

BACHELORARBEIT

zur Erlangung des akademischen Grades

„Bachelor of Science in Engineering“ im Studiengang
Informatik

Vergleich von Machine Learning Methoden zur Bestimmung von Grundstückspreisen in Wien

Ausgeführt von: Lorenz Herzberger
Personenkennzeichen: 1410257022

1. BegutachterIn: Patrick Sturm, MSc

Wien, 17.12.2018

Eidesstattliche Erklärung

„Ich, als Autor / als Autorin und Urheber / Urheberin der vorliegenden Arbeit, bestätige mit meiner Unterschrift die Kenntnisnahme der einschlägigen urheber- und hochschulrechtlichen Bestimmungen (vgl. Urheberrechtsgesetz idgF sowie Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt und Gedankengut jeglicher Art aus fremden sowie selbst verfassten Quellen zur Gänze zitiert habe. Ich bin mir bei Nachweis fehlender Eigen- und Selbstständigkeit sowie dem Nachweis eines Vorsatzes zur Erschleichung einer positiven Beurteilung dieser Arbeit der Konsequenzen bewusst, die von der Studiengangsleitung ausgesprochen werden können (vgl. Satzungsteil Studienrechtliche Bestimmungen / Prüfungsordnung der FH Technikum Wien idgF).

Weiters bestätige ich, dass ich die vorliegende Arbeit bis dato nicht veröffentlicht und weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt habe. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht.“

.....
Ort, Datum

.....
Unterschrift

Kurzfassung

Das Ziel der vorliegenden Bachelorarbeit ist es, einen Überblick über das Machine Learning bereitzustellen und gängige Methoden miteinander zu vergleichen. Für ein besseres Verständnis werden auch die Grundlagen und typische Probleme im Machine Learning beschrieben.

Dazu wird der Prozess, wie von einem Datensatz Vorhersagen über unbekannte Daten getroffen werden können, vorgestellt. Dieser erstreckt sich von der Analyse der Daten, über das Training von Modellen bis hin zur Evaluierung und Überprüfung von Modellen.

Dabei zeigt sich, dass die Auswahl und richtige Aufbereitung der Daten eine ähnliche Auswirkung auf die Präzision von Vorhersagen hat, wie die Wahl des korrekten Modells. Außerdem ergibt die Analyse der Ergebnisse, dass die Beschneidung der Daten Auswirkungen auf die Anwendbarkeit eines trainierten Modells hat. Ein Modell kann daher für einen bestimmten Zweck trainiert werden.

Schlagwörter: Maschinelles Lernen, Lineare Regression, Modell, Feature, Überwachtes Lernen

Abstract

The goal of the present Bachelor thesis is to give an insight into machine learning and to compare commonly used methods. In order to gain a better understanding, the fundamentals of machine learning and typical problems are also described here.

To do this, the process of predicting values for unknown data by learning from a given dataset is presented. This includes data analysis and training models as well as evaluating and reviewing models.

The tests show that the selection and editing of data has a similar impact on the accuracy of predictions as choosing the correct model. In addition, the analysis of the results shows that the truncation of the data has implications for the applicability of a trained model. A model can thus be trained for a specific purpose.

Keywords: Machine Learning, Linear Regression, Model, Feature, Supervised Learning

Inhaltsverzeichnis

1 Einleitung.....	7
1.1 Struktur der Arbeit.....	7
1.2 Terminologie.....	8
1.2.1 Modell.....	8
1.2.2 Feature.....	8
1.2.3 Instanz.....	8
1.3 Verwandte Arbeit.....	8
2 Einführung ins Machine Learning.....	9
2.1 Was ist Machine Learning.....	9
2.2 Arten von Machine Learning.....	9
2.3 Supervised Learning.....	10
2.4 Modell.....	10
2.5 Feature.....	12
2.6 Probleme und Lösungen beim Machine Learning.....	13
2.6.1 Probleme mit Daten.....	13
2.6.2 Probleme mit Algorithmen.....	14
2.7 Bias und Varianz.....	15
2.7.1 Definition Bias.....	16
2.7.2 Definition Varianz.....	16
2.7.3 Bias/Varianz Tradeoff.....	16
2.8 Evaluierung eines Modells.....	17
2.8.1 Metriken.....	18
2.9 Zusammenfassung.....	19
3 Methodik.....	19
3.1 Datenanalyse & Featureerzeugung.....	19
3.2 Pipelining & Featureauswahl.....	20
3.3 Training & Evaluierung von Modellen.....	20
3.4 Test der Modelle.....	20
3.5 Ergebnisanalyse.....	20

4 Datensatz.....	20
5 Modelle.....	21
5.1 Lineare Modelle.....	21
5.1.1 Lineare Regression.....	23
5.1.2 Ridge Regression.....	23
5.1.3 Lasso Regression.....	24
5.2 Decision Tree Modelle.....	24
5.2.1 Decision Tree.....	24
5.2.2 Random Forest.....	26
6 Technologien.....	26
6.1 Scikit-learn.....	26
6.2 pandas.....	27
7 Praktischer Teil.....	27
7.1 Datenanalyse & Featureerzeugung.....	27
7.2 Pipelining & Featureauswahl.....	30
7.3 Training & Evaluierung der Modelle.....	33
7.4 Test der Modelle.....	34
7.5 Analyse.....	36
8 Aussicht.....	37

1 Einleitung

Wir sind heutzutage überwältigt von der Menge an Daten die produziert werden. Die Allgegenwärtigkeit von Computern und dem Internet bieten schier grenzenlos wirkende Möglichkeiten sie aufzuzeichnen, zu teilen und zu vermehren. Was man früher einfach aus Irrelevanz gelöscht oder gar nicht erst aufgezeichnet hätte, wird dank billigem Speicher aufbewahrt. Machine Learning ist ein Werkzeug mit dem aus diesem Berg an Daten Schlüsse gezogen werden können. Die Wörter Machine Learning beschreiben, dass Computer von Daten lernen können. (Frei übersetzt, Arthur Samuel, 1959) Machine Learning ist das Feld der Wissenschaft das Computern die Fähigkeit gibt zu lernen ohne explizit programmiert zu werden. Jedoch ist das Feld des Machine Learnings sehr breit gefächert.

Ziel dieser Arbeit ist es einen Überblick über grundlegende Methoden des Machine Learning anhand eines realen Datensatzes zu geben. Zur Veranschaulichung werden alle Kaufverträge von Grundstücken in Wien seit 1987 bis 2013 [8] herangezogen. Für die Kaufverträge soll eine Vorhersage über den Quadratmeterpreis getroffen werden. Die Methoden beschrieben und verglichen werden.

1.1 Struktur der Arbeit

Im Teil „Einführung ins Machine Learning“ werden einige Grundlagen erläutert, die ein Verständnis für den Bereich Machine Learning vermitteln sollen. Dazu gehören das Trainieren von Modellen, Validierungsmethoden und bekannte Probleme im Machine Learning. Dieser Teil ist für weniger versierte LeserInnen um die daraufhin aufgestellte Methodik besser zu verstehen. Er kann übersprungen werden, wenn diese schon mit der Materie vertraut sind.

Ist ein Grundwissen aufgebaut, wird im Kapitel „Methodik“ festgelegt wie der Vergleich der Modelle aufgestellt wird. Anschließend folgt eine kurze Beschreibung des untersuchten Datensatzes.

Das Kapitel „Modelle“ vertieft, das aus „Einführung ins Machine Learning“ erarbeitete Wissen. Die theoretischen Hintergründe zu den einzelnen Modellen werden genauer erklärt. Unter anderem die Funktionsweise, sowie deren Stärken und Schwächen.

Ist die Methodik und das theoretische Wissen des Lesers aufgebaut, wird die praktische Umsetzung mit einem Framework im Kapitel „Praktischer Teil“ beschrieben. Und die Verhalten der einzelnen Modelle beschrieben. Die Ergebnisse des praktischen Teils werden im Schlussteil analysiert. Etwaige Ausreißer werden genauer unter die Lupe genommen.

1.2 Terminologie

1.2.1 Modell

Mathematisches, bzw. statistisches Modell, welches zur Herleitung von Schlüssen aus bestehenden Daten verwendet wird.

1.2.2 Feature

Repräsentation von Daten welche von Modellen für Vorhersagen verwendet werden kann.

1.2.3 Instanz

Einzelner Eintrag aus einem Datensatz, welche einem Modell entweder zum Training oder zur Vorhersage gegeben wird.

1.3 Verwandte Arbeit

Sollte das Bedürfnis bestehen sich tiefergehend mit der Theorie und den mathematischen Grundlagen, sowie Methoden zur Validierung zu befassen, wird hier vor allem Data Mining von Ian Witten empfohlen. Auf dem auch große Teile der Arbeit aufbauen. Außerdem ist die Dokumentation des verwendeten Frameworks, Scikit-learn, sehr informativ aufgebaut, wenn es zu praktischen Anwendung der Modelle geht.

2 Einführung ins Machine Learning

2.1 Was ist Machine Learning

Machine Learning bezeichnet das Feld der Informatik, in dem es darum geht Computer aus Daten lernen zu lassen um in weitere Folge Vorhersagen treffen zu können [1], S 4. Daten sind dabei Beobachtungen aus der realen Welt, beispielsweise die Entwicklung von Aktienpreisen über einen gewissen Zeitraum. Eine Vorhersage könnte dann aussagen welche Aktien sich am besten für eine Investition eignen.

Diese Vorhersagen können mithilfe eines mathematischen Modells getroffen werden, welches die Beziehung zwischen einzelnen Anhaltspunkten in den vorhandenen Daten beschreibt [2], S 2. Um bei dem Beispiel mit den Aktien zu bleiben, könnte es etwa eine Formel sein die ehemalige Aktienpreise einer Firma, deren Einkommensentwicklung und die Entwicklung der Aktienpreisen von Firmen in der selben Sparte beinhaltet. Nun können mathematische Modelle lediglich numerische Werte mit einander in Verbindung stellen. Jedoch sind nicht alle Daten die gesammelt werden numerische Werte, wie zum Beispiel die Marke eines Autos.

Um dieses Dilemma lösen zu können werden Features benötigt. Sie sind numerische Repräsentationen von Daten [2], S 3. Es gibt viele Wege um Rohdaten in Features umzuwandeln. Bei der Umwandlung ist zu beachten, dass die Wahl der Features sich stark auf die Präzision eines Modells auswirken kann. Sind zu wenige aussagekräftige Features vorhanden kann das Modell keine präzise Vorhersage treffen. Ebenso können zu viele oder irrelevante Features dazu führen, dass ein komplexeres Modell benötigt wird um ähnlich gute Vorhersagen zu treffen, wie ein simples Modell mit besseren Features.

