hälfte der Punkte der Übungsaufgaben fallen, eine 6.0 in der Klausur vorhersagen. Das führt dazu, dass Studenten, die die Zulassung bereits früher einmal erreicht haben und sie nicht erneut erlangt haben, keine vernünftige Klausurnote vorhergesagt bekommen, weil sie unter der Zulassungsgrenze der Punkte liegen. Noch gravierender ist, dass Fächer, die eine andere Klausurzulassungsgrenze haben, gar keine realistische Vorhersage bekommen, denn wie sich

zeigt, arbeiten die meisten Studenten nur, bis sie die Zulassung gerade eben erreicht haben. Liegt die Zulassung dann beispielsweise unterhalb der Hälfte der Punkte, bekommen auch die meisten Studenten eine Prognose mit der Note 6.0.
Des Weiteren hat er alle Studenten nicht betrachtet, die sich im zweiten oder höheren Prüfungsversuch befanden. Eine sehr unbegründete Entscheidung, da sich das Übungspunkt und Klausurverhalten besagter Studenten nicht derer unterscheidet, die sich im ersten Prüfungsversuch befanden.
Außerdem lautet sein Ergebnis, dass die Punkte auf den einzelnen Übungsblättern, auf eine Dimension reduziert (siehe Abschnitt 3.1.4), zusammen mit den Übungspunkten pro bearbeitetes Blatt die besten Ergebnisse erzeugt. Das liegt daran, dass er die Dimensionsreduktion (*PCA*) unbegründet verpflichtend in seiner *GridSearch* eingebettet hat. Wie sich herausstellt, ist es ein wenig besser die Gesamtsumme, statt der dimensionsreduzierten Übungsblätter, zu verwenden.
Ein weiteres, kleines Problem ist, dass seine Selektierung der Studenten, die mindestens eine Abgabe gemacht haben, einen Fehler aufwies. Das liegt aber an einem Verhalten des automatischen Abgabesystems (*AUAS*). Sobald einmal „Blatt abgeben“ aufgerufen wird, erfolgt eine Einrichtung der Abgabe. Ab diesem Zeitpunkt erhält man zumindest 0 Punkte, obwohl noch gar keine Daten hochgeladen wurden. Auch solche Fälle finden sich in dem von ihm selektierten Daten wieder, obwohl er sie eigentlich auslassen wollte.
Wie sich später zeigt, ist es besser, die Klausurpunkte vorherzusagen, weil sie in kleineren Schritten vorhanden sind, als 11 mögliche Notenschritte. Diese Idee hat Müller auch nicht beachtet.
Dass entweder die Noten für manche Studenten unterschiedlich vergeben worden, oder die Noten im Datensatz fehlerhaft sind, ist ihm auch nicht aufgefallen.

Zum Schluss sind aber weder die verarbeiteten Daten, noch sein Forschungscode verfügbar, was dazu führt, dass sich diesen Problemen nochmal angenommen werden muss, aber an den oben beschriebenen Stellen andere Entscheidungen getroffen und die Verbesserungen eingebracht werden.

Aus seiner entwickelten Vorhersage kann noch nicht vernünftig abgeleitet werden, für wen Hilfeleistungen angeboten werden sollten. Dafür liefert eine Vorhersage der Punkte auf dem nächsten Übungsblatt nützliche Informationen und dieses Unterfangen wurde auch in noch keiner bekannten Arbeit durchgeführt.
Als größte Erweiterung der Arbeit steht aber die Anwendung, mit der Prognosen für Klausuren erstellt und die Schätzer laufend weiterentwickelt werden können, denn in der Anwendung von Müller ist nichts dergleichen implementiert worden. Besagte Anwendung ist nicht einmal lauffähig, weil erforderliche Bibliotheken nur ein Abbild seiner lokalen *Python*-Distribution, *Miniconda* sind und somit auf anderen Systemen nicht funktionieren. Da ohnehin keine vernünftigen Funktionen implementiert wurden, ist es nicht sinnvoll, Zeit in die Rettung dieses Systems zu investieren.

1.3 Vorgehen

Um zu einer Lösung für die oben genannten Probleme zu gelangen, wurde die folgende Aufteilung auf die jeweiligen Abschnitte durchgeführt:

- Abschnitt 2: Die vorhandenen Daten werden analysiert, graphisch dargestellt und die nötigen Umformungen für eine mathematische Vorhersage getroffen.
- Abschnitt 3: Vorhersagen für die verschiedenen Anwendungsmodi werden aufgestellt, verglichen und optimiert.
- Abschnitt 4: Die Vorhersagen werden in einer Serveranwendung implementiert, damit die Erkenntnisse dieser Arbeit genutzt werden können.
- Abschnitt 5: Der Klausurschätzer wird mit neuen, unbekannten Eingaben getestet und bewertet.
- Abschnitt 6: Alle wichtigen Ergebnisse werden zusammengefasst, diskutiert und eine Empfehlung gestellt, wie sie zukünftig verbessern werden können.

2 Analyse des Datensatzes

Als Erstes muss der Datensatz analysiert werden. Dieser Abschnitt beschäftigt sich dafür zunächst mit einer genauen Übersicht über die vorhandenen Daten und geht dann in eine Selektierung und Verarbeitung dieser über, damit mathematische Berechnung darauf durchgeführt werden können. Zum Schluss folgt noch die Erforschung der Zusammenhänge zwischen den Werten.

2.1 Übersicht

Die gegebenen Daten wurden an der Heinrich-Heine-Universität Düsseldorf im Wintersemester 2015/2016 während des Kurses „Informatik I“ erhoben und dienen nun als Grundlage für die Vorhersage zukünftiger Leistungen von Studenten.

2.1.1 Features

Für jeden Studenten gibt es in den Daten eine Reihe an Attributen, die im maschinellen Lernen auch *Features* genannt werden. In Tabelle 1 werden alle *Features* aufgelistet, die ein Student in dem Datensatz haben kann.

Tabelle 1: Alle *Features* des Datensatzes

Name	Zahlenbereich	Beschreibung
summe-blatt$N$	$\mathbb{N}_0 \cup NaN$	$N \in [1, 12]$, $\sum_{m \in M}$ (blatt$N$-aufgabe$m$)
summe-uebungs-punkte	$\mathbb{N}_0 \cup NaN$	$\sum_{n \in N}$ (summe-blatt$n$)
zugriffe-auf-ilias	$\mathbb{N}_0 \cup NaN$	Summe der Zugriffe auf das *ILIAS*
fach	$\{$“$informatik$”, “$mathematik$”, “$physik$”, “$informations\text{-}wissenschaften$”, “$chemie$”, “$studium\ universale$”, “$anderer\ studiengang$”, “$unbekannt$”$\} \cup NaN$	Studiengang des Studenten
semester	$\mathbb{N} \cup NaN$	Semester in dem die Veranstaltung besucht wurde
pruefungsversuch	$\{1, 2, 3\} \cup NaN$	Nummer des Prüfungsversuchs
klausur	$\{$“ : $hauptklausur$”, “ : $nachklausur$”$\} \cup NaN$	Teilnahme an Hauptklausur, Nachklausur oder keiner
summe-klausur-punkte	$\mathbb{R}_0^+ \cup NaN$	$\sum_{x \in X}$ (klausur-x)
note	$\{1.0, 1.3, 1.7, 2.0, 2.3, 2.7, 3.0, 3.3, 3.7, 4.0, 4.3, 4.7, 5.0\} \cup NaN$	erhaltene Note in der Abschlussklausur
blatt$N$-aufgabe$M$	$\mathbb{N}_0 \cup NaN$	$N \in [1, 12]$, $M \in [1, 5]$ erreichte Punktzahl für Aufgabe M auf Blatt N
klausur-X	$\mathbb{R}_0^+ \cup NaN$	$X \in \{$“$algorithmus$”, “$arrays$”, “$ausgaben$”, “$codeanalyse$”, “$funktionen$”, “$multiple\text{-}choice$”, “oo”, “$schleifen$”, “$tests$”$\}$ erreichte Punktzahl in der dazugehörigen Klausuraufgabe
name	$Text$	Anonymer Name des Studenten
deanonymization	$32bit\ Hash$	Entschlüsselungskey um Zuweisung zu Studenten zu entschlüsseln
tutor1, tutor2	$Text$	Anonymer Name der Tutoren aus den praktischen Übungen

Es handelt sich insgesamt um 90 Spalten, welche nicht alle verwendet werden sollten, weil einige Attribute keine Auswirkungen auf die erbrachte Leistung haben, es nicht sicher ist, ob diese Daten auch in Zukunft erhoben werden oder um die Laufzeit zu verringern. Welche *Features* davon selektiert werden wird im nächsten Abschnitt 2.2 geklärt.

2.1.2 Größen im Datensatz

Leider ist der Datensatz mit nur 826 Einträgen sehr klein. Dazu kommt noch, dass es davon nur 568 Studenten ernsthaft versucht, also mindestens ein Blatt abgeben, haben.
Zur Prognose der erwarteten, nächsten Leistungen in den Übungsaufgaben können diese auch verwendet werden, aber zur Prognose der Leistung in der Klausur können von denen sogar nur noch 297 Einträge genutzt werden.
Dazu kommen noch 49 Personen, die die Zulassung bereits erhalten haben und sich nicht darum bemüht haben, sie erneut zu erlangen. Für diese Studenten soll natürlich auch eine Vorhersage erstellt werden.

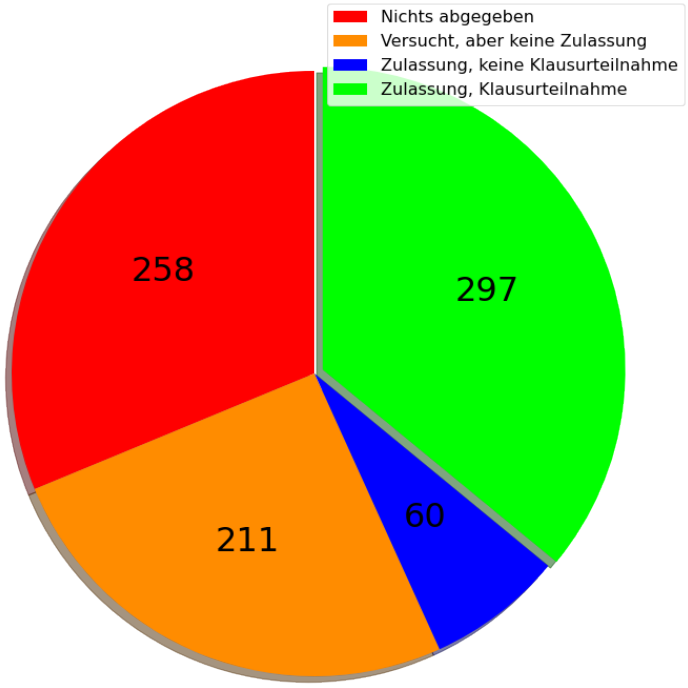

Abbildung 1: Übersicht über die Anzahl an Einträgen

Die Verteilung der Studenten auf Studiengänge wird in Abbildung 2 dargestellt. Hieraus wird ersichtlich, dass, wie zu erwarten, die Informatiker am meisten vertreten sind. Unerwartet ist aber, dass es sehr viele unzuordenbare Studenten gibt und die Informationswissenschaftler sogar die Mathematiker und Physiker zusammen übertreffen.

Aus der Abbildung 3 wird deutlich, dass sich mehr Personen in geraden Semestern befinden, als in den ungeraden. Da diese Veranstaltung nur im Wintersemester angeboten wird, haben sich wohl mehr Studenten zum Sommersemester eingeschrieben, als zum

Wintersemester.

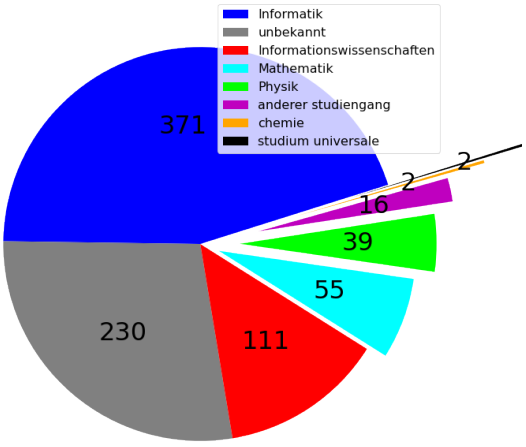

Abbildung 2: Verteilung der Studenten auf Studiengänge

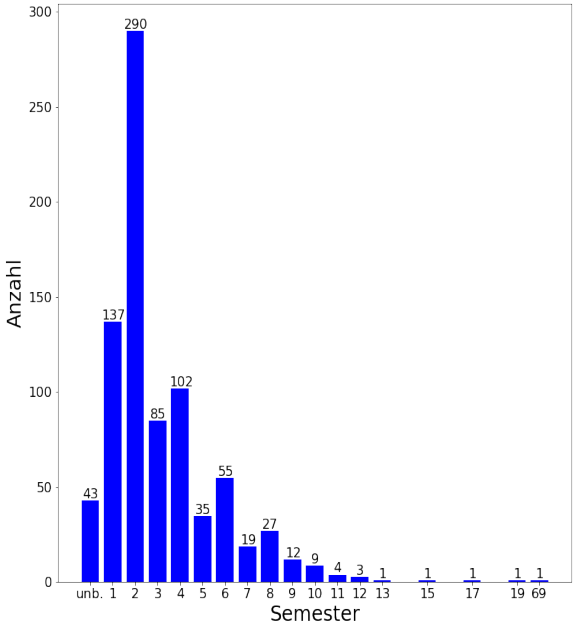

Abbildung 3: Verteilung der Studenten auf Semester

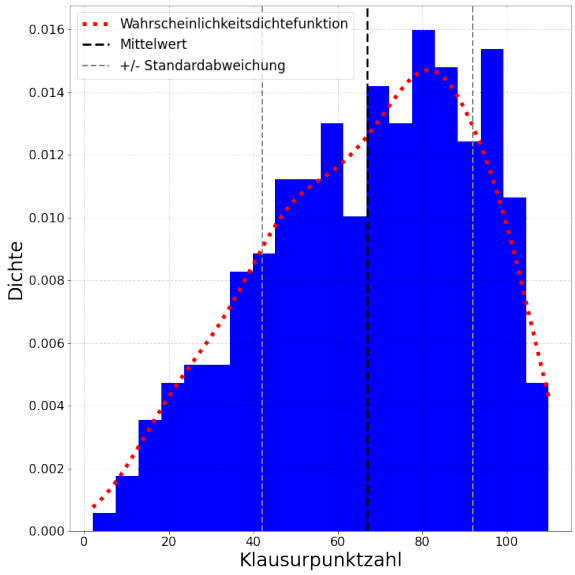

Abbildung 4: Verteilung der Klausurpunkte

Die Klausur haben erfreulicherweise weit mehr bestanden, als durchgefallen sind. Der Durchschnitt der Klausurpunktzahl liegt bei 60.6, die zugehörige Standardabweichung bei 30.8 und der Median bei 66. Die Masse der Studenten, die nur 0 Punkte erreicht haben sticht in Abbildung 4 besondern heraus. Das kann vermutlich viele Gründe haben, wie zum Beispiel eine Anmeldung ohne Erscheinen oder eine Anmeldung aus Neugier, wie die Klausuraufgaben aussehen werden. Als ein ernstgemeintes, realistisches Klausurergebnis kann das dann allerdings in der Vorhersage nicht gewertet werden.
Das Wichtigste ist aber, dass man anhand Abbildung 4 entnehmen kann, dass die erzielte Punktzahl, zumindest in dieser Klausur, annähernd normalverteilt ist, was später noch einmal relevant wird.