Der Prozess eine Vorhersage zu treffen besteht daher im Groben daraus die geeignetsten Features für das geeignetste Modell zu finden. Das ist allerdings ein zweiseitiges Schwert, denn die Wahl des Modells hat ebenso Auswirkung auf das Ergebnis wie die Wahl der Features [2], S 3.

2.2 Arten von Machine Learning

Die Anwendungsbereiche von Machine Learning sind mannigfaltig und werden grob in vier Bereiche eingeteilt: supervised, unsupervised, semisupervised und reinforcement learning. Die Einteilung ist dabei abhängig vom Maß an „supervision“. In dieser Arbeit kommen jedoch nur Supervised Learning Methoden zum Einsatz, da bereits Input- sowie Outputdaten bekannt sind.

2.3 Supervised Learning

Supervised Learning kommt dann zum Einsatz wenn es darum geht einen Output für einen gegebenen Input vorherzusagen und bereits Input/Output Paare bekannt sind. Ziel ist es ein Modell anhand dieser Input/Output Paare zu trainieren, welches präzise Vorhersagen für bisher ungesehene Daten treffen kann. Es muss also gut von einem Trainingsdatensatz auf einen Testdatensatz verallgemeinern können.

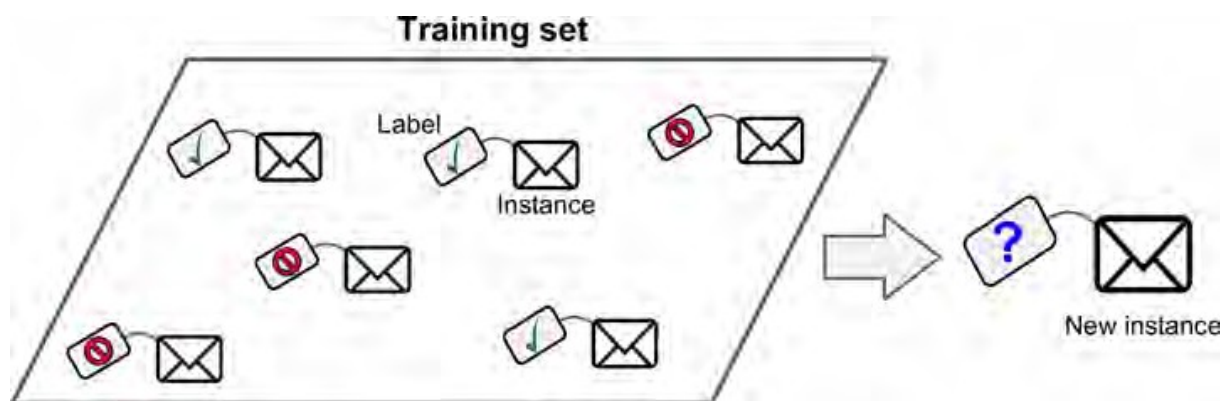


Abbildung 1: Trainingsdatensatz für Supervised Learning am Beispiel von Spammail-Klassifizierung ([1] S 8)

Die zwei größten Anwendungsgebiete für Supervised Learning sind Klassifizierungs- und Regressionsprobleme. Bei einer Klassifizierung geht es darum für einen gegebenen Input eine Klasse aus einer vordefinierten Liste vorherzusagen. Ein klassisches Beispiel hierfür ist die Klassifizierung von Iris-Blumen anhand ihrer Blütenblattlänge, bei welcher die Klassen die drei unterschiedlichen Arten sind: Iris Setosa, Iris Virginica und Iris Versicolor [3], S 15. Im Gegensatz dazu wird bei der Regression versucht eine Gleitkommazahl, eine reale Zahl, etwa den Preis eines Autos zu schätzen, wobei Attribute wie der Kilometerstand oder das Baujahr ins Gewicht fallen.

Die Varietät der Daten ist dabei ein ausschlaggebender Faktor wie komplex ein Modell sein kann ohne sich zu sehr an einen Trainingsdatensatz anzupassen. Für gewöhnlich hält ein größerer Datensatz eine größere Varietät bereit, wodurch besser verallgemeinert werden kann [3], S 27.

2.4 Modell

Ein Modell stellt im Supervised Learning einen approximierten Zusammenhang zwischen Input und Output dar [3], S 27.

Das Schätzen eines kontinuierlichen Wertes, wie zum Beispiel eines Grundstückspreises, fällt unter die Regressionsaufgaben. Die Regression beschreibt in der Statistik und weitergehend auch im Machine Learning das modellieren eines Zusammenhangs zwischen unabhängigen Variablen und einer abhängigen Variable. Die unabhängigen Variablen sind dabei der Input und die abhängige Variable der Output. Der Output ist also abhängig vom gegebenen unabhängigen Input. Im Fall dieser Arbeit ist die abhängige Variable der Grundstückspreis und die unabhängigen Variablen sind die restlichen Werte im Datensatz, wie zum Beispiel die Grundstücksfläche oder der Bezirk.

Die simpelste Variante eines Regressionsmodells ist das Lineare Modell. Es stellt den Zusammenhang zwischen Input und Output als lineare Funktion dar. Typischerweise sieht das Modell dann folgendermaßen aus:

$$y = x[0] \cdot w[0] + x[1] \cdot w[1] + \dots + x[n] \cdot w[n] + b$$

Formel 1: Lineares Modell

- y... Vorhersage
- x ... Feature
- w ... Gewichtung des Features
- b ... Bias

X[0] bis X[n] sind dabei die Features einer Instanz des Datensatzes. w und b sind dabei die erlernten Parameter aus dem Training. Diese Parameter bestimmen wie eine Vorhersage des Modells für einen gegebenen Input ausfällt.

Die Parameter w und b werden über Lernalgorithmen bestimmt. Ein Lernalgorithmus passt die Parameter so an, dass ein Modell den Zusammenhang zwischen Input und Output repräsentiert. Um das zu bewerkstelligen benötigt der Lernalgorithmus eine Metrik anhand derer er bestimmen kann wie gut das Modell angepasst ist. Diese Metrik wird über eine Cost oder auch Loss Function bestimmt, welcher die genutzte Metrik minimiert. Im Fall einer Linearen Regression versucht der Lernalgorithmus die Summe der Fehler über alle Vorhersagen zu minimieren [1], S 107.

Ist dieser Prozess abgeschlossen, ist das trainierte Modell für Vorhersagen einsatzbereit. Jedoch passiert dieses Training an einem Trainingsdatensatz. Ziel ist es aber, dass das Modell präzise Vorhersagen für unbekannte Daten treffen kann. Damit sich ein Modell nicht

zu sehr an einen Trainingsdatensatz anpasst, bieten viele Lernalgorithmen die Möglichkeit der Regularisierung. Die Regularisierung ermöglicht, dass der Lernalgorithmus in seiner Anpassungsfähigkeit eingeschränkt wird. Der Grad dieser Regularisierung wird über sogenannte Hyperparameter bestimmt, welche das Modell nicht direkt beeinflussen aber den Algorithmus der dieses erstellt [1], S472. Die Regularisierung unterscheidet sich aber von Lernalgorithmus zu Lernalgorithmus, weswegen diese näher in der Sektion Modelle beschrieben werden.

2.5 Feature

Features sind Repräsentationen von Rohdaten, sie werden erstellt, sodass ein Modell damit umgehen kann. Sie können viele Formen annehmen, wobei die meisten entweder als Continuous Feature oder als Categorical Feature auftreten.

Den Unterschied für einen Machine Learning Algorithmus macht dabei aus, dass ein Continuous Feature geordnet werden kann. So ist etwa ein Gebäude mit einer Höhe von 10,3 Metern kleiner als eines mit 11,5 Metern Höhe. Ein Categorical Feature ist jedoch meist nicht vergleichbar. Beispielsweise ist ein Porsche ein Porsche und kein Ferrari, es existieren keine Werte dazwischen.

Verschiedene Modelle können mit unterschiedlichen Arten von Features besser umgehen. Im Fall eines Linearen Modells werden Gewichtungen und Featurewerte multipliziert und aufaddiert (siehe: Formel 1: Lineares Modell). Sie könnte ein Categorical Feature nicht handhaben.

Es ist eine der Hauptaufgaben von Datenwissenschaftlern Features in die richtige Form für eine gegebene Aufgabenstellung zu bringen. Ein üblicher Weg um Categorical Features nutzbar für eine Lineare Regression zu machen ist das One-Hot-Encoding. Dabei wird ein Feature in mehrere Features aufgespalten und nur die zutreffende Kategorie wird mit einer 1 versehen während die anderen auf 0 gesetzt werden. So kann den Features bei der Linearen Regression eine Gewichtung zugeordnet werden [3], S 213.

Automarke	Ferrari	Porsche	Toyota	BMW
Ferrari	1	0	0	0
Porsche	0	1	0	0
Toyota	0	0	1	0
BMW	0	0	0	1

Tabelle 1: One Hot Encoding am Beispiel von Automarken

Im Kapitel „Praktischer Teil“ wird näher beschrieben welche Features wie umgewandelt werden um für die verschiedenen Modelle nutzbar gemacht zu werden.

2.6 Probleme und Lösungen beim Machine Learning

Beim Erstellen eines Modells sind die Haupteinflussfaktoren die gewählten Daten und der gewählte Lernalgorithmus. In diesem Kapitel werden Probleme die mit Daten und Algorithmen auftreten können behandelt.

2.6.1 Probleme mit Daten

Mit Daten können die mehrere Probleme auftreten und diese zu beheben involviert manuelle Arbeit, wie das Beschaffen von weiteren Daten oder ausbessern ebenjener.

2.6.1.1 Unzureichende Menge an Daten

Sind nicht genug Daten vorhanden kann ein Lernalgorithmus nur schwer verallgemeinern. Michele Bank und Eric Brill [4] beschreiben zu diesem Thema etwa das verschiedene Machine Learning Algorithmen ähnlich gut funktionieren sobald eine gewisse Menge an Daten vorhanden ist.

2.6.1.2 Unrepräsentative Daten

Problematisch ist es auch, wenn die Daten zwar in ausreichender Menge vorhanden sind, sie aber Informationen einseitig darstellen. Das kann oft die Folge davon sein, dass die Methode zur Beschaffung von Daten fehlerhaft ist [1], S 22. Wenn etwa versucht werden würde ein Wahlergebnis vorherzusagen, dabei aber nur Menschen mit Hochschulabschluss befragt werden, werden die Daten ein verzerrtes Bild abliefern.

2.6.1.3 Fehler- bzw. Lückenhafte Daten

Wenn Daten fehler- oder lückenhaft sind, wird im Training kein gutes Modell erzeugt werden können. Ein Lernalgorithmus versucht den Zusammenhang zwischen Input und Output herzustellen und dies ist nicht möglich wenn der Input Fehler aufweist. In diesem Fall ist es hilfreich diese entweder zu ergänzen oder aus dem Trainingsdatensatz zu entfernen [1], S 25.

2.6.1.4 Irrelevante Features

Sind die Daten gesäubert von Fehlern, kann es immer noch sein das ein Modell keine präzisen Vorhersagen trifft. Das passiert, wenn die verwendeten Daten irrelevante Features aufweisen [1], S 22. So wäre es vermutlich nicht hilfreich bei der Bestimmung eines Aktienkurses einzubeziehen wieviele Katzen gerade auf Island leben.

2.6.2 Probleme mit Algorithmen

Bei der Erstellung eines Modells kann es hauptsächlich zu zwei Problemen kommen: Der Überanpassung oder die Unteranpassung an einen Trainingsdatensatz. Je nach Komplexität des Datensatzes oder gewählten Modells kann einer der beiden Effekte auftreten [3], S 31.

2.6.2.1 Überanpassung

Die Überanpassung tritt meist auf, wenn ein zu komplexes Modell für einen Datensatz angewendet wird oder aber zu wenig Trainingsdaten vorhanden sind. Grund dafür ist, dass komplexe Modelle „noise“ auch als Teil des, den Daten unterliegenden, Muster zuschreiben kann und dies in seine Schätzungen einbaut. Das hat zu Folge, dass das trainierte Modell zwar präzise Vorhersagen am Trainingsdatensatz treffen kann, jedoch versagt wenn es eine Vorhersage für eine neue Instanz treffen muss.

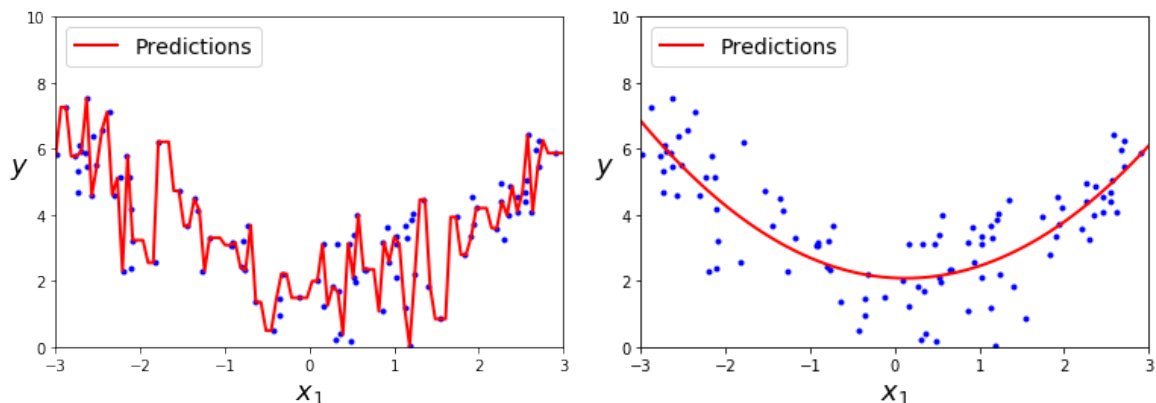


Abbildung 2: Gegenüberstellung von einem überangepassten Modell (links) und einem gut verallgemeinernden Modell (rechts).

Es gibt einige Möglichkeiten dem entgegenzuwirken. So kann es zum Beispiel helfen die Anzahl der Trainingsdaten zu erhöhen. Auch der Noise in den Trainingsdaten kann reduziert werden indem Ausreißer entfernt werden oder fehlerhafte Daten entweder vervollständigt oder gelöscht werden. Eine weitere Möglichkeit ist es ein simpleres Modell zu wählen oder den Lernalgorithmus durch Regularisierungen einzuschränken [3], S 29.

2.6.2.2 Unteranpassung

Dem gegenüber steht die Unteranpassung. Sie ist das genaue Gegenteil und tritt auf wenn ein zu simples Modell für einen zu komplexen Datensatz gewählt wird. Wenn ein Modell etwa zu wenige Parameter bietet um ein komplexes Problem lösen zu können, wird es höchstwahrscheinlich zu einer Unteranpassung kommen.

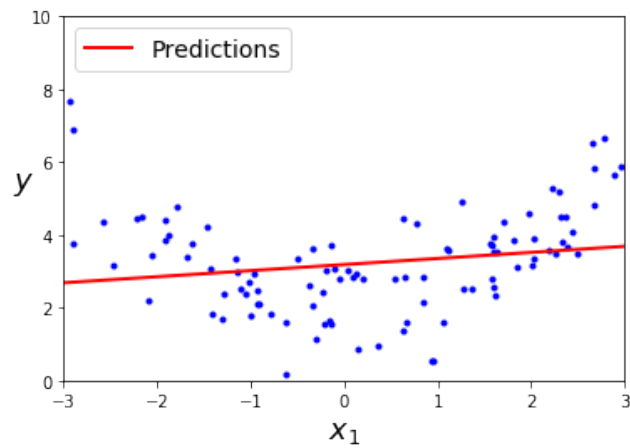


Abbildung 3: Unterangepasstes Lineares Modell

Um dieses Problem zu lösen stehen mehrere Möglichkeiten zur Verfügung. So kann ein komplexeres Modell gewählt werden, welches sich besser anpassen kann. Sowie bei der Überanpassung kann auch die Regularisierung angepasst werden. Dem Lernalgorithmus können aber auch bessere Feature zur Verfügung gestellt werden [3], S 30.

2.7 Bias und Varianz

Fehler bei Vorhersagen können in zwei Kategorien unterteilt werden. Solche die durch Bias verursacht werden und andere die durch Varianz verursacht werden [5], S 1. Ein perfektes Model würde keinen Bias und keine Varianz aufweisen und immer perfekte Vorhersagen treffen. In der Realität ist dies jedoch kaum zu erreichen, weswegen hier kurz darauf eingegangen wird.

Die Folgende Illustration stellt die verschiedenen Auswirkungen von Bias und Varianz als Präzision an einer Zielscheibe dar:

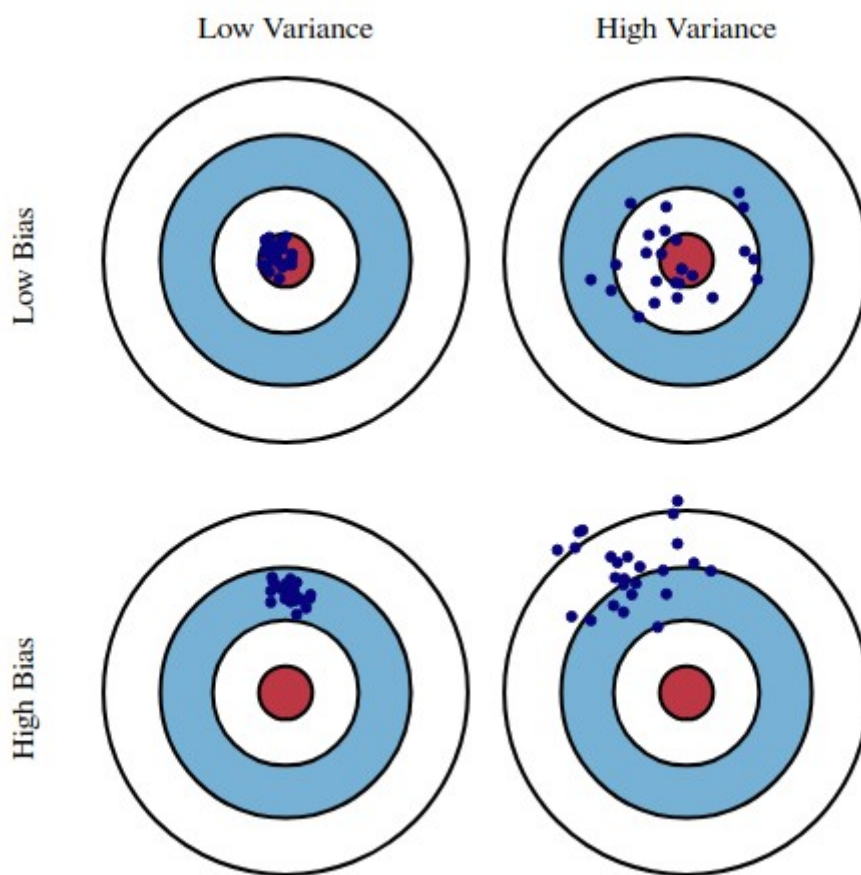


Abbildung 4: Auswirkung von Bias und Varianz ([5], S 1)

2.7.1 Definition Bias

Fehler die durch Bias verursacht werden, werden als durchschnittliche Distanz zwischen einem vorhergesagten Wert und dem tatsächlichen Wert über mehrere Vorhersagen gemessen. [5], S 1.

2.7.2 Definition Varianz

Fehler die durch Varianz verursacht werden, werden als die Distanz zwischen dem durchschnittlichen Fehler über mehrere Vorhersagen an unterschiedlichen Datensätzen (z.B.: Trainings- und Testdatensatz) gemessen [5], S 1.

2.7.3 Bias/Varianz Tradeoff

In der Sektion Probleme mit Algorithmen wurden als Hauptprobleme die Unter- bzw. Überanpassung beschrieben. Diese Probleme reflektieren die Fehler durch Bias und Varianz. Weist ein Modell einen niedrigen Bias auf so trifft es höchstwahrscheinlich gute Vorhersagen am Trainingsdatensatz auf, jedoch schlechte wenn es auf neue Daten angewendet wird. Was ein typisches Beispiel einer Überanpassung ist. Gleichzeitig wird die Varianz sehr hoch sein, da die Präzision am Trainingsdatensatz ausfällt, während sie am Testdatensatz niedrig ausfällt.

Umgekehrt gilt dasselbe. Weist ein Modell einen hohen Bias auf, wird die Varianz niedrig ausfallen. Daher gilt es eine Balance zwischen Bias und Varianz zu finden um eine möglichst verallgemeinerndes Modell zu finden.

2.8 Evaluierung eines Modells

Die Evaluierung von Modellen ist einer der wichtigsten Bestandteile im Machine Learning Prozess. Es benötigt eine systematische Vorgehensweise um herausfinden zu können welches Modell für eine gegebene Aufgabenstellung am besten geeignet ist.