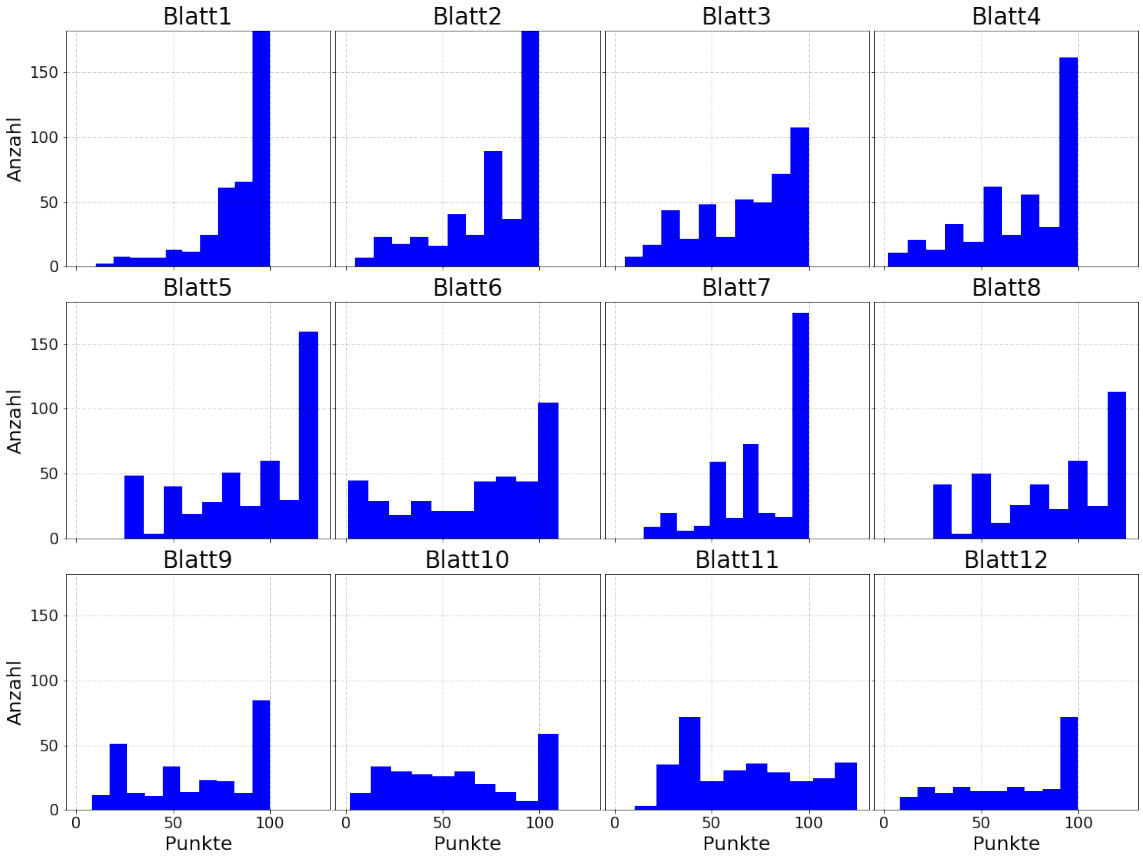

Abbildung 5: Punkteverteilung Blätter

Die Häufigkeiten der Punkte auf den Blättern scheinen keiner Verteilungsfunktion zu folgen. Einige Blätter sehen zwar gleichverteilt aus, aber nach dem Hypothesentest aus Anhang A liegt das aber eher am Zufall.

2.2 Verarbeitung

Die Regression kann nicht direkt auf den Datensatz losgelassen werden, denn mit *Strings* und *NaN*s ist es unmöglich zu rechnen. Daher müssen die Daten erst weiterverarbeitet werden. Hier werden nun Wege vorgestellt um das sinnvoll umzusetzen.

2.2.1 Verwendbare Features

Von den in Tabelle 1 aufgelisteten *Features* sind `name`,`tutor1` und `tutor2` generierte, anonyme Namen und werden deshalb verworfen.
Die Spalten `blattN-aufgabeM` werden auch nicht benutzt, weil sie zu detailliert sind und es in anderen Veranstaltungen höchstwahrscheinlich eine andere Anzahl an Aufgaben für die Übungsblätter gibt.
Des Weiteren werden andere Klausuren auch nicht die gleiche Aufgabenstruktur haben, welche aus „algorithmus“, „arrays“, „ausgaben“, „codeanalyse“, „funktionen“, „multiple-choice“, „oo“, „schleifen“ und „tests“ besteht. Also wird auch `klausur-X` nicht mehr benötigt.
Nun exisiteren nur noch die Attribute, mit denen die erwarteten Leistungen in den nächsten Übungsaufgaben und in der Klausur vorhergesagt werden könnten.
Im Folgenden werden die Einträge aber noch weiterverarbeitet.

2.2.2 Faktorisierung

Wie Abbildung 2 zeigt, sind Informatik, Informationswissenschaften, Mathematik und Physik am meisten vertreten. Sie werden in ihren Gruppen belassen und weniger repräsentierte Fächer unter der Gruppe „Andere“ zusammengefasst.
Nun sind *Strings* für eine Regression immer noch nicht zulässig. Deswegen werden sie mithilfe der Funktion in Gleichung (1) faktorisiert.

$$f(fach) = \begin{cases} 0 & \text{,falls } fach = \text{“}informatik\text{”} \\ 1 & \text{,falls } fach = \text{“}mathematik\text{”} \\ 2 & \text{,falls } fach = \text{“}physik\text{”} \\ 3 & \text{,falls } fach = \text{“}informationswissenschaften\text{”} \\ 4 & \text{,sonst} \end{cases} \tag{1}$$

Das Gleiche muss auch noch für `klausur` gemacht werden:

$$f(klausur) = \begin{cases} 1 & \text{,falls } klausur = \text{“} : hauptklausur\text{”} \\ 2 & \text{,falls } klausur = \text{“} : nachklausur\text{”} \end{cases} \tag{2}$$

2.2.3 NaN's

In Tabelle 1 stehen selbst nach dem Entfernen einiger Attribute immer noch *NaN*s (*Not-a-Number* Werte). Diese können in dem Programm nicht verwendet werden, aber Entfernen würde den ohnehin schon kleinen Datensatz noch weiter verkleinern. Außerdem liefern die *NaN*s auch Informationen mit:
Wenn es keine Werte in `summe-blattN`, `klausur-punkte`, `pruefungsversuch` oder `zugriffe-auf-ilias` gibt, bedeutet das, dass dort nichts gemacht wurde. Somit werden sie durch 0 ersetzt. Wenn keine Informationen zu `semester` vorliegen, könnte man betroffene Studenten entweder einem zufälligen Semester zuweisen, oder nur dem ersten Semester. Dies würde sie aber einer Gruppe zuweisen, in der sie sich gar nicht befinden und die Korrelation zur erbrachten Leistung könnte geschwächt werden. Deswegen werden sie einem „nullten" Semester zugewiesen. Es kann zwar nicht im „nullten" Semester studiert werden, aber es ist zumindest eine gute Möglichkeit, um diese Einträge nicht zu verwerfen. *NaN*s in *summe-klausur-punkte* und `note` werden provisorisch beibehalten, denn auch Studenten, die die Klausur nicht geschrieben haben, liefern immer noch Informationen über das Abgabeverhalten. So kann für die Klausurvorhersage leicht selektiert werden, wer die Klausur geschrieben hat und wer nicht.
Nachdem alle *Strings* und *NaN*s ersetzt wurden, können alle *Features* durch die Datentypen *integer* oder *float* repräsentiert werden.
In Tabelle 2 stehen nur noch die verwendbaren und umgeformten Attribute.

Tabelle 2: Verwendbare *Features* des Datensatzes

Name	Zahlenbereich	Beschreibung
summe-blatt$N$	$\mathbb{N}_0$	$N \in [1, 12]$, erreichte Punktzahl auf Blatt N
summe-uebungs-punkte	$\mathbb{N}_0$	$\sum_{n \in N}$ (summe-blatt$n$)
zugriffe-auf-ilias	$\mathbb{N}_0$	Summe der Zugriffe auf das *ILIAS*
fach	$\{0, 1, 2, 3, 4\}$	Studiengang des Studenten
semester	$\mathbb{N}_0$	Semester in dem die Veranstaltung besucht wurde
pruefungsversuch	$\{0, 1, 2, 3\}$	Nummer des Prüfungsversuchs
klausur	$\{0, 1, 2\}$	Teilnahme an Hauptklausur, Nachklausur oder keiner
summe-klausur-punkte	$\mathbb{R}_0^+ \cup NaN$	erreichte Punktzahl in der Abschlussklausur
note	$\{1.0, 1.3, 1.7, 2.0, 2.3, 2.7, 3.0, 3.3, 3.7, 4.0, 4.3, 4.7, 5.0\} \cup NaN$	erhaltene Note in der Abschlussklausur

2.2.4 Wahl der Zielgröße

Das offene Problem ist jetzt noch die Zielgröße beziehungsweise die Ausgabe der Vorhersage.
Es stellt sich die Frage: „Was soll eigentlich vorhergesagt werden?“
Die naheliegendste Idee ist natürlich die Note.
Aber Fächer wie „Professionelle Softwareentwicklung“ oder „Softwareentwicklung im Team“ vergeben keine Noten, trotzdem sollen sie von diesem System profitieren und nicht davon ausgeschlossen werden.
Dazu kommt noch, dass die Noten in dem gegebenen Datensatz sehr inkonsistent sind. Bei gleicher Punktzahl gibt es unterschiedliche Noten, wie Abbildung 6 zeigt.

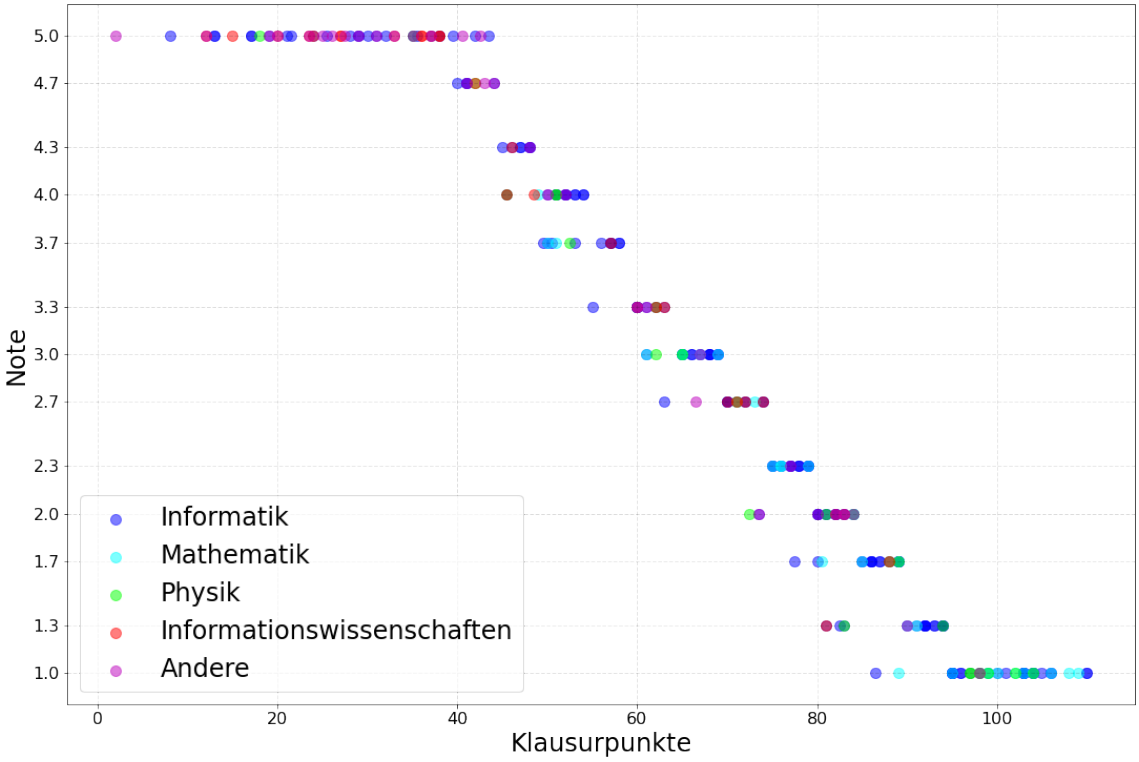

Abbildung 6: Punktzahl in der Klausur und die erhaltene Note

Die Lösung ist, nicht die Klausurnote vorherzusagen, sondern die Klausurpunkte.
Dadurch wird auch noch die Vorhersage besser, da man nicht mehr an die 11 Notenschritte $\{1.0, 1.3, 1.7, 2.0, 2.3, 2.7, 3.0, 3.3, 3.7, 4.0, 5.0\}$ gebunden ist, sondern genauere Schritte zwischen keine Punkte und Maximalpunktzahl hat.
Da aber die Klausuren unterschiedliche maximal erreichbare Punkte haben, wird eine vergleichbare Größe, wie die Note, benötigt.

Wird eine Note benötigt, wird die Punktzahl der Klausur vorhergesagt und dann mittels der Formel in Gleichung (3) in eine Note umgewandelt.

$maxPunkte$ = Maximalpunktzahl
$erreichtePunkte$ = erreichte Punktzahl in der Klausur

$$punkte = \left\lfloor 20 \cdot \frac{erreichtePunkte}{maxPunkte} \right\rfloor$$

$$note = \begin{cases} 5.0 & \text{,falls } punkte < 10 \\ 4.0 & \text{,falls } punkte = 10 \\ 3.7 & \text{,falls } punkte = 11 \\ 3.3 & \text{,falls } punkte = 12 \\ 3.0 & \text{,falls } punkte = 13 \\ 2.7 & \text{,falls } punkte = 14 \\ 2.3 & \text{,falls } punkte = 15 \\ 2.0 & \text{,falls } punkte = 16 \\ 1.7 & \text{,falls } punkte = 17 \\ 1.3 & \text{,falls } punkte = 18 \\ 1.0 & \text{,falls } punkte \geq 19 \end{cases} \tag{3}$$

2.3 Zusammenhänge

Nun soll der Zusammenhang der Einflussgrößen mit den Zielgrößen verglichen werden, um zu sehen, welche *Features* eine größere Rolle spielen, als andere. Je stärker der Zusammenhang mit den Zielgrößen ist, desto genauer wird die Vorhersage. Deswegen müssen die *Features* gefunden werden, die den stärksten Zusammenhang erreichen.

Die Korrelation (Zusammenhang) zweier *Feature*vektoren wird durch den Korrelationskoeffizienten sehr gut dargestellt.

2.3.1 Berechnung des Korrelationskoeffizienten

Der Korrelationskoeffizient von zwei Vektoren x und y berechnet sich nach *Karl Pearson* [Spe61] durch

$\bar{x} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i$ (Mittelwert von x)

$\bar{y} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} y_i$ (Mittelwert von y)

$$r_{pearson}(x,y) = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2 \cdot \sum_{i=0}^{n-1}(y_i - \bar{y})^2}} \tag{4}$$

In Abschnitt 2.1.2 wurde bereits festgestellt, dass die Klausurpunkte dieser Klausur normalverteilt sind. Da es aber nicht gegeben ist, dass dieses Auftreten immer der Fall ist und zusätzlich nicht gegeben ist, dass der Zusammenhang zwischen den Einflussgrößen und den Klausurpunkten unbedingt linear ist, wird anstatt der Berechnung nach *Pearson*, die Berechnung nach *Charles Spearman* [Spe61] verwendet, weil dieser robuster gegen Ausreißer ist und vor allem auch nicht-lineare Zusammenhänge erlaubt. Sie baut aber auch auf der Berechnung von *Pearson* auf, wie anhand von Gleichung (5) zu erkennen ist:

$\overline{rang(x)} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} rang(x_i)$ (Mittelwert aller Ränge von x)

$\overline{rang(y)} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} rang(y_i)$ (Mittelwert aller Ränge von y)

$$r_{spearman}(x,y) = \frac{\sum_{i=0}^{n-1}(rang(x_i) - \overline{rang(x)})(rang(y_i) - \overline{rang(y)})}{\sqrt{\sum_{i=0}^{n-1}(rang(x_i) - \overline{rang(x)})^2 \cdot \sum_{i=0}^{n-1}(rang(y_i) - \overline{rang(y)})^2}} \tag{5}$$

Der Rang wird berechnet, indem alle Werte des Vektors aufsteigend sortiert werden. Der Rang ist dann der Index des jeweiligen Wertes. Haben mindestens zwei den gleichen Wert (*Tie*), so ist der Rang genau der Mittelwert aller betreffenden Werte.

Korrelationskoeffizienten um 1 und -1 deuten auf eine starke Korrelation und Korrelationskoeffizienten um 0 auf eine schwache Korrelation hin. Wenn die Zahl negativ ist, dann

bedeutet das, dass je niedriger der Wert aus x, desto höher ist der entsprechende Wert aus y.

Es ist aber auch möglich, dass mehr als ein *Feature* eine höhere Korrelation mit der jeweiligen Zielgröße erreicht. Dann tritt der Fall ein, dass die Kombination dieser *Features* einen noch höheren Zusammenhang erreichen kann, als einzeln betrachtet. Dafür wird der multiple Korrelationskoeffizient verwendet, der den Zusammenhang bestimmt, den zwei Vektoren x_1, x_2 zusammen mit einem dritten Vektor y haben:

$$r(x_1, x_2, y) = \sqrt{\frac{r_{x_1,y}^2 + r_{x_2,y}^2 - 2 \cdot r_{x_1,x_2} \cdot r_{x_1,y} \cdot r_{x_2,y}}{1 - r_{x_1,x_2}^2}} \tag{6}$$

Aus Gleichung (6) geht hervor, dass der multiple Korrelationskoeffizient größer wird, wenn die Vektoren x_1 und x_2 jeweils mit y eine hohe Korrelation erreichen, aber auch dadurch verringert wird, wenn x_1 und x_2 miteinander eine hohe Korrelation erreichen. Die Korrelation der Kombination dieser beiden wird also maximal, wenn sie jeweils einen starken Zusammenhang zu y erreichen, aber gegenseitig unabhängig sind. Insgesamt wird die Vorhersage also nur deutlich verbessert, wenn neue *Features* hinzugenommen werden, die keinen Zusammenhang mit den schon verwendeten *Features* aufweisen.

2.3.2 Korrelationen der Übungsblätter

Wird $r_{spearman}(x, y)$ aus Gleichung (5) auf die *Features* angewendet, die für die Vorhersage der Leistungen in den nächsten Übungsaufgaben dienen, so erhält man die Korrelationskoeffizientenmatrix Abbildung 7.
Die `zugriffe-auf-ilias` sind leider nicht in Schritten, sondern nur als Summe am Ende des Semesters vorhanden, weswegen sie nicht mit den Blättern verglichen werden können.