Die Präzision eines Modells an seinen Trainingsdaten ist kein guter Indikator dafür wie gut es mit neuen unabhängigen Daten umgehen kann. Das Modell hat seinen Entscheidungsprozess anhand dieser aufgebaut. Daher wird die Präzision sehr optimistisch ausfallen [6], S 163.

Aufgrund dessen ist es notwendig ein Modell anhand von ihm unbekannten Daten zu evaluieren. Dies stellt kein Problem dar, wenn genug Daten vorhanden sind. Wie im Kapitel „Probleme mit Daten“ beschrieben kann das allerdings oft schwierig sein. Deshalb wird ein vorhandener Datensatz oft in drei Teile aufgeteilt: Trainingsdaten, Validierungsdaten, Testdaten.

Die Trainingsdaten werden verwendet um ein oder mehrere Modelle zu trainieren, die Validierungsdaten dazu ein Modell zu wählen oder die Parameter zu optimieren. An den Testdaten wird schließlich die entgeltliche Präzision gemessen. Dabei ist zu beachten, dass die drei Teile jeweils ein unabhängiges Set bilden damit ein möglichst reales Szenario nachgestellt werden kann [6], S 164.

Hierbei entsteht jedoch ein Dilemma, denn um ein gutes Modell erstellen zu können werden möglichst viele Trainingsdaten benötigt. Diese werden jedoch beschnitten, da ein Teil für die Validierung und den entgeltigen Test verwendet werden. Um dieses Dilemma zu umgehen gibt es mehrere Methoden, wobei die „k fold cross validation“ die Populärste ist.

Bei der „k fold cross validation“ werden die Trainingsdaten in k Subsets aufgeteilt. Typischerweise sind es 10 Subsets, da diese Anzahl erfahrungsgemäß gute Ergebnisse liefert [6], S 168. In diesem Fall werden 9 der Subsets für das Training eines Modells verwendet und das Letzte zum testen. Dieser Vorgang wird 10 mal wiederholt, sodass jedes Subset sowohl zum Trainieren als auch zur Validierung verwendet wird. Ist dieser Vorgang abgeschlossen kann eine Aussage über den Bias und die Varianz eines Modells getroffen werden.

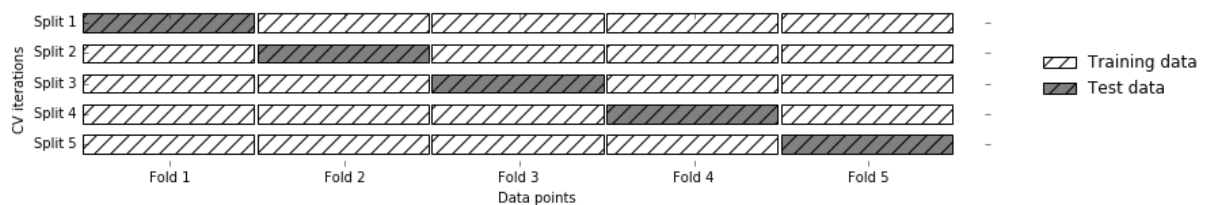


Abbildung 5: Beispiel für eine „5 fold cross validation“ ([3], S 254).

Ein Nachteil besteht darin, dass mehrmals trainiert und getestet werden muss. Was in anbeacht größerer Datensätze die Durchlaufzeit erhöht. Es ist also sehr Rechenintensiv gute Messungen zu erhalten [6], S 168.

2.8.1 Metriken

Präzisionswertungen fallen je nach Aufgabenstellung anders aus. So wird bei einem Klassifizierungsproblem etwa die Fehlerrate von Vorhersagen bestimmt. Bei einem Regressionsproblem kann jedoch nicht einfach bestimmt werden ob eine Vorhersage richtig oder falsch ist. Da ein numerischer Wert geschätzt wird, treten Fehler in verschiedenen Größen auf [6], S194.

2.8.1.1 Root Mean Squared Error

Die wohl am häufigsten benutzte Metrik um die Präzision von Regressionsmodellen festzustellen ist der Mean Squared Error [6], S 195. Hierbei wird der Fehler aller Vorhersagen zusammengerechnet und ein Durchschnittsfehler über alle Vorhersagen errechnet. In dieser Arbeit wird der Root Mean Squared Error zum Vergleich herangezogen. Das heißt lediglich, dass der errechnete Fehler in einer ähnlichen Dimension ausfällt, wie die Vorhersage.

$$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}}$$

Formel 2: Root Mean Squared Error

- p ... Vorhersage
- a ... erwarteter Wert
- n ... Anzahl der Instanzen

2.8.1.2 Mean Absolute Error

Der Mean Absolute Error ist eine Alternative zum Root Mean Squared Error [6], S 195. Der einzige Unterschied liegt darin, dass die Vorzeichen der Fehler nicht in Betracht gezogen werden und durch die fehlende Potenzierung Ausreißer nicht so stark ins Gewicht fallen.

$$\frac{|p_1 - a_1| + \dots + |p_n - a_n|}{n}$$

Formel 3: Mean Absolute Error

- p ... Vorhersage
- a ... erwarteter Wert
- n ... Anzahl der Instanzen

Zusammenfassend sind sowohl der Root Mean Squared Error als auch der Mean Absolute Error Metriken, welche eine Distanz zwischen einem Vektor von Vorhersagen und Zielwerten messen [1], S 39.

Da diese Arbeit einen generellen Vergleich der Modelle behandelt, wird auch die Größe der Rechenkomplexität in Betracht gezogen. Komplexere Modelle können zwar eine höhere Präzision erreichen aber sie zu trainieren oder zu befragen könnte gleichzeitig hinsichtlich der Rechenleistung umso intensiver sein. Deshalb werden auch Zeitmessungen vorgenommen die zum Vergleich dienen sollen.

2.9 Zusammenfassung

Im Allgemeinen geht es darum, dass ein Modell den Zusammenhang zwischen Input und Output in einem Trainingsdatensatz darstellen kann. Daten müssen als Features dargestellt werden sodass Modell sie verwenden können. Um präzise Vorraussagen für neue Instanzen treffen zu können, muss ein Modell mithilfe eines anpassbaren Lernalgorithmus trainiert werden. Trainierte Modelle können anhand entsprechender Metriken verglichen werden und das beste Modell für eine gegebene Aufgabenstellung gewählt werden.

3 Methodik

Ziel der Arbeit ist es die Modelle zu vergleichen anhand ihrer Präzision und Geschwindigkeit an einem realen Datensatz. Dazu sind die folgenden Schritte notwendig:

3.1 Datenanalyse & Featureerzeugung

Als erstes wird eine Analyse der vorhandenen Daten durchgeführt. Dabei wird festgestellt welche Features des Datensatzes signifikant für Vorhersagen sind. Außerdem werden auch Kombinationen aus verschiedenen Features erstellt und überprüft ob diese eventuell eine größere Korrelation mit dem Zielwert besitzen.

3.2 Pipelining & Featureauswahl

Ist die Analyse abgeschlossen, wird der Datensatz aufbereitet sodass er von den Modellen genutzt werden kann. Dazu wird eine Pipeline erstellt welche die Trainings- und Testdaten in ein einheitliches Machine Learning taugliches Format umwandelt. Diese Pipeline extrahiert die notwendigen Daten aus den Rohdaten. Um jedoch eine bessere Vergleichbarkeit gewährleisten zu können, werden zwei verschiedene Pipelines erstellt. Das geschieht, da unterschiedliche Modelle mit verschiedenen Daten besser umgehen können.

3.3 Training & Evaluierung von Modellen

Ist die Vorbereitung der Daten abgeschlossen, werden mehrere Modelle trainiert und einander gegenübergestellt. Dazu wird ihre Präzision an einem Validierungsdatensatz mithilfe von Cross Validation überprüft.

3.4 Test der Modelle

Anschließend werden die Modelle mit einem Testdatensatz getestet um ihre Präzision an bisher unbekannten Daten zu testen.

3.5 Ergebnisanalyse

Abschließend werden etwaige Ausreißer und andere mögliche unerwartete Ergebnisse näher beleuchtet.

4 Datensatz

Der bearbeitete Datensatz besitzt die folgenden 17 Spalten. Im praktischen Teil wird abgehandelt, welche Spalten verwendet werden beziehungsweise wie die Korrelationen zwischen den einzelnen Features und welche neuen Features aus Kombinationen mehrerer Spalten produziert werden.

Spalte	Datentyp	Beispiel
Id	Numerisch	7671
price_per_m2	Numerisch	224.351483991587
transaction_ym	Numerisch	2013-12
plot_area	Numerisch	4279
plot_share	Numerisch	1
property_type	Kategorisch	Betriebsobjekt
cadestral	Kategorisch	Kagran
district	Kategorisch	Donaustadt
contract_type	Kategorisch	Kaufvertrag
land_use	Kategorisch	GB
land_use_share	Numerisch	1
protection_zone	Kategorisch	No
building_ban	Kategorisch	No
building_height	Numerisch	10.5
building_floors	Numerisch	6
building_year	Numerisch	1990
longitude	Numerisch	16.4641634
latitude	Numerisch	48.259071

Tabelle 2: Übersicht über den verwendeten Datensatz

5 Modelle

In der Sektion Einführung ins Machine Learning wurde bereits angeschnitten was ein Modell im Supervised Machine Learning ausmacht und welche Faktoren einen Einfluss auf Training und Vorhersagen haben können. Dazu gehören im allgemeinen der angewendete Lernalgorithmus und die jeweilige Regularisierung. Im Folgenden werden die Modelle, welche im „praktischen part“ zum Einsatz kommen vorgestellt. In den einzelnen Abschnitten zu den Modellen wird ihre Funktionsweise erklärt sowie die Regularisierungen die auf sie angewendet werden können. // REPHRASE praktischer part when finsihed

5.1 Lineare Modelle

Lineare Modelle gehören zu den simpelsten Machine Learning Modellen. Sie stellen den Zusammenhang zwischen Input und Output als lineare Funktion der Input Features dar. Das bedeutet, sie erstellen eine Vorhersage indem sie die gewichtete Summe aller Input Features plus eines Bias zusammenaddieren [3], S 47.

$$y = x[0] \cdot w[0] + x[1] \cdot w[1] + \dots + x[n] \cdot w[n] + b$$

Formel 4: Lineares Modell

- y ... Vorhersage
- x ... Feature
- w ... Gewichtung des Features
- b ... Bias

w und b sind hierbei unbekannt und müssen erst durch Training erlernt werden. Dazu können verschiedene Lernmethoden verwendet werden.

Wenn der verwendete Datensatz nur ein Feature besitzen würde dann dann sähe die Formel folgendermaßen aus: $y = x[0] \cdot w[0] + b$

Graphisch dargestellt wäre es eine Gerade in einem zweidimensionalen Raum.

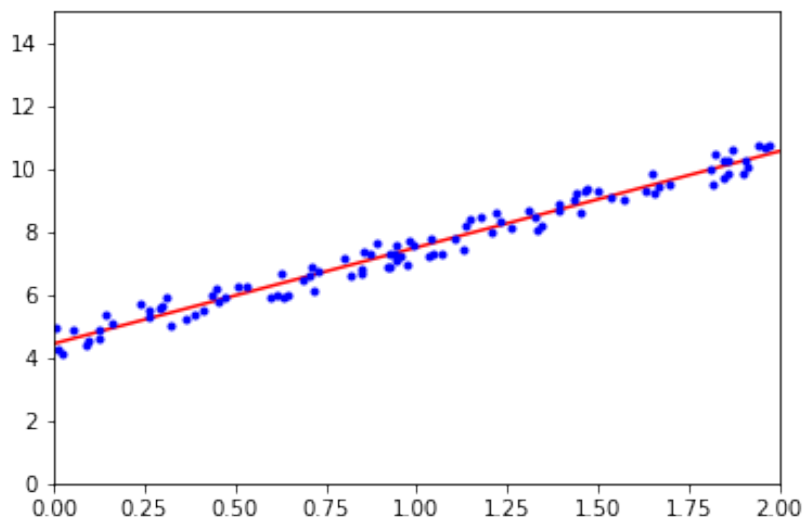


Abbildung 6: Vorhersagen eines Linearen Modells im zweidimensionalen Raum.

Je mehr Features verwendet werden desto mehr Dimensionen erhält das Lineare Modell, was wiederum seine Komplexität steigert. „Eine Lineares Regressionsmodell zeichnet sich dadurch aus, dass Vorhersagen entlang einer Linie liegen wenn ein Feature verwendet wird, auf einer Fläche wenn zwei Features verwendet werden und in einer Hyperebene wenn mehr Features verwendet werden [3], S 48. Damit wird ein Lineares Modell präziser je mehr Features vorhanden sind, jedoch führt zunehmende Komplexität zu Überanpassung [3], S 51.

5.1.1 Lineare Regression

Bei der eine Linearen Regression, auch Kleinste-Quadrat Methode werden die Parameter w und b so gewählt, dass die Summe der quadrierten Fehler minimal ausfällt [6], S 128.

Mathematisch löst die Kleinste-Quadrat Methode folgendes Problem [7]:

$$\min_w ||(Xw - y)||^2$$

- X ... Featurevektor
- w ... Gewichtungsvektor
- y ... Ergebnisvektor

Die herkömmliche Lineare Regression bietet jedoch keine Möglichkeit zur Regularisierung, woraus resultiert, dass es keine Möglichkeit gibt den Lernalgorithmus zu beeinflussen.

Die Rechenkomplexität des Trainings fällt dabei wie folgt: $O(n \cdot p^2)$ [7]. Sprich sie steigt mit der Anzahl an Features exponentiell an. Jedoch ist können Vorhersagen schnell getroffen werden. Bei Vorhersagen liegt die Rechenkomplexität bei $O(n)$ [1], S 110.

5.1.2 Ridge Regression

Die Ridge Regression ist eine Erweiterung der Linearen Regression. Auch bei der Ridge Regression wird die Kleinste-Quadrat Methode angewandt, allerdings wird sie hier um einen L2-Regularisierungsterm erweitert. Dieser verhindert eine Überanpassung, da der Regularisierungsterm die Gewichtungen der einzelnen Features gering hält. Daraus resultiert, dass die abhängige Variable weniger sensibel gegenüber den unabhängigen Variable ist.

Der L2-Regularisierungsterm sieht wie folgt aus:

$$\alpha \cdot \|w\|^2$$

- α Komplexitätsparameter
- $\|w\|$... Gewichtsvektor

Der Term addiert die quadrierte Summe aller Gewichtungen und multipliziert sie mit einem Komplexitätsparameter. Je höher α ausfällt, desto stärker werden die Gewichtungen beschnitten und sie können sich 0 annähern aber erreichen nie 0 [7]. Wenn α auf 0 gesetzt wird dasselbe Modell erzeugt wie bei der herkömmlichen Linearen Regression.

5.1.3 Lasso Regression

Die Lasso Regression ist wie schon die Ridge Regression eine Erweiterung der Linearen Regression. Ihre Funktionsweise ähnelt der der Ridge Regression. Der Unterschied besteht darin, dass der Kleinste-Quadrate Methode ein L1 statt einem L2-Regularisierungsterm hinzugefügt wird.

Der L1-Regularisierungsterm sieht wie folgt aus:

$$\alpha \cdot \|w\|$$

- α Komplexitätsparameter
- $\|w\|$... Gewichtsvektor

Der Unterschied zum L2-Regularisierungsterm liegt darin, dass manche Werte im Gewichtsvektor auf 0 gesetzt werden. Daraus folgt dass manche Features nicht ins endgültige Modell einfließen. Dadurch wird ein schlankeres Modell erstellt. Die Lasso Regression vollzieht also eine Art Featureauswahl indem sie die Gewichtungen von unwichtigen Features auf 0 setzt. Wie bei der Ridge Regression hängt der Grad der Regularisierung von α ab [3], S 55.

5.2 Decision Tree Modelle

5.2.1 Decision Tree

Decision Trees sind sehr vielseitige Machine Learning Algorithmen. Man sie für eine Vielzahl von Problemen einsetzen [6], S 209. Da das behandelte Problem in dieser Arbeit aber einen numerischen Output verfolgt, wird hier nur auf die Regression mit Decision Trees eingegangen.

Im Prinzip ist ein Decision Tree eine Abfolge von binären Abfragen, auch Tests genannt. Das bedeutet wenn eine bestimmte Bedingung erfüllt wird, dann bewegt man sich im Entscheidungsprozess eine Ebene nach links oder respektive rechts runter [1], S 167.

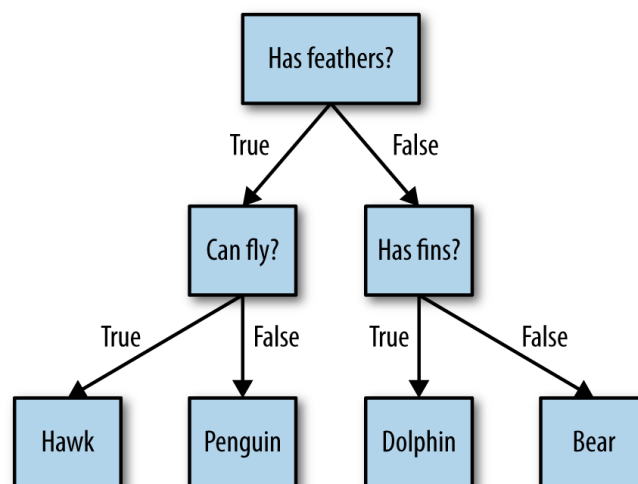


Abbildung 7: Simpler Decision Tree für eine Tierklassifizierung ([3], S 73)

Um einen Decision Tree zu erstellen, sucht der Lernalgorithmus unter allen möglichen Tests den Test, der am Aussagekräftigsten über die Zielvariable ist [3], S 74. Im Fall eines Continuous Feature überprüft der Test ob Feature A größer als Wert B ist. Graphisch dargestellt wird eine Decision Boundary durch eine Punktwolke gelegt.

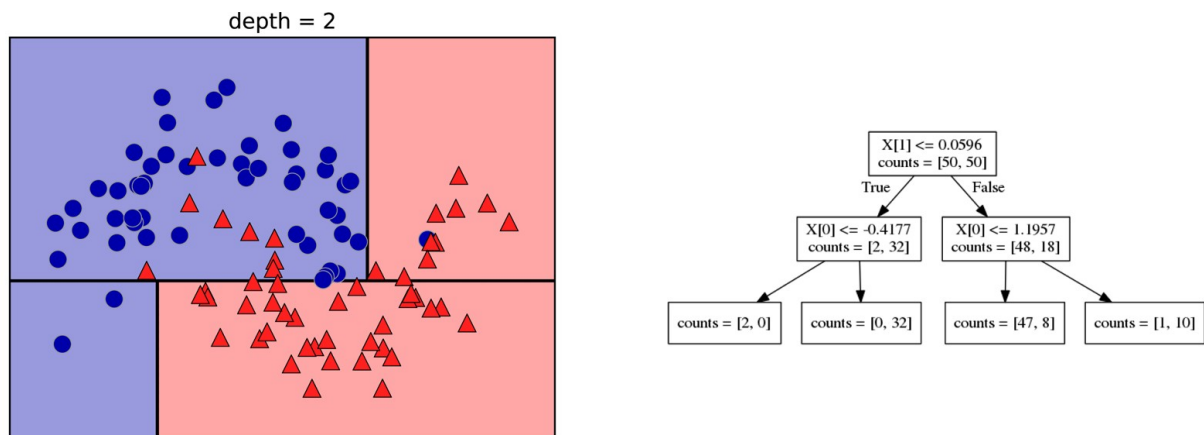


Abbildung 8: Decision Boundary eines Decision Trees mit der Tiefe 2 ([3], S 75)

Wenn man den Decision Tree dabei nicht regularisiert, dann tendiert er stark zur Überanpassung ans Trainingsset. Das liegt daran, dass die Decision Boundaries rekursiv produziert werden können bis das Trainingsset optimal aufgeteilt ist [3], S 75. Dies wiederum führt dazu, dass wenn mit dem trainierten Modell neue Vorhersagen getroffen werden, die Fehlerrate extrem hoch ist.

Um der Überanpassung entgegenzusteuern gibt es prinzipiell zwei populäre Strategien. Entweder der Decision Tree wird erstellt und Teile des Trees welche wenig Informationsgehalt werden entfernt oder die Erstellung eines zu komplexen Trees wird verhindert indem etwa eine maximale Tiefe oder maximale Anzahl an Blättern angegeben wird. Diese Strategien werden als Pruning, zu deutsch Beschneidung, genannt [3], S 76.

Für gewöhnlich beträgt die Rechenkomplexität um einen balancierten Tree zu erstellen $O(n_{samples} \cdot n_{features} \cdot \log(n_{samples}))$ und die Komplexität einer Abfrage $O(\log(n_{samples}))$. Der Lernalgorithmus versucht prinzipiell immer balancierte Trees zu erstellen, jedoch ist das nicht immer möglich [7]. Das verwendete Framework nutzt jedoch eine effiziente Variante für die Erstellung, welche sicherstellt, dass die oben genannte Rechenkomplexität eingehalten wird.