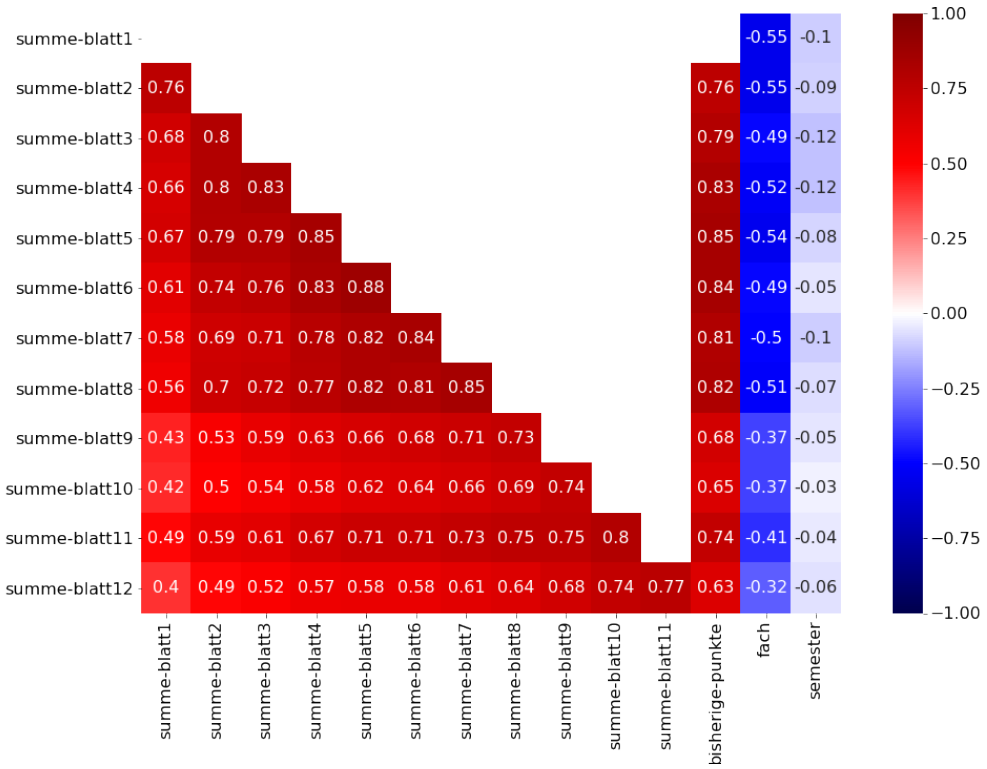

Abbildung 7: Korrelationskoeffizienten `summe-blattN` mit vorherigen Blättern und weiteren *Features*

Leider liefern `fach` aber vor allem `semester` nicht den gewünschten Zusammenhang mit den erreichten Punkten in den jeweiligen Übungsblättern. Das `semester` ist mit den Übungspunkten unkorreliert und `fach` erreicht nur eine mittlere Korrelation, wobei noch angemerkt werden muss, dass ein starker Sprung ab dem neunten Blatt stattfindet.
Die, auf der Nebendiagonalen zu sehenden, Korrelationskoeffizienten sind alle ≥ 0.73, was für eine starke Korrelation zwischen dem aktuellen und dem vorherigen Blatt spricht. Für jedes Übungsblatt hat also die erreichte Punktzahl auf dem vorherigen Blatt die größte Aussagekraft und je weiter ein Blatt in der Vergangenheit liegt, desto geringer ist der Zusammenhang. Trotzdem sieht man auch hier einen Sprung ab dem neunten Blatt, weil viele Studenten aufhören, sobald sie die Klausurzulassung erreicht haben, was bei den meisten Studenten auf dem achten Blatt geschieht.
Werden die Zusammenhänge letzten **beiden** Blätter kombiniert und mit dem aktuellen Verglichen ergeben sich entsprechende multiple Korrelationskoeffizienten von 0.76, 0.81, 0.87, 0.88, 0.91, 0.89, 0.89, 0.77, 0.77, 0.83 und 0.81, welche somit durchgehend bessere Werte erzielen, als nur das letzte Blatt allein. Das `fach` und die Summe der bisherigen Punkte, die ein wenig schlechter abschneiden, als das letzte allein Blatt betrachtet, erreichen auch eine bessere multiple Korrelation, wenn sie mit dem letzten Blatt zusammen

betrachtet werden. Sie sind aber beide immer noch schlechter als die Kombination der letzten zwei Blätter.
Insgesamt folgt, dass das `semester`, `fach` und auch die `bisherige-punkte` nicht mehr für die Regression genutzt werden, weil die die letzten zwei Blätter einen höheren Zusammenhang besitzen. Ob noch das drittletzte Blatt hinzugenommen werden sollte, kann erst in Abschnitt 3.2.2 geklärt werden.

2.3.3 Korrelationen mit Klausurpunktzahl

Was noch fehlt, ist die Korrelationen für die Vorhersage der `klausur-punkte`. Wird $r_{spearman}(x, y)$ aus Gleichung (5) auf die *Features* angewendet, die für die Vorhersage der Leistungen in der Klausur dienen, so erhält man die Korrelationskoeffizientenmatrix Abbildung 8. Diese Analyse soll dazu dienen, die *Features* zu finden, aus denen die meisten Informationen für die Klausurvorhersage genommen werden können, also werden nur Daten von Studenten genutzt, die einen ernsthaften Klausurversuch unternommen haben.

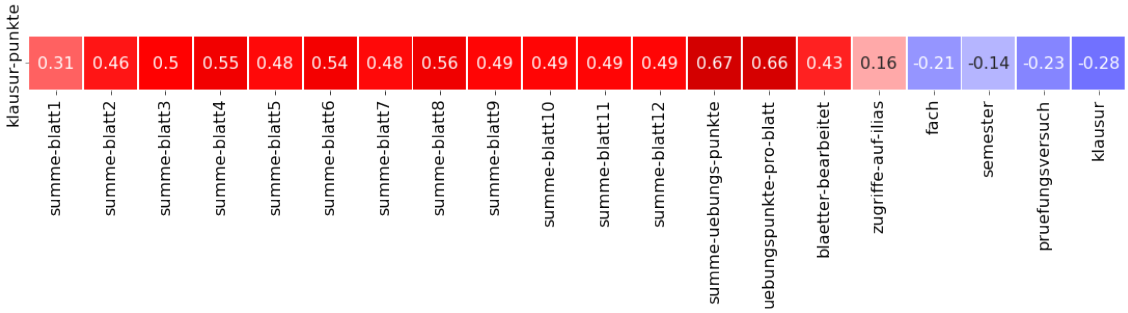

Abbildung 8: Korrelationskoeffizienten Klausurpunkte

Hier erreichen die `summe-blattN` nur eine mittlere Korrelation. Noch schlimmer sieht es für `zugriffe-auf-ilias`, `fach`, `semester`, `pruefungsversuch` und `klausur` aus. Sie scheinen alle keinen Zusammenhang mit der Klausurpunktzahl zu haben. Das beste *Feature*, mit einem Korrelationskoeffizienten von 0.67, sind die `summe-uebungspunkte`. Zwei weitere *Features* stechen auch heraus: `uebungspunkte-pro-blatt` und `blaetter-bearbeitet`. In `uebungspunkte-pro-blatt` steht die durchschnittliche Punktzahl aller **bearbeiteten** Blätter. Sie repräsentieren also die Information, in wie vielen Blättern ein Student die Zulassung erreicht hat. Dies wird benötigt, um die Studenten aufzuteilen, die nur um 600 Punkte erreicht haben, also aufgehört haben, sobald die Zulassung erworben wurde. Die Idee der beiden stammt von meinem Vorgänger Johannes Müller [Mü18] und lautet, dass ein Student, der in wenigen Blättern sehr viele Punkte erreicht hat und aufhört sobald er die Klausurzulassung erreicht hat, die Themen dieser Blätter sehr

gut verstanden hat und weniger für die Klausur lernen muss.
Die Information, in wie vielen Blättern ein Student die Zulassung erreicht hat (`uebungspunkte-pro-blatt`), ist demnach essentiell wichtig und wird zusammen mit der Information, wie viele Punkte er insgesamt erreicht hat (`summe-uebungs-punkte`) für die Regression verwendet. Beide zusammen verwendet, erreichen einen multiplen Korrelationskoeffizienten von 0.694. Die Blätter im Einzelnen betrachtet, haben einen geringeren Zusammenhang, als diese beiden und verschlechtern deswegen den multiplen Korrelationskoeffizienten und somit auch die Genauigkeit der Vorhersage. Unter weiterer Verarbeitung können sie gegebenenfalls noch mitverwendet werden. Ob das einen Nutzen hat, kann aber erst später, in Abschnitt 3.1.7, geklärt werden.

Mit Abbildung 9 zeigt sich nochmal, dass es in Abschnitt 2.2.4 eine gute Entscheidung war, die Punktzahl statt der Note vorherzusagen. Der Zusammenhang mit der Klausurnote ist für jedes *Feature* kleiner.

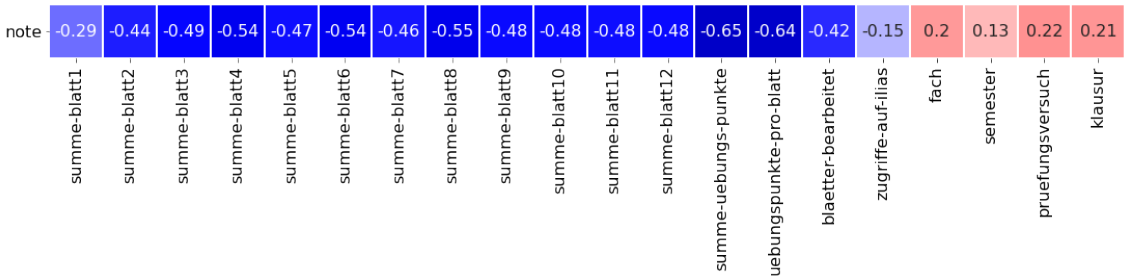

Abbildung 9: Korrelationskoeffizienten Klausurnote

In Abbildung 10 werden die Werte aus Abbildung 8 nochmal genauer als Plot dargestellt, um die Werte bildlich nachvollziehen zu können.

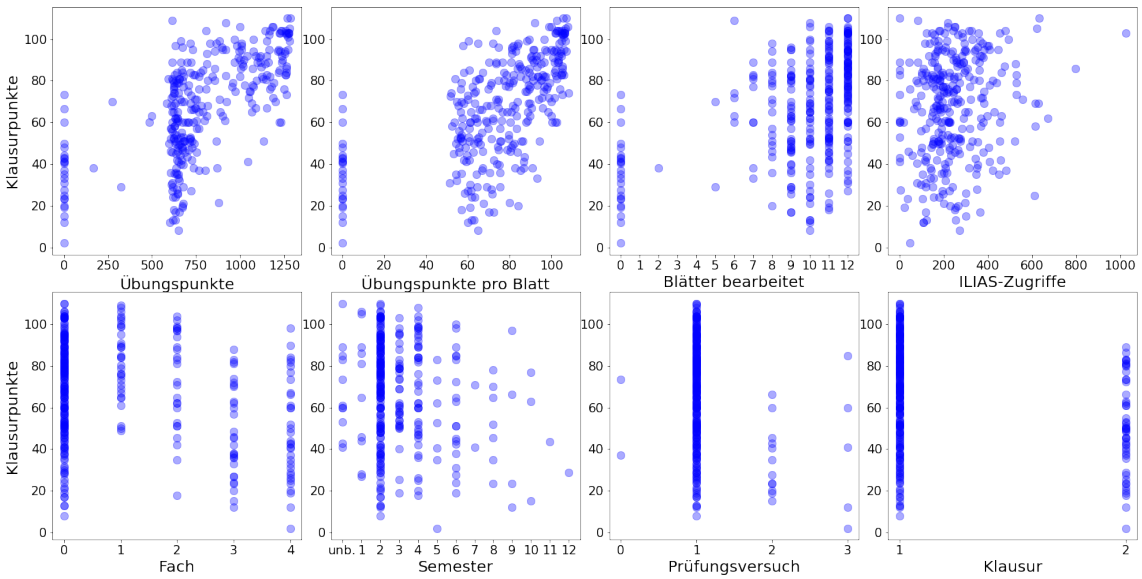

Abbildung 10: Korrelationen Klausurpunkte genauer

Aus Abbildung 8 und Abbildung 10 geht hervor, dass die Summe aller Übungspunkte das wichtigste *Feature* ist. Eine genauere Ansicht in Abbildung 11, in der die erreichte Punktzahl in der Klausur gegenüber den erreichten Übungspunkten aufgetragen wird, ergibt, dass sich der Zusammenhang mit einem Korrelationskoeffizienten von 0.67 bestätigt und sogar schon eine grobe lineare Schätzerfunktion erkannt werden kann.

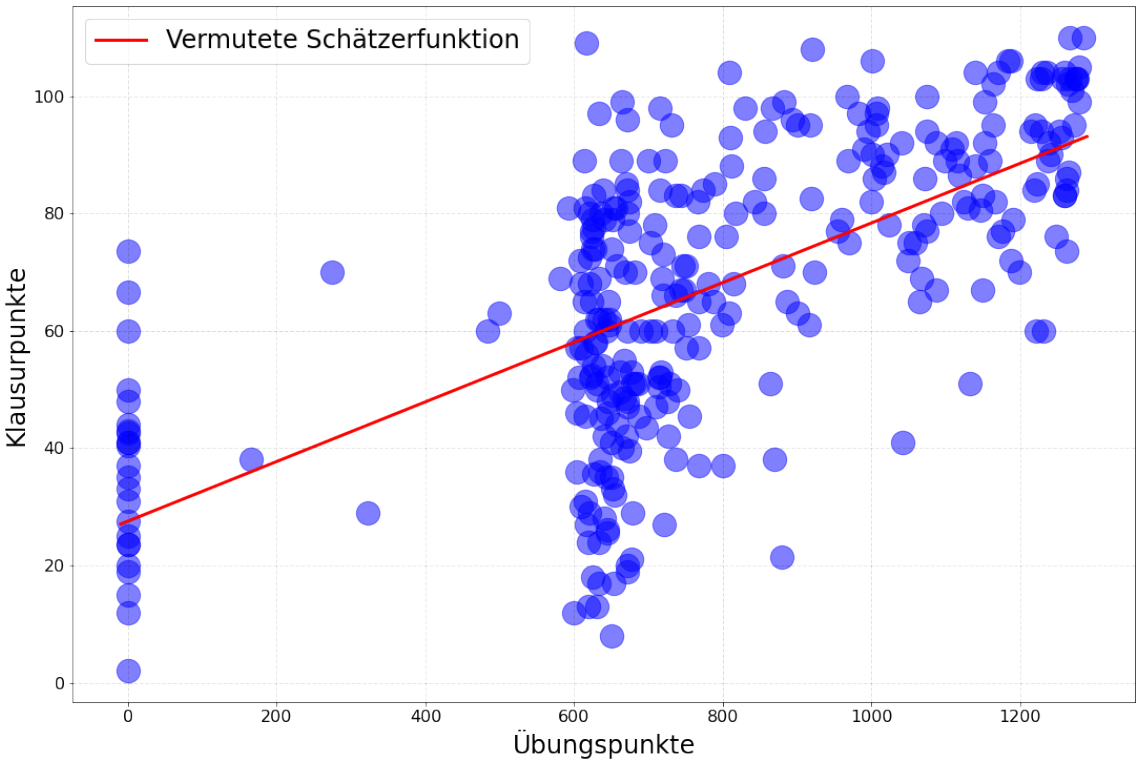

Abbildung 11: Übungspunkte mit zugehöriger erreichter Punktzahl in der Klausur

Die meisten Studenten liegen zwar in der Nähe der vermuteten, linearen Schätzerfunktion, aber es ist auch zu erkennen, dass die Studenten im mittleren Bereich zwischen 600 und 900 Übungspunkten stark abweichen können. Durch Hinzunahme einer weiteren Dimension, der Übungspunkte pro bearbeitetes Blatt, wird dieser Bereich aber noch weiter aufgeteilt.
Nachdem nun auch noch aus diesem Datensatz die bestmöglichen Einflussgrößen mit ihren Zusammenhängen zu den Zielgrößen gefunden wurden, kann endlich mit der Regression, also der Vorhersage, begonnen werden.

3 Vorhersage

In diesem Kapitel wird der Hauptteil der Anwendung, die Vorhersage der Klausurnote und der erwarteten nächsten Leistungen in den zukünftigen Aufgaben, entwickelt.
Dafür werden mathematische Methoden aus dem maschinellen Lernen verwendet, einem Teilgebiet der statistischen Datenanalyse.

3.1 Endgültige Vorhersage der Leistung in der Klausur

Zunächst wird vorausgesetzt, dass alle Übungsaufgaben beendet und korrigiert wurden, also alle Daten genutzt werden können. In diesem Unterkapitel geht es nur um die voraussichtliche Klausurpunktzahl und wie anhand dieser der Schätzer schrittweise weiterentwickelt wird.

3.1.1 Lineare Regression

Betrachtet werden zunächst eindimensionale Abbildungen. Die Normalform einer linearen Funktion lautet

$$y = m \cdot x + b \tag{7}$$

bei der m und b gegeben sind. In der Linearen Regression haben wir aber nun die Vektoren $\vec{x}$ und $\vec{y}$ gegeben und es werden m und b gesucht.
Es sollen die Werte für m und b gefunden werden, welche genau durch die Datenpunkte gehen. Die Abweichung, hier quadratisch, soll also **minimal** sein.

$$\{m, b\} = \arg\min_{m,b} \sum_{i=0}^{n-1} ((m \cdot x_i + b) - y_i)^2 \tag{8}$$

Die Umformung im Anhang B.1 ergibt, dass

$$\begin{aligned} m &= \frac{\sum_{i=0}^{n-1} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n-1} (x_i - \bar{x})^2} \\ b &= \bar{y} - m \cdot \bar{x} \end{aligned} \tag{9}$$

mit $\bar{x} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i$ und $\bar{y} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} y_i$ also den jeweiligen Mittelwerten.

Wird nun für $\vec{x}$ die `summe-uebungs-punkte` und für $\vec{y}$ die `klausur-punkte` in Gleichung (9) eingesetzt erhält man $m = 0.05081465$ und $b = 27.54752377733802$, dessen resultierende Gerade schon in Abbildung 11 eingezeichnet wurde.

Zunächst müssen die beiden Mittelwerte ausgerechnet werden. Diese Rechnung erfolgt schon in $\mathcal{O}(n)$. Danach müssen die Einträge erneut durchlaufen werden, um die zwei Summen in m zu berechnen. Das ist erneut in $\mathcal{O}(n)$ realisierbar. Alles weitere sind nur noch einfache Operationen. Insgesamt ist die Laufzeit um den Parameter m zu bestimmen also $\in \mathcal{O}(n)$.

3.1.2 Lineare Regression mehrerer Prädikatoren

Für die Regression soll aber nicht nur `summe-uebungs-punkte` verwendet werden. In Abschnitt 2.3.3 wurde festgestellt, dass auch `uebungspunkte-pro-blatt` eine hinreichend signifikante Korrelation von 0.66 und beide zusammen einen multiplen Korrelationskoeffizienten von 0.694 mit den Klausurergebnissen erreichen. Um beide *Features* gleichzeitig verwenden zu können, muss die lineare Regression umgeformt werden.
Jetzt ist x kein Vektor $\vec{x}$, sondern eine Matrix X der Größe $[n \times (p+1)]$ und y eine Matrix Y der Größe $[n \times 1]$.

$$Y = X \cdot M \tag{10}$$

Gesucht eine Matrix M $[(p+1) \times 1]$ in der p die Anzahl der *Features ist*. Wenn `summe-uebungs-punkte` und `uebungspunkte-pro-blatt` genommen wird, gilt $p = 2$. Zusätzlich existiert noch eine Achsenverschiebung, der *intercept*.

$$M = \arg\min_{m} \sum_{i=0}^{n-1} (x_i^T \cdot m - y_i)^2 \tag{11}$$

Die Umformung im Anhang B.2 ergibt folgende Gleichung für M:

$$M = (X^T X)^{-1} X^T Y \tag{12}$$

Sei nun X eine Matrix, der Form:
$[1,$ `summe-uebungs-punkte`^{T}, `uebungspunkte-pro-blatt`$^T]$.
Dann ergibt sich Für M folgende $[3 \times 1]$ Matrix: $\begin{pmatrix} 25.46177543547452 \\ 0.04087739 \\ 0.13131597 \end{pmatrix}$

Der 3D-Plot in Abbildung 12 zeigt die beiden *Features* `summe-uebungs-punkte` und `uebungspunkte-pro-blatt` mit zugehörigen `klausur-punkte`. Zusätzlich wurde die Ebene, die durch M definiert wird, eingezeichnet.

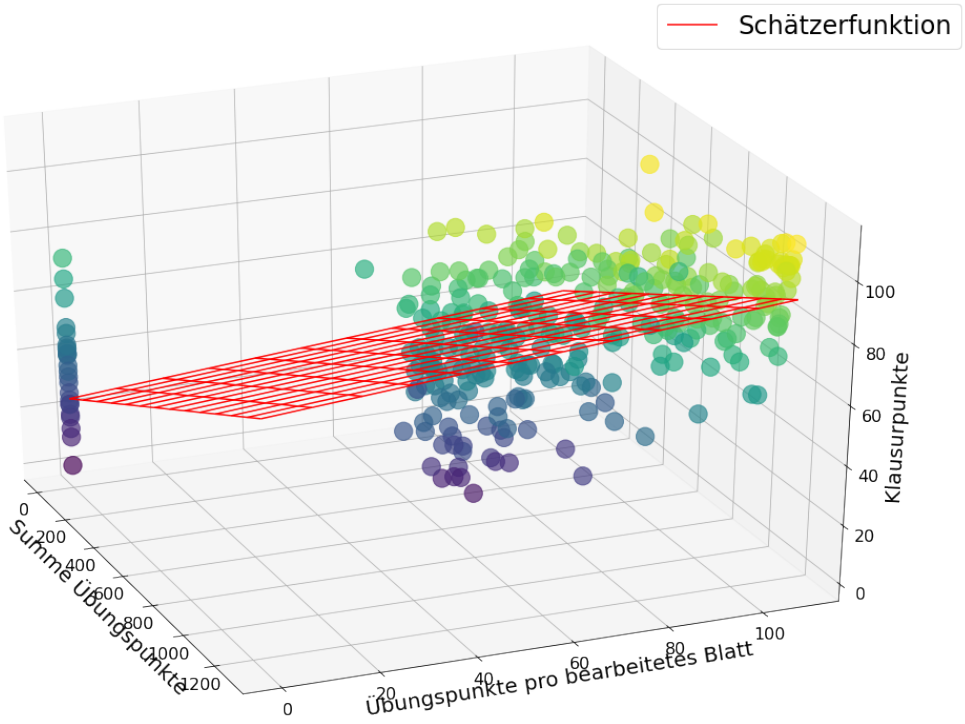

Abbildung 12: Datenpunkte der wichtigsten *Features* mit eingezeichneter Schätzerfunktion

Die Berechnung von M im Allgemeinen erfolgt laut [PVG+11] in $\mathcal{O}(n \cdot p^2)$.

3.1.3 Lineare Regression mehrerer Prädikatoren nicht-linearer Zusammenhänge

Es ist nicht gegeben, dass die oben genannten Attribute einen linearen Zusammenhang haben. Es sieht zwar in Abbildung 12 danach aus, aber das bedarf weiterer Untersuchung. Wenn die Matrix X durch alle möglichen polynomiellen Kombinationen, die es zu einem angegebenen Grad gibt, erweitert wird, können auch nicht-lineare Zusammenhänge von einer linearen Regression modelliert werden. Wird zum Beispiel der Grad 2 festgelegt, dann hat X die Form $[1, \vec{a}, \vec{b}, \vec{a}^2, \vec{a} \circ \vec{b}, \vec{b}^2]$ beziehungsweise auf diesen Datensatz angewendet:
$[1, \texttt{summe-uebungs-punkte}^T, \texttt{uebungspunkte-pro-blatt}^T,$
$\texttt{summe-uebungs-punkte}^2, \texttt{summe-uebungs-punkte}^T \circ \texttt{uebungspunkte-pro-blatt}^T,$
$\texttt{uebungspunkte-pro-blatt}^2]$
In Abbildung 13 wurde M mittels des, auf den zweiten Grad transformierten, Datensatzes bestimmt.

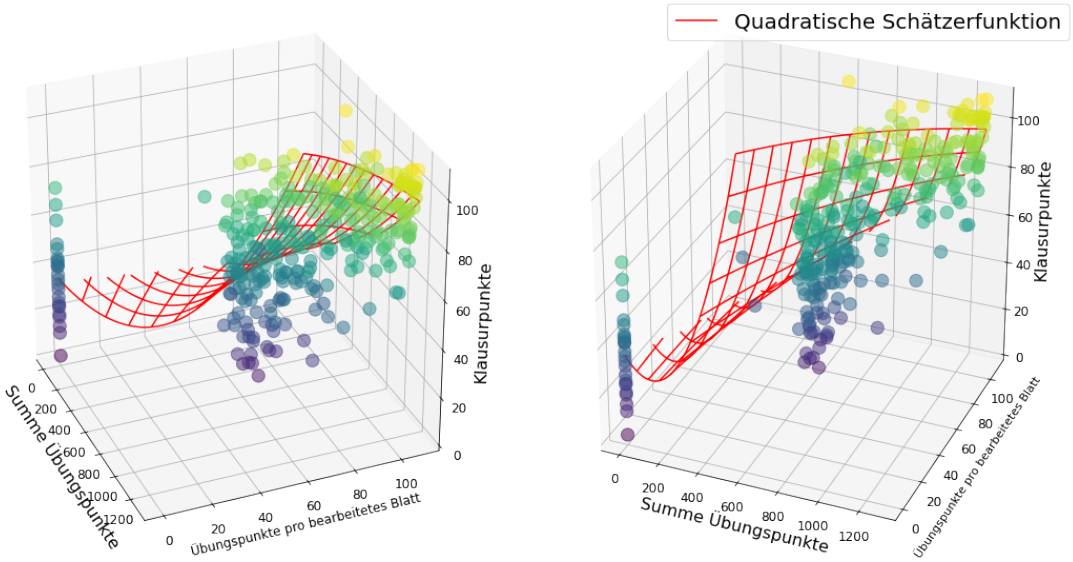

Abbildung 13: Datenpunkte mit nicht-linearer Schätzerfunktion

So werden dem Datensatz quasi nur „neue *Features*“ angehängt. Der Grad darf aber nicht zu hoch gewählt werden, da X auf $\binom{grad+anzahlFeatures}{grad}$ viele *Features*, also auch Spalten, transformiert wird und eben dieses exponentielle Wachstum des Datensatzes zu einer längeren Laufzeit führt.

Das nächste Problem, das ein hoher Grad mit sich bringt, ist das *overfitting*, bei dem eine zu genaue Vorhersage gemacht wird, sprich die Parameter der Vorhersagefunktion sind zu genau auf diesen Datensatz angepasst, wie in Abbildung 14 zwischen 100 und 500 Punkten zu sehen ist. Da in diesem Datensatz die Datenpunkte an diesen Stellen befinden und der Abstand zu allen **minimal** sein soll, so verläuft die Funktion genau zwischen den isolierten 5 Punkten. Ein Anstieg ab 0 Punkten und ein wieder Abfallen ab 175 Punkten ist aber nicht realistisch und bei neuen, unbekannten Eingaben wird die Genauigkeit der Vorhersage in diesem Bereich sehr schlecht sein.

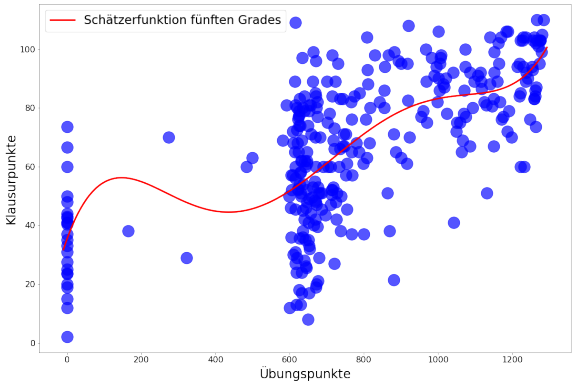

Abbildung 14: Overfitting zwischen 100 und 500 Übungspunkten

3.1.4 Dimensionsreduktion

Dass die einzelnen Blätter keinen Zusammenhang mit den Klausurergebnissen haben konnte in Abschnitt 2.3.3 nicht ausgeschlossen werden und eine Berechnung des Schätzers mit 14 Spalten hat zu hohe Kosten. Deswegen wird versucht, die Anzahl an *Features* zu verringern, ohne dass dabei zu viele Informationen verloren gehen. Die *principal component analysis* leistet genau das.
Die Dimension der Eingabe X wird reduziert, aber von der **Varianz** soll dabei so wenig wie möglich verloren gehen, beziehungsweise sie soll **maximiert** werden.
Durch die Umformung in Anhang D ergibt sich:

$$\begin{aligned} &\Lambda \text{ ist Diagonalmatrix mit Eigenwerten von } X^T X \\ &V \text{ ist Matrix in der jede Spalte ein zugehöriger Eigenvektor von } \Lambda \end{aligned} \tag{13}$$

Zur Erinnerung: X hat die Größe $[n \times p]$ ($p =$ Anzahl *Features*). Dann ist $X^T X$ eine $[p \times p]$ Matrix und dessen Eigenwertmatrix natürlich auch.

Die Eigenwerte auf der Hauptdiagonalen in Λ werden anschließend absteigend sortiert und die gleichen Spaltenvertauschungen werden auf V angewendet. Um nun die Dimension auf eine Zahl $d \in \mathbb{N}$ zu reduzieren, werden die ersten d Spalten aus V gewählt und diese Matrix der Größe $[p \times d]$ dann von rechts mit X multipliziert. Es ergibt sich eine Matrix der Größe $[n \times d]$, wie gewünscht.
Um die Dimension zu reduzieren müssen also nur die Eigenwerte und zugehörige Eigenvektoren gefunden werden. Durch eine QR-Zerlegung, zum Beispiel, kann das nach [Sch19] in $\mathcal{O}(\frac{2}{3}p^3)$ realisiert werden.

3.1.5 Vergleich verschiedener Schätzungsfunktionen

Zur Regression können *Features* hinzugenommen oder weggelassen werden und welchen Grad die mehrdimensionalen Zusammenhänge haben ist auch nicht gegeben. Deswegen wird ein Maß benötigt, mit dem verschiedene Vorhersagefunktionen untereinander verglichen und die Beste unter ihnen ausgewählt werden kann.

MAE

Die durchschnittliche absolute Abweichung (*mean absolute error*) wird durch die Funktion in Gleichung (14) bestimmt.

$$MAE(X,Y) = \frac{1}{n}\sum_{i=0}^{n-1}|\hat{y}_i - y_i| \tag{14}$$

wobei $\hat{y}_i = x_i^T \cdot M$, also der vorhergesagte Wert für die Datenreihe i in X.

Der MAE ist sehr wichtig, weil er am verständlichsten ist und deswegen den Betroffenen als Rückmeldung für die Genauigkeit der Vorhersage dient.

MSE

Die durchschnittliche, quadratische Abweichung (*mean squared error*) wird durch die Funktion in Gleichung (15) bestimmt. Nach dieser Abweichung wird auch die lineare Regressionsfunktion minimiert. Je kleiner der MSE, desto genauer die Vorhersage.

$$MSE(X,Y) = \frac{1}{n}\sum_{i=0}^{n-1}(\hat{y}_i - y_i)^2 \tag{15}$$

R^2

In Abschnitt 2.3.1, Gleichung (4) wurde der Korrelationskoeffizient nach *Pearson* definiert. Wird er quadriert (Umformung im Anhang C), ergibt sich das Bestimmtheitsmaß R^2, welcher in Gleichung (16) zu sehen ist.

$$R^2(X,Y) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{16}$$

Ähnlich, wie bei dem ursprünglichen Korrelationskoeffizienten von Pearson, ergibt eine starke Korrelation, beziehungsweise hier eine exakte Schätzung von Y, einen R^2 nahe 1 und eine schlechte Schätzung einen R^2 nahe 0.

Wenn immer mehr *Features* hinzugenommen werden, sollte theoretisch die Vorhersage immer genauer werden. Gleichzeitig *overfittet* die Vorhersage auf den gegebenen Daten und das Ausrechnen von den Schätzungsparametern M dauert länger, wie in Abschnitt 3.1.3 bereits festgestellt.
Der adjustierte R^2-Wert ist für dieses Problem eine bessere Wahl, da er auch noch die Anzahl der verwendeten Vektoren mit einbezieht.

$$\overline{R^2(X,Y)} = 1 - (1 - R^2(X,Y))\frac{n-1}{n-p-1} \tag{17}$$

Je mehr voneinander unabhängige Variablen für die Lineare Regression verwendet werden, desto kleiner wird der adjustierte R^2.

3.1.6 Train-Test-Split

Die existierenden Werte für die Klausurpunktzahl werden benutzt, um den Schätzer zu bestimmen. Dann werden die jeweiligen Ausgaben mittels dieses Schätzers prophezeit und diese vorhergesagten Werte werden dann mit den tatsächlichen verglichen. Daraus folgt aber, dass die ursprünglichen, tatsächlichen Werte dafür benutzt wurden, sich selbst vorherzusagen.
Das wird ganz einfach gelöst, indem der Datensatz auf Trainings- und Testdaten aufgeteilt wird. Die Parameter des Schätzers werden durch das *Trainingsset* bestimmt und anschließend werden die Testdaten dafür genutzt, um die Vergleichswerte aus vorherigem Abschnitt 3.1.5 zu bilden.
Damit die Wahl der Aufteilung auf Trainings- und Testdaten nicht vom Zufall abhängt, so wird jeder Eintrag, also jede Zeile, einmal getestet und auf allen anderen wird trainiert. Diese Methode heißt *Leave one out*, weil immer ein Eintrag aus dem *Trainingsset* ausgelassen wird.
Das heißt aber auch, dass die Kosten dieser Art des Testens enorm hoch sind, weil genauso viele Regressionen gebildet werden, wie es Stichproben gibt. Deswegen kann diese Art der Aufteilung auch nur durchgeführt werden, weil der Datensatz, vor allem für die Vorhersage der Klausurergebnisse, sehr klein ist.