5.2.2 Random Forest

Der Random Forrest gehört zu den Ensemble Methoden. Diese haben die Eigenschaft, dass sie nicht aus einem einzelnen Modell sondern aus mehreren bestehen. Der Random Forest besteht aus mehreren Decision Trees. Soll eine Vorhersage getroffen werden, gibt jeder Tree seine Vorhersage ab, von welchen dann ein Durchschnitt errechnet wird [3], S 85.

Die Idee hinter dem Random Forest ist es, die Tendenz der Decision Trees zur Überanpassung auszugleichen. Die einzelnen Decision Trees bekommen dabei, jeweils leicht abgeänderte Trainingssets oder unterschiedliche Features um eine Diversität zwischen

den Trees zu erzeugen [3], S 86. Dadurch wird die Varianz in den Vorhersagen des Random Forests reduziert.

Die Regularisierung des Random Forests erfolgt dabei durch die Beschränkung der Features pro Decision Tree oder wie die Erzeugung von Subsets der Trainingsdaten beschränkt wird [3], S 86.

Die Rechenkomplexität des Random Forests ist dabei sehr variabel. Sie hängt davon ab wie komplex die untergeordneten Decision Trees erzeugt werden und wieviele Trees erzeugt werden.

6 Technologien

6.1 Scikit-learn

scikit-learn ist eine Open Source Machine Learning Library, geschrieben in Python. Sie stellt die gängigsten Machine Learning Methoden für Klassifizierungs-, Regressions und Clusteringaufgaben zur Verfügung.

6.2 pandas

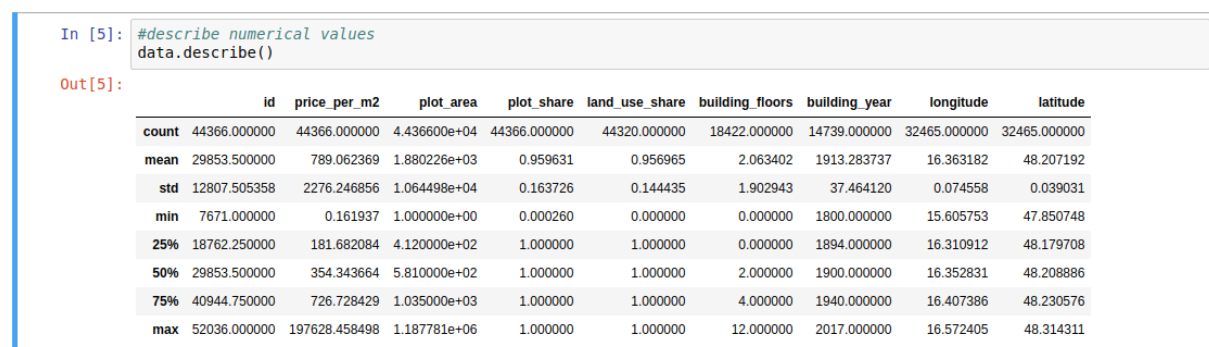
pandas ist eine Open Source Library zur Datenanalyse. Sie stellt einfach zu verwendende Datenstrukturen und Datenanalyse Tools bereit. Geschrieben in Python.

7 Praktischer Teil

Um die vorgestellten Modelle zu vergleichen wird ihre Performance anhand der vorgestellten Metriken überprüft. Dazu wird ein Datensatz aller Kaufverträge von Wiener Grundstücken zwischen 1987 und 2013 herangezogen [8]. Mithilfe der Modelle soll für Kaufverträge eine Quadratmeterpreis vorhergesagt werden. Da die Auswahl und Aufbereitung der Daten einen erheblichen Einfluss auf die Präzision der Modelle hat, werden die Modelle zweimal getestet. Im ersten Anlauf werden die Modelle mit unaufbereiteten Daten getestet, im zweiten Anlauf wird versucht, weniger dafür aussagekräftigere Features zu verwenden. Die Modelle werden abschließend gegenübergestellt und verglichen ob sich diese unter verschiedenen Bedingungen anders verhalten.

7.1 Datenanalyse & Featureerzeugung

Die Datenanalyse beginnt mit einer Übersicht über alle Spalten die im Datensatz vorhanden sind. Nachdem die Daten mithilfe von pandas als Dataframe eingelesen wurden kann mit dem Befehl *describe()* ein erster Blick auf diese geworfen werden:



```
In [5]: #describe numerical values
data.describe()
```

Out[5]:

	id	price_per_m2	plot_area	plot_share	land_use_share	building_floors	building_year	longitude	latitude
count	44366.000000	44366.000000	4.436600e+04	44366.000000	44320.000000	18422.000000	14739.000000	32465.000000	32465.000000
mean	29853.500000	789.062369	1.880226e+03	0.959631	0.956965	2.063402	1913.283737	16.363182	48.207192
std	12807.505358	2276.246856	1.064498e+04	0.163726	0.144435	1.902943	37.464120	0.074558	0.039031
min	7671.000000	0.161937	1.000000e+00	0.000260	0.000000	0.000000	1800.000000	15.605753	47.850748
25%	18762.250000	181.682084	4.120000e+02	1.000000	1.000000	0.000000	1894.000000	16.310912	48.179708
50%	29853.500000	354.343664	5.810000e+02	1.000000	1.000000	2.000000	1900.000000	16.352831	48.208886
75%	40944.750000	726.728429	1.035000e+03	1.000000	1.000000	4.000000	1940.000000	16.407386	48.230576
max	52036.000000	197628.458498	1.187781e+06	1.000000	1.000000	12.000000	2017.000000	16.572405	48.314311

Abbildung 9: Überblick über numerische Spalten des Datensatzes

Diese Übersicht bietet einen Einblick in die Eigenschaften der verschiedenen Spalten. *count*, *mean*, *min*, *max* sind dabei selbsterklärend. Zu beachten hierbei ist das pandas nicht vorhandene Werte nicht miteinbezieht, so sind in der Spalte *building_year* nur 14739 Einträge vorhanden. Die %-Reihen bezeichnen jeweils ein Quartil, in der Spalte *price_per_m2* liegen 25% der Einträge unter 181,68€, 50% unter 354,34€ und 75% unter 726,73€.

Auffällig an der Spalte *price_per_m2* ist, dass der Durschnitt bei 789,06€ liegt aber der Maximalwert bei knapp 200000,-€. Das deutet an, dass diese Spalte mindestens einen Ausreißer beinhaltet.

Für die numerischen Werte des Dataframes kann auch eine Korrelationsmatrix erstellt werden Abbildung 10. Diese gibt für alle Werte eine positive oder negative Korrelation im Bereich [-1,1] an. Je näher die Werte an 1 oder -1 liegen, desto höher die Korrelation.

```
In [7]: corrmatrix = data.corr()

In [8]: corrmatrix["price_per_m2"].sort_values(ascending=False)
Out[8]: price_per_m2      1.000000
building_floors      0.385326
plot_share           0.002082
latitude            -0.011045
plot_area            -0.028618
longitude            -0.038409
land_use_share       -0.050230
building_year        -0.051976
id                   -0.133134
Name: price_per_m2, dtype: float64
```

Abbildung 10: Korrelation numerischer Spalten mit dem Quadratmeterpreis

Von den vorhandenen Daten ist ersichtlich, dass die einzige Spalte die eine signifikante Korrelation mit dem *price_per_m2* hat *building_floors* ist. Allerdings ist zu beachten, dass hier nur lineare Korrelationen hervorgehoben werden. Zum Beispiel steigt der Quadratmeterpreis scheinbar stark, je mehr Stockwerke ein Gebäude hat.

Mit dem Befehl *describe(include="O")* kann noch ein Überblick über die kategorischen Daten erzeugt werden.

```
In [6]: #describe categorical values
data.describe(include="O")
Out[6]:
```

	transaction_ym	property_type	cadastral	district	contract_type	land_use	protection_zone	building_ban	building_height
count	44366	44366	44366	44294	44366	44260	44366	44366	7800
unique	324	16	88	23	11	56	2	2	97
top	1988-12	unbebaut	Eßling	Donaustadt	Kaufvertrag	WI/GBI	No	No	6.5
freq	296	15345	2270	8476	38853	12099	43306	43857	2763

Abbildung 11: Überblick über kategorische Spalten des Datensatzes

Die kategorischen Daten beschreiben Eigenschaften wie die Zugehörigkeit eines Grundstücks zu einem bestimmten Bezirk oder einer Grundstückswidmung. Der Umgang mit kategorischen Daten ist schwieriger, da nicht einfach eine Korrelationsmatrix erzeugt werden kann. Allerdings bieten einige Modelle wie der Decision Tree die Möglichkeit Feature Bedeutungen zu erzeugen. Diese sagen aus wie sehr diese Features bei einer Schätzung ins Gewicht fallen.

Jedoch können einige der Features in numerische Werte umgewandelt werden. So ist es möglich aus der Spalte *transaction_ym* die Jahreszahl zu extrahieren. Die Spalte *building_height* lässt sich auch als Zahl interpretieren. Allerdings ist die Spalte mit lediglich

7800 Einträgen schwach vertreten und bei näherer Betrachtung auch sehr unregelmäßig, sodass zu viel händische Arbeit notwendig wäre um diese brauchbar zu machen.

Nach diesem oberflächlichen Einblick werden einige Features erzeugt, welche Möglicherweise eine signifikantere Aussagekraft über den Quadratmeterpreis haben. Dem Datensatz werden keine zusätzlichen Daten hinzugefügt.

In Abbildung 9 ist sichtbar, dass für gut 75% der Daten Längen- und Breitengrad vorhanden sind. Jedoch zeigt Abbildung 10, dass diese alleine keine signifikante lineare Korrelation mit dem Quadratmeterpreis haben. Die lineare Korrelation ermöglicht es nicht zu bestimmen ob die Kombination der Längen- und Breitengrade entscheidend sind. Aus den Positionsdaten kann aber die Distanz zum Stadtzentrum errechnet werden.

```
In [10]: latitude_center = 48.210033
longitude_center = 16.363449

In [11]: index, row in data.iterrows():
data.loc[index, 'distance_to_center'] = haversine(row['longitude'], row['latitude'], longitude_center, latitude_center)

In [12]: corrmatrix = data.corr()
corrmatrix["price_per_m2"].sort_values(ascending=False)

Out[12]: price_per_m2      1.000000
building_floors    0.385326
plot_share         0.002082
latitude          -0.011045
plot_area         -0.028618
longitude          -0.038409
land_use_share     -0.050230
building_year      -0.051976
id                -0.133134
distance_to_center -0.268828
Name: price_per_m2, dtype: float64
```

Abbildung 12: Korrelationen nach Einfügen der Distanz zum Stadtzentrum

Die neue Spalte weist eine höhere Korrelation auf als die meisten anderen Spalten. Die Korrelation hat sich gegenüber der Längen- und Breitengrad um ein vielfaches erhöht. Sie könnte also sehr hilfreich bei der Bestimmung von Preisen sein.