Diese Methode ist erwartungstreu/unverzerrt (*unbiased*), weil alle möglichen Kombinationen betrachtet werden, im Gegensatz zu Johannes Müllers Arbeit [Mü18], in der die Aufteilung auf 5 Blöcke vom Zufall abhängt und es somit immer eine Abweichung gibt.
Da diese Aufteilung nur zur Aufstellung der Vorhersagemethoden dient und nicht in der endgültigen Vorhersage verwendet wird, stellt die stark erhöhte Laufzeit kein Problem dar.

3.1.7 GridSearch

Nun existiert ein geeignetes Maß, um Schätzungen zu vergleichen und es werden direkt mehrere lineare Regressionen mit verschiedenen Kombinationen von Eingabe*features* und Graden ausprobiert. Dafür werden die besten Parameter auf einer Art Gitter gesucht, also jede Kombination von *Features*, Graden und, wenn `summe-blattN` genutzt wird, auch

die Dimension der *PCA* durchlaufen. Pro *Feature*kombination sind die besten Ergebnisse in Tabelle 3 aufgelistet.

Im Gegensatz zu Johannes Müller [Mü18], der in seiner Arbeit die *PCA* zwischen 1 bis 3 Dimensionen unbegründet verpflichtend gemacht hat, werden in dieser *GridSearch* auch andere Kombinationen zugelassen. Dazu kommt noch, dass auch höhere Dimensionen der *PCA* erlaubt werden und Transformationen, statt bis zum siebten Grad, bis zum zehnten Grad stattfinden. Durch höhere Dimensionen der *PCA* wird auch die Anzahl der Eingabe*features* größer. Dadurch, dass hier aber der adjustierte R^2 verwendet wird, wird das mit einkalkuliert.

Tabelle 3: Ergebnisse der *GridSearch* auf *Features*, Grad und ggf. *PCA*-Dimension

Features	Bester Grad	$\overline{R^2}$	MSE	MAE
`summe-uebungs-punkte`	7	0.4463	340.659	14.615
`uebungspunkte-pro-blatt`	3	0.4259	353.205	14.957
PCA(`summe-blattN`, 11)	1	0.1742	491.774	18.018
`summe-uebungs-punkte`, `uebungspunkte-pro-blatt`	2	0.4576	332.674	14.419
`summe-uebungs-punkte`, *PCA*(`summe-blattN`, 1)	1	0.4328	347.860	14.904
`uebungspunkte-pro-blatt`, *PCA*(`summe-blattN`, 1)	1	0.4328	347.860	14.904
`summe-uebungs-punkte`, `uebungspunkte-pro-blatt`, *PCA*(`summe-blattN`, 1)	1	0.4534	334.110	14.479

Das Ergebnis der *GridSearch* ergibt, dass `summe-uebungs-punkte` und mit `uebungspunkte-pro-blatt` zusammen den besten adjustierten $\overline{R^2}$-Wert von 0.4576 erreichen und zwar bei einem Grad von 2. Dabei ist die Vorhersage im Gesamten auf 0.023 Klausurpunkte genau und als absolute Abweichung ergeben sich ± 14.419 Klausurpunkte, also circa drei Notenstufen. Das bedeutet, dass aus Sicht von Organisatoren bei ähnlichen Daten der Klausurdurchschnitt sehr genau sein wird, aber die Note aus Sicht eines Studenten nur grob geschätzt werden kann.
Somit wurde zumindest auf diesem Datensatz eine gute Methode gefunden, mit der die Leistungen der Studenten in der Klausur vorhergesagt werden können.

Johannes Müller [Mü18] hatte damals einen R^2-Wert von 0.814 erreicht. Das lag daran, dass er die Anzahl der *Features* nicht mit einkalkuliert hat, keine Studenten hinzugenommen hat, die sich im zweiten oder dritten Prüfungsversuch befanden, nicht alle Studenten ausgefiltert hat, die nichts abgegeben haben, und vor allem, allen Studenten, die die Klau-

surzulassung nicht erreicht haben, also nicht mal die Klausur schreiben durften, eine 6.0 zugewiesen hat. Dies alles führt zu einer sehr hohen Genauigkeit, wofür aber kein *machine learning* notwendig ist, denn wenn die Punktzahl unter 600 ist, dann ist die Note 6.0. Eine exakte Vorhersage bei fast der Hälfte der Einträge.
Werden aber, zum Vergleich, die bisherigen Ergebnisse dieser Arbeit auf den von Johannes gewählten Datensatz angewendet, ergibt sich ein R^2-Wert von 0.8773. Es wurde also eine Steigerung von 7.78% erreicht! Der durchschnittliche, absolute Fehler liegt dann bei etwa 0.472 Notenpunkten.

3.1.8 Andere Vorhersagemethoden

Auch andere Vorhersagemethoden wurden ausprobiert, aber weil die lineare Regression die besten Ergebnisse erreichte, werden sie nicht weiter erklärt. In Tabelle 4 sind alle Regressionen aufgelistet, die ausprobiert wurden.

Tabelle 4: Vergleich verschiedener Regressoren

Vorhersagemethode	Bester Grad	$\overline{R^2}$	MSE	MAE
Lineare Regression	2	0.4576	332.674	14.419
Bayesian	2	0.4546	334.484	14.472
Ridge	2	0.4523	335.880	14.5
AdaBoost	1	0.4404	343.189	14.785
Support Vector Machine	2	0.4047	365.102	15.486
Least-angle	2	0.3694	386.732	16.078
Random Forest	2	0.3305	410.584	15.919
Decicion Tree	3	0.1063	548.066	18.371
(11)Nearest Neighbors	-	0.4414	342.586	14.693

Wie zu erkennen ist, liefern alle Vorhersagen einen schlechteren adjustierten R^2, *MSE* und *MAE*-Wert, als die lineare Regression.
Bei *Nearest Neighbor* werden die Klausurergebnisse derjenigen Studenten genutzt, dessen Übungspunktverhalten am ähnlichsten zu der getesteten Person sind. Theoretisch hört sich das nach der besten Vorhersagemethode an, deswegen ist es sehr enttäuschend, dass es keine besseren Ergebnisse geliefert hat. Vermutlich liegt es daran, dass viel zu wenig Stichproben vorhanden sind.

3.2 Fortlaufende Vorhersage

Der Zweite Teil der Vorhersage beschäftigt sich mit dem Bewerten der vergangenen Leistungen in den Übungsaufgaben und wie eine Prognose für das nächste Blatt erstellt werden kann. Die Informationen, die aus den beiden Kapiteln gewonnen werden können, werden dann gemeinsam genutzt, um hinterherhängende Studenten dem *Mentoring* zuzuführen.

3.2.1 Bewertung der zuletzten Leistungen

Es sollen nur Studenten miteinander verglichen werden, die sich zumindest an den Aufgaben versucht haben, denn wer keine Eigenleistung zeigt, bei dem werden Hilfeleistungen zu nichts führen. Auch das *Mentroing*-Programm oder Tutoren werden nicht für einen die Arbeit erledigen.
Für dieses Problem ist maschinelles Lernen allein keine gute Wahl, weil es immer mal sein kann, dass ein Blatt zu schwierige Aufgaben beinhaltet und viele Studenten eine geringe Punktzahl erreichen. Deswegen wäre es nicht optimal alle Hörer des Kurses dem *Mentoring*-Programm zuzuführen. Es muss also auch immer, nicht nur ein Vergleich zu schon vergangenen Veranstaltungen, sondern auch zu dem aktuellen Durchschnitt gezogen werden. Eine Kombination aus Durchschnitt und Standardabweichung ist hier besser gewählt, was aber kein Problem darstellt, da all diese Informationen aus dem übergebenen Datensatz entnommen werden können.

Wie in Abschnitt 2.1.2 schon festgestellt wurde, folgen die Punkteverteilungen der Übungspunkte keiner Verteilungsfunktion, weswegen nicht mit üblichen Quantilen gearbeitet werden kann. Das Maß des Durchschnitts abzüglich der doppelten Standardabweichung hat sich aber als hinreichendes herausgestellt, denn damit werden genügend Studenten ausgewählt, die im Vergleich zu anderen zu wenig Punkte erreicht haben. In Tabelle 5 sieht man die verschiedenen arithmetischen Mittel, zugehörige Standardabweichungen und wie viele Studenten zu schlechte Leistungen erbracht, es jedoch wenigstens versucht, haben. Da die Wahl der Standardabweichung, um „genügend“ viele Studenten zu treffen, etwas subjektiv ist, kann in der Anwendung selbst entschieden werden, wie weit die Studenten ausgewählt werden sollen.

Tabelle 5: Übersicht Durchschnitte und Zahl betroffener Studenten

Blatt N	Durchschnitt $\overline{\texttt{summe-blattN}}$	Standard-abweichung σ	Anzahl bearbeitet $\lvert\texttt{summe-blattN} > 0\rvert$	Anzahl kritischer Studenten $\lvert\texttt{summe-blattN} \leq \overline{\texttt{summe-blattN}} - 2\sigma\rvert$
1	86.52	18.35	482	31
2	76.89	25.46	496	35
3	68.14	27.01	444	8
4	71.12	27.44	433	19
5	87.88	33.54	466	0
6	67.31	35.23	404	0
7	76.17	24.11	404	9
8	85.25	32.79	397	0
9	62.37	30.35	278	0
10	59.11	32.69	261	0
11	67.0	31.96	312	0
12	67.17	29.24	210	0

Die betroffenen Studenten werden in Abschnitt 3.2.3 noch weiter selektiert, nachdem eine Vorhersage für kommende Übungsblätter gemacht werden kann.

3.2.2 Vorhersage der Leistungen in den nächsten Übungsaufgaben

Um die Punktzahl auf dem nächsten Übungsblatt zu prophezeien, muss erneut überprüft werden, welche *Features* und welcher Grad genutzt werden. In Abschnitt 2.3.2 wurde bereits festgestellt, dass das vorherige Blatt immer die höchste Korrelation hat und das in Verbindung mit dem vorletzten Blatt sogar noch weiter verbessert werden kann. Auf der anderen Seite gibt es aber das Problem, dass je weiter ein Blatt in der Vergangenheit liegt, desto weniger Aussagekraft hat es. Deswegen muss wieder über eine *GridSearch* herausgefunden werden, wie viele vergangene Übungsblätter für die Vorhersage genutzt werden sollten.
Es war auch möglich die Summe aller bisher erreichten Punkte und die bisherigen Punkte pro bearbeitetes Blatt zu berechnen. Auch eine *PCA* auf den bereits bearbeiteten Blättern wurde durchgeführt, aber mit diesen *Features* wurden für alle Blätter schlechtere Vorhersagen getroffen, weshalb in Tabelle 6 nur noch die letzten n Blätter inklusive Grad stehen.

Tabelle 6: Ergebnisse der *GridSearch* auf Anzahl vergangener Blätter und Grad

Vorhersage Blatt	letzten n Blätter	Bester Grad	$\overline{R^2}$
2	1	6	0.2693
3	2	2	0.4278
4	3	2	0.5854
5	3	2	0.5582
6	2	4	0.6690
7	3	2	0.5459
8	2	3	0.6061
9	3	2	0.4691
10	2	3	0.5589
11	4	1	0.6407
12	2	2	0.6524

Wie auch schon bei der Vorhersage der Klausurpunktzahl am Ende der Vorlesungszeit, ist der am häufigsten auftretende, beste Grad 2.
Durchschnitt der letzten n Blätter ist 2.45. Also werden für die Vorhersage der Punkte in den nächsten Übungsaufgaben die letzten 2 Blätter verwendet, ausgenommen natürlich Blatt 2.

3.2.3 Auswahl für Mentoring

In Abschnitt 3.2.1 wurden bereits die kritischen Studenten bestimmt, die im Verhältnis zu den anderen Hörern des Kurses, schlechte Leistungen erbracht haben. Es gibt aber auch viele Studenten, die die ersten Blätter nicht ernst genommen haben und sich erst nach einigen Blättern intensiv mit ihnen beschäftigt haben. Maschinelles Lernen soll nun dabei helfen, diese Studenten zu identifizieren und nur diejenigen Studenten herauszufiltern, die für das Mentoring-Programm geeignet sind, also Eigeninitiative zeigen, aber zu wenig Punkte erreicht haben, ohne Aussicht auf Verbesserung.
Für Diejenigen, die unter dem Durchschnitt abzüglich doppelter Standardabweichung befinden, wird eine Vorhersage für das nächste Blatt gemacht. Erreichen sie eine Punktzahl unter 50, werden sie dementsprechend markiert, sodass die Organisatoren des Faches dementsprechende Schritte einleiten können.

Tabelle 7: Zahlen über hilfsbedürftige Studenten

Blatt N	Anzahl kritische Studenten $\lvert$`summe-blattN` $\leq \overline{\texttt{summe-blattN}} - 2\sigma\rvert$	Anzahl gerettete Studenten $\lvert$`summe-blattN+1` $\geq 50\rvert$	Für Hilfeleistung geeignet $\lvert\widehat{\texttt{summe-blattN+1}} < 50\rvert$
1	31	9	31
2	35	9	35
3	8	1	7
4	19	7	17
5	0	0	0
6	0	0	0
7	9	2	9
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0

Tabelle 7 kann entnommen werden, dass leider nicht alle Studenten, die sich noch retten konnten, ausgefiltert wurden, weil sie eine unvorhersehbare Abweichung zu den Vorherigen aufzeigen. Das Wichtigste ist aber, dass **alle** ausgefilterten Studenten auf dem Blatt danach eine Punktzahl von über 50 Punkten erreichten.

3.2.4 Vorhersage der Leistungen in der Klausur

In Abschnitt 3.1.7 hat sich ergeben, dass `summe-uebungs-punkte` und `uebungspunkte-pro-blatt` bei einem Grad von 2 die besten Vorhersagen errechnen. Das Problem ist nun, dass diese Attribute erst während des Semesters gebildet werden, es aber sehr wichtig ist, dass auch schon während der Veranstaltung eine Prognose für die Klausur erstellt werden kann, damit Studenten und Organisatoren darauf reagieren können. Das stellt aber kein Problem dar, denn dann wird eben die Summe aller bisherigen Blätter, beziehungsweise die Übungspunkte pro bearbeitetes Blatt, bis zum aktuellsten Blatt berechnet, abhängig davon die wahrscheinlichste Punktzahl in der Klausur errechnet und gegebenenfalls mittels der Formel aus Gleichung (3) in eine Note umgewandelt.

Tabelle 8 zeigt, wie sich die Vorhersage der Klausurpunkte über die Übungsblätter hinweg entwickelt.

Tabelle 8: Kontinuierliche Vorhersage der Klausurpunktzahl

Aktuellstes Blatt	Bester Grad	$\overline{R^2}$	MSE	MAE
1	4	0.0809	563.642	19.361
2	3	0.1819	501.698	17.880
3	3	0.2500	459.977	17.176
4	3	0.3205	416.700	16.206
5	2	0.3253	413.782	16.243
6	2	0.3486	399.504	15.927
7	2	0.3712	385.642	15.580
8	2	0.4014	367.120	15.202
9	2	0.4021	366.687	15.083
10	2	0.4295	349.891	14.753
11	2	0.4491	337.868	14.494
12	2	0.4576	332.674	14.419

Zu bemerken ist, dass sich der Grad schnell wieder bei 2 einpendelt, aber viel interessanter ist, dass wirklich **jedes** Blatt die Vorhersage ein wenig genauer macht.

4 Anwendungssoftware

4.1 Übersicht

Jetzt werden alle Erkenntnisse, die während dieser Arbeit angesammelt wurden, genutzt, um eine Anwendung zu erstellen, damit sie auch außerhalb verwendet werden können.

Die Anwendung soll für jedes Fach benutzt werden können. Sie haben aber unterschiedliche Bewertungssysteme, also müssen erst Algorithmen entwickelt werden, die die Übungsblätter der verschiedenen Fächer vergleichbar macht.

4.1.1 Skalierung

In dem hier analysierten Datensatz gibt es eine maximale Punktzahl von 100 Punkten, die auf jedem Blatt erreicht werden kann. Da alle Ergebnisse aus diesem Datensatz stammen, so können sie auch nur auf sehr ähnlichen Daten vernünftige Ausgaben erreichen. Also müssen alle zukünftigen Eingaben auch auf diese Punktzahl skaliert werden, damit die Vorhersage auch vernünftige Werte vorhersagt.
Zusätzlich kann es auch sein, dass andere Fächer für jedes einzelne Blatt eine andere Maximalpunktzahl vorgesehen haben, als auf den anderen. Ergo muss jedes Blatt einzeln mithilfe der entsprechenden Maximalpunktzahl auf 100 skaliert werden. Die allgemeingültige Formel ist:

$$skaliertePunkte = erreichtePunkte \cdot \frac{100}{maxPunkte} \tag{18}$$

Jetzt können die auch Fächer mit anderen Punkteskalas vernünftig vorhergesagt werden und die Anwendung benutzen.

4.1.2 Entfernen und Hinzufügen von Übungsblättern

Andere Fächer können aber nicht nur unterschiedlich viele Punkte verteilen, sondern auch noch unterschiedlich viele Übungsblättern anbieten. Falls mehr als 12 Übungsblätter existieren, müssen diese für die Vorhersage gegebenenfalls reduziert werden, weil sonst keine Schätzer für Vorhersagen ab dem 13. Blatt existieren. Gleiches gilt für das Trainieren von Schätzern. Johannes Müller [Mü18] entwarf bereits einen Algorithmus, um Übungsblätter zu entfernen oder künstlich zu erstellen, aber er lieferte leider falsche Ausgaben (Beweis im Anhang E). Deswegen werden in Algorithmus 1 und Algorithmus 2 neue Algorithmen

vorgestellt, die diese Aufgaben besser und effizienter umsetzten. Zumindest Algorithmus 1 basiert sogar noch auf der grundlegenden Idee Müllers.

Algorithmus 1 Vektorbasierter Algorithmus um Übungsblätter hinzuzufügen

function TRANSFORMIEREBLAETTER($blaetter$)
  $spalten \leftarrow$ Anzahl an Spalten der Matrix $blaetter$
  $durchschnitte \leftarrow \frac{\sum_{j=0}^{spalten-1} blaetter_{nj}}{spalten}$ ▷ Durchschnitt jeder Zeile
  $indices \leftarrow [1, 2, 3, ..., 12 - spalten]$
  $indices \leftarrow$ RUNDEN$(indices \cdot \frac{11}{13-spalten})$ ▷ Regelmäßige Abstände
  $blaetter_{n\ indices} \leftarrow durchschnitte_n$ ▷ Verschiebendes Einfügen
  return $blaetter$
end function

Algorithmus 1 belegt $\mathcal{O}(n + 1 + 12) = \mathcal{O}(n)$ viel zusätzlichen Speicher und hat eine Laufzeit von $\mathcal{O}(n \cdot spalten) = \mathcal{O}(n)$, weil sich $spalten$ nur zwischen 0 und 11 befinden kann.

Algorithmus 2 Vektorbasierter Algorithmus um Übungsblätter zu entfernen

function TRANSFORMIEREBLAETTER($blaetter$)
  $spalten \leftarrow$ Anzahl an Spalten der Matrix $blaetter$
  $ueberfluessig \leftarrow spalten - 12$
  $blaetter \leftarrow blaetter \cdot \frac{12}{spalten}$ ▷ Skaliere Punkte, damit Maximalpunktzahl passt
  $bearbeitet = (blaetter \neq 0)$ ▷ Matrix aus 0 und 1, ob das Blatt bearbeitet wurde
  $anzahlBearbeitet = \sum_{j=ueberfluessig}^{spalten-1} bearbeitet_{nj}$
  $entfernt \leftarrow \sum_{j=0}^{ueberfluessig-1} blaetter_{nj}$ ▷ Summe zu entfernender Spalten für jede Zeile
  $entfernt \leftarrow \frac{entfernt}{anzahlBearbeitet}$
  lösche die ersten $ueberfluessig$ Spalten aus $blaetter$
  $blaetter[bearbeitet] \leftarrow blaetter_n[bearbeitet] + entfernt_n$ ▷ auf bearbeitete Blätter aufgeteilt
  return $blaetter$
end function

Algorithmus 2 belegt $\mathcal{O}(n \cdot spalten + 2n + 2) = \mathcal{O}(n \cdot spalten)$ viel zusätzlichen Speicher und hat eine Laufzeit von $\mathcal{O}(n \cdot spalten)$.

4.2 Werkzeuge

Weil es sich um sehr sensible Daten handelt, muss viel Wert auf die Sicherheit gelegt werden. Zudem darf die Anwendung nicht systemabhängig sein. Für diese Funktionalität ist *Docker* das richtige Werkzeug. Damit läuft die Anwendung in einem isolierten Container und nur über überwachte *Ports* wird Zugriff gewährt. Gleichzeitig werden alle benötigten Bibliotheken und Frameworks heruntergeladen und innerhalb dieses Docker-Containers installiert.
Die Anwendung ist in *Python 3* geschrieben, weil sie die populärste im Bereich maschinelles Lernen ist und es dafür, sowie für Datenverarbeitung, viele gute Bibliotheken gibt. *scikit-learn* liefert schon Methoden, wie eine lineare Regression und Transformation der Daten auf höhere Grade.
Für ansonsten anstehende Datenverarbeitungen wird vektorbasierte Programmierung mithilfe von *pandas* und *NumPy* benutzt, weil diese Operationen größtenteils in der Programmiersprache *C* geschrieben sind und dadurch einen erheblichen Geschwindigkeitsvorteil liefern.
Weil nach einer Webanwendung gefragt war, wird das *Webframework Flask* genutzt, um HTTP-Anfragen zu handhaben. *Flask* liefert einen kleinen Kern mit den wichtigsten Funktionalitäten, kann aber in jede benötigte Richtung erweitert werden.

4.3 Bedienung

Mittels des Befehls docker-compose up wird der Server gestartet. Alle benötigten Bibliotheken und Frameworks werden heruntergeladen und innerhalb des Docker-Containers installiert. Daraufhin wird der Server gestartet und die Schätzer mit allen verfügbaren Trainingsdatensätzen trainiert. Nun kann nun über den Port 8080 mit der Anwendung kommuniziert werden. Das heißt, es können entweder neue Trainingsdaten hinzugefügt, oder eine Vorhersage für einen übergebenen Datensatz bestimmt werden. Das geht entweder über übliche *Post* und *GET*-Anfragen, oder über das bereitgestellte *HTML-GUI*, auf das nun zugegriffen werden kann. Tabelle 9 zeigt, welche Pfade und Methoden es gibt und wie Daten mit dem Server kommuniziert werden.

Tabelle 9: Beschreibung aller Funktionen der Anwendung

Pfad	Anfrage	POST-BODY	Beschreibung
/	*GET*		Startseite von der eine Vorhersage gebildet, oder die Schätzer auf neuen Daten trainiert werden können.
/vorhersage	*POST*	• `maxBlattPunkte` ($Zahl$ oder Liste dieser) • `blattNamen` ($Text$) • `stdMultiplikator` ($Zahl$) • `daten` (CSV)	Rückgabe ist eine *CSV*-Datei mit Klausurvorhersage, wenn alle übergebenen Studenten zum aktuellen Zeitpunkt die Klausur schreiben würden und Markierung von Studenten, die aktuell Hilfe gebrauchen könnten.
/vorhersage	*GET*		Übersicht über letzte Vorhersage mit Diagrammen.
/vorhersage /download	*GET*		Letzte Vorhersage wird als *CSV*-Datei runtergeladen.
/training	*POST*	• `maxBlattPunkte` ($Zahl$ oder Liste dieser) • `blattNamen` ($Text$) • `maxKlausurPunkte` ($Zahl$) • `klausurPunkteName` ($Text$) • `note` ($Text$) • `daten` (CSV)	Entweder `maxKlausurPunkte` und `klausurPunkteName`, oder `note` werden übergeben. Trainiert die Schätzer mit den neuen Daten.

Wird eine Vorhersage über die *HTML*-Ansicht angefordert, die in Abbildung 15 zu sehen ist wird automatisch die Vorhersage heruntergeladen und es wird auf die Übersichtsseite zu dieser weitergeleitet, in der Diagramme, wie zum Beispiel in Abbildung 16, vorzufinden sind.

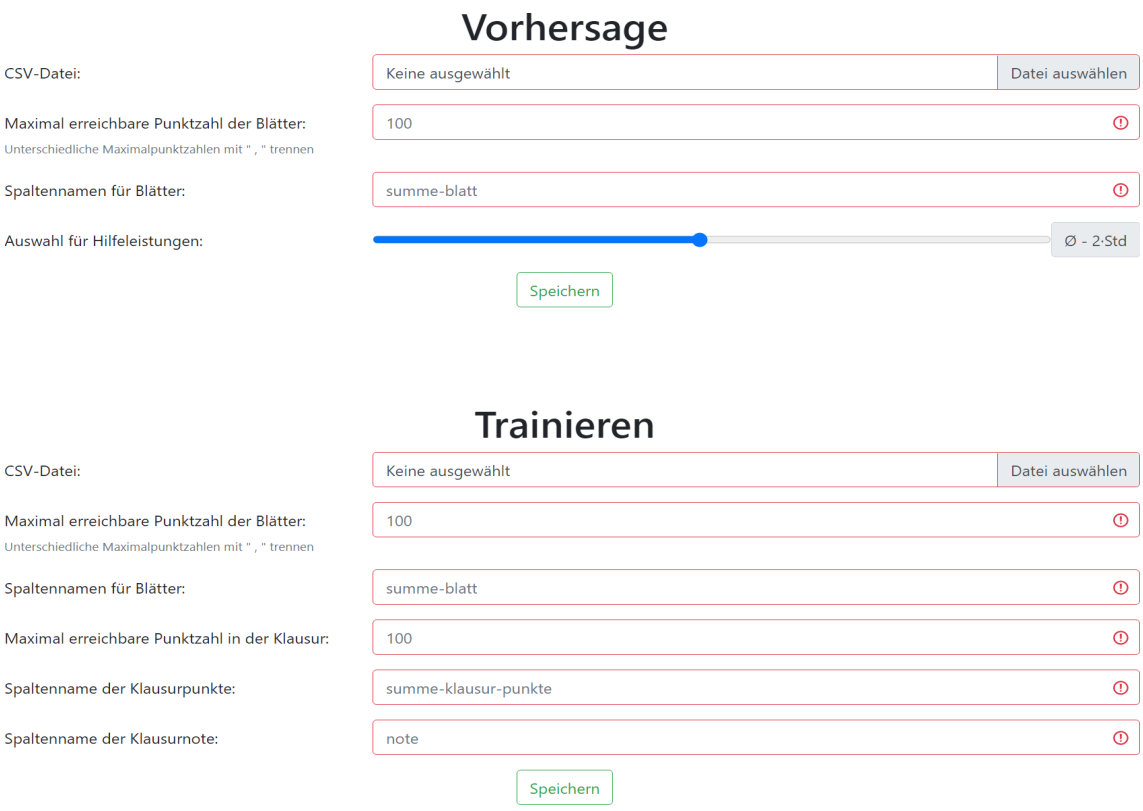

Abbildung 15: HTML-GUI

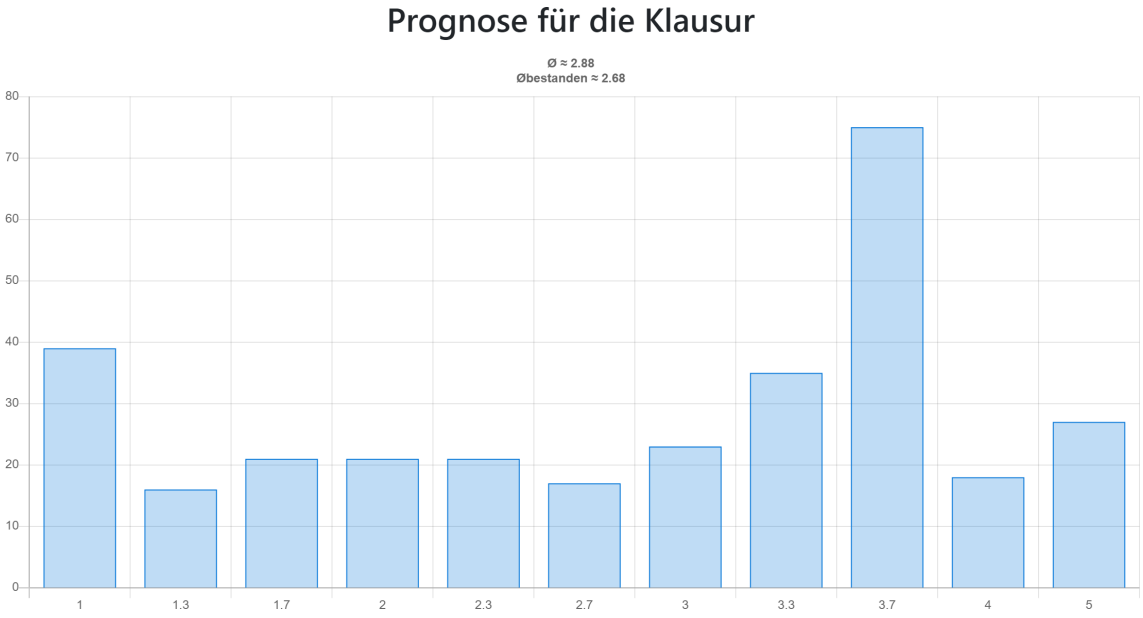

Abbildung 16: HTML-Vorhersage der Klausurnoten

5 Bewertung der Klausurvorhersagen

Zur Bewertung der Güte der Schätzfunktion wurden von dem Betreuer dieser Arbeit, Dr. Jens Bendisposto, drei weitere Datensätze übergeben.

Der erste Datensatz enthält Übungspunkte von 11 Übungsblättern und Klausurnoten von einem Fach, bei dem es sich nicht um „Informatik I“, sondern um „Rechnerarchitektur“ handelt. Die Daten können sich also stark von dem ursprünglich untersuchten unterscheiden. Leider sind keine Klausurpunkte, sondern nur die Noten vorhanden, weshalb nicht selektiert werden kann, wer wirklich zur Klausur erschienen ist, beziehungsweise wer einen ernsthaften Versuch unternommen hat. Wenn die Ergebnisse in Punkten vorhanden wären, könnten alle Studenten, die 0 Punkte haben, vom Vergleich ausgelassen werden.

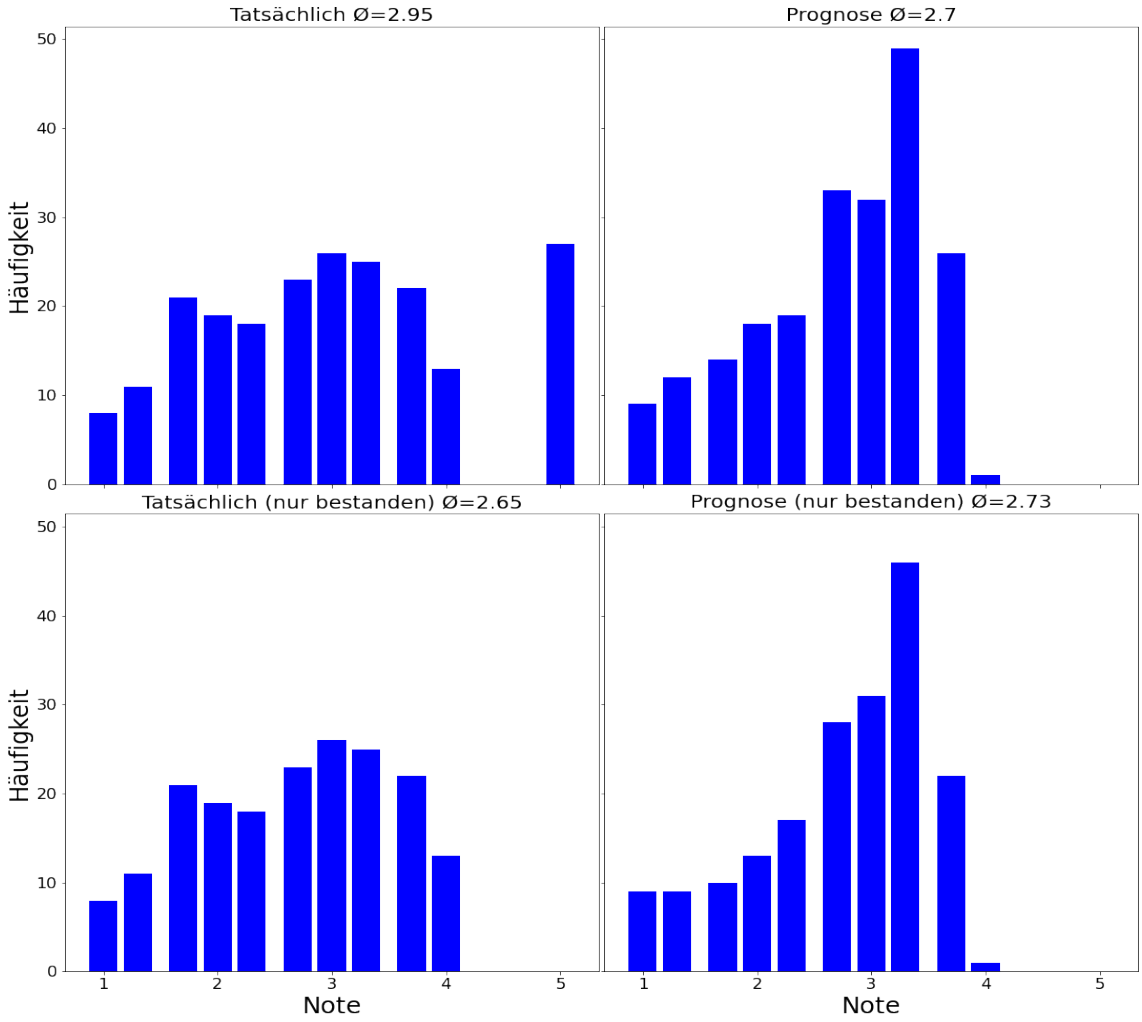

Abbildung 17: Notenvergleich des ersten Datensatzes mit der Prognose

Wenn alle Studenten miteinbezogen werden, die eine 5.0 erhalten haben, also auch Studenten, die nicht erschienen sind, ergibt sich eine Notenschnittabweichung von 0.25 und eine durchschnittliche, absolute Abweichung von 1.26 Notenpunkten, zwischen der Vorhersage und den tatsächlichen Klausurergebnissen.
Werden nur alle Einträge betrachtet, die bestanden haben, ergibt sich eine Notenschnittabweichung von 0.08 und eine durchschnittliche, absolute Abweichung von 1.07 Notenpunkten.
Letztere ist sogar noch relativ gut, wenn man bedenkt, dass es immer eine gewisse Abweichung gibt, für die unter anderem der Zufall verantwortlich ist. Das Wichtigste ist, dass sie sich im Gesamten betrachtet gegenseitig ausgleichen.