Wie vorhin beschrieben wird die Spalte transaction_ym umgewandelt in eine Spalte die lediglich das Jahr angibt. Dadurch kann wieder eine Korrelation hergestellt werden.

```

In [13]: date = data.transaction_ym.apply(pd.to_datetime, format="%Y-%m")

data["year"] = date.dt.year

corrmat = data.corr()
corrmat["price_per_m2"].sort_values(ascending=False)

Out[13]: price_per_m2      1.000000
building_floors    0.385326
year              0.131843
plot_share        0.002082
latitude         -0.011045
plot_area        -0.028618
longitude        -0.038409
land_use_share    -0.050230
building_year     -0.051976
id               -0.133134
distance_to_center -0.268828
Name: price_per_m2, dtype: float64

```

Abbildung 13: Korrelation nach Einfügen des Transaktionsdatums als Zahl

Auch die Jahreszahl hat eine relativ hohe Korrelation mit dem Preis, sie fällt ähnlich aus wie die der id. Lediglich die Vorzeichen sind unterschiedlich. Nach näherer Überprüfung, liegt dies daran, dass die Einträge nach Datum organisiert sind und deshalb die Jahreszahlen mit steigender id fallen (Korrelation: year/id -0,997929).

Als letztes zusätzliches Feature wird werden plot_share und plot_area zusammengerechnet um eine relative_share zu errechnen. Also wieviele Quadratmeter tatsächlich gekauft wurden.

```

In [14]: relative_share = data.plot_area * data.plot_share
data["relative_share"] = relative_share

corrmat = data.corr()
corrmat["price_per_m2"].sort_values(ascending=False)

Out[14]: price_per_m2      1.000000
building_floors    0.385326
year              0.131843
plot_share        0.002082
latitude         -0.011045
relative_share    -0.026335
plot_area        -0.028618
longitude        -0.038409
land_use_share    -0.050230
building_year     -0.051976
id               -0.133134
distance_to_center -0.268828
Name: price_per_m2, dtype: float64

```

Abbildung 14: Korrelation nach Einfügen der tatsächlich gekauften Quadratmeter

Die Korrelation mit dem Preis fällt sehr gering aus. Scheinbar hat die Kauffläche einen geringen Einfluss auf den Preis.

7.2 Pipelining & Featureauswahl

Nach der Analyse und Erzeugung von Features, werden zwei Pipelines erstellt. Diese sollen eine Vergleichbarkeit der Modelle ermöglichen. Darum werden für beide Pipelines dieselben Beschneidungen am Datensatz vorgenommen. Die Pipelines unterscheiden sich darin, dass

eine die neu erzeugten Features verwendet und auf signifikantere Features setzt während die andere Pipeline mehr dafür auch insignifikantere Daten verwendet.

Zunächst werden grobe Ausreißer entfernt da diese die zu starken Verzerrungen führen wie in der Sektion „Unrepräsentative Daten“ beschrieben. Besonders auffällig in der Analyse waren die Quadratmeterpreise selber. Nach einer näheren Betrachtung des Datensatzes ist hervorgegangen, dass kaum Werte vorhanden sind welche einen Quadratmeterpreis von 2500,-€ überschreiten. Deshalb werden alle Einträge entfernt die zu stark abweichende Preise aufweisen.

Da die Distanz zum Stadtzentrum eine hohe Korrelation mit dem Preis aufgewiesen hat werden alle Einträge entfernt welche keine Positionsdaten haben. Weiters ist aufgefallen, dass einige Grundstücke scheinbar nicht in Wien liegen und über 50km vom Stadtzentrum entfernt liegen. Aufgrund dessen werden alle Einträge entfernt, welche über 15km vom Zentrum entfernt sind.

```
In [16]: data = data[data.distance_to_center < 15]
         data = data[data.price_per_m2 < 2500]
         data.shape

Out[16]: (30441, 21)
```

Abbildung 15: Filtern von Ausreißern

Mittels der *shape*-Eigenschaft kann ermittelt werden, dass von anfänglichen 44366 Einträgen noch 30441 übrig sind. Wobei die meisten aufgrund von fehlenden Positionsdaten wegfallen. Denn von Anfang an hatten nur 32465 Einträgen Längen- und Breitengrade.

Um nicht noch mehr Daten aus dem Datensatz entfernen zu müssen, werden bei den restlichen Einträgen fehlende Daten durch den Durchschnitt der jeweiligen Spalte ergänzt. Wie in „Fehler- bzw. Lückenhafte Daten“ erwähnt, werden die Daten ergänzt um sie für die Modelle vorzubereiten. Die gewählte Strategie, fehlende Daten durch einen Durchschnitt zu ersetzen ist schlicht die simpelste Methode.

Sobald die Daten in ein einheitliches Format gebracht wurden und keine Spalte leer ist, werden die Features des Dataframes in einer für Modelle verarbeitbare Form gebracht. Dazu werden die numerischen Features mittels eines *Standardscalers* um 0 zentriert. Das geschieht, da viele Machine Learning Modelle ohne das Skalieren der Features ein schlechtes Verhalten aufweisen, wenn die Features nicht normalverteilt sind [7].


```
In [27]: from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler

         num_pipeline = Pipeline([
             ('std_scaler', StandardScaler())
         ])
```

Abbildung 16: Pipeline für numerische Features

Wie im Kapitel „Feature“ beschrieben, werden die kategorischen Features mittels eines *OneHot-Encoders* in Zahlenwerte umgewandelt,.

```
In [28]: from sklearn.preprocessing import OneHotEncoder

         cat_pipeline = Pipeline([
             ('one_hot', OneHotEncoder())
         ])
```

Abbildung 17: Pipeline für kategorische Features

Sind die beiden Pipelines definiert können in einer übergeordneten Pipeline zusammengeführt werden. Dadurch ist es möglich jegliche Auswahl an Features in eine Matrix zu verwandeln, welche von den Lernalgorithmen und Modellen genutzt werden kann.

```
In [29]: from sklearn.compose import ColumnTransformer

         num_attribs = list(data_num)
         cat_attribs = list(data_cat)

         full_pipeline = ColumnTransformer([
             ("num", num_pipeline, num_attribs),
             ("cat", cat_pipeline, cat_attribs)
         ])
```

Abbildung 18: Kombinierte Pipeline

Ist die übergeordnete Struktur der Pipeline spezifiziert, können die beiden Pipelines definiert werden. Erstere wird nur auf numerische Features setzen, während die zweite Pipeline nurkategorische Features verwenden wird. Im folgenden wird aufgelistet welche Features für die jeweiligen Pipelines verwendet werden:

Pipeline A Numerisch	Pipeline B Kategorisch
building_floors	property_type
distance_to_center	district
relative_share	contract_type
year	protection_zone
	building_ban
	land_use

Table 3: Features für beide Pipelines

Nachdem beide Pipelines erzeugt wurden können Daten durch einen einfachen *transform*-Befehl in das entsprechende Format gebracht werden.

7.3 Training & Evaluierung der Modelle

Mit den fertig formatierten Daten können nun die Modelle trainiert und mittels Cross-Validation evaluiert werden. Wie im Kapitel Evaluierung eines Modells erwähnt wird mit 10 Folds evaluiert, da es allgemein zu genauen Ergebnissen führt. Verglichen werden die Modelle anhand des Root Mean Squared Errors aus welchem ein Durchschnitt über die 10 Validierungsläufe errechnet wird sowie eine Standardabweichung. Trainiert werden die Modelle mit den Daten von 1987-2010.

```
In [22]: #train and cv linear regression for data_prep_a
from sklearn.linear_model import LinearRegression

lin_reg_a = LinearRegression()
lin_reg_a.fit(data_train_prep_a, data_train_labels_a)

scores = cross_val_score(lin_reg_a, data_train_prep_a, data_train_labels_a, scoring="neg_mean_squared_error", cv=10)
lin_reg_rmse_scores_a = np.sqrt(-scores)

display_scores(lin_reg_rmse_scores_a)

Scores: [428.07599805 383.70809955 368.58897131 323.70836266 342.89403222
 344.57305531 359.15661085 371.24994065 308.75595641 336.02724009]
Mean: 356.67382670940106
Standard Deviation 32.14833902938146
```

Abbildung 19: Training und Validierung eines Linearen Modells

Dieser Vorgang wird für alle Modelle jeweils einmal pro Pipeline durchgeführt. Die Ergebnisse sind in der folgenden Tabelle:

Modell	Pipeline A (numerisch)		Pipeline B (kategorisch)	
	Durchschnitt	Standard-abweichung	Durchschnitt	Standard-abweichung
Linear	379,721 €	39,800 €	353,573 € - 26,148	38,766 € - 1,034
Ridge	379.720 €	39.800 €	353,540 € - 26,18	38.734 € - 1,066
Lasso	379,689 €	39,757 €	355,041 € - 24,648	38,456 € - 1,301
Decision Tree	469,644 €	38,340 €	346,750 € - 122,894	29,566 € - 8,744
Random Forrest	365,416 €	30,210 €	341,031 € - 24,385	28,887 € - 1,323

Table 4: Ergebnisse der 10-fold Cross Validation

Was hier auffällt ist, dass alle Modelle präzisere Ergebnisse erzielen, wenn sie mit den kategorischen Daten trainiert werden. Das könnte daran liegen, dass die kathegorische Pipeline schlicht mehr Daten verwendet aufgrund der größeren Anzahl an Features.

Auffällig ist hierbei, dass die linearen Modelle alle eine ähnliche Präzision erreichen, das spricht dafür, dass sich die Regularisierung bei der gegebenen Menge an Daten kaum auswirken auf das Valdierungsergebnis.

Das schlechteste Verhalten weist hierbei der Decision Tree auf, der mit numerischen Daten trainiert wurden. Auch hier liegt es nahe, dass die geringe Anzahl an Features eine negative Auswirkung auf die Präzision hat gegenüber dem Modell, welches mit kategorischen Daten trainiert wurde.