Das größte Problem der Vorhersage, in Abbildung 17 rechts zu sehen, ist die Menge an Studenten im 3er Notenbereich. Insbesondere bei 3.7. In der Realität verteilen sich diese Studenten auf die umliegenden Noten, doch in der Vorhersage türmen sie sich dort, was daran liegt, dass in dem Datensatz sehr viele Studenten eine mittelmäßige Punktzahl erreicht haben.
Nichtsdestotrotz liefert der Schätzer eine erstaunlich gute Prognose für die Klausur, wenn bedacht wird, dass immer noch auf nur einem Datensatz trainiert wurde, der auch noch einem anderen Fach entspricht und es sich hier um eine komplett unbekannte Eingabe handelt.

Beim zweiten Datensatz gibt es wieder 11 Übungsblätter, diesmal aber die Klausurpunkte statt der Note. Das heißt, Studenten, die nicht an der Klausur teilgenommen haben, also 0 Punkte bekommen haben, werden ausgefiltert. Zusätzlich stammen die Daten wieder aus einer „Informatik I“ Vorlesung. Allerdings sieht es den Daten nach so aus, als ob die Zulassung dieses Mal bei circa 40% lag.

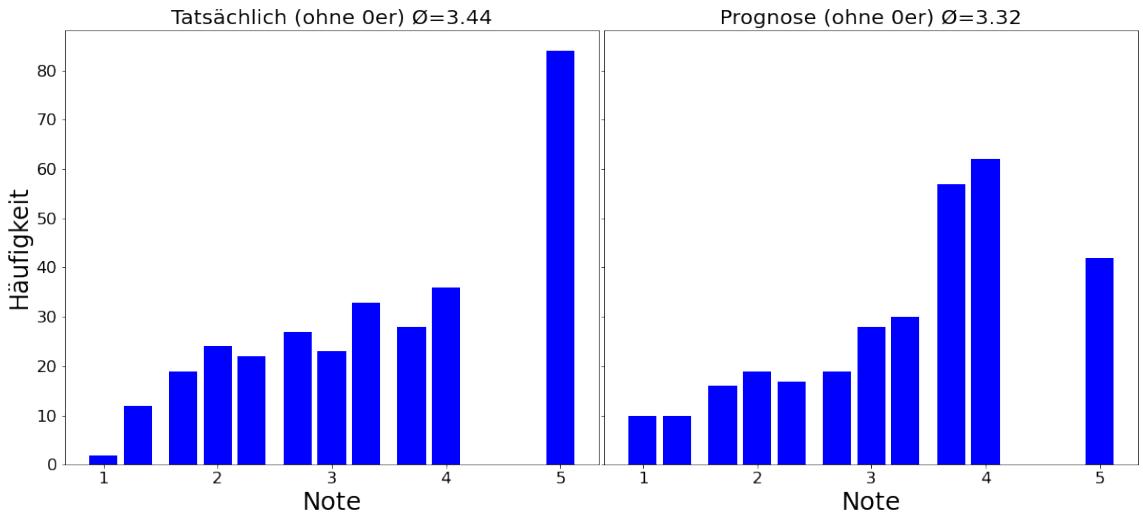

Abbildung 18: Notenvergleich des zweiten Datensatzes mit der Prognose

Es ergibt sich eine Notenschnittabweichung von 0.12 und eine durchschnittliche, absolute Abweichung von 0.8 Notenpunkten zwischen Vorhersage und tatsächlichen Noten.
Aber auch hier zeigt sich wieder das Problem, dass sich in der Prognose zu viele Werte bei 3.7 bis 4.0 befinden und sich nicht auf die umliegenden Noten verteilen.
Für den Rest der Vorhersage sieht es auch wieder sehr gut aus: Die Werte stimmen gut überein und die abflachende Tendenz zur 1.0 ist in Abbildung 18 klar wiederzufinden.

Im letzten Datensatz konnten erneut 11 Übungsblätter bearbeitet werden und dieses Mal stehen auch wieder die Klausurpunkte zu Verfügung. Also werden Studenten, die nicht an der Klausur teilgenommen haben, erneut ausgefiltert und nicht mit verglichen. Dieser Datensatz stammt aus der Vorlesung „Theoretische Informatik“ und dessen Daten lassen darauf schließen, dass die Zulassung nicht bei der Hälfte der Punkte lag, sondern bei festen 40 von 92 Punkten (circa 43.48%). Es gab auf den einzelnen Blättern nur maximal 8 Punkte zu erreichen, außer auf dem 10. Blatt, denn dort konnten 4 Punkte mehr erreicht werden. Die Übungspunkte wurden also nicht so genau vergeben, wie in den anderen Datensätzen.

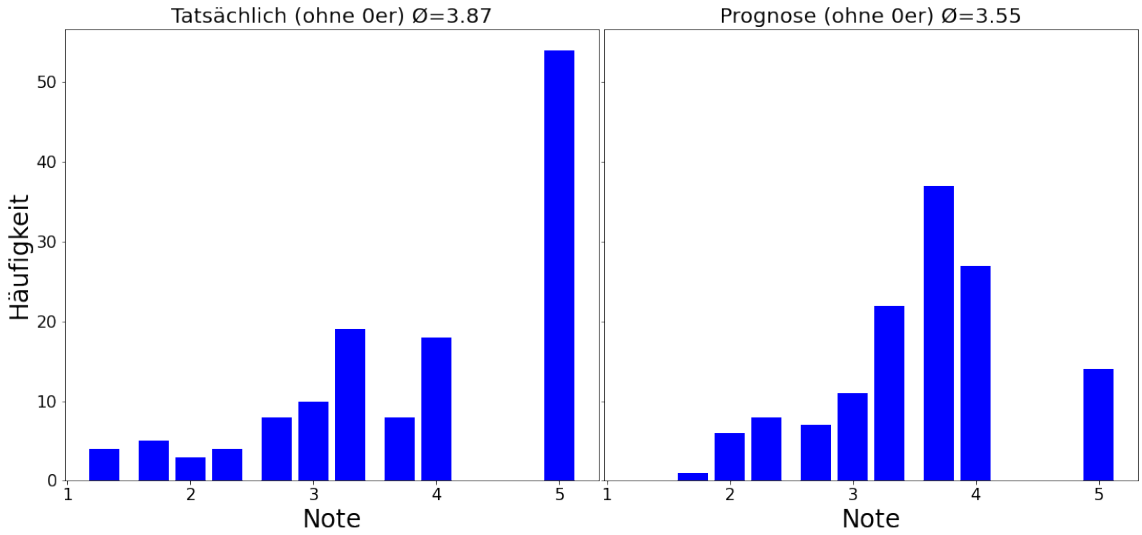

Abbildung 19: Notenvergleich des dritten Datensatzes mit der Prognose

Nur Abbildung 19 nach, handelt es sich wohl um die schlechteste Vorhersage, mit einer Notenschnittabweichung von 0.32. Der 3er Bereich sieht zwar noch sehr gut geschätzt aus, der 4er und 5er Bereich dafür umso ungenauer. Die Studenten, die tatsächlich eine 5.0 erhalten haben, werden in der Vorhersage zu hoch eingeschätzt.
Auch wenn es nicht danach aussieht, aber es ergibt sich trotzdem eine durchschnittliche, absolute Abweichung von 0.78 (etwas über zwei Notenschritten), womit es sich, zumindest aus Studentenperspektive, um die genauste Vorhersage, der drei betrachteten, handelt.

5.1 Zusammenfassung der Bewertung

Insgesamt liefert der Klausurschätzer, der nur auf dem ursprünglichen Datensatz aus „Informatik I“ trainiert wurde, auf allen drei neuen Datensätzen ganz gute Vorhersagen, mit einer schlimmsten Schnittabweichung von 0.32 im letzten Datensatz und einer schlimmsten durchschnittlichen, absoluten Abweichung von 1.07 im zuerst getesteten. Das, obwohl sie sich stark unterscheiden und jeder Datensatz seine eigenen Merkmale hat, wie zum Beispiel einem anderen Fach, einer anderen Zulassungsgrenze, einer geringen Maximalpunktzahl in den Übungen oder einer Verteilung der Studenten auf nur mittelmäßige Übungspunktzahlen.

Es besteht aber noch das Problem, dass zu viele Studenten auf einen spezifischen Bereich vorhergesagt werden, weil sie nicht weiter differenziert wurden. Dies führt dazu, dass es zu einer Anhäufung in diesem Bereich kommt, obwohl sie sich in der Realität auf die umliegenden Noten verteilen.

6 Fazit

6.1 Zusammenfassung

Um die Klausurnote, auch während des Semesters, vorherzusagen werden zwei wichtige Eigenschaften des Abgabeverhaltens eines Studenten verwendet:

1. Die Informationen, wie gut ein Student alle bisherigen Themen verstanden hat (**Summe aller Übungspunkte**).
2. Die Informationen, wie gut ein Student alle von ihm angeschauten Themen verstanden hat, um die Studenten zu unterteilen, die aufgehört haben, sobald sie die Zulassung erreicht haben (**Übungspunkte pro bearbeitetes Blatt**).

Diese Daten werden in die Form $[1, Summe, PpbB, Summe^2, Summe \cdot PpbB, PpbB^2]$ gebracht und daraufhin eine lineare Regression genutzt um, abhängig von ähnlichen Daten aus vorherigen Veranstaltungen, die wahrscheinlichste Klausurpunktzahl vorherzusagen.

Damit Studenten gefunden werden, die aktuell nicht mit den Aufgaben zurechtkommen, werden zunächst alle Studenten ausgewählt, die unter dem Durchschnitt des aktuellen Blatts abzüglich der doppelten, in der Serveranwendung aber wählbaren, Standardabweichung liegen. Für diese ausgewählten Studenten wird eine Vorhersage für das nächste Übungsblatt gemacht, die aus den Punkten der zwei zuletzt veröffentlichten Blättern besteht, welche auch wieder in die Form $[1, vorletztes, letztes, vorletztes^2, vorletztes \cdot letztes, letztes^2]$ gebracht werden. Anschließend wird wieder eine lineare Regression genutzt, in der die Daten mit derer früherer Studenten verglichen werden, um die wahrscheinlichste Punktzahl auf dem nächsten Blatt vorherzusagen. Alle Studenten, die unter die Hälfte der Punkte in der Prognose des nächsten Übungsblatts fallen, das letzte Blatt aber bearbeitet haben, also einen Willen zeigen, sind für Hilfeleistungen geeignet und werden dementsprechend markiert.

Mit der entwickelten Anwendung können beide Prognosen automatisch generiert und visualisiert werden. Zudem können fertige Datensätze über Übungspunkte mit zugehörigen Klausurergebnissen gespeichert werden, um zukünftige Vorhersagen immer genauer zu machen.

6.2 Eigene Auffassung

Während ich die Bachelorarbeit von Johannes Müller [Mü18] gelesen habe, sind mir einige Fehler, Verbesserungen und Erweiterungen eingefallen. Auf die Arbeit rückblickend, ist es

sehr erfreulich, dass alle zu einem, wenn auch nur gering, besseren Ergebnis geführt haben.

Der beste Teil dieser Arbeit ist für mich die Serveranwendung, weil in dieser alle Ergebnisse dieser Arbeit verewigt werden durften. Besonders, weil in der Anwendung Müllers keinerlei Schätzfunktionen implementiert waren und es nun endlich möglich ist, die entwickelten Vorhersagen nutzen zu können. Das Programmieren der Anwendung, aber auch das Entwickeln aller Algorithmen, die zur Forschung verwendet wurden, hat mir viel Freude bereitet, weil vektorbasierte und mathematische Programmierung in *Python* durch *numpy*, und *pandas* sehr angenehm wird.
Da ich bisher keine Erfahrung mit *Flask*, *Jinja*, *Werkzeug*, *Javascript*, *Chart.js* und Tests in *Python* gesammelt habe, hat es auch einige Zeit gedauert, bis ich mich überall eingelesen habe. Ich muss trotzdem sagen, dass ich mit meiner Programmierung der Anwendung sehr zufrieden bin, weil alle gewünschten Funktionen implementiert sind, die verschiedenen Bereiche der Anwendung sehr gut aufgeteilt wurden, hoffentlich alle ungültigen Eingaben abgefangen werden und sie auch noch akzeptabel gut aussieht, obwohl ich kein Webentwickler/-designer bin.

6.3 Ausblick

Aus oben genannten Gründen bietet die Serveranwendung noch Platz für Erweiterungen und besonders Verbesserungen des Codes.

Die Vorhersage kann noch verbessert werden, indem eine Lösung dafür gefunden wird, wie eine Anhäufung an Studenten auf einer Note verhindert und **realistisch** verteilt werden kann. In Abbildung 18 links zu sehen ist nämlich, dass eine Anhäufung auf einer Note, in diesem Fall 5.0, auch eintreten kann.
Ansonsten kann sie noch weiter verbessert werden, weil mehr Daten vorhanden sind. Meine Arbeit, genauso wie Müllers Arbeit, basierten auf dem gleichen kleinen Datensatz. Da die neuen Daten erst gegen Ende verfügbar wurden, blieb nicht viel Zeit zur Verbesserung der Vorhersage. Nun sind aber auch mal andere Daten vorhanden und somit können zukünftig die Vorhersagen noch verändert oder verbessert werden. Vor allem, weil jetzt eine Anwendung existiert, die von den Organisatoren der jeweiligen Veranstaltungen verwendet werden kann, so existiert auch eine zentrale Stelle, in der alle endgültigen Klausur- und Übungspunktergebnisse anonym gespeichert werden können.
Möglicherweise liefert, durch die neuen Daten, eine andere Vorhersage bessere Ergebnisse. Ein starker Kandidat dafür ist zum Beispiel die *kNearestNeighbor*-Regression.

Mehr Informationen bedeuten auch mehr Ideen.

Anhang

Anhang A Hypothesentest auf Gleichverteilung der Blätter

Um zu überprüfen, ob die Punkte, die auf den Blättern erreicht wurden, gleichverteilt sind, wird ein $\mathcal{X}^2$-Anpassungstest mit einem Signifikanzniveau von 10% angewendet.

H_0 : Die Punkteverteilung der Blätter ist **gleichverteilt**.
H_1 : Die Punkteverteilung der Blätter ist **nicht gleichverteilt**.

Der $\mathcal{X}^2$-Koeffizient berechnet sich durch

$$\begin{aligned} n &= \text{Anzahl Studenten, die das Blatt bearbeitet haben} \\ n_j &= \text{Anzahl Studenten, die j viele Punkte erreicht haben} \\ \mathcal{X}^2 &= \sum_{j=1}^{maxPunkte} \frac{\left(n_j - \frac{n}{maxPunkte}\right)^2}{\frac{n}{maxPunkte}} \end{aligned} \tag{19}$$

Tabelle 10: $\mathcal{X}^2$-Werte der einzelnen Blätter

Blatt	Maximalpunktzahl	$\mathcal{X}^2$
1	100	6335
2	100	3787
3	100	2342
4	100	2855
5	125	3648
6	110	673
7	100	6228
8	125	3050
9	100	890
10	110	562
11	125	1580
12	100	885

Werden alle Daten für die einzelnen Blätter eingesetzt, ergeben sich die Werte aus Tabelle 10. Wie auch schon in Abbildung 5 erkennbar, liefern Blatt $6, 9, 10$ und 12 eine Punkteverteilung, die am ehesten einer Gleichverteilung ähneln, weil sie die geringsten $\mathcal{X}^2$-Werte erreichen.
Der kritische Wert für eine $\mathcal{X}^2$-Verteilung mit einem Signifikanzniveau von 10% liegt für 100, 110 und 125 Maximalpunkte bei 117.41, 128.3 und 144.56. Wie zu erkennen ist, überschreiten alle Blätter die entsprechenden kritischen Punkte, die meisten sogar weit. Also muss die *Nullhypothese* abgelehnt werden und daraus folgt wiederum, dass die Punkteverteilung der Blätter nicht gleichverteilt sind.

Anhang B Gleichungen der linearen Regression

B.1 Umformen der Minimierungsfunktion

$$\{m, b\} = \arg\min_{m,b} \sum_{i=0}^{n-1} ((m \cdot x_i + b) - y_i)^2$$

um $\sum_{i=0}^{n-1} ((m \cdot x_i + b) - y_i)^2$ zu minimieren, wird zunächst nach b abgeleitet.

$$\sum_{i=0}^{n-1} ((m \cdot x_i + b) - y_i)^2 \frac{d}{db} = \sum_{i=0}^{n-1} 2 \cdot (m \cdot x_i + b - y_i)$$

und gleich 0 gesetzt

$$\begin{aligned} 0 &= \sum_{i=0}^{n-1} 2 \cdot (m \cdot x_i + b - y_i) \\ &= \sum_{i=0}^{n-1} (m \cdot x_i + b - y_i) \\ &= m \cdot \sum_{i=0}^{n-1} x_i + n \cdot b - \sum_{i=0}^{n-1} y_i \\ &= n \cdot m \cdot \bar{x} + n \cdot b - n \cdot \bar{y} \\ &= n \cdot (m \cdot \bar{x} + b - \bar{y}) \\ b &= \bar{y} - m \cdot \bar{x} \end{aligned} \tag{20}$$

mit $\bar{x} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} x_i$ und $\bar{y} = \frac{1}{n} \cdot \sum_{i=0}^{n-1} y_i$
Wir haben also schon $b = \bar{y} - m \cdot \bar{x}$. Jetzt wird b eingesetzt

$$\begin{aligned}
\sum_{i=0}^{n-1}((m \cdot x_i + b) - y_i)^2 &= \sum_{i=0}^{n-1}(m \cdot x_i + \bar{y} - m \cdot \bar{x} - y_i)^2 \\
&= \sum_{i=0}^{n-1}(m \cdot (x_i - \bar{x}) - (y_i - \bar{y}))^2 \\
&= \sum_{i=0}^{n-1} m^2 \cdot (x_i - \bar{x})^2 - 2m \cdot (x_i - \bar{x})(y_i - \bar{y}) + (y_i - \bar{y})^2 \\
&= m^2 \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})^2 - 2m \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y}) + \sum_{i=0}^{n-1}(y_i - \bar{y})^2
\end{aligned} \tag{21}$$

nach m abgeleitet

$$\begin{aligned}
m^2 \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})^2 - 2m \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y}) + \sum_{i=0}^{n-1}(y_i - \bar{y})^2 \frac{d}{dm} \\
= 2m \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})^2 - 2 \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})
\end{aligned}$$

und gleich 0 gesetzt

$$\begin{aligned}
0 = 2m \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})^2 - 2 \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y}) \\
2m \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})^2 = 2 \cdot \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y}) \\
m = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n-1}(x_i - \bar{x})^2}
\end{aligned} \tag{22}$$

Also insgesamt $m = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n-1}(x_i - \bar{x})^2}$, was sich mit den Ergebnissen von [Bon17] deckt.
Da die zu minimierende Funktion eine Summe von Quadraten ist, so ist sie eine Parabel mit ausschließlich nicht-negativen Werten. Somit hat sie nur eine Extremstelle, welche ein globales Minimum ist.

B.2 Umformen der Minimierungsfunktion als Matrix

$$M = \arg\min_{m} \sum_{i=0}^{n-1} (x_i^T \cdot m - y_i)^2$$

um $\sum_{i=0}^{n-1} (x_i^T \cdot m - y_i)^2$ zu minimieren, wird die Gleichung zunächst in Matrixschreibweise umgewandelt.

$$\begin{aligned}
\sum_{i=0}^{n-1} (x_i^T \cdot m - y_i)^2 &= (X \cdot m - Y)^2 \\
&= (Xm - Y)^T \cdot (Xm - Y) \\
&= (m^T X^T - Y^T) \cdot (Xm - Y) \\
&= m^T X^T Xm - m^T X^T Y - Y^T Xm + Y^T Y \\
&= m^T X^T Xm - 2m^T X^T Y + Y^T Y, \text{ weil } (Y^T Xm)^T = m^T X^T Y
\end{aligned} \tag{23}$$

nach m abgeleitet

$$m^T X^T Xm - 2m^T X^T Y + Y^T Y \frac{d}{dm} = 2X^T Xm - 2X^T Y$$

und gleich 0 gesetzt

$$\begin{aligned}
0 &= 2X^T Xm - 2X^T Y \\
&= 2 \cdot (X^T Xm - X^T Y) \\
X^T Xm &= X^T Y \\
m &= (X^T X)^{-1} X^T Y
\end{aligned} \tag{24}$$

Also insgesamt $M = (X^T X)^{-1} X^T Y$, was sich mit den Ergebnissen von [Bon17] deckt.

Anhang C Quadrat von Pearsons Korrelationskoeffizienten

$$r_{pearson}(x, y) = \frac{\sum_{i=0}^{n-1} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1} (x_i - \bar{x})^2 \cdot \sum_{i=0}^{n-1} (y_i - \bar{y})^2}}$$

$$
\begin{aligned}
r^2(x,y) &= \frac{\left(\sum_{i=0}^{n-1}(x_i-\bar{x})(y_i-\bar{y})\right)^2}{\sum_{i=0}^{n-1}(x_i-\bar{x})^2 \cdot \sum_{i=0}^{n-1}(y_i-\bar{y})^2} \\
&= \frac{\left(\sum_{i=0}^{n-1}(x_i-\bar{x})(y_i-\bar{y})\right)^2}{\sum_{i=0}^{n-1}(x_i-\bar{x})^2 \cdot \sum_{i=0}^{n-1}(y_i-\bar{y})^2} \cdot \frac{\sum_{i=0}^{n-1}(x_i-\bar{x})^2}{\sum_{i=0}^{n-1}(x_i-\bar{x})^2} \\
&= \left(\frac{\sum_{i=0}^{n-1}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=0}^{n-1}(x_i-\bar{x})^2}\right)^2 \cdot \frac{\sum_{i=0}^{n-1}(x_i-\bar{x})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2} \\
&= m^2 \cdot \frac{\sum_{i=0}^{n-1}(x_i-\bar{x})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}, \qquad m \text{ aus Gleichung (22)} \\
&= \frac{\sum_{i=0}^{n-1}(\hat{y}_i-\bar{\hat{y}})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}
\end{aligned}
$$

Da Geschätzte und Ursprüngliche den gleichen Mittelwert haben, gilt $\bar{\hat{y}} = \bar{y}$

$$
\begin{aligned}
&= \frac{\sum_{i=0}^{n-1}(\hat{y}_i-\bar{y})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2} \\
&= \frac{\sum_{i=0}^{n-1}(y_i-\bar{y})^2 - \sum_{i=0}^{n-1}(y_i-\hat{y})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2} \\
&= 1 - \frac{\sum_{i=0}^{n-1}(y_i-\hat{y})^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}
\end{aligned}
\tag{25}
$$

Anhang D Umformen der PCA-Gleichung

Umformung grob nach [Sha13]
Zunächst wird von jeder Spalte X_j der jeweilige Spaltendurchschnitt abgezogen:

$$X_j = X_j - \frac{1}{n} \cdot \sum_{i=0}^{n} x_i$$

Das führt dazu, dass der neue Mittelwert der Spalten 0 ist.
Sei V eine $[p \times 1]$ Matrix, mit der die Projektionen $Y = XV$ dargestellt werden können und für die $V^T V = 1$ gilt.
Definiere nun Varianz für XV.

$$\begin{aligned} Var(XV) = \sigma^2 &= \frac{1}{n} \cdot (XV - \bar{X}V)^2 \\ &= \frac{1}{n} \cdot (XV)^2, \text{ weil } \bar{X} = 0 \\ &= \frac{1}{n} \cdot (XV)^T \cdot (XV) \\ &= \frac{1}{n} \cdot (V^T X^T XV) \end{aligned} \tag{26}$$

Die Varianz soll **maximiert** werden. Da X fest ist, muss V gesucht werden:

$$\arg\max_V \sigma^2 = \arg\max_V V^T X^T XV$$

Nach Definition ist

$$\begin{aligned} V^T V &= 1 \\ (V^T V - 1) &= 0 \\ \Lambda(V^T V - 1) &= 0 \end{aligned} \tag{27}$$

Für irgendeine Matrix Λ der Größe $[p \times p]$.

$$\begin{aligned} \arg\max_V V^T X^T XV &= \arg\max_V V^T X^T XV - \Lambda(V^T V - 1) \\ &= \arg\max_V V^T X^T XV - \Lambda V^T V + \Lambda \end{aligned} \tag{28}$$

nach V abgeleitet

$$V^T X^T XV - \Lambda V^T V + \Lambda \frac{d}{dV} = 2X^T XV - 2\Lambda V$$

und gleich 0 gesetzt

$$\begin{aligned} 0 &= 2X^T XV - 2\Lambda V \\ 0 &= 2 \cdot (X^T XV - \Lambda V) \\ X^T XV &= \Lambda V \end{aligned} \tag{29}$$

Das sieht nach dem Eigenwertproblem aus. Gesucht ist also eine Eigenwertmatrix Λ zu $X^T X$ und den zugehörigen Eigenvektoren in V. Um die beste Varianz zu erhalten, sollten also auch die größtmöglichsten Eigenwerte zuerst genommen werden.

Anhang E Johannes Müllers Algorithmus zur Anpassung der Blattsummen

Algorithmen aus [Mü18]:

Algorithmus 3 Anpassung der Blattsummen

```
1: num_features ← # der vorhandenen Blattsummen
2: num_desired ← # der gewünschten Blattsummen
3: old_features ← Array der gegebenen Features
4: new_features ← leeres Array der Länge num_desired
5:
6: if num_features = 0 then
7:     return [0, ..., 0] der Länge num_desired
8: else if num_features = 1 then
9:     return [old_features[0], ..., old_features[0]], der Länge num_desired
10: else
11:     return new_features ← [0, ..., 0]
12: end if
13:
14: fraction ← (num_desired − 1)/(num_features − 1)
15: if fraction ≥ 1 then                                  ▷ Kreieren von Blättern
16:     old_mean ← mean(old_features)
17:     taken ← []
18:     for i ∈ [0, ..., num_features] do
19:         new_index ← round(fraction · i)
20:         taken += new_index
21:         new_features[new_index] ← old_features[i]
22:     end for
23:     for i ∈ taken (complement) do
24:         new_features[i] ← old_mean
25:     end for
26:     flatten(new_features, taken (complement))          ▷ Siehe Algorithmus 4
27: else                                                   ▷ Fasse Features zusammen
28:     for i ∈ [0, ..., num_features − 1] do
29:         new_index ← round(fraction · i)
30:         adjust ← (num_desired/num_features)
31:         new_features[new_index] += (old_features[i] ∗ adjust)
32:     end for
33: end if
34: return new_features
```

Algorithmus 4 Einfaches egalisieren der Blattverteilung

```
1: function flatten(features, used_indices)
2:     remainder ← 0
3:     new_value ← 0
4:
5:     for i ∈ used_indices do
6:         min ← min(features[i − 1], features[i + 1])
7:         max ← max(features[i − 1], features[i + 1])
8:         features[i] ← features[i] + remainder
9:         new_value ← features[i], beschnitten auf [min, max]
10:        remainder ← features[i] − new_value
11:        features[i] ← new_value
12:    end for
13:    return features
14: end function
```

Betrachte nun folgende Instanz:
$num\_features = 4$
$num\_desired = 5$
$old\_features = [100, 2, 2, 100]$
$new\_features = [0, 0, 0, 0, 0]$

Nun wird der Algorithmus befolgt.
$fraction = \frac{5-1}{4-1} = \frac{4}{3}$
$old\_mean = 51$
$taken = []$

$for\ i \in [0, 1, 2, 3]$
$\quad \underline{i = 0}:$
$\quad new\_index = 0$
$\quad taken = [0]$
$\quad new\_features = [100, 0, 0, 0, 0]$
$\quad \underline{i = 1}:$
$\quad new\_index = 1$
$\quad taken = [0, 1]$
$\quad new\_features = [100, 2, 0, 0, 0]$
$\quad \underline{i = 2}:$
$\quad new\_index = 3$
$\quad taken = [0, 1, 3]$
$\quad new\_features = [100, 2, 0, 2, 0]$
$\quad \underline{i = 3}:$

$new\_index = 4$
$taken = [0, 1, 3, 4]$
$new\_features = [100, 2, 0, 2, 100]$
$\underline{i = 4}$:
$new\_index = 5$
$taken = [0, 1, 3, 4, 5]$
ERROR, da $old\_features[4]$ nicht existiert.
Hier wurde aber wahrscheinlich nur ein -1 vergessen,
denn in der Schleife des Zusammenfassens von *Features* wurde richtig iteriert
und im Code ist dieser Fehler auch nicht aufzufinden.

$for\ i \in [2]$
$new\_features = [100, 2, 51, 2, 100]$

Jetzt wird die Methode $flatten([100, 2, 51, 2, 100], [2])$ aufgerufen.
$remainder = 0$
$new\_value = 0$

$for\ i \in [2]$
$min = min(2, 2) = 2$
$max = max(2, 2) = 2$
$new\_features = [100, 2, 51, 2, 100]$
$new\_value = 2$
$remainder = 51 - 2 = 49$
$new\_features = [100, 2, 2, 2, 100]$

Insgesamt ist die Rückgabe also $new\_features = [100, 2, 2, 2, 100]$.
$\sum new\_features = 206$

$\overline{new\_features} = \frac{206}{5} = 41.2$
Der Durchschnitt der Eingabe $[100, 2, 2, 100]$ ist aber $\frac{204}{4} = 51$.
Vorher wurden also 51% der Punkte erreicht und nach der Transformation wurden nur noch 41.2% der Punkte erreicht. Dadurch wird er Student schlechter, als er eigentlich ist und bekommt schlechtere Vorhersageergebnisse.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Quellcodeverzeichnis

Literatur

[Ask16] ASKINADZE, Alexander: Anwendung der Regressions-SVM zur Vorhersage studentischer Leistungen. In: *GvD*, 2016, S. 15–20

[Bon17] BONTEMPI, Gianluca: Statistical foundations of machine learning. (2017)

[Dam00] DAMBOLENA, Ismael G.: A regression exercise: Predicting final course grades from midterm results. In: *Problems, Resources, and Issues in Mathematics Undergraduate Studies* 10 (2000), Nr. 4, S. 351–358

[Gér19] GÉRON, Aurélien: *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019

[Gri18] GRINBERG, Miguel: *Flask web development: developing web applications with python*. Ö'Reilly Media, Inc.", 2018

[HLS14] HU, Ya-Han ; LO, Chia-Lun ; SHIH, Sheng-Pao: Developing early warning systems to predict students' online learning performance. In: *Computers in Human Behavior* 36 (2014), S. 469–478

[Hun07] HUNTER, J. D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95. http://dx.doi.org/10.1109/MCSE.2007.55. – DOI 10.1109/MCSE.2007.55

[Mü18] MÜLLER, Johannes: *Notenvorhersage mit maschinellem Lernen*, Heinrich Heine Universität Düsseldorf, Bachelorarbeit, 2018

[McK10] MCKINNEY Wes: Data Structures for Statistical Computing in Python. In: WALT Stéfan van d. (Hrsg.) ; MILLMAN Jarrod (Hrsg.): *Proceedings of the 9th Python in Science Conference*, 2010, S. 56 – 61

[MM17] MWALUMBWE, Imani ; MTEBE, Joel S.: Using learning analytics to predict students' performance in Moodle learning management system: A case of Mbeya University of Science and Technology. In: *The Electronic Journal of Information Systems in Developing Countries* 79 (2017), Nr. 1, S. 1–13

[Oli06] OLIPHANT, Travis E.: *A guide to NumPy*. Bd. 1. Trelgol Publishing USA, 2006

[PVG+11] PEDREGOSA, F. ; VAROQUAUX, G. ; GRAMFORT, A. ; MICHEL, V. ; THIRION, B. ; GRISEL, O. ; BLONDEL, M. ; PRETTENHOFER, P. ; WEISS, R. ; DUBOURG, V. ; VANDERPLAS, J. ; PASSOS, A. ; COURNAPEAU, D. ; BRUCHER, M. ; PERROT, M. ; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[RPR13] RAMESH, VAMANAN ; PARKAVI, P ; RAMAR, K: Predicting student performance: a statistical and data mining approach. In: *International journal of computer applications* 63 (2013), Nr. 8

[Sch19] SCHÄDLE: Kapitel 3 LGS. In: *Numerik I Skript*. Schädle, 2019, S. 27–31

[Sha13] SHALIZI, Cosma: *Advanced data analysis from an elementary point of view*. 2013

[Spe61] SPEARMAN, Charles: The proof and measurement of association between two things. (1961)

[tea20] TEAM, The pandas d.: *pandas-dev/pandas: Pandas 1.1.1*. http://dx.doi.org/10.5281/zenodo.3993412. Version: aug 2020

[VDWCV11] VAN DER WALT, Stefan ; COLBERT, S C. ; VAROQUAUX, Gael: The NumPy array: a structure for efficient numerical computation. In: *Computing in Science & Engineering* 13 (2011), Nr. 2, S. 22