7.4 Test der Modelle

Abschließend werden alle trainierten Modelle noch mit einem Testdatensatz getestet. Der Testdatensatz enthält die restlichen Einträge im Datensatz von 2011-2013. Dazu wird Testdatensatz ebenfalls mit den beiden Pipelines formatiert und für die jeweiligen Modelle zum schätzen vorbereitet. Anschließend werden von allen Modellen Schätzungen vorgenommen.

```

In [32]: #prepare test_data_a for testing
test_data_a = data_test[["price_per_m2", "building_floors", "year", "distance_to_center", "land_use", "relative_share"]

test_labels_a = test_data_a["price_per_m2"].copy()
test_set_a = test_data_a.drop("price_per_m2", axis=1)

test_data_prep_a = full_pipeline_a.transform(test_set_a)

#prepare test_data_b for testing
test_data_b = data_test[["price_per_m2", "property_type", "district", "contract_type", "protection_zone", "building_ban",
                           "distance_to_center", "land_use", "relative_share"]]

test_labels_b = test_data_b["price_per_m2"].copy()
test_set_b = test_data_b.drop("price_per_m2", axis=1)

test_data_prep_b = full_pipeline_b.transform(test_set_b)

```

Abbildung 20: Vorbereitung des Testsets

```

In [33]: #prepare predictions on test set

#predictions from A-Models
lin_reg_pred_a = lin_reg_a.predict(test_data_prep_a)
ridge_reg_pred_a = ridge_reg_a.predict(test_data_prep_a)
lasso_reg_pred_a = lasso_reg_a.predict(test_data_prep_a)
tree_reg_pred_a = tree_reg_a.predict(test_data_prep_a)
forest_reg_pred_a = forest_reg_a.predict(test_data_prep_a)

#predictions from B-Models
lin_reg_pred_b = lin_reg_b.predict(test_data_prep_b)
ridge_reg_pred_b = ridge_reg_b.predict(test_data_prep_b)
lasso_reg_pred_b = lasso_reg_b.predict(test_data_prep_b)
tree_reg_pred_b = tree_reg_b.predict(test_data_prep_b)
forest_reg_pred_b = forest_reg_b.predict(test_data_prep_b)

```

Abbildung 21: Schätzungen werden von allen Modellen vorgenommen

Abschließend wird für alle Schätzungen der Root Mean Squared Error und der Mean Absolute Error erhoben.

Modell / Score	Pipeline A	Pipeline B	Differenz
Linear/RMSE	2627,123	2636,074	+ 8,951
Linear/MAE	968,479	929,828	- 38,651
Ridge/RMSE	2627,128	2636,538	+ 9,41
Ridge/MAE	968,480	929,737	- 38,743
Lasso/RMSE	2627,892	2661,903	+ 34,011
Lasso/MAE	968,778	936,862	- 31,916
Tree/RMSE	2577,915	2588,385	+ 10,47
Tree/MAE	916,636	868,148	- 48,488
Forest/RMSE	2537,880	2598,129	+ 60,249
Forest/MAE	855,338	869,397	+ 14,009

Table 5: Testergebnisse

Gegenüber den Validierungsergebnissen zeigt sich hier, dass die Modelle von ihrer Präzision nicht weit auseinander liegen. Je nach genommener Metrik liegt jeweils ein anderes Modell vorne. Lediglich beim Random Forest liegt das von Pipeline A trainerte Modell eindeutig vorne.

7.5 Analyse

Die Ergebnisse zeigen, dass sich die *linearen Modelle* kaum unterscheiden und je nach Metrik die Modelle von *Pipeline A* respektive *Pipeline B* sich besser eignen. Die Diskrepanzen zwischen den Validierungsergebnissen und den endgültigen Testergebnissen deuten auf eine Überanpassung von allen Modellen an den Trainingsdatensatz hin.

Ein weiterer Testlauf, in welchem die Testdaten im gleichen Maße wie die Trainingsdaten beschnitten wurden, hat ähnliche Ergebnisse geliefert, wie die Validierungsergebnisse. Demnach sehen die Validierungsdaten deshalb so vielversprechend, da sowohl Trainings- als auch Validierungsfolds Quadratmeterpreise in ähnlichen Dimensionen beinhalten.

Interessant ist auch, dass das Decision Tree Modell von *Pipeline A*, das im Validierungslauf am schlechtesten abschneidet, im abschließenden Testlauf das viertbeste Ergebnis erzielt. Dieser kann mit den numerischen Daten scheinbar besser verallgemeinern als die linearen Modelle.

Der *Random Forest* schneidet mit einem signifikanten Vorsprung zu den anderen Modellen ab. Allerdings nur, wenn das Modell mit *Pipeline A* trainiert wurde. Jedoch sind mit der Präzision des *Random Forest* Modells die größten Berechnungszeiten verbunden. Der *Random Forest* ist in diesem Vergleich das einzige Modell, das zu den Ensemble Methoden gehört und damit mehrere Modelle verbindet.

Mit kategorischen Daten schneidet der *Decision Tree* allerdings besser ab als ein *Random Forest*. Dies könnte daran liegen, dass die einzelnen *Decision Trees* jeweils zu wenig informative Features enthalten haben und deswegen zusammen eine insgesamt schlechtere Schätzung abgeben.

8 Aussicht

Wie im Kapitel „Analyse“ beschrieben, bleiben noch einige Untersuchungen offen, welche den Rahmen dieser Arbeit sprengen. In weiteren Experimenten könnte man versuchen die kategorischen Daten zu reduzieren und mithilfe von weiterer Analyse herausfinden, welche Kategorien wichtig für den Entscheidungsprozess eines Modells sind. Zum Beispiel, ob die Überprüfung, ob ein Grundstück eine gewisse Widmung hat, überhaupt eine Relevanz besitzt oder die Komplexität der Daten unnötig erhöht.

Weiters wäre es interessant, den Datensatz um einige Features zu erweitern und zu überprüfen, ob diese eine positive Auswirkung auf die Testergebnisse haben. Zum Beispiel, wie hoch das Durchschnittseinkommen eines Bezirkes ist oder ob die Nähe zu öffentlichen Verkehrsmitteln und Arbeitsstellen gegeben ist. Die Möglichkeiten dieses Experiment zu erweitern sind mannigfaltig.

Abschließend bleibt zu sagen, dass sich die praktische Anwendung des Machine Learnings noch in einem frühen Stadium befindet und sich noch stark verändern kann. So bieten Wiener Universitäten erst seit kurzem eigene Studiengänge zu den Gebieten Machine Learning und Data Science an. Es ist jedoch ein vielversprechendes Zeichen, das darauf hindeutet, dass das Machine Learning sowohl in der Wissenschaft als auch in der Wirtschaft vermehrt an Bedeutung gewinnt.

Literaturverzeichnis und Internetressourcen

- [1] A. Géron, „Hands-On Machine Learning with Scikit-Learn and TensorFlow“, 1. Aufl., O'Reilly Media, 2017
- [2] A. Zheng, A. Casari, „Feature Engineering for Machine Learning“, 1. Aufl., O'Reilly Media, 2018
- [3] A. C. Müller, S. Guido, „Introduction to Machine Learning with Python“, 3. Aufl., O'Reilly Media, 2017
- [4] M. Banko, E. Brill, „Scaling to Very Very Large Corpora for Natural Language Disambiguation“, [online], Verfügbar unter: <http://www.aclweb.org/anthology/P01-1005> [Zugang am 30.10.2018]
- [5] S. Fortmann-Roe, „Understanding the Bias-Variance Tradeoff“, [online], Verfügbar unter: <http://scott.fortmann-roe.com/docs/BiasVariance.html> [Zugang am 30.10.2018]
- [6] I. H. Witten, E. Frank, M. A. Hall, C. J. Pall, „Data Mining: Practical Machine Learning Tools and Techniques“, 4. Aufl., 2016
- [7] „Scikit-learn; Machine Learning in Python“, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
- [8] Datenquelle: Stadt Wien - <https://www.data.gv.at/katalog/dataset/5fc523d5-c299-4d97-889f-01ed247b10fa>, „Kaufpreissammlung Liegenschaften Wien“ [online], [Zugang 30.5.2018], editiert von Dr. Klaudius Kalcher und Moritz Feigl

Abbildungsverzeichnis

Abbildung 1: Trainingsdatensatz für Supervised Learning am Beispiel von Spammail-Klassifizierung ([1] S 8).....	10
Abbildung 2: Gegenüberstellung von einem überangepassten Modell (links) und einem gut verallgemeinernden Modell (rechts).....	14
Abbildung 3: Unterangepasstes Lineares Modell.....	15
Abbildung 4: Auswirkung von Bias und Varianz ([5], S 1).....	16
Abbildung 5: Beispiel für eine „5 fold cross validation“ ([3], S 254).....	18
Abbildung 6: Vorhersagen eines Linearen Modells im zweidimensionalen Raum.....	22
Abbildung 7: Simpler Decision Tree für eine Tierklassifizierung ([3], S 73).....	25
Abbildung 8: Decision Boundary eines Decision Trees mit der Tiefe 2 ([3], S 75).....	25
Abbildung 9: Überblick über numerische Spalten des Datensatzes.....	27
Abbildung 10: Korrelation numerischer Spalten mit dem Quadratmeterpreis.....	28
Abbildung 11: Überblick über kategorische Spalten des Datensatzes.....	28
Abbildung 12: Korrelationen nach Einfügen der Distanz zum Stadtzentrum.....	29
Abbildung 13: Korrelation nach Einfügen des Transaktionsdatums als Zahl.....	30
Abbildung 14: Korrelation nach Einfügen der tatsächlich gekauften Quadratmeter.....	30
Abbildung 15: Filtern von Ausreißern.....	31
Abbildung 16: Pipeline für numerische Features.....	32
Abbildung 17: Pipeline für kategorische Features.....	32
Abbildung 18: Kombinierte Pipeline.....	32
Abbildung 19: Training und Validierung eines Linearen Modells.....	33
Abbildung 20: Vorbereitung des Testsets.....	35
Abbildung 21: Schätzungen werden von allen Modellen vorgenommen.....	35

Tabellenverzeichnis

Tabelle 1: One Hot Encoding am Beispiel von Automarken.....	12
Tabelle 2: Übersicht über den verwendeten Datensatz.....	21
Table 3: Features für beide Pipelines.....	33
Table 4: Ergebnisse der 10-fold Cross Validation.....	34
Table 5: Testergebnisse.....	35

Abkürzungsverzeichnis

RMSE	Root Mean Squared Error
MAE	Mean Absolute Error